Serdica J. Computing **1** (2007), 255–266

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

GENERALIZED NETS MODEL OF AN E-LEARNING SYSTEM SOFTWARE ARCHITECTURE[*]

Aleksandar Dimov, Sylvia Ilieva

Abstract. Component-based software engineering and software architecture are tightly connected areas in computer science. Software architecture presents the functionality of the system as decomposition into components, the properties of these components and the connectors between them. This paper illustrates a methodology for application of the theory of Generalized Nets (GNs) as a language for description of software systems architecture. According to this methodology, every component in the system, as well as every connector is represented by a single GN transition. This way the positions of the transition describe the ports of components and connectors in the system. This paper introduces a model of the component-based architecture of the e-learning system ARCADE, which is created, with respect to the proposed methodology for description with GNs. The four main subsystems are regarded as components in the GNs model. Their additional sub-modules are presented as the services provided by the components. Method calls are regarded as the connectors between these components. Further, the GNs model is compared with the existing UML diagrams, specifying the design of ARCADE.

ACM Computing Classification System (1998): D.2.11.
Key words: Component-based software engineering; Software architecture; Generalized Nets; e-Learning systems.

[*]The paper has been presented at the International Conference Pioneers of Bulgarian Mathematics, Dedicated to Nikola Obreshkoff and Lubomir Tschakaloff , Sofia, July, 2006.

1. Introduction.

Component-Based Software Engineering (CBSE) emerged because of the increased interest in reuse of preexisting software functionality. The introduction of the notion of component [6, 11] is the key innovation in this discipline. Component is an independently developed piece of software and can be integrated with other components in order to build larger units. The interaction point of components with the rest of the system is called interface. It should be clearly specified, because it defines the services that components provide and require for their execution. During integration components may be adapted by writing wrappers, but their source code may not be modified.

Although promising, the approach of building software systems with components is a tedious task, as reuse of preexisting software code hides some problems [7]. Study of software architecture is a key concept for simplification of the design, implementation and maintenance of complex software systems. Thus, component-based software engineering and software architecture are tightly connected areas in computer science. To address problems with integration of components clear distinction should be established between the communicational and computational artifacts within programs [9, 10]. Computational aspects of systems are introduced by components, while their interconnections are maintained by *connectors*.

In order to achieve maximum benefit when reason about software architecture, one should be able to specify it in a formal way. The so-called Architecture Description Languages (ADLs) are aimed at description of software architecture with uniform means at different levels of abstraction [4, 8].

This paper shows an experimental model of the software architecture of ARCADE system [1, 3] that applies the notation of Generalized Nets [2] as an ADL. The model is based on the publications about the architecture of ARCADE.

The paper is organized as follows: Section 2 makes a brief introduction to GNs and gives an overview of the principles for creation of software architecture models with them. Section 3 describes the model of the architecture of ARCADE system and finally, Section 4 concludes the paper.

2. Methodology for description of software architecture with GNs.

This section gives general information about modeling of software architecture with GN. They follow the basic methodology for construction of GN, described in [2] and reflect the requirement towards ADLs for description of components, connectors and their configurations, as outlined in [8]. GNs have many applications, but currently, they are not used as a notation for description of software architecture. In this paper the methodology for usage of GNs for software architecture description is applied, as originally described in [5].

GNs is notation, aimed at description of parallel processes. Their main building unit is the so-called transition, which should have one or more input and output positions (places). Tokens are used to present the information, processed by the GN. They may move from transition's input to output positions, if a corresponding condition predicate is true.

The first problem that system architects should solve in order to build the GN model is to identify the constituent components, connectors and their attachments of the system. Further, a single GN transition is assigned for every component and connector. This way input interfaces should correspond to input positions of a transition and respectively, output interfaces – to output positions.

Essential aspects for description of components and connectors[*] are their semantics and syntax. Syntax is usually defined by their input and output ports. Hence the corresponding transition places belonging to the sets L' and L'' (which describe the ports) take part in the modeling of the syntax of components and connectors. Other part of GNs that may be used to model syntax of architectural elements are the extended transition's type $\Box$ and capacities of positions and throughput capacities of the arrows between positions, given in the matrix M.

