# DAGSTUHL REPORTS

**Volume 3, Issue 4, April 2013**

**ISSN 2192-5283**

*Published online and open access by*
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.
Online available at http://www.dagstuhl.de/dagrep

*Publication date*
August, 2013

*Bibliographic information published by the Deutsche Nationalbibliothek*
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at `http://dnb.d-nb.de`.

*Aims and Scope*
The periodical *Dagstuhl Reports* documents the program and the results of Dagstuhl Seminars and Dagstuhl Perspectives Workshops.
In principal, for each Dagstuhl Seminar or Dagstuhl Perspectives Workshop a report is published that contains the following:
- an executive summary of the seminar program and the fundamental results,
- an overview of the talks given during the seminar (summarized as talk abstracts), and
- summaries from working groups (if applicable).

This basic framework can be extended by suitable contributions that are related to the program of the seminar, e.g. summaries from panel discussions or open problem sessions.

Report from Dagstuhl Seminar 13141

# Formal Verification of Distributed Algorithms

**Edited by**

**Bernadette Charron-Bost[1], Stephan Merz[2], Andrey Rybalchenko[3], and Josef Widder[4]**

1 **École polytechnique, Palaiseau, FR,** `charron@lix.polytechnique.fr`
2 **INRIA, Nancy, FR,** `stephan.merz@loria.fr`
3 **TU München, DE,** `rybal@in.tum.de`
4 **TU Wien, AT,** `widder@forsyte.tuwien.ac.at`

## Abstract

The Dagstuhl Seminar 13141 "Formal Verification of Distributed Algorithms" brought together researchers from the areas of distributed algorithms, model checking, and semi-automated proofs with the goal to establish a common base for approaching the many open problems in verification of distributed algorithms. In order to tighten the gap between the involved communities, who have been quite separated in the past, the program contained tutorials on the basics of the concerned fields. In addition to technical talks, we also had several discussion sessions, whose goal was to identify the most pressing research challenges. This report describes the program and the outcomes of the seminar.

**Seminar** 1.–5. April, 2013 – www.dagstuhl.de/13141
**1998 ACM Subject Classification** C.2 Computer-Communication Networks, D.3.1 Formal Definitions and Theory, D.2.4 Software/Program Verification, F.3 Logics and Meanings of Programs
**Keywords and phrases** Distributed algorithms, semi-automated proofs, model checking
**Digital Object Identifier** 10.4230/DagRep.3.4.1
**Edited in cooperation with** Thomas Nowak

## 1 Executive Summary

*Bernadette Charron-Bost*
*Stephan Merz*
*Andrey Rybalchenko*
*Josef Widder*

While today's society depends heavily on the correct functioning of distributed computing systems, the current approach to designing and implementing them is still error prone. This is because there is a methodological gap between the theory of distributed computing and the practice of designing and verifying the correctness of reliable distributed systems. We believe that there are two major reasons for this gap: On the one hand, distributed computing models are traditionally represented mainly in natural language, and algorithms are described in pseudo code. The classical approach to distributed algorithms is thus informal, and it is not always clear under which circumstances a given distributed algorithm actually is correct. On the other hand, distributed algorithms are designed to overcome non-determinism due to issues that are not within the control of the distributed algorithm, including the system's timing behavior or faults of some components. Such issues lead to a huge executions space which is the major obstacle in applying verification tools to distributed algorithms.

The rationale behind our Dagstuhl seminar was that closing the methodological gap requires collaboration from researchers from distributed algorithms and formal verification. In order to spur the interaction between the communities, the program contained the following overview talks on the related subjects:

**Distributed algorithms:** Eric Ruppert (York University)
**Semi-automated proofs:** John Rushby (SRI)
**Parameterized model checking:** Muralidhar Talupur (Intel)

In addition to the tutorials, we organized several open discussion rounds. The seminar participants identified modeling issues as a central question, which confirmed one of our motivation for the seminar, namely, the lack of a universal model for distributed algorithms. Hence, one of the discussion rounds was exclusively devoted to this topic. Unlike sequential programs, whose semantics is well understood and closely follows the program text, the executions of distributed algorithms are to a large extent determined by the environment, including issues such as the distribution of processes, timing behavior, inter-process communication, and component faults. Models of distributed algorithms and systems embody different assumptions about how the environment behaves. These hypotheses are often left implicit but are of crucial importance for assessing the correctness of distributed algorithms. The discussions during the seminar raised the awareness of these issue among the researchers, and showed that research in this area is a necessary first step towards a structured approach to formal verification of distributed algorithms. In addition to modeling, we discussed issues such as benchmarks, implementation of distributed algorithms, or application areas of distributed algorithms.

To round-off the technical program, we had several short presentations by participants who presented their past and current work in the intersection of formal methods and distributed algorithms, and a joint session with the other seminar going on concurrently at Dagstuhl on Correct and Efficient Accelerator Programming. The topics of the talks spanned large parts of the concerned areas, for instance, there were talks on model checking techniques such as partial order reductions or abstractions, and their applications to distributed algorithms; several talks focused on proof assistants, and how they can be used to verify distributed algorithms; some talks considered concurrent systems, and some focused on transactional memory. The atmosphere during these sessions was very constructive, and the short talks were always followed by elaborate and insightful discussions.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Partial-Order Reductions: Landscape & Practice

*Péter Bokor (ALTEN Engineering, Berlin, DE)*

**Joint work of** Bokor, Péter; Kinder, Johannes; Serafini, Marco; Suri, Neeraj
**Main reference** P. Bokor, J. Kinder, M. Serafini, N. Suri, "Supporting domain-specific state space reductions through local partial-order reduction," in Proc. of the 26th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE'11), pp. 113–122. IEEE Press, New York City, 2011.
**URL** http://dx.doi.org/10.1109/ASE.2011.6100044

This talk is about mainstream partial-order reductions (such as DPOR), their extensions (LPOR), new approaches (our current ongoing work Ostrich), and applications (to fault-tolerant distributed protocols).

### 3.2 Automated Repair of Distributed Systems: Beyond Verification

*Borzoo Bonakdarpour (University of Waterloo, CA)*

Although verification of concurrent and distributed applications has recently made considerable attention in the recent years, correct construction of such applications still remains a challenge. This is simply due to the inherently complex structure of concurrent applications caused by non-determinism and occurrence of faults (in a distributed setting). To deal with the subtleties of developing concurrent applications, my position is to focus on formal methods that automatically build such applications that are correct-by-construction. In this talk, I briefly describe the efforts made for achieving correctness by construction for concurrent/distributed applications in the area of automated repair of concurrent models.

### 3.3 Formal Proofs in Coq of Local Computation Systems

*Pierre Castéran (University of Bordeaux, FR)*

**Joint work of** Castéran, Pierre; Filou, Vincent; Fontaine, Allyx
**Main reference** P. Castéran, V. Filou, "Tasks, types and tactics for local computation systems," Studia Informatica Universalis 9(1):39–86, 2011.
**URL** http://studia.complexica.net/index.php?option=com_content&view=article&id=186%3Atasks-types-and-tactics-for-local-computation-systems-pp-39-86-

We present a library written for the Coq proof assistant, for reasoning about a quite abstract model of distributed computing based on graph rewriting: Local Computations Systems. A first development allowed us to prove some facts about the expressive power of several sub-classes of such systems, e.g., impossibility results and certified transformations. Directions for future evolutions will be also discussed, in particular reasoning on dynamic graphs and self-stabilizing systems.

## 3.4 Semantics of Eventually Consistent Systems

*Alexey Gotsman (IMDEA Software, Madrid, ES)*

**Joint work of** Burckhardt, Sebastian; Gotsman, Alexey; Yang, Hongseok
**Main reference** S. Burckhardt, A. Gotsman, H. Yang, "Understanding Eventual Consistency," Technical Report MSR-TR-2013-39, Microsoft Research, 2013.
**URL** http://software.imdea.org/~gotsman/papers/distrmm.pdf

Modern geo-replicated databases underlying large-scale Internet services guarantee immediate availability and tolerate network partitions at the expense of providing only weak forms of consistency, commonly dubbed eventual consistency. At the moment there is a lot of confusion about the semantics of eventual consistency, as different systems implement it with different sets of features and in subtly different forms, stated either informally or using disparate and low-level formalisms.

