Metadata Repository Support for Legacy Knowledge Discovery in Public Administrations

Adriana Maria C. M. Figueiredo[1], Aqueo Kamada[1], Luciano L. Damasceno[1], Marcos Antonio Rodrigues[1], Manuel de Jesus Mendes[2]

[1]Centro de Pesquisas Renato Archer- CenPRA
DSSD - Divisão de Software para Sistemas Distribuídos
Rodovia Dom Pedro I Km 143,6 - CEP 13082-120 - Campinas (SP) – Brazil
<name>.<surname>@cenpra.gov.br
[2]Unisantos, Santos, Brazil
mj.mendes@terra.com.br

Abstract: One of the big challenges of public administration is the need to understand and evolve legacy systems for the purpose of documentation, improvement, modification, interoperability, porting, migrations, reuse, redesign and/or redeployment. There is a need for standardization in legacy transformation that will enable integration and interoperability between different solutions. The OMG has issued a request soliciting proposals for a metamodel to capture knowledge from legacy systems (referred to as the Legacy Knowledge). In this paper we examine how a MOF-based repository can support the definition, modeling, exchanging and integration of metamodels created for legacy knowledge discovery.

1 Introduction

Legacy systems can be understood as any useful and deployed software and data that run on specific platform, written in specific language and, normally, has been long time in production environment. Despite their obsolescence, legacy systems present an enormous commercial value and continue to provide a competitive advantage by supporting unique business processes and containing invaluable knowledge and historical data. This kind of operational legacy software often resists evolution because its strategic value and its low ability to adapt through factors not exclusively related to its functionality. Some of such factors are the system's difficulty to be understood or maintained in an effective cost manner, its core business knowledge that is hard to enhance or impossible to replicate, its difficulty to interoperate, or its dependence on old technologies or specific architectures. These systems, even though with enormous investments, they work.

For a large number of organizations the need for legacy migration is usually triggered by certain external events such as the need to provide Web-access to an existing application or the need to abandon an unsupported platform. In [3], legacy migration is defined as the transformation of data and procedures from the old to new

system. For government organizations, legacy migration is a crucial part of their day-to-day business requirements. Legacy transformation provides the means to address these requirements using systematic and low risk approaches.

The OMG has issued a request [6] soliciting proposals for a metamodel representing the structure of the legacy software and its related artifacts (referred to as the Legacy Knowledge). The primary purpose of this metamodel is to provide the ability to document legacy systems, discover reusable components in legacy software, support transformations to other languages, or enable other potential transformations. The metamodel will also enable information about legacy software artifacts to be exchanged among different tools. Standardization of legacy transformation metamodels will reduce the risk of undertaking software improvement initiatives. The ability to share common information across projects that use a variety of tools and processes will lessen the time, risk and cost of software transformations.

Common Warehouse Metamodel – CWM [5] is another relevant OMG specification related to legacy transformation. A primary objective of the CWM is to define a metamodel of a generic data warehouse architecture. Additionally, the CWM specification covers basic transformations among all types of data sources and targets: object-oriented, relational, record, multidimensional, XML, OLAP, and data mining.

OMG`s request for a legacy knowledge metamodel also seeks a common repository structure to represent information about existing software assets and its operating environment, creating, though, knowledge bases.

In this paper we present a MOF-based repository system, which will enable the definition, modeling and exchanging of models that represent existing software assets. In this section we addressed the motivation of this work. In section 2 we review main concepts on information and knowledge as metadata entities captured by the concepts and relations described in the MOF Model. MOF and XMI foundations are presented as well. In section 3, a MOF-based repository system is detailed. An e-government scenario application is presented in section 4 and finally, in section 5, final considerations and future work are presented.

2 Foundations

The distinction between data, information and knowledge is often blurred, but for our purpose, data is a raw thing, like a fact represented as an item. Information is data given context, and vested with meaning and significance [8]. Information is made by data with metadata and context and it is inherently static. Knowledge is applied information, that is transformed through reasoning and reflection into beliefs, concepts, and mental models, and that is used to produce results and is learned from experience and happens in the human brain. It is inherently dynamic and changing.

