Paper

UML Simulation of a Topology Configuration Model

Zbigniew Zieliński[a], Andrzej Stasiak[a], and Włodzimierz Dąbrowski[b]

[a] *Military University of Technology, Warszawa, Poland*
[b] *Warsaw University of Technology, Polish-Japanese Institute of Information Technology, Warsaw Poland*

Abstract—The article presents the application of simulation methods for topological models to analyze and design information systems. By using UML extensions and the UAL language it is possible not only to build a topological model for software, but also to perform efficient simulations of topological models. Additionally, it is possible to take into account the restrictive conditions stored in UAL and OCL languages. To execute the simulation the authors used an simulator from IBM. These concepts and methods are illustrated by examples.

Keywords—configuration topology, system modeling, UML model simulation.

1. Introduction

Methods for modeling information systems using UML is now widely used and can be regarded as a standard in modeling systems [1]. For some time, modeling in UML is complemented with the possibility of executing simulation models in real time [2]–[4]. The benefits of executing simulation models are undeniable. By carrying out simulation models it is possible to: better understand the dynamics of the modeled systems and processes, the early detection of errors in modeling the information system artifacts, the early validation of models to the specifications of the designed systems, the analysis of different variants of constructed solutions, the selection of the best solution in a given situation topology, and the detection and prevention of deadlocks and other undesirable factors in the processes of communication between system artifacts. The use of simulation is conducive to the usage of the UAL language (UML Action Language) to describe the models. Due to this approach a more precise formal description can be applied [2], [4], [5]. A more formal description of the action and semantics facilitates the automatic verification of the correctness of models. In the work [6], thanks to the integration of security models with the architecture models (described in UML) of a specialized system with multilevel security (MLS-type system) and simulation of these models, the verification of many security problems concerning the designed system was possible at the modeling stage. Among the recently developed project models there is a model of a system topology. In this paper a discussion will be conducted on the use of simulation methods for design artifacts expressed in UML, in particular, for topological models.

2. Topology Models

The concept of topology is understood as the type of model that illustrates the dependency between the resources of the modeled information system [7]–[9]. An important feature of the topological approach is the ability to plan and then verify the modeled deployment scenarios.
Planning the deployment architecture is a difficult process that can lead to the emergence of a large number of errors. The risk of error can be reduced by applying the following recommendations at the design stage [10], [11]:

- the introduction of the architecture deployment planning process at an early stage of manufacturing applications;
- the communication link between the structure and the architecture deployment by dividing and the re-use of architecture topology templates;
- the use of early deployment architecture validation scenario, and thus the identification of artifacts incompatibility of the model.

Deployment modeling is a bridge between modeling and application development. The topology model defines the elements of infrastructure and their location and dependencies between them.
Application of modeling using the topology shortens the life cycle of the application as follows:

- the application developer can check whether the application meets the requirements of the deployment;
- the architectural engineer can be sure that the established requirements of architecture deployment, the application can be run correctly;
- the analyst can implement best practices and company standards for architecture deployment and test scenarios to reproduce the proper architecture deployment.

By the concept of topology, we mean the deployment architecture model. This model is composed of artifacts called elements. A single element represents an application fragment or fragments of deployed infrastructure (including servers, server software, databases, operating systems) and

be linked with other elements. An element contains information in its structure on the requirements and limitations that must be fulfilled by it and the associated element.
Modeling deployment architecture involves the construction of models at several different levels of abstraction. These levels are as follows [7], [8], [12]:

– application structure level,

– logical model level,

– Physical model level,

• application deployment level.

3. Model Simulation

The UML action language (UAL) is a part of the standard maintained by the Object Management Group (OMG) which is an extension of the UML language standard, as a semantics language of the UML action language. UAL is used to construct executable models that are independent from the language, as well as platforms and tools. In earlier practices, as defined in the MDD (Model Driven Development), system designers create "executable code" using notation, a specific implementations language (Platform Specific Implementation (PSI), ex.: C, C++, Java, C#). In addition, developers have to track changes in each of them and tools for their implementation. Currently, the Object Management Group (OMG) developed a language of action semantics and created a standard fUML (Foundational Subset for Executable UML Models) – the fundamental subset for executable UML models and the Action Language for Foundational UML (Alf).
UAL is a subset of definitions defined in the Alf standard, which allows to define a complete system at a higher level of abstraction, and then allows to simulate the model (created in UAL), its debugging and code generation. With the standardization of UAL, the designers will not have to learn many new programming languages – till now related to specific instruments, and they are only limited to the ability to transform models stored in UAL. With UAL, system designers can build it, independently from the target technology platform (PSM and PSI), and the created model can be debugged, simulated, and then started in order to find and correct the errors. Developed models are also independent from the target platform implementation and allows for its determination through choosing the appropriate transformation (UAL2PSI).