We address this problem by proposing a framework for formal and declarative specification of the semantics of eventually consistent systems using axioms. Our framework is fully customisable: by varying the set of axioms, we can rigorously define the semantics of systems that combine any subset of typical guarantees or features, including conflict resolution policies, session guarantees, causality guarantees, multiple consistency levels and transactions. We prove that our specifications are validated by an example abstract implementation, based on algorithms used in real-world systems. These results demonstrate that our framework provides system architects with a tool for exploring the design space, and lays the foundation for formal reasoning about eventually consistent systems.

This is joint work with Sebastian Burckhardt (MSR Redmond) and Hongseok Yang (Oxford).

## 3.5 A Fault-tolerant Communication Mechanism for Cooperative Robots

*Serge Haddad (ENS Cachan, FR)*

**Joint work of** El Haddad, Joyce; Haddad, Serge
**Main reference** J. El Haddad, S. Haddad, "A fault-tolerant communication mechanism for cooperative robots," International Journal of Production Research, 42(14):2793–2808, 2004.
**URL** http://dx.doi.org/10.1080/00207540410001705185

Operations in unpredictable environments require coordinating teams of robots. This coordination implies peer-to-peer communication between the team's robots, resource allocation, and coordination. We address the problem of autonomous robots which alternate between execution of individual tasks and peer-to-peer communication. Each robot keeps in its permanent memory a set of locations where it can meet some of the other robots. The proposed protocol is constructed by two layered modules (sub-algorithms: a self-stabilizing scheduling and a construction of a minimum-hop path forest). The first self-stabilizing algorithm solves the management of visits to these locations ensuring that, after the stabilizing phase, every visit to a location will lead to a communication. We model the untimed behaviour of a robot by a Petri net and the timed behaviour by an (infinite) Discrete Time Markov Chain. Theoretical results in this area are then combined in order to establish the proof of the algorithm. The second self-stabilizing algorithm computes the minimum-hop path between a specific robot's location and the locations of all the other robots of the system inorder to implement routing.

## 3.6 Scaling Up Interactive Verification

*Gerwin Klein (NICTA & UNSW, Sydney, AU)*

This talk gives a brief overview of the formal verification of the seL4 microkernel. I will cover the proof of functional correctness, later high-level security properties, the extension of the proof to the binary level, and the effect of maintenance on the verification. After this, the idea is to open the floor to a more free-form discussion on the experience of large-scale software verification and the applicability of our experience to the distributed algorithms section.

## 3.7 Parameterized Model Checking of Fault-Tolerant Broadcasting Algorithms

*Igor Konnov (TU Wien, AT)*

**Joint work of** John, Annu; Konnov, Igor; Schmid, Ulrich; Veith, Helmut; Widder, Josef
**Main reference** A. John, I. Konnov, U. Schmid, H. Veith, J. Widder, "Counter Attack on Byzantine Generals: Parameterized Model Checking of Fault-tolerant Distributed Algorithms," arXiv:1210.3846v2 [cs.LO], 2013.
**URL** http://arxiv.org/abs/1210.3846v2

We introduce an automated parameterized verification method for fault-tolerant distributed algorithms (FTDA). FTDAs are parameterized by both the number of processes and the assumed maximum number of Byzantine faulty processes. At the center of our technique is a parametric interval abstraction (PIA) where the interval boundaries are arithmetic expressions over parameters. Using PIA for both data abstraction and a new form of counter abstraction, we reduce the parameterized problem to finite-state model checking. We demonstrate the practical feasibility of our method by verifying several variants of the well-known distributed algorithm by Srikanth and Toueg. Our semi-decision procedures are complemented and motivated by an undecidability proof for FTDA verification which holds even in the absence of interprocess communication. To the best of our knowledge, this is the first paper to achieve parameterized automated verification of Byzantine FTDA.

## 3.8 Verification of a Quasi Certification Protocol over a DHT

*Fabrice Kordon (UPMC, Lab. LIP6, Paris, FR)*

**Joint work of** Kordon, Fabrice; Bonnaire, Xavier; Cortés, Rudyar; Marin, Olivier

Building a certification authority that is both decentralized and fully reliable is impossible. However, the limitation thus imposed on scalability is unacceptable for many types of information systems, such as e-government services. We propose a solution to build an highly reliable certification authority, based on a distributed hash table and a dedicated protocol ensuring a very low probability of arbitrary failure. Thus, in practice, false positives should never occur. This talk briefly presents the protocol and shows its verification in two steps: (1) a formal model to assess that the protocol behaves as expected in an "ideal world" where communications are reliable, and, (2) a probabilistic analysis to evaluate the probability of failure of the certification.

## 3.9 Finding Non-terminating Executions in Distributed Asynchronous Programs

*Akash Lal (Microsoft Research India, Bangalore, IN)*

**License** Creative Commons BY 3.0 Unported license
© Akash Lal
**Joint work of** Emmi, Michael; Lal, Akash
**Main reference** M. Emmi, A. Lal, "Finding non-terminating executions in distributed asynchronous programs," in Proc. of the 19th Int'l Symp. on Static Analysis (SAS'12), LNCS, Vol. 7460, pp. 439–455, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-33125-1_29

Programming distributed and reactive asynchronous systems is complex due to the lack of synchronization between concurrently executing tasks, and arbitrary delay of message-based communication. As even simple programming mistakes have the capability to introduce divergent behavior, a key liveness property is *eventual quiescence*: for any finite number of external stimuli (e.g., client-generated events), only a finite number of internal messages are ever created.

In this work we propose a practical three-step reduction-based approach for detecting divergent executions in asynchronous programs. As a first step, we give a code-to-code translation reducing divergence of an asynchronous program $P$ to completed state-reachability, i.e., reachability to a given state with no pending synchronous tasks, of a polynomially-sized asynchronous program $P'$. In the second step, we give a code-to-code translation under-approximating completed state-reachability of $P'$ by state-reachability of a polynomially-sized recursive sequential program $P''(K)$, for the given analysis parameter $K$. Following Emmi et al.'s delay-bounding approach, $P''(K)$ encodes a subset of $P'$, and thus of $P$, by limiting scheduling nondeterminism. As $K$ is increased, more possibly divergent behaviors of $P$ are considered, and in the limit as $K$ approaches infinity, our reduction is complete for programs with finite data domains. As the final step we give the resulting state-reachability query to an of-the-shelf SMT-based sequential program verification tool.

We demonstrate the feasibility of our approach by implementing a prototype analysis tool called Alive, which detects divergent executions in several hand-coded variations of textbook distributed algorithms.

## 3.10 A Framework for Formally Verifying Software Transactional Memory (and Other Concurrent Algorithms)

*Victor Luchangco (Oracle Corporation, Burlington, US)*

**License** Creative Commons BY 3.0 Unported license
© Victor Luchangco
**Joint work of** Lesani, Mohsen; Luchangco, Victor; Moir, Mark
**Main reference** M. Lesani, V. Luchangco, M. Moir, "A framework for formally verifying software transactional memory algorithms," in Proc. of the 23rd Int'l Conf. on Concurrency Theory (CONCUR'12), LNCS, Vol. 7454, pp. 516–530. Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-32940-1_36

We present a framework for verifying transactional memory (TM) algorithms. Specifications and algorithms are specified using I/O automata, enabling hierarchical proofs that the algorithms implement the specifications. We have used this framework to develop what we believe is the first fully formal machine-checked verification of a practical TM algorithm: the NOrec algorithm of Dalessandro, Spear and Scott. Our framework is available for others to use and extend. New proofs can leverage existing ones, eliminating significant work and complexity.