Modeling of semantics of architectural elements has two facets – gray or black box representation. When modeling the element as a gray box some internal states of the execution should be included in the model. They are described with auxiliary (hexagonal) positions and their number corresponds to the number of internal states that should be modeled. These additional positions should be connected in loops in order to represent internal states. Other auxiliary positions may be used to model terminal states of the execution of the element. The actual processing is described mainly with the transition conditions r, characteristic functions Φ and the priorities of the positions π_L. Transition conditions represent the control flow inside the component. Processing of user data is modeled by the characteristic functions of GN and priorities of the positions determine the sequence of the processing activities of the components.

When a component or connector is modeled as a black box, the corresponding transition will include auxiliary positions only for terminal states. This means that it will not have connected in loops positions. In this case, the services of the component are globally modeled as atomic operations, which are independent from each other. Thus, priorities of the positions will be less significant and may be neglected.

In the next section we continue with concrete model of ARCADE system.

[*]Both components and connectors are also denoted as architectural elements throughout this paper.

3. Software architecture model of ARCADE.

3.1. Brief overview of ARCADE system. *ARCADE* (*Architecture for Reusable Courseware Authoring and Delivery*) is an e-learning platform, developed in the department of Information technologies at FMI, University of Sofia, by a team of engineers, specialists and students. It has been developed, using the newest trends for organization, authoring and delivery of distance courses. Nowadays, the system is used for management of the department's master courses. The system was designed following a component-based approach and its main subsystems are:

- Course and Curriculum Management Module (*CCMM*) – this subsystem realizes different functions that provide services supporting the work of instructors and course administrators.
- Users and System Management Module – (*USMM*) – this subsystem implements most of the functions that concern system administrators.
- Communications module (*CM*) – it has five services, which offer different communication options between *ARCADE* users.
- Assessments and Assignments Module (*AAM*) – this subsystem implements the functionality of services for student assessments.

All of the described above subsystems communicate through a database, which stores information for the users and users' groups. General architectural scheme, showing the relationships between the defined subsystems (without the database) is shown on Fig. 1.

3.2. GN model of ARCADE. A GN-model created in this paper is shown on Fig. 2 and assumes that all subsystems shown on Fig. 1 are components. According to the *ARCADE* implementation, components are connected with each other by methods calls, and a database, i.e. the architecture combines object-oriented style with repository style.

Place names in Fig. 2 are omitted for simplicity. They are formed as follows: $l_i^{\prime,j}$for input places, $l_i^{\prime\prime,j}$ for output places and m^j for auxiliary places, where i denotes place number (counted from top) and j denotes transition number. Further is assumed that subcomponents within the four basic system components represent services that are modeled by the corresponding input and output places of transitions.

For all transitions, that describe the model components from Fig. 2, are valid the following meanings of the places:

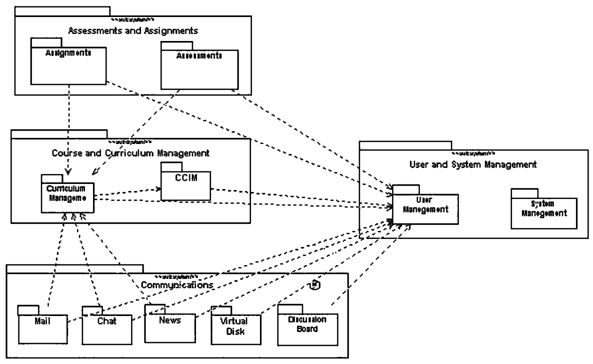

Fig. 1. Architecture of *ARCADE* system (modeled by *UML*)

