Applying MDA in Developing Intermediary Service for Data Retrieval

Danijela Boberić Krstićev
University of Novi Sad
Faculty of Sciences
Trg Dositeja Obradovića 4, Novi Sad
Serbia
+381214852873
dboberic@uns.ac.rs

ABSTRACT

In this paper, service for data retrieval from existing library management system is described. This service intermediates between library management system which provides data and system which requires that data. The main idea is that this service should support various protocols for data retrieval. Also, this service should be flexible for future update and simple enough for integration into any library management system. Service presented in this paper is developed by using Model Driven Architecture (MDA) approach. Different models (proposed by MDA) of this service are presented in this paper. Models are presented by using UML 2.0 specification. Transformations from models to Java programming code are done by using AndroMDA framework.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures – *Data abstraction*

General Terms

Design, Standardization

Keywords

Model-driven architecture, web services, AndroMDA, information retrieval

1. INTRODUCTION

Model-driven architecture (MDA) is a software design approach for the development of software systems. The aim of MDA is to enhance interoperability, portability and productivity of computing systems by means of abstract models [9]. Using MDA MDA it is possible to create design of system which is independent of its implementation. As result of using MDA approach we get models that become highly reusable assets.

Models play a major role in MDA and main idea of MDA is to create different models at different levels of abstraction. Some of models may be independent of software platforms, while others will be specific to particular platforms [13]. Precisely, the Computational Independent Model (*CIM*), the *Platform Independent Model* (*PIM*) and the *Platform Specific Model* (*PSM*) are the three model types that have been largely adopted by the software community. For every development life cycle phase of an application, MDA suggests to elaborate a corresponding model through a well-defined notation.

MDA separates the business logic from the complexity of the execution platforms. This separation is done through the efficient use of models in the software development process where enterprise architectures are supported by automated model transformations. In order to represent abstract view of the system various modelling standards (Unified Modelling Language (UML) [17], the Meta-Object Facility (MOF)[14] and the XML Metadata Interchange (XMI)[19]) are used.

The Model-Driven Architecture has been proved as a very promising approach to accelerate the software development and there are many papers describing usage of MDA in software development. MDA approach can be applied in any phase of software development. For instance, in the paper [7] MDA is applied to create graphical user interface (GUI) of the Amazon Integration and Cooperation Project for modernization of hydrological monitoring. In that paper, GUI source code is generated from UML models by using the AndroMDA framework [1]. Also, MDA approach is used in the process of building a healthcare enterprise information system on Java 2 enterprise edition (J2EE) platform. The process of developing such system is described in the paper [15]. In the paper [8], a method that uses the model transformation technology of MDA to generate unit test cases from a platform-independent model of the system is presented. The main idea of that method is to first create UML sequence diagrams which will be transformed into a general unit test case model and then transformations are applied on that general model to generate platform specific (JUnit, SUnit etc.) test cases that are concrete and executable. Moreover, using MDA approach provides benefit to the development of system which solves its problems with cooperative organization consisting of several heterogeneous computing components and in the paper [11] is proposed the development process of the multi-agent community computing system using MDA approach.

In this paper, different models and the standards behind the MDA are introduced. These concepts are applied in the developing service for bibliographic record retrieval, following the AndroMDA methodology to demonstrate how logic implementation can be reduced to the minimum when core business functionality is addressed during the modelling phase. The presentation of this paper proceeds in five sections. At the beginning of the paper, we described main functionalities of service for bibliographic record retrieval. Those functionalities are presented through computation independent model. After that, we continued by presenting platform independent model of service for record retrieval. In the fourth Section we discussed programming code which is generated by AndroMDA framework.

BCI'12, September 16–20, 2012, Novi Sad, Serbia.

Local Proceedings also appeared in ISBN 978-86-7031-200-5, Faculty of Sciences, University of Novi Sad.

At the end of the paper, we gave brief summary of the paper's main points.

2. COMPUTATION INDEPENDENT MODEL

In order to build a large-scale software it is necessary that all requirements are clearly defined, so the OMG recommends defining a computation independent model (CIM) in order to represent the application in its global environment. The CIM model describes what the system is expected to do, but hides all details of the system's structure to remain independent of how that system will be implemented. In the field of software engineering, CIM is well known as a domain model created by domain experts. CIM model may describe system functionalities through application of use case and activity diagrams and the actors that interact with the system. The CIM model should act as a connector between those that are experts about the domain and those that are experts of the design and construction.

