

Globalizing Domain-Specific Languages

Edited by

Betty H. C. Cheng¹, Benoit Combemale², Robert B. France³,
Jean-Marc Jézéquel⁴, and Bernhard Rumpe⁵

- 1 Michigan State University – East Lansing, US, chengb@cse.msu.edu
- 2 University of Rennes, FR, benoit.combemale@irisa.fr
- 3 Colorado State University – Fort Collins, US, france@cs.colostate.edu
- 4 University of Rennes, FR, Jean-Marc.Jezequel@irisa.fr
- 5 RWTH Aachen, DE, Rumpe@se-rwth.de

Abstract

This report documents the program and the outcomes of the Dagstuhl Seminar 14412 “Globalizing Domain-Specific Languages” held in October 2014.

Complex, data-intensive, cyber-physical, cloud-based etc. systems need effective modeling techniques, preferably based on DSLs to describe aspects and views. Models written in heterogeneous languages however need to be semantically compatible and their supporting individual tools need to be interoperable. This workshop discusses possible and necessary forms of interoperation their benefits and drawbacks and in particular whether there is a general pattern on coordination, composition and interoperation possible. Main goal was to establish a research programme towards such techniques.

Seminar October 5–10, 2014 – <http://www.dagstuhl.de/14412>

1998 ACM Subject Classification I.6.5 [Simulation and Modeling]: Model Development – Modeling methodologies, D.3.0 [Programming Languages]: General – Standards, D.2.13 [Software Engineering]: Reusable Software – Domain engineering, Reuse models

Keywords and phrases Modelling, Domain Specific Language, Software, Coordination, Globalization, Heterogeneous Complex Systems, DSL, UML, Composition

Digital Object Identifier 10.4230/DagRep.4.10.32

Edited in cooperation with Katrin Hölldobler

1 Executive Summary


Betty H. C. Cheng

Benoit Combemale

Robert B. France

Jean-Marc Jézéquel

Bernhard Rumpe

License  Creative Commons BY 3.0 Unported license

© Betty H. C. Cheng, Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, and Bernhard Rumpe

Model Driven Engineering (MDE) aims to reduce the accidental complexity associated with developing complex software-intensive systems, through the development of technologies that enable developers to systematically create, evolve, analyse, and transform various forms of abstract system models.

Current MDE language workbenches, in both academia and industry, support the development of Domain-Specific Modeling Languages (DSMLs) that can be used to create models



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Globalizing Domain-Specific Languages, *Dagstuhl Reports*, Vol. 4, Issue 10, pp. 32–50

Editors: Betty H. C. Cheng, Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, and Bernhard Rumpe



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

that play pivotal roles in different development phases. Language workbenches such as EMF, Metaedit+ or MPS support the specification of the abstract syntax, concrete syntax and the static and dynamic semantics of a DSML. These workbenches aim to 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. 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 (incl., syntax and semantics) makes it very difficult for developers 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, that is, the use of multiple modeling languages to support coordinated development of diverse aspects of a system.

Discussions during the seminar will focus on how multiple heterogeneous modeling languages (or DSMLs) will need to be related to determine how different aspects of a system influence each other. We have identified three forms of relationships among DSMLs that can be used as a starting point for discussions: interoperability, collaboration, and composition. These forms of language integration will need to address challenging issues that arise from the heterogeneity of modeling languages. 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 is thus another topic that will be discussed in the seminar.

The goal of the seminar was to develop a research program that broadens the current DSML research focus beyond the development of independent DSMLs to one that provides support for globalized DSMLs. In the globalized DSMLs vision, integrated DSMLs provide the means for teams working on systems that span many specialized domains and concerns to determine how their work on a particular aspect influences work on other aspects.

Working Groups

In the seminar we started the following four working groups which are producing results during the workshop and compiling them into a State-Of-The-Art report afterwards:

- Group 1a** Motivating Use Cases for the Globalization of DSLs Definition of the main scenarios motivating the globalization of DSLs
- Group 1b** Conceptual Model of the Globalization of Domain-Specific Languages Definition of the common vocabulary and foundations of the globalization of DSLs
- Group 2** Globalized Domain Specific Language Engineering Challenges of the globalization of DSLs from the language designer point of view
- Group 3** Domain Globalization: Using Languages to Support Technical and Social Coordination Challenges of the globalization of DSLs from the language user point of view

2 Table of Contents

Executive Summary

| | |
|--|----|
| <i>Betty H. C. Cheng, Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, and Bernhard Rumpe</i> | 32 |
|--|----|

