Model Driven Architecture (Mda)

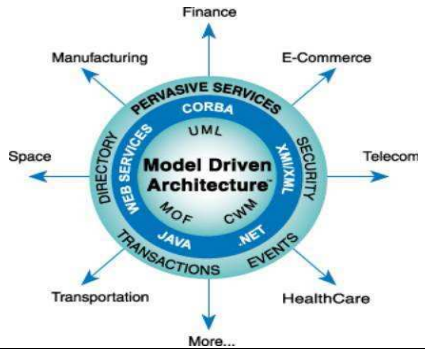
Sachin Patel, Nilesh Parmar *
**

Abstract

The Model Driven Architecture (MDA) separates the system business aspects from the system implementation aspects on a specific technology platform. MDA proposes a software development process in which the key notions are models and model transformation, where the input models are platform independent and the output models are platform specific and can be transformed into a format that is executable. The principles of MDA for the task of study program development are applied using two hemispheres model driven (2HMD) approach, which assumes modeling and use of procedural and conceptual knowledge and may be applied in the context of modeling the knowledge about the domain. From the point of view of 2HMD approach each course of the study program may be considered as a knowledge provider, which satisfies particular knowledge requirements. The particular "Specific" knowledge requirements are derived from the knowledge model that consists of functional and conceptual "hemispheres", which reflect knowledge derived from original academic or industrial requirements or standards.

Keywords: Service Oriented Architecture (SOA), Object Model Group (OMG), XML Metadata Interchange, XML Metadata Interchange, Meta Object Facility (MOF), and Common Warehouse Metamodel (CWM).

1. Introduction: The Model-Driven Architecture (MDA) defines an approach to modeling that separates the specification of system functionality from the specification of its implementation on a specific technology platform. In short it defines guidelines for structuring specifications expressed as models. The MDA promotes an approach where the same model specifying system functionality can be realized on multiple platforms through auxiliary mapping standards, or through point mappings to specific platforms. It also supports the concept of explicitly relating the models of different applications, enabling integration and interoperability and supporting system evolution as platform technologies come and go.



*Asst Prof PCST
**M.Tech (SCS DAVV) Student

The MDA is part of a collection of modeling-oriented standards from the OMG. These standards include:

1. ***XML Metadata Interchange (XMI)***. XMI is a standard interchange mechanism used between various tools, repositories and middleware. XMI can also be used to automatically produce XML DTDs and XML schemas from UML and Meta Object Facility (MOF) models, providing an XML serialization mechanism for these artifacts.
2. ***XML Metadata Interchange***. The UML defines the notation and semantics for modeling diagrams, as you learned earlier.
3. ***Common Warehouse Metamodel (CWM)***. CWM is the OMG data warehouse standard, and is a good example of applying the MDA paradigm to an application area.
4. ***Meta Object Facility (MOF)***. The MOF provides the standard modeling and interchange constructs that are used in the MDA. The common foundation of the MOF, the UML and CWM are MOF-based, enables the potential model/metadata interchange and interoperability between tools. The MOF is the mechanism through which models are analyzed in XMI.

Two- hemisphere model driven*:* Two-hemisphere model driven (2HMD) approach has been successfully applied for modeling and software design domain. One of the most distinguished features of this model is its applicability for both human understanding and automatic transformations. In this paper we illustrate the way how2HMD approach can be applied to the task of knowledge modeling for educational purposes. This task becomes more and more important because of urgent need of knowledge provision in the knowledge based economy. The need of providing knowledge timely and accurately arises in different settings, starting from industrial needs to training employees in new technologies and ending with universities that have to become more and more flexible in terms of their curricula and course development.

2. Architecture Diagram of Model-Driven Architecture:

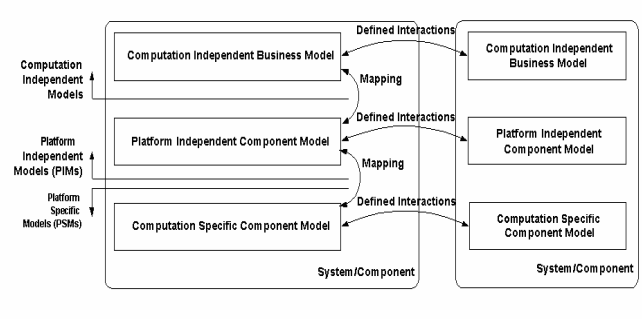


Figure presents an overview of the MDA.

The MDA is based on the idea that a system or component can be modeled via two categories of models: Platform Independent Models and Platform Specific Models.