4. Simulating Models

Currently, it is possible to simulate the behaviors described in UML models. All kinds of behaviors described in UML are namely supported: activity, interaction, state of the machines, and behaviors described in UAL. During simulation, the performed UML diagrams are animated – providing information, such as: another element to perform, elements already performed, the current position of the tokens, active states, etc. The traditional functions of the debugger are also available, such as traps, restart, suspend and renewal. One can also "inject defined events" in event-driven simulation models.
The model simulation can be used together with the configuration planning (Deployment Planning). It allows to visualize the execution of interaction, described in UML, between elements of the topology model – units of implementation. Communication between them will be animated and it is possible to visualize the history of communication with numbered arrows overlapping topology diagrams.
To be able to simulate models UML and topology should be equipped with Rational Software Architect with an additional package: Rational Software Architect Simulation Toolkit.
The solution gives the following benefits:

– enables early understanding of the system so that one can remove the potential drawbacks of their behavior (even during modeling);

– allows to understand how the behavior affects the static structure of the model after developing a diagram of the complex structure;

– allows to understand how the behavior affects the distribution and also to understand the potential impact of the available infrastructure of the built application;

– simulations can work on UML models further specified by the Action Language (UAL), if we assume building strict specifications; this means that simulations can be performed at a very early design stage; then one can try to eliminate any serious design errors and problems, particularly in relation to the availability of infrastructure and networks, as well as later, in order to identify logical errors in behavior.

The current version of the simulator allows:

– creating (stopping/restarting) a session of the executed model;

– selecting paths of implementation in model execution session;

– inserting traps in the model execution session;

– variables modification in the model execution session;

– directing the model execution session to a specific element of the UML diagram;

– execution of models written in UAL (currently it is a textual representation and not pictographic);

– running multiple model execution sessions;

– deleting events that are sent to the model execution session;

– reviewing the history of messages (e.g., for comparing scenario execution paths);

– selection of events and signals in the model execution session;

– animating topology models.

5. Simulation Environment

The IBM Rational Software Architect (RSA) environment is a set of integrated tools that support the manufacturing processes (design and the construction of the software), using the technique of modeling in UML [1], [10], [11]. Such an integrated application allows to unify all activities related to the mentioned stages of software engineering. Consolidating multiple functionalities in one tool allows to significantly increase the productivity, resulting in higher work efficiency of a team of analysts, designers, and programmers. RSA makes it possible to verify the architecture design, which allows for easier change management, contributing to raising the quality of the created software.
Within RSA the following modules can be specified [2], [3]:

- Rational Software Modeler (a tool for visual modeling and designing applications);
- Rational Web Developer (visual tool designed for web services and web application developers);
- Rational Aplication Developer for WebSphere Software (an integrated environment that allows to design, create, analyze, test, profile, and deploy web applications and portals, applications in Java and J2EE technologies, and applications that use Web Services);
- Rational Software Architect Simulation Toolkit – an additional package that provides functionality for simulation and animation of UML and topological models.

RSA is a tool that supports the Model Driven Development approach that is focused on the production of models and their transformations. Rational Software Architect enables the use of forward engineering methods (e.g., transformation from UML to C++) and reverse engineering methods (creation of UML models based on existing application code such as C++).

6. Example

As an example of simulation models we will consider a process fragment of manufacturing software support recruitment activities at one of the Polish universities on the first degree studies. The recruitment process involves the following entities:

- Candidate – the person applying for the right to study, who has successfully completed the matriculation examination;
- Maturity Exam Committee – (in our system) is the authority that certifies compliance of maturity exam results with those obtained by the candidate (the Regional Examination Commission forwards the results to the Central Examination Board);
- Recruitment Commission – is an institutional body, appointed by the rector who carries out the process of recruiting candidates. The commission announces the results of the recruitment process to the candidates;
- Recruitment Department – at the university, it performs the direct support of candidates, i.e., taking documents from candidates and issuing documents to students.

Prior to the recruitment process at each university, an algorithm is determined, according to which the Recruitment Commission (RC) will conduct the recruitment.
In our further considerations we will confine ourselves to the model of recruitment activities for undergraduate studies. With some simplification we can assume that this algorithm is defined as the weighted average of the results of the candidate's maturity exam from a set of subjects [i], taken at a level of [j] where $\mu_{i,j}$ denotes the weight

$$\text{number of points} = \sum_{i=1,j=1}^{m,n} (\mu_{i,j} * \text{result}_{i,j}) .$$

RC determines, on the basis of results introduced by the candidates, the threshold of eligibility for studying $\gamma[i,j]$ (independently for each faculty, (and/or) direction and type of study) – i.e., the minimum number of points entitling the candidate for enrollment, and then the list specifying the place (rank) of each candidate.
The modeling process begins with the creation of requirements for a system which for the need of the publication will be restricted to the presentation of the modeled functionality (Fig. 1).
The domain model, developed on the basis of the requirements model, allows the formulation of the rules on the

