

A practical experience on model-driven heterogeneous systems integration

Marko Txopitea¹, Jorge Ibáñez¹, Antonio Estévez², José D. García², Javier Padrón², Carlos López², Beatriz Alustiza³, José L. Roda⁴

¹ Open Norte, S.L., Madariaga Etorbidea, 1 – 4. Ezkerra, 48014 Bilbao, España
{mtxopitea, jibanez}@opennorte.com

² Open Canarias, SL, Elías Ramos González, 4, ofc. 304, S/C de Tenerife, 38001 España
{aestevez, jdgarcia, jpadron, clp}@opencanarias.com

³ IZFE , S.A., Pinares Plaza, 1 – 4. solairua, 20001 Donostia – San Sebastián, España
balustiza@gipuzkoa.net

⁴ ULL, Escuela Técnica Superior de Ingeniería Informática,
Universidad de La Laguna, La Laguna, España
jlroda@ull.es

Abstract. This work describes our experience in a MDA based project for the integration of developments in heterogeneous systems carried out in IZFE (The Statutory Society for Information Technology, pertaining to the Statutory Department of Gipuzkoa-Spain). Starting from J2EE source code examples, we achieved an UML metamodel and a cartridge for a proprietary tool name BOA, that allows to generate automatically J2EE applications based on the IZFE development framework. Furthermore, these applications can interoperate in an integrated way with other legacy systems like FileNet content manager and CICS transaction server.

Keywords: MDD, MDA, UML, methodology, systems integration, legacy systems, J2EE.

1 Introduction

Open Norte together with Open Canarias has carried out an R&D&I project in IZFE (The Statutory Society for Information Technology, pertaining to the Statutory Department of Gipuzkoa-Spain), applying in a real corporative environment the Model-Driven Architecture (MDA) [1], an approach to the Model-Driven Software Development (MDS) [2].

The objective of this project was to start up a high productivity system for J2EE [3] applications development and their integration with legacy systems. More specifically, with the content manager FileNet [4] and with the transactional monitor CICS [5], as well as a relational database. The high productivity has been achieved thanks to the automatic code generation through the proprietary tool named BOA [6], starting from UML [7] models, following the scheme of the Figure 1.

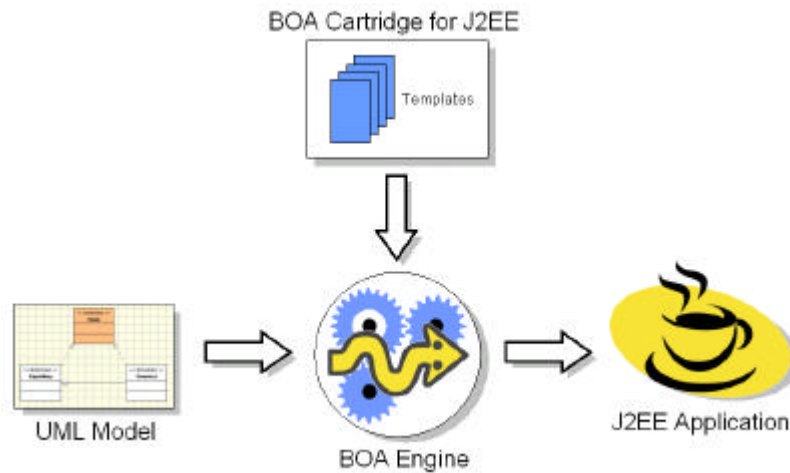


Figure 1. BOA use schema.

We choose BOA because this tool is developed by one of the companies part of the project (Open Canarias), so it gives us various benefits: we don't need to learn how it works; we don't need to pay any license; if needed we can make changes in the kernel of the engine to fully reach the goals of the project; it was suitable to carry out a project like we intended to.