Platform Independent Models are further divided into Computation Independent Business Models and Platform Independent Component Models. As the name implies Platform Independent model do not take technology-specific considerations into account, a concept that is equivalent to logical models in the structured methodologies and to essential models within usage-centered techniques. The Computation Independent business Models IBM represent the business requirements and processes that the system or component supports and the Platform Independent computation Model are used to model the logical business components, also called domain components. Platform Specific Models bring technology issues into account and are effectively transformations of PIMs to reflect platform-specific considerations.

The MDA has four primary strengths:

1. The MDA defines a coarse separation of views. When you are modeling it is important to consider issues such as "what should the system do" and "what processes should the system support" without having to worry about how it will support it because it enables you to consider a wide range of options. It can also be beneficial to consider the logical domain architecture separately from the physical implementation architecture because the logical architecture will change much slower than the underlying technologies, allowing you to reuse your logical model as the system evolves over time.
2. The MDA defines a viable strategy for system integration. The MDA explicit support for modeling relationships between systems/components can help to promote system integration issues, and hence promote greater levels of reuse via system integration, within your organization. My experience is that integration, in particular legacy integration, is a significant issue for most software development efforts. Yet few books cover this critical topic, including most of my own, written under the apparent assumption that all systems are developed in "green-field" environments where everything is being built for the first time.
3. The MDA may motivate a new breed of modeling tools. The separation of views offers the potential for tools that automatically generate the "next model down", in particular PSMs from PIMs, based on automated mappings.
4. The MDA may support tool integration. Part of the overall vision of the OMG is to provide a set of standards for tool vendors to follow and thereby support integration of those tools.

Although the MDA is interesting, and in the near term I suspect we will hear a lot about it from the OMG and its supporters, there are several potential challenges that you need to be aware of. However, I also think that it's possible to take an agile approach to the MDA. In the end, gut feel tells me that less than 5% of all organizations can truly take advantage of the MDA. Model-driven architecture (MDA) is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems. It was launched by the Object Management Group (OMG) in 2001.

3. MDA tools: The Object Management Group organization provides rough specifications rather than implementations, often as answers to Requests for Proposals. The OMG documents the overall process in a document called the MDA Guide. Basically, an MDA tool is a tool used to develop, interpret, compare, align, measure, verify, transform, etc. models or Meta models. In the following section "model" is interpreted as meaning any kind of model or Meta model. In any MDA approach we have essentially two kinds of models: Initial Model are created manually by human agents while derived model are created automatically by programs. For example an analyst may create a UML initial model from its observation of some loose business situation while a Java model may be automatically derived from this UML model by a Model transformation operation.

An MDA tool may be one or more of the following types:

- Creation Tool: A tool used to elicit initial models and/or edit derived models.
- Analysis Tool: A tool used to check models for completeness, inconsistencies, or error and warning conditions. Also used to calculate metrics for the model.
- Transformation Tool: A tool used to transform models into other models or into code and documentation.
- Composition Tool: A tool used to compose (i.e. to merge according to a given composition semantics) several source models, preferably conforming to the same metamodel.
- Test Tool: A tool used to "test" models as described in Model-based testing.
- Simulation Tool: A tool used to simulate the execution of a system represented by a given model. This is related to the subject of model execution.
- Metadata Management Tool: A tool intended to handle the general relations between different models, including the metadata on each model and the mutual relations between these models.
- Reverse Engineering Tool: A tool intended to transform particular legacy or information artifact portfolios into full-fledged models.

Some tools perform more than one of the functions listed above. For example, some creation tools may also have transformation and test capabilities. There are other tools that are solely for creation, solely for graphical presentation, solely for transformation, etc.

One of the main aims of the MDA is to separate design from architecture. As the concepts and technologies used to realize designs and the concepts and technologies used to realize architectures have changed at their own pace, decoupling them allows system developers to choose from the best and most fitting in both domains. The design addresses the functional (use case) requirements while architecture provides the infrastructure through which non-functional requirements like scalability, reliability and performance are realized. MDA envisages that the platform independent model (PIM), which represents a conceptual design realizing the functional requirements, will survive changes in realization technologies and software architectures.

One of the characteristics of MDA tools is that they mainly take models as input and generate models as output. In some cases however the parameters may be taken outside the MDA space like in model to text or text to model transformation tools.

