08331 Abstracts Collection
Perspectives Workshop: Model Engineering of Complex Systems (MECS)
— Dagstuhl Seminar —

Uwe Aßmann[1], Jean Bezivin[2], Richard Paige[3], Bernhard Rumpe[4] and Douglas C. Schmidt[5]

[1] TU Dresden, D
[2] INRIA - LINA Nantes, F
jean.bezivin@inria.fr
[3] University of York, GB
paige@cs.york.ac.uk
[4] TU Braunschweig, D
B.Rumpe@sse-tubs.de
[5] Vanderbilt University, USA

Abstract. From 10.08. to 13.08.2008, the Dagstuhl Seminar 08331 "Perspectives Workshop: Model Engineering of Complex Systems (MECS)" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Conceptual modeling, model, modeling language, modeling method, model quality, modeling proccess, model cost

08331 Manifesto – Model Engineering for Complex Systems

Complex systems are hard to define [1]. Nevertheless they are more and more frequently encountered. Examples include a worldwide airline traffic management system, a global telecommunication or energy infrastructure or even the whole legacy portfolio accumulated for more than thirty years in a large insurance company. There are currently few engineering methods and tools to deal with them in practice. The purpose of this Dagstuhl Perspectives Workshop on Model Engineering for Complex Systems was to study the applicability of Model Driven Engineering (MDE) to the development and management of complex systems.

Keywords: Conceptual modeling, model, modeling language, modeling method, model quality, modeling proccess, model cost

Joint work of: Aßmann, Uwe; Bézivin, Jean; Paige, Richard F.; Rumpe, Bernhard; Schmidt, Douglas C.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2008/1603

Modeling Heterogeneous Real-time Components in BIP

Ananda Basu (VERIMAG - Gières, F)

We present a methodology for modeling heterogeneous real-time components.

Components are obtained as the superposition of three layers : Behavior, specified as a set of transitions; Interactions between transitions of the behavior; Priorities, used to choose amongst possible interactions. A parameterized binary composition operator is used to compose components layer by layer. We present the BIP language for the description and composition of layered components as well as associated tools for executing and analyzing components on a dedicated platform. The language provides a powerful mechanism for structuring interactions involving rendezvous and broadcast.

We show that synchronous and timed systems are particular classes of components. Finally, we provide examples showing the utility of the BIP framework in heterogeneous component modeling.

Keywords: Component based construction, Heterogeneous systems, Modeling and simulation, Code generation

Joint work of: Basu, Ananda; Bozga, Marius; Sifakis, Joseph

Full Paper:
http://www-verimag.imag.fr/∼async/bipPublis.php

See also: A. Basu, M. Bozga, J. Sifakis : Modeling Heterogeneous Real-time Components in BIP. In SEFM, pp. 3-12 (2006)

Model Engineering of Complex Systems (MECS) : runtime abstractions and runtime models

Nelly Bencomo (Lancaster University, UK)

I would like to do a short talk about runtime abstractions and runtime models for the development and operation of Complex Systems.

Keywords: Runtime models, runtime abstractions

Full Paper:
http://www.comp.lancs.ac.uk/∼bencomo/MRT

Model Driven Management of Complex Systems: Implementing the Macroscope's vision

Jean Bezivin (INRIA, F)

Several years ago, first generation model driven engineering (MDE) tools focused on generating code from high-level platform-independent abstract descriptions. Since then, the target scope of MDE has much broadened and now addresses for example testing, verification, measurement, tool interoperability, software evolution, and many more hard issues in software engineering. In this paper we study the applicability of MDE to another difficult problem: the management of complex systems. We show how the basic properties of MDE may be of significant help in this context and we characterize and extend MDE by the concept of a "megamodel", i.e. a model which elements may themselves be models. We sketch the basic characteristics of a tool for handling megamodels and we apply it to the example of the Eclipse.org ecosystem, chosen here as a representative illustration of a complex system. The paper finally discusses how the proposed original approach and tools may impact the construction and maintenance of computer based complex systems.

This document is based on a paper presented at ECBS'2008.

Keywords: MDE; Complex systems; Macroscope

Joint work of: Barbero, Mikaël; Bézivin, Jean

Model-Driven System Administration – The TUNe Experience and Perspectives

Benoit Combemale (University of Toulouse, F)

Autonomic computing is recognized as one of the most promizing solutions to address the increasingly complex task of distributed environments' administration. In this context, many projects relied on software components and architectures to provide autonomic management frameworks.

