ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 3

USING STATE COORDINATOR PATTERN FOR TRANSITION FROM DESIGN INDEPENDENT TO PLATFORM INDEPENDENT MODEL[1]

Lina Čeponienė, Lina Nemuraitė, Eugenijus Ambrazevičius

Kaunas University of Technology, Department of Information Systems
Studentų 50 - 308, LT-51368 Kaunas, Lithuania

Abstract. Development of the information system is a stepwise process, starting with requirements definition and proceeding with design and implementation steps. Existing modelling approaches concentrate on formalisation of requirements definition or start at the design model and analyse its transformation to code under specific platform. In this work, a possible method for deriving the design model by integrating requirements definition with chosen software architecture is presented. The proposed transformation based on State Coordinator pattern is a particular case of variety of possible transformations, but it demonstrates the way to formalize step from requirements to design and to shorten the development process.

Keywords: UML, OCL, Model Driven Architecture, information system, services, requirements, design.

1. Indroduction

Development of the Information System (IS) is a stepwise process, starting with requirements definition and proceeding with design and implementation steps. Existing modelling approaches concentrate on formalisation of requirements definition or start from the design model and analyse its transformation to code under specific platform. Going from requirements to design remains the most vague step in this process.

The principles of Model Driven Architecture (MDA) [11] are based on the idea that the foundation of software development is Platform Independent Model (PIM) of software system. PIM is the design model obtained from Business Model (formerly Computation Independent Model or CIM) and it serves for definition of a set of Platform Specific Models (PSM). From each PSM implementations on different platforms may be generated. MDA activities are concentrated on going from PIM to PSM and from PSM to code. By our view it is advisable to start from requirements definition and translate them to design model. Elaborated requirements model named Design Independent Model (DIM) was introduced in [6, 7]; here the idea of transformation from requirements specification to design model is presented.

There are informal proposals to introduce the architectural model and to join it with requirements for obtaining the design model (for example, [8]). Our proposal for deriving the design model from requirements definition integrated with chosen software architecture is more definitive: during architectural design elements of requirements specification are allocated to architectural elements defined by chosen design pattern. It results in platform independent design (PIM), and still more steps are needed to obtain the code but the last are straightforward and many implementations for code generation from PSM already have been made.

The proposed transformation is a particular case of variety of possible transformations but it demonstrates the way to formalize step from requirements to design. More specifically, the transformation is devoted for service-oriented design [16] of information systems using proposed State Coordinator pattern. This pattern was based on the idea of coordination of services on the base of information about persisted states of entities of problem domain. According its definition, service is an operation offered as an interface that stands alone in the model, without encapsulating state [10]; statelessness of service means that it is independent of context, and any client can use any instance of a particular service without regard to the history of instance. In reality, the use of services in information systems depends on states of information entities, which comprise service execution context. The State Coordinator pattern serves for loose connection of stateless services into system that operates on the base of

[1] The work is supported by Lithuanian State Science and Studies Foundation according to Eureka programme project „IT-Europe" (Reg. No 3473)

information about persisted states of entities. The choice of State Coordinator pattern was based on that fact that states machine is widely recognized as behavioural model for systems composed of services [3, 4]. By using states of entities, our approach differs from majority of proposed techniques where emphasis is made on modelling of business processes but information modelling is limited to definition of types of messages and variables[1], textual notes [5], or not considered at all.

The proposed design method consists of two steps – making comprehensive specification of requirements (Design Independent Model (DIM)) and applying the architectural pattern for going from requirements to design (PIM in MDA terminology). It is demonstrated, that DIM specification (based on UML, OCL [14, 15], and principles of contract-based design [9]) makes it possible to formalize transition from requirements to design and shorten development process.

2. Principles for Specifying Requirements

2.1. Textual Requirements Specification Template

Detail requirements model must define the overall state and behaviour of information systems independently of future design. But before creating the formal requirements model these requirements are described informally.

