Model-driven architecture

From Wikipedia, the free encyclopedia

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.[1]

Contents

Overview

The model-driven architecture approach defines system functionality using a platform-independent model (PIM) using an appropriate domain-specific language (DSL).

Then, given a platform model corresponding to CORBA, .NET, the Web, etc., the PIM is translated to one or more platform-specific models (PSMs) that computers can run. This requires mappings and transformations and should be modeled too.

The PSM may use different DSLs or a general purpose language.. Automated tools generally perform this translation.

The OMG organization provides rough specifications rather than implementations, often as answers to Requests for Proposals (RFPs). Implementations come from private companies or open source groups.

MDA principles can also apply to other areas such as business process modeling (BPM) where the PIM is translated to either automated or manual processes.

Related standards

The MDA model is related to multiple standards, including the Unified Modeling Language (UML), the Meta-Object Facility (MOF), XML Metadata Interchange (XMI), Enterprise Distributed Object Computing (EDOC), the Software Process Engineering Metamodel (SPEM), and the Common Warehouse Metamodel (CWM). Note that the term “architecture” in Model-driven architecture does not refer to the architecture of the system being modeled, but rather to the architecture of the various standards and model forms that serve as the technology basis for MDA.

Executable UML is another specific approach to implement MDA

Trademark

The Object Management Group holds trademarks on MDA, as well as several similar terms including Model Driven Application Development, Model Based Application Development, Model Based Programming, and others.[2] The main acronym that has not yet been deposited by OMG until now is Model-driven engineering (MDE). As a consequence, the research community uses MDE to refer to general model engineering ideas, without committing to strict OMG standards.

Model-driven architecture topics

MDA approach

OMG focuses Model-driven architecture on forward engineering, i.e. producing code from abstract, human-elaborated modelling diagrams (e.g. class diagrams). OMG's ADTF (Analysis and Design Task Force) group leads this effort. With some humour, the group chose ADM (MDA backwards) to name the study of reverse engineering. ADM decodes to Architecture-Driven Modernization. The objective of ADM is to produce standards for model-based reverse engineering of legacy systems.[3] Knowledge Discovery Metamodel (KDM) is the furthest along of these efforts, and describes information systems in terms of various assets (programs, specifications, data, test files, database schemas, 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.

Of particular importance to model-driven architecture is the notion of model transformation. A specific standard language for model transformation has been defined by OMG called QVT.

MDA tools

The OMG organization provides rough specifications rather than implementations, often as answers to Requests for Proposals (RFPs). 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 metamodels.[4] In the following section "model" is interpreted as meaning any kind of model (e.g. a UML model) or metamodel (e.g. the CWM metamodel). In any MDA approach we have essentially two kinds of models: *initial models* are created manually by human agents while *derived models* 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 (e.g. author, date of creation or modification, method of creation (which tool? which transformation? etc.)) and the mutual relations between these models (i.e. one metamodel is a version of another one, one model has been derived from another one by a transformation, etc.)

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 characteristics of MDA tools is that they mainly take models (e.g. MOF models or metamodels) 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.

Implementations of the OMG specifications come from private companies or open source groups. One important source of implementations for OMG specifications is the Eclipse Foundation (EF). Many implementations of OMG modeling standards may be found in the Eclipse Modeling Framework (EMF) or Graphical Modeling Framework (GMF), the Eclipse foundation is also developing other tools of various profiles as GMT. Eclipse's compliance to OMG specifications is often not strict. This is true for example for OMG's EMOF standard, which Eclipse approximates with its ECORE implementation. More examples may be found in the M2M project implementing the QVT standard or in the M2T project implementing the MOF2Text standard.

One should be careful not to confuse the *List of MDA Tools* and the List of UML tools, the former being much broader. This distinction can be made more general by distinguishing 'variable metamodel tools' and 'fixed metamodel tools'. A UML CASE tool is typically a 'fixed metamodel tool' since it has been hard-wired to work only with a given version of the UML metamodel (e.g. UML 2.1). On the contrary, other tools have internal generic capabilities allowing them to adapt to arbitrary metamodels or to a particular kind of metamodels.

Usually MDA tools focus rudimentary architecture specification, although in some cases the tools are architecture-independent (or platform independent).

Simple examples of architecture specifications include:

- Selecting one of a number of supported reference architectures like Java EE or Microsoft .NET,
- Specifying the architecture at a finer level including the choice of presentation layer technology, business logic layer technology, persistence technology and persistence mapping technology (e.g. object-relational mapper).
- Metadata: information about data.

