

# UML<sup>®</sup> vs. DSLs: A false dichotomy

omg/08-09-03

**Andrew Watson**  
**Vice-President and Technical Director**  
**Object Management Group<sup>™</sup>**

The article on Microsoft's Domain-Specific Language (DSL) Tools in the March 2007 edition of "Butler Group Review" may have left the impression that modelling using General-Purpose Languages (such as UML) and Domain-Specific Languages are incompatible opposites. In fact the situation is more nuanced; there's a spectrum of standardisation from strict use of GPL standards, though standard language dialects, all the way to creating and using bespoke Domain-Specific Languages for particular applications. Languages across this spectrum are compatible with Model-Driven Development (MDD) techniques, including OMG's Model-Driven Architecture<sup>®</sup> (MDA<sup>®</sup>). This article is a short overview of the relationship between DSLs and GPLs, their common basis on meta-data frameworks like OMG's MOF<sup>™</sup>, and the application of both to MDD and MDA.

Let's start with a short history of graphical software modelling. It divides cleanly into two eras - "Before UML" and "After UML".

Before the creation of the Unified Modelling Language (UML) standard, different modelling gurus advocated incompatible notations, deterring many potential users. The modelling tools market was consequently tiny and fragmented, and models were used only for sketching and documenting, rather than fully integrated into the software development process. That began to change in the mid-1990s when the Object Management Group (OMG) acted as forum for agreement between thought-leaders in the field, and the UML standard was born. By removing incompatibilities between competing notations, UML eliminated the major obstacle to widespread use of graphical modelling, and spectacular growth in the field ensued. By 2004 BZ research reported more than 2/3 of software development organisations using UML, with 82% predicting they would use it in future. According to Gartner, UML is now used by more than 10 million IT professionals. The existence of a standard notation set has released pent-up demand and created an industry.

UML itself is a family of 13 closely-related diagramming notations, each tailored to a different aspect of application design, and all defined using the Meta-Object Framework (MOF), OMG's standard for defining modelling languages. UML's MOF foundation means that a UML diagram is more than just a pretty picture - a MOF-aware modelling tool captures the meaning of the diagram in machine-readable form, allowing it to be used as the basis for automatically generating parts of the application code. This MOF basis is essential to use of OMG's Model-Driven Architecture (MDA), a framework for creating software designs that copes with multiple implementation technologies and the need for maintenance over extended software lifetimes. MDA uses MOF-defined models to create and manipulate precise, detailed, machine-readable descriptions of application structure and behaviour that are independent of what programming languages, operating systems or databases may be used to implement them. MDA is supported by OMG's several hundred member companies, who include both software users and vendors, and its basis in freely-available standards has resulted in a thriving community of tool vendors and open-source tool offerings. Use of MOF-based standards allows tools from multiple vendors to be used together on a single project - a vital feature, since even the largest tool vendor may not provide support for all necessary software platforms. An MDA deployment at Deutsche Bank Bauspar serves to emphasise this; a maintenance upgrade of an existing COBOL back-office mainframe system to integrate it with a Web-based front-end involved the creation of around a

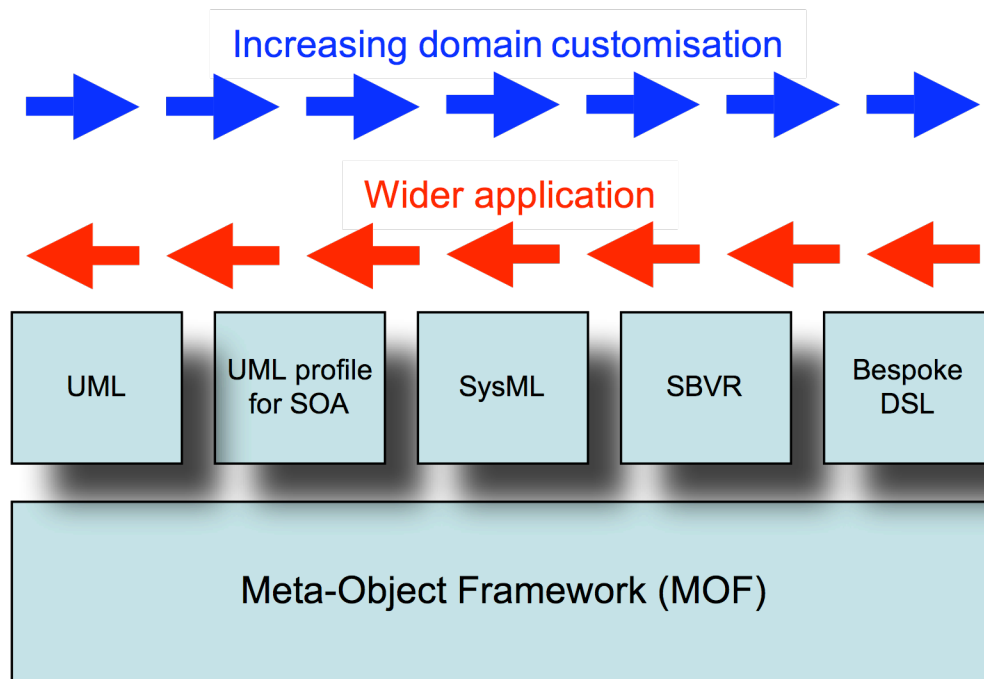


Figure 1: Modelling languages from General-Purpose to completely Domain-Specific, all based on the MOF language-definition framework.

dozen different kinds of software artefacts, including COBOL modules, Oracle database schemas, DB2 database schemas, EJBs, XML-RPC interface definitions and JUnit test classes.

