Chapter 23

USING METAMODELS TO PROMOTE DATA INTEGRATION IN AN E-GOVERNMENT APPLICATION SCENARIO

Adriana Maria C. M. Figueiredo, Aqueo Kamada, Luciano L. Damasceno, Manuel de Jesus Mendes and Marcos Antonio Rodrigues

Abstract: The increasing popularization of Internet has created unprecedented expectations and promises when dealing with a more intelligent use of the available data. This use is based on metadata and on a new dimension of metadata modeling techniques. An initiative supported by OMG culminated in the current MOF – Meta-Object Facility specification, which is an open standard with facilities to definition and manipulation of metadata and metamodel. This paper describes a metadata management system that is based on the MOF specification and presents one usage of this system in an e-government scenario.

Key words: Metadata, Meta-Object Facility, Metadata Management System

1. INTRODUCTION

E-Governments initiatives around of the world are trying to make available to the citizens many of the public services accessible via Web. The most common principles in these initiatives are: i) cooperation among different public administrative units in the sense that they need to rationalize their resources and integrate their software assets; ii) device and user interface independence, meaning that users don't have any limitation on how they can access public services, whether by PC, hand-held, WebTV, mobile phone or any other device; iii) transparent access to the public services, in the sense that the citizen doesn't need to know the different administrative units involved in the execution of a selected service.

Adherent to these principles the main goals of these initiatives are: i) to allow independent public administrative units to cooperate amongst themselves to deliver services; ii) to provide public administrative units with a shared and common semantic description of concepts on which they can cooperate; iii) to provide services that can be deliverable through the coordinated use of legacy systems spread over the public administrative units.

In this context, to aggregate disparate existing services and legacy systems, moving the focus of interoperability from monolithic system to message sequences that cross public administrative units and geographic boundaries, an innovating and effective infrastructure to share and manage metadata is fundamental to service collaboration and cooperation. The incompatibility and proprietary solutions for the metadata administration have been the primary difficulty for a transparent access to data resident in different platforms. The elimination of these problems will accelerate the integration of applications, which require exchange of metadata, such as e-government, data warehousing, e-commerce and information portal applications.

A new dimension of metadata modeling and managing techniques was proposed by OMG with the introduction of the open standards Meta-Object Facility – MOF and XML Metadata Interchange – XMI [OMG 2003] that provide a comprehensive way to express and represent metadata, distributed access and standard data exchange mechanism.

This work presents the development of a set of tools in order to provide an effective and consistent framework to manage metadata based on the standards MOF and XMI.

In section 1, we address the motivation of this work. In section 2, the main foundations of MOF and XMI are presented. The architecture and tools that comprise the metadata management system are detailed in section 3. Section 4 describes the use of the implementation in an *e*-government scenario. Finally, in section 5, final considerations and future work are presented.

2. META-OBJECT FACILITY

The Meta-Object Facility (MOF) [OMG 2003] is a model driven distributed object framework for defining, managing and integrating metadata in software systems. To achieve its objectives, MOF defines an abstract language to describe metamodels and a generic framework for managing, in terms of repository, the metadata described by the metamodel.

The core of the MOF approach to metadata management is openness. The objective is to provide a framework that supports many existing metadata specification propositions and standards, and that allows new ones to be added as needed. To achieve this goal, MOF uses a four-layer metamodeling architecture.

Figure 1 illustrates the four-layer architecture adopted by MOF. A major difference between this architecture and other metamodeling architectures [MDC 1999, EIA 1994] is the fact that the meta-metamodel is defined using its own metamodeling constructs.

The top layer, M3, is the meta-metamodel level and corresponds to the MOF Model, the only model residing in this layer. The MOF Model forms the base for the metamodeling architecture and its main responsibility is to define the language to specify metamodels. Examples of meta-objects in the meta-metamodeling layer are: MOF::Class, MOF::Association and MOF::Attribute.

The next layer, M2, is populated with metamodels instantiated from the MOF Model. UML and CWM [OMG 2003] are examples of standard metamodels defined by OMG to the software development domain. However, M2 layer is not restricted to standard metamodels. New metamodels can be defined and existing ones can be extended in order to accomplish specific contexts' requirements. Examples of meta-objects in the metamodeling layer are: UML::Class, UML::Attribute, CWM::Table and CWM::Record.

