

# C3NET: Smart Environment For .NET Code Generation Using MDA

B.C. Pelayo G-Bustelo

J.M. Cueva Lovelle  
Computer Science Department  
University of Oviedo  
C/ Calvo Sotelo s/n  
33007-Oviedo (Spain)

A.A. Juan Fuente

*Abstract - The **MDA** (Model Driven Architecture) is a new way of writing specifications and developing applications, based on a platform-independent model. A complete MDA specification consists of a definitive platform-independent base UML model, plus one or more platform-specific models and interface definition sets, each describing how the base model is implemented on a different middleware platform. A MDA works above the level of middleware platform, for example .NET. The denominated **Framework C3NET** (Code of 3er level in .NET), that it makes the stage of code generation at low level for the platform .NET. It is designed to serve as tool back-end for MDA. C3NET is designed of flexible form allowing to future extensions of the platform NET and their different implementations.*

**Keywords:** MDA, XMI, MOF, C3NET, UML, Model

## 1 Introduction

Model Driven Architecture (MDA) allows the construction of computer systems from models that can more easily be understood than the code of a programming language. This architecture was formulated by Object Management Group [5]. The Model-Driven Architecture starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform.

MDA provides an approach for, and enables tools to be provided for: a) specifying a system independently of the platform that supports it, b) specifying platforms, c) choosing a particular platform for the system, and d) transforming the system specification into one for a particular platform.

The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns.

## 2 The Basic Concepts

Let's look at the concepts that are at the core of MDA.

### 2.1 System

MDA concepts in terms of some existing or planned system. That system may include anything: a program, a single computer system, some combination of parts of different systems, a federation of systems, each under separate control, people, an enterprise, a federation of enterprises. . .

## 2.2 Model

A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language.

## 2.3 Model-Driven

MDA is an approach to system development, which increases the power of models in that work. It is model-driven because it provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification.

## 2.4 Architecture

The architecture of a system is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors. The MDAS prescribes certain kinds of models to be used, how those models may be prepared and the relationships of the different kinds of models.

## 2.5 Viewpoint and View

A viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system. Here “abstraction” is used to mean the process of suppressing selected detail to establish a simplified model.

The concepts and rules may be considered to form a viewpoint language. The MDA specifies three viewpoints on a system, a computation independent viewpoint, a platform independent viewpoint and a platform specific viewpoint.

A viewpoint model or view of a system is a representation of that system from the perspective of a chosen viewpoint.

## 2.6 Platform

A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented. Examples:

- Generic platform types: Object, Batch and Dataflow.
- Technology specific platform types: CORBA and J2EE.
- Vendor specific platform types: CORBA, J2EE and Microsoft .NET

## 2.7 Application

The term application is used to refer to a functionality being developed. A system is described in terms of one or more applications supported by one or more platforms.

## 2.8 Platform Independence

Platform independence is a quality, which a model may exhibit. This is the quality that the model is independent of the features of a platform of any particular type.

Like most qualities, platform independence is a matter of degree. So, one model might only assume availability of features of a very general type of platform, such as remote invocation, while another model might assume the availability a particular set of tools for the CORBA platform. Likewise, one model might assume the availability of one feature of a particular type of platform, while another model might be fully committed to that type of platform.

## 2.9 MDA Viewpoints

The computation independent viewpoint focuses on the on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined.

The platform independent viewpoint focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another. A platform independent view may use a general purpose modeling language, or a language specific to the area in which the system will be used.

The platform specific viewpoint combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

## 2.10 The Computation Independent Model (CIM)

A computation independent model is a view of a system from the computation independent viewpoint. A CIM does not show details of the structure of systems.

It is assumed that the primary user of the CIM, the domain practitioner, is not knowledgeable about the models or artifacts used to realize the functionality for which the requirements are articulated in the CIM. The CIM plays an important role in bridging the gap between those that are experts about the domain and its requirements on the one hand, and those that are experts of the design and construction of the artifacts that together satisfy the domain requirements, on the other.

## 2.11 Platform Independent Model (PIM)

A platform independent model is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type.

