ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 3(32)

SYNERGETIC EFFECT COMBINING UML AND PETRI NETS IN DESIGNING MODERN INFORMATION SYSTEMS

Irmantas KATILIUS, Ignas SKUČAS, Vida MELNINKAITĖ

Vytautas Magnus University
Vileikos 8, LT-3035 Kaunas, Lithuania

Abstract. This paper reviews the most archetypal features of UML and Petri nets. Formed modeling structural scheme reflects the trends of designing modern information systems. This paper also overviews the possibility of converting UML net into Petri net. At the end, the paper gives a conclusion, and also a prospective of further research – using methods presented here.

Development of organizational information systems (IS) recently becomes an important subject. Lately, the Unified Modeling Language (UML) became the standard in creating software, and in designing relatively simple organizational information systems. Some UML diagrams are also used for organizational sectional modeling, reengineering of business processes, and process analysis. By current view UML is used for specification of task sequences, and in the process of creating software for generation of program code class diagrams are being used. Class diagrams are sourced using activity diagrams, which are based on empirical reasoning and are not described by mathematical methods.

UML methodology only allows very limited modeling of simple systems, to say nothing of the possibilities to evaluate process dynamics. That's why Petri nets are often tried to use for designing of systems. High development of IS is determined by formal mathematical methods. Petri nets are formalism for performing process characterized by parallel links analysis. Petri nets characterize parallel structures, flexible modeling, and highly developed mathematical formalism. However, the results show that it is still failed to implement universal modeling of dynamic processes. Using methodology of Petri nets and its extensions (ex. Object Petri nets) is not absolutely rational because additional efforts are needed for evidence of objective features, besides by modeling systems using objective methodologies, particularly UML, cumulative experience is being lost.

It limits system development possibilities, especially in a case of advanced systems, where deeper analysis of activity alternatives is necessary, demanding to evaluate dependences on variable business conditions under the influence of external factors, also continuous modernized data mining systems influencing system's flexibility and quality requirements, and that is the basis of dynamic IS creation. Therefore this paper proposes comprehensive information system creation model, which combines internals of both, net and object designing methodologies. This model also applies to simplified cases when information system (IS) development stages may be executed by using only UML features, while in a general system creation case hybrid modeling scheme is recommended. For that purpose methods of transformation from UML to Petri net and modeling possibilities' expansion ways, which allow better implementation of work sequence management, are being improved in the process. The conclusion state that implemented IS using suggested modeling scheme characterizes flexibility, and enables appending new features to the system without blocking its exploitation. Concerning work, particularly mark management methods enable aiming that implementing IS would have modeling structure, possibilities to change system's functions and features by supplementing system with separate program modules without changing whole system.

Keywords: Organization proceeding modeling, designing, dynamic processes, UML and Petri nets.

1. Indroduction

Modern business world is changing rapidly; therefore implementing information systems projects face necessity of new methods and methodologies creation. Historically it developed so that software was being created not for modeling of business processes but aiming automation of some separate procedures proceeding in organizations.

The purpose of modeling is creating and describing models of organizations static and dynamic processes. Organizations may develop their activity by several scenarios. Often those scenarios are changeable in time, have seasonal characteristics, and are influenced by casual factors. Software often is "behind" with changing business requirements as mostly is designed to perform only limited set of business functions or can act by generally known,

often outdated business behavior scenarios. Research shows that most of the expenses related to software care (and modernizing) emerge because of inappropriate or insufficient organizational needs in modeling system creation stage, or due to information system's inadequacy to real running processes in organizations. These features are still insufficiently emphasized in creation of modern information systems seeking efficient management of business processes. Although semantic and object oriented models are fairly understandable for consumers, they don't have dynamic aspects modeling possibilities. Information systems designing must link organizational goals with processes existing in organization [2]. Goals can be described as particular forms of running processes. Often particular ideal situations are designed in designing process, and it is forgotten that business processes are changeable. In designing process problematic situations come up that are tried avoided by various system restrictions. However in real life those situations are not avoided. Consequently we get inflexible information system that does not meet modern requirements or can't function at all.

Structure and behavior of modern information systems often become very complicated. Their designing uses multi-layer architecture, detailed class structure. Information systems must characterize parallel features, and ability of reacting to external factors. Software created must meet system flexibility and configuration requirements. Principle of object modeling enables to systemically abstract and manage features of such system. Modeling tools supporting the principle allow transforming those features into appropriate object structure and behavior models.

Our goal is to suggest the ways of information system modeling, which would allow creating information system characterizing flexibility, i.e. adjustment to new environment conditions and changeable business needs, and achieve that without newly redesigning the system.