2 Platforms Used in the Project

IZFE is responsible for maintaining an Information Technology network with a wide range of machines: from an IBM 2086 mainframe to more than 130 Windows, Unix and GNU/Linux servers. IZFE is responsible for more than 90 development projects a year and at the moment, it has more than 300 heterogeneous applications running and in a state of continuous evolution. The number of people working directly on these development projects has reached 165, apart from those collaborating within the closed environment of suppliers and providers.

As for the IZFE systems related to this project, we can point out WebSphere Application Server 5.1 for z/OS [8] and DB2 database server 7.1.0 for z/OS [9]. Apart from this technologies, as explained in the next section, the aim of this project has been to achieve an integrated infrastructure of model-driven software development for the IZFE Framework based on Struts [10] for J2EE web applications development, the content manager FileNet 3.0 for Windows and the transactional monitor CICS Transaction Server 2.3 for z/OS.

3 Used Methodology

Given the variety and complexity of the platforms to integrate, we decided to use a bottom-up methodology. Starting from representative J2EE code samples of each technology, we detected and generalized common aspects, and through them, we achieved to define the models and metamodels. Then, we proceeded to integrate the environments (see Figure 2).

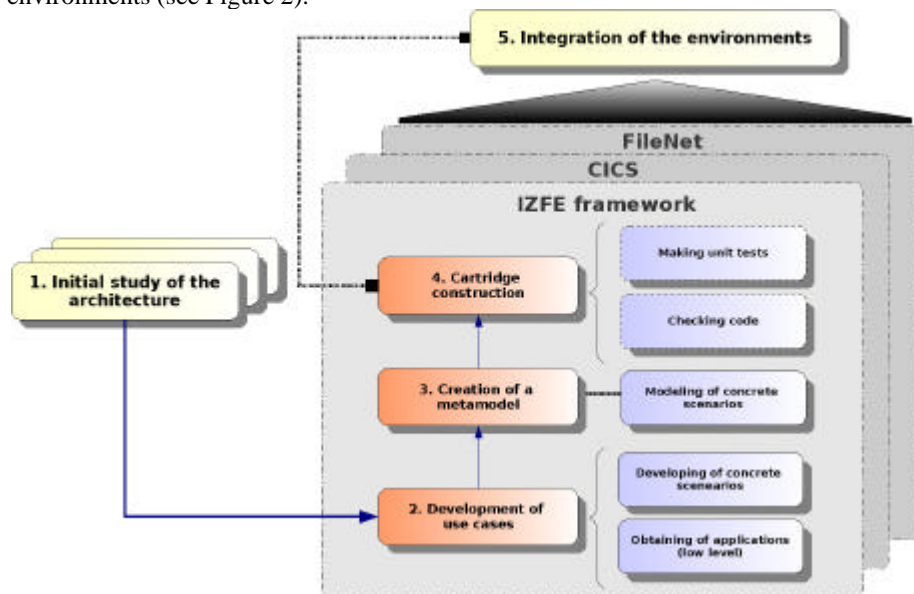


Figure 2. Arrangement of the tasks within the used methodology.

Task 1: Initial study of the technology and architecture. At this stage all the available information about the software architecture related to the technologies implied in this project is obtained and studied. All this information is checked, validated and developed. As far as possible, we try to make the study as near as possible to real life circumstances.

Task 2: Development of use cases. At this stage, our goal was to define the required functionalities from the analysis of the applications previously developed by IZFE. We defined the concrete scenarios to be resolved and the concrete functions to be included. Then we implemented a series of applications covering the most common tasks needed by IZFE for each platform. Once verified these programs in with those technologies that already exist, we started identifying structures and components susceptible to be considered separately. In this process new patterns and templates were developed.

Task 3: Creation of a metamodel. We have used UML Profiles [11] to define metamodels. UML profiles allow the customization and extension of the UML syntax and semantics to define specialized modeling languages for particular domains. The basic principle for defining each profile is to obtain generalizations between different

programming languages, platforms and technologies, as well as to incorporate other relevant aspects related to the integration of legacy systems and applications.

