

Applying MDA development approach to a Hydrological Project

Breno Lisi Romano, Gláucia Braga e Silva, Adilson Marques da Cunha and Walter Itamar Mourão
Brazilian Aeronautics Institute of Technology

Abstract

This paper describes the application of an MDA development approach to the Project of Amazon Integration and Cooperation for Modernization of Hydrological Monitoring. This project is currently under the development of the Brazilian Aeronautics Institute of Technology. The adopted approach involves among other issues the source-code generation from UML models using the AndroMDA Framework. This development approach was implemented in the four different phases of: Modeling; Stereotypes and Properties Setting; Documentation; and Script Generation. Besides these implemented phases this paper describes also the configuration of the development environment. At the end, some of the main results are reported in terms of major findings and obtained outcomes.

Key Words: AndroMDA, Code Generation, Development Environment, Documentation, UML Models.

1. Introduction

The Model Driven Architecture (MDA) is based on standards established by the Object Management Group (OMG). It provides an open approach mainly to face challenges coming from business and technology changes.

The MDA separates business and application logic from underlying platform technology.

Nowadays, several different Platform-Independent Models (PIM) have been built using Unified Modeling Language (UML) and other associated OMG modeling standards.

PIM supports applications and integrated systems business functionalities and behaviors. It can be built by using MDA on virtually almost any open or proprietary platform including Web Services, J2EE and others [1].

PIM represents business functionality and behavior of an application not including its technology-specific code implementation. It enables interoperability both within and across platform boundaries. Once not tied, business and technical aspects of an application or integrated system can evolve at its own pace. Usually, business logic

responds to business needs, and technology takes advantage of new developments as business requires it.

Although, before the MDA specification, there were already some code generation tools from models or Domain Specific Languages (DSL), the MDA addresses the code generation from models in a wide, structured and unbound way front of technologies or suppliers.

The AndroMDA Framework is one of the open source MDA generator developed for multiple platforms. It supports UML Models and is extensible by means of cartridges. A cartridge provides a metamodel, a set of transformation rules and code templates to define how UML models are transformed to a specific platform code. It defines transformations for web platforms such as Spring, JSF, Hibernate, Struts, and others. Even though it uses UML models, AndroMDA does not provide a UML Case tool. Therefore, an external tool is needed to define UML models in an XMI format which AndroMDA can understand [2].

Within this context, this paper tackles a development approach, based on MDA, used for generating logical and physical models, and some CRUD (Create, Read, Update, and Delete) source codes from a UML model using resources of the AndroMDA Framework.

This approach was applied on the Project of Amazon Integration and Cooperation for Modernization of Hydrological Monitoring (*Projeto de Integração e Cooperação Amazônica para Modernização do Monitoramento Hidrológico - ICA-MMH B*).

This project has been developed in a collaborative work involving the Brazilian Aeronautics Institute of Technology (*Instituto Tecnológico de Aeronáutica - ITA*) and the Brazilian National Water Agency (*Agência Nacional de Águas - ANA*), supported by the Brazilian Projects and Studies Foundation (*Financiadora de Estudos e Projetos - FINEP*) [3].

The remaining of this article is organized as follows. The second section shows some related works. The third section describes a systemic view of the adopted project. The development approach is shown on the fourth section. The obtained results are reported on fifth section. Finally, the sixth section presents some conclusions and suggestions for future work.

2. Related Works

Mizuta and Huang [4] propose an approach in which grid services are represented by class models in UML. The UML output in XML Metadata Interchange (XMI) format derived from a CASE tool. It is used as input for the AndroMDA Framework to generate a suite of source code files and related settings for the GT3 (Globus Toolkit 3).

Bezivin et al [5] present an e-business development approach, as an illustrative example, related to a travel agency problem, based on two different MDA applications: the first using UML and the second using the Enterprise Distributed Object Computing (EDOC).

