Globalizing Modeling Languages

Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert B. France,
Jean-Marc Jézéquel, Jeff Gray

Globalizing Modeling Languages

Supporting Language Heterogeneity in the Development and Runtime Management of Software-Intensive Systems

Benoit Combemale [a], Julien DeAntoni**, Benoit Baudry[a], Robert B. France***, Jean-Marc Jézéquel*, Jeff Gray*****

[a] Inria, France
** University of Rennes 1, France*
*** University of Nice Sophia-Antipolis, France*
**** Colorado State University, USA*
***** University of Alabama, USA*

In the software and systems modeling community, research on domain-specific modeling languages (DSMLs) is focused on providing technologies for developing languages and tools that allow domain experts to develop system solutions efficiently. Unfortunately, the current lack of support for explicitly relating concepts expressed in different DSMLs makes it very difficult for software and system engineers to reason about information spread across models describing different system aspects. Supporting coordinated use of DSMLs leads to what we call the globalization of modeling languages. This article presents a research initiative that broadens the DSML research focus beyond the development of independent DSMLs to one that supports globalized DSMLs, that is, DSMLs that facilitate coordination of work across different domains of expertise.

Domain-Specific Modeling Languages

Model-Driven Engineering (MDE) aims at reducing the accidental complexity associated with developing complex software-intensive systems [1]. A primary source of accidental complexity is the wide gap between the high-level concepts used by domain experts to express their specific needs and the low-level abstractions provided by general-purpose programming languages [2]. Manually bridging this gap, particularly in the presence of changing requirements, is costly in terms of both time and effort. MDE approaches address this problem through the use of modeling techniques that support separation of concerns and automated generation of major system artifacts (e.g., test cases, implementations) from models. In MDE, a model describes an aspect of a system and is typically created for specific development purposes. Separation of concerns is supported through the use of different modeling languages, each providing constructs based on abstractions that are specific to an aspect of a system. For example, Generalized Stochastic Petri Nets can be used to create performance models, while the notation provided by the Simulink tool is adapted to simulation models. MDE technologies also provide support for manipulating models; for example, support for querying, transforming, merging, and analyzing (including executing) models. Modeling languages are thus at the core of MDE.

Incorporating domain-specific concepts and high-quality development experience into MDE technologies can significantly improve developer productivity and system quality. This realization has led to work, starting in the late nineties, on MDE language workbenches that support the development of domain-specific modeling languages (DSMLs) and associated tools (e.g., model editors and code generators). A DSML provides a bridge between the (problem) space in which domain experts work and the implementation (programming) space. Domains in which DSMLs have been developed and used include automotive, avionics, and cyber-physical systems.

Hutchinson et al. provide some indications that *DSMLs can pave the way for wider industrial adoption of MDE* [3]. Research on systematic development of DSMLs has produced a technology base that is robust enough to support the integration of DSML development processes into large-scale industrial system development environments. Current DSML workbenches support the development of DSMLs to create models that play pivotal roles in different development phases. Workbenches such as Microsoft's DSL tools, MetaCase's MetaEdit+, JetBrains' MPS, Eclipse Modeling Framework (EMF) and Generic Modeling Environment (GME) support the specification of the abstract syntax, concrete syntax and the static and dynamic semantics of a DSML. These workbenches address the needs of DSML developers in a variety of application domains.

The development of modern complex software-intensive systems often involves the use of multiple DSMLs that capture different system aspects. In addition, models of the system aspects are seldom manipulated independently of each other. System engineers are thus faced with the difficult task of relating information presented in different models. For example, a system engineer may need to analyze a system property that requires information scattered in models expressed in different DSMLs. Current DSML development workbenches provide good support for developing independent DSMLs, but provide little or no support for integrated use of multiple DSMLs. *The lack of support for explicitly relating concepts expressed in different DSMLs makes it very difficult for developers to reason about information spread across different models.*

Globalized DSML Challenge: Looking Ahead

Past research on modeling languages focused on their use to bridge wide problem-implementation gaps. A new generation of complex software-intensive systems, for example, smart health, smart grid, building energy management, and intelligent transportation systems, presents new opportunities for leveraging modeling languages. The development of these systems requires expertise in a variety of domains. Consequently, different types of stakeholders (e.g., scientists, engineers and end-users) must work in a coordinated manner on various aspects of the system across multiple development phases. DSMLs can be used to support the work of domain experts who focus on a specific system aspect, but they can also provide the means for coordinating work across teams specializing in different aspects and across development phases.