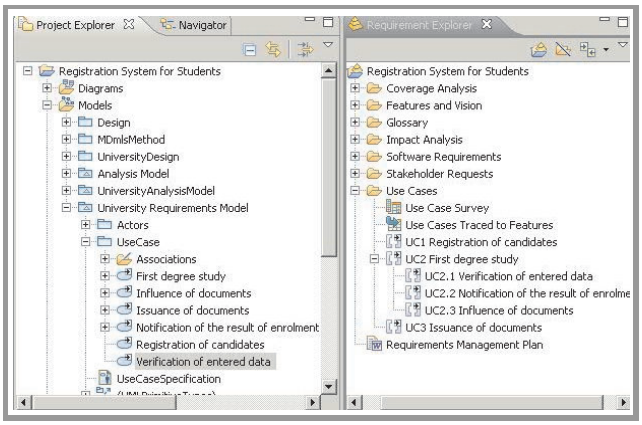


Fig. 1. Association of the verbal form with the visual form of the functional requirements.

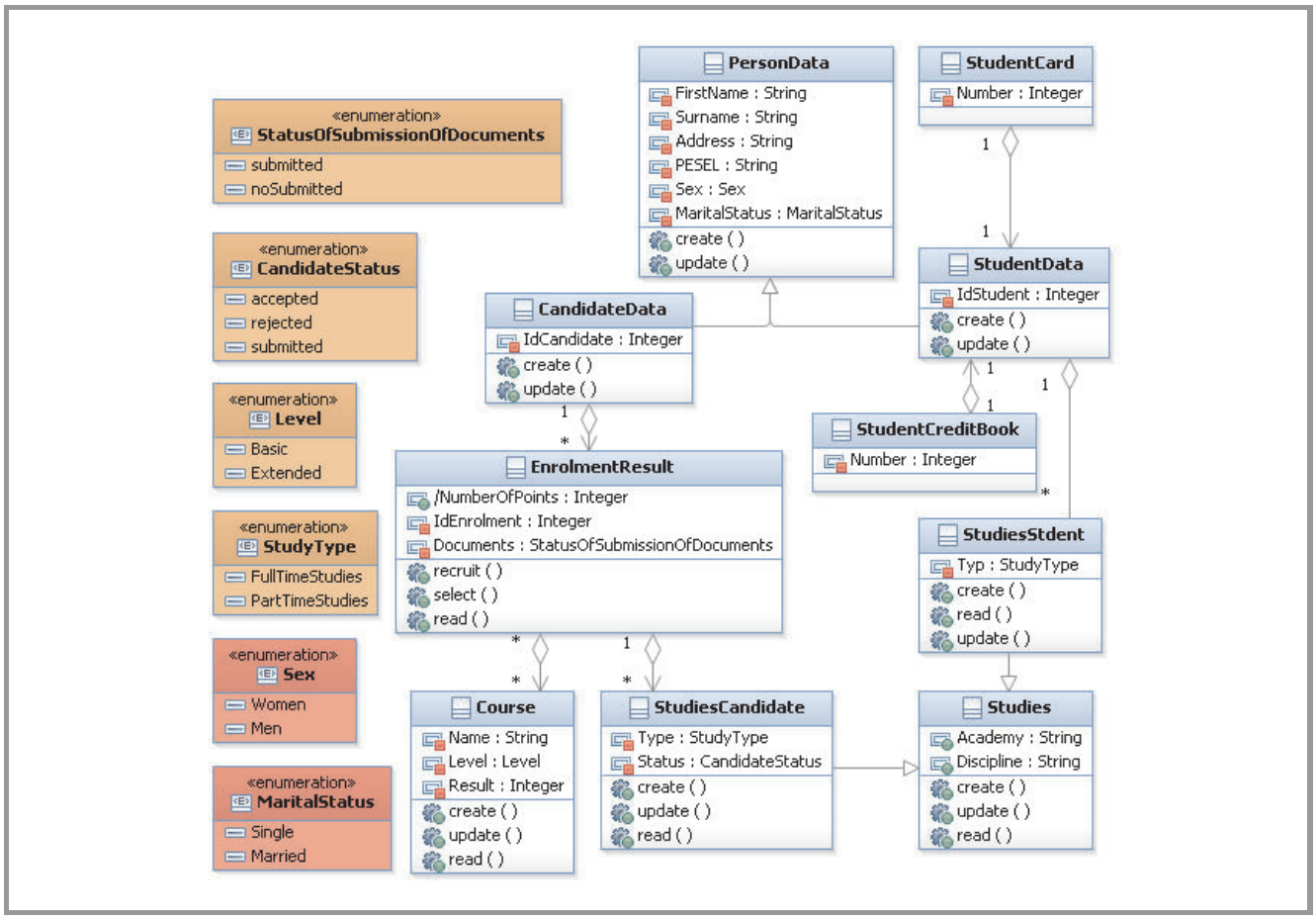


Fig. 2. An example of the domain model.

