Third Int. Workshop on GNs, Sofia, 1 Oct. 2002, 49-58

ON SOME GENERALIZED NET MODELS OF MULTIAGENT SYSTEMS - PART II

Trifon Trifonov[1] and Krassimir Atanassov[1]
[1] CLBME - Bulg. Academy of Sciences, Acad. G. Bonchev Str., 105 Block, Sofia-1113, Bulgaria,
e-mails: trifon.trifonov@mail.clbme.bas.bg, krat@bgcict.acad.bg

Abstract: Some generalized nets (an extension of the Petri net) models of multiagent systems are described.

Keywords: Agent, Generalized net, Multiagent system

1. INTRODUCTION

Multiagent systems are tools for system modelling, which rely on task distribution among several autonomous and intelligent units, called agents. Such systems quickly find application in areas, such as engineering, computer science and artificial intelligence. Many authors have tried to formalize the terms "agent" and "multiagent system", and today there exist many different approaches to multiagent architectures (see [2]-[8]). Multiagent systems develop fast, because they offer a new look at the problem of process modelling. They supply parallel task execution by distributing it among autonomous intelligent units (agents), which inhabit a dynamic environment, about which they sometimes do not have full information. As in [1], we will think that an agent is "...rational, communicative, active, autonomous..., flexible, adaptive, self-learning and mobile". We consider the main components of the agents to be:

- input
- output
- central decision-making unit
- unit for message generating and interpreting.

In [1], the authors have proposed generalized nets as tools for multiagent system modelling. The full definition of the concept of a Generalized Net (GN) can be found in [9]. Some of the reasons for this choice are that:

- GNs have proved to be a suitable tool for modelling other formal objects for artificial intelligence (e.g. expert systems [10], machine learning [11])
- like multiagent systems, GNs supply parallel task execution through appropriate algorithms (see [9])
- multiagent systems are commonly used for process modelling or solution of a particular problem, so it seems appropriate to look for another modelling tool their representation, such being GNs
- apart from being a formal tool, GNs have a convenient visual representation which to help monitoring a process, simulated by a multiagent system.

In [1] we regarded the possibility for modelling an entire multiagent system in the terms of GNs, constructing the first GN-model of a multiagent system. We will remind this model and then present other possible models.

2. SECOND GENERALIZED NET MODEL OF A MULTIAGENT SYSTEM

In the present model we shall consider the following preliminaries :

- the problem we want to solve is given - this could be a particular task, or a particular process;
- the connections between agents are preliminary determined;
- the role of every agent in the system is cleared out.

In [1] we used the term *A-transition.*

Definition: *A-transition* is a transition

$$< L_1, L_2, t_1, t_2, r, M, \Box >,$$

for which:

1) there exists a place $P \in L_1 \cap L_2$ (static condition),

2) $L_1 - L_2 \neq \emptyset$ or $L_2 - L_1 \neq \emptyset$,

3) for every place $x \in L_1 - L_2$ either an A-transition $Z_1 =< L_1', L_2', ... >$ exists and $x \in L_2' - L_1'$, or it is an input place for whole GN,

4) for every place $x \in L_2 - L_1$ either an A-transition $Z_1 =< L_1', L_2', ... >$ exists and $x \in L_1' - L2'$, or it is an output place for whole GN,

5) in every moment of the active period of the GN there is exactly one token T_P in place P (dynamic condition).

We call place P the *agent place* and token T_P - the *place token,* i.e., the token will loop only in this place. We shall assume that every A-transition represents exactly one agent. A multiagent system is modelled by a GN E, for which

$$(\forall Z \in A)(Z \text{ is an A-transition})$$

The logic of an agent is built in the predicate index matrix r of the A-transition, and the characteristics of the state token represent the agent state in an arbitrary moment of the GN model time. Agent communication is performed by transition of tokens from one A-transition to another via their common place - output for the first transition and input for the second one. When an agent, modelled by an A-transition, decides to initiate communication, its state token splits into two tokens, one of which is the new state token, and the other token carries in its characteristics information about the other agent. The last token is placed into an appropriate output place, from which it is accepted by the other agent, and merges with its state token.