4. MDA Approaches: In this first part we examine the importance of models and modeling, and introduce the four key principles of MDA. We then look at IBM's support for MDA and the leadership role that IBM has played in defining the MDA approach and its supporting standards. Developing enterprise-scale applications today requires an approach to software architecture that helps architects evolve their solutions in flexible ways. This approach should permit reuse of existing efforts in the context of new capabilities that implement business functionality in a timely fashion, even as the target infrastructure itself is evolving. Two important ideas are now considered central to addressing this challenge:

- Service-Oriented Architectures (SOA). Enterprise solutions can be viewed as federations of services connected via well-specified contracts that define their service interfaces. The resulting system designs are frequently called Service Oriented Architectures (SOA). Flexibility can be enhanced in a system's architecture by organizing a system as a collection of encapsulated services making calls on operations defined through their service interfaces. Many organizations now express their solutions in terms of services and their interconnections.
- OMG. Object Management Group (OMG) is a consortium, originally aimed at setting standards for distributed object-oriented systems. The OMG is a non-profit consortium created in 1989 to promote the theory and practice of object technology for the development for distributed.

Four principles underlie the OMG' view of MDA:

- Models expressed in a well-defined notation are a cornerstone to understanding systems for enterprise-scale solutions.
- The building of systems can be organized around a set of models by imposing a series of transformations between models, organized into an architectural framework of layers and transformations.
- A formal underpinning for describing models in a set of metamodels facilitates meaningful integration and transformation among models, and is the basis for automation through tools.
- Acceptance and broad adoption of this model-based approach requires industry standards to provide openness to consumers, and foster competition among vendors.
- Software Product Lines. Frequently, there is a great deal of commonality among the systems an organization develops and maintains. We see recurring approaches at every level of an enterprise software project, from having standard domain models that capture core business processes and domain concepts, to the way in which developers implement specific solutions to realize designs in code. Organizations gain a great deal of efficiency when patterns can be defined by skilled practitioners and propagated across the IT organization. This represents a move toward a software product-line view of development that promotes planned

reuse of assets, along with an increasing level of automation, to realize solutions for large parts of the systems being developed. More generally, we can understand the application of well-defined patterns in a product-line view of development as a way to transform descriptions of a solution from one level of abstraction to a lower level of abstraction.

These two ideas have had significant influence on the thinking of the Object Management Group (OMG), a consortium of software organizations that develops and supports specifications to improve the practice of enterprise software development and deployment. (There will be more on the important role the OMG plays in the next section). The OMG has created a conceptual framework separates business-oriented decisions from platform decisions to allow greater flexibility when architecting and evolving these systems. This conceptual framework and the standards that help realize it is what the OMG calls "Model Driven Architecture (MDA). Application architects use the MDA framework as a blueprint for expressing their enterprise architectures, and employ the open standards inherent in MDA as their "future proofs" against vendor lock-in and technology churn.

5. MDA usages: Here are the basic assumptions and parameters governing MDA usage today:

- Models help people understand and communicate complex ideas.
- Many different kinds of elements can be modeled, depending on the context. These offer different views of the world that must ultimately be reconciled.
- We see commonality at all levels of these models in both the problems being analyzed and the proposed solutions.
- Applying the ideas of different kinds of models and transforming them between representations provides a well-defined style of development, enabling the identification and reuse of common approaches.
- In what it calls "model driven architecture," the OMG has provided a conceptual framework and a set of standards to express models, model relationships, and model-to-model transformations.
- Tools and technologies can help to realize this approach, and make it practical and efficient to apply.

6. Conclusion: As we have seen from this paper Object Management Group in MDA is widely accepted and there are a lot of tools supporting this. It offers a lot of advantages over the current way of software development, but there are some down-sides particularly with the standardizations. OMG is working on formal standards for parts of the development process but this is something that is necessary for the acceptance of model driven development. MDA has a chance to succeed in large, distributed, industrial software development, but it is still not so useful in small business and small development project. However MDA is a work in progress and still has a lot of issues that need to be resolved before we can say for sure

7. References:

1. Object Management Group. MDA Guide Version 1.0.1., 2003
2. Model Driven Architecture (MDA) FAQ, Object Management Group, http://www.omg.org/mda/faq_mda.htm#whatismda
3. Stephen J. Mellor, Kendall Scott, Axel Uhl, Dirk Weise (2004) : *MDA Distilled*, Addison Wesley Professional, March.
4. Hailpern and Tarr, (2006) *Model-driven Development: The Good, the bad and the ugly*, IBM Systems Journal, vol.45, no 3.
5. Object Management Group, Model Driven Development homepage: http://www.omg.org/mda
6. Alan Brown, Staff IBM: An introduction to Model Driven Architecture Part I, http://www.ibm.com/developerworks/rational/library/3100.html
7. Alan Brown, Staff IBM: An introduction to Model Driven Architecture Part II, http://www.ibm.com/developerworks/rational/library/apr05/brown/index.html
8. John D. Poole, *Model-driven Architecture: Vision, Standards and Emerging*