In this work, the template for initial textual description of requirements is used. This template is based on UML use case diagram and domain model of the system. The domain model serves as a glossary of terms that can be used in writing use cases. Use case model is used to capture the user requirements to the system by detailing all the actions that users can perform.

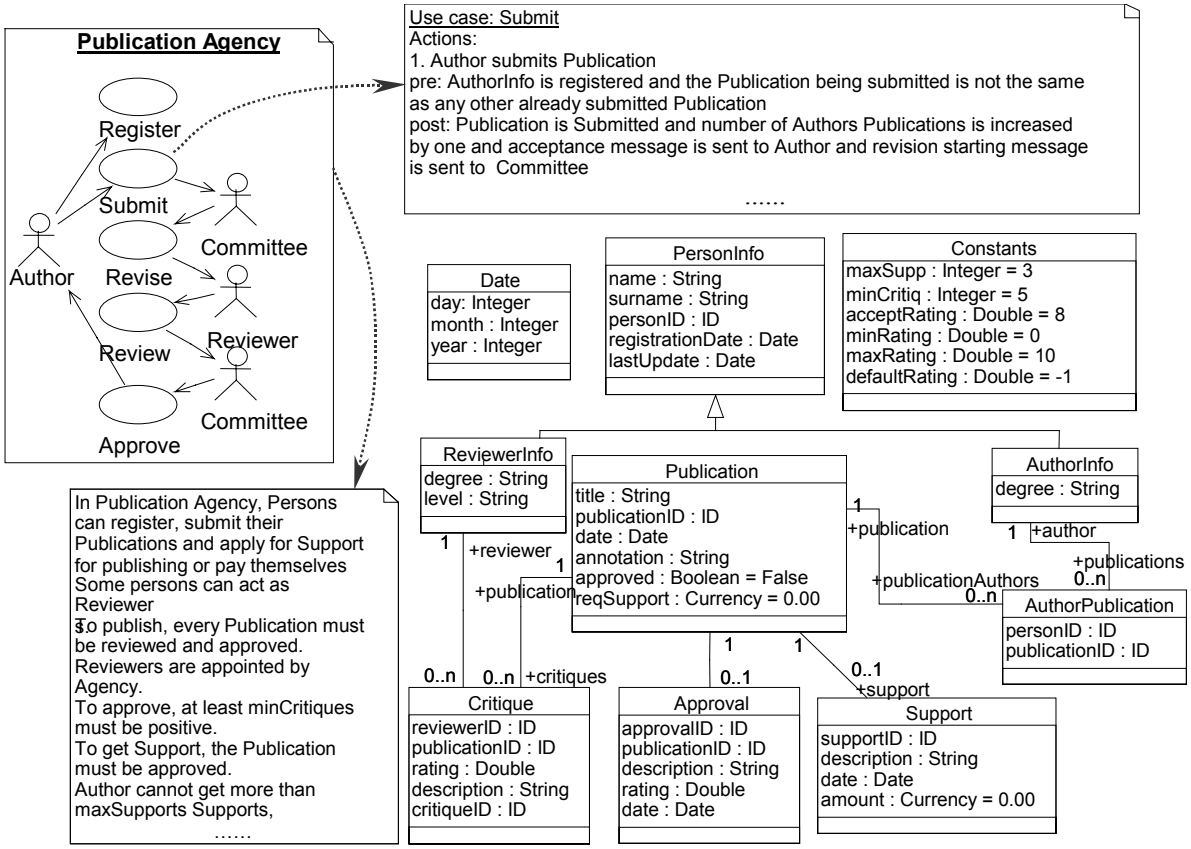

Figure 1. Fragments of use case and domain models of Publication Agency

Each action of the use case is described using pre and post conditions. As new objects are discovered during the use case description the domain model is updated. The example of applying the proposed requirements definition template for IS of Publications Agency is presented in Figure 1 where the models and fragments of description of problem domain and use cases are presented.