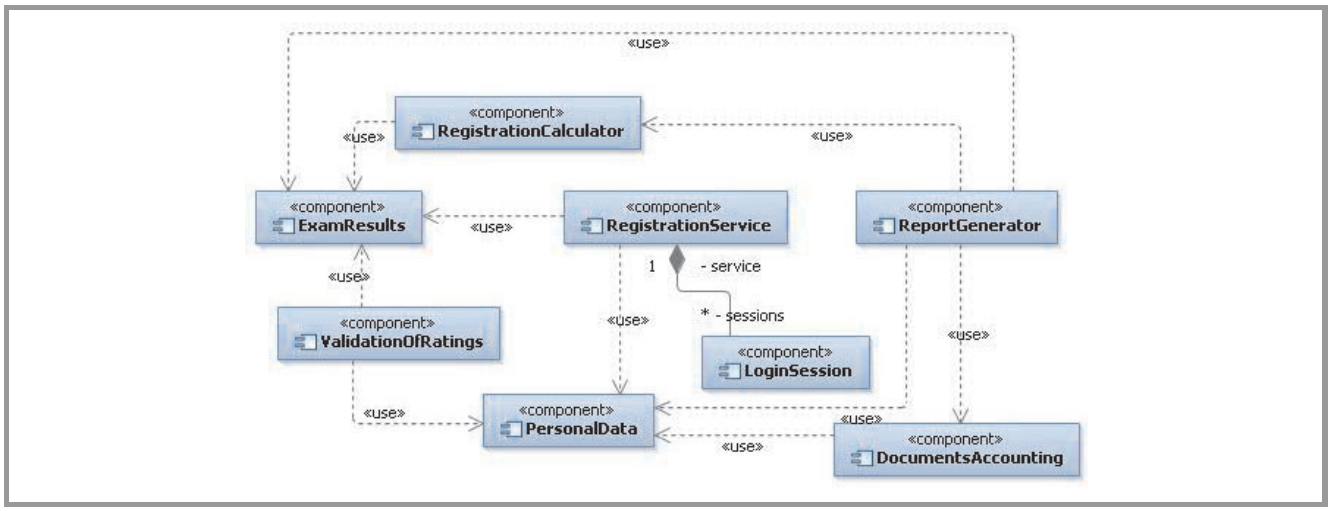


Fig. 3. The components model for the student registration system.

recruitment process, and it is shown in Fig. 2. However, not all rules can be described using UML models. For a more complete formalization of the model, it is necessary to use OCL and UAL languages.

An example of a service resulting from the requirements of the system is the authorization service. This service is defined by the LoginSession component (which will be appealed to all services that require authorized access to them).

For example, the student registration service (RegistrationService component) is related to many of their sessions (Fig. 3), for each user a separate object session, maintaining the state only when the user is logged in (Loggedin state (Fig. 4)).