The advantages and disadvantages of the model are cleared out in [1], but for the sake of the presentation completeness we will summarize them here. At first, it is seen that we model a prebuilt multiagent system, as every agent's logic should is preliminarly determined. Then, the connections between agents are fixed at the design phase of model and cannot be altered. If the agents in the multiagent system should be highly communicative, then all the connections between them can overload the net structure, making it too complex even

for computer simulation. In [1] various solutions are offered to reduce the negative effects of such disadvantages. These include applying of the optimization operator $\mathcal{G}_4$, which is described in details in [9], as well as using Self-Modifying GNs (SMGN) to model adaptive agent architectures.

In fact the main disadvantage of this GN model is that it supposes the multiagent system is built "at once". It can be considered as a direct representation of a multiagent system in GNs. Although this is a formal translation from one object to another, in practical use it may be difficult to apply it directly. It is commonly not suitable for one to think in terms of multiagent systems and to design in terms of GN In the next section we will regard a rather different approach. We will consider a particular GN model and will enhance it with a multiagent system.

3. DESIGNING A MULTIAGENT SYSTEM IN A GENERALIZED NET MODEL

Let us suppose we want to design a working GN model, solving a problem or simulating a process. Let us also suppose we want to use a multiagent system in order to obtain parallel execution and task distribution. We assume that we are able to build an appropriate GN model. We shall not concentrate on that step of the solution, more information about designing GN models can be found in [9]. Let

$$E =<< A, \pi_A, \pi_L, c, f, \theta_1, \theta_2 >, < K, \pi_K, \theta_K >, < T, t^0, t^* >, < X, \Phi, b >>$$

be the GN model. We will build a new model E', which has a multiagent system built-in. Let

$$Z =< L_1, L_2, t_1, t_2, r, M, \Box >, Z \in A$$

Let us describe a new transition (see Fig. 1)

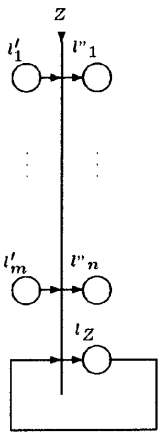

Fig. 1

$$Z' =< L'_1, L'_2, t_1, t_2, r', M', \Box >,$$

corresponding to Z, where

$$L'_1 = L_1 \cup \{l_Z\}, L'_2 = L_2 \cup \{l_Z\}, l_Z \notin L$$

$$(\forall l_1 \in L_1)(\forall l_2 \in L_2)(r'_{l_1,l_2} = r_{l_1,l_2} \& M'_{l_1,l_2} = M_{l_1,l_2})$$
$$(\forall l_1 \in L_1)(r'_{l_1,l_Z} = false \& M'_{l_1,l_Z} = 0)$$
$$(\forall l_2 \in L_2)(r'_{l_Z,l_2} = false \& M'_{l_Z,l_2} = 0)$$
$$r'_{l_Z,l_Z} = true, M'_{l_Z,l_Z} = 1$$

In fact for every transition $Z \in A$ we specify an unique place l_Z and token k_Z, where $l_Z \notin L$ and $k_Z \notin K$.

Let us consider the new GN model

$$E' =<< A', \pi_A, \pi_L, c', f', \theta_1, \theta_2 >, < K', \pi_K, \theta_K >, < T, t^0, t^* >, < X', \Phi', b >>$$

where

$$A' = \{Z' | Z \in A\}$$
$$L_a = \{l_Z | Z \in A\}$$
$$K_a = \{k_Z | Z \in A\}$$
$$K' = K \cup K_a$$
$$(\forall l \in L)(c'(l) = c(l)) \& (\forall l_Z \in L_a)(c'(l_Z) = 1),$$
$$(\forall k \in K)(X'(k) = X(k)),$$
$$(\forall k \in K)(\Phi'(k) = \Phi(k)).$$

The only elements we have not yet fully defined are the predicate index matrix r', the predicate evaluating function f', and the characteristic assignment functions X' and Φ'. These are the GN elements that will be used to model the multiagent system. Informally, we add to each transition Z in the GN a new place l_Z, which is input and output for the transition. This place represents an agent, attached to the transition. Inside each "agent" place there must always be exactly one token, k_Z, which, as in the previous model, represents the agent state. Similarly we will call it the state token. No tokens may enter or leave the agent place, so that the state token does not have any direct interaction with other tokens. We have already defined the characteristic functions X' and Φ' on all the tokens in K. Considering a state token, we must define these functions in an appropriate way in order to model the behaviour of a multiagent system. As in the previous model, we cannot give strict definitions for these functions, as we are offering a general model of a multiagent system. However, if we assume that there is a problem defined, then there are certain properties that the characteristic functions should have. The X' function describes the initial characteristics of the tokens, and therefore the initial agent states. If the regarded multiagent system supposes initial agent knowledge and logic, then X' should be defined to describe the initial characteristics of each agent's state token, according to the agent's position in the multiagent system. The Φ' function is more interesting, as soon as it describes the token characteristic in the next moment of time. The agent paradigm supposes that an agent is an intelligent unit and still agents depend on each other and communicate in the whole multiagent system. In this GN model, Φ' describes the whole behaviour of an agent. As long as Φ' depends on the previous characteristic of the state token, it determines the next move of the agent, according to its previous actions and its initial state. We assume that if the function is defined to handle as many cases of agent behaviour as possible, then it can model intelligent reactions of an agent. On the other hand, Φ' can depend not