Layer M1 corresponds to metadata layer and it consists of models representing a domain of information. Examples in the model layer are the class "Person" and the attribute "Profession".

Finally, in layer M0 reside instances or data described by M1 models. Examples: "Ester", "teacher".

The MOF is intended to support a wide range of usage patterns and applications. Two distinct viewpoints are provided for possible usage patterns:

Modeling Viewpoint: it is the designer's viewpoint that uses the MOF to define information models for a particular domain. The model is then used to drive subsequent software design and implementation steps.

Data Viewpoint: it is the programmer's viewpoint that uses a meta-level to get understanding of an element in the lower level.

In addition to the open architecture that allows the definition of metamodels from different domains, the MOF specifies standard mappings to expose instances of MOF compliant metamodels with Corba IDL and Java interfaces. The prime purpose of these mappings is to define interfaces for information models described in terms of the MOF Model. These

interfaces ensure structural and logical consistency in manipulating the metadata described by a metamodel.

The XML-based Metadata Interchange format (XMI) [OMG 2003] is another OMG’s specification proposed to enable easy interchange of MOF based metadata in distributed heterogeneous environments. The XMI specification basically defines the mapping of a MOF metamodel to XML Document Type Definition (DTD) or XML Schema. The generated XML DTD or XML Schema is used to encode the metadata into XML documents and to decode XML documents and reconstructs the metadata. Any metadata repository or tool that can encode and decode XMI streams can exchange metadata with other repositories or tools with the same capability. Applications can be made more intelligent and automatic by adding metadata information to the runtime environment through the use of XMI documents, transferred over the Internet and validated and interpreted at a remote location.

Figure 1 – MOF Metamodeling Architecture

3. GRM – A METADATA MANAGEMENT SYSTEM

The GRM system is part of a pilot project being developed at Research Center *Renato Archer* (CenPRA) with the main objective of creating a

platform to support the development of Internet collaborative services in the government domain.

A central component of this platform is a metadata repository where complex artifacts like business rules and collaboration models can be stored. GRM is the system being developed to fulfill this key requirement. It is based on the MOF architecture and is expected to be used in both viewpoints described in section 2. From *the modeling* viewpoint, GRM can store metamodels and corresponding models. From the data viewpoint, metamodels can be accessed to get the understanding of heterogeneous metadata.

3.1 GRM ARCHITECTURE

GRM's architecture is illustrated in Figure 2. The components of this architecture are based on the repository system architecture presented in [Bernstein, 1998] and they are described next.

Metamodeling Tools: these are client softwares that access the repository manager in order to allow the definition and manipulation of the artifacts (i.e. metamodels) stored in the repository. Examples of tools are graphical editors, XMI importer/exporter and compilers. Language generators are a very important kind of tool for GRM. In this case, a generator maps, automatically, a metamodel to MOF standard mappings such as XMI, Corba or Java, generating XML DTD, XML Schema, Corba and Java interfaces and their respective implementation;

Repository Manager: this component and the information model form the repository engine. Services are provided to access and manage the repository itself and the metamodels it stores. Features like version control, check-in/check-out, and access control and configuration management are required for a shared use of the repository;

Information Model: the information model specifies the structure and semantics of the metamodels stored in the system;

Persistence: the artifacts manipulated by the repository manager are stored in databases (RDBMS, OODBMS) and/or files system.

In the bottom of Figure 2 are presented boxes with some of the most relevant tools of GRM repository system. On the left side are the metamodeling tools and on the right side tools for modeling and metamodeling. In the next sub-section we describe with more details the implementation of the MODL compiler and the XML DTD generator.

Figure 2- GRM Architecture

3.2 IMPLEMENTATION DETAILS

GRM has been realized using the Open Source Complex Information Manager (CIM) [Unisys 2002] developed by Unisys Corporation. CIM is an implementation of the MOF 1.4 and JMI specifications [JCP 2002]. Its goal is to provide a platform independent metadata infrastructure for developing model driven tools and applications suites. CIM standard edition is available with the following features: a GUI-based administrative tool for configuring and managing the CIM, Java interfaces generator and metamodel server generator, XMI importer/exporter, access control and persistence using XMI format files.