## 3.11 Verification of Fault-Tolerant Distributed Algorithms in the Heard-Of Model

*Stephan Merz (LORIA, Nancy, FR)*

**Joint work of** Débrat, Henri; Merz, Stephan
**Main reference** H. Debrat, S. Merz, "Verifying fault-tolerant distributed algorithms in the Heard-Of model," Archive of Formal Proofs, 2012.
**URL** http://afp.sf.net/entries/Heard_Of.shtml

Distributed algorithms are quite subtle, both in the way they function and in the hypotheses assumed for their correctness. Moreover, many different computational models exist in the literature, but comparisons between algorithms expressed in different models is difficult. Formal verification can help ascertain the correctness of a given algorithm w.r.t. well-specified hypotheses. We present work on the formal verification of fault-tolerant distributed algorithms in the Heard-Of model introduced by Charron-Bost and Schiper [1, 2]. In this model, algorithms execute in communication-closed rounds and are subject to hypotheses expressed in terms of Heard-Of sets, i.e., the sets of processes from which messages are received in a given round. We formally prove a reduction theorem that justifies verifying these algorithms in a coarse-grained (synchronous) model of execution. In this way, entire system rounds become the unit of atomicity, and verification becomes much simpler than when interleavings of individual process actions are considered. We have verified six different Consensus algorithms that differ with respect to the presence of a coordinator, the types and numbers of faults they tolerate (both benign and Byzantine failures are considered), and the degree of synchrony that is required for correctness. Both the reduction proof and the verification of the various algorithms are carried out in the proof assistant Isabelle/HOL [3], and they are available online [4].

### References

1 Bernadette Charron-Bost and André Schiper. The Heard-Of model: computing in distributed systems with benign faults. Distributed Computing 22(1):49-71, 2009.
2 Martin Biely, Bernadette Charron-Bost, Antoine Gaillard, Martin Hutle, André Schiper, and Josef Widder. Tolerating corrupted communication. Proc. 26th Annual ACM Symposium on Principles of Distributed Computing (PODC'07), pp. 244–253. ACM, New York City, 2007.
3 Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. Isabelle/HOL. A Proof Assistant for Higher-Order Logic. LNCS 2283, Springer, 2002.
4 Henri Debrat and Stephan Merz. Verifying fault-tolerant distributed algorithms in the Heard-Of model. Archive of Formal Proofs, http://afp.sf.net/entries/Heard_Of.shtml, 2012.

## 3.12 Verifying Consensus . . . Using Process Calculi, State Machines, and Proof Checkers

*Uwe Nestmann (TU Berlin, DE)*

We focus on the gap between pseudo code, as often used for the description of distributed algorithms, and correctness proofs, as usually written in math-enhanced natural language. Trying to bridge the gap, we discuss the use of process calculi and state machines, as well as their connection. We also briefly report on the mechanisation of state-machine-based correctness proofs within the proof assistant Isabelle.

**References**

1 R. Fuzzati, M. Merro and U. Nestmann. Distributed Consensus, Revisited. *Acta Inf.*, 44(6):377–425, 2007.

2 M. Kühnrich and U. Nestmann. On Process-Algebraic Proof Methods for Fault Tolerant Distributed Systems. In D. Lee, A. Lopes and A. Poetzsch-Heffter, eds, *FMOODS/FORTE*, volume 5522 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2009.

3 P. Küfner, U. Nestmann and C. Rickmann. Formal Verification of Distributed Algorithms – From Pseudo Code to Checked Proofs. In J. C. M. Baeten, T. Ball and F. S. de Boer, eds, *IFIP TCS*, volume 7604 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2012.

4 U. Nestmann and R. Fuzzati. Unreliable Failure Detectors via Operational Semantics In V. A. Saraswat, ed, *ASIAN*, volume 2896 of *Lecture Notes in Computer Science*, pages 54–71. Springer, 2003.

5 U. Nestmann, R. Fuzzati and M. Merro. Modeling Consensus in a Process Calculus. In R. M. Amadio and D. Lugiez, eds, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 393–407. Springer, 2003.

## 3.13 Tutorial on Distributed Algorithms

*Eric Ruppert (York University, Toronto, CA)*

I gave some background information on the way distributed algorithm designers model distributed systems and define correctness properties. I briefly described some of the challenges faced in designing distributed algorithms and some techniques used to overcome them, with examples of algorithms that use the techniques. These techniques include quorums, repeated reads to obtain consistent views, timestamps, helping, using CAS to synchronize, pointer swinging and locks.

## 3.14 Getting the Best out of General-Purpose Tools: Theorem Provers and Infinite-Bounded Model Checker

*John Rushby (SRI, Menlo Park, US)*

In traditional "by hand" formal verification of distributed algorithms it is often beneficial to work with a model of computation specialized to the issue of primary concern (e.g., timed automata). But the best developed, most powerful mechanized verification tools tend to be general-purpose (or specialized to a different model than the one you want). I describe and demonstrate some techniques for getting the best out of general-purpose tools through adjustments to (the representation of) models that better exploit the underlying automation. I cover representation of nondeterminism in a theorem prover (illustrated using Byzantine Agreement in PVS) and timed systems in an infinite bounded model checker (illustrated using Biphase Mark in SAL). I also briefly describe computational reflection, and some prospects and hopes for the future.

## 3.15 An Epistemic Perspective on Consistency of Concurrent Computations

*Andrey Rybalchenko (TU München, DE)*

**Joint work of** von Gleissenthal, Klaus; Rybalchenko, Andrey
**Main reference** K. v. Gleissenthall, A. Rybalchenko, "An Epistemic Perspective on Consistency of Concurrent Computations," arXiv:1305.2295v1 [cs.LO], 2013.
**URL** http://arxiv.org/abs/1305.2295

Consistency properties of concurrent computations, e.g., sequential consistency, linearizability, or eventual consistency, are essential for devising correct concurrent algorithms. In this paper, we present a logical formalization of such consistency properties that is based on a standard logic of knowledge. Our formalization provides a declarative perspective on what is imposed by consistency requirements and provides some interesting unifying insight on differently looking properties.

## 3.16 Unidirectional Channel Systems Can Be Tested

*Philippe Schnoebelen (ENS Cachan, FR)*

**Joint work of** Jančar, Petr; Karandikar, Prateek; Schnoebelen, Philippe
**Main reference** P. Jančar, P. Karandikar, P. Schnoebelen, "Unidirectional channel systems can be tested," in Proc. of the 7th IFIP TC1/WG2.2 Int'l Conf. on Theoretical Computer Science (TCS'12), LNCS, Vol. 7604, pp. 149–163, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-33475-7_11

"Unidirectional channel systems" (Chambart & Schnoebelen, CONCUR 2008) are systems where one-way communication from a sender to a receiver goes via one reliable and one unreliable (unbounded fifo) channel. Equipping these systems with the possibility of testing regular properties on the contents of channels makes verification undecidable. Decidability is preserved when only emptiness and nonemptiness tests are considered: the proof relies on a series of reductions eventually allowing us to take advantage of recent results on Post's Embedding Problem.

## 3.17 Formal Verification of Distributed Algorithms at TTTech

*Wilfried Steiner (TTTech Computertechnik, Vienna, AT)*

System design for safety-critical systems and mixed-criticality systems, such as aerospace or space applications, is inherently complex and demands a level of quality assurance often only to be met by the use of formal methods. This is due to the tightly interwoven requirements of fault tolerance, the ability to sustain partial failures of the system, and real-time control. One key element of a safety-critical system is its communication infrastructure, which more and more determines the overall system architecture. With its central role, the correct design of the communication infrastructure, and in particular the distributed algorithms that the infrastructure implements, is a crucial pre-requisite for mission success. In this talk we discuss how formal methods have been used during the design of the TTEthernet communication infrastructure and their general use at TTTech.