only of the previous characteristics of the considered token, but on all the other tokens' characteristics. Therefore, by describing Φ' appropriately, we can represent some interaction between the agents, which hardly can be called "communication", but more suitable is to be called "dependence". An agent can depend not only on the state of another agent, but also of the previous states it has been in. Of course, considering some restrictions in GNs, no characteristic function can depend on future events, or else a deadlock can occur. And finally, Φ' can depend on other GN elements that are outside the multiagent system model. Although, if we allow for a certain model some agent to depend on an arbitrary GN element, this could make a model unclear. Since every agent place is attached to a transition, it will be reasonable to impose the restriction for an agent to depend only on the elements, concerning its transition, e.g. its predicates, arc capacities, input and output places, and the tokens inside them. In fact this is not a restriction, since we allow any agent's behaviour to depend on other agents' characteristics. Thus, an agent can receive indirect information about any particular element of the GN by "contacting" the corresponding agents. Implying such condition is not a restriction but a "design rule" in order to produce a clear model. Another possibility is to divide each agent's characteristics in two groups - "public" and "private". Then we require an agent's Φ' function to depend only on such characteristic of another agent, which are classified "public". Such classification has descriptive, rather than semantic meaning and it should be considered only if necessary. For example, let us regard a multiagent system with its agent's characteristics classified. If we decide to expand such a system by adding new transitions and agents, then they should consider only public characteristics of other agents, which would reflect on the definition of their characteristic functions. As conclusion, the agent behaviour, described by the Φ' function, is determined by the agent's history, by the states of other agents and indirectly by the state of the whole GN model.

The predicate index matrix r' and predicate evaluating function f' define the application of the multiagent system in the GN. Regarding the model we discuss, we suppose that there is an adequate reason to design a multiagent system in a GN - e.g. to aid solving a problem, or to help in some process simulation. As long as E' is an extension of E, we may assume that most of the elements in r' are the same as in r. In fact, for a certain transition we need to alter only the predicates for the arcs, which must let through tokens according to the state of an agent (the agent token characteristics). Of course, the predicate condition may be more complex and describe composite dependences on several agents. Since the predicate index matrix is the most flexible element of the GN, it can serve in the interaction between the multiagent system and the rest of the GN model. For example, the control over an entire transition can be turned over to an agent, by defining appropriately the transition's predicate index matrix. On the other hand, agent depending conditions can easily be mixed with other conditions, producing a composite predicate, depending partially on an agent and partially on another condition, regarding the whole GN model. In conclusion, by defining these GN elements: r', f', X', Φ', we can consider the task of describing a multiagent system inside a GN model completed.

Formally, the model just described seems an adequate solution. It is simple and builds the multiagent system over an existing GN. We have reduced the number of the GN elements to be described to four - r', f', X', Φ'. The predicate index matrix r', as well as the evaluating function f' can be easily defined, having in mind the task we want to solve. However, we have completely disregarded the complexity that the agent modelling functions X' and Φ' can reach, thus making the real implementation and simulation of the model impossible. Since

they are both used to describe agent behaviour, they gather all the load in the description of a multiagent system. It is not improbable for these function not to have any convenient representation. Thus, for a certain multiagent system we may know that it is formally modellable, but it can be difficult to implement it. This is a disadvantage, as long as GNs are commonly used to simulate a process. Since X' and Φ' describe in details how should the agent behave in different cases, we receive a long and unclear definition. This reduces the abstraction level and takes the model closer to the particular implementation and further from the multiagent system idea. There is another disadvantage of this approach. GNx are often used not only to simulate, but also analyze processes. The time in GNs allows to track down temporal dependence and order of multiple events. Since we assume that the calculation of characteristic function does not take more than one step of the model - a very rough approximation indeed, this model of a multiagent system hides clear temporal causal relationships. The dependence of X' and Φ' on characteristics of other tokens is regarded in [12] as an "ideal connection". The solution to this problem is to replace ideal connections with material connections. This approach is discussed in the next section.