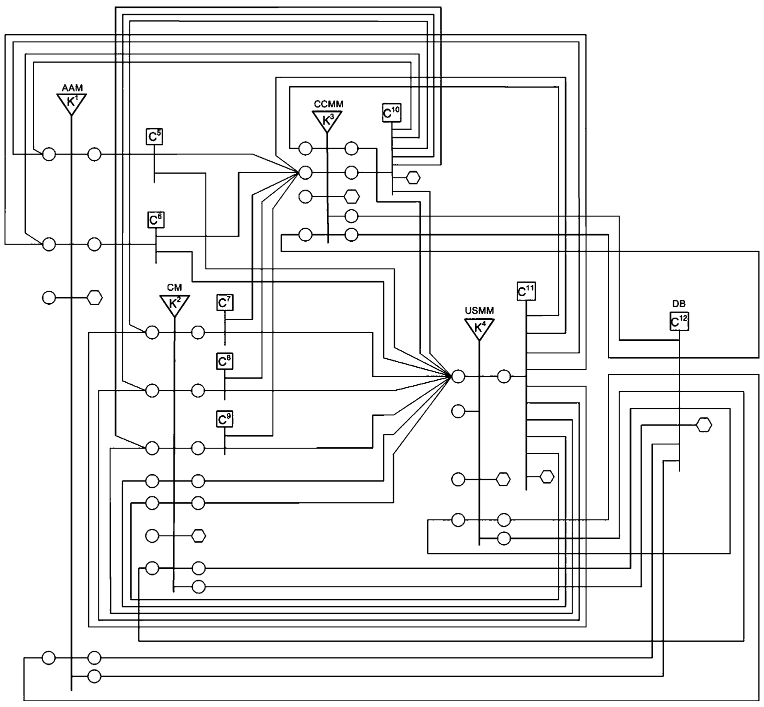

Fig. 2. GN model of the software architecture of *ARCADE*

$l'^{,j}_{m_j}$ – get the result from the query to the database;

$l''^{,j}_{n_j-1}$– request for reading from database;

$l''^{,j}_{n_j}$ – request for writing into the database;

m^j – auxiliary places, needed to drive obsolete tokens out of the model, where j is transition number, m_j is the count of input places in transition j, and n_j is the count of output places for j, $j \in\{1,2,3,4\}$.

The following general symbols are used in the model for some of the transitions condition predicates:

$W_R =$ "$\mu(\alpha, l'^{,j}_i) =$ '*read from database*' ",

$W_W =$ "$\mu(\alpha, l'^{,j}_i) =$ '*write to database*' ",

$W_1 =$ "*It is necessary to call the service for curriculum management* (*in CCMM*)",

$W_2 =$ "*It is necessary to call the service for users management* (*in USMM*)",

$UR = \neg(W_{11} \vee W_{12} \vee W_R \vee W_{W4})$.

3.3. Models of the ARCADE components.

3.3.1 Assessments and assignments module. The places of the transition, that models the assessments and assignments module (K^1), have the following meanings:

$l'^{,1}_1$ and $l''^{,1}_1$ – input and output interface of the service for assignments;

$l'^{,1}_2$ and $l''^{,1}_2$ – input and output interface of the service for assessments;

$l'^{,1}_3$ – system users input.

The index matrix with transition K^1 predicates is:

$$(1) \qquad r^1 = \begin{array}{c|ccccc} & l''^{,1}_1 & l''^{,1}_2 & m^1 & l''^{,1}_3 & l''^{,1}_4 \\ \hline l'^{,1}_1 & W_1 \vee W_2 & 0 & UR & W_R & W_W \\ l'^{,1}_2 & 0 & W_1 \vee W_2 & UR & W_R & W_W \\ l'^{,1}_3 & W^1_1 & W^1_2 & UR^1 & 0 & 0 \\ l'^{,1}_4 & W^1_1 & W^1_2 & UR^1 & 0 & 0 \end{array}$$

where:

$W^1_1 =$ "*It is necessary to call the assignments service*",

$W^1_2 =$ "*It is necessary to call the assessments service*",

$UR^1 = \neg(W^1_1 \vee W^1_2)$.