CIM model of service for bibliographic record retrieval is presented in the figure 1.This service should act as an intermediary between search protocols and existing library system. The main aim of this service is to enable search and retrieval of data from various libraries using different standard protocols for search and retrieval. Currently, the most known protocols for searching bibliographic data are Z39.50 [5] and SRU [16] protocols. Those protocols should provide interoperable communication between different library management systems. Namely, one system may search and retrieve data from other system without knowing software architecture of that other system. It is only necessary that both systems have implementation of client and server side of appropriate protocol. There are ready-to-use open source solutions for client and server side of Z39.50 and SRU protocols and they just need to be integrated in an existing library management system. In the figure 1, a server side of Z39.50 and SRU protocols are presented as actors on the use case diagram. Integration of client side of protocol should not be a difficult, but implementation of server side of protocol requires much more effort. In order to integrate server side of a protocol it is necessary to process massage which is sent through that protocol. A massage contains query defined by query language supported by the protocol and that query language is independent of underlying system, so it is necessary to transform that query into query which could be understood by underlying system. Also, in order to provide interoperability it is necessary that data are exchanged in some standard form, so system which implements server side of some protocol must supply mechanism for transformation data in appropriate form. If system would like to support communication through different protocols, then those tasks must be done for each protocol.

The main aim of service presented in this paper is to enable simpler implementation of various protocols for data retrieval. Namely, service should support search and retrieve of data no matter which protocol is used. This functionality is presented by the use case *Find records* in the figure 1. Furthermore, this service will accept massages from different protocols and transform queries into CQL query language [3] (use case *Transform to CQL query*). After query transformation, query will be sent to appropriate components (wrappers) which are responsible for transformation of CQL query into a query language supported by underlying system. For instance, wrapper may be responsible for transformation of CQL language into SQL language in accordance with architecture of database of underlying system.

This task is represented with use case Delegate CQL to wrapper. Every underlying system should have its own wrapper, and wrapper is not part of service presented in this paper. After executing query and retrieving data from library management system, the final task of this service will be to transform retrieved records into form supported by protocol through which query is sent (use case Transform record).

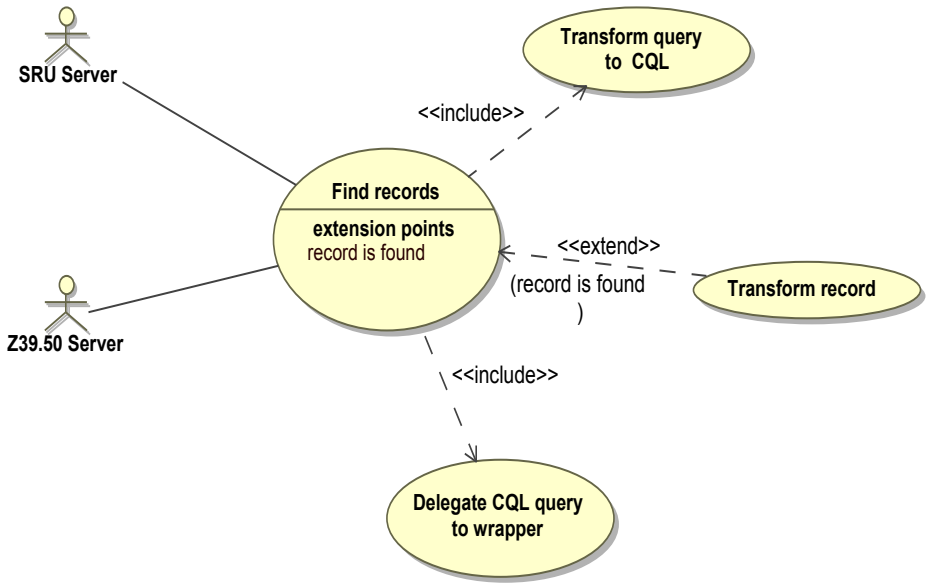

Figure 1: Computation independent model of service for bibliographic records retrieval.

3. PLATFORM INDEPENDENT MODEL

After we got the CIM model of software, the next step in software development using MDA approach is to create platform independent model (PIM). The platform independent model should be considered as independent from the underlying technology and should give only a view of the system regardless on execution platform. In other word, PIM is model of system designed in such way that it can execute on technologically independent virtual machine [6]. However, the PIM model must contains enough information in order to enable code generation using appropriate CASE tool.