## 3.18 Tutorial on Parameterized Model Checking

*Murali Talupur (Intel SCL, Hillsboro, US)*

With the move towards multi-core processors and SoCs (systems-on-chip) parameterized verification of distributed protocols has taken on a new urgency. Protocols like cache coherence protocols, bus lock protocols form the bedrock on which these processors are built and verifying them is a challenging task. In this talk I will describe a highly scalable and automated method, called the CMP+ Flows method, for formally and parametrically verifying protocols. As the name indicates the method has two components. The first component, the CMP method, is a compositional reasoning technique that uses abstraction to reduce an unbounded parameterized verification problem to a finite problem that can then be model checked. The abstraction operation is completely automatic but the user has to supply lemmas (or candidate invariants) to progressively refine the abstraction. Though the CMP method imposes less manual burden than pure theorem proving, supplying lemmas is still a non-trivial task, especially for large industrial protocols. The second component of our method addresses this gap by showing how to derive invariants automatically from informal design artifacts called Flows. Flows are essentially partial orders on system events, such as sending and receiving of messages, that architects typically use to conceive the protocols. These are readily available in design documents and as we show they yield powerful invariants. The combined CMP+ Flows method is extremely scalable while imposing minimal burden on the user. Using this method we have verified multiple industrial strength cache coherence protocols and other co-ordination protocols. To our knowledge no other method has been used successfully to verify protocols of such sizes.

## 3.19 Correctness without Serializabilty: Verifying Transactional Programs under Snapshot Isolation

*Serdar Tasiran (Koc University, Istanbul, TR)*

We present a static verification approach for programs running under snapshot isolation (SI) and similar relaxed transactional semantics. In a common pattern in distributed and concurrent programs, transactions each read a large portion of shared data, perform local computation, and then modify a small portion of the shared data. Requiring conflict serializability in this scenario results in serial execution of transactions or worse, and performance suffers. To avoid such performance problems, relaxed conflict detection schemes such as snapshot isolation (SI) are used widely. Under SI, transactions are no longer guaranteed to be serializable, and the simplicity of reasoning sequentially within a transaction is lost. In this paper, we present an approach for statically verifying properties of transactional programs operating under SI. Differently from earlier work, we handle transactional programs even when they are designed not to be serializable.

In our approach, the user first verifies his program in the static verification tool VCC pretending that transactions run sequentially. This task requires the user to provide program annotations such as loop invariants and function pre- and post-conditions. We then apply a source-to-source transformation which augments the program with an encoding of the SI semantics. Verifying the resulting program with transformed user annotations and specifications is equivalent to verifying the original transactional program running under SI—a fact we prove formally. Our encoding preserves the modularity and scalability of VCC's verification approach. We applied our method successfully to benchmark programs from the transactional memory literature. In each benchmark, we were able to verify the encoded program without manually providing any extra annotations beyond those required for verifying the program sequentially. The correctness argument of the sequential versions generalized to SI, and verification times were similar.

## 3.20 (Dis)Proof Automation: What We Can Do, What We Could Do and What Is Needed?

*Christoph Weidenbach (MPI für Informatik, Saarbrücken, DE)*

After an introduction to the underlying principles of designing automated reasoning systems, I discuss FOL($T$), the hierarchic combination of a theory $T$ and first-order logic. In particular for the case where $T$ is a language of arithmetic, e.g., linear rational arithmetic. I show that this language is expressive enough to represent timed, hybrid, and probabilistic automata. Superposition-based reasoning delivers a decision procedure for known decidable reasoning challenges and beyond. The language also strictly generalizes the SMT setting as it considers universally quantified variables in addition to constants.

### 3.21 Efficient Checking of Link-Reversal-Based Concurrent Systems

*Josef Widder (TU Wien, AT)*

**License** Creative Commons BY 3.0 Unported license
© Josef Widder
**Joint work of** Függer, Matthias; Widder, Josef
**Main reference** M. Függer, J. Widder, "Efficient checking of link-reversal-based concurrent systems," in Proc. of the 23rd Int'l Conf. on Concurrency Theory (CONCUR'12), LNCS, Vol. 7454, pp. 486–499, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-32940-1_34

Link reversal is an algorithmic method with various applications. Originally proposed by Gafni and Bertsekas in 1981 for routing in radio networks, it has been later applied also to solve concurrency related problems as mutual exclusion, resource allocation, and leader election. For resource allocation, conflicts can be represented by conflict graphs, and link reversal algorithms work on these graphs to resolve conflicts. In this talk I explain that executions of link reversal algorithms on large graphs are similar (a notion which I make precise) to executions on smaller graphs. This similarity then allows to verify linear time temporal properties of the large systems, by verifying a smaller one.

## 4 Panel Discussions

The tentative program included three time slots for discussions. It was intended to have several working groups in parallel where specific topics would be discussed, and then report on the outcomes in a joint session. However, just listing the topics for the working groups sparked a lively discussion, in which most of the participants participated actively. Because the format of seminar-wide discussions turned out to be fruitful, we decided to stick with seminar-wide, moderated sessions.

In the first session, participants were asked what they considered the major open questions in the area of formal verification of distributed algorithms, and what kind of information from the other community they would need to make progress in this area. We identified modeling as the most urgent point: While the formal methods community is used to have a precise description of the object they consider (programming languages, input languages of model checking tools, etc.), distributed algorithms are typically given in pseudo code only. This formalization gap appeared to be crucial, and it was decided to devote the second discussion session to this subject. The final discussion session was devoted to the next steps we could take to bring the concerned communities together. We list the major topics that were discussed:

### 4.1 Session 1: What are the problems?

- There exists a large variety of **modeling frameworks**, corresponding to different classes of systems. The DA community usually presents models and algorithms informally (using text and pseudo-code), whereas FM researchers and tools require formal semantics definitions. Is formalization just a tedious detail or is it a contribution in itself? Is there a classification of relevant computational models and their relationships?
- A collection of **benchmark problems** could give some guidance to the FM community. Different formal methods could tackle these problems at different levels of abstraction,

enabling their comparison over practically relevant case studies. These should include novel paradigms of the DA field and different application areas. A collection of verified algorithms, as well as results of **verification competitions** would help DA researchers evaluate the potential and the usability of FM methods and tools.

- Within the FM community, an important line of research is about the **integration of different techniques** such as model checking, proof assistants, SMT solving, etc. Doing so requires consistent semantic models and raises the issue of trust in the results obtained in this way.
- Beyond the verification of distributed algorithms at a high level of abstraction, issues of **verified implementation** of distributed systems, as well as formal assurances on non-functional properties such as **performance or security** are of great interest. To what extent can formal verification play a role in **certification processes** that are used in safety-critical settings? Is it possible to obtain guarantees on systems that integrate verified and unverified components?

## 4.2 Session 2: Modeling

- The main purpose of a formal model is to clearly express the **semantics** of an algorithm or system. There is a tradeoff between models expressed in natural language and pseudo-code (concise and readable but potentially ambiguous) and formal semantics description (complete and precise but maybe cluttered with too much detail).
- Different models are geared towards different **purposes**. For example, there is a tradeoff between efficient checking vs. the ease of expressing distributed algorithms.
- For distributed algorithms that are intended to operate continually, specifying **initial states** can be problematic. For example, there is a subtle difference between the Consensus problem where all nodes start from scratch, and the repeated Consensus problem where nodes may start at vastly different times.
- The DA community should provide a catalogue of the most important **existing models** for distributed algorithms. This includes variations of shared memory vs. message passing models, round-based vs. asynchronous models, failure models etc. The **system model** should be distinguished from the **computational model**.

## 4.3 Session 3: Follow-up

- A **follow-up seminar** should be organized in a few years, presenting progress on research into the issued raised during the seminar. Besides another Dagstuhl seminar, a workshop gathering researchers from the DA and FM communities could be organized as a satellite of a major conference, such as FLOC 2014 in Vienna. It would also be useful to raise the awareness of the topics discussed in this seminar by inviting DA researchers to give talks at FM conferences, and vice versa.
- The topic of **models** was identified as the most important one, and it would be useful to work out a catalogue or classification, as mentioned above.
- Research about the application of FM methods and tools on interesting distributed algorithms would benefit from maintaining a list of verified algorithms, e.g. in the form of a **Wiki** page to which different people could contribute.