Overview of Talks


| | |
|---|----|
| Globalizing DSLs through Contextualized Modeling <i>Colin Atkinson</i> | 36 |
| Globalizing DSLs: overall consistency and large scale collaboration <i>Cedric Brun</i> | 37 |
| Globalization of Modeling Languages: A Formal Semantics Approach <i>Barrett Bryant</i> | 38 |
| Hybrid systems modeling and simulation: challenges and solutions <i>Benoit Caillaud</i> | 38 |
| Application-Driven Globalization of Modeling Languages <i>Betty H. C. Cheng</i> | 38 |
| Experience integrating DSLs and Formal Methods for Coordinating Vehicles <i>Siobhan Clarke</i> | 40 |
| Globalization of Domain Specific Modelling Languages <i>Tony Clark</i> | 40 |
| Language Engineering Workbench <i>Benoit Combemale</i> | 41 |
| Golden Models in Engineering and Formal Methods <i>Julien DeAntoni</i> | 41 |
| Globalizing Domain-Specific Languages: A few thoughts on the open-world <i>Thomas Degueule</i> | 42 |
| Social Translucence <i>Robert B. France</i> | 42 |
| The Need for Multilevel DSMLs <i>Ulrich Frank</i> | 43 |
| Towards Families of DSLs <i>Jean-Marc Jézéquel</i> | 43 |
| Globalization of modeling languages: definition and challenges <i>Gabor Karsai</i> | 43 |
| Globalizing Modeling Languages <i>Marjan Mernik</i> | 44 |
| Smart Emergency Response as an Application Use Case <i>Pieter J. Mosterman</i> | 44 |
| Domain-Specific Tooling Infrastructure <i>Oscar M. Nierstrasz</i> | 45 |
| Composition of Languages <i>Bernhard Rumpe</i> | 45 |

| | |
|--|----|
| Globalization of DSLs or All about Boxes and Lines | |
| <i>Martin Schindler</i> | 46 |
| A benchmark for globalizing domain-specific languages | |
| <i>Friedrich Steimann</i> | 46 |
| “Globalizing DSLs”: Defining coordination among modeling languages | |
| <i>Juha-Pekka Tolvanen</i> | 47 |
| A view On the Globalization of Modeling Languages | |
| <i>Antonio Vallecillo</i> | 47 |
| Globalizing Models with the MPS Language Workbench | |
| <i>Markus Völter</i> | 48 |
| DSL related research challenges | |
| <i>Mark van den Brand</i> | 49 |
| Participants | 50 |

3 Overview of Talks

3.1 Globalizing DSLs through Contextualized Modeling

Colin Atkinson (*Universität Mannheim, DE*)

License  Creative Commons BY 3.0 Unported license
© Colin Atkinson

As software systems increase in size and complexity, and are expected to cope with ever more quantities of information from ever more sources, there is an urgent and growing need for a more view-oriented approach to software engineering. Views allow stakeholders to see exactly the right information, at exactly the right time, in a way that best matches their capabilities and goals. Domain-specific languages are a key foundation for supporting views by allowing them to display their contents in a customized way, but the current generation of software language engineering technologies do not go far enough. In particular, they currently lack the ability to convey the precise relationship between the information shown in a view and the information it is a view of. They also focus on describing how model elements should be visualized but provide little or no support for describing how stakeholders should edit and interact with them.

The premise of this talk is that software language engineering technologies need to evolve to support an enhanced approach to modeling in which model content can be set in context relative to the underlying source from which it is derived – an approach we refer to as “contextualized modeling”. These technologies would then be more accurately characterized as “view engineering” technologies rather than “language engineering” technologies since they would support all aspects of view definition, including the context in which the content is to be interpreted and the mechanisms by which model elements are to be visualized and edited. Some of the key additional capabilities that the current generation of language engineering technologies need to support in order to become globalized, viewpoint engineering languages include:

Enriched Designation. The most important context information in a view is its model elements’ location in the three key hierarchies of the underlying information model – the classification hierarchy, the inheritance hierarchy and the containment (i.e. ownership) hierarchy. These are supported to various degrees in today’s language engineering technologies through a mix of explicit symbolism and location-defining designators (a.k.a. headers) in model elements. However, they are not supported in a uniform and consistent way, and are often severely limited in what they can express. In particular most contemporary language engineering technologies only allow one level of classification to be expressed at a time. Fully contextualized modeling requires a comprehensive, systematic and deep designation notation which allows a model element’s exact location in each hierarchy to be expressed in its designator.

Explicit Elision Symbolism. Since views almost always convey only a subset of the information contained in the underlying model, an important requirement in viewpoint engineering is to support the description of what things are not included in a view, as well as the description of what things are. This is a challenging task since it involves subtle interactions between explicit omission statements (e.g. “...” in UML generalization sets), explicit completeness statements (e.g. complete and disjoint in UML generalization sets) and background “world” assumptions (e.g. “open world” versus “closed world” assumption). Fully contextualized modeling therefore requires comprehensive and systematic support for elision, both in the form of explicit elision symbols and elided model element designators.

Explicit Derivation Symbolism. As well as omitting information from the underlying subject of a view it is possible to derive new information that the subject does not explicitly contain. Such derivation operations can be driven by the application of basic characterization relationships such as inheritance and classification (e.g. subsumption) or by more complex inference operations based on the principles of logic. In both cases, contextualized modeling must incorporate the ability to express what information in a view has been derived and what information has been explicitly asserted by a human modeller. This is important for resolving conflicts and signalling the weight that should be given to the information represented within views.

Language Symbiosis. Domain-specific representations of information have the advantage that they are optimized for particular classes of stakeholders or communities of experts, whereas general-purpose languages have the advantage that they are widely known and can represent information in quasi-standard ways. In order to enjoy both benefits simultaneously, contextualized models should be represented by highly flexible, symbiotic languages that allow different visualizations of model elements to be mixed and interchanged at will.