Work structure chosen: firstly, short UML review, UML application in designing, and weaknesses and strengths are proposed. Further, Petri nets and their application in designing, object Petri nets, weaknesses and strengths are described in the paper. Other sections present methods of converting from UML to Petri net, also how using combination of UML and Petri net makes new designing model. 3rd section presents modeling possibilities extension using mathematical methods, also states issues that are faced in information systems designing (lack of quality characteristics, etc), and what is suggested for solving the problem. Fourth section proposes conclusions and indicates directions of further research.

2. Strengths and weaknesses of UML and Petri nets in designing

2.1. UML models and their structures

UML (Unified Modeling Language) was acknowledged actual standard of object modeling area. With help of UML it is possible to present systems requirements, structure and behavior in diagram form, without influence of specifics of system's implementation tools. UML is a language of visual specification, using diagrams of various purposes for most specifications. Diagrams highly relieve work when describing complicated connections that often have simple consistent structure and are hardly written in simple text. UML models connect particular features of organization's activity models and structures them, preset decisions are quite highly developed, introduced in [7].

For business systems modeling UML is used more and more often. UML modeling language characterizes developed, universally approved object features. UML modeling methodology offers several diagrams for representation of various businesses (or other systems) models [3]. This allows designing information systems conclude business processes models.

When modeling relatively simple systems offered modeling model may be as shown in Figure 1.

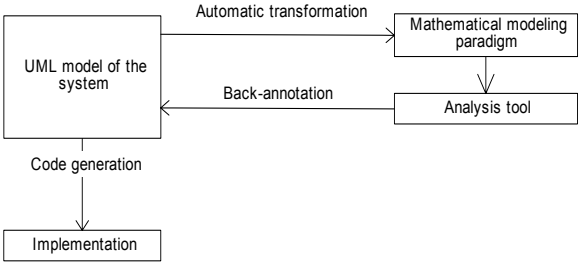

Figure 1. Abstract scheme of system designing

The aim of UML is to help visualize, specify and document artifacts of information systems. We believe that reader is familiar with UML diagrams and their features. You can find more about UML in [7]. In this paper we will only mention several key UML aspects. UML may be separated into three parts:

Modeling items. These elements can be categorized into four key parts: Structural, Behavior, Grouping, Other.

Relationship items. These items can also be separated into four main parts: Dependence, Association, Generalization, and Realization.

Diagrams. With help of diagrams it is possible to graphically picture desirable system. Each diagram represents static or dynamic part of system. Each part (static or dynamic) solves different systems description problems [7].

Although UML methodology is highly developed, possibility of writing UML models in analytical form is missed. Additionally UML methodology uses a lot of graphic items that in one case make models clearly understandable (readable), but if used too many items models written with UML become overloaded and inappropriate for efficient, particularly dynamic processes modeling. Also UML methodology is inappropriate for modeling of real business processes, which characterize parallel relations [1].

Often becoming parallel processes performed in organizations, different roles and authority of actors taking part in processes, changes of resource state are very important when trying to understand systems behavior and they must be written visually. UML notation does not outline strict rules by which action-processed entities could be differed from the entities that perform that action, and that is often useful when modeling simple modern systems. UML models designed for behavior modeling are not developed as well as models modeling static system structure.

According to structural schemes it is possible to describe pending process in different ways by using same UML diagrams. Activity diagram reflects dynamics of the system and pictures succession of process action performance. Activity diagram is often used for picturing work sequences. A business process activity diagram shows document movement in business process and illustrates answers into questions: What is happening in work sequence?; What parallel actions may be performed?; What are possible alternative ways of work sequence performance?.

Activity diagram also describes business roles and responsibility areas – in other words, which performs this or that action.

Role and responsibility areas in activity diagram are divided into columns. Those columns show which business unit (employee, system, etc) and where acts in work sequence implementation. In [7] it is many examples of activity diagrams, which image the same business process. These examples show how with help of simple transformation activity diagram is restructured.

By structural view only one – entity-relation – model, which differs only by graphical marking in various methodologies, was applied for system modeling in object modeling. Meanwhile for modeling of dynamic system part several essentially different models are applied. For collaboration modeling, which is oriented into roles and their interactions, and is needed for task performing, collaboration and sequence diagrams are dedicated. Sequence diagram perfectly fits when picturing strictly consistent processing of tasks, however should parallelism or any other indeterminacy occurs, problems are faced. It becomes complicated to picture optimal succession of function performance. When picturing parallel actions using these diagrams, excess task specifications should be used, and separate compulsory consistent performance sequence should be assigned to performance of each task. Collaboration diagram is also used for description of UML behavior. But they do not describe behavior visually. Collaboration diagram is made of class diagram by giving each operation a performance sequence number. Complex numeration and plenty of explanatory text makes this diagram unattractive and inappropriate for behavior modeling [6].

