Comparative analysis of MDA tools

Krzysztof Pietraszek[1]
[1] Institute of Computer Science,
University of Natural Sciences and Humanities,
3 Maja Str. 54, 08-110 Siedlce, Poland

Abstract: This paper presents the standard assumptions Model-Driven Architecture (MDA) and describes several tools supporting MDA approach.

Key words: Model-Driven Architecture (MDA), MDA tools.

1 Introduction

For some time we see increasing role of information systems in various areas of life and business. Software is used for learning, fun, work, and many other activities. They are used as a management support department or the whole enterprise. With the software you can control various devices beginning from simple computer and ending specialized robots.

Along with the development of information systems has increased demands placed on them. Today's systems must be fully customized functionality to meet the needs of future users. An integral feature of their quality must be appropriate and effective action. In addition, the entire production process and later maintenance of systems should be efficient and carry with them the lowest costs.

Currently the development of information systems is often compared with the development of computer hardware. In the last few years have seen rapid progress in hardware technology. Manufactured processors are fast but the external memory are more and more capacity. Otherwise is the case with software. Its complexity, frequent changes in requirements at different stages of formation and the requirements of parallel co-operation with other systems, make the manufacturing process is much more difficult and more expensive. This makes the result of many project activities is a product with less functionality and significant defects, leading in many cases, the failure of the project. Another factor aggravating the increased probability of failure is exceeding the cost and time production systems.

Existing situation has resulted in the search began for other methods of software development based on existing traditional methods. The resulting new technique called modern object-oriented approach to promote the reuse of software component approach, etc. Methods of Modern attach great importance to the maintenance and upgrading of existing systems. Provide better ways and methods of pro-

ducing reliable applications. Solve common problems in designing and implementing a validated commonly acceptable patterns of software. Despite such progress, the activities of manufacturing software based on modern technology still does not guarantee the success of projects. The cost of production and the amount of work has remained high. This was largely due to the lack of concrete solutions to support or even automate the process of software development.

The answer to these problems initiated CASE tools that support building systems in different phases of software life cycle. By functionality tools include the specification of requirements, design, code generation frame, preparation and testing, etc. In addition to the activities connected with the production of CASE applications also support the creation of project documentation and project management. The presence of a CASE tool brought much benefit and facilities that relieve designers in the software production process. However, despite a big step forward, some problems remain unresolved. Amount of work associated with the creation of a full application code was still high. In addition, cost of maintaining systems and the ability to integrate with other software left much to be desired.

In view of the situation OMG consortium has set up a new standard for Model-Driven Architecture (MDA), whose task is to solve the above problems by substantially automating the manufacturing process.

2 Model-Driven Architecture

Standard Model-Driven Architecture [1][2][3] was developed by a consortium of OMG (Object Management Group). The general idea of MDA is to separate the functional specification of system solutions to specific technology platforms. The essence of MDA is to build a model of an application-independent platform, an implementation - PIM (Platform Independent Model). Then the PIM model is transformed to an implementation platform-specific model - PSM (Platform Specific Model called) with the appropriate conversion template for the selected technologies. The PSM model is automatically generated code for the platform system. By using PIM model, we can create applications for different platforms, modifications to the system, system integration, etc. At the level of PIM made any action related to the maintenance of the system. Model-independent platform allows for rapid change of technology platforms, allowing the company is not permanently associated with specific solutions. Additionally, it can also be used CIM model that presents business processes in an enterprise important for the modeled system. MDA uses multiple standards such as UML, MOF, CWM, XMI developed by OMG. By the idea of MDA significantly reduce manufacturing costs and maintenance of systems and their prospective owners are not dependent on technology platforms.

MDA is closely associated with the presence of tools to support its implementation. Most MDA tools gives a big easier in the form of ready-built definition of transformation and code generation templates for selected technologies such as Java / EJB, CORBA, Web Services, C # /.NET. Actually, there are many commercial tools and open source. But these are not a universal tool for every application and implement more or less level the concepts of MDA..

