Coordinating domain-specific modeling languages provides support for language heterogeneity in software-intensive systems’ development and runtime management.

In the software and systems modeling community, domain-specific modeling language (DSML) research is focused on providing technologies for developing languages and tools that allow domain experts to develop system solutions efficiently. Unfortunately, it’s very difficult for software and systems engineers to reason about information spread across models describing different system aspects because of the current lack of support for explicitly relating concepts expressed in different DSMLs. Here, we describe a research initiative that broadens the DSML research focus beyond independent DSML development 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 to reduce the accidental complexity associated with developing complex software-intensive systems. A primary source of this 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. Manually bridging this gap, particularly in the presence of changing requirements, is costly in terms of both time and effort. MDE approaches this problem through the use of modeling techniques that support separation of concerns and automated generation of major system artifacts (for example, test cases and 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, whereas the notation provided by the Simulink tool is adapted to simulation models. MDE technologies also provide support for manipulating models, such as for querying, transforming, merging, and analyzing (including executing) models. Modeling languages are thus at MDE’s core.
Incorporating domain-specific concepts and high-quality development experience into MDE technologies can significantly improve developer productivity and system quality. This tactic has led to work, starting in the late 1990s, on MDE language workbenches that enable the development of tool-supported DSMs. A DSM bridges the problem space in which domain experts work and the implementation (or programming) space. Domains in which DSMs have been developed and used include automotive, avionics, and cyberphysical systems.

John Hutchinson and his colleagues provided some indication that DSMs can pave the way for wider industrial adoption of MDE. Research on systematic DSM development has produced a technology base robust enough to support the integration of DSM development processes into large-scale industrial system development environments. Current DSM 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, the Eclipse Modeling Framework (EMF), and the Generic Modeling Environment (GME) support the specification of the abstract syntax, concrete syntax, and static and dynamic semantics of a DSM. These workbenches address DSM developers’ needs in a variety of application domains.

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

GLOBALIZED DSM CHALLENGE: LOOKING AHEAD

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

Supporting coordinated use of DSMs leads to what we call the globalization of modeling languages. This has also led to the recognition that sociotechnical coordination, including

Supporting coordinated use of DSMs leads to what we call the globalization of modeling languages.

trol of the teams to ensure product quality. The objective is to offer support for communicating relevant information, coordinating development activities and associated technologies within and across teams, and imposing control over development artifacts produced by multiple teams.

Coordination and related separation of concerns issues have been software engineering’s focus since early work on modularized software. David 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 sociocultural distance. This has also led to the recognition that sociotechnical coordination, including
coordination of the stakeholders and the technologies they use to perform their development work, is a major systems development challenge.\textsuperscript{5} DSMs support sociotechnical coordination by providing the means for stakeholders to bridge the gap between how they perceive a problem and its solution on the one side, and the programming technologies used to implement that solution on the other. When they’re supported by mechanisms for specifying and managing their interactions, DSMs also support coordination of work across multiple teams. In particular, proper support for coordinated use of DSMs leads to language-based support for social translucence, where the relationships between DSMs are used to extract the information needed to make a development team aware of relevant information needed to make a different purpose (such as a generalized stochastic Petri net used for performance analysis). Interoperable DSMs 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 model development.

Collaboration relationships among modeling languages provide support for coupled model development. DSMs in such a relationship are referred to as collaborative modeling languages. The model development expressed in a collaborative modeling language can directly influence the form and the correction of models created using other collaborative modeling languages. For example, developers can use consistency relationships defined across DSMs to ensure consistency among the different models they create. Model-authoring tools for collaborative DSMs are thus coupled. Collaborative DSMs can support a priori as well as a posteriori global analysis of properties.

Interoperable and collaborative DSMs support DSM interactions without deriving new forms of information from that which is spread across different models. However, some situations call for creating new forms by combining information scattered in other models—for example, to support system documentation generation and test cases, or to provide support for simulating global system behavior. Model composition (such as weaving and merging) is thus the third form of interaction facilitated by explicit definitions of relationships across elements in different DSMs.

These ideas can be applied at various phases of the development life-cycle, 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 DSMs to coordinate work can potentially have a beneficial impact on the running systems’ management. Different model kinds are currently used as runtime abstraction layers to support reasoning about the system or even adapting it.\textsuperscript{6} These model-based runtime environments can leverage explicitly defined relationships across DSMs to coordinate the manipulation of models at runtime.

C hallenging issues will need to be addressed to realize the above forms of language integration. Relationships among the languages will need to be defined explicitly 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 DSMs.\textsuperscript{7}

\section*{Acknowledgments}

This work is supported by the GEMOC initiative (http://gemoc.org), an open, international initiative that brings together a community to develop software language engineering breakthroughs supporting DSMs.

\section*{References}


Benoit Combemale is an associate professor at the University of Rennes, France, and a research scientist at Inria. His research interests include model-driven software engineering, software language engineering, domain-specific languages, and validation and 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, France. 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.

---

**Expert Online Courses — Just $49.00**

**Topics:**
- Project Management
- Software Security
- Embedded Systems
- and more.

**www.computer.org/online-courses**

---