A very common technique for achieving platform independence is to target a system model for a technology-neutral virtual machine. A virtual machine is defined as a set of parts and services which are defined independently of any specific platform and which are realized in platform-specific ways on different platforms. A virtual machine is a platform, and such a model is specific to that platform. But that model is platform independent with respect to the class of different platforms on which that virtual machine has been implemented. This is because such models are unaffected by the underlying platform and, hence, fully conform to the criterion of platform independence.

## 2.12 Platform Specific Model (PSM)

A platform specific model is a view of a system from the platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

## 2.13 Platform Model

A platform model provides a set of technical concepts, representing the different kinds of parts that make up a platform and the services provided by that platform. It also provides, for use in a platform specific model, concepts representing the different kinds of elements to be used in specifying the use of the platform by an application.

A platform model also specifies requirements on the connection and use of the parts of the platform, and the connections of an application to the platform. A generic platform model can amount to a specification of a particular architectural style.

## 2.14 Model Transformation

Model transformation is the process of converting one model to another model of the same system.

The platform independent model and other information are combined by the transformation to produce a platform specific model. There are many ways in which such a transformation may be done. However it is done, it produces, from a platform independent model, a model specific to a particular platform.

## 2.15 Pervasive Services

Pervasive services are services available in a wide range of platforms. MDA will provide common, platform independent models of pervasive services. It will provide mappings for transforming models, which draw on these pervasive service PIMs, to platform specific models using the services as provided by a particular platform.

## 2.16 Implementation

An implementation is a specification, which provides all the information needed to construct a system and put it into operation.

# 3 MDA's Implementations

This section gives an overview of some MDA-oriented tools, some of which are pure code generation tools, others more full fledged model-driven tools. They may all be part of someone's MDA process. Of course, UML tools may also be considered MDA tools, and will often be central in model-driven development.

- AndroMDA, an open source template-based tool for J2EE code generation from UML/XMI. Uses VTL (Velocity Template Engine) as scripting language and Netbeans MDR as a model API.
- ArcStyler, is a commercial MDA tool from Interactive Objects. It is bundled with MagicDraw UML-tool, but can also support other UML-tools through tool adaptors.
- The ATL Engine is a QVT-based transformation language, developed by the INRIA Atlas team. The ATL Engine is currently available as open source under Eclipse GMT. It is developed as a set of Eclipse plugins and works as a development IDE for transformations, with execution and debugging. Currently integrates with EMF and MDR. The ATL Engine will be developed further within the ModelWare IP.
- MOFScript, a model to text transformation tool, based on one of the OMG MOF Model to Text Transformation submissions - Eclipse plugin, based on metamodels/models in EMF.
- The IBM Model Transformation Framework (MTF) is an EMF based model transformation framework, for now available at alphaWorks. It provides a declarative means of specifying metamodel relationships, similar to that of QVT relations.
- MTL Engine - Another QVT-like implementation, by the INRIA Triskell team. Uses the MTL language. Integrates with Netbeans MDR and Eclipse EMF.
- ModFact A MOF Repository and QVT-like engine from LIP6, Paris. Based on the TRL language. LIP6 are also working on an open source ModelBus implementation, which will enable MDD tools interoperability.
- Generative Model Transformer (GMT), an eclipse project that is providing/will provide model transformation technology for the eclipse platform. Currently the FUUT-je tool, a code generator tool, is the primary GMT deliverable. (ATL, mentioned above, provides core transformation technology...)
- Kent Modelling Framework (KMF), a tool for generation of languages with support for dynamic constraint checking.
- OpenArchitectureWare, a flexible, template-based generator framework integrated with XMI.
- OpenMDX, an open source MDA environment, which integrates with several tools through XMI and supports code generation towards several target platforms (J2EE, .Net).
- UMT - (UML Model Transformation Tool) - UMT is an open source UML/XMI-based tool for model transformation and code generation purposes, which uses XSLT and Java for generation.
- XDoclet, an open source, attribute based code generation tool for J2EE. Not really model-based, but can be combined with generation tools such as UMT to achieve good model-based value.