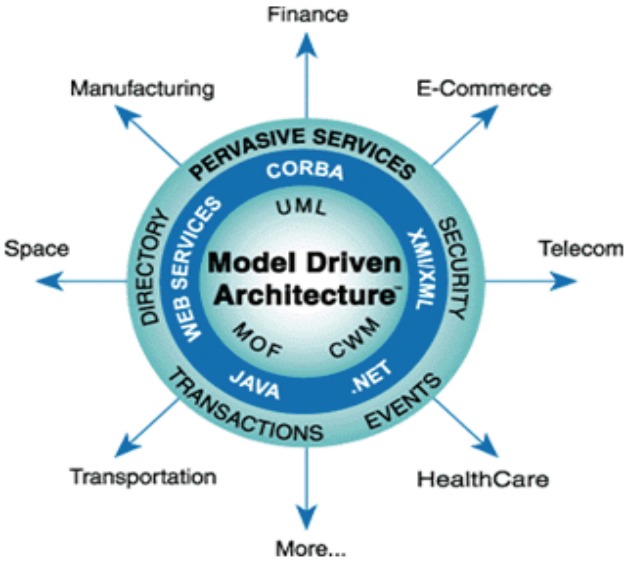

Figure. 1. MDA architecture [4]

2.1 Types of models in MDA

Model is a key concept in the standard MDA. The model is a description of the part or the whole system in a well defined language such as UML. Models are transformed into other models or in the application code. Transformations are made possible by a well-defined transformations.

CIM [5] provides all or part of a business enterprise. Languages that are used to model the business include vocabulary allows the designer to specify business processes and the relationship between processes, etc. Currently, this language is such UML. The business model is not necessarily talking about software used by the company. Whenever a part of the business of the company is supported by the software defined logical model describing the software is already written. Business and software models describe quite different categories of systems in the real world. System requirements are delivered exactly the business model. Automated CIM to PIM transformation is impossible, because the choice of which parts of the CIM will be supported by software, is function of people.

Typically, all projects in the MDA standard begins with the creation of ***PIM*** model expressed in UML. There are two levels of PIM: basic PIM and PIM marked. Marked PIM is based on the primary PIM and also includes a platform-independent aspects of technological behavior. The purpose of PIM is to present the structure and dynamics of the system in a platform-independent. PIM model is mainly expressed in the form of UML class and state. PIM models on the second level contain markings related to technology, although the elements for specific platforms such as J2EE or. NET are absent. These models are called models marked (called brand models). Dimensions that are identified at this PIM level, i.e. session, the object is common to

all platforms. Additional terms may be: persistence, transactions, security, configuration level, etc. By adding these concepts to the PIM model, additional information is hidden, permitting a more precise model transformations to PSM. Indications are transmitted in the form of UML extensions, i.e. stereotypes, marks

After creation, the PIM model is stored in the repository in the MOF standard and provides an input parameter for the transformation of the ***PSM*** model. To express PSM models are specializations and extensions to the UML. UML Profile is a standardized group of extensions that make up the stereotypes and marks that define the UML environment matched to particular use, such as modeling for a specific platform such as the UML Profile for CORBA, UML for EJB. Before performing the transformation of PIM → PSM selected target platform or platform on which we want to transform the model. During the transformation properties and various additional information that is contained in the PIM model is a valuable source of transformation for the specific requirements of technology platform. Moreover, between PSM models arising from the same PIM can create special bridges that are used for communication. A good example is a system that is represented by a model PSM works with relational database provided by the second model PSM. Transformation of PIM to PSM is the most difficult step in the whole manufacturing process. The MDA tools are often produced PSM model requires a large number of fixes that could be converted to the application code. Now with the evaluation and transformation of UML profiles is practically possible to generate a complete model of PSM. There are four ways to transform PIM to PSM:

- manual,
- transformation using a defined profile,
- transformation that uses signs and patterns,
- automatic transformation.

An **implementation model** is generated code the system. Code generation is based on the PSM model, which is nearby at a level from the application code. Currently on the market we are quite diverse in terms of MDA tools with code generation. Some form a full code, possibly completed by hand to a small extent while others generate most of the code in the form of frameworks and methods for the programmer corrects the deficiencies. Transforming to the Code can apply to both applications and generation of SQL scripts for the target database system. The code is generated based on specific technologies included in the model, i.e. PSM JAVA / EJB, C # /. NET etc. In addition, the PSM model can represent a specific physical architecture of the system such as system components, code libraries, etc., which is included and considered in developing the code. Because current MDA tools support the production of Web-based systems mainly communicate with the database is in the code used is the division of three-layer:

- presentation layer: simple forms to add, edit, delete, which are created automatically, a large number of MDA tools generally do not create the GUI, the designer must himself create the appropriate IDE interface and connect the business logic,
- business logic layer: contains the code of services, technologies such as Web Services, etc.
- data access layer: uses Hibernate, DAO, etc.