Metadata is information about data and its main objective is to facilitate the access, the management, the processing and the sharing of a great collection of structured and/or non-structured data [4]. The metadata definition is strongly related to the modeling and meta-modeling concepts. Modeling refers to the description of structured metadata representing the data that compose a domain, as well as the relationships among such data. The collection of these metadata encompasses a model

of the domain. Metamodeling refers to the ability to represent and manipulate metamodels, which are models that define constructs used when modeling a domain.

The Meta-Object Facility – MOF [5] is a model driven distributed object framework for defining, managing and integrating metadata in software systems. MOF has been designed with the main objectives of being:

- open: it is capable of describing a wide range of metamodels; and
- extensible: it is a core model and is capable of extension by inheritance and composition.

In order to achieve this, the MOF uses a layered metadata architecture, which key feature is a meta-modeling layer that provides a common language that ties together the metamodels and models. The MOF metadata architecture is illustrated in Figure 1.

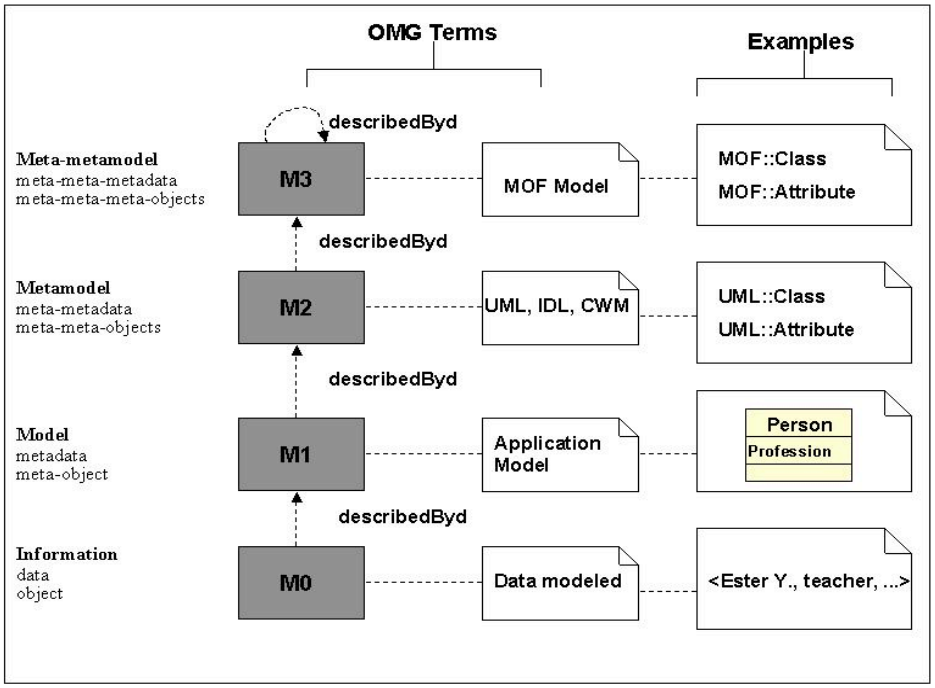

Fig. 1. MOF metadata architecture

The MOF Model, in layer M3, defines an abstract language to describe metamodels. Metamodels, in layer M2, are instances of the MOF Model. A Model, in layer M1, is an instance of a metamodel. Finally, in layer M0 reside instances or data described by M1 models.

Standard mappings provided by MOF and JMI [7] specifications, expose instances of MOF compliant metamodels to CORBA IDL and Java interfaces, respectively. The prime purpose of these mappings is to define a generic framework for managing, in

terms of repository, the metadata described by the metamodel. The standards interfaces ensure structural and logical consistency in manipulating the metadata described by the metamodel.

As for the problem of metadata exchange, the OMG has standardized the XML Metadata Interchange – XMI [5]. XMI defines how XML tags are used to represent serialized MOF-compliant models in XML. MOF-based metamodels are translated to XML Document Type Definitions (DTDs) or XML Schemas and models are translated into XML Documents that are consistent with their corresponding DTDs or XML Schema. XMI is an interchange mechanism to be used between various tools, repositories and middleware.

3 GRM – A MOF-based Repository System

The Centro de Pesquisas Renato Archer – CenPRA is particularly interested in creating a platform to support the development of Internet collaborative systems in the government context. A central component of this platform is a MOF-based repository to manipulate metamodels from different domains. GRM is the MOF-based repository system developed for this purpose.