The type of transition K^1, that determines the condition for its activation, is: $\Box^1 = \vee(l'^{,1}_1, l'^{,1}_2, l'^{,1}_3)$.

3.3.2 Communications Module. The places of the transition that models the communications module (K^2) have the following meanings:

$l_1^{\prime,2}$, $l_1^{\prime\prime,2}$ – input and output interface of the e-mail service;

$l_2^{\prime,2}$, $l_2^{\prime\prime,2}$ – input and output interface of the virtual chat rooms service;

$l_3^{\prime,2}$, $l_3^{\prime\prime,2}$ – input and output interface of the news service;

$l_4^{\prime,2}$, $l_4^{\prime\prime,2}$ – input and output interface of the virtual disk;

$l_5^{\prime,2}$, $l_5^{\prime\prime,2}$ – input and output interface of the discussion board;

$l_6^{\prime,2}$ – system users input.

The index matrix with transition K^2 predicates is:

$$(2)\quad r^2 = \begin{array}{c|cccccccc}
 & l_1^{\prime\prime,2} & l_2^{\prime\prime,2} & l_3^{\prime\prime,2} & l_4^{\prime\prime,2} & l_5^{\prime\prime,2} & m^2 & l_6^{\prime\prime,2} & l_7^{\prime\prime,2} \\
\hline
l_1^{\prime,2} & W_1 \vee W_2 & 0 & 0 & 0 & 0 & UR & W_R & W_W \\
l_2^{\prime,2} & 0 & W_1 \vee W_2 & 0 & 0 & 0 & UR & W_R & W_W \\
l_3^{\prime,2} & 0 & 0 & W_1 \vee W_2 & 0 & 0 & UR & W_R & W_W \\
l_4^{\prime,2} & 0 & 0 & & W_1 \vee W_2 & 0 & UR & W_R & W_W \\
l_5^{\prime,2} & 0 & 0 & 0 & 0 & W_1 \vee W_2 & UR & W_R & W_W \\
l_6^{\prime,2} & W_1^2 & W_2^2 & W_3^2 & W_4^2 & W_5^2 & UR^2 & 0 & 0 \\
l_7^{\prime,2} & W_1^2 & W_2^2 & W_3^2 & W_4^2 & W_5^2 & UR^2 & 0 & 0
\end{array}$$

where:

$W_1^2 =$ "*It is necessary to the e-mail service*";

$W_2^2 =$ "*It is necessary to call the virtual chat rooms service*";

$W_3^2 =$ "*It is necessary to call the news service*";

$W_4^2 =$ "*It is necessary to call the virtual disk service* ";

$W_5^2 =$ "*It is necessary to call the discussion board service*";

$UR^2 = \neg(W_1^2 \vee W_2^2 \vee W_3^2 \vee W_4^2 \vee W_5^2)$.

The type of transition K^2 is: $\square^2 = \vee(l_1^{\prime,2}, l_2^{\prime,2}, l_3^{\prime,2}, l_4^{\prime,2}, l_5^{\prime,2}, l_6^{\prime,2})$.

3.3.3. Course and curriculum management module. The places of the transition that models the course and curriculum management module (K^3) have the following meanings:

$l_1^{\prime,3}$, $l_1^{\prime\prime,3}$ – input and output interface of the courses management service;

$l_2^{\prime,3}$, $l_2^{\prime\prime,3}$ – input and output interface of the curriculum management service;

$l_3^{\prime,3}$ – system users input.