MDA concerns

Some key concepts that underpin the MDA approach (launched in 2001) were first elucidated by the Shlaer-Mellor method during the late 1980s. Indeed a key absent technical standard of the MDA approach (that of an action language syntax for Executable UML) has been bridged by some vendors by adapting the original Shlaer-Mellor Action Language (modified for UML). However during this period the MDA approach has not gained mainstream industry acceptance; with the Gartner Group still identifying MDA as an "on the rise" technology in its 2006 "Hype Cycle",[5] and Forrester Research declaring MDA to be "D.O.A." in 2006.[6] Potential concerns that have been raised with the OMG MDA approach include:

- Incomplete Standards: The MDA approach is underpinned by a variety of technical standards, some of which are yet to be specified (e.g. an action semantic language for xtUML), or are yet to be implemented in a standard manner (e.g. a QVT transformation engine or a PIM with a virtual execution environment).[7][8]
- Vendor Lock-in: Although MDA was conceived as an approach for achieving (technical) platform independence, current MDA vendors have been reluctant to engineer their MDA toolsets to be interoperable. Such an outcome could result in vendor lock-in for those pursuing an MDA approach.
- Idealistic: MDA is conceived as a forward engineering approach in which models that incorporate Action Language programming are transformed into implementation artifacts (e.g. executable code, database schema) in one direction via a fully or partially automated "generation" step. This aligns with OMG's vision that MDA should allow modelling of a problem domain's full complexity in UML (and related standards) with subsequent transformation to a complete (executable) application.[9] This approach does, however, imply that changes to implementation artifacts (e.g. database schema tuning) are not supported . This constitutes a problem in situations where such post-transformation "adapting" of implementation artifacts is seen to be necessary. Evidence that the full MDA approach may be too idealistic for some real world deployments has been seen in the rise of so-called "pragmatic MDA".[10] Pragmatic MDA blends the literal standards from OMG's MDA with more traditional model driven mechanisms such as round-trip engineering that provides support for adapting implementation artifacts.
- Specialised Skillsets: Practitioners of MDA based software engineering are (as with other toolsets) required to have a high level of expertise in their field. Current expert MDA practitioners (often referred to as Modeller/Architects) are scarce relative to the availability of traditional developers.[11]
- OMG Track Record: The OMG consortium who sponsor the MDA approach (and own the MDA trademark) also introduced and sponsored the CORBA standard which itself failed to materialise as a widely utilised standard.[12]

- Uncertain Value Proposition (UVP): As discussed, the vision of MDA allows for the specification of a system as an abstract model, which may be realized as a concrete implementation (program) for a particular computing platform (e.g. .NET). Thus an application that has been successfully developed via a pure MDA approach could theoretically be ported to a newer release .NET platform (or even a Java platform) in a deterministic manner – although significant questions remain as to real-world practicalities during translation (such as user interface implementation). Whether this capability represents a significant value proposition remains a question for particular adopters. Regardless, adopters of MDA who are seeking value via an "alternative to programming" should be very careful when assessing this approach. The complexity of any given problem domain will always remain, and the programming of business logic needs to be undertaken in MDA as with any other approach. The difference with MDA is that the programming language used (e.g. xtUML) is more abstract (than, say, Java or C#) and exists interwoven with traditional UML artifacts (e.g. class diagrams). Whether programming in a language that is more abstract than mainstream 3GL languages will result in systems of better quality, cheaper cost or faster delivery, is a question that has yet to be adequately answered.
- MDA was recognized as a possible way to bring various independently developed standardized solutions together. For the simulation community, it was recommended as a business and industry based alternative to yet another US DoD mandated standard.[13]

Conferences

Among the various conferences on this topic we may mention ECMDA (http://www.ecmda-fa.org/), the European Conference on MDA and also MoDELS, former firmed as <<UML>> conference series (till 2004), the Italian Forum on MDA (http://mdaforum.soluta.net) in collaboration with the OMG. There are also several conferences and workshops (at OOPSLA, ECOOP mainly) focusing on more specific aspects of MDA like model transformation, model composition, and generation.

Code generation controversy

Code generation means that the user abstractly models solutions, which are connoted by some model data, and then an automated tool derives from the models parts or all of the source code for the software system. In some tools, the user can provide a skeleton of the program source code, in the form of a source code template where predefined tokens are then replaced with program source code parts during the code generation process.