In the legacy domain, we believe that work on legacy migration will require different metamodels such as metamodel for data transformation, metamodel for representing languages and metamodel for describing platforms.

According to Bernstein in [1], the goal of a repository is to store models and contents of engineered artifacts, such as software, documents, maps and information systems. In this sense, we define the objective of a MOF-based repository as being to store MOF compliant metamodels. In the next sub-sections, the architecture, main functionalities and GRM implementation details are presented.

3.1 GRM Architecture

The components of a repository system include a database, a repository manager, an information model and tools for populating the database and accessing its contents [1]. GRM is composed by the same components and the basic difference is that the information model is the MOF Model and the contents manipulated are metamodels described by the MOF Model. Additionally, GRM supports XMI format as the mechanism for metadata exchange. Figure 2 illustrates GRM architecture.

The database provides standard database management facilities, such as persistent storage, keys, data integrity and etc. The repository manager and the MOF 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, access control and configuration management are needed for a shared use of the repository.

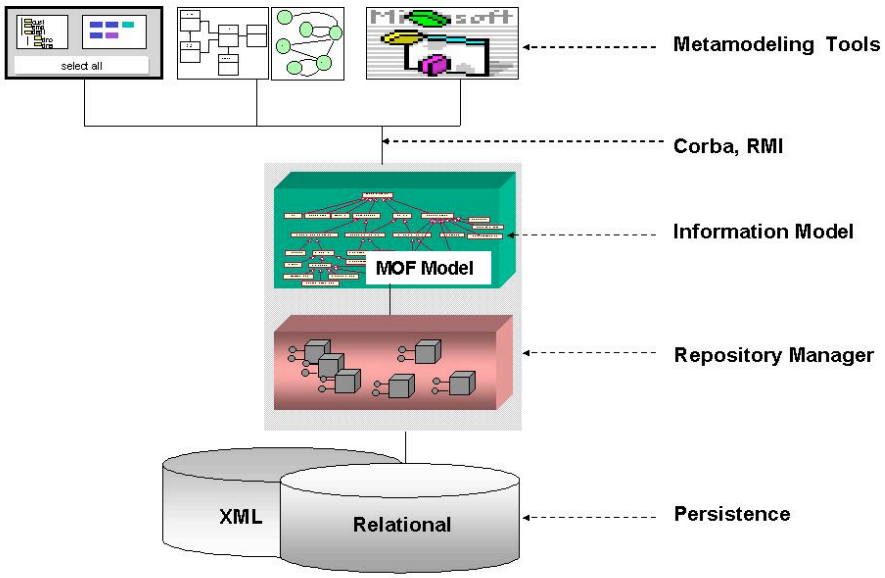

Fig. 2. GRM architecture

GRM provides a set of metamodeling tools for the definition, access and manipulation of metamodels stored in the repository. Relevant tools are described next:

- MODL compiler. Meta-object Description Language [DSTC 2001] is a textual notation to describe MOF metamodels. An MODL compiler is an alternative to a graphical editor, providing a way to describe complete metamodels using a textual notation.
- XML DTD/XML Schema generator. This tool, automatically, generates XML DTD/XML Schema for a metamodel by applying the XMI generation rules. The generated XML DTD/XML Schema is used to validate models against its metamodel.
- Generic browser. A generic metadata browser enables the navigation through the metamodels and their respective models.
- Corba/Java server generators. These two tools are very important in a MOF-based repository system. They automatically generate standard Corba IDL/Java interfaces and their corresponding server implementation. The generated code is a standard software component of a repository, which information system is the metamodel mapped.
- XMI import/export APIs: These APIs enable the streaming of metadata in the XMI format.

3.2 Implementation Details

GRM specific requirements are: i) platform and operating system independence; ii) adherence to open standards proposed in the distributed system context; iii) use of the orthogonality concept when adding a new feature, applying it to any metamodel.

The repository has been realized using the Open Source Complex Information Manager (CIM) [9] developed by Unisys Corporation. CIM is a Java implementation of the MOF and JMI specifications. Its goal is to provide a platform independent metadata infrastructure for developing model driven tools. 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.