2.2 Standard and technologies used by MDA

MDA approach uses multiple standards and technologies [1] that allow for system modeling, transformation, code generation, etc.. Just like MDA, the OMG is a consortium idea. Presented below are the most important standards and technologies used by the MDA.

MOF is a standard that defines a language for defining modeling languages. Standard is the basis for the four-levels models in MDA (Figure 2).

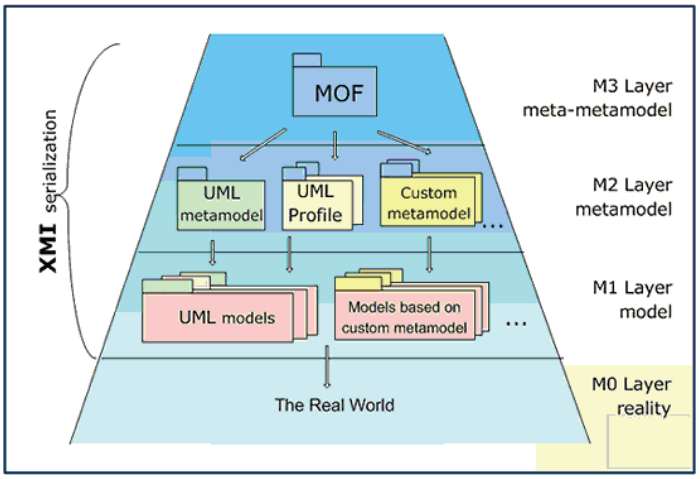

Figure 2. Four-levels models [6]

The lowest level contains the application data such as object instances at runtime, or specific records database tables. Level M1 is used for modeling systems that contain the system model: classes, model logical databases etc. The level of M2 shows metamodels that describe models. Elements of the level of M2 is defined as a DSL (Domain Specific Language) - modeling languages for a specific area. M2-level models are models such as UML, which includes more than one level and thus describes the data and metadata applications adequately for class diagrams and their components. At the level of M3 is the meta-metamodel, something like EBNF notation, which allows you to define the syntax of language and its. It is at the level of M3 is the MOF and can formulate their own metamodels. Examples are the products of MOF, UML and CWM. MOF definition contains interface specifications for the MOF repository. The interface allows to get information about the M1-level model of MOF-based repository. The interface is defined using the CORBA-IDL and is useful in many environments. Using MOF to define modeling languages we can define transformations between modeling languages. Because the transformations are defined in terms of modeling language requiring metamodels it can be applied to any model written in one of these languages.

QVT (Query, Views, Transformations) [2][7] is a standard for model transformation defined by the OMG consortium. Model transformation is the process of

transforming the model "Ma" consistent with the metamodel "MMa" in model "Mb" consistent with the metamodel 'MMb'. QVT defines a standard method of transformation of the source and target models integrates standard OCL.

OCL (Object Constraint Language) is a formal language used to describe constraints in UML models and MOF. The use of OCL invariants, we can define classes, initial conditions and final operation, the input and output attributes of classes. It does not force any action in the event of irregularities. Standard also allows for determining conditions and invariants in activity diagrams, state, sequence and collaboration. Using OCL expand the power of UML / MOF and allow the creation of a more accurate and detailed models. Each OCL expression always evaluates the value and describes what is the value, but never dictates how the expression should be calculated. OCL expressions are often mapped into programming languages, such as Java, C #. The use of OCL makes UML system model becomes more complete. In the context of MDA is to generate more precise and complete model of PSM from PIM model and a more complete code. With the MDA approach, OCL value increased significantly.

UML [8][9] is a standard modeling language at the level of M2 four-levels models. Almost all the models that are the essence of design models for UML. UML is integral to the MDA standard and is defined in MOF. UML is a formal language used by analysts, designers and developers. It is used to describe parts of the existing reality under which the system will be created. UML is often regarded as modeling language with four one perspective. The four perspectives provide a structure of the system at different levels of abstraction and detail and represents a perspective outside the system structure. We have the following perspective:

- perspective of a use case - explains the features of the system from the user perspective,
- logical perspective - presents architecture of the system modeled by the designer,
- implementation perspective - presents components of the system in terms of implementation,
- process perspective - presents a fragmentation of the system operations and the executive organs,
- implementation perspective - presents physical structure of the system.