An often cited criticism is that the UML diagrams just lack the detail which is needed to contain the same information as is covered with the program source. Some developers even claim that "the Code *is* the design".[14][15]

See also

- ATLAS Transformation Language
- Model Transformation Language

- Automatic programming
- Domain-driven design
- Enterprise Resource Planning
- Executable UML
- Executable Architecture
- Meta-Object Facility
- Metamodeling
- Model-driven engineering
- Model-driven integration
- Model-driven security
- Model Driven Interoperability
- Modeling Maturity Levels
- Platform-independent model
- Platform-specific model
- Software factory
- Unified Modeling Language
- Universal Systems Language
- QVT
- Web engineering
- WebML

References

1. "OMG pursues new strategic direction to build on success of past efforts" (http://www.omg.org/news/releases/pr2001/2001-03-08a.htm)
2. http://www.omg.org/legal/tm_list.htm
3. adm website http://adm.omg.org
4. Bézivin, J, Gérard, S, Muller, P-A, and Rioux, L (2003). "MDA components: Challenges and Opportunities" (PDF). In: Metamodelling for MDA.
5. "Hype Cycle for Emerging Technologies, 2006" (http://www.gartner.com/DisplayDocument?ref=g_search&id=494180&subref=simplesearch) $495.00
6. "MDA Is DOA, Partly Thanks To SOA" (http://www.forrester.com/Research/Document/Excerpt/0,7211,39156,00.html)
7. "UML - Unified or Universal Modeling Language? UML2, OCL, MOF, EDOC - The Emperor Has Too Many Clothes" (http://www.jot.fm/issues/issue_2003_01/column1)
8. "MDA: Nice Idea. Shame about the..." (http://www.theserverside.com/tt/articles/article.tss?l=MDA_Haywood)
9. "Bringing MDA to Eclipse, using a pragmatic approach" (http://www.java-forum-stuttgart.de/jfs/2006/folien/A5_Schoenhage_Compuware.pdf)
10. "A Response to Forrester" (http://www.bptrends.com/publicationfiles/04-06-COL-MDA-ResponseToForrester-Frankel.pdf#search=%22%22pragmatic%20MDA%22%22)
11. "Are You Ready For the MDA?" (http://www.agilemodeling.com/essays/readyForMDA.htm)
12. "The Rise and Fall of CORBA" (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=396)
13. "Avoiding Another Green Elephant" (http://arxiv.org/ftp/arxiv/papers/1011/1011.6671.pdf)
14. http://www.developerdotstar.com/mag/articles/reeves_design_main.html by Jack W. Reeves
15. Bleading-Edge (http://www.bleading-edge.com/)

Further reading

- Kevin Lano. "Model-Driven Software Development With UML and Java". CENGAGE Learning, ISBN 978-1-84480-952-3
- David S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, ISBN 0-471-31920-1
- Meghan Kiffer *The MDA Journal: Model Driven Architecture Straight From The Masters*. ISBN 0-929652-25-8

- Anneke Kleppe (2003). *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison-Wesley. ISBN 0-321-19442-X
- Stephen J. Mellor (2004). *MDA Distilled, Principles of Model Driven Architecture*. Addison-Wesley Professional. ISBN 0-201-78891-8
- Chris Raistrick. *Model Driven Architecture With Executable UML*. Cambridge University Press, ISBN 0-521-53771-1
- Marco Brambilla, Jordi Cabot, Manuel Wimmer, *Model Driven Software Engineering in Practice*, foreword by Richard Soley (OMG Chairman), Morgan & Claypool, USA, 2012, Synthesis Lectures on Software Engineering #1. 182 pages. ISBN 9781608458820 (paperback), ISBN 9781608458837 (ebook). http://www.mdse-book.com
- Stanley J. Sewall. *Executive Justification for MDA*
- Soylu A., De Causmaecker Patrick. *Merging model driven and ontology driven system development approaches pervasive computing perspective*, in Proc 24th Intl Symposium on Computer and Information Sciences. 2009, pp 730–735.

External links

- OMG's MDA Web site (http://www.omg.org/mda/)
- Model-Driven Software Development Course, B. Tekinerdogan, Bilkent University (http://www.cs.bilkent.edu.tr/~bedir/CS587-MDSD/)

Categories: Systems engineering | Unified Modeling Language