Comparing the two models described until this point, we see that they differ a lot. The first model describes a whole multiagent system, while the second extends a GN model to contain a multiagent system. The first model is clear, since it represents a multiagent system using all GN elements, but is restricted, as we may use only A-transitions, describe processes only in terms of multiagent systems by predefining a static structure of the net. The second model is more flexible, since it can combine GN elements with multiagent system elements, but its definition uses a small subset of the GN elements, thus making the modelling of more complex multiagent system difficult. We shall consider another model, which combines some of the advantages of both models.

4. THIRD GENERALIZED NET MODEL OF A MULTIAGENT SYSTEM

Let us consider the term extended A-transition.

Definition: *Extended A-transition* is called a transition

$$< L_1, L_2, t_1, t_2, r, M, \Box >,$$

for which (see Fig. 2):

1) there exists place $P \in L_1 \cap L_2$, which is called *agent place* (static condition),

2) $L_1 - L_2 \neq \emptyset$ or $L_2 - L_1 \neq \emptyset$,

3) **there exists** a place $x \in L_1 - L_2$ either an extended A-transition $Z_1 =< L_1', L_2', ... >$ exists and $x \in L_2' - L_1'$, or it is an input place for whole GN, *or*

4) **there exists** $x \in L_2 - L_1$ either an extended A-transition $Z_1 =< L_1', L_2', ... >$ exists and $x \in L_1' - L2'$, or it is an output place for whole GN,

5) in every moment of the active period of the GN there is exactly one token T_P in place P, which is called *state token* (dynamic condition).

Places x that satisfy 3) or 4) will be called *A-places.* Any arc between two A-places will be called an *A-arc.*

The main difference between *A-transition* and *extended A-transition* is that we do not restrict the extended A-transition to have common places *only* with other A-transitions. Informally, an *extended A-transition* is a hybrid between a normal transition and an *A-transition.* Since here the universal quantifier along with 2) implies the existential quantifier,

the term *extended A-transition* is more general than the term *A-transition*, i.e. every *A-transition* is an *extended A-transition.*

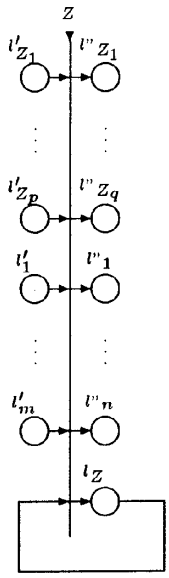

Fig. 2

Let Z_1 be a transition and Z_2 be an A-transition. Then, if the operation $Z_1 \cup Z_2$ is correctly defined, the transition $Z = Z_1 \cup Z_2$ is an extended A-transition. This is easily checked, as conditions 1), 2) and 5) are fulfilled from the corresponding conditions in the definition of the A-transition Z_2. Since 2) is true, then either $L_1 - L_2 \neq \emptyset$ or $L_2 - L_1 \neq \emptyset$. If $L_1 - L_2 \neq \emptyset$, then 3) in the definition of A-transition implies 3) in the definition of extended A-transition. Else, $L_2 - L_1 \neq \emptyset$, then 4) in the definition of A-transition implies 4) in the definition of extended A-transition. Therefore, "3) or 4)" is true and Z is an extended A-transition.

Let us consider a GN model:

$$E' =<< A', \pi'_A, \pi'_L, c', f', \theta'_1, \theta'_2 >, < K', \pi'_K, \theta'_K >, < T', t'^0, t'^* >, < X', \Phi', b' >>$$

and a GN model of a multiagent system, built according to the A-transition model

$$E'' =<< A'', \pi'_A, \pi''_L, c'', f'', \theta''_1, \theta''_2 >, < K'', \pi''_K, \theta''_K >, < T'', t''^0, t''^* >, < X'', \Phi'', b'' >>$$

All transitions in A'' are A-transitions. We can assume that:

$$K' \cap K'' = \emptyset$$

$$L' \cap L'' = \emptyset$$

Furthermore, let us assume that $|A'| = |A''|$. This can be achieved by adding empty transitions to either A' or A''. And finally, let us assume that there exists a function $f : A' \rightarrow A''$, so that f is a bijection and $(\forall Z \in A')(Z$ and $f(Z)$ have the same names). This will allow to merge all the transitions of the both GNs. The union operator over GNs is described in [8]. Now let us consider the model $E = E' \cup E''$. If $E =<< A, ... >, ... >$, then it is true that $(\forall Z \in A)(Z$ is an extended A-transition). The received GN E is a GN model with a multiagent system.

