Abstract. Model Driven Architecture (MDA) is an approach to application modeling and generation that has received a lot of attention in recent months. Championed by the Object Management Group (OMG), many organizations are now looking at the ideas of MDA as a way to organize and manage their application solutions, tool vendors are explicitly referring to their capabilities in terms of “MDA compliance”, and the MDA lexicon of platform-specific and platform-independent models is now widely referenced in the industry.

In spite of this interest and market support, there is little clear guidance on what MDA means, where we are in its evolution, what is possible with today’s technology, and how to take advantage of it in practice. This paper addresses that need by providing an analysis of how modeling is used in industry today, the relevance of MDA to today’s systems, a classification of MDA tooling support, and examples of its use. The paper concludes with a set of recommendations for how MDA can be successful in practice.

Keywords: Software architecture – Software design – Unified Modeling Language (UML)

1 Introduction

Building enterprise-scale software solutions has never been easy. The difficulties of understanding highly complex business domains are typically compounded with all the challenges of managing a development effort involving large teams of engineers over multiple phases of a project spanning many months. The time-to-market pressures inherent to many of today’s product development efforts only serve to compound the problems.

In addition to the scale and complexity of many of these efforts, there is also great complexity to the software platforms for which enterprise-scale software are targeted. Most IT organizations rely on a complicated assortment of infrastructure technologies that have evolved over multiple years, consist of a variety of middleware software acquired from many vendors, and assembled through various poorly documented integration efforts of varied quality.

To develop applications in this context requires an approach to software architecture that helps architects evolve their solutions in flexible ways, reusing 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 how to address this: Service-Oriented Architectures and Software Product Lines.

1.1 Service oriented architectures

An approach gaining a lot of support in the industry today is based on viewing enterprise solutions as federations of services connected via well-specified contracts that define their service interfaces. The resulting system designs are frequently called Service Oriented Architectures (SOAs) [1]. Systems are composed of collections of services making calls on operations defined through their service interfaces. Many organizations now express their solutions in terms of services and their interconnections.

A number of important technologies have been defined to support an SOA approach, most notably when the services are distributed across multiple machines and connected by the Internet. These web service approaches rely on intra-service communication protocols such as the Simple Object Access Protocol (SOAP), allow the web
service interfaces (expressed in the Web Services Definition Language – WSDL) to be registered in public directories and searched in Universal Description, Discovery and Integration (UDDI) repositories, and share information in documents defined in the eXtensible Markup Language (XML) and described in standard schemas.

1.2 Software product lines

Organizations are also beginning to recognize that there frequently is a great deal of commonality in the systems they develop. Recurring approaches are seen 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. A great deal of efficiency can be gained if patterns can be defined by more skilled developers and propagated across the IT organization.

Recognizing this, many organizations are moving toward a software product line view of their development in which planned reuse of assets is supported, and an increasing level of automation can be used to realize solutions for large parts of the systems being developed [2, 3]. More generally, the underpinning of any product line approach can be viewed as a way to transform descriptions of a solution at one level of abstraction into descriptions at a lower level of abstraction by applying well-defined patterns.

1.3 The new development imperative

Recognizing the need for greater flexibility in the development of enterprise-scale solutions, the OMG has created a conceptual framework\(^2\) that 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, 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 that are inherent to MDA to define standard transformations between models as their “future proofing” against vendor lock-in and technology churn.

The OMG’s MDA approach provides an open, vendor-neutral approach to system interoperability via OMG’s established modeling standards: Unified Modeling Language (UML), Meta-Object Facility (MOF), and Common Warehouse Meta-model (CWM). Platform-independent descriptions of enterprise solutions can be built using these modeling standards and can be transformed into a major open or proprietary platform, including CORBA, J2EE, .NET, XMI/XML, and Web-based platforms.

While this all sounds fine in theory, what is the reality in practice? This paper looks at the principles and practice of MDA, and provides a perspective on the current state of the practice in MDA as realized in IBM’s modeling, design, and implementation technologies.

1.4 Related work

The ideas of MDA are becoming widely discussed in the software industry, and there are a number of sources of help for those interested in applying MDA in practice. The three primary sources are:

- **OMG Materials.** The OMG provides the primary source for many MDA ideas, consisting of a number of specifications, whitepapers, and presentation materials that discuss the MDA approach (see www.omg.org). These tend to be at two levels of detail – either they are detailed specifications aimed at technologists implementing those specifications, or they are high level overviews of concepts and standards aimed at positioning the MDA approach. A small, but growing, set of materials are aimed at filling the space between these two extremes for those wanting to understand more about MDA in the context of current development approaches, and how MDA can be applied in practice.

- **Books and papers.** Recognizing the gaps in OMG materials, a number of books and papers are currently appearing in print. The three primary texts available are from Kleppe et al. [4], Frankel [5], and Mellor et al. [6]. These offer perspectives on the key OMG standards and express general comments on their value, supported by brief practical insights into MDA in practice.

- **Broader industry and academic materials.** As the MDA approach gains support, a number of materials are becoming available that addresses practical application of MDA, its strengths, and its limitations. Currently, this material is very variable in focus, depth, and quality. Typically, these discussions focus around specific technology solutions. An Internet search on this topic will reveal many of these materials.

In this paper we complement many of these existing materials with an overview of the principles and practice of MDA appropriate to the needs of today’s software practitioners requiring a balanced overview of MDA and its role in development.

2 Why model?

Models provide abstractions of a physical system that allow engineers to reason about that system by ignoring extraneous details while focusing on the relevant ones. All forms of engineering rely on models as essential to un-

\(^2\) In this context a conceptual framework is a set of key concepts and structures that guides the planning, understanding, and realization of an enterprise solution.