Task 4: Cartridge construction. Having defined the metamodel, we defined for each platform a complete model sample and we started to build the cartridge, whose function is to read the model (created from the corresponding UML profile) apply the needed templates and generate the source code.

We have used BOA [6] to build the cartridges. This tool is able to read the model, previously exported to the XMI [12] format, and parse the templates in XSLT format that give the Java source code as result. We achieved to generate the 100% of the source code for each technology. Once the cartridge is done, thanks to a plug-in for Eclipse, the user can easily generate new J2EE applications from the models (each of them belongs to the UML profile that the cartridge understands).

Task 5: Integration. The creation of cartridges and metamodels for each specific domain should provide enough information to define a shared and unified metamodel. Likewise, an important work on cartridges integration is required.

4 Project Development

The development of the project followed a sequence through different technological environments. The IZFE framework was the first, as it was already identified as a key factor for the success of the project, as well as for its high level of complexity. The IZFE framework is a J2EE framework which runs on a Websphere Application Server. This is a server which is used extensively in the development of corporative web applications. The second environment considered was a transactional manager, identified as a CICS environment. In this environment there existed inherited processes and logic at a corporative level with a high strategic value. The third environment dealt with the development of an in company file manager, which was FileNet environment. In this environment there was a great quantity of high critical content, used in some areas of IZFE.

Each environment had definite tasks applied to them, using the methodology previously explained. The final phase of the project was to make a big effort to integrate all the environments into a common integrated metamodel.

4.1 IZFE Framework

Task 1: Architecture study. IZFE framework is used for the creation of applications in a corporative business environment. It is based on the Struts framework version 1.1. IZFE framework is divided into a series of subsystems, with the listener, control and presentation subsystems being of the first importance as well as the business and the special security subsystems relevant in the corporative environment. Furthermore, this framework promotes the declarative development through its support of the parametrical definition of some behaviours and functions on applications.

Once guides and reference information had been studied, we replicate the IZFE framework into our own simulation framework to perform our studies.

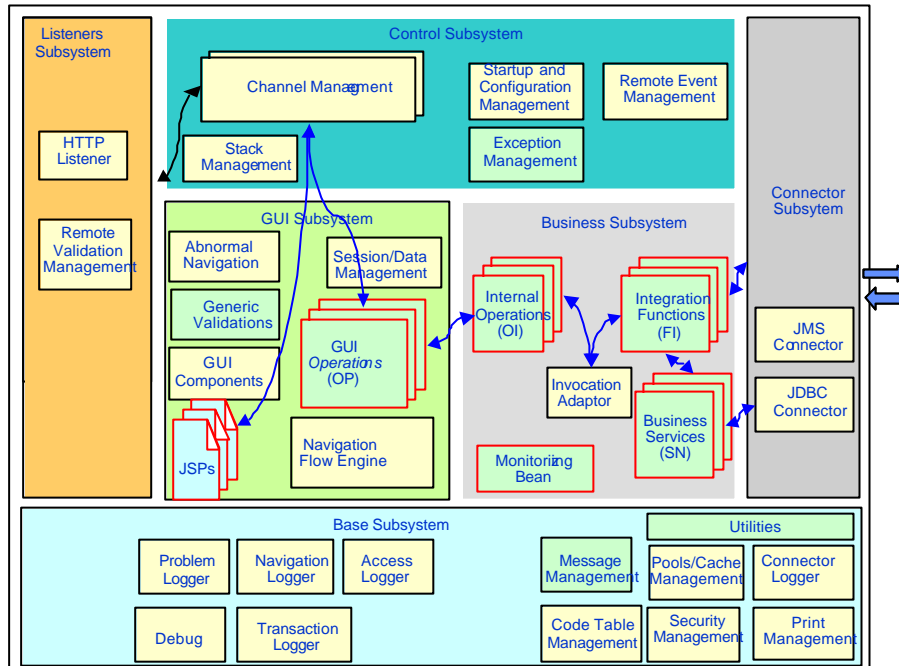


Figure 3. IZFE Framework Architecture.