PIM model for service for bibliographic records retrieval is given in the figure 2. The *Mediator* class provides a unique interface to communicate with different server components that implement the appropriate protocols. This class accepts query from server side of protocols and returns bibliographic records in the format which is defined by server. In this way it is enabled that regardless of protocol which is used, we always use the same method to submit query and retrieve search results. That means that our system becomes more scalable and it is possible to add some new search and retrieval protocols without refactoring this service.

The Mediator class can accept queries defined by different query languages and all those queries should be transformed in an internal query language which will be further forward to wrapper components. The QueryConverter interface is responsible for transformation of query. This interface has two implementation classes: CQLConverter and RPNConverter. In this implement-ation, it was chosen that accepted queries will be transformed into object representation of CQL query language which is defined by SRU standard. If we are going to add support for new query language it is necessary just to add new class which will implement interface *QueryConverter* shown in the Figure 2, but architecture of service remains the same.

One of the reasons for choosing CQL query language as a query that will be forwarded to library system is that concepts defined in the Z39.50 standard query language could be easily mapped to the corresponding concepts defined by CQL query language. CQL query language has very rich semantic, so it could be used for creating various types of queries. Also, because it is based on the concept of Context set, it is extendable and it allows usage of various types of Context sets for different purposes. So, we don't need to use CQL just for the purpose of searching bibliographic material it could be, for example, used for searching geographical data. According to those facts, we assumed that CQL is general query language and that probably any query language could be transformed in it. In the case that there is a new query language, it is necessary to perform mapping of this new query language into CQL query language or if it is not feasible to extend the object model of CQL query language with new concepts.

The next tasks of this service is to return records in the format which was defined by the client which sent request, and this task is delegated to the *RecordSerializer* interface. Bibliographic records which are retrieved after executing query should be in the form of an XML document.

Current implementation of service should support transformation of bibliographic records into XML document which can be instance of UNIMARCslim XML schema [18], MARC21slim XML schema [12] or Dublin Core XML schema [4] and because of that currently there are three classes which implement interface *RecordSerializer* (*UnimarcSerializer*, *Marc21Serializer* and *DublinCoreSerializer*). Adding support for new format would require creating a new class which would implement interface *RecordSerializer*.