## Participants

- Béatrice Bérard
UPMC, Lab. LIP6 – Paris, FR
- Péter Bokor
ALTEN Engineering – Berlin, DE
- Borzoo Bonakdarpour
University of Waterloo, CA
- Pierre Castéran
University of Bordeaux, FR
- Bernadette Charron-Bost
Ecole Polytechnique – Palaiseau, FR
- Marie Duflot
LORIA & INRIA Nancy, FR
- Cormac Flanagan
University of California – Santa Cruz, US
- Matthias Függer
TU Wien, AT
- Alexey Gotsman
IMDEA Software – Madrid, ES
- Serge Haddad
ENS – Cachan, FR
- Gerwin Klein
NICTA & UNSW – Sydney, AU
- Igor Konnov
TU Wien, AT
- Fabrice Kordon
UPMC, Lab. LIP6 – Paris, FR
- Akash Lal
Microsoft Research India – Bangalore, IN
- Victor Luchangco
Oracle Corporation – Burlington, US
- Stephan Merz
LORIA – Nancy, FR
- Uwe Nestmann
TU Berlin, DE
- Thomas Nowak
Ecole Polytechnique – Palaiseau, FR
- Eric Ruppert
York University – Toronto, CA
- John Rushby
SRI – Menlo Park, US
- Andrey Rybalchenko
TU München, DE
- André Schiper
EPFL – Lausanne, CH
- Klaus Schneider
TU Kaiserslautern, DE
- Philippe Schnoebelen
ENS – Cachan, FR
- Wilfried Steiner
TTTech Computertechnik – Wien, AT
- Murali Talupur
Intel SCL – Hillsboro, US
- Serdar Tasiran
Koc University – Istanbul, TR
- Helmut Veith
TU Wien, AT
- Christoph Weidenbach
MPI für Informatik – Saarbrücken, DE
- Jennifer L. Welch
Texas A&M University – College Station, US
- Josef Widder
TU Wien, AT
- Karsten Wolf
Universität Rostock, DE

Report from Dagstuhl Seminar 13142

# Correct and Efficient Accelerator Programming

**Edited by**

**Albert Cohen[1], Alastair F. Donaldson[2], Marieke Huisman[3], and Joost-Pieter Katoen[4]**

1 **ENS – Paris, FR,** Albert.Cohen@inria.fr
2 **Imperial College London, GB,** alastair.donaldson@imperial.ac.uk
3 **University of Twente, NL,** Marieke.Huisman@ewi.utwente.nl
4 **RWTH Aachen University, DE,** katoen@cs.rwth-aachen.de

**Abstract**

This report documents the program and the outcomes of Dagstuhl Seminar 13142 "Correct and Efficient Accelerator Programming". The aim of this Dagstuhl seminar was to bring together researchers from various sub-disciplines of computer science to brainstorm and discuss the theoretical foundations, design and implementation of techniques and tools for correct and efficient accelerator programming.

**Seminar** 1.–4. April, 2013 – www.dagstuhl.de/13142
**1998 ACM Subject Classification** C.1.4 Processor Architectures: Parallel Architectures, D.1.3 Programming Techniques: Concurrent Programming, D.3.4 Programming Languages: Processors – Compilers, Optimization, F.3.1 Logics and Meanings of Programs: Specifying and Verifying and Reasoning about Programs
**Keywords and phrases** Accelerator programming, GPUs, Concurrency, Formal verification, Compilers, Memory models, Architecture, Parallel programming models
**Digital Object Identifier** 10.4230/DagRep.3.4.17
**Edited in cooperation with** Jeroen Ketema

## 1 Executive Summary

*Albert Cohen*
*Alastair F. Donaldson*
*Marieke Huisman*
*Joost-Pieter Katoen*

In recent years, massively parallel accelerator processors, primarily GPUs, have become widely available to end-users. Accelerators offer tremendous compute power at a low cost, and tasks such as media processing, simulation and eye-tracking can be accelerated to beat CPU performance by orders of magnitude. Performance is gained in energy efficiency and execution speed, allowing intensive media processing software to run in low-power consumer devices. Accelerators present a serious challenge for software developers. A system may contain one or more of the plethora of accelerators on the market, with many more products anticipated in the immediate future. Applications must exhibit portable correctness, operating correctly on any configuration of accelerators, and portable performance, exploiting processing power and energy efficiency offered by a wide range of devices.

The seminar focussed on the following areas:

- Novel and attractive methods for constructing system-independent accelerator programs;
- Advanced code generation techniques to produce highly optimised system-specific code from system-independent programs;
- Scalable static techniques for analysing system-independent and system-specific accelerator programs both qualitatively and quantitatively.

The seminar featured five tutorials providing an overview of the landscape of accelerator programming:

- Architecture – Anton Lokhmotov, ARM
- Programming models – Lee Howes, AMD
- Compilation techniques – Sebastian Hack, Saarland University
- Verification – Ganesh Gopalakrishnan, University of Utah
- Memory models – Jade Alglave, University College London

In addition, there were short presentations from 12 participants describing recent results or work-in-progress in these areas, and two discussion sessions:

- Domain specific languages for accelerators;
- Verification techniques for GPU-accelerated software.

Due to the "correctness" aspect of this seminar, there was significant overlap of interest with a full week seminar on *Formal Verification of Distributed Algorithms* running in parallel. To take advantage of this overlap a joint session was organised, featuring a talk on verification of GPU kernels by Alastair Donaldson, Imperial College London (on behalf of the *Correct and Efficient Accelerator Programming* seminar) and a talk on GPU-accelerated runtime verification by Borzoo Bonakdarpour, University of Waterloo, on behalf of the *Formal Verification of Distributed Algorithms* seminar.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Weak Memory Models: A Tutorial

*Jade Alglave (University College London, GB)*

In this talk I presented several behaviours observed on current processors such as Intel x86, IBM Power and ARM. These behaviours demonstrate that one cannot assume Sequential Consistency (SC) to model executions of concurrent programs.

I also explained which synchronisation one should use to enforce an SC behaviour on these examples. I concluded with an excerpt of the PostgreSQL database software which features two idioms that do not behave in an SC manner if not synchronised properly.

### 3.2 Estimating the WCET of GPU-Accelerated Applications using Hybrid Analysis

*Adam Betts (Imperial College London, GB)*

**Joint work of** Betts, Adam; Donaldson, Alastair F.

Gaining good performance from applications running on GPUs remains very challenging. One approach is to generate several kernel variants and then choose the best among this set using performance analysis techniques. Most approaches to performance analysis focus on average-case behaviour, but this sometimes yields ties between several kernels. In this talk, we present work performed in the context of the CARP project which focuses on estimating an outlier execution time, namely the Worst-Case Execution Time (WCET), in order to break ties and to estimate performance from a worst-case perspective. The technique we present is based on a mixture of dynamic and static analyses, which operates by collecting execution times of small program segments from measurements and then combining the data using a static program model. The talk focuses on extensions needed to incorporate concurrency into the timing model, in particular how to model the stalls experienced by warps (groups of threads on NVIDIA hardware) while other warps are in flight. We present our tool to estimate the WCET of GPU kernels, which is based on GPGPU-sim, and results when analysing several benchmarks from the CUDA SDK.

This work was supported by the EU FP7 STREP project CARP.

## 3.3 GPUVerify: A Verifier for GPU Kernels

*Alastair F. Donaldson (Imperial College London, GB)*

**License** 🅭 Creative Commons BY 3.0 Unported license
© Alastair F. Donaldson
**Joint work of** Betts, Adam; Chong, Nathan; Collingborne, Peter; Donaldson, Alastair F.; Ketema, Jeroen; Kyshtymov, Egor; Qadeer, Shaz; Thomson, Paul;
**Main reference** A. Betts, N. Chong, A.F. Donaldson, S. Qadeer, P. Thomson, "GPUVerify: a Verifier for GPU Kernels," in Proc. of the 27th Annual ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'12), pp. 113–132, ACM, 2012.
**URL** http://dx.doi.org/10.1145/2398857.2384625

The motivation for general-purpose computing on using graphics processing units (GPUs) is *performance*, thus GPU programmers work hard to write highly-optimised low-level kernels in OpenCL and CUDA. It is hard to write such optimised code correctly, thus there is wide scope for bugs, especially data races, in GPU kernels.

