

Domain-Driven Software Development — A World of Transformations

Shane Sendall

IBM Research
Zurich Research Laboratory
CH-8803 Rüschlikon, Switzerland
sendall@acm.org

Extended Abstract:

Software development teams are faced with bridging the gap between the problem, as envisaged by the stakeholders and constrained by the environment, and a software solution, which is built upon the abstractions offered by current software technologies. Unfortunately, too often the abstractions offered are limited and disparate with respect to the problem space. Reducing this gap would facilitate more sophisticated problems to be tackled in software development projects, and it would comparatively reduce development costs and time-to-market, and remove errors caused by the disparity. In this talk, I will explore a number of techniques for improving current software development practice, which relate to the theme of domain-driven software development.

Domain-driven software development is concerned with making use of languages that better capture the problem by using abstractions that are more familiar to experts in the domain. These domain-specific languages are made executable either directly (compilation or interpretation) or through tool-supported refinement/elaboration to computational models that can be executed, e.g., to a mainstream programming language where one can make use of existing frameworks, components, services, etc. In the later case, real value is added to software development only if we can automate as much as possible the transformation step(s). Automating these steps requires languages that can express such transformations in a concise and maintainable manner.

The principles of abstraction, separation of concerns, and problem decomposition are essential in providing intuitive and manageable domain-specific languages. The practice of software modeling has become a significant way of applying these principles to software development. Over the last few years, the software development industry has gone through the process of standardizing visual modeling notations. The Unified Modeling Language (UML) [UML] is the product of this effort, and it unifies scores of notations that were proposed in the '80s and '90s. The language has gained significant industry support and became an Object Management Group (OMG) standard in 1997. Nowadays, the majority of software modeling techniques and approaches use UML.

UML gives numerous options to developers for specifying software systems. A UML model can graphically depict the structure and/or behavior of the system under discussion from a certain viewpoint and at a certain level of abstraction. This is desirable as one can typically better manage the complexities of a system description through the use of multiple models, where each captures a different aspect of the solution. Models can also be refined and decomposed into other models. Thus, models can be utilized not only in a horizontal manner (to describe different system aspects), but also in a vertical manner (to be refined from higher to lower levels of abstraction). Nevertheless, working with multiple, interrelated models that describe a software system require significant effort to ensure their overall consistency.

It is also necessary to point out that models are used in software development with a number of purposes. These include:

- **Problem Documentation**
The model is used to communicate an understanding of the problem, and/or ensure that there is a common understanding of the problem between stakeholders.
- **Solution Blueprints**
The model defines a view of the structure/function/behavior of system to develop, which is elaborated to produce the final system.
- **Solution Documentation**
The model is used to document the implemented system.
- **Analysis**
The model is used to explore the problem or solution space.
- **Verification/Validation**
The model can be used as a means to gauge the correctness of the delivered system.
 - Informal: inspection, review, testing, etc.
 - Formal: specification-based testing, proofs (refinement, existence/absence of properties), animation of model
- **Execution**
The model is the code.

Being aware of various purposes that a model can possess, e.g., according to the categories given above, allows methodologists to clearly create and integrate models into software development activities. In fact, significant tool support is only possible once a systematic approach is devised for relating and composing models.

In this presentation, I will present the vision of domain-driven software development and briefly discuss how it relates to current standards and initiatives such as the Unified Modeling Language (UML), the Meta Object Facility (MOF) [MOF], Model Driven Architecture (MDA) [MDA, KWB03], and OMG's Query/View/Transformation (QVT) Request for Proposal [QVT]. Thereafter, I will describe various techniques and approaches that can be used in the realization of the transformations needed in the context mentioned above. I will also present a number of ideas for improving the way models can be conceived and developed.

Biography:

Dr. Shane Sendall is a researcher in the e-business solutions group at IBM Zurich Research Laboratory. His research interests are focused on improving model-based software development practices. In particular, he is enthused by: the foundations and applications of model transformation and techniques and languages that support it, software development tools that support UML and OCL specification, techniques to smooth and/or automate the transition from software requirements to design artifacts. He received his PhD in computer science from the Swiss Federal Institute of Technology in Lausanne. Contact him at sendall@acm.org

References

- [KWB03] A. Kleppe, J. Warmer, and W. Bast; “MDA Explained: the Practice and Promise of Model-Driven Architecture”. Addison-Wesley, 2003.
- [MDA] OMG Model Driven Architecture web site; <http://www.omg.org/mda/>
- [MOF] OMG Meta Object Facility web site;
<http://www.omg.org/technology/documents/formal/mof.htm>
- [QVT] OMG TC; “MOF 2.0 Query/Views/Transformations RFP”.
<http://cgi.omg.org/cgi-bin/doc?ad/02-04-10>
- [UML] OMG Unified Modeling Language web site; <http://www.uml.org/>