Task 2: Use case development. We analyzed two sample application came with the framework. These applications were tested and run in our simulation framework. Having these applications as a reference, the requirements could be defined for a new application and reengineering techniques were used in its implementation. During this phase, unitary components were identified, which could be used as parametric components in the metamodel.

Task 3: Metamodel creation. The objective of the metamodel is to use a higher level of abstraction to describe the IZFE framework which retains the requirements of the IZFE framework. We described the framework following a MVC pattern [13], in which the domains are clearly defined and focused on the functions of self contained web applications. With this simplification we gain in level of reuse and correctness in the applications development process, as well as a better distribution of the work to be done by separating between different domains or aspects of the project, i.e. initialization, view, business logic, and persistence domains.

Task 4: Cartridge construction. Our goal is to get 100% automatic code generation, which results in a considerable rise in the problem complexity, above all in the definition of the business logic. In order to reach this objective, state diagrams were used, incorporating into these states action semantics [14] which are described in the

specifications 1.5 of the UML. To reach this approximation, Action Specification Language was employed (ASL) [15], and with certain modifications a grammar and a parser were developed, using a compiler from the SableCC [16] compiler. In this way a cartridge which generated 100% of the application code was built. Now, IZFE, instead of programming these applications directly, uses the metamodel defined in terms of a UML profile in order to represent their needs graphically. The system is capable of automatically generating applications from these diagrams.

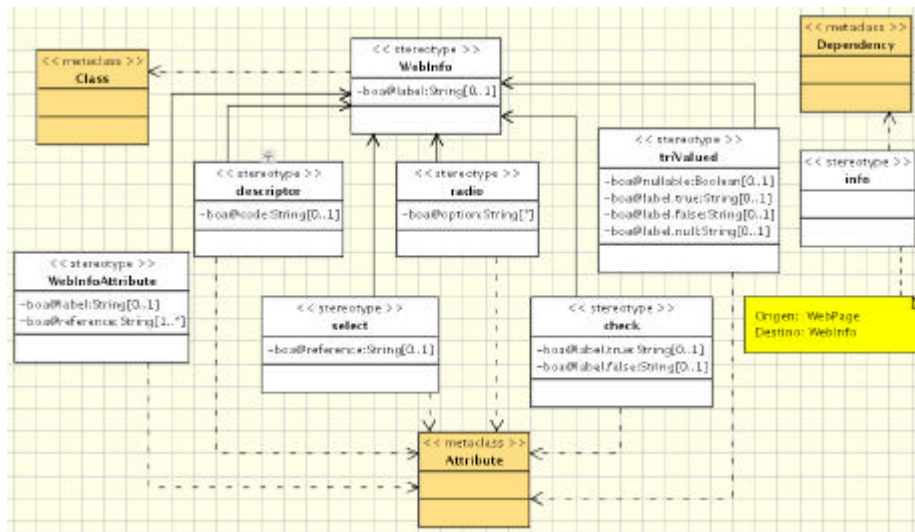


Figure 4. Sample piece of the IZFE framework metamodel to model web applications

4.2 CICS Environment

Task 1: Architecture study. The objective to be reached was the running of complex application components hosted in CICS through J2EE components. An exhaustive study of this area was needed as no similar development programs with these requirements have been developed before. Two key problems were identified: (1) the communication with the EIS (Enterprise Information system) and (2) the formatting of types between domains, i.e., J2EE and Cobol-based CISC applications.

Task 2: Use case development. JCA [17] was used to solve the communication problem. Numerous references, documents and examples of use of JCA libraries were consulted. A CICS ECI Resource Adapter was implemented and set up on a CTG (IBM CICS Transaction Gateway). The second problem identified in Task 1 was solved using the JRIO [18] library. Although this second problem was critical too, few examples and documentation were available. Finally were obtained the required minimum functions through unitary tests in order to validate the solution.