The formal way to produce a GN model is rather inconvenient for it imposes to have already designed GN and multiagent system models. In fact it is far more difficult to design them separately, because they work together. Before presenting informal methods for the design of such a multiagent system GN model, let us look more closely at the model just described. Formally, such a model is a GN, consisting of extended A-transitions. Informally, just as with A-transitions, we will regard the agent place in an extended A-transition as an agent. The state token's characteristic will display the agent state. The communication between agents is performed in the same way as in the A-transition model. Since an input A-place is an output A-place for another extended A-transition or it is an input for the whole GN, any tokens in it should merge with the state token to inherit some of the characteristics and thus receive information. Similarly, when an agent decides to transmit information to another extended A-transition or the environment, its state token splits and the new token, bearing outgoing information, is transferred to the appropriate output A-place. This communication model was already discussed in Section 2, and in more detail in [1]. We will only concentrate on the new features the extended A-transition provides. At first, compared to the previous model, the communication between agents is realized through a material connection, rather than an ideal connection (see [12]). This approach has the disadvantage that communication needs more model time to be performed than the simple dependence in the characteristic functions, considered in the previous section. However, the material connection allows a much clearer representation of the model. Since communication is performed via the visible GN elements in a GN model simulation - such as the tokens and transitions, it is much easier in this model to display casual relationships between agents. Compared to the second model, here we have true communication, rather than just dependence on characteristics. In the multiagent system concept, the communication between two agents is a process, which invloves both agents. The second model displays only the role of the agent, which initiates the communication. By defining dependences of X' and Φ', it is theoretically possible to obtain information about another agent without its participation. Therefore, the model currently considered describes better the dynamics in communication. Another advantage of this model is the usage of the extended A-transition. As with the A-transition model, it allows a clearer and more intuitive definition of an agent. The predicate index matrix r and the predicate evaluation function f carry out most of the agent logic, connected with communication. Agent behaviour is managed again by the characteristic functions X and Φ, but they only depend on previous agent states, i.e. the state token characteristics.

The considered model combines two approaches. It allows to solve the task not only in terms of multiagent systems, but by combining the advantages of the two instruments. In fact in the GN model, built of extended A-transitions, exist two separate contours. Thus the GN and the multiagent system can work paralelly and independent. If necessary, the model can be built in such way, that the GN contour uses information, gathered by the multiagent systems - state token characteristics, and the agents in the multiagent system can use information of the GN model components. This can easily be done only by using GN components - predicates and characteristic function. However, following the idea, that the model should be clear, at the design phase should be pointed the roles of both the multiagent contour and the ordinary GN contour. The two contours should not be mixed - i.e. one place to be used in both contours. This is only suitable as means for material connection between the two contours. However, for the model to be more effective, the two contours should be separated both structurally and semantically and any connection between them should be defined in the predicates and/or the characteristic functions ("ideal connection").

We can look at this model at two different angles. At first, we can regard it as a GN model, which has be enhanced with a multiagent system. Note that we do not require the multiagent system contour to be continuous, so in one GN model there could be several separate multiagent systems, each built for a different purpose. Therefore, we can enhance an arbitrary GN model with different multiagent systems. On the other hand, we can regard this model as a multiagent system, connected to a non-agent-based system, such as the GN. In both cases we get a model, which combines the flexibility of the GN, and the distributed intelligence of the multiagent system.

As we discussed, the disadvantages of the second model - nonintuitive and hard definitions, difficult realisation of the model, could be overcomed by using extended A-transitions. There still exist the same problems, which occured with the A-transition model - fixed connections between the agents and overloaded structure of the model. The solutions can be the same as in the A-transition model (see [1]) namely usage of SMGN, serving agents and optimization operator in order to reduce the number of places. Still, in the A-transition model these problems were of much greater importance, since we used the GN to model only a multiagent system. In the extended A-transition model, we can rearrange the roles, played by the multiagent system and the rest of the GN in order to optimize the multiagent system representation. There is no formal way to define which tasks are suitable for the agent contour and which are suitable for the GN contour. However, in this model we have the possibility to restructure the model anytime we encounter hardships in the model representation. This optimization can be not only to the multiagent contour - by shifting responsibility to the GN, but a GN model can also benefit by designing a multiagent contour to undertake some of the work.