At the moment, we have added to CIM an MODL compiler, a XML DTD generator and a generic browser to metamodels and their respective models. We have decided for the implementation of an MODL compiler because CIM comes with only one way to populate the repository: importing metamodels expressed in XMI format. XMI was specified to be a machine exchange format and it is neither succinct, nor easily readable or writable. MODL provides an alternative way to populate the repository. The generic browser provides a way to visualize the metamodels and the XML DTD generator is used to validate models against its metamodel

4 Application Scenario in a Public Administration

In the public administration there are many administrative units, such as agencies, departments and public companies, and each of them contains legacy data used to manage internal processes. Typical problems in this context are the use of different names, structures or scales for the same kind of information, as well as information represented at different levels of granularity, refinement, or precision.

In this scenario, we propose a framework to, effectively, discover, manage and share knowledge among distinct administrative public units. The proposed framework is illustrated in Figure 3.

The Knowledge and Metadata Framework is composed by various MOF-based repository systems, automatically generated by GRM. The information system of each of this repository is a MOF-compliant metamodel where CWM Data Resources Metamodels, CWM Data Analysis Metamodels, CWM Extensions (CWMX) Metamodels and EDOC Entity Metamodel are stored. Metamodels that represent object-oriented, relational, record, multidimensional, and XML data resources compose the Data Resources Metamodels. On the other hand, CWM Data Analysis Metamodels conceive the metamodels that represent data transformations, OLAP, data mining, information visualization, and business nomenclature. And finally, the CWMX is a non-normative model extension to the CWM metamodels [5] that facilitates and enables the access to the legacy system. CWMX consists of: Entity Relationship, COBOL Data Division, DMS II, IMS, Essbase, Express, InformationSet, and Information Reporting.

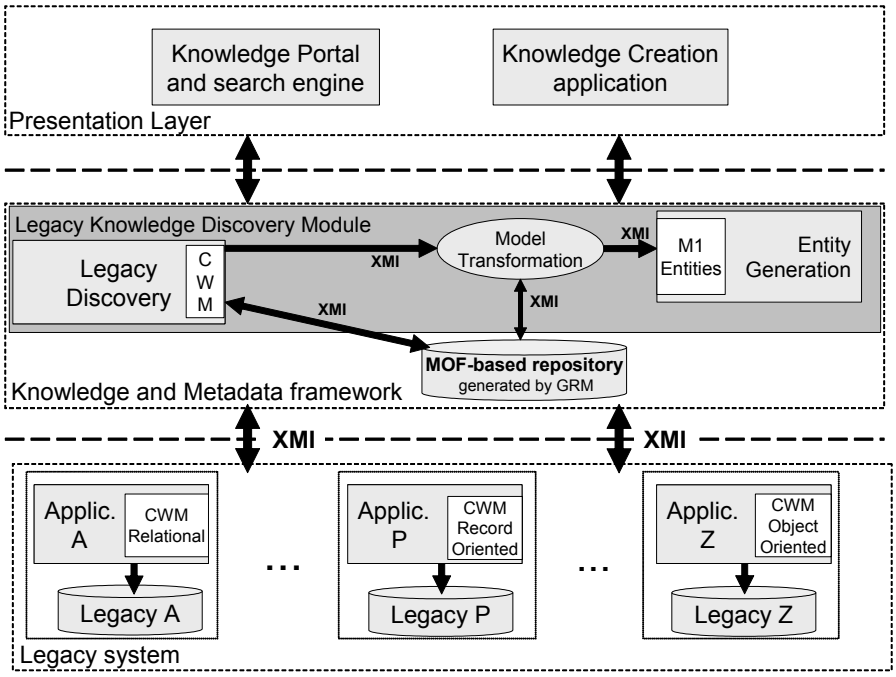

Fig. 3. Knowledge and Metadata Framework

In Knowledge and Metadata Framework tier, there are others components like:

- The Model Transformation, a component that reads an M1 transformation rule stored in the repository and generates the transformation code that executes the rules on the XMI document containing the M1 model and the M0 data.
- The Legacy Discovery Module, sends requests to the legacy systems and receives an answer as a XMI document containing the requested information (M0 data) and the meta-information (M1 model) that describes it.

- And the Entity Generation Module that receives XMI documents and fills the M0 EDOC Entities, i.e., representations of concepts in the application domain [5], with the M0 data contained into these documents.