2.2. Design Independent Model

Behaviour of information system is tightly related with its state and has many aspects: interactions, state transitions, control flows, and data flows. These aspects are represented by different kinds of UML diagrams comprising views under the same model of target system. In this work, the possibility to achieve consistency of requirements model is based on adjusted subset of UML 2.0 meta model named Design Independent Model (DIM) [6, 7]. DIM represents

overall structure and behaviour of the system, but, differently from PIM, no design decisions should be made in it. In DIM, class, sequence and state diagrams are used, supported with OCL constraints; all kinds of diagrams are interrelated and may be derived from the other ones. DIM was devoted for development of services but it also may be applied for many software or information systems where explicit management of business processes is not required.

In DIM, use cases are mapped to interfaces of the target system, associated with actors – service users – and communicating with external systems (service providers), whose interfaces must be used by the target system to provide requested services. Interfaces are defined by the sets of operations identified from steps of use case specifications. Every operation is defined by its signature, pre and post conditions. Besides the interfaces, DIM includes entities and states of entities modelling the state of problem domain and sub-domains associated with particular interfaces.

For development of DIM, one or more sequence diagrams for every use case are constructed from specifications of use cases (Figure 2). These sequence diagrams must represent all desirable interactions between actors (service users), interfaces of the system and possibly the interfaces to external systems, if they are required to fulfil service requests.

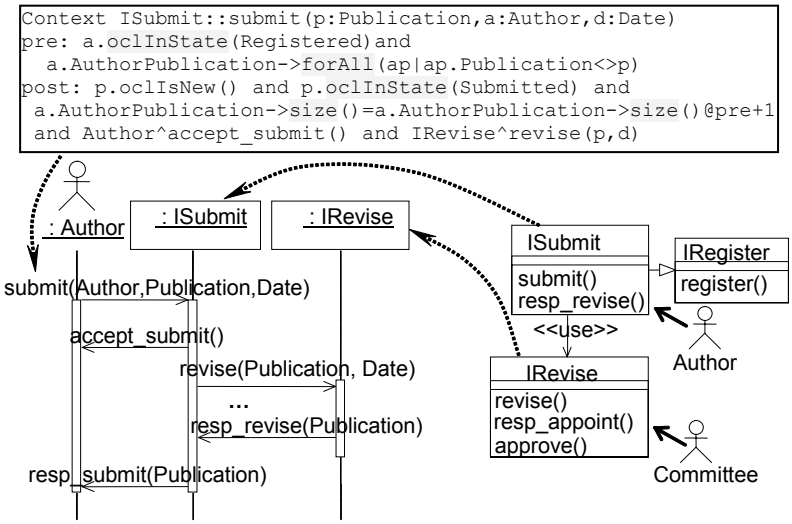

Figure 2. Publication submission sequence diagram and related interfaces in DIM of Publication Agency

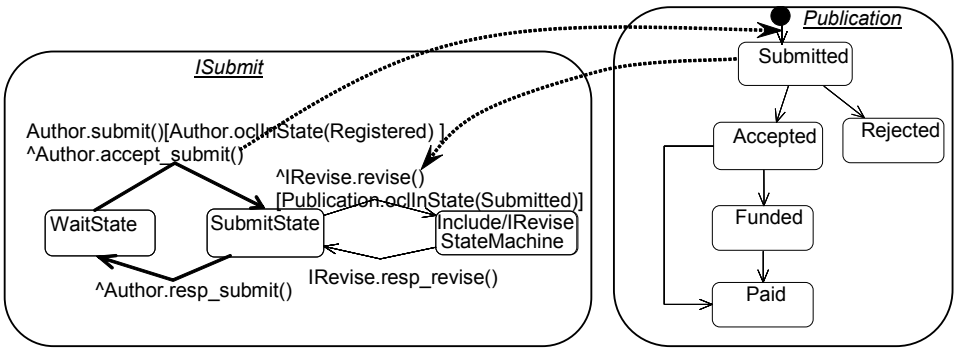

Figure 3. Fragment of state machines of Publication Agency demonstrating interrelationships between state machines of interfaces and state machines of entities