Finally, Gharavi et al [6] explore an MDA approach for AJAX web applications, proposing a UML scheme for modeling AJAX user interfaces. In their research it is reported the adoption of the AndroMDA Framework for creating an AJAX cartridge to generate the corresponding AJAX application code using back-end integration.

3. A Systemic View

In the ICA-MMH B Project, hydrometeorological information are collected by remote measurement stations comprised of data acquisition equipments which can be linked to Data Collection Platforms - DCP (*Plataformas de Coletas de Dados - PCD*) located nearby watersheds.

These measurements are taken from different geographic points on regular or even irregular intervals, or determined by hydrometeorological events.

The collected data become available through the Internet to be accessed by users of hydric resources such as institutions, corporations, involved neighboring countries, public and environmental organizations, farmers, research institutes, and other interested groups.

Some of the main applications for these data are hydric resources availability estimations, hydrological variability analyses, climate change previsions, and meteorological and critical events forecasting.

Within this context, the ICA-MMH B System of Systems – SoS is comprised of the following systems, as shown in Figure 1: Data Acquisition System; Data Treatment System; Monitoring, Control, and Decision Support System; Data Diffusion System; and the proposed Application Database System.

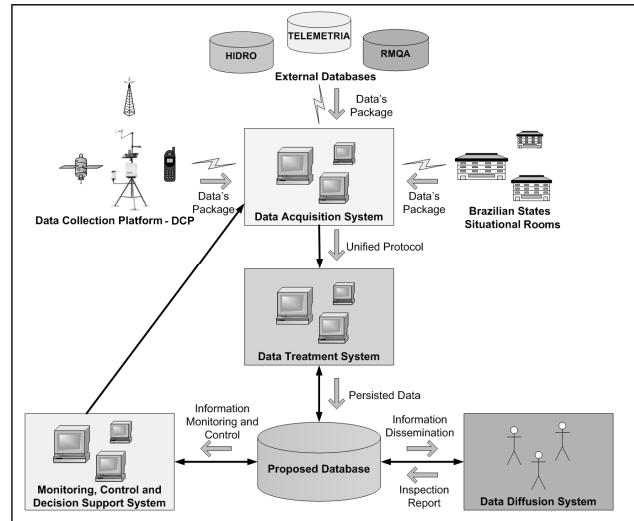


Figure 1. Information flow of ICA-MMH B SoS [3]

According to the information flow shown in Figure 1, besides hydrometeorological information collected from DCP, there is a need for interoperability among the Data Acquisition System, External Databases, and Brazilian States Situational Rooms.

On the other hand, the ICA-MMH B Project must also interact through Web Services (WS) with others ANA systems. Some examples are WS for providing security access, geographic data supply, and others.

Considering this scenario, an approach based on UML models has been adopted for the development of the ICA-MMH B Project and its systems shown in Figure 1.

In this approach, these models are transformed into deployable components for different platforms and technologies, by using the AndroMDA Framework, adopted in this project. Physical data models, source codes, and documentations are some examples of these components.

4. The Development Approach

This section presents the application of a development approach adopted in the ICA-MMH Project.

Figure 2 shows an overview of the used development approach.

This approach is divided into the following four phases, based on MDA: Phase 1 - Modeling; Phase 2 - Stereotypes and Properties Setting; Phase 3 - Documentation; and Phase 4 - Script Generation. It is also supported by the AndroMDA Framework of resources and by a suitable development environment.

Next sections describe this approach in more details.

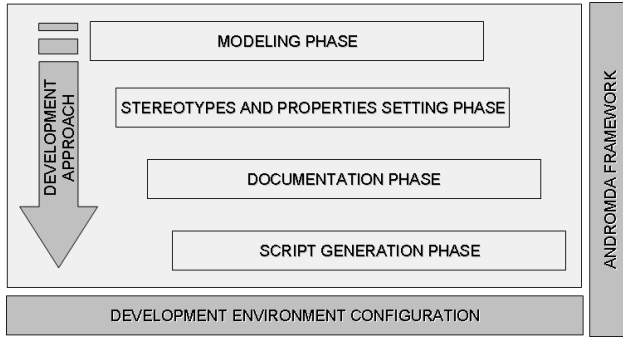


Figure 2. Used development approach overview

4.1 Development Environment Configuration

The proposed architecture for the ICA-MMH B Project is comprised of the following three layers: presentation, business and persistence [7]. The main technologies used on each layer were, respectively, Java Server Face – JSF, Spring, and Hibernate.