Viewpoint Environment Definition. A user's experience of a view is determined not only by the way in which its contents are displayed, but also by the way in which the user can interact with the model and, when it is editable, input information. This impacts all aspects of the environment in which the view is displayed, including the menu items, the pallets of predefined types and models elements and the range of operations that can be applied to the content (e.g. checking, printing, persisting etc.). The engineering of viewpoints therefore involves much more than just the engineering of languages it also involves the definition of the associated interaction experience.

In this talk the vision of contextualized modeling is introduced and the key ingredients needed to turn the current generation of software language engineering technologies into fully globalized viewpoint-engineering technologies explained.

3.2 Globalizing DSLs: overall consistency and large scale collaboration

Cedric Brun (Obeo – Nantes, FR)

License  Creative Commons BY 3.0 Unported license
© Cedric Brun


Domain specific languages have shown their efficiency in designing more precisely and easing the creation of tools. The system definition ends up being split by concern which mostly maps to the type of stakeholders involved, but in the end we need to have a view of the consistency of the system globally.

DSLs have relationships in between them, a DSL might expose some aspects through interfaces formalizing how others can use or extend it. This lack of formalization leads to shortcomings in regard to the technical integrations of those DSLs and this gets even more complex when collaboration gets involved.

As a provider of technologies enabling the use of DSLs at a large scale tackling these challenges has a huge potential, complexity could still be managed thanks to using languages which are domain specifics yet more stakeholders could be involved and the whole process would require less coordination.

3.3 Globalization of Modeling Languages: A Formal Semantics Approach


Barrett Bryant (University of North Texas, US)

License  Creative Commons BY 3.0 Unported license
© Barrett Bryant

The goals of this research are to 1) formalize the semantics of Domain-Specific Modeling Languages (DSMLs), 2) use semantic formalization for automatic generation of model-driven engineering software tools, and 3) use semantic formalization and tool generation to facilitate DSML composition. This will allow automating many tasks that are currently done ad hoc in a manual hand-crafted manner.

3.4 Hybrid systems modeling and simulation: challenges and solutions

Benoit Caillaud (INRIA Bretagne Atlantique – Rennes, FR)

License  Creative Commons BY 3.0 Unported license
© Benoit Caillaud

Hybrid systems combine continuous and discrete time dynamics, expressed in a single language, or as combination of several of several dedicated languages. Two key challenges regarding hybrid systems are:

- Integration of discrete and continuous time models at a semantics level. How can one co-simulate a system model combining models with radically different semantics: discrete time dynamical systems on one hand, and continuous time dynamical systems on the other hand. The discrete time dynamics is often expressed in a data flow or automata based language, while the continuous dynamics results from a system of ordinary or algebraic differential equations (resp. ODEs and DAEs). Several techniques can be used to address this problem, depending on the overall system architecture and the assumptions that can be made on the overall system behaviour. These techniques range from simple asymmetric co-simulation methods, where time is handled by the numeric differential equation solver, to more involved techniques, for example, waveform relaxation.
- A second challenge is compositionality and modular compilation of acausal continuous time models. They are often expressed using algebraic differential equations (DAEs), where the data flow orientation of an incomplete model may depend on its environment. Hence, generating simulation code for a component, without knowing its precise environment, is a difficult problem. The problem becomes even more severe when considering hybrid systems with DAEs, found for example in Modelica models. The main reason is that, not only the dataflow orientation may change dynamically, but the differentiation index may change, depending on the discrete state of a model.

3.5 Application-Driven Globalization of Modeling Languages

Betty H. C. Cheng (Michigan State University – East Lansing, US)

License  Creative Commons BY 3.0 Unported license
© Betty H. C. Cheng

Our society is now demanding software for engineered systems at levels of complexity that transcend historical precedents and the state of the art in software development concepts, methods and tools. Major challenges arise due to many factors: unprecedented functions

and complexity; significant uncertainty about requirements [1], designs and components; the inability due to scale to impose centralized control on system definition, development, deployment, operation and evolution; hard performance requirements; and continual scattered failure due to accidents and attacks. Such systems have recently been called Ultra-Large-Scale (ULS) Systems. New knowledge and methods are needed in both traditional and emerging areas of software research to enable successful development of ULS systems. Neither industry nor the academic community is well-positioned to develop this knowledge by itself.

Software and its corresponding models are key enablers of ULS systems—the source of new behaviors as well as the flexibility to adapt under uncertainty and change — but also, due to shortcomings in our knowledge, the key impediment to developing such systems. Difficulties are being experienced today in a wide range of application areas, including defense, transportation systems, financial systems, medical-based systems, etc. Demands for function, safety, flexibility, responsiveness, availability, security, privacy and integration far outstrip current knowledge and engineering capabilities.

A specific subset of ULS systems are high-assurance systems (HAS), those systems designed to tolerate component failures, and even direct attacks, in order to continue operation and preserve system integrity. The study of high-assurance systems is inherently multidisciplinary, requiring collaboration of educators and researchers from a wide variety of scientific disciplines and application domains [5]. Moreover, academic researchers need to work closely with industrial partners to ensure that solutions address the scale and complexity found in the real world, and to facilitate the transition of research results into practice.