Sequence diagrams represent interaction related aspect of behavioural requirements. The "engine" of behaviour of object system is the state machine; semantics of functioning information system may be represented by transition system, affected by external events, where interactions between different actors and parts of the system take place and system moves from one compound state to another (Figure 3). If state machines would be derived from sequence diagrams, inconsistencies between interactions would be removed, as interactions of one classifier would be integrated from several sequence diagrams. It is also possible to begin from class diagrams or state charts and generate possible interactions for the purpose to revise and improve constraints or other elements of initial diagrams.

Class diagrams of interfaces and entities with respective constraints represent the final definition of requirements. It is the most informative view, from which design model may be obtained, but to develop this view comprehensively it is necessary to consider and integrate other views.

3. From Requirements to Design Models

In this section the step going from design independent model to design (PIM) is analyzed. Design model is different from requirements model in several aspects: design model should be supplemented with control classes or components; as result, sequence and state diagrams should be extended respectively; methods for operations must be elaborated (the last activity may remain in the responsibility of designer, or be automated). We consider the architectural design, during which elements of requirements specification are allocated to architectural elements.

For service-oriented design, the State Coordinator pattern (Figure 5) was constructed on the base of Facade and State patterns [2, 12, 13]). It is obvious, that for practical development of information system considerably more patterns should be used. In large systems, coordinator may be attached to every composite service. Coordinator handles incoming message and passes it to services according its actual context. It does it with assistance of Checker that checks preconditions and post-conditions of operations. Operations of services must be stateless so the information about states is captured by entities, and all constraints describing services subject to state changes are kept in Constraint base. There are many ways to proceed from requirements to design though we believe that it would be valid to use State Coordinator in design related with service-oriented architecture when business processes are not explicitly defined as e.g. in [1, 5], but managed using information about states of entities of problem domain.