The index matrix with the predicates for transition K^3 is as follows:

(3)
$$r^3 = \begin{array}{c|ccccc} & l_1''^{,3} & l_2''^{,3} & m^3 & l_3''^{,3} & l_4''^{,3} \\ \hline l_1'^{,3} & W^2 & 0 & UR_1^3 & W_R & W_W \\ l_2'^{,3} & W^3 & W_1 \vee W_2 & UR_2^3 & W_R & W_W \\ l_3'^{,3} & W^3 & W_2 & UR_3^3 & 0 & 0 \\ l_4'^{,3} & W^3 & W_2 & UR_3^3 & 0 & 0 \end{array}$$

where:

$W^3 =$ "*It is necessary to call the courses management service*";
$UR_1^3 = \neg(W_2 \vee W_R \vee W_W)$;
$UR_2^3 = \neg(W_1 \vee W_2 \vee W^3 \vee W_R \vee W_W)$;
$UR_3^3 = \neg(W^3 \vee W_2)$.
The type of transition K^3 is: $\Box^3 = \vee(l_1'^{,3}, l_2'^{,3}, l_3'^{,3})$.

3.3.4. User and system management module. The places of transition (K^4) that models the user and system management module, have the following meanings:

$l_1'^{,4}$, $l_1''^{,4}$ – input and output interfaces of the user management service;
$l_2'^{,4}$ – system users input.
The index matrix with the predicates for transition K^4 is:

(4)
$$r^4 = \begin{array}{c|cccc} & l_1''^{,4} & m^4 & l_2''^{,4} & l_3''^{,4} \\ \hline l_1'^{,4} & W_2 & UR_1^3 & W_R & W_W \\ l_2'^{,4} & W_2 & \neg W_2 & 0 & 0 \\ l_3'^{,4} & W_2 & \neg W_2 & 0 & 0 \end{array}$$

The transition type is: $\Box^4 = \vee(l_1'^{,4}, l_2'^{,4})$.

Characteristic functions, that set the tokens characteristics, are similar to previous cases. Tokens get as new characteristic first – the returned result by the service, and second – a request to an external component service if necessary. Tokens do not change their characteristics upon entrance in any of the places $l_{n_j-1}''^{,j}$, $l_{n_j}''^{,j}$ and m^j, where $j \in \{1, 2, 3, 4\}$.

3.4. Connector models in ARCADE. In the presented GN model, the connector transitions (excluding the database) are used only to move tokens towards:

- The input interfaces of the called service;

- The corresponding input interfaces of caller components, whet the tokens contain the result returned by the called component.

The tokens characteristics do not change while they are passing through the CM transitions. The arcs' capacities are infinite, because the performance of the systems, built according to the object oriented style is limited only by the components. This way, only the predicates of transitions C^j, $j \in\{5,\ldots,11\}$ are important for the system model. For $j \in\{5,\ldots,9\}$, they are as follows:

(5) $$r^j = \begin{array}{c|cc} & l_2^{\prime,3} & l_1^{\prime,4} \\ \hline l_a & W_1 & W_2 \end{array}$$

where: $l_a \equiv l_1^{\prime\prime,1}$ for $j = 5$, $l_a \equiv l_2^{\prime\prime,1}$ for $j = 6$, $l_a \equiv l_1^{\prime\prime,2}$ for $j = 7$, $l_a \equiv l_2^{\prime\prime,2}$ for $j = 8$, $l_a \equiv l_3^{\prime\prime,2}$ for $j = 9$.

The index matrix with transition C^{10} predicates is:

(6) $$r^{10} = \begin{array}{c|ccccccc} & l_1^{\prime,1} & l_2^{\prime,1} & l_1^{\prime,2} & l_2^{\prime,2} & l_3^{\prime,2} & m^{10} & l_1^{\prime,4} \\ \hline l_1^{\prime\prime,3} & W_1^{10} & W_2^{10} & W_3^{10} & W_4^{10} & W_5^{10} & UR^{10} & W_2 \end{array}$$

where:

$W_1^{10} =$ "$\mu(\alpha, l_1^{\prime\prime,3}) =$ '*returned result to the assignments service*'";
$W_2^{10} =$ "$\mu(\alpha, l_1^{\prime\prime,3}) =$ '*returned result to the assessments service*'";
$W_3^{10} =$ "$\mu(\alpha, l_1^{\prime\prime,3}) =$ '*returned result to the e-mail service*'";
$W_4^{10} =$ "$\mu(\alpha, l_1^{\prime\prime,3}) =$ '*returned result to the virtual chat rooms service*'";
$W_5^{10} =$ "$\mu(\alpha, l_1^{\prime\prime,3}) =$ '*returned result to the news service*'";
$UR^{10} = \neg(W_1^{10} \vee W_2^{10} \vee W_3^{10} \vee W_4^{10} \vee W_5^{10} \vee W_2)$.