In this talk I presented GPUVerify, a technique and tool for verifying race-freedom of GPU kernels. GPUVerify performs analysis of GPU kernels by reducing the verification task to that of verifying a sequential program. This is achieved by exploiting (a) the fact that data race-freedom is a *pairwise* property, allowing a reduction to an arbitrary pair of threads, and (b) the observation that as long as data races are not tolerated, it is sufficient to consider a single, canonical schedule between pairs of barriers. Combined with source-level predicated execution, this allows a GPU kernel to be transformed into a sequential program that models round-robin execution of a pair of threads, equipped with instrumentation to perform data race detection. Proving data race freedom then amounts to proving correctness of this sequential program; in GPUVerify this is achieved by reusing the Boogie verification system.

During the talk I gave a demo of GPUVerify, and also illustrated manually how the automatic translation into Boogie is achieved.

This work was supported by the EU FP7 STREP project CARP.

## 3.4 Bulk Synchronous Streaming Model for the MPPA-256 Manycore Processor

*Benoît Dupont de Dinechin (Kalray – Orsay, FR)*

**License** 🅭 Creative Commons BY 3.0 Unported license
© Benoît Dupont de Dinechin

The Kalray MPPA-256 is an integrated manycore processor manufactured in 28nm CMOS technology that consumes about 10W for 230GFLOPS at 400MHz. Its 256 data processing cores and 32 system cores are distributed across 16 shared-memory clusters and 4 I/O subsystems, themselves connected by two networks-on-chip (NoCs). Each Kalray core implements a general-purpose Very Long Instruction Word (VLIW) architecture with 32-bit addresses, a 32-bit/64-bit floating-point unit, and a memory management unit.

This talk explains the motivations and the directions for the development of a streaming programming model for the MPPA-256 processor. Like the IBM Cell/BE or the Intel SCC, the Kalray MPPA-256 architecture is based on clusters of general-purpose cores that share a local memory, where remote memory accesses require explicit communication. By comparison, GP-GPU architectures allow direct access to the global memory and hide the resulting latency with massive hardware multithreading. On-going port of OpenCL to the MPPA-256 processor may only reach limited performances and gives up run-time predictability, as the global memory has to be emulated by software with a Distributed Shared Memory (DSM) technique.

The alternative we propose is to develop a stream-oriented programming model called 'Bulk Synchronous Streaming' (BSS), by adapting the classic Bulk Synchronous Parallel (BSP) model. The BSP model belongs to the family of symmetric parallel programming models for distributed memory supercomputers, like Cray SHMEM and Co-Array Fortran. The adaptations envisioned for the BSS include: maintaining the global data objects in the DDR memory, instead of distributing them across the local memories; enabling execution of BSP-like programs with a number of processing images larger than the number of clusters, by streaming their execution onto the available clusters; extending the precise BSP performance model to the BSS model. The result can be characterized as a generalized vector execution model, since the global data updates are not visible until after the superstep synchronizations.

## 3.5 Analysis of Shared Memory and Message Passing Parallel Programs

*Ganesh L. Gopalakrishnan (University of Utah, US)*

In this tutorial, a general introduction to shared memory and distributed memory programming is provided. It is important to have well-structured concurrent systems so that their debugging becomes easier. After a discussion of the general nature of large-scale computational frameworks such as Utah's Uintah system, the discussion shifts to the specifics of GPU programming. The University of Utah GPU formal correctness checking tool GKLEE. Various issues pertaining to the correctness of GPU programs are discussed including race detection, resource utilization checking using symbolic methods, and finally memory consistency issues and floating-point accuracy. A discussion of message passing verification around the ISP tool then follows. The tutorial ends with a discussion of how to make tangible impact by lowing expectations in a productive way: pick problems such as determinism and reproducibility if overall correctness is too difficult to achieve, or to even state clearly.

## 3.6 Compilation Techniques for Accelerators Tutorial

*Sebastian Hack (Universität des Saarlandes, DE)*

I gave an overview over four compilation techniques that are relevant to accelerators.

Type-based vectorization uses the type system to guide the compiler's vectorization choices. This way, SIMD programming can be done in a portable yet efficient way because the programmer is relieved of using intrinsics or other half-hearted language extensions.

Another approach uses abstract interpretation to decide which parts of a kernel have to be vectorized or not. This is important for accelerators that use explicit SIMD instruction sets. On such machines, retaining scalar computations is essential to mediate the overhead of vectorization.

The polyhedral model uses polyhedra to represent loop nests of a certain kind. Using integer linear programming techniques, many existing loop transformations can be rephrased to the problem of finding an affine schedule of a dependence graph annotated with dependence polyhedra and iteration space polyhedra.

Finally, I briefly spoke about the rapid development of domain-specific languages (DSLs) using language virtualization. Here, one embeds the DSL in a host language by hijacking its syntactic analysis. On top of that a custom compiler infrastructure is being built which then generates code for accelerators.

## 3.7 Accelerator Programming Models

*Lee Howes (AMD – Sunnyvale, US)*

**Main reference** B.R. Gaster, L. Howes, "Can GPGPU Programming be Liberated from the Data Parallel Bottleneck?" IEEE Computer, 45(8):42–52, IEEE, 2012.
**URL** http://dx.doi.org/10.1109/MC.2012.257

Accelerator devices have become increasingly capable over time and will continue to do so. With increasing flexibility of scheduling, shared virtual memory across SoCs, device context switching and more there is a much wider range of programming models that such devices will be able to support. In this talk we discuss some of the limitations of current accelerator programming models that have arisen due to limitations in hardware capabilities or early design decisions, and some ways they may evolve to become more flexible. If time allows we will also look at some of the current leading edge models to give a broader view of what is currently available.

## 3.8 Specification and Verification of GPGPU Programs using Permission-Based Separation Logic

*Marieke Huisman (University of Twente, NL)*

**Joint work of** Huisman, Marieke; Mihelčić, Matej

Graphics Processing Units (GPUs) are increasingly used for general-purpose applications because of their low price, energy efficiency and computing power. Considering the importance of GPU applications, it is vital that the behaviour of GPU programs can be specified and proven correct formally. This talk presents our ideas how to verify GPU programs written in OpenCL, a platform-independent low-level programming language. Our verification approach is modular, based on permission-based separation logic. We present the main ingredients of our logic, and illustrate its use on several example kernels. We show in particular how the logic is used to prove data-race- freedom and functional correctness of GPU applications.

This work was supported by the EU FP7 STREP project CARP.

## 3.9 Interleaving and Lock-Step Semantics for Analysis and Verification of GPU Kernels

*Jeroen Ketema (Imperial College London, GB)*

**Joint work of** Collingbourne, Peter; Donaldson, Alastair F.; Ketema, Jeroen; Qadeer, Shaz
**Main reference** P. Collingbourne, A.F. Donaldson, J. Ketema, S. Qadeer, "Interleaving and Lock-Step Semantics for Analysis and Verification of GPU Kernels," in Proc. of the 22nd European Symp. on Programming (ESOP 2013), LNCS, Vol. 7792, pp. 270–289, Springer, 2013.
**URL** http://dx.doi.org/10.1007/978-3-642-37036-6_16

In this talk I present a semantics of GPU kernels the parallel programs that run on Graphics Processing Units (GPUs). We provide a novel lock-step execution semantics for GPU kernels represented by arbitrary reducible control flow graphs and compare this semantics with a traditional interleaving semantics. We show for terminating kernels that either both semantics compute identical results or both behave erroneously.

The result induces a method that allows GPU kernels with arbitrary reducible control flow graphs to be verified via transformation to a sequential program that employs predicated execution. We implemented this method in the GPUVerify tool and experimentally evaluated it by comparing the tool with the previous version of the tool based on a similar method for structured programs, i.e., where control is organised using if and while statements.

This work was supported by the EU FP7 STREP project CARP.