In order to support this architecture, a development environment was configured based on commercial and open source Computer Aided Software Engineering (CASE) tools.

One of these used tools was the IBM Rational Software Architect (RSA) 6.0, provided by its academic agreement with ITA. This tool was used mainly in the Modeling Phase and in the Documentation Phase.

In order to transform UML models built on the RSA 6.0 tool into source codes, it was adopted the AndroMDA Framework 3.4., and the generated source codes were successfully implemented by using Eclipse 3.4.

Finally, the Database Management System chosen for implementing the obtained physical data model was the Oracle 11g Spatial required by the ANA technical team.

4.2 Phase 1 - Modeling

In this phase, a class model was built from a detailed and specific analysis of an Integrated Heterogeneous Hydrometeorological database previously built using relational modeling [3]. Throughout a hard work process, each entity from this database was carefully mapped to an object in order to build an appropriate object-relational model.

This mapping process included the relevance of each object for the business, its abstraction level, and relationships between objects. In the first activity of the mapping process, several interactions with the ANA technical team were necessary in order to clarify object meanings within the project context.

According to the best practices of traditional object-oriented approaches, some definitions had to be made over the obtained class model: required attributes for each

object; attributes visibility; instance level for relationships (by considering associations, aggregations, compositions, and external links); and class level for relationships (by considering generalization and realization). For each relationship, multiplicity and navigability were specified.

A design pattern was also used to compose the class model elements nomenclature for the physical data model requested from the ANA technical team. The composition rules for this pattern are shown in Table 1.

A class model fragment used to exemplify this design pattern is presented in Figure 3.

Table 1. Rules for Nomenclature Pattern

Class	
Composition Rule	Project Name Trigram + TB + _ + Relevant Name of the Class
Example	ICATB_LABORATORIO
Attribute	
Composition Rule	Class Name Trigram + _ + Data Type Trigram + _ + Relevant Name of the Attribute
Example	lab_id, lab_nu_cnpj, lab_tx_razaosocial
Data Type Trigram is optional.	
Relationship	
Composition Rule	Class Name Trigram (From) + _ + Class Name Trigram (To) + _ + Attribute Name
Example	rpl_lab_id

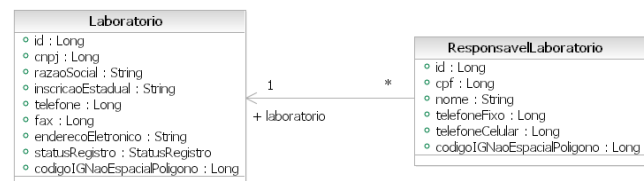


Figure 3. A Class Model Fragment

4.3 Phase 2 - Stereotypes and Properties Setting

In a model driven development, the UML models should be set with some properties which define an expected behavior for the generated code.

According to AndroMDA guidelines, these required properties are known as stereotypes and tagged values. The tagged values are grouped into stereotypes and applicable on specific UML elements.

Considering the adopted development environment, the stereotypes and tagged values are available due to the integration between RSA 6.0 and AndroMDA 3.4.

In this ICA-MMH B Project, the stereotypes and tagged values were applied over classes, attributes and relationships [7].

The stereotypes and tagged values applied over classes - are described below:

- **<<Entity>>**: Used when the class should be persisted as a physical table in the database. The “andromda persistence schema” tagged value was applied to define the schema of the physical table;
- **<<PersistentClass>>**: Used for persisted classes. The “andromda persistence table” tagged value was applied to define the name of the physical table; and
- **<<Manageable>>**: Used to generate CRUD (Create, Read, Update and Delete) source codes.