Task 3: Metamodel creation. The metamodel was developed by identifying the general functional components (i.e. application components and resource adapter) and parameterizing the information which is needed to customize them for each application. Use cases developed in the previous task (Task2) were used as reference.

Task 4: Cartridge construction. Finally, the cartridge was built. This cartridge contains a descriptor where the profiles are defined with each of the stereotypes assigned to the corresponding templates. Beforehand, the generated systems were checked. In this way, and using a generation engine, IZFE can describe the model graphically through simple UML diagrams, and generate 100% of the code needed for the connection to CISC programs.

4.3 FileNet Framework

Task 1: Architecture study. FileNet is a document manager and workflow tool with its own framework based on Struts. It has an API for J2EE which allows access to most of its functional components. IZFE has developed and maintains a simplified API which makes easier the running of the contents for the organization's internal uses.

Task 2: Use case development. Two samples from two different applications were selected which made use of the API of IZFE. Based on the provided source code and using inverse reengineering, we extracted the common functions from the real scenarios. Finally a series of unitary tests were made in IZFE's environment.

Task 3: Metamodel creation. The metamodel was developed identifying functional components susceptible to be generalized, and then parameterizing the minimum information needed to customize them for every specific application. All this was carried out using previously developed use cases. We defined a concept called repository that we merged with other elements in the integration task.

Task 4: Cartridge construction. For the construction of the cartridge each one of the stereotypes were mapped out to the units of generation. A template was defined for each unit of generation, which allows to the generation engine to create the source code. The use of a cartridge allows, through the definition of UML diagrams, for the 100% generation of an access code of the resource contents defined in the corresponding document manager.

4.4 Integration

The final task defined in Section 3 (Task5) corresponds to the integration of platforms using a single metamodel. We decided to conceive this metamodel as an extension of the IZFE framework metamodel, as this one is the most complex and extent of them, and the rest of platform domains are complementary to this one.

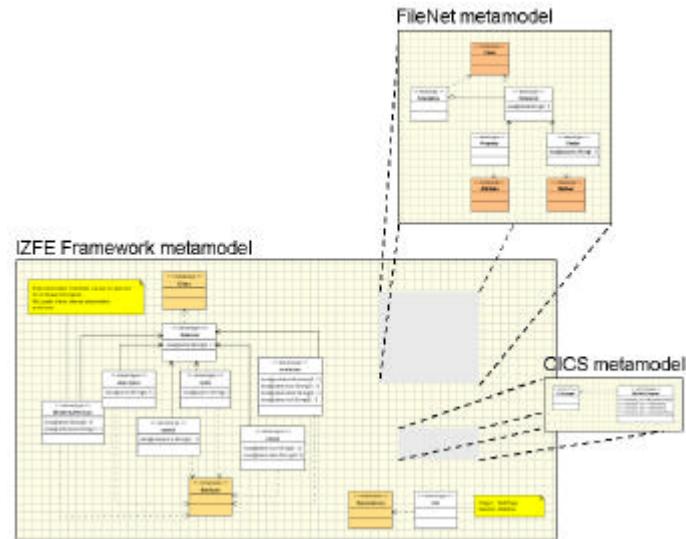


Figure 5. Unified metamodel.

We firstly carried out the integration of CICS with IZFE framework. In order to do so, CICS metamodel elements were incorporated into the IZFE framework metamodel, as well as new kinds of relations to allow the interaction with them. The resulting extended metamodel allows now the modelling of interaction between IZFE framework applications and CICS programs both in the presentation and business logic domain. In the first case, web forms are allowed to interact with CICS programs to retrieve information. In the second case, more complicated interactions between IZFE framework components and CICS business logic processes are allowed to be modelled using state diagrams.