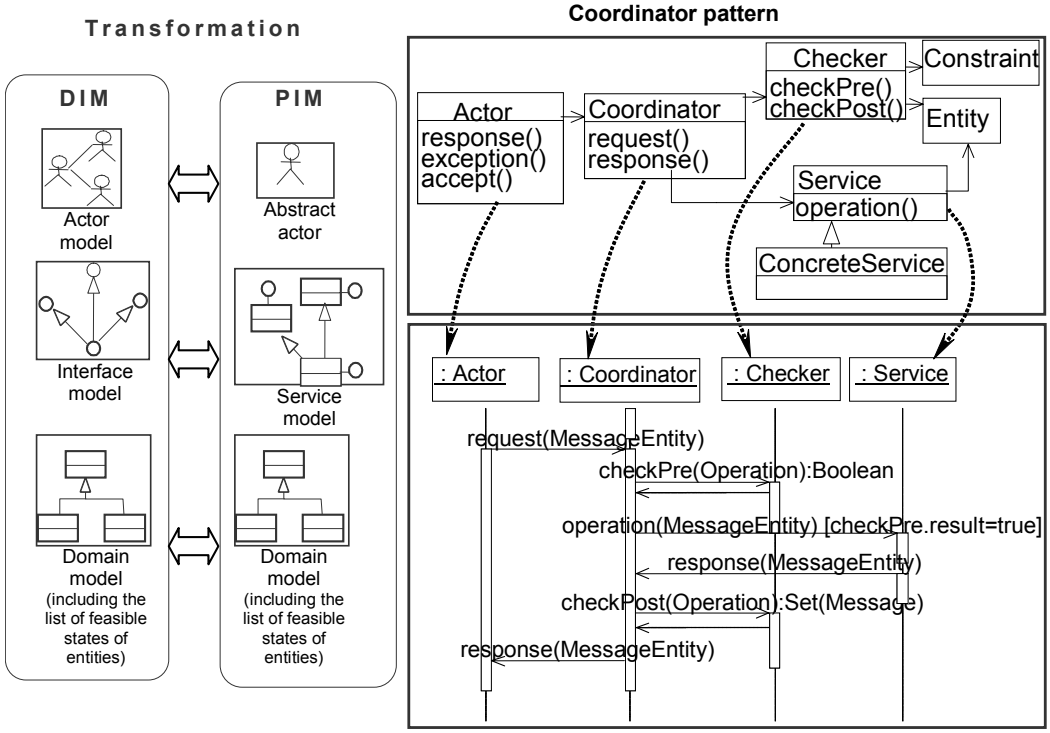

Figure 4. Principles of transition from DIM to PIM using State Coordinator pattern

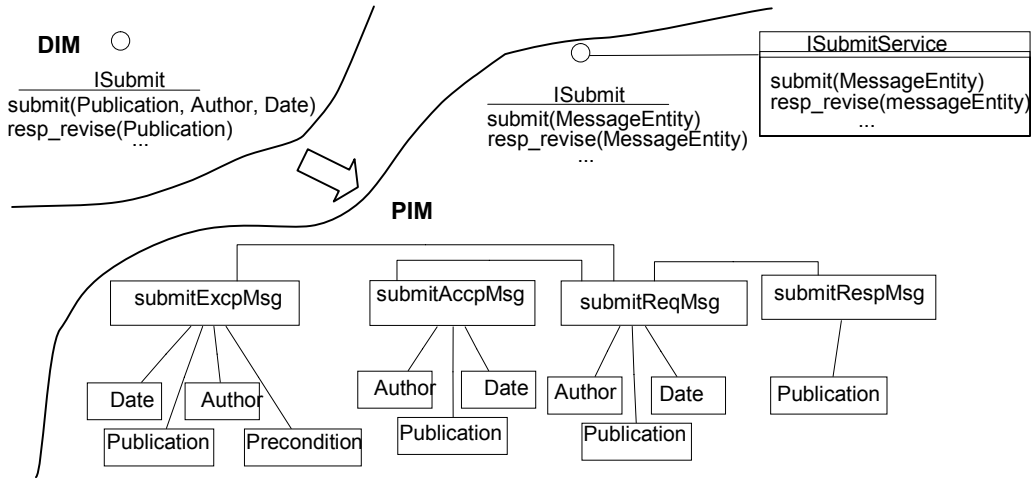

Figure 5. Example of the transformation of the publication submission interface and its operation

Transformation from DIM to PIM, based on State Coordination pattern, is presented in Figure 5. Transformations mainly are straightforward: DIM entities are represented as PIM entities, interfaces as interfaces; message entities (also called value objects) are created for request, response, acceptance and exception messages associated with checking of preconditions; preconditions and message expressions from post-conditions are allocated to constraints, and body conditions together with operation parameters are destined to services for oncoming detailed design.

Really, body conditions specified in comprehensive manner define all information necessary for generation of code of methods implementing operations of services. In Figure 6 the example of transformation from DIM to design model for a certain interface and its operation (publication submission) is presented. The interface is transformed into the service class and its interface. For the considered operation, request, response, acceptance and exception message entities are created.

In Figure 6, the sequence diagram of publication submission in PIM is presented. This diagram corresponds to the sequence diagram in DIM presented in Figure 3, but here the Coordinator and Checker classes are introduced that perform handling of incoming messages and checking their context; if preconditions of requested services are satisfied according information kept about states of information entities of the system, the acceptance messages are sent and services are delivered according contract, possibly in collaboration with other (internal or external) services, required for fulfilment of request (Submit Service uses Revise service as shown in Figure 6). If preconditions are unsatisfied, the exceptional messages are sent.