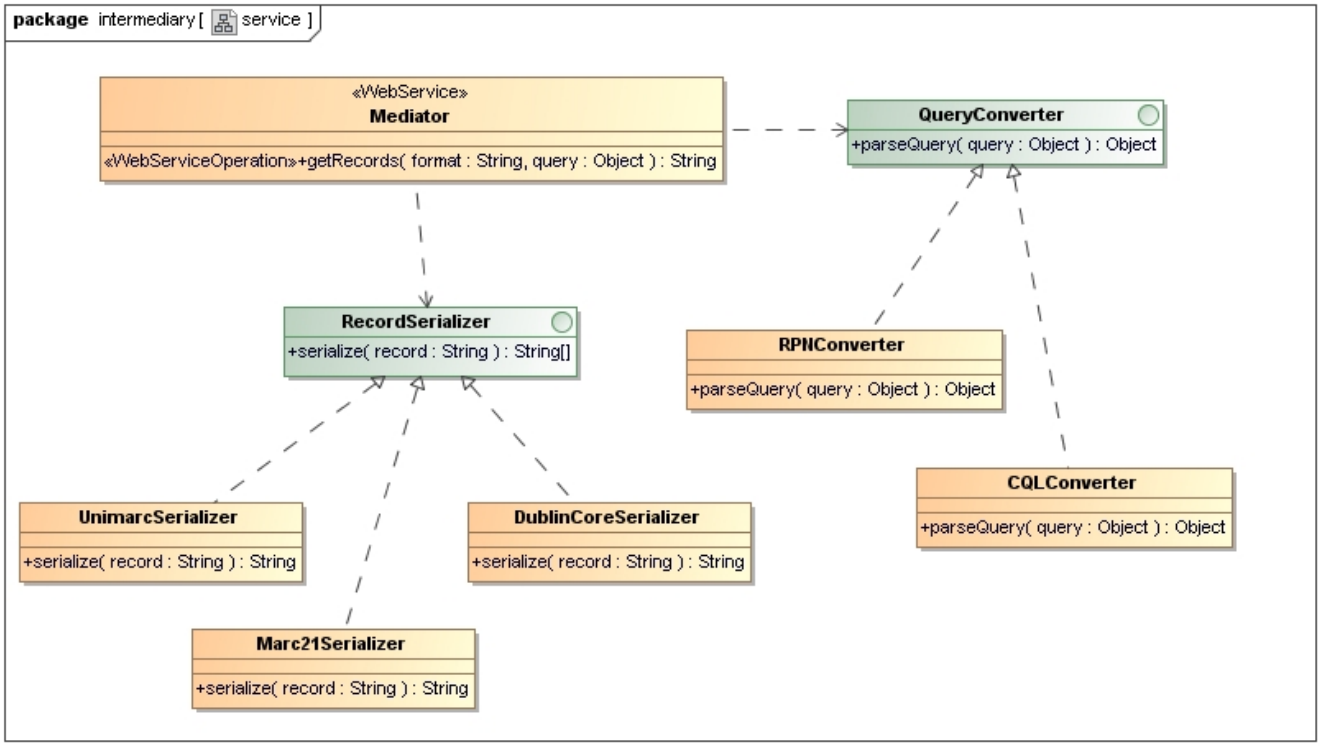

Figure 2: Platform independent model of service for bibliographic records retrieval.

4. ANDROMDA AND CODE GENERATION

Once the platform independent model has been accurately defined, platform specific details can be handled in the platform specific model (PSM). The role of PSM model type is to ease code generation that fit the underlying execution platform. MDA suggests using UML profiles to create language-specific code models. For example, an UML profile for a Java EE platform allows code models creation that ensures Java EE development after code generation.

However, this step can be skipped by using AndroMDA [1]. This is an open source, Java-based tool supporting model driven development. It has an open and pluggable design, such that many of its components can be extended easily. The core concept of AndroMDA is the use of so called cartridges. A cartridge describes the transformation rules from PIM over PSM to implementation code. AndroMDA is mostly used by developers working with J2EE technologies. Out-of-the-box AndroMDA can setup a new J2EE project from scratch, in which code is generated from a UML model.

In order to transform model into programming code, AndroMDA requires usage of stereotypes and tagged values which will be considered during the process of code generation. Choosing appropriate stereotypes and tagged value AndroMDA can generate code for Hibernate, EJB, Spring, WebServices, and Struts. Also, it is necessary to have installed Java 2 Virtual Machine (Java 2 SDK), at least version 1.5, and Maven or Ant. It is recommended to use Maven because most of the AndroMDA tools come with a Maven plugin.

After environment is set up, the next step is to use AndroMDA to create empty project for chosen technology. That project can be later imported into integrated development environment, such as Eclipse. The most significant sub-directories in the project structure are as follows:

- <projectDir>/mda/src -where the UML model file is located.
- <projectDir>/mda/conf -where the AndroMDA engine configuration file is located.
- <projectDir>/core/target -where the code generator places most of the resulting files. Those files will be overwritten on subsequent runs of the code generator so we should not make any modifications to them.
- <projectDir>/core/src -source files that require manual implementation go here. Files in this directory will not be overwritten on subsequent runs of the code generator.

This service for data retrieval is implemented in Java programming language and it is implemented as XML Web Service using CXF JAX-WS framework [2], so those options must be specified when AndroMDA creates initial project. Furthermore, elements of the PIM model described in the previous section should have stereotypes describing which class will be transformed into web service. AndroMDA has its own UML profile which should be used when it comes to modelling of web service. Namely, class which is going to be web service must have stereotype *<<WebService>>* and its operations which will be exposed as web service operations must have stereotype *<<WebServiceOperation>>*. In the figure 2, class *Mediator* and its operation *getRecords(String query, String format):String[]* has those stereotypes. Also package containing *Mediator* class must have stereotype *<<XMLSchema>>*. This stereotype defines a package which is mapped to a schema namespace, so that an .xsd file is created for this package and imported by the wsdl files that reference the schema. The default values for style and use of web service are wrapped and literal, respectively, but using tagged values andromeda_webservice_style and andromeda_web_service_use those values can be changed.