The index matrix with transition C^{11} predicates is:

(7) $$r^{11} = \begin{array}{c|cccccccccc} & l_2^{\prime,3} & l_3^{\prime,3} & l_1^{\prime,1} & l_2^{\prime,1} & l_1^{\prime,2} & l_2^{\prime,2} & l_3^{\prime,2} & l_4^{\prime,2} & l_5^{\prime,2} & m^{11} \\ \hline l_1^{\prime\prime,4} & W_1^{11} & W_2^{11} & W_3^{11} & W_4^{11} & W_5^{11} & W_6^{11} & W_7^{11} & W_8^{11} & W_9^{11} & UR^{11} \end{array}$$

where:

$W_1^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ "*returned result to the course management service*";
$W_2^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the curriculum management service*'";
$W_3^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the assignments service*'";
$W_4^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the assessments service*'";
$W_5^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the e-mail service*'";
$W_6^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the virtual chat rooms service*'";
$W_7^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ '*returned result to the news service*'";

$W_8^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ *'returned result to the virtual disks service'*";
$W_9^{11} =$ "$\mu(\alpha, l_1^{\prime\prime,4}) =$ *'returned result to the discussion boards service'*";
$W^6 = \neg(W_1^{11} \vee W_2^{11} \vee W_3^{11} \vee W_4^{11} \vee W_5^{11} \vee W_6^{11} \vee W_7^{11} \vee W_8^{11} \vee W_9^{11})$.

The types of the described transitions, that determine the conditions for their activation, are: $\square^j = \vee(l_i)$, where $j \in\{5,\ldots,11\}$, $i \in \{1,\ldots \|L_1^j\|\}$. L_1^j is the set of the input places of transition C^j.

The model of the data base (C^{12}) is different from the described above connector models. The transition condition here is:

$$(8)\qquad r^{12} = \begin{array}{c|ccccc} & l_4^{\prime,3} & l_4^{\prime,1} & l_7^{\prime,2} & l_4^{\prime,4} & m^7 \\ \hline l_3^{\prime\prime,3} & 1 & 0 & 0 & 0 & 0 \\ l_4^{\prime\prime,3} & 0 & 0 & 0 & 0 & 1 \\ l_3^{\prime\prime,4} & 0 & 0 & 0 & 1 & 0 \\ l_4^{\prime\prime,4} & 0 & 0 & 0 & 0 & 1 \\ l_6^{\prime\prime,2} & 0 & 0 & 1 & 0 & 0 \\ l_7^{\prime\prime,2} & 0 & 0 & 0 & 0 & 1 \\ l_3^{\prime\prime,1} & 1 & 0 & 0 & 0 & 0 \\ l_4^{\prime\prime,1} & 0 & 0 & 0 & 0 & 1 \end{array},$$

The transition type of r^{12} is $\square^{12} = \vee(l_3^{\prime\prime,3}, l_4^{\prime\prime,3}, l_3^{\prime\prime,4}, l_4^{\prime\prime,4}, l_6^{\prime\prime,2}, l_7^{\prime\prime,2}, l_3^{\prime\prime,1}, l_4^{\prime\prime,1})$. In this model are affected neither the details of simultaneous processing of read and write requests, nor the storage of information on physical level. The characteristic functions of this transition are:

$\Phi_{l_4^{\prime,i}}$: "*Token α in place $l_4^{\prime,i}$ obtains as a new characteristic the data for the component i*", *for* $i \in \{1, 3, 4\}$;

$\Phi_{l_7^{\prime,2}}$: "*Token α in place $l_7^{\prime,2}$ obtains as a characteristic the data for the communications module*".