The stereotypes and tagged values applied over attributes - are described below:

- **<<Identifier>>**: Used when the attribute is the class identifier (optional);
- **<<PersistentProperty>>**: The “andromda persistence column” tagged value was applied to define the physical name of the attribute;
- **<<Unique>>**: Used when the attributes have unique values; and
- **<<PersistentAttribute>>**: The “andromda persistence column length” tagged value was applied to define the maximum size of the attribute.

The stereotypes and tagged values applied over relationships - are described below:

- **<<PersistentProperty>>**: The “andromda persistence column” tagged value was applied to define the physical name of the foreign key; and
- **<<PersistentAssociation>>**: Used for many-to-many relationship. The “andromda persistence table” tagged value was applied to define the name of the physical table related to the relationship.

Figure 4 shows an example of some used stereotypes over a class and its attributes.

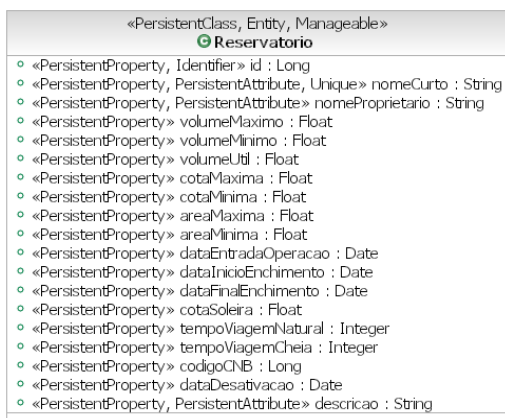


Figure 4. Used stereotypes over a class

Although the AndroMDA Framework had provided almost all properties to define the expected behavior for

the generated code, it was necessary to extend its properties by creating a new stereotype called “Persistent End Association”.

This stereotype, applied over relationships, was used to define a unique group including foreign keys, instead of including only attributes of the owner class.

4.4 Phase 3 – Documentation

This documentation phase was divided into two steps: the documentation of each model element, and the automatic generation of class model documentation.

On the first step, the relevant information about the elements of the model was registered on the modeling tool. This was done through periodic audio conferences involving both ITA and ANA technical teams.

These interactions were considered an important activity of this phase in order to clarify some questions about the business domain.

The first documentation report was obtained by using the report generator resource of the RSA 6.0. Even though this documentation is presented in a web project, on the javadoc format, and it is suitable for a class model, some specific information about a data dictionary (physical model) is not provided. In this case, specific information may include: the physical name of classes and attributes; foreign keys of each physical table; and the content of used tagged values.

In order to fulfill this need, an automated process for generation this complementary information was developed through a Hibernate cartridge customization for the project's scope, using a template language, called Apache Velocity [8], supported by Eclipse 3.4.

Throughout this phase, a considerable effort of time was spent with improvements in the UML model due to a continuous maturity related to the business domain of the ICA-MMH B Project.

4.5 Phase 4 - Script Generation

The Script Generation was the quickest phase of this development approach mainly because all necessary configurations were already done with the complete execution of previous phases.

Generated outcomes were the Data Definition Language (DDL) Script of the physical model and the CRUD source codes on JSF for the manageable classes.

An example of this mapping process from a class to a DDL Script is shown in Figure 5.



```

create table bdhr.ICATB_CARACTERIZACAO_VHM (
  cvh_id NUMBER(19) not null,
  cvh_tx_periodicidade VARCHAR2(100) not null,
  cvh_nu_intervalo NUMBER(10) not null,
  cvh_ic_onstream NUMBER(1) not null,
  cvh_tx_observacao VARCHAR2(4000),
  cvh_st_registro VARCHAR2(255) not null,
  cvh_cd_igpontopoligono NUMBER(19),
  cvh_cd_igpontolinha NUMBER(19),
  cvh_cd_iggeometria NUMBER(19),
  cvh_met_id NUMBER(19) not null,
  cvh_mto_id NUMBER(19) not null,
  cvh_sto_id NUMBER(19) not null,
  cvh_unm_id NUMBER(19) not null,
  cvh_est_id NUMBER(19) not null,
  cvh_pcd_id NUMBER(19),
  cvh_ogu_responsavel NUMBER(19) not null,
  cvh_ogu_operador NUMBER(19) not null,
  primary key (cvh_id)
);
  
```

Figure 5. Mapping from a class to a DDL script

For the same class, Figure 6 presents an example of a JSF Human Machine Interface (HMI) from the generated CRUD source code shown in Figure 5.