UML is a graphical notation and offers a variety of diagrams used for modeling the structure and dynamics of the system. In the MDA approach, UML notation is used when creating the model, PIM and PSM imaging model, which is automatically generated from the PIM. Additionally, the UML defines the extensions in the form of stereotypes and marks that allow for precise modeling of systems and better transformations. In addition, there are many UML profiles for specific use such as CORBA, EJB. Another extension of the UML Action Semantics, which, combined with UML makes Executable UML. After that, the UML standard OCL is used, which allows transmission constraints on the elements of the models.

UML can also be used to design a data warehouse with the participation of CWM.

CWM (Common Warehouse metamodel) [10] is a standard designed to describe and manage the metadata associated with databases. Standard containers can

metamodeling data and thus determines their architecture. CWM is located at the four-layer M2 models (Figure 2) equally with UML, and this means that it is defined using MOF. CWM standard provides a universal approach to metadata about data sources, a target data transformation, analysis, operations and processes that create, manage warehouse data and provide information about its use.

XMI is recommended by the MDA standard for the exchange of metadata between repositories of different tools. Initially, metadata are exchanged using standard CORBA-IDL, which proved to be inefficient for sending large models. The XMI models are not written in the form of graphical notation, but in the form of text in XML format understood by the computer. Because the MOF is the primary standard, which defines standards for modeling such as UML or CWM, all models are imported to the MOF metamodels and saved in XML, which allows for a certain universality. XMI standard defines:

- rule schema definition (XSD) for the transformation of MOF metamodels based on XML schemas (XML Schemas),
- rules for creating XML documents with the encoding and decoding MOF based metadata,
- design principles for XMI based on schemas and XML documents,
- rules for importing XML DTD to MOF-based metamodel.

Objects recorded in XML documents using the appropriate tags and their attributes. You can also store links to other objects in the same document or separately. Each XML document has a defined template so that it is possible to validate metamodels. Using UML models can immediately save it to XML as the standard is equipped with the appropriate schemas documents. If you save the custom model to describe its metamodel in the MOF and then create XML schemas. The use of XMI is becoming increasingly popular due to its platform-independent XML.

3 Comparison of MDA to ols

MDA tools are those that reflect a greater or lesser extent, a series of processes leading to the generation of part or all software based on MDA standard. A key aspect of the characteristics of all these tools is the model transformation, which is the target model generation procedure based on the source model. In order to solve complex transformation model transformations should be clearly specified. Common specification QVT transformation is technology, which was described in one of these chapters. Transformation specification defines the behavior of the tools of transformation during transformation. Overall, the MDA tools, we can distinguish the transformation model to the model and the model code. MDA frameworks identifies four types of transformation:

- PIM to PIM - platform independent, lies in improving the model and is used to filter PIM model or specialization.,
- PIM to PSM - is used for the transformation of the model selected infrastructure performance,
- PSM to PSM - refine the model refers to the selected platform,
- PSM to PIM - convert a model for the selected platform for the model independent,

- PSM to application code - the program code is generated for a given platform (e.g. Java or database schema in SQL) in text form.

Figure 3 presents two types of transformation:

- model to application code - vertical arrow presents transformation leading from M1 to M0 layer,
- model to model - transformation illustrates the horizontal arrow leading from the source to the target model in layer M1.

Metamodels layers are located vertically from the layer M0 to M3, while the MDA is presented in the form of source and target (meta) models, avoiding any reference to models of PIM, CIM and PSM. Such a move is intended to transforming the basic classification model to code, and model to model.

In this chapter, are analyzed different MDA tools, a variety of implementations and features offered. Each realizes MDA in its own way using various technologies and terminology. So far there is no tool that covers all the requirements of standard MDA. There are two reasons:

- no unified standard for the specification transformation or transformation model,
- absence of demand tools to support MDA.

Comparison of MDA tools [11] will be presented in a table containing a summary of the following criteria: manufacturer, model for model transformation and model to code, available on the market, support for development platforms, development life cycle (levels of abstraction). Previously, each of the tools will be described in terms of functionality, etc.