We already see that the extended A-transition model can be created not only in the formal way, described above, i.e. by uniting a GN model with an A-transition model. We can start by designing a new GN model (as described in [9]) and if we encounter a process, which can be conveniently modelled with the use of multiagent system, then we simply include a multiagent contour in the GN model we are designing. Another approach would be if we start designing a multiagent system and we decide to include some non-multiagent elements, which can be described with GN elements. Here we see the last disadvantage of the A-transition model overcome. There is no need to design the whole model "at once". We can build a GN model and then simulate it. If we decide to add other GN components or even other multiagent systems, we can achieve this just by adding appropriate contours and defining their elements.

5. CONCLUSION

We have considered different approaches to multiagent systems modelling with GNs. The first model (A-transition model) is used to model a whole multiagent system, which is already designed. This is a simple and intuitive model, using different GN elements to model agents, agent communication and agent logic. Its main disadvantage is the restrictions to use only A-transitions, which can result in difficulties in practical realizations of the model. The second model describes how to enhance a GN with amultiagent system. A formal method for the GN conversion is presented and the multiagent system is built only of four GN elements - the predicate index matrix, predicate evaluating function and the characteristic function. In this simple model agent communication is substituted by function dependence, i.e. "ideal

connection". Problems with the pratical application can arise when the dependences of agents on one another are complex, or when the agent logic is too complicated to use a simple function. The model, considered in the last section, is closest to the creating a suitable modelling instrument, using multiagent systems. Starting from the idea of multiagent system representation in terms of GNs, we conclude that the best solution is to combine both of them in order to produce a better model. Such mixed model is represented in terms of GNs, because they can represent other formal objects - expert systems, neural networks, UML diagrams, which can be included to receive a highly heterogenic GN model, which can be used in simulation of wider variety of processes.

REFERENCES

[1] Trifonov T., K. Atanassov, On Some Generalized Net Models of Multiagent Systems, Proceedings of the Second Int. Workshop on Intuitionistic Fuzzy Sets and Generalized Nets, Warszawa, 24-25 July 2002 (in press).

[2] Hodjat B., C. Savoie, M. Amamiyan Adaptive Agent Oriented Software Architecture, http://lanl.arxiv.org/pdf/cs.DC/9812014, 1998.

[3] Cohen P., A. Cheyer, M. Wang, S. C. Baeg, OAA: An Open Agent Architecture, AAAI Spring Symposium, 1994.

[4] Wolpert D., K. Wheeler, K. Tumer, General principles of learning-based multiagent systems, http://lanl.arxiv.org/pdf/cs.MA/9905005, 1999.

[5] Wolpert D., K. Tumer, An introduction to collective intelligence, http://lanl.arxiv.org/pdf/cs.LG/9908014, 1999.

[6] Haddadi A., Communication and Cooperation in Agent Systems, Springer, Berlin, 1995.

[7] Intelligent Agents, (M. Wooldridge, N. Jenkings, Eds.), Springer, Berlin, 1995.

[8] Intelligent Agents. V, (J. Müller, M. Singh, S. Rao, Eds.), Springer, Berlin, 1999.

[9] Atanassov K., Generalized Nets, Singapore, World Scientific, 1991.

[10] Atanassov K., Generalized Nets in Artificial Intelligence, Vol. 1: Generalized Nets and Expert Systems, Sofia, 'Prof Marin Drinov' Publishing House of Bulgarian Academy of Sciences, 1998.

[11] Atanassov K., H. Aladjov, Generalized Nets in Artificial Intelligence, Vol. 2: Generalized Nets and Machine Learning, Sofia, 'Prof Marin Drinov' Publishing House of Bulgarian Academy of Sciences, 2000.

[12] Atanassov K., Radeva V., On the abstract systems with properties, Advanced Studies in Contemporary Mathematics, Vol.4, No. 1, 2001, 55-71.

[13] Koycheva E., T. Trifonov, H. Aladjov, Modelling Of UML Sequence Diagrams With Generalized Nets, Proceedings of the First Int. IEEE Symposium "Intelligent Systems" IS'2002, Varna, Sept. 10-12, 2002, Vol. 3, 79-84.