Supporting coordinated use of DSMLs leads to what we call *the globalization of modeling languages*, that is, *the use of multiple modeling languages to support coordinated development of diverse aspects of a system*. One can make an analogy with world globalization in which relationships are established between sovereign countries to regulate interactions (e.g., travel and commerce related interactions) while preserving each country's independent existence. The term "DSML globalization" is used to highlight the desire that DSMLs developed in an independent manner to meet the specific needs of domain experts, should also have an associated framework that regulates interactions needed to support collaboration and work coordination across different system domains.

Globalized DSMLs aim to support the following critical aspects of developing complex systems: *communication* across teams working on different aspects, *coordination* of work across the teams, and *control* of the teams to ensure product quality.

In the globalized DSML vision, integrated DSMLs support teams working on systems that span many domains and concerns to determine how their work on a particular aspect influences work on other concerns. The objective is to offer support for communicating relevant information, and for coordinating

development activities and associated technologies within and across teams, in addition to providing support for imposing control over development artifacts produced by multiple teams.

Coordination and related separation of concerns issues have been the focus of software engineering since early work on modularizing software. For example, Parnas' use of the term "work product" to denote a module that can be the source of independent development is also a focus of team demarcation across design and implementation tasks. Modularity in modern software-intensive systems development leads to well-known coordination problems, such as problems associated with coordinating work over temporal, geographic or socio-cultural distance [4]. This has also led to the recognition of socio-technical coordination, including coordination of the stakeholders and the technologies they use to perform their development work, as a major system development challenge [5].

In this context, DSMLs can be used to support socio-technical coordination by providing the means for stakeholders to bridge the gap between how they perceive a problem and its solution, and the programming technologies used to implement a solution. DSMLs also support coordination of work across multiple teams when they are supported by mechanisms for specifying and managing their interactions. In particular, proper support for coordinated use of DSMLs leads to language-based support for *social translucence*, where the relationships between DSMLs are used to extract the information needed to make a team working on a system aspect aware of the relevant work performed by teams working on other aspects. Such awareness is needed to minimize the counter-productive form of social isolation that can occur when work is distributed across different teams.

On the Globalization of Modeling Languages

To support globalization, the use of multiple heterogeneous modeling languages will need to be related to determine how different aspects of a system influence each other. We identify three forms of relationships among modeling languages that can be used to support interactions across different system aspects: *interoperability, collaboration,* and *composition*.

Interoperable modeling languages provide support for the exchange of information across models expressed in the languages. Interoperable DSMLs can be developed in a relatively independent manner, but relationships defined across the different DSMLs allow information expressed in one model to be related to information contained in models expressed in different DSMLs. These DSML relationships facilitate the development of integrated modeling tool chains in which information from a model built for a specific purpose (e.g., a SysML model used to describe system architecture) is used to decorate a model that serves a different purpose (e.g., a Generalized Stochastic Petri Net used for performance analysis). Interoperable DSMLs have the lowest coupling of the three relationships we identified; the focus is on supporting coordinated use of modeling tools, as opposed to tightly coupling the development of the different models.

Collaboration relationships among modeling languages are used to support coupled development of models. DSMLs in a collaboration relationship are referred to as *collaborative modeling languages*. The development of a model expressed in a collaborating modeling language can directly influence the form of models created using other collaborating modeling languages. For example, consistency relationships defined across the DSMLs can be used by developers to ensure that the different models they create are consistent with each other. Model authoring tools for collaborating DSMLs are thus coupled. Collaborating DSMLs can be used to support *a priori* as well as *a posteriori* global analysis of properties.