- Middlegen, an open source, database driven code generator based on JSBC, Velocity, Xdoclet and Ant.
- MCC (Model Component Compiler), a commercial product from InferData, supporting generation towards J2EE
- OptimalJ, a commercial product from Compuware, uses a notation of patterns to achieve PSM transformations. Has an integrated UML tool for analysis, but uses a slightly different notation (structural) for the MDA-part of the tool.
- Xactium XMF Mosiac, a commercial model-based mapping, generation and execution tool suite
- SosyInc Modeler and Transformation Engine - The transformation engine provides generation of GUI and server-side, based on models OASIS/UML and rules for application structure and business rules.
- Model-in-Action, and MDA tool suite from Mia software supporting code generation and model to model transformation in a flexible framework.

## 4 C3NET

### 4.1 General Scheme of Framework C3NET

The C3NET Scheme is show in the figure 1. Next sections gives an overview of some aspects.

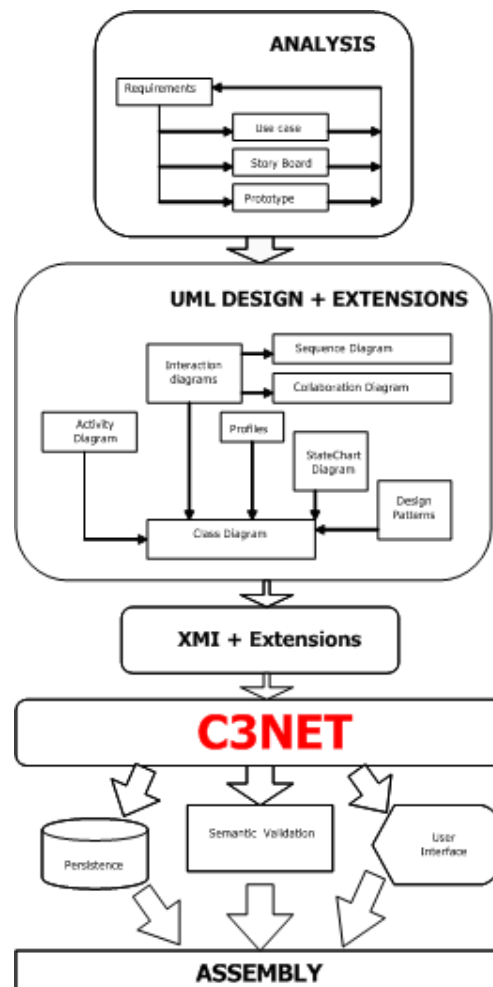


Figure 1: General Scheme of Framework C3NET

### 4.1.1 Analysis

It is gotten up as newness the integration of the analysis in the MDA. A systematic step of the requirements of analysis to design. The prototypes are the base for the user interfaces. A use case diagram shows the relationship among use cases within a system or other semantic entity and their actors. Use case diagrams show actors and use cases together with their relationships. The use cases represent functionality of a system or a classifier, like a subsystem or a class, as manifested to external interactors with the system or the classifier.

### 4.1.2 Design

From the Scenes the static and dynamic specifications are obtained. The diagrams of states must be extended with a language based on OCL and other specifications in a neutral language OO that facilitates the generation in diverse languages: Java, C#.

Interaction diagrams come in two forms based on the same underlying information, specified by a Collaboration and possibly by an Interaction, but each form emphasizes a particular aspect of it. The two forms are sequence diagrams and collaboration diagrams. A sequence diagram shows the explicit sequence of communications and is better for real-time specifications and for complex scenarios. A collaboration diagram shows an Interaction organized around the roles in the Interaction and their relationships. It does not show time as a separate dimension, so the sequence of communications and the concurrent threads must be determined using sequence numbers.

A Collaboration includes an ensemble of ClassifierRoles and AssociationRoles that define the participants needed for a given set of purposes. Instances conforming to the ClassifierRoles play the roles defined by the ClassifierRoles, while Links between the Instances conform to AssociationRoles of the Collaboration. ClassifierRoles and AssociationRoles define a usage of Instances and Links, and the Classifiers and Associations declare all required properties of these Instances and Links.

An Interaction is defined in the context of a Collaboration. It specifies the communication patterns between the roles in the Collaboration. More precisely, it contains a set of partially ordered Messages, each specifying one communication; for example, what Signal to be sent or what Operation to be invoked, as well as the roles to be played by the sender and the receiver, respectively.