We designed such a component-based autonomic management framework, but observed that the interfaces of a component model are too low-level and difficult to use. Therefore, we introduced UML diagrams for the modeling of deployment and management policies. However, we had to adapt/twist the UML semantics in order to meet our requirements, which led us to define DSMLs.

In this talk, we present our experience in designing the Tune system and its support for management policy specification, relying on UML diagrams and on DSMLs. We analyse these two approaches, pinpointing the benefits of DSMLs over UML.

Keywords: Autonomic Computing, System Administration, DSML, Models at runtime

Joint work of: Combemale, Benoit; Hagimont, Daniel

Full Paper:
http://www.modelsconference.org/

Pattern-Based Model-to-Model Transformation

Juan De Lara (Univ. Autónoma de Madrid, ES)

We present a new, high-level approach for the specification of model-to-model transformations based on declarative patterns.

These are (atomic or composite) constraints on triple graphs declaring the allowed or forbidden relationships between source and target models. In this way, a transformation is defined by specifying a set of triple graph constraints that should be satisfied by the result of the transformation.

The description of the transformation is then compiled into lower-level operational mechanisms to perform forward or backward transformations, as well as to establish mappings between two existent models. In this paper we study one of such mechanisms based on the generation of operational triple graph grammar rules. Moreover, we exploit deduction techniques at the specification level to generate more specialized constraints (preserving the specification semantics) reflecting pattern dependencies, from which additional rules can be derived.

Keywords: Model Transformation; Graph Transformation; Graph Constraints; Logics; Triple Graph Grammars

Joint work of: De Lara, Juan; Guerra, Esther

Full Paper:
http://arxiv.org/abs/0805.4745v1

Software Language Engineering and Software Linguistics for Ultra-Large Scale Systems

Jean-Marie Favre (LSR - IMAG, F)

Ubiquitous computing, Ultra-Large-Scale Systems and Systems of Systems lead mankind to cope with an unprecedented level of complexity.

What is clear is that the design and implementation of such heterogeneous systems is now far beyond the capacity of a single human brain. This means that from now on, we will have to cope, not only with the complexity of systems, but also with the complexity of the human structures responsible for building and evolving those systems. In other words, complex systems implies complex arrangements of stakeholders, skills, processes and viewpoints.

Therefore, dealing with complex systems means dealing not only with complex graph of communicating (sub)systems, but also with graph of communicating stakeholders. Communication is possible only when suitable languages are

selected. Software languages are those languages used in the context of software intensive systems either to support

1. system/system communication,
2. system/stakeholder communication,
3. stakeholder/stakeholder communication.

We claim that Engineering Complex Software Systems implies considering Software Languages as first-class entities and that Software Language Engineering (SLE) is of paramount importance when dealing with Ultra-Large-Scale systems.

In fact SLE submsumes many different approaches that so far have been studied by separated communities and theirfore largely disconnected.

This includes for instance

- metamodel engineering,
- ontology engineering,
- viewpoint engineering,
- schema engineering,
- grammarware engineering,
- etc.

A brief map of Software Language Engineering challenges is given.

Since "Engineering" is the application of "Scientific Knowledge" to practical problem, the science part should not be neglected. We claim that even more important is to study Software Linguistics, the scientific study of Software Languages over space, time and projects.

We bet that Model Engineering of Ultra-Large-Scale Systems cannot be realisatically acheived without further research in Software Language Engineering and Software Linguistics. This could be one of the important challenges in the future of Informatics. Software Languages could appear to be key.

Keywords: Software Langage Engineering, Software Linguistics, Model Driven Engineering, MDE, model, metamodel, megamodel, language

CacOphoNy: Metamodel-Driven Software Architecture Reconstruction (Background Paper from WCRE 2004)

Jean-Marie Favre (LSR - IMAG, F)

Two fields has attempted to manage the complexing of complex systems :

1. Software Architecture
2. Model Driven Engineering

Each fields have its own standards (1) IEEE 1471 standards defines Software Architecture concepts :

- stakeholders,

- view points
- views

(2) MDA approaches are based on

- models
- metamodels
- languages
- transformations

This paper presents how IEEE 1471 and MDE can be merged to deal with the complexity of very large software product.

Full Paper:
http://megaplanet.org/jean-marie-favre/papers/CaCOphONYMetaModelDrivenSoftwareArchitectureReconstruction.pdf

See also: Working Conference on Reverse Engineering, WCRE 2004

On Horizontal and Vertical Relationships between Models

Martin Gogolla (Universität Bremen, D)

Detecting, modeling and managing relationships between models are central tasks within model-driven engineering. By taking a naive view on model development, we distinguish in a vertical dimension between domain-specific models, core models, and executable models. An example for a vertical relationship is the refinement relationship beween a core model and an executable model. Relationships may also refer to a horizontal dimension. For example, two UML/OCL core models may be equivalent in the sense that the same system states and the same state transitions are captured, but the invariants from the first model are encoded as preconditions in the second model.