Interoperable and collaborating DSMLs support DSML interactions without deriving new forms of information from information spread across different models. However, there are situations that call for combining information scattered in other models to create new forms; for example, to support generation

of system documentation, test cases, or to provide support for simulating global system behavior. Model *composition* (e.g., weaving and merging) is thus the third form of interaction that is facilitated by explicit definitions of relationships across elements in different DSMLs.

The prior discussion focused on the use of DSMLs to coordinate the work of developers. These ideas can be applied at various development life-cycle phases, ranging from early analysis to system runtime. Models can also be used to coordinate work done by different components, subsystems, or services. The use of DSMLs to coordinate work can potentially have a beneficial impact on the management of running systems. Different types of models are currently being used as runtime abstraction layers to support reasoning about the system or even adapting it [6]. Explicitly defined relationships across DSMLs for runtime models can be leveraged by these model-based runtime environments to coordinate the manipulation of models at runtime.

Challenging issues will need to be addressed to realize the above forms of language integration. Relationships among the languages will need to be explicitly defined in a form that corresponding tools can use to realize the desired interactions. Requirements for tool manipulation are thus another topic that will be a focus for future work in the area of DSML globalization.

This grand challenge is currently supported by the GEMOC initiative (see http://gemoc.org). GEMOC is an open international initiative that brings together a community to develop software language engineering breakthroughs that support interoperable, collaborative, and composable modeling languages. The GEMOC initiative provides a framework that facilitates collaborative work on the globalization of DSMLs.

[1] Schmidt, Douglas C. Guest editor's introduction: Model-driven engineering. Computer, IEEE, 2006, vol.39, n°2, p.25-31
[2] France, Robert and Rumpe, Bernhard. Model-driven development of complex software: A research roadmap. In: Future of Software Engineering. IEEE Computer Society, 2007. p.37-54
[3] Hutchinson, John, Whittle, Jon, Rouncefield, Mark, et al. Empirical assessment of MDE in industry. In: 33rd International Conference on Software Engineering. ACM, 2011. p.471-480
[4] Herbsleb, James D. and Grinter, Rebecca E. Architectures, coordination, and distance: Conway's law and beyond. Software, IEEE, 1999, vol. 16, n° 5, p.63-70
[5] Herbsleb, James D. Global software engineering: The future of socio-technical coordination. In: Future of Software Engineering. IEEE Computer Society, 2007. p.188-198
[6] Blair, Gordon, Bencomo, Nelly, and France, Robert B. Models@ run.time. Computer, IEEE, 2009, vol. 42, n°10, p.22-27

Benoit Combemale *is an associate professor at the University of Rennes and a research scientist at Inria. His research interests include model-driven software engineering, software language engineering, domain-specific languages and Validation & Verification. Contact him at benoit.combemale@inria.fr*

Julien Deantoni *is an associate professor in the Department of Computer Science at the University of Nice Sophia Antipolis. His research focuses on the connection between model driven engineering and concurrency theory by using the notion of logical time. Contact him at julien.deantoni@inria.fr*

Benoit Baudry *is a research scientist at Inria. His research interests include metamodeling and model-based software analysis and testing. Contact him at benoit.baudry@inria.fr*

Robert B. France *is a professor in the Department of Computer Science at Colorado State University. His research focuses on model-driven development, models at runtime, software-product lines, domain-specific languages and formal methods. Contact him at france@cs.colostate.edu*

Jean-Marc Jézéquel *is a professor at the University of Rennes and director of IRISA. His research focuses on the theory and practice of Model Driven Engineering. Contact him at jezequel@irisa.fr*

Jeff Gray *is a professor in the Department of Computer Science at the University of Alabama. His research focuses on model transformation, domain-specific languages, and computer science education. Contact him at gray@cs.ua.edu*