Designing and implementing high-assurance systems typically requires the integration of several enabling computing technologies (e.g., model-driven software engineering [2, 9], sensor networks, autonomic computing [7]) within a particular application domain (e.g., transportation systems, telecommunication networks, electronic medical records, digital supply chain). In addition, human factors play a critical role, since most realistic applications involve a blend of humans and machines. Each of these orthogonal dimensions may involve a wide range of modeling languages, all of which need to be integrated in order to provide the overall functionality with the required assurance [4, 6]. Model integration challenges exist at the structural, semantic, and application domain levels [3]. Beyond the technical challenges, it is important for the research to address the usability of the model-based systems for a wide range of stakeholders, from domain experts to policy analysts and other types of decision-makers [8].

References

- 1 Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H.C. Cheng and Jean-Michel Bruehl, RELAX: a language to address uncertainty in self-adaptive systems requirement, In Requirements Engineering, Vol. 15, Issue 2, pages 177–196, Springer London, 2010
- 2 Heather Goldsby and Betty H. C. Cheng, Automatically Discovering Properties That Specify the Latent Behavior of UML Models, MoDELS (1), pages 316–330, 2010
- 3 B. H. C. Cheng, A. Ramirez, and P. K. McKinley, Harnessing evolutionary computation to enable dynamically adaptive systems to manage uncertainty, Combining Modelling and Search-Based Software Engineering (CMSBSE), Keynote for 2013 1st International Workshop on, 2013
- 4 Andres J. Ramirez, Betty H. C. Cheng, Nelly Bencomo, and Pete Sawyer, Relaxing Claims: Coping with Uncertainty While Evaluating Assumptions at Run Time., MoDELS, 2012
- 5 Robert B. France, James M. Bieman, Sai Pradeep Mandalaparty, Betty H. C. Cheng, and Adam C. Jensen, Repository for Model Driven Development (ReMoDD)., ICSE, 2012

- 6 Andres J. Ramirez, Erik M. Fredericks, Adam C. Jensen and Betty H. C. Cheng, Automatically RELAXing a Goal Model to Cope with Uncertainty., SSBSE, 2012
- 7 Philip K. McKinley, Betty H. C. Cheng, Andres J. Ramirez, and Adam C. Jensen, Applying evolutionary computation to mitigate uncertainty in dynamically-adaptive, high-assurance middleware., In J. Internet Services and Applications, pages 51–58, 2012
- 8 Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty H. C. Cheng, Philippe Collet, Benoit Combemale, Robert B. France, Rogardt Heldal, James H. Hill, Jörg Kienzle, Matthias Schöttle, Friedrich Steimann, Dave R. Stikkolorum, and Jon Whittle, The Relevance of Model-Driven Engineering Thirty Years from Now, Model-Driven Engineering Languages and Systems – 17th International Conference, MODELS 2014, Valencia, Spain, September 28 – October 3, 2014. Proceedings pages 183–200, 2014
- 9 Adam C. Jensen, Betty H. C. Cheng, Heather Goldsby, and Edward C. Nelson, A Toolchain for the Detection of Structural and Behavioral Latent System Properties, Model Driven Engineering Languages and Systems, 14th International Conference, MODELS 2011, Wellington, New Zealand, October 16–21, 2011. Proceedings, pages 683–698, 2011
- 10 Peter H. Feiler, Kevin Sullivan, Kurt C. Wallnau, Richard P. Gabriel, John B. Goodenough, Richard C. Linger, Thomas A. Longstaff, Rick Kazman, Mark H. Klein, Linda M. Northrop, and Douglas Schmidt, Ultra-Large-Scale Systems: The Software Challenge of the Future, Software Engineering Institute, Pittsburgh, 2006

3.6 Experience integrating DSLs and Formal Methods for Coordinating Vehicles

Siobhan Clarke (Trinity College Dublin, IE)

License © Creative Commons BY 3.0 Unported license
© Siobhan Clarke

Our overall research goal was to define a Domain Specific Language for applications that would take advantage of a middleware that caters for vehicles coordinating in real-time. We wanted to allow an application developer build an application (e.g., an intersection collision avoidance system or a managed highway) using our vehicle coordination protocol where we could then verify the safety of the application. It was intended that the safety checking would be achieved by integrating a formal method language. However, in our experience, this was challenging primarily because our requirements were not matched by any potentially related formal language. This is a general challenge for globalising languages, as their integration may still not address all requirements in the “new”, global context.

3.7 Globalization of Domain Specific Modelling Languages

Tony Clark (Middlesex University, GB)

License © Creative Commons BY 3.0 Unported license
© Tony Clark

The globalisation of domain specific modelling languages can be achieved through an understanding of language composition. This involves the specification, implementation and deployment of multiple languages and their associated artefacts including syntax, semantics, documentation, methods etc. In order to understand and achieve globalisation it is fruitful to

take a component-based view of languages where a language component defines a collection of interfaces that expose those parts of a language specification or implementation that are necessary to integrate it with other languages. At the time of writing it is not clear how to express or use such language components and this is an important research area that needs to be addressed. A useful approach may be to develop a common meta-language that can be used to articulate languages as components. Furthermore, a common framework based on such meta-concepts may be useful for globalisation where existing languages and their models can be wrapped in order to conform to the requirements of the framework.