Remaining diagrams for dynamics picturing (activity and state chart) show system‘s reaction to factors. Activity diagram allows describing parallel or indefinite behavior by using fictional states, which help picturing parallelism. When fictional states are used diagram becomes less understandable, additionally it is not clear how many fictional states should be used because often only during systems exploitation definite parameters turn out, which help to fully define system’s parallelism. When using this notation of object modeling, definite semantics and rules uniting diagrams used for different purposes are often lacked, therefore UML does not propose integral behavior model. UML language strongly lacks strictly described integration between dynamic models and structural items, such like subsystems, classes, interfaces and relationships. Through in-capsule features object oriented view fairly applies operation with distributional systems. From the other side, many object-oriented languages are supposed to have tools for processing parallelism between objects, but very rarely object-oriented language has a possibility to operate with parallelism inside the objects. Without these possibilities object is just a passive component, waiting for queries and performing his methods when got a query.

2.2. Petri nets

For modeling systems characterized with parallel relations, there is oriented Petri nets methodology. Items used in Petri nets (unlike UML) have no strict appliance descriptions and gives system modelers more freedom to model dynamic features. There are P elements (P items) – states, (places) and T elements – transitions. Passive entities of real world are pictured as P elements (conditions, places, resources, waiting pools, channels, etc). Active entities of real world are pictured as T elements (events, transitions, action, executions of statements, transmission of messages etc) [5].

Meta-model of Petri net is pictured in Figure 2. As you can see in Figure 3 Petri net does not characterize variety of elements like UML methodology. Therefore, when modeling systems with Petri net it is very important for every modeler to give a modeling object proper meaning in Petri net [9] [10].

When modeling modern complicated systems classic Petri nets potentials are not enough, therefore various extensions of Petri net are suggested. Currently the mostly used Petri nets extensions are: chromatic

extension, time extension, hierarchic extension (for modeling of large systems).

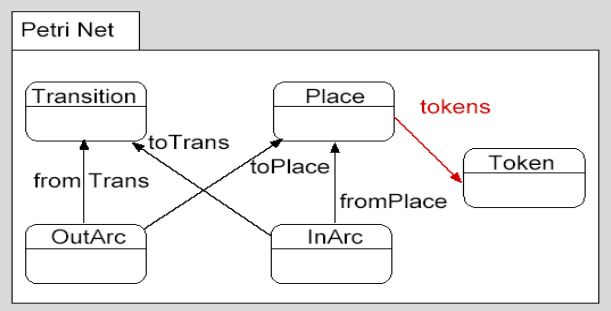

Figure 2. Meta model of Petri nets

Action dynamics are usually described by states transmissions. Most diagrams used for specification of semantic processes dynamics analyses states transmissions or communicational relations of actors. For understanding behavior of particular process, specification of existing and desirable process states is needed. States change when particular events occur – each event can change object state and conversely, for changing a state an event is needed.

However, Petri nets do not have highly developed object-modeling features. We will analyze two ways how to link Petri nets and object-oriented view. Analyzed ways contain all of useful pragmatic and theoretic features of both formalisms.

First way – Object Petri nets (PNO). Object Petri nets are appropriate for functional models, for they analyze system as a unit, without analyzing blocks that are responsible for performing operations.

Second way – Cooperative objects (COO). This formalism is intended for structural models, for they analyze system as set of agents that perform part of work and communicate with each other [11]. Each object is an exemplar of object class, and is identified with unique name. Between objects classes there may be relations of inheritance. Each cooperative object has a list of features and operations. Those operations are functions, which have a list of incoming parameters and a list of outgoing parameters. Operation can reach and modify values of attributes but cannot reach other objects or to create them. Each object class has its creation function for initialization of each objects value. Object performs operations only on demand; therefore objects are passive data storing entities. So their use in particular cases may be limited.

Despite all strengths of Petri nets they do not reflect data processing in system. Most operations that call a change of information system state also process part of data, for information system itself is real life's reflection in computer, therefore it is essential to have a possibility of approaching Petri nets marks as data structures. Modulation is an important feature of software designing, though when using Petri nets it is complicated wanting to structure system model and propose it as a whole of cooperating components. Therefore it is essential to enter new elements, which would give connection for each element of Petri net and describe how those elements communicate with each other through their connections. Here object oriented view may be applied again, for it proposes tested and proved concepts, after which Petri net may be approached as active object. The key strength of the Petri net is its simplicity (even knowing that parallel systems inherently are less understandable), wide facility of its application regarding their abstract nature, and their suitability for formal analysis of behavior [8].