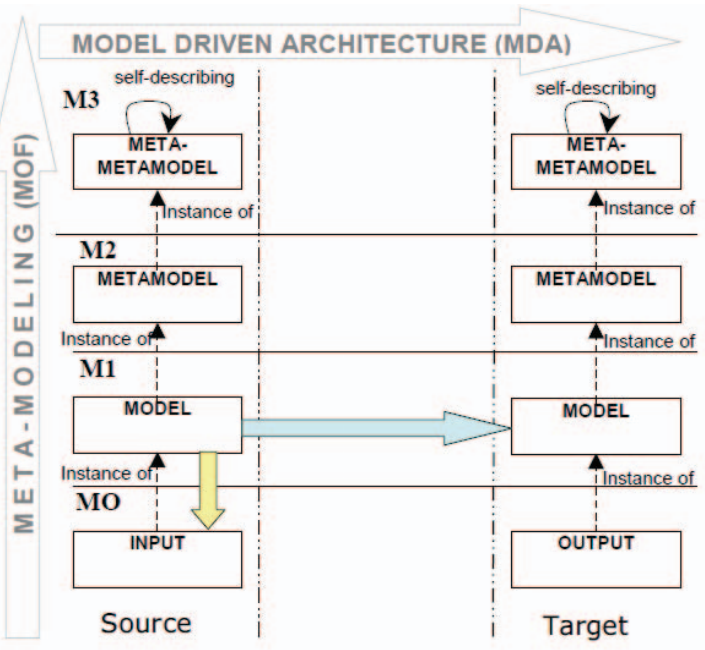

Figure 3. Model-to-code and model-to-model transformation tools

3.1 OptimalJ

OptimalJ [12] is a tool developed by Compuware, which provides methods for implementing the MDA approach, starting from the level of PIM. The tool supports UML 2.0 and allows for modeling the static structure and dynamics of the system using different diagrams. OptimalJ allows you to import and export of models using XMI standard. The product provides three levels of models: model fields, model application and model code. Model fields coincides with the PIM model and consists of two models: the model of classes and services. Model class includes a static structure of system and application data, while the service model is used to describe the system behavior. Model classes is in the form of class diagrams with each other forming different compounds such as associations, compositions, aggregations. Each class must have a defined primary key, which is later used to create the database. Service model is used for the specification of system behavior independent of implementation.

Application model is a model of PSM, which is represented by three layers: presentation, business logic and database. The presentation layer provides a graphical interface system, which may subsequently be reflected in the technologies JSP, Struts, etc. Model application business logic is represented by the EJB or DAO (Data Access Object). While the database layer represents the logical database schema, which is implemented on platforms such as MS SQL, Oracle, IBM DB2, MySQL.

The application code is generated from the PSM model through appropriate transformations. OptimalJ offers several tools for testing and debugging of application code created. In addition, the product provides features such as configuration, application servers JBoss, WebLogic, IBM Websphere, Oracle Applications Server. OptimalJ offers two types of patterns for the transformation: patterns of technology and implementation. Patterns of technology are used to transform PIM to PSM, and the patterns of implementation of the transformation of PSM in the application code. Patterns in OptimalJ are based on the language of TPL (called Template Pattern Language), which was specially created for this purpose by Compuware. With TPL can write your own definitions of patterns, but the tool provides a ready-made patterns of transformation for a specific J2EE.

In addition to the functionality described OptimalJ allows you to:

- create metamodels, provides ready-made according to MOF metamodels,
- use of OCL and XMI standards,
- web page design and navigation,
- provides plug-ins for the IDE, i.e. Eclipse, NetBeans,
- project support team.

JOptimal also provides a mechanism for active sync, which updates the application model and the source code with the changes incorporated in the model area. The functionality of the generated code and distinguishes between extensions added by the developers, and provide the developer makes changes in the model is a model and the code will be automatically synchronized. Active synchronization ensures consistency between code and models, resulting in application is easier to manage

and maintain. Active Synchronization is an intelligent process with which the parts of the applications that are subject to change and those that are not.

3.2 ArcStyler

ArcStyler [13] is one of the leading tools on the market of software as standard MDA. It supports the full life cycle of software applications developed in Java/J2EE and architectures. NET. ArcStyler was developed by Interactive Objects. The tool has three ways of modeling the system:

- creating models from scratch using UML and OCL,
- import existing models with the XMI standard,
- creating models based on the code (reverse engineering).

PIM in ArcStyler is represented in the form of classes that represent business objects in a manner independent of the technology platforms. The tool does not provide support for the PSM model. The idea is to use special markings PIM components that store information about the target platform. With this information, creating a platform-specific code. Model code is represented in the form of packages with source code provided as components.