3.8 Language Engineering Workbench

Benoit Combemale (University of Rennes, FR)

License © Creative Commons BY 3.0 Unported license
© Benoit Combemale

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. I present 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.

3.9 Golden Models in Engineering and Formal Methods

Julien DeAntoni (INRIA Sophia Antipolis – Méditerranée, FR)

License © Creative Commons BY 3.0 Unported license
© Julien DeAntoni

In many disciplines and for several years, models have been used to abstract the system under study. Depending on the model and its purpose, it brings very different properties ranging from re-usability to analyzability. Consequently, there is usually no single golden model of a system, but there are good models of a system where different dedicated models abstract the system for different purpose. Real time embedded systems are interesting candidates for modeling for two main reasons. On the one hand many properties like timing performance, time-to-market or safety are early and mandatory requirements to be satisfied at all steps of the design. On the other hand, the deployment of such system can target various heterogeneous platforms and this deployment strongly impacts the previously stated requirements.

For some years in the AOSTE team, I am studying how engineering models (e.g., based on UML or Ecore) and formal models (e.g., automaton or marked graph) can take benefits one from the other in order to improve the modeling of real time embedded systems. More precisely my current research focuses on two related topics. First, to enable reasoning on engineering models I put efforts to provide formal models that describe the behavioral

semantics of a language. This means providing an adequate meta-language to specify an explicit entity that represents the behavioral semantics and can thus be manipulated. Second, because different models are used to specify correctly a single system, it is important to understand the interaction among these different models and more precisely to understand how the explicit semantics can be used to provide a behavioral interface of the language amenable to reasoning, composition and generative techniques.

3.10 Globalizing Domain-Specific Languages: A few thoughts on the open-world

Thomas Degueule (University of Rennes, FR)

License  Creative Commons BY 3.0 Unported license
© Thomas Degueule


In the past few years, the development of model-driven engineering and associated tools has strengthened the trend supporting domain-specific language development by enabling domain experts to design their own DSL without requiring strong skills in languages or compilers construction. This has led to a widespread use of DSL in many areas, including software and system engineering.

While the facilities available for engineering isolated DSLs are getting closer to maturity, there is little support for comprehension on the interactions between several languages that are used within a single system or software. In an open world, where DSLs are designed independently by (possibly small) groups of developers, the need for “globalization” is increasingly felt.

To support the globalization of DSLs, we advocate the design of precise language interfaces. Language interfaces allow to abstract some of the complexity carried in the implementation of languages, by exposing meaningful information concerning a given aspect of a language and for a specific purpose (e.g. composition or coordination) in an appropriate formalism. We strongly believe that such interfaces will ease the cognitive and technical effort necessary for lifting the limitations of current approaches.

3.11 Social Translucence

Robert B. France (Colorado State University, US)

License  Creative Commons BY 3.0 Unported license
© Robert B. France

We are interested in designing systems that support communication and collaboration among large groups of people over computer networks. We begin by asking what properties of the physical world support graceful human-human communication in face-to-face situations, and argue that it is possible to design digital systems that support coherent behavior by making participants and their activities visible to one another. We call such systems “socially translucent systems” and suggest that they have three characteristics-visibility, awareness, and accountability-which enable people to draw upon their social experience and expertise to structure their interactions with one another. To motivate and focus our ideas we develop a vision of knowledge communities, conversationally based systems that support the creation, management and reuse of knowledge in a social context. We describe our experience in

designing and deploying one layer of functionality for knowledge communities, embodied in a working system called “Babble,” and discuss research issues raised by a socially translucent approach to design.

3.12 The Need for Multilevel DSMLs

Ulrich Frank (Universität Duisburg-Essen, DE)

License © Creative Commons BY 3.0 Unported license
© Ulrich Frank

The construction of DSMLs is facing an essential challenge: On the one hand, it makes sense to develop very specific languages that serve the particular needs of one organization only, because then we can expect it to effectively promote model quality and modelling productivity. On the other hand, global DSMLs promote economies of scale by enabling a wider range of reuse and integration. A multilevel hierarchy of DSML, which is inspired by the actual use of technical languages, enables to successfully address this challenge. On the top level, which corresponds to the concepts introduced in textbooks, global DSML would capture commonalities of a range of more specific DSML. Depending on the size and diversity of the domains to be covered, the number of required levels may vary. Realizing respective hierarchies of DSMLs convincingly demands for adequate modelling concepts and – more challenging – for multilevel (meta) programming languages. In addition to linguistic and technical considerations, there is need to build effective incentives for contributing to the development of DSML hierarchies.

3.13 Towards Families of DSLs

Jean-Marc Jézéquel (University of Rennes, FR)

License © Creative Commons BY 3.0 Unported license
© Jean-Marc Jézéquel

The engineering of systems involves many different stakeholders, each with their own domain of expertise. Hence more and more organizations are adopting Domain Specific Languages (DSLs) to allow domain experts to express solutions directly in terms of relevant domain concepts. This new trend raises new challenges about designing DSLs, evolving a set of DSLs and coordinating the use of multiple DSLs. In this talk we explore various dimensions of these challenges, and outline a possible research roadmap for addressing them. We detail one of these challenges, which is the safe reuse of model transformations.