Further the paper analyses how using object features of UML can extend Petri nets possibilities, and offers ways of converting UML models into Petri net.

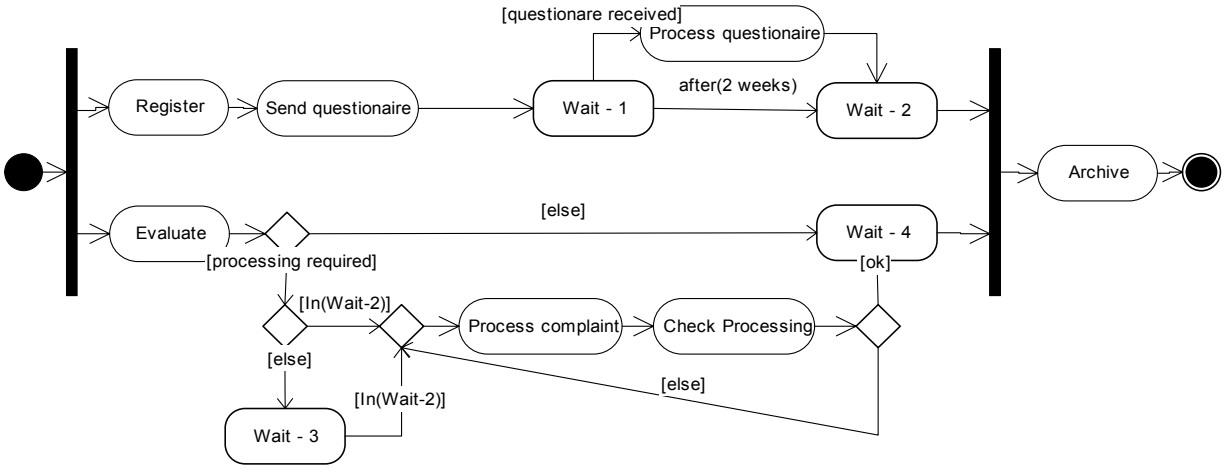

Figure 3. Activity diagram "Processing Complaints"

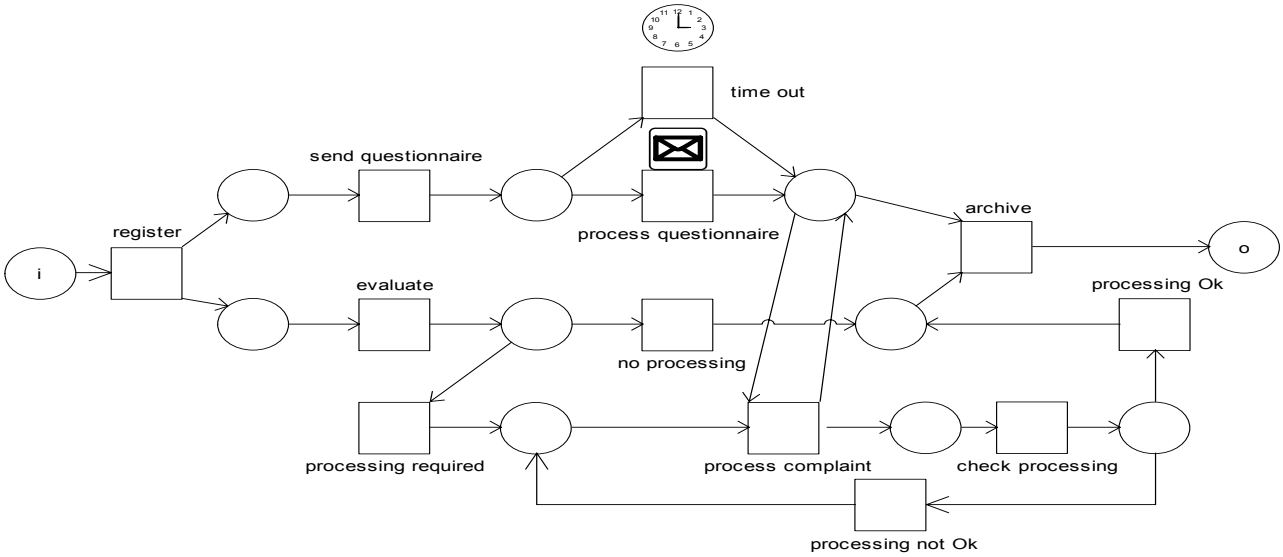

Figure 4. Petri Net "Processing Complaints"

2.3. Converting UML into Petri net

This section shows how UML diagrams can be converted into Petri net, and how such converting may be useful strengthening the possibilities of information systems modeling. Figure 3 pictures UML notation activity diagram "Processing Complaints", while Figure 4 shows Petri net corresponding to above-mentioned UML activity diagram.

