WHITE PAPER

Understanding tool requirements for Model Driven Architecture

Koos de Goede* and John Irizarry
February, 2008

** Koos de Goede is founder of @-portunity, an emerging enterprise with core competencies in Model Driven Architecture development. He can be reached at koos@atportunity.com.*

Introduction

Historically software design and development efforts face numerous challenges, beginning from the early stages of requirements gathering and continuing on through the entire Software Development Life Cycle. Software developers often subscribe to legacy practices, taking a code-only approach without employing the use of separately defined models. The written code forms their basis for expressing system models, usually enacted by third-party programming languages within development environments such as Eclipse. This approach is informal, intuitive and mostly living on whiteboards and word documents, or even in the developer's head; dissipating instead of permeating throughout the project lifecycle. The result is that performance can only be predicted by individual rather than organizational capability.

Model Driven Architecture (MDA™) solves this problem by facilitating model-driven development of Enterprise-Class applications that are based on open standards and are able to adapt to new hardware capabilities and software platforms. MDA, however, can only exist by means of high-quality tools that support the MDA standards.

This document briefly touches upon the MDA paradigm and intents to present a checklist of what MDA tool suites should be capable of. The checklist is illustrated with feature highlights from Blueprint ME, @-portunity's integrated MDA development environment.

MDA Foundations

Being the product of the Object Management Group's (OMG™) open standard adoption process, MDA enables enterprises to integrate what has been built with what will be built

based on well-established and open standards. In MDA, models, expressed in a well-defined notation, are the cornerstone in defining and understanding systems. Models are created at different levels of abstraction, separating the concerns of the business from the technical details of the actual software solution to be developed. Basically, three different types of models are constructed: one that contains the business specifications, one for the high level details of the platform and one that includes technical details of the target platform. By describing these models through a set of meta-models, transformation amongst models is facilitated, ultimately resulting in code generation.

"MDA enables enterprises to integrate what has been built with what will be built based on well-established and open standards."

MDA Benefits

The bottom-line benefits of MDA are significant- to business leaders and developers alike: reduced cost throughout the application life-cycle reduced development time for new applications and improved quality of even more complex systems. Moreover MDA allows for rapid inclusion of emerging technology benefits into existing systems. The realization that modeling is critical to the success of every enterprise-scale solution forms the impetus for a growing consensus that MDA promotes:

- Platform independency
- Domain specificity
- Productivity
- Uniformity
- Consistency

Platform independency

MDA greatly reduces the time, cost and complexity associated with re-targeting applications for different platforms, including those yet to be introduced.

Domain specificity

MDA enables implementations of industry specific applications over diverse platforms, through domain specific meta-models.

Productivity

MDA increases productivity, by permitting developers, designers and system administrators to use languages and concepts they are comfortable with, while enabling seamless communication and integration across the teams.

Uniformity

As from the onset, models and their definitions are structured to achieve consistency yielding applications that are implemented in a uniform manner; MDA increases the quality and robustness of the end product.

Consistency

Through the ability to perform bi-directional transformations, MDA assures that all models reflect current-state consistently.

MDA Standards

The definition of the MDA concept started late 2000 using OMG's open standard adoption process, only to reach maturity with the definition of the QVT specifications in 2007. The definite MDA standards require models to be compliant with OMG's Meta-Object Facility (MOF™). This guarantees that all models can be related to each other in order to be parsed, transferred, stored and transformed. More concrete, the following standards must be integrated in an MDA development environment in order to support the start-to-end model-driven development cycle:

- Meta-Object Facility (MOF)
- Unified Modeling Language (UML)
- Object Constraint Language (OCL)
- XML Metadata Interchange (XMI)
- Query/View/Transformations (QVT)
- Model-to-Text (M2T)

Meta-Object Facility (MOF)

MOF™ is an extensible model driven integration framework for defining, manipulating and integrating meta-data and data in a platform independent manner.

Unified Modeling Language (UML)

UML® is a standardized specification language for object modeling. It is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system.

Object Constraint Language (OCL)

OCL is a precise text language that provides constraint and object query expressions on any MOF-based model or meta-model that cannot otherwise be expressed by diagrammatic notation.

XML Metadata Interchange (XMI)

XMI is a model driven XML Integration framework for defining, interchanging, manipulating and integrating XML data and objects. XMI-based standards are in use for integrating tools, repositories, applications and data warehouses.