## 3.10 On the Correctness of the SIMT Execution Model of GPUs

*Alexander Knapp (Universität Augsburg, DE)*

**Joint work of** Knapp, Alexander; Habermaier, Axel; Ernst, Gidon
**Main reference** A. Habermaier, A. Knapp, "On the Correctness of the SIMT Execution Model of GPUs," in Proc. of the 21st European Symposium on Programming Languages and Systems (ESOP'12), LNCS, Vol. 7211, pp. 316–335, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-28869-2_16

GPUs use a single instruction, multiple threads (SIMT) execution model that executes batches of threads in lockstep. If the control flow of threads within the same batch diverges, the different execution paths are scheduled sequentially; once the control flows reconverge, all threads are executed in lockstep again. Several thread batching mechanisms have been proposed, albeit without establishing their semantic validity or their scheduling properties. To increase the level of confidence in the correctness of GPU-accelerated programs, we formalize the SIMT execution model for a stack-based reconvergence mechanism in an operational semantics and prove its correctness by constructing a simulation between the SIMT semantics and a standard interleaved multi-thread semantics. We discuss an implementation of the semantics in the K framework and a formalization of the correctness proof in the theorem prover KIV. We also demonstrate that the SIMT execution model produces unfair schedules in some cases.

## 3.11 Using early CARP technology to implement BLAS on the Mali GPU series

*Alexey Kravets (ARM Ltd. – Cambridge, GB)*

**Joint work of** Kravets, Alexey; van Haastregt, Sven; Lokhmotov, Anton

We have demonstrated how early CARP technology can implement BLAS (Basic Linear Algebra Subprograms) on accelerators. We presented a high-level (DSL) description of a BLAS subroutine and PENCIL (Platform-Neutral Compute Intermediate Language) code for this subroutine. Finally we showed how OpenCL code could be generated through the DSL-PENCIL workflow.

This work was supported by the EU FP7 STREP project CARP.

## 3.12 Accelerator architectures and programming

*Anton Lokhmotov (ARM Ltd. – Cambridge, GB)*

Special-purpose processors can outperform general-purpose processors by orders of magnitude, importantly, in terms of energy efficiency as well as execution speed. This talk overviews the key architectural techniques used in parallel programmable accelerators such as GPUs: vector processing and fine-grained multithreading, and challenges associated with correct and efficient accelerator programming.

## 3.13 Efficient Code Generation using Algorithmic Skeletons and Algorithmic Species

*Cedric Nugteren (TU Eindhoven, NL)*

**Joint work of** Nugteren, Cedric; Corporaal, Henk
**Main reference** C. Nugteren, P. Custers, H. Corporaal, "Algorithmic species: A classification of affine loop nests for parallel programming," ACM Transactions on Architecture and Code Optimization (TACO), 9(4), 40 pp., ACM, 2013.
**URL** http://dx.doi.org/10.1145/2400682.2400699

We presented a technique to fully automatically generate efficient and readable code for parallel processors. We base our approach on skeleton-based compilation and 'algorithmic species', an algorithm classification of program code. We use a tool to automatically annotate C code with species information where possible. The annotated program code is subsequently fed into the skeleton-based source-to-source compiler 'Bones', which generates OpenMP, OpenCL or CUDA code. This results in a unique approach, integrating a skeleton-based compiler for the first time into an automated compilation flow. We demonstrated the benefits of our approach using the PolyBench suite by presenting average speed-ups of 1.4x and 1.6x for CUDA kernel code compared to PPCG and Par4All, two state-of-the-art polyhedral compilers.

## 3.14 Formal Analysis of GPU Programs with Atomics via Conflict-Directed Delay-Bounding

*Zvonimir Rakamarić (University of Utah, US)*

**Joint work of** Chiang, Wei-Fan; Gopalakrishnan, Ganesh; Li, Guodong; Rakamarić, Zvonimir
**Main reference** W.-F. Chiang, G. Gopalakrishnan, G. Li, Z. Rakamaric, "Formal Analysis of GPU Programs with Atomics via Conflict-Directed Delay-Bounding," in Proc. of the 5th NASA Formal Methods Symposium (NFM'13), LNCS, Vol. 7871, pp. 213–228, Springer, 2013.
**URL** http://dx.doi.org/10.1007/978-3-642-38088-4_15

GPU based computing has made significant strides in recent years. Unfortunately, GPU program optimizations can introduce subtle concurrency errors, and so incisive formal bug-hunting methods are essential. This paper presents a new formal bug-hunting method for GPU programs that combine barriers and atomics. We present an algorithm called Conflict-directed Delay-bounded scheduling algorithm (CD) that exploits the occurrence of conflicts among atomic synchronization commands to trigger the generation of alternate schedules; these alternate schedules are executed in a delay-bounded manner. We formally describe CD, and present two correctness checking methods, one based on final state comparison, and the other on user assertions. We evaluate our implementation on realistic GPU benchmarks, with encouraging results.

## 3.15 Code Generation for GPU Accelerators in the Domain of Image Preprocessing

*Oliver Reiche (Universität Erlangen-Nürnberg, DE)*

**Joint work of** Reiche, Oliver; Membarth, Richard; Hannig, Frank; Teich, Jürgen

This talk presented the Heterogeneous Image Processing Acceleration (HIPAcc) Framework that allows automatic code generation for algorithms from the domain of image preprocessing on GPU accelerators. By decoupling the algorithm from its schedule in a domain-specific language (DSL), efficient code can be generated that leverages the computational power of such accelerators. The decoupling allows to map the algorithm to the deep memory hierarchy found in today's GPUs based on domain knowledge and an architecture model of the target machine. Based on the same algorithm description, tailored code variants can be generated for different target architectures, improving programmer productivity significantly. The generated low-level CUDA, OpenCL and Renderscript codes allow to achieve competitive performance on GPU accelerators from NVIDIA, AMD and ARM compared to hand written codes while preserving high productivity and portability.

## 3.16 Compositional Analysis of Concurrent Timed Systems with Horn Clauses and Interpolants (work in progress)

*Philipp Rümmer (Uppsala University, SE)*

**Joint work of** Hojjat, Hossein; Kuncak, Viktor; Rümmer, Philipp; Subotic, Pavle; Yi, Wang

Timed automata are a well established theory for modelling and verifying real-time systems, with many applications both in industrial and academic context. Although model checking of timed automata has been studied extensively during the last two decades, scalability of tools for analysing timed automata remains a concern, in particular when applied to instances of industrial size. When verifying networks of (concurrent) timed automata, the size of the combined state space can be a limiting factor. In this paper we present an interpolation-based predicate abstraction framework which encodes timed automata as sets of Horn clauses, with the help of Owicki-Gries and Rely-Guarantee encoding schemes.

## 3.17 Performance Portability Investigations for OpenCL

*Ana Lucia Varbanescu (TU Delft, NL)*

**Joint work of** Fang, Jianbin; Shen, Jie; Varbanescu, Ana Lucia

Multi- and many-core processors have become ubiquitous, as we — as software designers and developers — find them in all modern devices, from mobile phones to desktops, servers, and supercomputers. Their goal is to accelerate software applications, sometimes with additional constraints (like low power consumption or real-time guarantees).

To program these platforms, the users often use native, hardware-centric programming models, where applications are mapped directly in a platform-friendly form and mapped in unintuitive ways on the platform. We argue that accelerator programming needs tools that can raise the programmability (where programmability is a combination of performance, portability, and productivity). One such tool is OpenCL, a programming model aiming to tackle the problem of accelerator programming in a standardized way: it exposes a virtual computing platform to allow for functional portability, uses platform-specific backends to allow for high performance, and it provides a higher-level model of computation to help productivity.

In this talk, we present studies for three different performance portability aspects of OpenCL.

First, we compare the performance of OpenCL to that of alternative languages. When comparing OpenCL agains CUDA, we find that the results are very similar: 8 out of 12 applications have less than 10% difference in performance when programmed in the two different models. The remaining applications exhibit either compiler or programmer corner-cases, which we are able to solve and therefore match the performance of the two languages. When comparing OpenCL with OpenMP for multi-core CPUs, we expect to find much larger differences in performance. This is not always the case. In fact, about a third of the benchmarks we have used (self-made and from the Rodinia benchmark) perform better in OpenCL, the others being similar or better in OpenMP. In other words, OpenCL codes can perform well on multi-core CPUs. We are also able to show that when the performance diverges significantly in favor of OpenMP, the causes are in the GPU-friendly optimizations that have been applied to the OpenCL code. Finally, we show how GPU OpenCL codes can be ported systematically to CPUs, thus gaining a parallel implementation with definite performance gain, only at the cost of systematic, pseudo-standard code changes. Specifically, the GPU-friendly optimizations that need undoing are: extreme fine-grained tasks, memory access patterns for coalescing, local memory usage, and vectorization. We show that these changes can be done in the form of code specialization, and do not affect the core of the 'naive' implementation. Thus, we conclude that OpenCL can be performance portable from GPUs to CPUs, and the degree of this portability depends on code specialization.