Both diagrams in Figure 5 and in Figure 6 show same business process: "Processing complaint". UML activity diagrams are designed for picturing a cycle of objects existence. The diagram shown in figure 5 pictures life circle of the object "Complaint". States of UML diagrams are converted into Petri nets places, and UML diagrams transmissions are modeled to Petri nets transmissions. Relations between states and transmissions in Petri net are modeled with arcs. Marks of Petri net are in character with UML notation object, in this case – "Complaint". UML activity diagrams show object's existence and change of its features. While marks of Petri net just go from one Petri net place to other without picturing change of mark's features. Modeler himself has to understand that object modeled with Petri net assumes new features when moving particular transmission of the net, or when coming to a certain place. Thus in Petri net got from UML many additional transmissions occur; therefore net itself becomes overloaded with elements and becomes uninformative. This defect can be avoided by using Petri nets extensions, for example Chromatic Petri nets. Then particular features of modeled object may be modeled with colors.

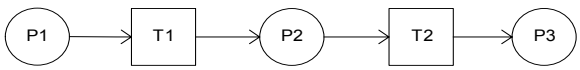

Figure 5. change of selling orders states pictured with Petri net

UML is object modeling language, core of which is picturing "object life circle". Also UML characterizes highly developed quantity of graphical elements, use of which is not completely regulated and leaves system analyst (creator) large freedom of actions.

When modeling business processes with Petri net, it is very important to give right business objects notions to elements of Petri net. With Petri nets marks often business objects are modeled, for example: goods, people, documents and change of their states during business process. Since UML is object modeling language, a question comes out how to picture signs and features of those objects. If a selling offer is modeled with a mark in Petri net, such signs as buyer's name, number of order, date of delivery should be pictured. Such signs are hardly pictured with marks in classic Petri net; therefore classic Petri net is extended with chromatic marks. In a Chromatic Petri net each mark has a value denoted as "color". Transmissions define values of created marks on the grounds of values of used marks, for example transmissions describes relation between values of "incoming marks" and "outgoing marks". Such use of Petri nets mark when one mark is one object of UML language is mostly met in literature, but then we do not use all facility of Petri net. Thus Petri nets transmission is "opened" once needed quantity of marks collect in particular state. In this case we get that from Petri nets state to nets transmissions and backwards go just one arc. In this way it becomes simple to write a business process pictured in figure 5. Let's analyze a case often met in business world – processing of selling order. With Petri nets state P1, P2, P3 selling order's state are modeled – received selling order, confirmed selling order, accomplished selling order. Nets transmissions mark a change of selling orders states.