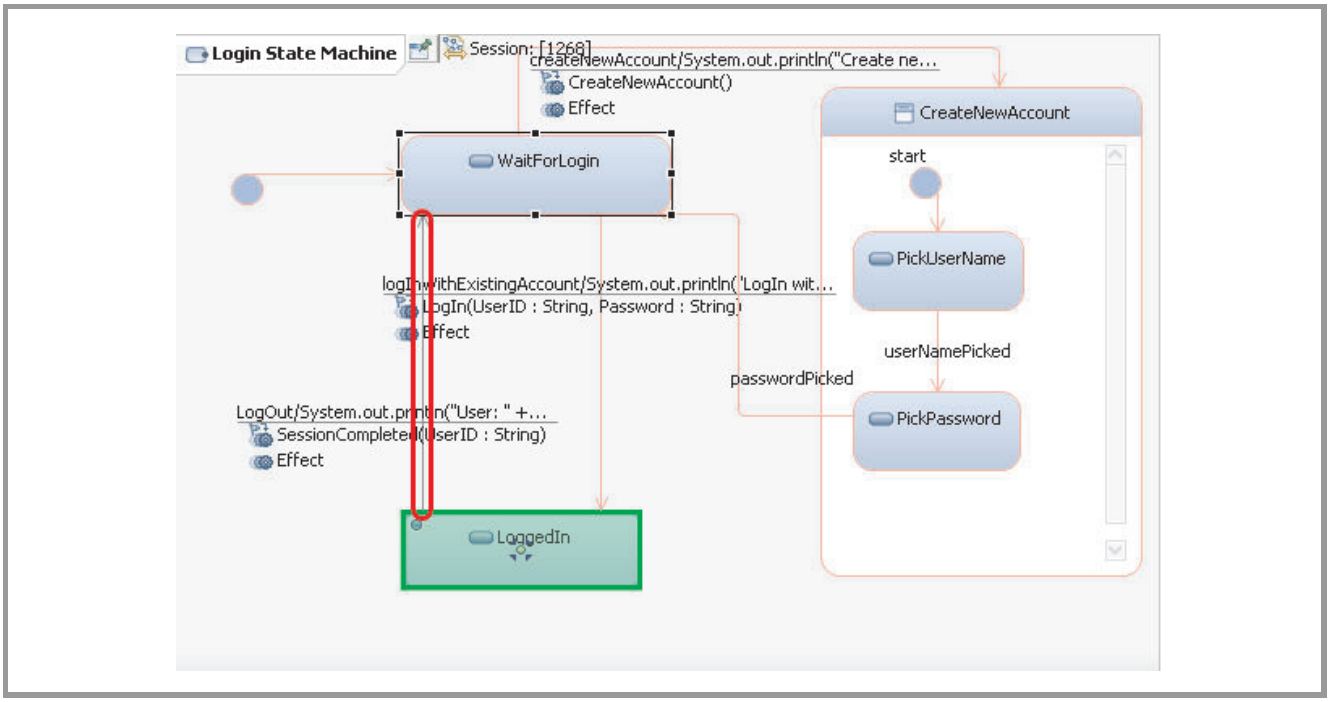


Fig. 4. Signaling the history of states (and transitions).

After changing the state forced by the sessionCompleted operation, the object session is destroyed, and the service goes into the WaitForLogin state.

An important part of the planned condition model simulation process is determination of the list of messages introduced into the console, which will be implemented in UAL. The code corresponding to the handling of subsequent messages is as follows:

- System.out.println("Create new user and yours password");
- System.out.println("LogIn with userId =" + msg.UserID + "and password =" + msg.Password);
- System.out.println("User:" + msg.UserID + "session is completed").

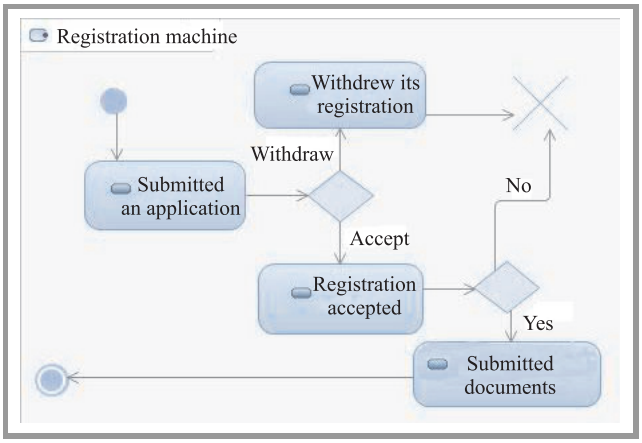


Fig. 5. The result of the machine state service animation – registration of students.

After taking into account the principles of building simulation models [2], [3], it is possible to obtain the signaling of history of states and transitions between them in the simulated model. An exemplary signaling of state history is shown in Fig. 4.

After changing the state forced by the sessionCompleted operation, the object session is destroyed, and the service goes into the WaitForLogin state.

Similar model test coverage can be observed by analyzing Fig. 5.

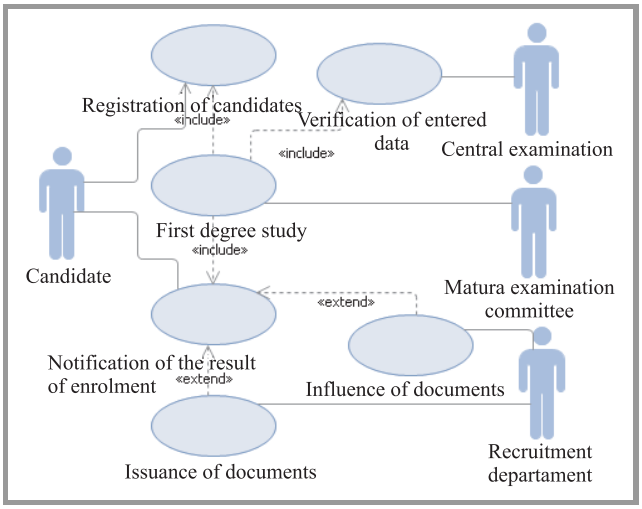


Fig. 6. A fragment of the use case model.

The service of registering candidates is an important functionality of the system being built (Fig. 6) and initiates the recruitment process, which is why the following part