Our objective is to improve this open source, adding the following features:

- an MODL compiler;
- a generic metadata browser to metamodels and their respective models;
- a generic way to describe models using the Human Understandable Text Notation – HUTN. This notation is also generated from a metamodel.
- persistence in a relational database;

- XML DTD and XML Schema generators;
- Corba IDL generator;
- an Object Constraint Language (OCL) interpreter to evaluate the constraints used to enhance the semantics of a metamodel;
- query mechanisms, allowing to search the repository for specific elements.

At the moment, we have implemented the MODL compiler and the XML DTD generator. We have decided for the implementation of an MODL compiler because CIM comes with only one-way to populate the repository: import of metamodels expressed in the XMI format. XMI was specified to be a machine exchange format and it is neither succinct, nor easily readable or writable.

Meta-Object Definition Language (MODL) [DSTC 2001] is a textual language for specifying MOF metamodels and it provides an alternative way to populate the repository. The MODL compiler was implemented in the Java programming language. The tools JavaCC [JavaCC 2002] and JJTree [JJTree 2002] were used in the lexical analysis and in the generation of the syntax tree, respectively.

Figure 3 illustrates the different approaches used by the XMI importer and the MODL compiler, regarding meta-objects generation and external references. The importer tool reads an XMI file (1), generates a DOM tree (2) that is used by a generator (3) to create meta-objects (4 and 5), that is, MOF instances.

The MODL compiler approach is similar to the XMI importer. The two differences are in step 2 and they are presented next.

1. the DOM tree is generated (2c) from the syntax tree (2a) created during the compilation of the metamodel; and
2. The MOF model is accessed (2b) in order to check external references used in the metamodel expressed in MODL.

Figure 3 illustrates an example, where metamodel C is inserted in the GRM repository using the XMI importer and the MODL compiler. The metamodel C imports constructs defined in metamodels A and B and when it is mapped to XMI format, metamodels A and B are mapped as well. If metamodels A and B do not exist in the repository at the moment that C is inserted, A and B are also inserted. In the case of the MODL compiler, external references, expressed in the *import* command, are checked as the DOM tree is generated. The MOF Model interfaces are accessed in order to get the information about the contents of the repository. In the case, metamodels A and B do not exist, when C is compiled, an exception is raised.

Figure 3 – Meta-objects generation in GRM system

Although the CIM comes with functionalities to import/export XMI files, the software does not generate XML DTD for a specific metamodel and, therefore, it is not possible to validate the syntax and the semantics of models instantiated from the metamodel. An XML DTD generator was implemented making heavy use of MOF Reflective Module. This module is defined in the MOF specification and its objective is to provide an introspection mechanism common to all metamodels. The XML DTD generator tool navigates through the metamodel, visiting all its constructs and for each one, reflective APIs are used in order to find out the meta-object that describes that construct. According to the rules defined in the XMI specification the construct can be mapped to an XML element or to an XML attribute.

4. AN E-GOVERNMENT APPLICATION SCENARIO

Governments around the world are looking for alternatives to deliver services in the Internet. Traditionally, government resources are spread over the public administrative units and a critical problem in this scenario is the

challenge of integrating these resources, respecting the autonomy of each single public unit.

At CenPRA, it is being developed a prototype with the main objective of validating an architecture that facilitates the integration of distributed and heterogeneous legacy systems and data. The major components of this architecture are illustrated in figure 4 and described next.

The service platform provides a uniform service access and management environment, across different administrative and technological services. The integration framework, focus of this paper, is comprised of modules based on MOF technology and with the main objective of enabling data and metadata integration. Legacy systems, also referred to as legacy layer, are composed of databases, applications, and systems with enormous value to the organization.

In this scenario, the CWM and EDOC Modules are components generated automatically by the GRM system and based on the CWM and EDOC metamodels, respectively. The code generated implements the interfaces that allow the access and manipulation of models instantiated from CWM or EDOC metamodels.

The CWM specification [OMG 2003] provides metamodels to represent common warehouse metadata such as object-oriented, relational, record, multidimensional, XML data resources and others. Additionally, CWM provides a transformation metamodel with constructs for specifying transformation rules.