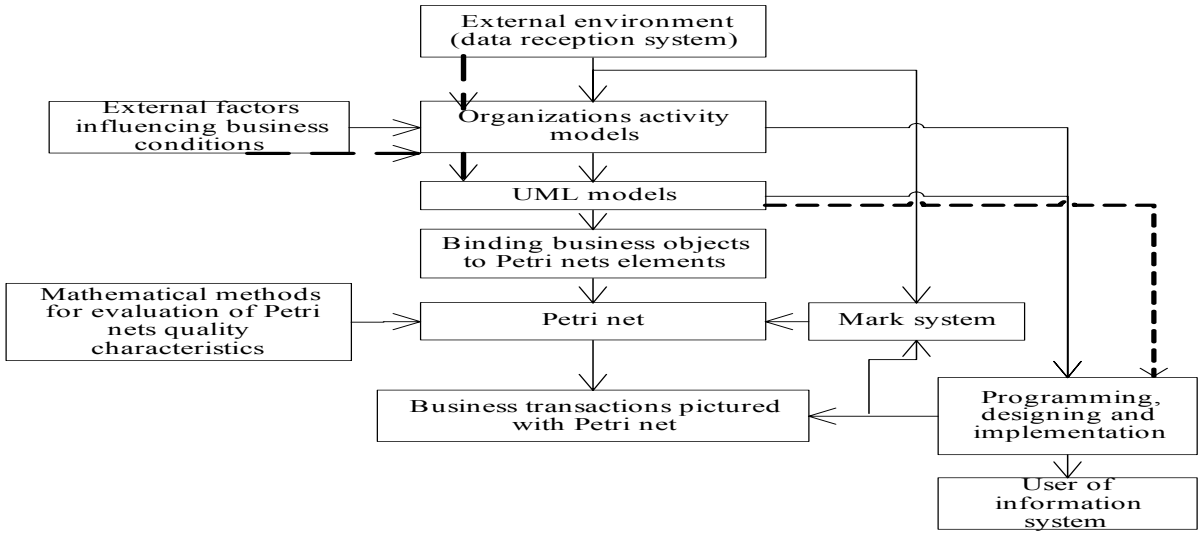

Figure 6. Hybrid model of information systems development

When such way of modeling is chosen, it comes out that any received order, when get into state P1, may be confirmed and accomplished in further stage. While in business processes limits exist, for example: order delivery date, order load weight or customer's working time. In each stage an order gets additional features or loses existing ones. Whole set of orders features may be modeled with Petri nets states, but in that case the net becomes hardly understandable for a user. Other way of modeling could be as order states (received order, confirmed order, accomplished order) are just particular features, not necessarily modeled with Petri nets states. This small example shows what issues may be faced if wrongly interpreted Petri nets elements and business notions. In this way models created using both, UML and Petri nets, enable further development of variety of structures, which allows modeling dynamic processes and by those structures creating program modules that implement information system.

2.4. Organizations information systems creation model characterizing synergetic features

Organizations activity models are important when highlighting essential processes of organizations activity and their development. For modeling organizations activity IDEF methods are used. We think that reader is familiar with above-mentioned methods of organizations activity modeling; therefore this paper mentions only several essential moments.

IDEF0 – *Activity Modeling*. IDEF0 is used for identification of business documents and other business objects, and for determining of relations between them.

IDEF3 – *Process Modeling*. IDEF3 is used for picturing processes dynamics, events influencing process and process change logic.

Ways of information systems creation are shown in Figure 6. Two ways of creating information system are distinguished there: to use only UML (for creating simpler information systems) and hybrid way, which comprises UML and Petri nets (for creating complicated information systems). As may be seen from above-mentioned figure, when using only UML modeling methodology (dotted line, Figure 8), in information systems creation after external environment and factors influencing business conditions, organizations activity models are received. After proceeding expert analysis activity models are described with UML diagrams and after UML diagrams program code are generated, and information system is created. However, in creation of complicated information systems, especially when deeper analysis of alternatives is needed, only UML diagrams are not enough, therefore use of hybrid creation of information system (black line, Figure 6) combining UML and Petri nets is suggested. As can be seen from a figure, it is suggested to convert UML diagrams received from organizations activity models into Petri nets. Only after performing business process analysis using facility of Petri nets methodology and after performing processes quality characteristics research by mathematical methods, program code may be generated and information system developed.

Use of such structure without above-mentioned purposes must enable creation of information systems models, which makes an opening for designed system in various stages of its designing to implement decision-making considering user's actions, changing business processes and process internals.

Also after transactions pictured with Petri net information system is designed, which may be implemented as modular software that gives system a possibility to be reorganized if business conditions change.

Attention should be paid that designing methodology offered by information system raises new tasks for user, because during systems exploitation, managing system, often taking nonstandard decisions; deeper knowledge and higher qualification are needed.

Facing complicated behavior of modern business systems or systems characterizing parallel features, tools offered by UML become inappropriate.

Sequence diagrams describe sequences of external or internal actions. The base of sequence diagrams – is to picture life line of an object. It is difficult to picture parallelism or possible indeterminations; therefore sequence diagram itself reminds more only a reflection of particular case, but not formal specification of behavior. *Behavior diagram* is just a textual expression of sequence diagrams information, which is got replenishing class diagram and giving numbers showing implementations succession to operations. In this diagram management and data flows are not separated visually. *State diagrams* enable to specify internal or external elements of the system. However, even these diagrams fail to fully write systems parallelism. Though this weakness was tried eliminated by installing a notion of additional states, still there is a lack of clear orders how to make diagrams specification by using those additional states. *Activity diagrams* that are got from state diagrams write specific behavior for particular task. However, UML does not show the relation between state and activity diagrams, elements in different diagrams are related only informally – by the same textual marking.

All these problems may be avoided by using above-mentioned hybrid methods for modeling business processes behavior. With Petri net even most complicated parallel systems may be written, sustaining essential object requirements. Formal nets semantics is able to write even complex object systems, and descriptions of behavior remain independent from systems implementation tools.