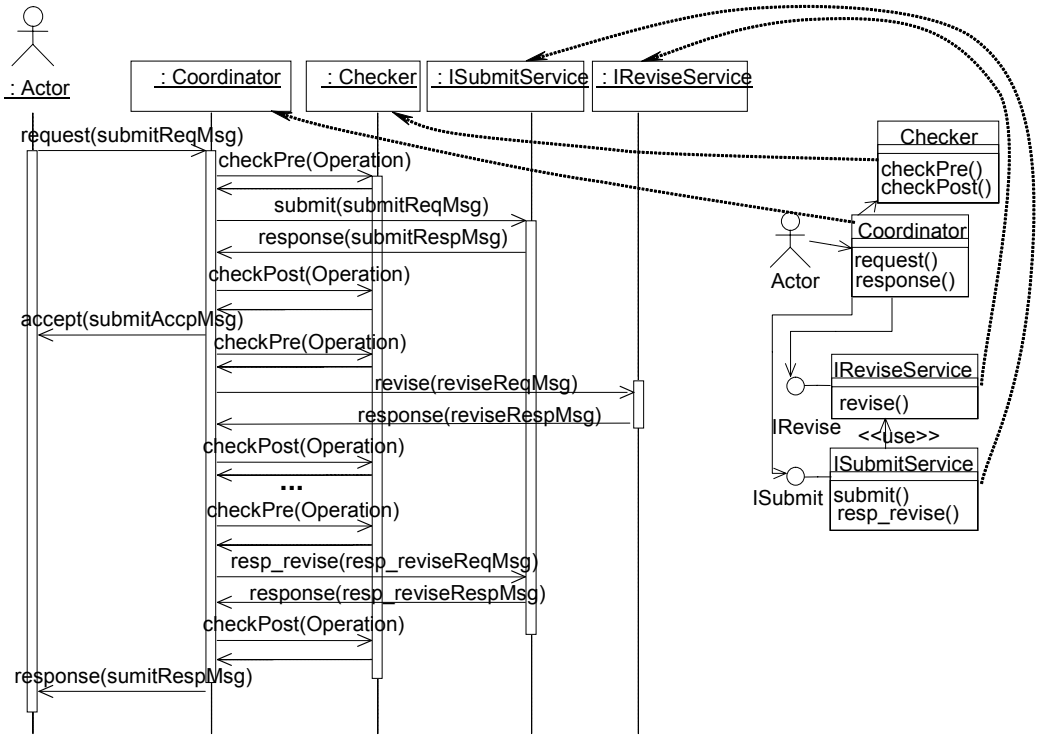

Figure 6. Publication submission sequence diagram and related interfaces in PIM of Publication Agency

Checking of post-conditions is devoted for unfolding the expressions of sending messages to other services. Message expressions enable to define compositional behaviour when the requested service uses other services: one or more messages may be sent in sequence, parallel or broadcasting mode when targets of messages are identified dynamically during the execution of requests. Really, the all required variety of interactions between services might be specified.

The abstract Checker may be implemented in different ways. In simple case, the pair of check operations might be created for every service introduced into system. In advanced case, it is possible to suppose the rule engine that is able to add, delete, read, check and transform operation constraints stored declaratively in the base.

In presented design some assumptions were made that may vary in different circumstances, for example, concrete naming scheme. Also, all preconditions here have textual descriptions, and exception messages are created by concatenation of negation of precondition and standard textual phrase. In practice, requirements for message entities may be predefined in requirements phase.

Development in MDA is associated with the suite of standards: UML, OCL, MOF, XMI, and working on the meta model level. The early prototype of UML CASE tool supporting Design Independent Modelling was implemented on the base of open source tool Argo UML. Today, the Eclipse platform seems the most promising environment for expansion of CASE tools having advanced possibilities for development in

MDA, but inconsistencies between large variety of versions and standards are discouraging. Currently, transformation from DIM to PIM using State Coordinator pattern is under trial implementation using Magic Draw API and JMI.

4. Conclusions

The main purpose of the work was to demonstrate that comprehensive definition of requirements of information system enables to obtain meaningful design in formal way and to shorten the development process.

Allocation of requirements to elements of architectural design is presented, during which elements defined in requirement specification are distributed to design elements following State Coordinator pattern proposed for service-oriented design of information systems.