AndroMDA does not require using any specific UML tool but it is necessary that PIM model can be exported as XMI document. For creating PIM model for this service we used MagicDraw case tool [10] version 16.6, which support exporting UML model into XMI format. When PIM model is completed, we can start code generator and following documents are created:

- *Mediator.wsdl* and *intermediary.xsd* document – WSDL document which describes web service and XML schema document describing data types.
- *MediatorSEI.java* -the service endpoint interface (SEI)
- *MediatorSEIImpl.java* -the class which implements the SEI
- *MediatorSEIImplTest.java* - UnitTest for the MediatorSEIImpl (does not require deployment to web service container/runtime)
- *MediatorWSDelegate.java* - the class that the MediatorSEIImpl delegates to, for each method implementation
- *GetRecordsImpl.java* – the class that the MediatorWSDelegate delegates to. For each operation of web service one class is generated. This class should contain actual implementation of web service.
- Mediator_WSClient.java - Java service client which calls each web service.

After implementing business logic of web service, this service could be deployed on any runtime environment, like a JBoss. AndroMDA creates build files which will be used in process of deployment and developer does not need to think about that.

5. CONCLUSION

In this paper, we presented the development of an illustrative example of service for information retrieval using different protocols based on applications of a Model-Driven Architecture (MDA) approach. Our main idea was to present usage of AndroMDA framework in generating programming code based on UML models.

In this paper, we described functionality of service through computation independent model and then we presented architecture of service through platform independent model. Our idea was to implement that service in Java programming language as XML web service and AndroMDA provide mechanism for generating such code. It is just necessary to use appropriate stereotypes and tagged values in the process of modelling. AndroMDA has specific UML profile containing all stereotypes and tagged values needed for web services modelling.

It is general impression that AndroMDA framework can generate application which could be deployed on some runtime environment, but still it is expected that developers must implement business logic by themselves. AndroMDA could be very useful to less seasoned developers because they do not need to think about software architecture. That task can be delegate to AndroMDA.

6. ACKNOWLEDGMENTS

The work is partially supported by Ministry of Education and Science of the Republic of Serbia, through project no. OI174023: "Intelligent techniques and their integration into wide-spectrum decision support".

7. REFERENCES

[1] AndroMDA, http://www.andromda.org

[2] Apache CXF: An open-source services framework, http://cxf.apache.org/

[3] CQL – Contextual Query Language, http://www.loc.gov/standards/sru/specs/cql.html

[4] DublinCore XML Schema, http://www.loc.gov/standards/sru/resources/dc-schema.xsd

[5] Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, http://www.loc.gov/z3950/agency/Z39-50-2003.pdf

[6] Gašević, D., Đurić, D., Devedžić, V., 2006, *Model driven architecture and ontology development*, Springer

[7] J. de Almeida Monte-Mor, Ferreira, E.O., Campos, H. F., Marques da Cunha, A., Vieira Dias, L. A. ,2011, *Applying MDA Approach to Create Graphical User Interface*s, Eighth International Conference on Information Technology: New Generations, Las Vegas, Nevada, USA

[8] Javed, A. Z., Strooper, P. A. and Watson, G. N., 2007, *Automated generation of test cases using model-driven architecture*, In *AST'07*, page 3, Washington, DC, USA, IEEE Computer Society.

[9] Kleppe, A., Warmer J. and Bast, W., 2003*, MDA Explained: The Model Driven Architecture—Practice and Promise*, Addison-Wesley Professional.

[10] MagicDraw case tool, https://www.magicdraw.com/

[11] Maalal, S., Addou, M., 2011, *A new approach of designing Multi-Agent Systems*, International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11.

[12] Marc21Slim XML Schema, http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd

[13] Mellor, Stephen J., Scott, K., Uhl A., Weise, D., 2004, *MDA Distilled: Principles of Model Driven Architecture*, Addison-Wesley

[14] Meta-Object Facility, http://www.omg.org/spec/MOF/2.4.1/

[15] Ramljak, D. Pukšec, J., Huljenić, D., Končar, M., Šimić, D., 2003, *Building enterprise information system using model driven architecture on J2EE platform*, in Proceedings of the 7th International Conference on Telecommunications, IEEE, pp. 521–526.

[16] SRU – Search/Retrieval using URL, http://www.loc.gov/standards/sru/

[17] Unified Modelling Language, http://www.uml.org/

[18] UNIMARCSlim XML Schema, http://www.bncf.firenze.sbn.it/progetti/unimarc/slim/documentation/unimarcslim.xsd.

[19] XML Metadata Interchange, http://www.omg.org/spec/XMI/2.4.1/