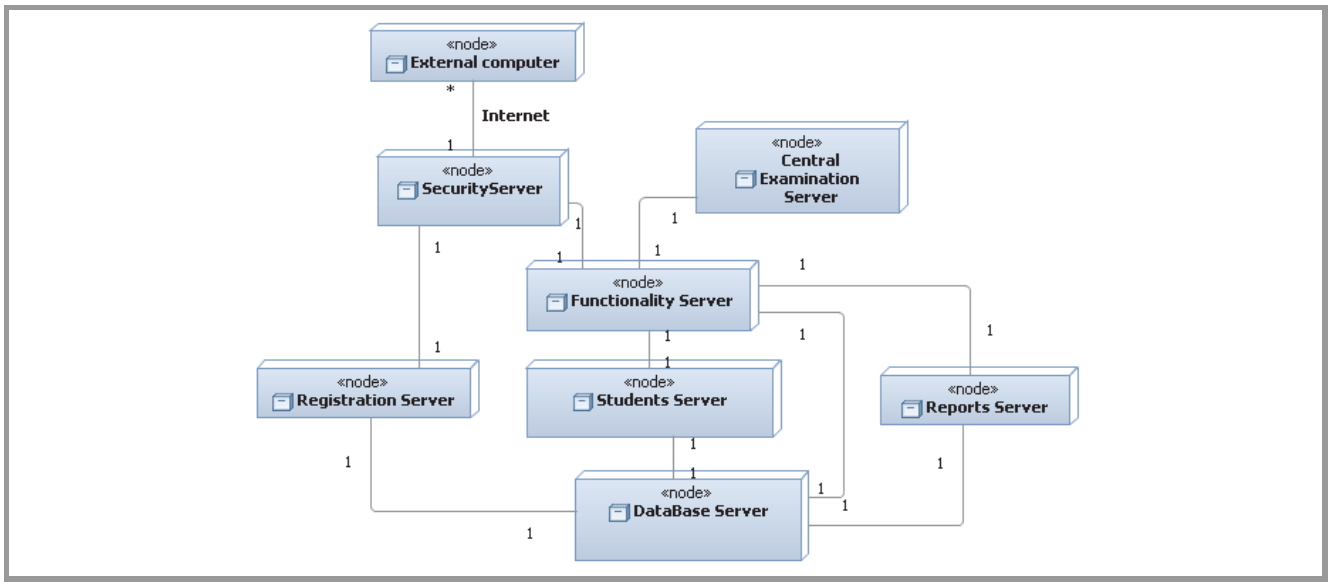


Fig. 7. The deployment diagram for the student registrations service.

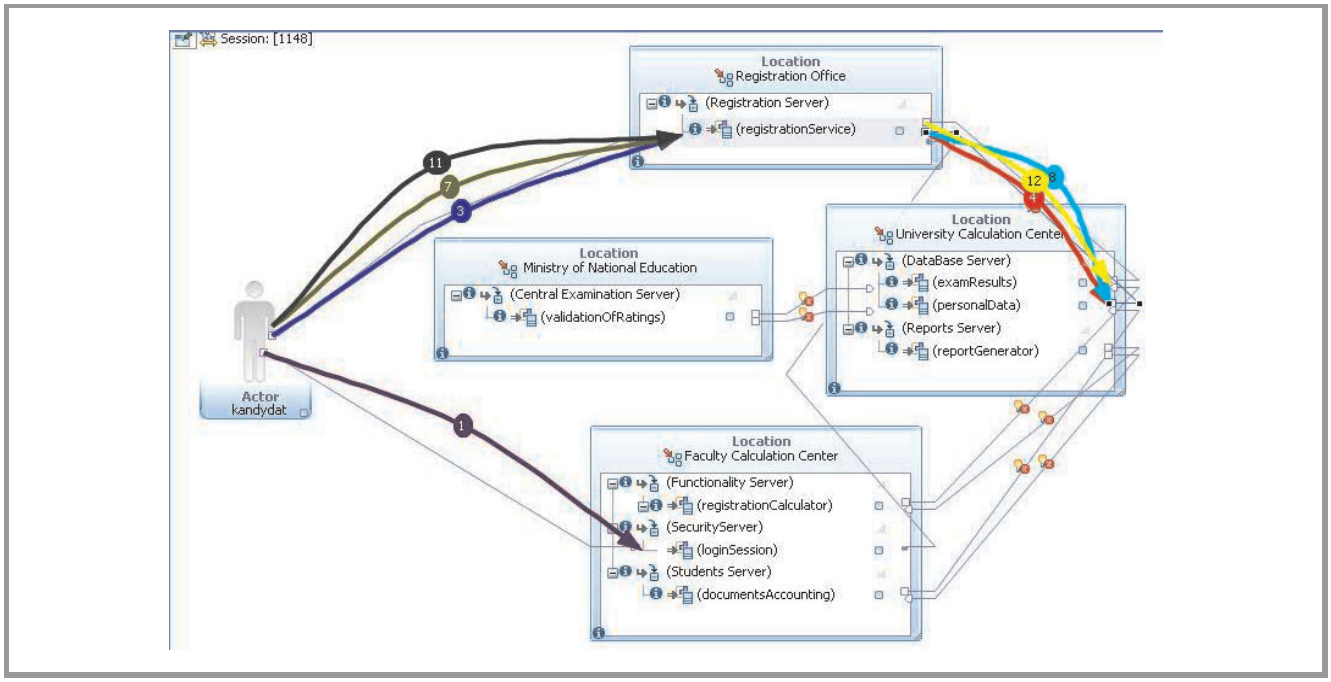


Fig. 8. The result of the state machine animation of the student registration service.

of this article will be limited to the presentation of its detail.

For the selected set of use cases (in the service of registering candidates) the deployment diagram was proposed (Fig. 7), which next has been transformed to the more detailed form of topology model.