Caracterizacao Variavel Hidrometeorologica

Periodicidade:

Intervalo:

On Stream:

Observacao:

Status Registro:

Codigo IGPonto Poligono:

Codigo IGPonto Linha:

Codigo IGGeometria:

Meio Transmissao:

Metodo Obtencao:

Status Operacional:

Unidade Medida:

Estacao:

Pod:

Responsavel:

Operador:

Figure 6. A JSF HMI from the CRUD source code

5. Main Results

This section summarizes the main results obtained from the application of this MDA development approach.

The main outcomes from this MDA development approach was produced by four members from the ITA technical team in a collaborative partial time work with

two members of the ANA technical team, for about 1.200 hours of during 3 months.

Some of the major findings during this process include: a logical model, represented by a class diagram; a physical model, represented by the DDL Script; and CRUD source codes. Table 2 shows the main resulting figures.

Table 2. The Main figures from applying this MDA development approach

Logical Model	
Classes	99
Enumerations	10
Attributes	680
Relationships	129
Physical Model	
Tables	102
Fields	640
CRUD Source Codes	
JSF pages	99

Figure 7 presents an example of a fragment from UML logical model documentation automatically generated by using the RSA 6.0.

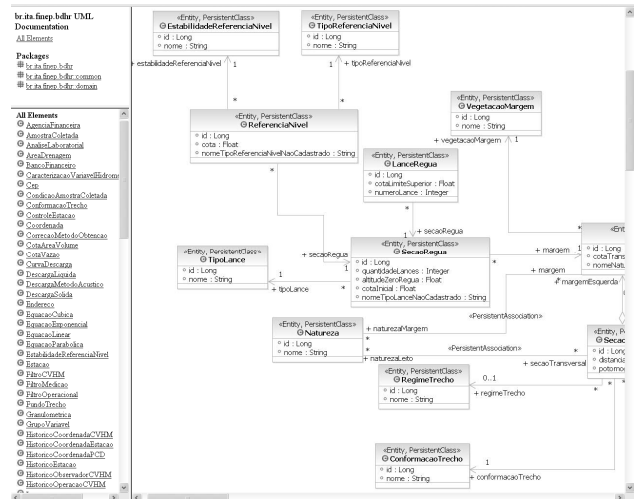


Figure 7. A Fragment of the UML logical model documentation

Another major finding from this MDA development approach application was the generation of the complementary information for physical model (data dictionary). This improvement and the creation of “Persistent End Association” stereotype was made available for the AndromDA Framework community, even though they were specifically built for the ICA-MMH B Hydrological Project.

Figure 8 shows an example of the automatically generated data dictionary documentation.

Classe Laboratório

Schema: bdhr

Nome físico da tabela: ICATB_LABORATORIO

Documentação:

Esta entidade armazena os dados cadastrais dos Laboratórios

Atributos				
Nome Lógico	Nome Físico	Documentação	Tipo do Dado	Domínio do Dado
° id	lab_id		Long	NUMBER(19)
° cnpj	lab_nu_cnpj	CNPJ do Laboratório de Análise da Qualidade de Água.	Long	NUMBER(19)
° razaoSocial	lab_tx_razaoSocial	Razão Social do Laboratório.	String	VARCHAR2(100)
° inscricaoEstadual	lab_tx_inscricaoEstadual	Inscrição Estadual do Laboratório.	String	VARCHAR2(50)
° telefone	lab_nu_telefone	Telefone do Laboratório.	Long	NUMBER(19)
° fax	lab_nu_fax	Fax do Laboratório	Long	NUMBER(19)
° enderecoEletronico	lab_tx_enderecoeletronico	Endereço Eletrônico do Laboratório.	String	VARCHAR2(255)
° statusRegistro	lab_st_registro	Define qual é o status do registro em questão: excluído, importado, importado repetido, novo, permanente e transferido.	StatusRegistro	VARCHAR2(255)
° codigoIGNaoEspacialPoligono	lab_cd_ignaoespacialpoligono	Referência a um código associativo de um objeto não espacial para um objeto espacial.	Long	NUMBER(19)

Associações			
Nome Lógico	Nome Físico	Documentação	Domínio do Dado
∧ variavelHidrometeorologica	lvh_vah_id		NUMBER(19)
∧ endereco	lab_end_id		NUMBER(19)

Figure 8. An example of the automatically generated data dictionary documentation

6. Conclusion

By applying MDA development approach on an actual project, it was possible to verify the main advantages of using MDA. In this case, this application has improved the development performance related to automatic generation of artifacts such as logical and physical models, UML documentation, and CRUD source codes.

UML models had to be built by advanced developers due to some specific settings that must be considered during the modeling phase.

In this case, the use of the AndroMDA Framework reduced this effort once it proposed an array of defined stereotypes and properties.

Even though a lot of artifacts are automatically generated, developer's interferences are recommended in this process mainly for the improvement of the CRUD pages in order to attend user aspects like usability.

Considering the open source community of the AndroMDA Framework, this work had represented a relevant research contribution because it reported an actual and much realistic experience of its use by proposing a complement for its already existing and defined stereotypes and properties.

For future works, considering the results reported on this paper, the authors suggest the specification of this MDA development process, as adopted on this ICA-MMH B hydrological project.

7. Acknowledgments

Authors of this paper would like to thank: the Brazilian Aeronautics Institute of Technology (ITA), for its technologic and scientific development incentives; the Brazilian National Water Agency (ANA), for the

opportunity of participating in the ICA-MMH B Project; the Brazilian Agency of Research and Projects Financing (FINEP), for this contract opportunity; the Casimiro Montenegro Filho Foundation (FCMF), for the availability of the infrastructure and scholarships; and the IBM for supporting its modeling tool during the academic agreement with ITA.

8. References

- [1] OMG, Object Management Group. Model Driven Architecture (MDA), 2009. Available at: <http://www.omg.org/mda/>
- [2] Valverde, F., Valderas, P., Fons, J., and Pastor, O. "A MDA-based Environment for Web Applications Development: From Conceptual Models to Code". In International Workshop on Web-oriented Software Technology (IWWOST 2007), in conjunction with ICWE 2007, Como, 2007, Italy
- [3] Braga e Silva, Gláucia; Romano, Breno Lisi; Campos, Henrique Fernandes de; Vieira, Ricardo Godoi; Cunha, Adilson Marques da; Dias, Luiz Alberto Vieira. "Integrating Amazoniac Heterogeneous Hydrometeorological Databases". IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2009
- [4] Mizuta, S. Huang, R. "Automation of Grid Service Code Generation with AndroMDA for GT3". 1st International Workshop on Information Networking and Application (INA), Taiwan, China, 2005, pp. 417 – 420
- [5] Bezivin J., Hammoudi S., Lopes D., Jouault F., "Applying MDA Approach for Web Service Platform". Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conf (EDOC 2004), 2004
- [6] Gharavi, V., Mesbah, A., and van Deursen, "A. Modeling and generating Ajax applications: A model-driven approach". In Proceedings of the 7th ICWE International Workshop on Web-Oriented Software Technologies (IWWOST'08), 2008, pp. 38 – 43
- [7] AndroMDA. AndroMDA Framework, 2009. Available at: <http://www.andromda.org/>
- [8] Apache Software Foundation. Apache Velocity, 2009. Available at: <http://velocity.apache.org/>