3.14 Globalization of modeling languages: definition and challenges

Gabor Karsai (Vanderbilt University, US)

License © Creative Commons BY 3.0 Unported license
© Gabor Karsai

The problem of globalization of modeling languages can be viewed as a system integration problem for modeling languages and tools. Integration of modeling languages deals with the

composition of heterogeneous modeling languages, so that the composite has a clear semantics – i. e. the composed models have a meaning, possibly beyond that of the component models. Integration of modeling tools deals with the orchestration of complex tool use cases, where heterogeneous tools work together to execute a complex tasks, e. g. co-simulation, formal verification across multiple models, etc. There three fundamental challenges associated with these topics: (1) determining the relevant properties of the modeling languages and tools that need to be captured in some formalism (e. g. metamodels), (2) determining the composition operators for metamodels that facilitate integration, and (3) assigning operational semantics to the composition of the modeling languages and tools. The first problem deals with developing the structural semantics of a model (or tool) integration language, the second deals with the specific composition operators available in the language, and the third addresses the operational semantics of the model integration language.

3.15 Globalizing Modeling Languages

Marjan Mernik (University of Maribor, SI)

License  Creative Commons BY 3.0 Unported license
© Marjan Mernik

The globalization of modeling languages is a newly emerged term defining a situation where multiple heterogeneous modeling languages are used for describing different aspects of a complex system. These aspects may or may not (partially) overlap. Moreover, there is an urgent need that these heterogeneous modeling languages interact to each other. Therefore, the syntax and semantics of modeling languages must be precisely defined. One form of interaction known from programming languages is language composability, which is a property of language specifications rather than a property of a language itself. They are several known forms of programming language composition: language extension (which subsumes also language restriction), language unification, self-extension, and extension composition. However this classification needs to be extended and adopted for globalization of modeling languages.

3.16 Smart Emergency Response as an Application Use Case

Pieter J. Mosterman (The MathWorks Inc. – Natick, US)

License  Creative Commons BY 3.0 Unported license
© Pieter J. Mosterman

Main reference P. J. Mosterman, D. E. Sanabria, E. Bilgin, K. Zhang, J. Zander, “A Heterogeneous Fleet of Vehicles for Automated Humanitarian Missions,” *Computing in Science & Engineering*, 16(3):90–95, IEEE, 2014.

URL <http://dx.doi.org/10.1109/MCSE.2014.58>

This presentation explores the cyber-physical systems paradigm in humanitarian missions, in particular in emergency response to natural disasters such as earthquakes. A smart emergency response system is presented to illustrate opportunities and challenges in cyber-physical system applications.

3.17 Domain-Specific Tooling Infrastructure

Oscar M. Nierstrasz (Universität Bern, CH)

License © Creative Commons BY 3.0 Unported license
© Oscar M. Nierstrasz

Globalization of modeling languages is about enabling the shared use of such languages for domains of common interest. We are working on better tools to rapidly extract models from source code, techniques to adapt development tools to specific domains, and generic DSLs that can be adapted to a variety of different analysis tools. The key challenges we identify are: (1) better environments and workbenches to support DSL engineering, (2) techniques to rapidly develop or adapt tools to specific domains, and (3) techniques to support the integration and coordination of multiple DSLs into software systems.

3.18 Composition of Languages

Bernhard Rumpe (RWTH Aachen, DE)

License © Creative Commons BY 3.0 Unported license
© Bernhard Rumpe

DSLs need to be modular, reusable and composable. Otherwise we will get a plethora of incompatible and complex languages that are expensive to develop, maintain and evolve. We therefore take the view of component based design, by adapting the mechanisms of modular components and their composition to languages. This means that a “language component” should have a crisp boundary, that encapsulates internals of the language and makes the language accessible only through its interface.

The notion of “language interface” is not easy to assess. Languages certainly exhibit parts of their abstract syntax, which is the main carrier for a language, for composing sub-languages together into more complex ones. However the concrete syntax needs also to be composed if applicable. Furthermore for a precise understanding of the emerging composed language, the semantics, which we like to be given as denotational semantics with semantics domain and semantics mapping needs to be composed as well. This is tricky, as semantics domains can be completely disjoint, overlapping or even identical, but encoded in different carriers. Last but not least, additional information about the symbols defined in a sub-model and exported to another part of a model, which is defined in another sub-language, must be transferable through the language interface.

While we do have a relatively good understanding of concrete compositions of computer languages, I useful and general theory for language composition, adaptation and thus their reuse is still in infancy. We discuss some results and more problems on the concepts of language engineering, including syntax, semantics and tooling that we made while our development of and with the MontiCore language workbench.

3.19 Globalization of DSLs or All about Boxes and Lines

Martin Schindler (MaibornWolff GmbH – München, DE)

License © Creative Commons BY 3.0 Unported license
© Martin Schindler

Main reference M. Schindler, “Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P,” Dissertation, RWTH Aachen, Aachener Informatik-Berichte – Software-Engineering, Vol. 11, 370 pages, Shaker Verlag, 2012.

URL <http://dx.doi.org/10.2370/9783844008647>

Globalization of DSLs means composing languages including all involved artifacts. For this a clear definition of a language component and its interface is needed. A language component has to encapsulate all parts of a language like concrete and abstract syntax, validation (e. g., context conditions), and transformations (e. g., code generators). On the other hand, a language interface only includes the necessary information needed for composing a language with other languages.