4. Conclusion.

This paper shows an example for application of the notation of GNs as an ADL. Model of software architecture of the ARCADE system is built, based on existing UML models of it, known from literature sources. The comparison of two approaches shows that a problem for the application of GNs as ADL with respect to UML is the large amount of over-crossing arcs in some cases. For example this is the situation with the object-oriented architectural style, used for design of the ARCADE architecture. Another difficult to describe with GNs architectural style is the client-server style. It is valid for these two styles, because the tokens (i.e. the information) have to pass in reverse direction (from right to left) to represent a returned result. These arcs are introduced with dashed lines on Fig. 1.

However, it is natural for UML to give opportunities for simpler presentation of object oriented architectural style in comparison with GNs as it is created especially for the needs of the object-oriented programming.

Main advantage of GNs, in comparison with UML and other ADLs in general is the possibility to follow the flow and processing of information inside the system in straightforward way. Moreover, the formal model makes it easier to reason about the system at different levels of abstraction.

Analysis of the described here model may infer some recommendations about improvement of *ARCADE* architecture. For example, unclearly differentiated connectors, deters the reuse of both components and their models. Thus, the architecture of the system could be reconsidered in order to differentiate connector and/or wrapping modules. This presumes the inclusion of other architecture styles besides the used object-oriented and repository.

The main direction for further research is the design and development of toolset supporting the construction of software architecture models, analysis of these models and the implementation and integration of actual systems based on the models.

Acknowledgements. The work reported in this paper is partially supported by the project BIS-21++ funded by the European Commission in FP6 INCO via contract no.: INCO-CT-2005-016639.

REFERENCES

[1] Aleksieva A., M. Petrov, B Bontchev. ARCADE Assessment Framework. In: Proceedings of 2nd Int. Conf. on Multimedia and ICTs in Education (m-ICTE2003), Badajoz, Spain, December 3–6, 2003.

[2] Atanassov, K. Generalized Nets. World Scientific, 1991.

[3] Bontchev B., T. Iliev. ARCADE - a Web-based Authoring and Delivery Platform for Distance Education. In: Proceedings of 1st Balkan Conference on Informatics (BCI'2003), Thessaloniki, Greece, 21–23 of November, 2003.

[4] Dimov A., S. Ilieva. System level modelling of component-based software systems. In: Proceedings of the 5th international conference on Computer systems and technologies (CompSysTech 2004), Rousse, Bulgaria June 17–18, 2004, II.7.1–II.7.6.

[5] Dimov A., S. Ilieva. Description of Software Architecture with Generalized Nets. *Journal of Information technologies and Control* **3** (2005), 18–27.

[6] D'Souza D., A. Wills. Objects, Components and Frameworks: The Catalysis Approach. Reading, MA: Addison-Wesley, 1998.

[7] Garlan D., R. Allan, J. Ockerbloom. Architectural Mismatch or why it's Hard to Build Systems out of Existing Parts. In: Proceedings of 17th International Conference on Software Engineering, 1995, 179–185.

[8] Medvidovic N., R. Taylor. A classification and comparison framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering* **26**, No. 1 (2000), 70–93.

[9] Perry D., A. Wolf. Foundations for the Study of Software Architectures. *ACM SIGSOFT Software Engineering Notes* **17** (1992), 40–52.

[10] Shaw M. Procedure calls are the assembly language of system interconnection: Connectors deserve first-class status. Proceedings of the Workshop on Studies of Software Design, Lecture Notes in Computer Science, Springer-Verlag, 1993.

[11] Szyperski C. Component Software: Beyond Object-Oriented Programming. Reading, MA: Addison-Wesley, 1998.

Department of Information Technologies
Faculty of Mathematics and Informatics
University of Sofia
5, James Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: `aldi@fmi.uni-sofia.bg`
`sylvia@acad.bg`

Received October 25, 2006
Final Accepted September 13, 2007