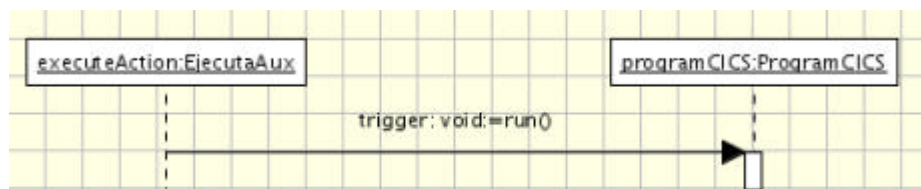


Figure 6. Sample sequence diagram showing CICS integration with IZFE framework in the view domain. In this interaction a web form triggers the execution of a CICS program.

The integration with FileNet was approached in a similar way. FileNet metamodel and cartridge were incorporated into the IZFE framework metamodel to be used along with it. This implies an enlargement of the target metamodel not only in the persistence domain (resource persistence) but also in the presentation domain. The integration was thought as a generic solution to maintain and manage resources, looking for a simply way of creating, editing and deleting resources from a content manager, in this case FileNet.

A two-step strategy was followed. Firstly, we included in the IZFE framework metamodel components specific to FileNet characteristics. Secondly, we generalized the possible interactions that should be done against FileNet and we raised the level of abstraction in order to describe them with a single generic component. A new stereotype was defined, which simplified even more the modelling and integration of FileNet interactions. This strategy was successful due to the low variability of the requirements to access FileNet resources.

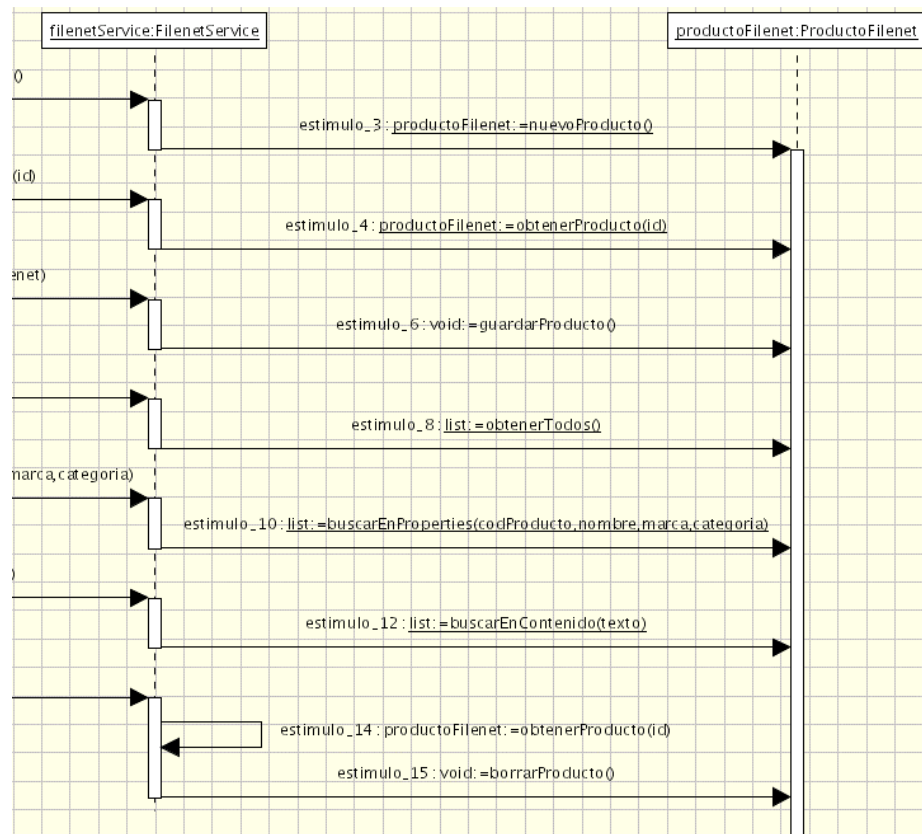


Figure 7. Sample sequence diagram showing FileNet integration in IZFE framework in the business domain.