3. Extension of modeling abilities by using mathematical methods

When implementing Petri net with UML features, it is important having possibility to perform evaluation of quality characteristics. When implementing information system with above-mentioned features, Petri net reflecting business transactions may be created, which would be the basis of software creation.

When writing Petri nets got from using mathematical logic code, with analytical expressions (actions and states) and creating their executable rules, different activity scenarios of organization may be got. Also before starting to create organizations information system, parameters are needed, which would help evaluating not only working of information system, but also activity of organization itself, and fitting its facility to changing environment conditions. In general case every organization may be analyzed as a system of massive service theory. In every organization (industrial or of any other activity) service of any requests is taking part (production of products, facility service, trade operations), and actions are taken by employees or by machines. Naturally each stage of activity has its limitary potential such as average request serving time, consistency time between two requests, average time of waiting in line, and other characteristics well-known from massive service theory. We will mention only several of them, mostly known.

Let's assume that we have a mass servicing system containing n servicing channels. The system services the flow of orders with the intensity λ and each channel services μ orders per hour (order servicing intensityμ). At most m orders may be pending. We will leave behind the interim calculations and give the probabilities of conditions:

$$P_0^{-1} = \sum_{i=0}^{n} \frac{\lambda^i}{i!} + \frac{1}{n!} \sum_{j=n+1}^{n+m} \frac{\alpha^j}{n^{j-n}}$$

Or $P_0^{-1} = \sum_{i=0}^{n} \frac{\alpha^i}{i!} + \frac{\alpha^n}{n!} \frac{\alpha/n - \left(\alpha/n\right)^{m+1}}{1-\alpha/n}, \frac{\alpha}{n} \neq 1,$

$$P_0^{-1} = \sum_{i=0}^{n} \frac{\alpha^i}{i!} + \frac{m\alpha^n}{n!}, \frac{\alpha}{n} = 1 .$$

The probability that the i channels will be occupied is as follows:

$$P_i = \frac{\alpha^i}{i!} P_0, i = 1,2,...,n.$$

The probability that all n servicing channels will be occupied and that j orders are pending is as follows:

$$P_{n+j} = \frac{\alpha^{n+j}}{n^j n!} \cdot P_0, j = 1,2,...,m.$$

An order will be rejected if all n channels are occupied and m orders are already pending:

$$P_{atm} = P_{n+m} = \frac{\alpha^{n+m}}{n^m n!} P_0 .$$

The relative capacity to service this system is $q = 1 - P_{atm} = 1 - \frac{\alpha^{n+m}}{n^m n!} P_0$, and the absolute capacity to service is $N = \lambda q = \lambda \left(1 - \frac{\alpha^{n+m}}{n^m n!} P_0 \right)$.

The average number of the occupied channels is equal to the absolute capacity to service the system divided by the intensity of order servicing:

$$\overline{n} = \frac{N}{\mu} = \alpha \left(1 - \frac{\alpha^{n+m}}{n^m n!} P_0 \right),$$

where $\overline{n}$ is the average number of the occupied service channels.

The average number of pending orders:

$$\overline{m} = \frac{\alpha^{n+1}}{nn!} \cdot \frac{P_0\left(m+1-m\frac{\alpha}{n}\right)\left(\frac{\alpha}{n}\right)^m}{\left(1-\frac{\alpha}{n}\right)^2}, \frac{\alpha}{n} \neq 1,$$

$$\overline{m} = \frac{\alpha^{n+1}}{nn!} P_0 \frac{m}{2}(m+1), \frac{\alpha}{n} = 1.$$

The average number of the submitted orders (being serviced and pending) is $\overline{k} = \overline{n} + \overline{m}$.

The average pending time of an order is

$$t_l = \frac{\alpha^n P_0}{nn!\mu} \cdot \frac{1-\left(\frac{\alpha}{n}\right)^n\left[1+m\left(1-\frac{\alpha}{n}\right)\right]}{\left(1-\frac{\alpha}{n}\right)^2}, \frac{\alpha}{n} \neq 1,$$

$$t_l = \frac{\alpha^n}{nn!\mu} P_0 \frac{m}{2}(m+1), \frac{\alpha}{n} = 1.$$

The average time of order processing is $t_s = t_l + \frac{q}{\mu}$.

Such an example sometimes fails to meet the reality in the complicated modern world. In the activity of organisations we often come across more sophisticated systems, which are influenced by external factors. Often, the orders are not pending; they are just withdrawn depending upon the length of the queue. The customer has a choice among several alternatives, e.g. to choose a competitive organisation with better servicing parameters. In the modern world, not everything is available for mathematical description and there are parameters that could have an influence upon the choice of the customer, namely: quantitative factors, i.e. order pending time, order servicing price; qualitative factors, i.e. quality of servicing, quality of the product, etc.