ArcStyler cartridge uses the term MDA model transformation. It supports creating and editing cartridges MDA to define their own rules of transformation. The language used to build the container is an object-oriented MDA JPython, which consists of the Java and Python. The tool also has a MDA engine which performs transformations based on the selected cartridge. Each container MDA introduces its own specific characteristics that represent their capabilities, which are performed and controlled by the engine MDA. In addition, the product provides a graphical tool to create ready-made cartridges and containers for several technologies and platforms such as Java, J2EE,. NET.

3.3 Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) [14] is an open source framework for developing model-driven applications initiated by IBM. In EMF, while source model can be modeled by UML and XML Schema, the target model generated as Java classes. It creates Java code for graphically editing, manipulating, reading, and serializing data based on a model specified in XML Schema, UML, or annotated Java. EMF is the basis for many of the tools within IBM WebSphere Studio and Eclipse projects. It contains the elements necessary to keep the development focus on the model, and not on the implementation details. The main focus areas are: generation of models that support customization, notification, referential integrity, and other essential features; generation of customizable model editors; and default serialization. Individual tools, like the serialization or the graphical editor, may be pulled out and used independently, but the full power comes when all areas are used together. EMF consists of three fundamental concepts:

- **EMF** - The core EMF framework includes a meta model, called Ecore for describing models and runtime support for the models including change no-

tification, persistence support with default XMI serialization, and a reflective API for manipulating EMF objects generically.

- **EMF.Edit** - The EMF.Edit framework includes generic reusable classes for building editors for EMF models. It provides content and label provider classes, property source support, and other convenience classes that allow EMF models to be displayed using standard desktop (JFace) viewers and property sheets. And it also provides a command framework, including a set of generic command implementation classes for building editors that support fully automatic undo and redo.
- **EMF.Codegen** - The EMF code generation facility is capable of generating everything needed to build a complete editor for an EMF model. It includes a GUI from which generation options can be specified, and generators can be invoked. The generation facility leverages the JDT (Java Development Tooling) component of Eclipse. Three levels of code generation are supported:
  - **Model** - provides Java interfaces and implementation classes for all the classes in the model, plus a factory and package (meta data) implementation class.
  - **Adapters** - generates implementation classes (called ItemProviders) that adapt the model classes for editing and display.
  - **Editor** - produces a properly structured editor that conforms to the recommended style for Eclipse EMF model editors and serves as a starting point from which to start customizing.

All generators support regeneration of code while preserving user modifications. The generators can be invoked either through the GUI or from a command line.
In addition to code generation, it provides the ability to save objects as XML documents for interchange with other tools and applications. Models can be created using annotated Java, XML documents, or modeling tools like Rational Rose, and then imported into EMF. The code generator turns a model into a set of Java implementation classes. These classes are extensible and able to regeneration - user can modify them by adding user-defined methods and instance variables. When the model changes, user can regenerate the implementation classes and user modifications will be retained. This works both ways - changes in the Java code can be used to update the model.

3.4 AndroMDA

AndroMDA [15] is an extended frame generator system which is used in the paradigm of MDA. It is worth noting that this is an open source tool and therefore has no licensing restrictions.

Design of PIM model is done by using an external modeling tools such as Magic Draw UML, ArgoUML, Poseidon or Enterprise Architect supports standard exchange models - XMI. The PIM model consists of structure and behavior of the system modeled using class diagrams, state diagrams and use case diagrams. Additionally, you can use to define the standard OCL constraints for elements of the models. The resulting PIM model is transformed to able to implement the compo-

nents of the platform, which can be a J2EE or. NET. AndroMDA as ArcStyler has special containers with the rules of transformation models for specific technologies. Cartridges are now available to support the transformation of: EJB, Spring, Struts, Hiberante, Axis, jBPM, Web Services, XSD, Java, C #, NHibernate, NSpring, PHP, HTML. In addition, AndroMDA has the ability to build their own containers or adapting existing ones. It is used in the Java language and the language of ATL (Atlas Transformation Language called) based on modeling standard QVT transformation.

In addition, by using a template tool allows you to generate various types of text output for the source code, scripts, databases, writing O / R mapping in the configuration files, web pages, etc. The templates are based on the well known template engines such as Velocity and FreeMarker. AndroMDA doesn't operate as a separate tool, it offers plug-ins for specific development environments (IDE) such as Eclipse and Visual Studio. Created on the PIM model code is located directly in the IDE, so you can compile and test the system. In addition, you must install the Maven, which is used for construction and project management.