Challenges of Model-Driven Evolution of Legacy Systems

Jeff Gray (University of Alabama at Birmingham, US)

As enterprises inevitably change, the IT systems that support them must adapt to those changes. There are literally several hundred billion lines of legacy code in production use. Hence, there is a need for well-defined, robust enterprise architectures that can be rapidly adapted to suit their evolving environment and meet changing requirements. To apply model-based techniques to such systems, it is beneficial to have an approach that would work bottom-up as well as top down. In other words, we have to deal not only with generation towards the platforms of the present and the future, but also with recovery from the platforms

of the past. Essentially, modernization of legacy systems involves applying techniques such as refactoring, restructuring, translation, migration and integration to existing software assets.

In my short talk, I will motivate the need for model-driven evolution of legacy systems and the many challenges that are faced in this research space. A key challenge as it relates to complexity is the sheer size of some legacy systems (in terms of lines of code) and the effort required to make changes. An initial step toward model-driven legacy evolution solving the problem is parsing the existing source and recovering the design intent in the form of models for the domain, and then evolve the models according to some change request. The combination of model transformation and program transformation may provide the needed link to automate the mappings between the two technical spaces represented by models and legacy source code. The talk will show some initial examples in the avionics and high-performance computing domains.

NOTE: Some of the ideas for this talk come from the EDOC MELS 2004 workshop.

Modeling Complex Systems at IBM Research

Alan Hartman (IBM India Research Lab, IN)

This talk briefly describes four research projects carried out at IBM Research Labs in Israel, Japan, India, and the US. Each of the projects provides tooling and methodologies for dealing with particular types of complex systems. Each of the projects is motivated by real business problems and pain suffered by IBM's customers.

1. The Model Driven Systems Engineering (MDSE) project provides tooling for the development of embedded systems in the automotive, electronics, and aerospace industries.
2. The System Grokker is a tool used by architects in an interactive manner for the understanding, analysis, and evolution of legacy systems.
3. The Component Business Model (CBM) together with the Service Oriented Modeling and Architecture Modeling Environment (SOMA-ME) are tools used by consultants from IBM Global Services to analyze enterprises and introduce Service Oriented Architecture into the IT systems which support the enterprise.
4. The support for product line engineering is an ongoing effort in the electronics and semi-conductor industries to provide support for product line architectures in the context of the industry specific component models.

Keywords: Systems Engineering, Product Line Engineering, Service Oriented Architecture, Legacy Modernization

Towards an Advanced Model Driven Engineering Toolbox

Frédéric Jouault (INRIA, F)

Model Driven Engineering (MDE) is frequently presented as an important change in software development practices. However behind this new trend, one may recognize a lot of different objectives and solutions. This paper first studies the multiple facets of MDE, and its evolution in the recent period. Several new usage scenarios (i.e., reverse engineering, and models at runtime), which have appeared following the initial forward engineering scenario (i.e., PIM to PSM), are described. Then, the MetaBoxed modeling toolbox is introduced, and some of its applications to these scenarios are described.

Joint work of: Jouault, Frédéric; Bézivin, Jean; Barbero, Mikaël

Tackling the Model Heterogeneity Problem

Gerti Kappel (TU Wien, AT)

Model transformations are considered to play a central role in model engineering, especially for the development of complex software systems. During the last decade, several kinds of dedicated model transformation languages have been proposed, which allow specifying and executing transformations between source and target metamodels and their corresponding models, respectively. However, none of these languages, not even the QVT-standard proposed by the OMG, became generally accepted as a state-of-the-art technology for model transformations mainly due to the following two reasons. First, they lack suitable reuse mechanisms in order to reduce the high effort of specifying recurring transformations such as needed for solving structural heterogeneities between metamodels. Second, these languages exhibit an inherent impedance mismatch between the specification and the execution of model transformations, thus hampering both, understandability and debuggability.

To tackle these limitations, we propose a framework called TROPIC (Transformations on Petri Nets in Color) for developing model transformations. First, TROPIC facilitates reusability by providing an initial library of generic transformation operators for solving structural heterogeneities which may be bound to arbitrary metamodels and by allowing to extend this library on demand with new, user-defined, transformation operators, optionally composed out of already existing ones. Second, TROPIC tackles the impedance mismatch by supporting a dedicated runtime model in terms of Coloured Petri Nets, allowing for a homogeneous representation of all transformation artefacts (i.e., models, metamodels and the transformation logic itself), which enhances understandability and debuggability of model transformations.