Quality parameters of organizations activity scenarios may be evaluated by parameters used in massive service theory [4], though when implementing or creating new information systems still not enough attention is given to these problems. It is sought that information system would start functioning as fast as possible without paying attention that later IS changes will be much more complicated for several reasons – already functioning system will have to be changed, organizations employees will already be familiar with the system, and processes existing in organization will be settled. If information system has not enough flexibility features, attempts to change such system can bring to systems kinks and at the same time to kinks of whole organizations activity, which may cause a necessity of changing information system at all, not improving present one.

4. Conclusion

When making modern IS creation models it is advisable to compound features of UML and Petri nets. Summation of UML and Petri nets models enables offering solid ways of writing processes. When converting UML models into Petri net, direct converting way is most efficient for primary analysis.

Other extensions of the Petri net (object Petri nets and COO) are not for implementation of converting UML into Petri net, and are used as separate modeling tools and require further research proving their objectivity's adequacy to settled standards.

This paper offered methodology of systems modeling, from organizations activity models to modular software that is implementing business transactions, and is created by the basis of Petri net with object features of UML. In modeling process with change of business conditions also changes information system, which influences logical relations denoting all of model activity transactions, also informational relations that cause them.

The paper offers the use of Petri nets for static and dynamic process modeling, extending their facility with object UML features; it also offers methods how to convert UML diagrams into Petri net and use it for modeling of organizations activity processes.

Facility of massive service theory was analyzed, improving quality characteristics of modeled processes. Decisions, giving Petri nets an object orientation maximizing UML features, were analyzed and offered. Cases are given how UML features are moved for created models, which characterize parallel relations; those features are both, schemes and general logical structures. Petri nets models made by mathematical logic code are written with analytical expressions (with actions – transmissions and states) and nets executable rules are created. Created methodology enables creating executable rules depending on various functions and on implementing marks systems, entering weighted ratio. This is how new methodology of information systems designing is implemented, which extends implementation facility of systems software designing that is much more efficient that using separate methods.

Petri nets theoretic aspects are further developed, especially creating Petri nets executable parameters after determined and stochastic values of weighted ratios. Results of further research will be represented in following works.

References

[1] **M. Abolhassani, J. Barjis.** Applying Business Objects for Simulation, 2001, 6. http://citeseer.nj.nec.com/

[2] **J. Barjis, J.F. Escola.** Simulation of organizational processes combining semantic analysis and Petri nets. *Department of Information Systems, Delft University of Technology, The Netherlands,* 2001. *http://is.twi.tudelft.nl/~barjis*.
Superior de Technologia de Setúbal,Rua Vale de Chaves, Estefanilha, Portugal.
http://www.est.ips.pt/docentesweb/jfilipe.

[3] **Goamaa H.** Designing Concurrent, Distributed, and Real-Time Applications with UML. 2002, 700, (*In Russian*).

[4] **I. Katilius, I. Skucas.** Simulation of Business Processes According to Petri Net Theory. *Information technology and control*, 2003, No.2(27). ISSN1392-124X. *Kaunas, Technologija*, 2003, 12-17.

[5] **D. Piterson.** (). Petri nets theory and system modeling. *Moskow, Mir*, 1984, 264. (*In Russian*).

[6] **S. Pllana, T. Fahringer.** UML based modelling of performance oriented parallel and distributed applications. *Proceedings of the 2002 Winter Simulation Conference, Austria*, 2002.

[7] **D. Rambo, A. Jakobcon, G. Booch.** Unified Modeling language. *Moscow. Piter*, 2002, 652. (*In Russian*).

[8] **W.M.P.Van der Aalst, K.M. Van Hee.** Framework for Business Process Redesign. *In J.R. Callahan, editor, Proceedings of the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (*WETICE* 95), 1995, 36-45, *Berkeley Springs. IEEE Computer Society Press.* *http://citeseer.nj.nec.com/*.

[9] **W.M.P. Van der Aalst.** The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers,* 8(1), 1998, 21-66, 53.

[10] **W.M.P. Van der Aalst.** Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems,* 2000, 19. *http://citeseer.nj.nec.com\publications.htm*.

[11] **G. Wirtz, H. Giese.** Using UML and Object – Coordination-Nets for Workflow Specification, 2000. *http://www.math.uni-muenster.de/cs/u/versys/index/html*.