The EDOC specification [OMG2003] provides metamodels to specify enterprise-distributed systems using a component-oriented approach. The Entity metamodel is used in our scenario to model entity objects that are representations of concepts of the government domain. For example, a citizen.

CWM Module is in the integration framework and in the legacy layer. In the legacy layer it is responsible for getting the data and transforming it to XMI format. Metadata and data are interchanged between the integration framework and the legacy layer as stream. In the integration framework, transformation models map metadata from legacy layer to an EDOC entity model and resulting entity objects are made available to the service platform.

For the realization of this scenario, it was adopted an open service platform provided by Fokus Institute [Fokus], which is one of CenPRA's partners in the development of the e-government platform. The main objective of Fokus' platform is to enable integration and composition of existing and new application services based on different programming languages, access and services technologies.

Figure 4– GRM in an eGovernment Scenario

5. CONCLUSIONS AND FUTURE WORK

In this paper a metadata management system to support metamodeling and distributed metadata access has been described. CWM and EDOC metamodels have been used to solve the problems posed by integrating legacy data in an e-government scenario. The components of GRM, created around Metadata Repository, such as MODL Compiler, Metamodel XML DTD Generator, Interfaces Generator, Metamodel Server Generator, Metamodel/Model Visual Tool, XMI Import/Export APIs and Query, improve the capabilities to generate metadata that needs to be exchanged among heterogeneous systems in an Internet based collaborative system. One of such improvements can be seen, for example, on MODL Compiler, which provides a flexible alternative way to populate the metadata repository. Another improvement is provided by a XML DTD generator, which navigates among a metamodel and maps its constructs to XML elements and attributes. New components are being implemented to increase the functionalities of the open software. The strength of GRM is in the fact

that its inception, and all its components, were developed having in mind three specific requirements: (1) platform and operating system independence; (2) adherence to open standards proposed in the distributed system context; and, (3) use of the orthogonality concept when features are added, applying them to any metamodel.

The use of MOF concepts and CWM and EDOC metamodels create a robust foundation to an extensible and flexible architecture that promotes data integration and, not less relevant; ensure that legacy data quality is maintained.

Further research is ongoing on the basis of the present work. Application Process integration is the next issue to be addressed in the prototype. Research to define a framework that could facilitate the development and composition of Web services is being developed. The first scenario for this framework includes the creation of government ontologies, semantic representation of data and process to support consistent description of information for e-Gov services composition.

REFERENCES

- Bernstein P. A (1998) "Repositories and Object oriented Databases", ACM/SIGMOD vol. 17, pages 88-96, 1998
<http://www.acm.org/sigs/sigmod/disc/disc99/disc/record/issues/9803/bernstein.pdf>
accessed in 05/19/2003
- DSTC (2001) "dMOF – User Guide (Release 1.1)", Distributed Systems Technology Centre (DSTC), University of Queensland, Brisbane, Australia, June/2001.
http://www.dstc.com/Downloads/CORBA/MOF/dMOF1_1.UserGuide.pdf
- EIA (1994) "CDIF - CASE Data Interchange Format – Overview", Electronic Industries Association – EAI, January, 1994
- Fokus – Fraunhofer Institute for Open Communication Systems
<http://www.fokus.fraunhofer.de/>
- JavaCC (2002) "Java Compiler Compiler (JavaCC) - The Java Parser Generator".
http://www.webgain.com/products/java_cc/documentation.html
- JCP (2002) "Java Metadata Interface (JMI) Specification", version 1.0, JSR40 – Java Community Process, June/2002.
<http://jcp.org/aboutJava/communityprocess/final/jsr040/index.html>
- JJTree (2002) "JJTree Reference Documentation".
http://www.webgain.com/products/java_cc/jjtree.html
- MDC (1999) "Open Information Model", Object Management Group, Needham-MA, USA, April/2002. <http://www.omg.org/cgi-bin/doc?ad/99-04-06>
- OMG (2003) "Catalog of OMG Modeling and Metadata Specifications", v. 1.4, Object Management Group, Needham-MA, USA, October/2003.
http://www.omg.org/technology/documents/modeling_spec_catalog.htm