It would be noticed that using UML language for deployment specification (Fig. 7) is not necessary, because a topology model extends a set of information concerning a system implementation as nodes, components, etc., and additional properties defining its location as hardware and software platforms.

Simulation model of the topology for the discussed fragment of the system, i.e., the candidate registration service is shown in the diagram (Fig. 8).

This model is a component of the system (Fig. 3) running on the nodes placed in specific (named) locations in the structure of the educational institution and relationships between them that define the flow path of messages.

The flows of messages are shown directly on the topology model in the form of numbered arrows.

It is worth noting here that during the process of animation, only those components of the system model from Fig. 3 participate, which are directly involved in the sce-

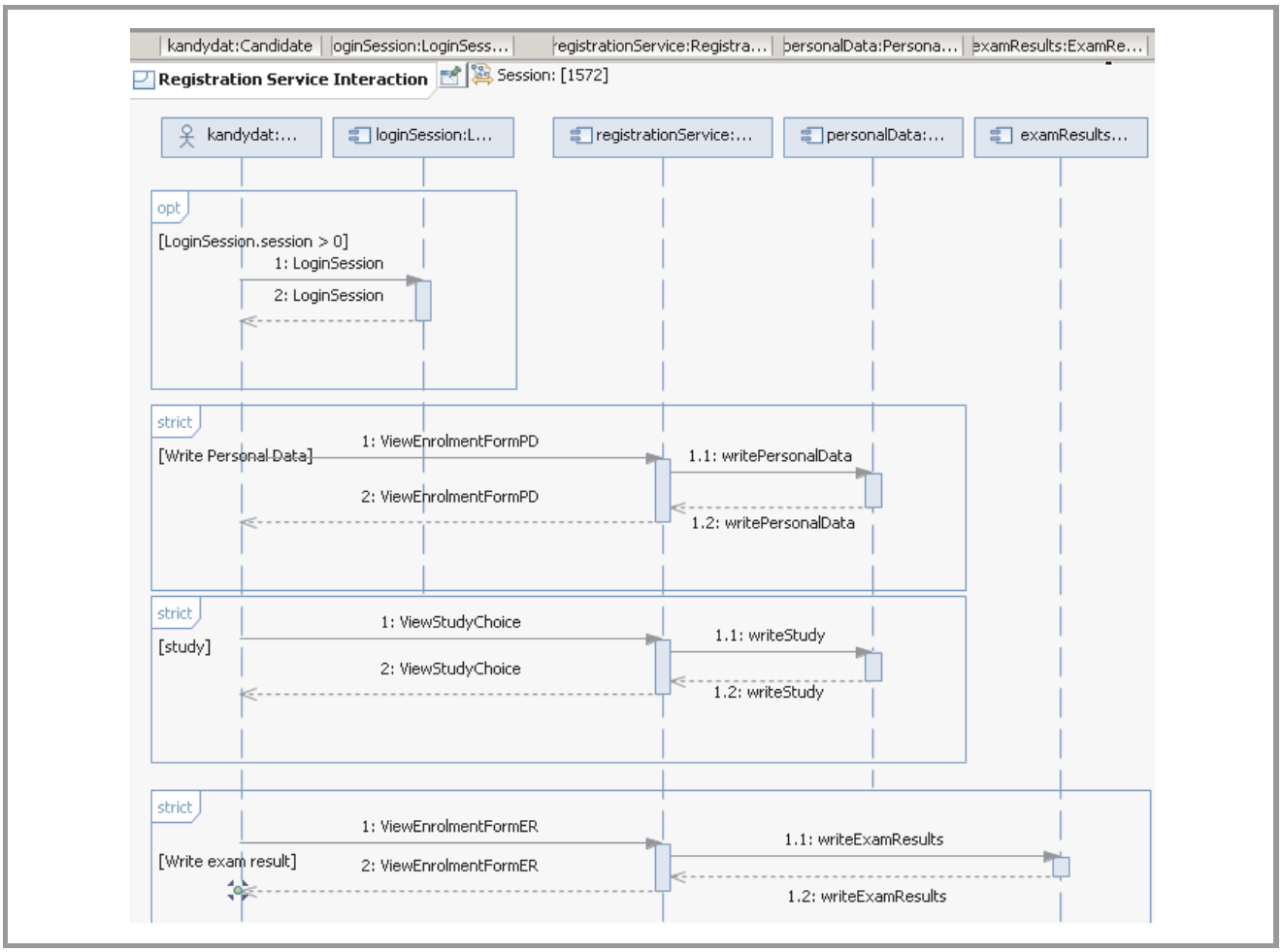


Fig. 9. Registration Service Interaction.

nario depicted in Fig. 9 (others are used in other scenarios).

The results show that the essential parameter of the process simulation is the behavioral diagram, which is to be animated. The animation can comprise not only on the presentation of the paths of starting the model (by indicating the place of the token, anticipating the next place, and highlighting the paths with a color, in which the token already was in), but we can also animate the messages flowing in from an established source to target.