Most of the applications built using AndroMDA is a web-based applications applying the three-layer architecture - the presentation layer, business logic layer and data access layer. The individual layers are implemented by different technologies depending on the chosen platform.

3.5 Integranova Model Execution System

The Integranova Model Execution System (MES) [16] is the commercially available software system that generates complete applications from software models. It supports model-to-code transformations. The source model is modeled via Modeler (very similar to UML) and the target model is

generated as Visual Basic or Java, JSP codes. Unlike other software solutions, Model Execution System isn't limited to building embedded systems (that lack GUIs), database infrastructure, or integration plumbing. Instead the Integranova Model Execution System takes class models, functional models, and presentation models and creates a completely functional and executable software application.

There are two main components that make up the Integranova Model Execution System, namely the Modeler and the Transformation Engine. The Modeler captures business objects as classes with attributes and connects them via relationships. Integranova Modeler allows analysts and developers to capture actual business logic, business rules, and system constraints without having them write any code. These rules and constraints are then translated by the Transformation Engine into language and architecture-specific code that is ready for compiling and deployment. The Transformation Engine (a collection of code generators) is run as a web service that allows clients to send XML definition files for models created in the Modeler. With a simple-to-use interface, clients select the kind of architecture and language support they desire for the application and upload the definition file. From there the web service collects the file, runs it through the appropriate code generator and mails back the code to the client.

In Integranova Model Execution System can be make Client/Server application by mixing VB and COM+ or internet application with JSP and EJB or even extranet application combining CF with VB and EJB.

3.6 Enterprise Architect

Enterprise Architect (EA) [17] was developed by Sparx Systems supports a full system design standard MDA. EA has its own editor for modeling systems using UML 2.1 standard and provides support for the exchange of models between repositories of different design tools through XMI. It also offers several additional tools to support steps in the creation of systems in the MDA standard, such as:

- possibility complete transformation PIM to PSM,
- synchronization of changes in the PIM and PSM,
- the use of templates transformation, UML Profiles, UML models during the code generation,
- possibility mapping contained in the code also changes in the models (reverse engineering),
- define their own transformations.

In addition, the tool has built-in ready for transformation definitions EJB, Java, C #, XSD, CORBA. EA also supports the transformation of the following languages implementations: C + +, VB.NET, VB, PHP. The main component of MDA in the EA is a template-based transformation engine, which generates a PSM model based on the PIM source model. MDA transformation templates are similar to the code generation templates and have the same syntax. EA tool allows you to generate DDL scripts, and create a logical model of the database via ODBC on the basis of physical implementation. EA supports these steps for Oracle, MS SQL, MS Access, PostgreSQL and others.

3.7 Objecteering

Objecteering tool Objecteering Software's [18] support among other standard MDA. The program provides a set of mechanisms for the production of a model driven application architecture. Objecteering manages the consistency of models and tracking of changes to the software life cycle from requirements analysis and design through code generation, testing and deployment. Objecteering consists of several modules, which are connected together into one complete tool gives MDA. These modules include Requirements, UML Modeler, UML Profile Builder, MDA Modeler, Java Developer, C + + Developer, C # Developer, SQL Designer.

The module "Requirements" allows you to collect user requirements in text form by using a dedicated editor or by importing from Word documents. In addition to creating a dictionary for the business area containing all possible terms. Based on the requirements and the dictionary, you can create reports and documentation according to specific templates in HTML or Word.

UML Modeler module enables full system modeling using UML version 2.0 which allows you to create a full PIM. Modeling uses the building blocks based on the requirements and the dictionary, which are stored in the repository. When design-

ing, you can use the existing UML profiles or created by yourself. Based on UML models can automatically generate documentation in HTML or RTF / Word. In addition, there XMI mechanism that enables the exchange of models between different tools.

UML Profile Builder allows the construction of UML profiles to define the rules of transformation. UML Profiles are formulated in a special editor, and using a special language similar to Java JLanguage.

MDA Modeler has a dedicated graphical tool for creating the definition of the code generation, documentation generation, and model on the target technology (PSM) of the PIM model. By using tools such as Java Developer, C + + Developer, C # Developer application code is created based on the PSM model. Database schemas for a particular platform are built by the SQL Designer. Available platforms are: Oracle, Sybase, MS SQL.