Another key MDA goal is lowering software maintenance costs. Creating an application may incur as little as 10% of the total cost of owning it over its lifetime. A recent MDA deployment at Disney Corporation underlined that while MDA can significantly reduce the cost of software development, even greater savings flow from the lower cost of maintaining higher-quality MDA-developed code, coupled with the use of machine-readable software models as long-term system documentation. Using widely-available open standards is vital to achieving these benefits, and although many people associate MDA exclusively with UML, from a technical standpoint the key MDA standard is not UML, but MOF. Strict use of standard UML lies at one end of a continuum of modelling languages, stretching from widely-used, standard GPLs all the way to bespoke DSLs specifically created for particular niche domains, and all supported by the MOF language-definition framework. The first step towards greater customisation is use of UML's lightweight customisation mechanism for creating "dialects" (called "profiles") tailored to particular platforms. Loading a profile definition into a standard UML tool allows it to support (and enforce) the characteristics of a particular platform. There are standard profiles for (amongst others) real-time systems & fault-tolerant systems, and work is underway on a standard profile for Service Oriented Architecture (SOA) platforms.

After profiles, the next step in this progression is languages more loosely based on standard GPLs, but reusing much of their syntax and semantics. OMG's Systems Modelling Language (OMG SysML™) standard is a good example. SysML is a DSL for graphical modelling of complex systems that include hardware, software, information, processes, personnel, and facilities. SysML reuses seven of UML 2's thirteen diagram types, and adds two more - requirements diagrams and parametric diagrams. SysML isn't a strict UML profile, but is instead defined by applying the UML profile mechanism to a subset of the language. Because SysML is based on UML, it's easy for systems engineers familiar with UML to

learn it, and for existing UML tool providers to support it. Use of a standard MOF foundation means that MOF tools can manipulate SysML models as easily as UML models.

Further along the continuum are standard DSLs not based on UML, such as OMG's new Semantics of Business Vocabulary and Rules (SBVR) standard. SBVR defines a MOF metamodel for representing business rules, but has its own text-based rules notations. Although this standard DSL for business rules seems destined to be widely used, it uses no UML-derived diagram types at all. However, once again, it shares a common MOF foundation with UML, and MOF model-manipulation tools can be used with SBVR-defined models, bringing the benefits of metamodeling tool standards to this domain.

Finally, the last point on the continuum is the creation of bespoke, user-defined modelling languages for particular domains. Creating a bespoke DSL is a complex task, but can be justified if the productivity and modelling-fidelity benefits are great enough, as discussed in March's article "Domain Specific Languages in Practice". However, creating a bespoke DSL for a niche domain does not necessitate abandoning standards; the MOF language definition framework can once again be used to define the DSLs and avoid vendor lock-in.

Everyone who uses a standard GPL such as UML benefits from others' use of the same language, because of the large pool of expertise in the industry, the thriving tools market, and healthy competition between tool vendors. Interestingly, and less obviously, the creators and users of DSLs can also benefit from these so-called "externalities". The visual syntax and semantics of DSLs are often reminiscent of widely-used standards like UML; for instance, Keith Short, one of the architects of Microsoft's DSL tools, described a Class Designer created using those tools and shipped by Microsoft as "familiar to anyone used to UML notation". It's quite possible that widespread adoption of a DSL is only possible if most of its users are already familiar with an existing standard language, allowing new users to be quickly be trained by explaining that it's "just like standard language X, except ...". Without common knowledge of the standard language, the learning curve for a new DSL may be too steep to permit general uptake. This is very noticeable in the world of domain-specific programming languages, where DSLs like the AWK text-processing language and the C-shell scripting language are clearly based on existing general-purpose languages (C, in this case). Hence, paradoxically, it's arguable that the market for DSLs and DSL tools would be a lot smaller without standardised GPLs (and in particular UML) from which to borrow syntax and semantics.

To conclude: the overwhelming predominance of UML in the modelling market means that most modellers use UML, and most users of MDA tools use UML. However, MDA and UML are orthogonal - there are UML users who do not use MDA (including many engineers who use imprecise but useful "UML sketches" to communicate with each other), and MDA users who use other modelling languages, including DSLs. It's also a mistake to regard GPLs (including UML) and DSLs as incompatible opposites. Both can be used within the same MOF standards framework, allowing end-users to enjoy their respective benefits while minimising the risk of becoming locked in to proprietary technology. As David Norton of Gartner recently wrote, DSLs and UML should be viewed as "fraternal twins, not competitors"

Copyright © 2008 Object Management Group (OMG)

This article originally appeared in the April 2007 edition of Butler Group Review.