Keywords: Model transformations, coloured Petri nets, reuse of model transformations, transformation operators

Joint work of: Kappel, Gerti; Wimmer, Manuel

Composition and Extension of Domain Specific Languages

Holger Krahn (TU Braunschweig, D)

This talk dissusses the composition and extension of Domain Specific Languages (DSLs) to simplify the development of new languages. It is based on experiences made during the development of the MontiCore framework.

Keywords: DSL

Joint work of: Krahn, Holger; Rumpe, Bernhard

Grammar convergence

Ralf Lämmel (Universität Koblenz-Landau, D)

Consider the situation where grammar knowledge is scattered over different program and documentation artifacts. For instance, there may be a language documentation (such as a "standard") that contains fragments of the syntax definition as well as samples. There is probably a parser specification that unavoidably commits to the details and idiosyncrasies of a particular technology. In fact, there may be multiple parser specifications that either compete with each other or define (intentionally) slightly different versions of the language at hand, or different semantics thereof. Also, there may be descriptions of the language that do not assume concrete syntax but that use an object model, an XML schema, algebraic data types, and others. Likewise, samples may be scattered, and may surface in quite different forms. Obviously, one would want to make sure that all scattered manifestations of grammar knowledge agree on each other in some reasonable sense. To this end, one can extract the scattered knowledge, and transform the extracted grammars by refactoring and abstraction so that one arrives at a common denominator.

Joint work of: Lämmel, Ralf; Zaytsev, Vadim

Model-Driven Software Evolution

Tom Mens (Université de Mons, BE)

This document contains the summary of a research project that I have just started up.

It explains some of the research challenges related to model-driven engineering and evolution of complex software systems that I am currently investigating.

Keywords: MDE, software evolution, quality improvement, inconsistency management, challenges, complexity

Model-Centric Software Adaptation (slides)

Oscar Nierstrasz (Universität Bern, CH)

We pose the challenge of *enabling change in long-lived software systems.* These systems must adapt and change to maintain their value, but mainstream languages and development environments tend to inhibit rather than enable change. To address this problem, we must cure two symptoms. First, the models we use to reason about software are disconnected from the code. Model-driven approaches help, but introduce other problems of model transformation and synchronization. Second, we lack good mechanisms to control the scope of change, especially in running systems. We argue that future systems should be model-centric, with models available at run-time, and context-aware, to manage and control the scope of change. With these mechanisms we should be able to monitor and reason about emergent properties of long-lived evolving systems.

Keywords: Software evolution, reflection, unanticipated change, context-awareness

Change-Enabled Software Systems (draft, to appear)

Oscar Nierstrasz (Universität Bern, CH)

Few real software systems are built completely from scratch nowadays. Instead, systems are built iteratively and incrementally, while integrating and interacting with components from many other systems.

Adaptation, reconfiguration and evolution are normal, ongoing processes throughout the lifecycle of a software system. Nevertheless the platforms, tools and environments we use to develop software are still largely based on an outmoded model that presupposes that software systems are closed and will not significantly evolve after deployment. We claim that in order to enable effective and graceful evolution of modern software systems, we must make these systems more amenable to change by (i) providing explicit, first-class models of software artifacts, change, and history at the level of the platform, (ii) continuously analysing static and dynamic evolution to track emergent properties, and (iii) closing the gap between the domain model and the developersŠ view of the evolving system. We outline our vision of dynamic, evolving software systems and identify the research challenges to realizing this vision.

Complex system models: engineering simulations

Fiona Polack (University of York, UK)

Complex systems are systems of systems. Modelling of components is straightforward, but environment must also be considered, and top-level requirements are for emergent behaviours. Simulation provides one approach to modelling, but simulation validation is hard. Some high integrity systems engineering approaches, relating to approximate validation and argumentation can be used.

Keywords: Complex Systems, modelling, Simulation, Validation

Use of Model-driven Techniques in Software Performance Analysis

Ralf Reussner (Universität Karlsruhe, D)

The talk will present a research agenda, where model-transformations will tackle the age-old problem of model-based performance engineering, namely, the problem, that while performance (or reliability) are properties of executing programms, an rough abstraction of the code (the software architecture) is used to predict such properties. As an architecture doe snot have a performance or a reliability, but only the executing software, one has to deal with the problem, that much information influencing these properties are not specified in software architectures. The approach will not only present architectural meta models for the prediction of performance, but also idea how to use parameterised transformations. The talk concludes with lessons learned from the use of model-driven techniques over the last reaearch where new research questions stem from.