For clarification, an example of the architecture's operation will be exposed. In this simple example, it is necessary to fill an EDOC Entity called Citizen resident in the EDOC Module with the resulting information extracted from the legacy system.

The modules present in the legacy system tier receive a request from the Legacy Discovery Module to query their databases to obtain the required citizen information stored into the different database resources. So, the modules realize the query and return the desired information.

Thus, the information (M0) will be encoded together with the respective model (M1) in a XMI document and transmitted to the above tier. In Knowledge and Metadata Framework tier, the XMI document is received by the Legacy Discovery Module that sends to the Model Transformation. The Model Transformation reads the transformation rule stored in the MOF repository and generates the trasnsformation code that executes the rules on M0 data. This transformed data fill the M0 EDOC Entities of the Entity Generation.

So, the Legacy Discovery Module is responsible for collecting knowledge from the legacy systems present in different database resources and make, the different models of the collected knowledge, available to the others administrative units.

Our approach brings many advantages to government administration and some of them are listed next:

- It provides a formal representation to knowledge;
- It provides an exchange mechanism, allowing knowledge to be interchanged among tools and repositories;
- The ability to share knowledge across public units will improve the understanding of government business and promote data rationalization.

5 Conclusion

In this paper we have presented a MOF-based repository system, which main goal is to support the definition, modeling, exchanging and integration of metadata spread over different legacy systems. Metamodeling tools, an MODL Compiler, Metamodel XML DTD Generator, Metamodel Server Generator, Metamodel/Model browser and XMI Import/Export APIs improve the capabilities to generate metadata that needs to be exchanged among heterogeneous systems in an Internet based collaborative system. New components are being implemented, to increase the functionalities of the open software.

The strength of GRM are: i) it is based on a metamodeling architecture that allows the definition of metamodels from different domains; and ii) it provides a metadata interchanging mechanism that relies on this architecture.

We have presented a preliminary usage of these artifacts in a government scenario, considering that in the public administration there are many administrative units, such as agencies, departments and public companies, and each of them has a lot of physical

and semantic heterogeneity in their legacy systems. We think that this government scenario is ideal to exercise the many aspects of meta-modeling structure of the legacy systems and their related artifacts, such as, the ability to document legacy systems, discover reusable components in legacy software, support transformations to other languages, or enable other potential transformations.

Further research is ongoing on the basis of the present work. Research is being developed to define a framework that uses information, about legacy software artifacts, to be exchanged among different tools in order to facilitate the development and composition of e-government services.

The framework shall support all aspects related to service collaboration such as workflow, choreography, contractual interfaces and so on. The metadata for the services description will be extended in order to incorporate all the necessary information for service composition (for example using Web Services Standard WSDL). This information will be mapped and stored in specific MOF-compliant repositories of the framework.

.

References

1. Bernstein P. A., "Repositories and Object oriented Databases", ACM/SIGMOD vol. 17, pages 88-96, 1998. http://www-agce.informatik.uni-kl.de/publications/edcoc.pdf accessed in Dec 1st, 2003
2. DSTC "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 accessed in Dec 1st, 2003
3. Heiler S., Lee W., Mitchell G., "Repository Support for Metadata-based Legacy Migration", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1999
4. Kerhervé B., Gerbé O "Models for Metadata or Metamodels for Data?", Proceedings of 2nd IEEE Metadata Conference, 1997.
5. OMG "Catalog of OMG Modeling and Metadata Specifications", Object Management Group, 2003, Needham-MA., USA. http://www.omg.org/technology/documents/
6. modeling_spec_catalog.htm accessed in Dec 1st, 2003
7. OMG Request For Proposal - Legacy Transformations: Legacy Knowledge Discovery Meta-Model (LKD), October 2003. http://www.omg.org/docs/lt/03-09-01.pdf accessed in Dec 1st, 2003
8. 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 accessed in Dec 1st, 2003.
9. Choo C. W., "Encyclopedia of Communication and Information". Three-volume encyclopedia, published in 2001 by Macmillan Reference USA, New York. http://choo.fis.utoronto.ca/macmillan/ accessed in Dec 1st, 2003.
10. Unisys "JMI-RI Documentation", CIM Version 1.3, Unisys Corporation, June/2002.
11. http://jcp.org/aboutJava/communityprocess/final/jsr040/index.html accessed in Dec 1st 2003