Query/View/Transformation (QVT)

QVT is a standard for writing transformation specifications between MOF based meta-models. A QVT engine is able to execute transformations and create (or update) a target model from a source model.

Model-to-Text (M2T)

M2T enables transformations of models to various text artifacts such as code, deployment specifications, reports and application documentation.

Tool Ecosystem: Polluted

Even before the crystallization of OMG's MDA standards, the tooling ecosystem attained a new category: MDA tools. Many of these tools claim to enable MDA development, but reality shows that only the code-generation part of MDA is supported. As a result, model-driven development has become a buzzword, the main objective believed to be the automation of 100% code generation. On the contrary, MDA should be used advantageously at various stages of the Software Development Life Cycle in order to enrich development methodologies

and to enforce architecture conformance. The key technologies embodied within MDA tools should be focused towards enabling developers to maximize their activities during the MDA process.

Towards a Healthy Ecosystem

Essential to any MDA tool is the need to integrate all required technologies. Blueprint Modeling Environment (Blueprint ME) is the first integrated MDA modeling and development environment offering interoperability of all ingredients of the MDA process.

> *"Many MDA tools claim to enable Model-driven development, but reality shows that only the code-generation part of MDA is supported."*

Based upon and fully integrated with the Eclipse IDE, Blueprint ME is an extendible environment that allows integration with a variety of programming languages and features the following facilities in line with OMG's MDA standards.

- Meta-modeling
- UML modeling
- Model-to-model transformations
- Model-to-text transformations
- Transformations execution
- Model repository

Meta-modeling

The meta-modeling facility combines the creation of meta-models and UML Profiles. Meta-models can be considered as an explicit description of how a domain-specific model is built. As UML alone is not able to represent the semantics of a meta-model in a convenient way, UML profiles are used to provide additional queries and constraints to a UML model.

UML modeling

The UML modeling facility includes a UML 2.1 compliant visual modeler that supports UML profiles and constraint validation. The UML modeler's hierarchical process structure facilitates the orderly construct of flows depicting the MDA process.

Model-to-model transformations

The model-to-model transformation facility enables the definition of model transformations. This facility is based on QVT Operational and includes a highly intuitive graphical user interface that features test and debug facilities. Within the model-to-model transformations facility, users can associate models through drag and drop operations and define the structure of the transformation model. The specifics of every operation can be defined using a property editor that guides the user towards further definition of transformation centric rules. Step-by-step analysis and test of transformations are handled by the integrated transformation debugger. Historical data on transformations is maintained and recorded.

Model-to-text transformations

The model-to-text transformation facility permits the design of artifacts in compliance with the M2T standard. The interface includes split window editors that separate template definition from the queries. The M2T implementation is template driven. Textual artifacts can be generated in arbitrary forms supporting regeneration without loss of manual changes made to previously generated output. The template editor can be extended with a user defined syntax coloring scheme.

Transformation execution engine

The transformation execution engine enables the user to define workflows to perform multiple sequential transformations (both QVT and M2T). Workflows can be defined in a way that the output of one transformation is the input of the next.

"Last but not least, usability is a key factor when it comes to MDA tool suites."

Model repository

The model repository facilitates storage for reusable modeling artifacts. The repository includes versioning and keeps track of artifact changes. The model repository allows storage of Models, meta-models, UML profiles, UML libraries, transformations and workflow definitions.

Usability

Last but not least, usability is a key factor when it comes to MDA tool suites. The most important usability requirements are the interoperability between the various standards and the integration of these standards and facilities in one single development environment. Blueprint ME combines those requirements with a highly intuitive user interface including features like real-time transformation debugging that allows users – both experienced and novice – to understand the complexity of MDA and unleash the power of model-driven development.

Blueprint ME integrated MDA IDE

Blueprint ME, is the first integrated modeling environment that allows end users to effectively produce complete designs in compliance with OMG's Model Driven Architecture standards (MOF, UML, OCL, XMI, QVT and M2T).

To find out more about Blueprint ME, please visit www.atportunity.com.

WHITE PAPER

About @-portunity

@-portunity is an emerging enterprise with core competencies in Model Driven Architectures. Our management team is comprised of seasoned professionals with years of collective experience in the areas of Software Architecture Development, and advanced Software Development Methodologies as applied in resolving real world situations. Our flagship product, the Blueprint Modeling Environment, is the first integrated suite of tools that allows end users to effectively produce complete designs in compliance with OMG's Model Driven Architecture standards. @-portunity is member of the Object Management Group™. To find out more visit us at www.atportunity.com.