Resulting design may be further subjected to detail design of operations and implementation in WSDL and web services framework. Proposed pattern is simple alternative for development of service-oriented information systems where explicit management of business processes is not required.

References

[1] **T. Andrews** et all. Business Process Execution Language for Web Services. *Version* 1.1, 2003. *Available at http://www-128.ibm.com/developerworks/library/specification/ws-bpel.*

[2] **B. Benatalah, M. Dumas, M.C. Fauvet, F.A. Rabhi, Q.Z. Sheng.** Overview of some patterns for architecting and managing composite web services. *ACM SIGecom Exchanges archive, Vol*.3, *Issue* 3, *Summer*, 2002, 9-16.

[3] **B. Benatallah, F. Casati, F. Toumani, R. Hamadi.** Conceptual Modeling of Web Service Conversations. *In: Goos, G., Hartmanis, J., van Leeuwen, J. (eds.): Proceedings of the* 15*th International Conference on Advanced Information Systems Engineering* (CAiSE'03), *LNCS Vol*.2681, *Springer Verlag, Klagenfurt, Austria*, 2003, 449-467.

[4] **D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella.** A foundational vision of e-services. *In: Goos, G., Hartmanis, J., van Leeuwen, J. (eds). Proc. of the CAiSE* 2003 *Workshop on Web Services, LNCS, Vol*.3095, 2003, 28-40.

[5] BPMN: Business Process Modelling Notation. *Version* 1.0 – *May* 3 2004, *BPMI. Available at http://www.bpmn.org.*

[6] **L. Ceponiene, L. Nemuraite.** Design Independent Modeling of Information Systems. *In: Barzdins, J., Caplinskas, A. (eds.) Databases and Information Systems. Selected Papers from the Sixth International Baltic Conference DB&IS*'2004. *Frontiers in Artificial Intelligence and Applications, Vol*. 118, *IOS Pres*., 2005, 224-237.

[7] **L. Ceponiene, L. Nemuraite, B. Paradauskas.** Design of schemas of state and behaviour for emerging information systems. *In: Thalheim, B., Fiedler, G. (eds.): Computer Science Reports*, Vol.14. *Brandenburg University of Technology at Cottbus*, 2003, 27–31.

[8] **A. Crain.** The simple artifacts of Analysis and Design. *The Rational Edge*, 8 *Jun* 2004. *Available at http://www-106.ibm.com.developerworks/ rational/library /4871.html.*

[9] **D.F. D'Souza, A.C. Wills.** Objects, Components, and Frameworks with UML. *The Catalysis Approach. Addison Wesley, Boston*, 1999.

[10] **E. Evans.** Domain Driven Design. Tackling complexity at the heart of software. *Addison-Wesley, Boston*, 2003.

[11] **A. Kleppe, J. Warmer, W. Bast.** MDA Explained: The Model Driven Architecture™: *Practice and Promise. Addison Wesley, Boston*, 2003.

[12] **I. Singh, S. Brydon, G. Murray, V. Ramachandran, T. Violleau, B. Stearns.** Designing Web Services with the J2EE™ 1.4 Platform JAX-RPC, SOAP, and XML Technologies. *Addison Wesley, Boston*, 2004.

[13] UML: Superstructure Specification. *Version* 2.0, *OMG document ptc*/03-08-02, 2003. *Available at http://www.omg.org*.

[14] UML: OCL. *Version* 2.0. *OMG document ptc*/03-08-08, 2003. *Available at http://www.omg.org*.

[15] **J.B. Warmer, A.G. Kleppe.** Object constraint language, The: Getting Your Models Ready for MDA. *Second Edition, Addison Wesley, Boston,* 2003.

[16] **O. Zimmerman, P. Krogdahl, C. Gee.** Elements of Service-Oriented Analysis and Design. *International Business Machines*, 6/16/2004. *Available at http://www-106.ibm.com/developerworks/library/ws-soad1*.