Once obtained the integrated metamodel, we also integrated the cartridges to obtain a single set of templates able to generate the J2EE applications.

5 Obtained Results

The most important results we obtained in this project are:

- Integration of software developments for the target systems using MDA
It was the main goal of the project and it can be said that we certainly achieved to generate J2EE applications which use IZFE Framework and integrate with CICS and FileNet.
- Capability to generate the 100% of the source code of the applications
As the created infrastructure is able to generate complete J2EE applications, IZFE can focus its efforts developing the UML-based models. It has to be remembered that for the most complex business logic, we included the possibility to use Action Specification Language in the state diagrams. Obviously, all this means that the coding effort suffers an spectacular reduction (only needed to maintain the cartridge).
- From this point the models created by IZFE turn in corporate assets for the organization
As the metamodels have been defined abstract enough, the models are independent from the continuous technological change and evolution so all IZFE's models will persist through time (although the cartridge can need to be updated or replaced).
- Normalization of the systems through UML-based models
Thanks to an intensive use of UML before executing BOA to generate the source code, more effort can be invested in the UML design phase, so it is easier to rationalize the models and apply design patterns and anti-patterns.
- Shorter and easier requirement capture cycles through application prototypes
To perform a requirement capture for a web application, it is usual to prepare one or more prototypes of the program to show them to the users and ensure that all agree what the final application will do. Thanks to BOA and the cartridge created in this project, IZFE can generate approximated prototypes very quickly and speed up all the requirement capture cycle.
- Rise in the quality of the systems to be developed
As code is generated automatically, corrective, adaptative and perfective maintenances go more agile and less expensive in time terms. IZFE can change whatever it wants in the XSLT templates and then this changes can be automatically applied on all the existing occurrences of the applications' source files. This way, human coding errors are minimized and when they happen they just need to be corrected once.

6 References

1. MDA – Model Driven Architecture. <http://www.omg.org/mda/>
2. MDSO – Model-Driven Software Development. <http://www.mdsd.info>
3. J2EE – Java 2 Platform, Enterprise Edition. <http://java.sun.com/javaee/index.jsp>
4. FileNet P8 Platform.
http://www.filenet.com/English/Products/FileNet_P8_Platform/index.asp
5. CICS Transaction Server for z/OS. <http://www-306.ibm.com/software/htp/cics/tserver/>
6. Estévez, A., García, F., Padrón, J., Roda, J.L.: An MDA-Based Framework to Achieve High Productivity in Software Development. Software Engineering and Applications, Track 436-218 (2004)
7. UML – Unified Modeling Language. <http://www.uml.org>
8. WebSphere Application Server for z/OS.
http://www-306.ibm.com/software/webserver/appserv/zos_os390/
9. DB2 database server for z/OS. <http://www-306.ibm.com/software/data/db2/zos/index.html>
10. Struts Framework. <http://struts.apache.org>
11. Fuentes-Fernández, L. and Vallecillo-Moreno, A. An Introduction to UML profiles. UPGRADE, The European Journal for the Informatics Professional, 5(2): pp 5-13. April 2004. ISSN: 1684-5285.
12. XMI – XML Metadata Interchange.
<http://www.omg.org/technology/documents/formal/xmi.htm>
13. MVC – Model View Controller pattern. <http://java.sun.com/blueprints/patterns/MVC-detailed.html>
14. Action Semantics Revised Final Submission. OMG document ad/01-08-04.SL.
15. ASL – The Action Specification Language Reference Manual. <http://www.kc.com>
16. SableCC Parser generator. <http://sablecc.org>
17. JCA – J2EE Connector Architecture. <http://java.sun.com/j2ee/connector/>
18. JRIO – Java Record I/O. <http://www-03.ibm.com/servers/eserver/zseries/software/java/jrio/overview.html>