In [1] it is shown how the UML/P, which is a subset of the UML including Java as an action language, can be developed in such a component based way. All languages of the UML/P (Classdiagrams, Statecharts, Objectdiagrams, Sequencediagrams, OCL, a language for testcases and Java) were developed completely independently of each other including a definition what is required from or provided for other languages. These required and provided interfaces were finally used to compose the UML/P keeping the language components unchanged. In this way the languages of the UML/P can be easily replaced by or reused with other languages.

The UML/P is just an example of language composition. However, at the end the composition of different DSLs should be just about connecting provided and required interfaces of the involved languages and should be as easy as drawing boxes and lines.

References

- 1 Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P (A Tooling Infrastructure for the Agile Development with the UML/P) Martin Schindler Shaker Verlag, ISBN 978-3-8440-0864-7. Aachener Informatik-Berichte, Software Engineering Band 11. 2012. <http://dx.doi.org/10.2370/9783844008647>

3.20 A benchmark for globalizing domain-specific languages

Friedrich Steimann (Fernuniversität in Hagen, DE)

License © Creative Commons BY 3.0 Unported license
© Friedrich Steimann

The globalization of domain-specific languages (DSLs) has been achieved if we are able to refactor across DSLs. This is so because refactoring, i. e., the behaviour-preserving change of a system, requires full anticipation of the effect of every change on the behaviour of the system, and hence the joint semantics of the DSLs that are being used.

While being able to refactor across DSLs may seem like a high hurdle, we have found that, once behaviour-preserving changes are mastered in each participating language, cross-language refactoring is little more than identifying the (model or program) elements through which interaction occurs, and translating the changeable properties of these elements from one language to another (so that changes can be propagated).

It seems that the globalization of DSLs requires the same: identification of the elements through which interaction occurs, and full awareness of what changes of these elements translate to in each language. This can be greatly simplified by using a language interoperability infrastructure on which all participating languages are built; in absence of such an infrastructure, it will be much harder (and likely requires effort quadratic in the number of participating languages).

3.21 “Globalizing DSLs”: Defining coordination among modeling languages

Juha-Pekka Tolvanen (MetaCase – Jyväskylä, FI)

License © Creative Commons BY 3.0 Unported license
© Juha-Pekka Tolvanen

Domain-Specific Modeling has become increasingly popular in the past decade. These languages allow raising the level of abstraction away from the solution domain (code) to the problem domain with obvious benefits such as improved development productivity and product quality. While typically the domain-specific modeling languages are built for a narrow area within a company the next obvious step is to “globalize” languages so that coordinated use of domain-specific languages becomes possible. We identify some key challenges for research in three contexts: organization, language and technology. In the organizational context often already a single DSL may change organizational tasks, roles and structures. How multiple coordinated ones while influence to organizations and to development processes. In the language context, the coordination must be specified at the level of language specification but it is not clear how currently used metamodeling languages allow to do that. For example OMG’s MOF does not even identify “language” so it is questionable how it can then integrate a number of them? Finally, and within the technology context, it is not realistic to expect that all languages can be coordinated within a single tool so what kind of tool integration approach would work among a set of tools?

3.22 A view On the Globalization of Modeling Languages

Antonio Vallecillo (University of Malaga, ES)

License © Creative Commons BY 3.0 Unported license
© Antonio Vallecillo

DSLs are proliferating, as system modelers and designers are finding them useful for their purposes, and as tool support starts to be readily available for them (editors, validators, code generators, etc.). The previous “one language fits all” approach (e.g. Java, UML) has given path to “one language for each purpose”, and this is where the need to make public, coordinate and combine languages (i.e., globalize them) has risen.

“Globalizing modeling languages” was defined in the original GEMOC paper as “The use of multiple languages to support coordinated development of diverse systems aspects”. However, I only agree in part with such definition. First, I see it is more adequate for defining what Multi-Viewpoint Modeling is/should be about: “The combination of multiple languages to support coordinated specification, analysis and development of diverse systems aspects”

Thus, in my view, “Globalizing a Modeling Language” means “Making a Modeling Language amenable for integration into a (standard) Multi-Viewpoint Modeling environment.”

In this context, it is important to note that (a) Globalized MLs need to be combinable and integrable (b) Interfaces at different levels should be defined, and (c) Standardization should play a key role here.

In our group we have been working in this area, in the context of the Reference Model for Open Distributed Processing (RM-ODP), an ISO & ITU-T international standard that provides a mature framework for the specification of large complex systems, using viewpoints. RM-ODP defines five viewpoints and their associated Viewpoint Languages (VPL), as well as explicit correspondences between the VPLs. This is an example of the coordination and integration of separate languages, focusing on disparate concerns, and using correspondents to relate them.

In addition, we have been working on the combination of DSMLs, studying the problems, issues and challenges involved in this area.