Keywords: Architectural meta model, analysis of non-functional properties, model-driven software development

Complexity by the Dozen: Variability as a Complexity Driver in Product Lines and Adaptive Systems

Klaus Schmid (University of Hildesheim, D)

No matter how complex modeling is in a single system situation, this complexity is multiplied in the real world as typically not a single system, but rather a whole set of products must be engineered.

This problem is intrinsically related to the increasing need for flexible, self-adaptive and even context-aware systems - or to put it differently product lines at runtime.

We thus propose two presentations:

- Challenge presentation: in this we lay out
  a) the specific challenges and situations brought by product lines to the model-based community
  b) the specific challenges that are brought about by the migration from development time variability to runtime variability
- Technical presentation: in this we decribe a specific approach that supports the seamless migration from product lines to runtime variability

Attached material:

- a paper (to appear at DSPL-workshop / SPL conference) on issues resulting from the migration from development time to runtime variability
- a presentation on a specific approach that supports the integrated realization of develpment time and runtime variability (based on earlier presentation at Groningen)

Principles of Model-Based Development in Embedded Systems Development

Bernhard Schätz (TU München, D)

Model-based development of complex systems is simplified by several principles: a syntactic as well as semantic meta model; multi-dimensional modularization of the product under development including multi-hierarchy, multi-view, and multi-phase; analytic (quantitative and qualitative) and synthetic methods for both the syntactic and the semantic meta model. We show how these principles can be used for the development of embedded systems.

Ontologies for MECS

Steffen Staab (Universität Koblenz-Landau, D)

Ontologies encapsulate domain complexity. The talk shows some challenges and means to join the modeling of complex systems using ontologies.

Keywords: Ontology, MDE, MDA, UML, TwoUse

A Journey through the Secret Life of Models

Antonio Vallecillo (University of Malaga, ES)

Although Model Driven Software Development (MDSD) is achieving significant progress, it is still far from becoming a real Engineering discipline.

In fact, many of the difficult problems of the engineering of complex software systems are still unresolved, or simplistically addressed by many of the existing MDSD approaches. In this paper we outline three of the currently outstanding problems of MDSD on which we are now working, and propose some hints on how they can be addressed. The challenges are: the specification of the behavioral semantics of metamodels; the addition of time to these specifications so that models can be properly animated, simulated and analyzed; and the use of viewpoints and correspondences for specifying large-scale software systems.

Keywords: Model behavior, Time, Model Analysis, Viewpoint Modeling

Towards Model-Driven Software-Migration

Andreas Winter (Universität Koblenz-Landau, D)

Software migration deals with transferring existing software systems into new environments without changing their functionality. Usually, software migration is a first step in software maintenance which enables comprehensive software evolution activities.

Successfully migrating existing software systems requires analyzing and reusing the behavior of existing systems. From a model driven perspective, classical model driven software migration can be viewed as a transformation step transforming the original platform independent model into the new platform specific implementation.

This talk debunks classical model driven migration as a fairy tale and identifies new challenges in model driven software evolution to cope with legacy systems during software migration. These challenges include the need for integrated meta models, effective strategies to populate and extend models as well as reverse, forward and horizontal transformations.

Conquer Complexity using Object Scenarios for Analysis and Test

Albert Zündorf (Universität Kassel, D)

We propose to use Object Diagrams and their evolution during the execution of some complex operation as a means to reduce the complexity of software development. Our approach is use case scenario driven. We first split a scenario in a sequence of smaller steps with a textual description. Next, each step is analysed, one at a time, with an object diagram representing a concrete situation within the desired program. Focusing on a single concrete situation again reduces complexity. Once the whole scenario is analysed, the sequence of object diagrams is turned into an automatic JUnit test. This test creates the object structure modeling the start situation, invokes the modeled operations and ckecks whether the resulting object structures match the scenario model.

Next, the control structures of the analysed operations is developed using high level pseudo code notation. This breaks down the complexity of the whole operation into smaller steps. Then, for one step at a time, the pseudo code is rephrased to map on the object model developed during the analysis phase. Finally, each step is programmed or modeled with Fujaba.

This approach works not only for the development of new functionality. We use a very similar approach also for maintenance. During maintenance, we visualize the runtime object structures that are deployed by the operations we want to modify. Then we discuss concrete scenarios and how the behavior shall be modified. And then this is turned into program modifications.

Keywords: Story driven modelling, modelling process, model driven engineering