A sequence diagram presents an Interaction, which is a set of Messages between ClassifierRoles within a Collaboration, or an InteractionInstanceSet, which is a set of Stimuli between Instances within a CollaborationInstanceSet to effect a desired operation or result.

Statechart diagrams represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances. Typically, it is used for describing the behavior of class instances, but statecharts may also describe the behavior of other entities such as use-cases, actors, subsystems, operations, or methods.

A class diagram is a graph of Classifier elements connected by their various static relationships. A “class” diagram may also contain interfaces, packages, relationships, and even instances, such as objects and links. Perhaps a better name would be “static structural diagram” but “class diagram” is shorter and well established. A class diagram is a graphic view of the static structural model. The individual class diagrams do not represent divisions in the underlying model.

### 4.1.3 XMI+Extensions (Actions)

XMI with necessary extensions (actions) for the semantics which UML lacks and in addition it is needed to specify the behavior of the methods (algorítmica of the methods). The information of the profiles and the implementation of Patrons of Design. The main purpose of XMI is to enable easy interchange of metadata between modelling tools (based on the OMG-UML) and metadata repositories (OMG-MOF based) in distributed heterogeneous environments. XMI integrates three key industry standards:

- XML - eXtensible Markup Language, a W3C standard
- UML - Unified Modeling Language, an OMG modeling standard

- MOF - Meta Object Facility, an OMG metamodeling and metadata repository standard

The integration of these three standards into XMI marries the best of OMG and W3C metadata and modeling technologies, allowing developers of distributed systems to share object models and other metadata over the Internet. XMI, together with MOF and UML form the core of the OMG metadata repository Architecture.

#### 4.1.4 C3NET

C3NET gather the XMI + extensions (actions) and the integration and generate code making the semantic validation and generate the persistence with the help of persistence motors, the interfaces of users with the help of generating tools of interfaces and generate the code in IL with tools.

#### 4.1.5 Assembly

Assembly .NET Framework is a partially compiled code library for use in deployment, versioning and security. In the Microsoft Windows implementation of .NET, an assembly is a PE (portable executable) file. There are two types, process assemblies (EXE) and library assemblies (DLL). An assembly can consist of one or more files. Code files are called modules. An assembly can contain more than one code module and since it is possible to use different languages to create code modules this means that it is technically possible to use several different languages to create an assembly.

## 5 Conclusions

MDA is a powerful tool for repetitive software or management software with repetitive task. The value proposition for MDA is than the cost of building and maintaining systems is significantly lower. The combination of the following makes a powerful business argument:

- Raising the level of abstraction so that people can express themselves and communicate with man and machine alike more productively.
- Raising the level of reuse to a higher level of granularity so that bigger systems can be composed from ever-larger elements.
- Relying on the power of design-time interoperability, so that models can stand alone, ready for combination with other models in an additive manner.

C3NET is the suggestion to .NET framework that covers a set of tasks from code generation to assemblys definitions.

## 6 References

- [1] Arlow, J and Neustadt, I. Enterprise Patterns and MDA. Building better software archetype patterns and UML. Addison Wesley, 2004.
- [2] Frankel, D. Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley Press, 2003.
- [3] Harman, P. MDA: An Idea Whose Time Has Come. Cutter Consortium, 2003.
- [4] Kleppe, A , Warmer, J. and Bast, W. MDA Explained: The Model Driven Architecture Practice and Promise. Addison Wesley, 2003.
- [5] Object Management Group (OMG) <http://www.omg.org>
- [6] OMG Staff Strategy Group, "MDA Guide Version 1.0.1." June 2003.
- [7] Pelayo García-Bustelo, B. C. in spanish "C3NET: Herramienta para el desarrollo de lenguajes en la Plataforma .NET". Proyecto de Fin de Carrera EPSIG, Universidad de Oviedo June 2004
- [8] Selic, B. "The Pragmatics of Model-Driven Development" IEEE Software, Vol. 20, #5, September 2003
- [9] Soley, R. and OMG Staff Strategy Group, "Model Driven Architecture." November 2000.