The three major challenges that we see in the globalization of modeling languages are the following. First, there is the need for defining mechanisms, process and tools for the Combination/Integration/Unification of languages (which needs establishing correspondences between them, at all levels: Abstract Syntax, Concrete Syntax and Semantics); and needs to deal with heterogeneous (and not always combinable) semantics. Second, correspondences between metamodels, and between models, needs to be specified in an efficient, correct, usable and maintainable manner, and using different approaches (depending on the use we want to make of them). The third challenge that we want to highlight is about being able to reason about the information expressed across the different models, so that properties of the overall system (including emergent properties) can be inferred, proved or denied.

Tool support is essential in this context for achieving these goals, given the complexity of the domain and of the systems being specified. Without tools any proposed solution will be useless.

3.23 Globalizing Models with the MPS Language Workbench

Markus Völter (Völter Ingenieurbüro, DE)

License  Creative Commons BY 3.0 Unported license
© Markus Völter

Developing software often requires using a number of tools and languages. In embedded software, for example, it is common to use C and its IDE, Matlab/Simulink, a number of custom XML files, a requirements management tool such as DOORS and possibly a UML tool and a variant management tool. The integration of such a zoo of tools is often a major source of (accidental) complexity in development projects. The GEMOC initiative addresses this challenge.

Back in the ‘good old days’ when everything was text files and command line executables running on the unix shell. This approach had two important properties: the infrastructure was extremely generic (unix shell, pipes, text editors) and the actual contents were easily extensible and composable (new text file formats/languages and new command line tools); a productive environment for a given project or domain could easily be built from the generic infrastructure plus a few custom extensions.

Language Workbenches can be used to create (domain-specific) development environments


that results in many of the same advantages that we all valued in the unix shell-world. A language workbench is an extremely generic infrastructure that is easily extensible with new languages. It is easy to create domain-specific development tools that can address different concern of the system with suitable abstractions, but are nonetheless very well integrated in terms of syntax, semantics and tooling.

JetBrains MPS is such a language workbench; over the last few years we have used it to build mbeddr, an open source environment optimized for embedded software development. It consists of a set of 50+ C extensions as well as languages for requirements and documentation. It also directly integrates formal verification tools. Connecting to artifacts outside of mbeddr is possible via external references.

Language Workbenches, MPS and systems like mbeddr can be an important contribution for managing the overall complexity of Globalized DSLs.

3.24 DSL related research challenges

Mark van den Brand (TU Eindhoven, NL)

License  Creative Commons BY 3.0 Unported license
© Mark van den Brand

Model Driven Software Engineering is extremely popular in the High Tech Industry. They are facing two challenges. The first challenge is to increase the quality of the code, an ongoing quest. The second challenge is to get grip on the rapidly increasing amount of software. The high tech companies are exploring all options that model driven software engineering offers. Domain specific languages (DSLs) is one of the explored routes. Some of these companies have a whole range of DSLs. So far, these DSLs are developed in house, using different technologies, although that EMF and related technologies is the most important platform being used. Although that the high tech industry is a software intensive industry they do consider themselves not a software industry. They are DSL users and not DSL developers, or formulated differently they are tool users and not tool developers. Some of the high tech companies outsource the DSL and supporting tooling to suppliers.

The DSLs are used to describe different aspects of the high tech systems that are being produced. This means equivalent concepts used in the different DSLs have to be the same over these DSLs. One way of doing this is have a common semantic framework. In close cooperation with ASML we are developing a semantic framework (semantics based language workbench) based on Event-B. This semantic framework is used to give a formal for one of the DSLs used within ASML. The goal of the project is to use this framework to develop new DSLs in the future, in order to ensure the consistency of semantic concepts. The identification of common semantic concepts and capturing them in a semantics based workbench is the first challenge. Related challenges are the correctness of model transformations and modularity of meta models. Each of these challenges is related to the globalization of domain specific languages.

Participants

- Colin Atkinson
Universität Mannheim, DE
- Cedric Brun
Obeo – Nantes, FR
- Barrett Bryant
University of North Texas, US
- Benoit Caillaud
INRIA Bretagne Atlantique –
Rennes, FR
- Betty H. C. Cheng
Michigan State University –
East Lansing, US
- Tony Clark
Middlesex University, GB
- Siobhán Clarke
Trinity College – Dublin, IE
- Benoit Combemale
University of Rennes, FR
- Julien Deantoni
INRIA Sophia Antipolis –
Méditerranée, FR
- Thomas Degueule
University of Rennes, FR
- Robert B. France
Colorado State University, US
- Ulrich Frank
Universität Duisburg-Essen, DE
- Jean-Marc Jézéquel
University of Rennes, FR
- Gabor Karsai
Vanderbilt University, US
- Ralf Lämmel
Universität Koblenz-Landau, DE
- Marjan Mernik
University of Maribor, SI
- Pieter J. Mosterman
The MathWorks Inc. –
Natick, US
- Oscar M. Nierstrasz
Universität Bern, CH
- Bernhard Rumpe
RWTH Aachen, DE
- Martin Schindler
MaibornWolff GmbH –
München, DE
- Friedrich Steimann
Fernuniversität in Hagen, DE
- Eugene Syriani
University of Alabama, US
- Janos Sztipanovits
Vanderbilt University, US
- Juha-Pekka Tolvanen
MetaCase – Jyväskylä, FI
- Antonio Vallecillo
University of Malaga, ES
- Mark van den Brand
TU Eindhoven, NL
- Markus Völter
Völter Ingenieurbüro, DE