7. Conclusion

This paper presents the essential capabilities of the simulation process of UML models, which affect the animation capabilities of topological models.

It is worth noting here that UML models significantly reduce the complexity of the topology modeling process, allowing to automatically identify links between elements of the topology model based on relationships defined in behavioral models, and the structure of the built system (relationships in the topology model from Fig. 7 were automatically created after assigning components to nodes from Fig. 3) working in their environment of which behavior is defined in Fig. 8. We can say that the relations in UML models provide guidance to their mappings (via the drag-and-drop method) on topology models.

Nowadays, the modern CASE tools (including Rational Software Architect used in this article) allow to change the elaborate approach to a transformational approach (according to the MDA concept), automating the software development processes that nevertheless requires the use of language semantics of actions to build executable UML models, which may provide an effective basis for building models of system implementation.

References

[1] W. Dąbrowski, A. Stasiak, and M. Wolski, *Modelowanie systemów informatycznych w języku UML 2.1*. Warszawa: PWN, 2007 (in Polish).

[2] M. Mohlin, "Model Simulation in Rational Software Architect: Simulating UML Models", IBM, 2010.

[3] M. Mohlin, "Model Simulation in Rational Software Architect: Communicating Models", IBM, 2010.

[4] E. Anders, "Model Simulation in Rational Software Architect: Activity Simulation", IBM, 2010.

[5] "OMG, Action Language for Foundational UML (Alf)" [Online]. Available: http://www.omg.org/spec/ALF/1.0

[6] Z. Zieliński, A. Stasiak, and W. Dąbrowski, "Zastosowanie metod symulacji modeli UML do analizy i projektowanie komputerowych systemów specjalizowanych przetwarzających danych z ochroną wielopoziomową", *Przegląd Elektrotechniczny*, no. 2, pp. 120–125, 2012 (in Polish).

[7] "Planning deployment with the topology editor", IBM Tutorial, 2008.

[8] "Modeling deployment topologies", IBM Tutorial, 2008.

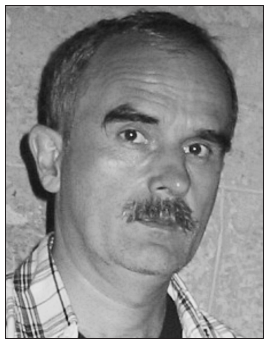
[9] S. Willard, *General topology*. Mineola, N.Y.: Dover Publications, 2004.

[10] M. Narinder, "Anatomy of a topology model used in IBM Rational Software Architect Version 7.5, Part 2: Advanced concepts", IBM, 2008

[11] M. Narinder, "Anatomy of a topology model in Rational Software Architect Version 7.5: Part 1: Deployment modeling", IBM, 2008.

[12] W. Dąbrowski, A. Stasiak, and K. Markowski, "Modelowanie systemów IT z wykorzystaniem topologii konfiguracji", *Przegląd Elektrotechniczny*, no. 9, pp. 239–242, 2010 (in Polish).

Zbigniew Zieliński received the M.Sc. in Computer Sciences from the Cybernetics Faculty of Military University of Technology, Warsaw, Poland in 1978, and the Ph.D. degree in Computer Systems in 1988. He is currently an Assistant Professor of Computer Systems in the Institute of Teleinformatics and Automation of Cybernetics Faculty, Military University of Technology. His current research interests are in the areas of computer systems dependability, processors network diagnosis methods, fault-tolerant systems, as well as virtualization and system security.
E-mail: zzielinski@wat.edu.pl
Faculty of Cybernetics
Military University of Technology
Gen. S. Kaliskiego st 2
00-908 Warsaw, Poland



Andrzej Stasiak is an expert in the field of design of information systems. From years he is a member of program committees of conferences on Software Engineering and Real Time Systems. From 1987 is an Assistant Professor of Computer Systems in the Institute of Teleinformatics and Automation of Cybernetics Faculty, Military University of Technology. He gained his professional experience directing some complex IT projects.
E-mail: astasiak@wat.edu.pl
Faculty of Cybernetics
Military University of Technology
Gen. S. Kaliskiego st 2
00-908 Warsaw, Poland

Włodzimierz Dąbrowski received the M.Sc. in Electrical Engineering from the Warsaw University of Technology, Warsaw, Poland in 1992, and the Ph.D. degree in Control Systems Theory from the Warsaw University of Technology in 1999. He is currently an Assistant Professor in the Institute of Control and Industrial Electronics, Faculty of Electrical Engineering, Warsaw University of Technology. From 2000 he is Assistant Professor in Polish-Japanese Institute of Informatics, Software Engineering Department, Warsaw. His current research interests are in the areas of design methods for computer systems, software engineering, and software project management.
E-mail: w.dabrowski@ee.pw.edu.pl
Faculty of Electrical Engineering
Warsaw University of Technology
Koszykowa st 75
00-662 Warsaw, Poland