3.8 Summary

Currently on the market there are many solutions to commercial and open source approach to support MDA. Their functionality varies from advanced to simple carrying a minimum of assumptions MDA. The above tools support different programming languages, technologies and platforms. Most of them have their own definition of transformation languages, and templates. Some create an independent environment that supports MDA and others are in the form of concrete IDE plugin, and require the support of specific tools such as UML modeling, e.g. Magic Draw. Moreover, most of these programs is mainly dedicated to designing Web applications.

Having sketched the image and functionality of particular tools can provide the most important requirements for an application to support the process of software development in the MDA standard:

- presence of UML modeling tools,
- opportunity to transform PIM → PSM → code of the application,
- the ability to define their own transformations,
- artifacts repository,
- exchange of models between different tools using XMI,
- UML profiles for a wide range of technologies such as EJB, CORBA, UML models used in the transformation,
- support for as many programming languages, technologies and databases,
- generation of documentation based on models,
- synchronization changes on each level models.

Below in tabular form is presented a general comparison tool based on criteria such as manufacturer, the ability to transformations the model into the model, the ability to transformations the model to code availability (license), support for development platforms, databases, UML modeling tool, etc., development life cycle.

Table 1. Summary - part 1

MDA tools	Vendor	Model to model	Model to code
Optimal J	CompuWare	X	X
ArcStyler	Interactive Objects	X	X
Eclipse Modeling Framework	IBM		X
AndroMDA	AndroMDA.org	X	X
Integranova Model Execution System	Sosy Inc.		X
Enterprise Architect	Sparx Systems	X	X
Objecteering	Objecteering Software	X	X

Table 2. Summary - part 2

MDA tools	Availability	Support for	UML	Development life cycle
Optimal J	Developer Edition, Professional Edition, Architecture Edition	J2EE, Oracle, DB2, MSSQL, MySQL	X	Domain Model, Application Model, Code Model
ArcStyler	Enterprise Edition, Architect Edition	J2EE, .NET	X	Application Model, Define Target Technology, Transformation and deployment
Eclipse Modeling Framework	Open Source	J2EE	X	Specify EMF Model, Generate Code, Customize generated code, Regenerate
AndroMDA	Open Source	J2EE, .NET, DB2, Firebird, MSSQL, MySQL, Oracle, PostgreSQL		Application Model, Code Model
IntegraNova Model Execution System	Integranova MES	.NET, EJB, COM, Oracle, MSSQL, DB2, MySQL, PostgreSQL	X	Model Business Logic, Transform
Enterprise Architect	Ultimate, Systems Engineering, Business & Software Engineering, Corporate, Professional	J2EE, .NET, CORBA, PHP, Oracle, MSSQL, Access, PostgreSQL, Informix, InterBase, FireBird	X	Domain Model, Application Model, Code Model
Objecteering	Objecteering Enterprise Edition, Objecteering UML Free Edition, Objecteering SOA Edition	J2EE, .NET, C++, Oracle, Sysbase, MSSQL	X	Model Business Logic, Domain Model, Application Model, Code Model

References

1. A. Kleppe, J. Warmer, W. Bast, 2003. *MDA Explained: The Model Driven Architecture – Practice and Promise,* Addison-Wesley Professional.
2. S. Mellor, K. Scott, A. Uhl, D. Weise, 2004. *MDA Distilled: Principles of Model-Driven Architecture,* Addison-Wesley Professional.
3. J. Siegel, OMG Staff Strategy Group, 2004. *Developing in OMG's Model-Driven Architecture,* OMG White Paper.
4. http://www.omg.org/mda
5. OMG, 2003. *MDA Guide Version 1.0.1.*
6. http://www.jot.fm
7. OMG, 2005, *Revised submission for MOF 2.0 Query/View/Transformation RFP*.
8. S. Mellor, M. Balcer, 2002. *Executable UML: A Foundation for Model-Driven Architecture,* Addison-Wesley Professional.
9. D. Varro, A. Pataricza, 2003. *UML Action Semantics for Model Transformation Systems,* Citeseer.
10. OMG, 2003. *Common Warehouse Metamodel Specification*.
11. N. Tariq, N. Akhter, *Comparison of Model Driven Architecture based tools*.
12. http://www.compuware.com
13. http://www.interactive-objects.com
14. http://www.eclipse.org/modeling/emf
15. http://www.andromda.org
16. http://www.integranova.com
17. http://www.sparxsystems.com
18. http://www.objecteering.com