Second, we zoom in on one of the performance gaps between the CPU and GPU versions of OpenCL: local memory. In theory, using the OpenCL local memory for GPUs is considered an optimization, while using it on the CPUs is considered a mistake. However, in our extensive experimental work, we have seen a large variation in the behavior of applications that use local memory. Therefore, we considered a microbenchmarking approach, where we tested a whole set (over 40!) memory access patterns and their impact on the performance of local memory for GPUs and CPUs. The results are stored into a performance database. Given that we are able to formalize the memory access patters, we are then able to query this database and find out the predicted impact of using local memory for various applications, and decide whether applying this optimization will pay off or not. We conclude that such an approach is a viable alternative for performance prediction for local memory behavior.

Third, we present our contribution in combining the strength of OpenCL on both GPUs and CPUs: Glinda. Our Glinda framework allows (pseudo-)automated workload distribution of imbalanced applications on heterogeneous platforms, thus combining (only when needed) the execution of the same application on both CPUs and GPUs, concurrently. We briefly discuss the modules of Glinda, focusing more on the workload characterization and the auto-tuning, and show our results on acoustic ray tracing for fly-over noise. We conclude that OpenCL's functional and (partial) performance portability are both instrumental for such an integrated solution.

## 3.18 Accelerating Algorithms for Biological Models and Probabilistic Model Checking

*Anton Wijs (TU Eindhoven, NL)*

**Joint work of** Bošnački, Dragan; Edelkamp, Stefan; Hilbers, Peter; Ligtenberg, Willem; Odenbrett, Max; Sulewski, Damian; Wijs, Anton

In this talk, I will discuss two applications using GPU computation capabilities from different domains. In the first application, biological networks resulting from genetic perturbation experiments are reduced in order to extract the vital information. We show that this boils down to computing the transitive reduction of weighted graphs. This can be performed ideally using GPUs. We explain how an extended version of the Floyd-Warshall algorithm can be parallelised effectively for the GPU. Through experimentation, we recorded speedups up to 92 times compared to a sequential implementation.

In the second application, we capitalize on the fact that most of the probabilistic model checking operations involve matrix-vector multiplication and solving systems of linear equations. Since linear algebraic operations can be implemented very efficiently on GPGPUs, the new parallel algorithms show considerable runtime improvements compared to their counterparts on standard architectures. We implemented our parallel algorithms on top of the probabilistic model checker PRISM. The prototype implementation was evaluated on several case studies in which we observed significant speedup over the standard CPU implementation of the tool.

**References**

1 D. Bošnački, S. Edelkamp, D. Sulewski and A.J. Wijs, *Parallel Probabilistic Model Checking on General Purpose Graphics Processors,* International Journal on Software Tools for Technology Transfer 13(1):21–35, (2011).

2 D. Bošnački, M.R. Odenbrett, A.J. Wijs, W.P.A. Ligtenberg and P.A.J. Hilbers, *Efficient Reconstruction of Biological Networks via Transitive Reduction on General Purpose Graphics Processors,*BMC Bioinformatics 13:281 (2012).

## 4 Discussion Sessions

### 4.1 Domain specific languages for accelerators

*Albert Cohen (ENS – Paris, FR)*

The discussion covered three main questions:

1. Understanding the rationale for using or designing DSLs in the context of hardware accelerators.
2. Listing the most successful frameworks and illustrations of the approach, sharing and collecting people's experience.
3. Digging into the design and implementation of DSLs, including language embedding, staging, debugging.

The participants came from different horizons and included end-users (applications), language and compiler designers, and hardware accelerator experts. The field, the motivations, and experiences appeared to be very diverse. But some fundamentals and general lessons could be identified, aiming to maximize the usability of hardware accelerators, and to simplify the design of programming and development flows.

### 4.2 Verification techniques for GPU-accelerated software

*Alastair F. Donaldson (Imperial College London, GB)*

The aim of this session, involving roughly half the seminar participants, was to brainstorm the key issues surrounding verification techniques for GPU-accelerated software, to gather opinions about: the most important kinds of bugs that occur when programming GPUs, how verification technology might help in *optimization* of GPU kernels, and what infrastructure is most appropriate to use in building analysis tools. In additional we discussed a wish list for future developments in this area.

The following is a rough summary of the discussion.

### Bugs affecting GPU programmers

- **Data races:** There was consensus that this is one of the most important classes of defect to be checked, with anecdotal evidence that engineers working for major GPU vendors also share this view. We discussed the problem of distinguishing erroneous data races from benign data races, and the idea of providing annotation facilities so that programmers could mark certain accesses as participating in benign races, to suppress false positives.
- **Bugs caused by compiler behaviour related to data races:** The discussion of benign races led to comments that even if the programmer intends a race to occur, this may lead to unpredictable behaviour from the compiler, since the semantics for racy code is in general not defined.
- **Barrier divergence:** It was suggested that this is not such a serious class of defect because it is usually fairly obvious when a barrier has been accidentally placed in non-uniform conditional code, but that nevertheless it is valuable to have techniques to check for this issue as barrier divergence can lead to deadlocks.
- **Bugs inherited from sequential programming:** Out-of-bounds accesses, accessing uninitialised memory, problems related to pointer type casting, etc., are just as problematic in GPU kernels as in sequential programs.
- **Floating point issues:** We discussed a bit the problems that can arise when discrete decisions are based on floating point computations; how this can lead to subtle problems and may be affected due to re-association of operations as a result of thread nondeterminism. We also debated the extent to which there is standardised floating point precision across architectures, and noted the inclusion of implementation-defined 'fast' transcendental functions in OpenCL.

### Using verification techniques to help with code optimisation

The idea of using verification and symbolic execution to check for issues like memory bank conflicts in CUDA kernels has already been explored through the PUG and GKLEE tools at the University of Utah. We discussed the possibility for using verification to identify redundant synchronisation caused by cautious programming; i.e., figuring out that a particular barrier can be removed without inducing data races.

### Infrastructure for building analysis tools

We discussed the advantages and disadvantages of analysis techniques that work at low-level program representations. Working at a 'close-to-the-metal' representation level for architecture $X$ may help to find subtle bugs related to $X$, including bugs introduced by compilers, but may not be meaningful for other architectures. Working at a level such as LLVM IR allows the reuse of existing infrastructure (in this case CLANG/LLVM), but means that analysis is being performed with respect to one particular compiler framework, whereas a kernel may be compiled using a different framework. Source code is a common ancestor for all architectures, but does not take into account compiler effects at all; additionally source level analysis carries significant front-end implementation overhead which is avoided when working at the level of IR. A pragmatic compromise is to perform analysis on the "lowest-level common ancestor" representation. It was suggested that SPIR might be a suitable candidate.

The Ocelot framework[1] was also discussed as a promising base on which to build analysis tools.

### Areas for future investigation

The following is a list of ideas and questions related to future work that were raised during the discussion.

- It is likely that future architectures and programming models will allow an increased level of dynamic memory allocation, meaning that techniques for heap analysis, such as separation logic, might play a larger role in GPU kernel verification.
- Relatedly, we discussed the role of linked data structures in kernel programming, including tricks programmers use to copy linked data structures into GPU global memory without re-wiring pointers; this seems like a challenging target for static analysis.
- Better techniques for analysis of floating point properties were mentioned several times; there is also wide demand for this in sequential program analysis.
- Can verification techniques be used for program repair, e.g., to suggest where to place barriers to eliminate data races, or to suggest where to move barriers to avoid barrier divergence?
- Most techniques so far have focussed on lightweight properties such as race-freedom. It would be interesting to look at functional verification, either of kernels themselves, or of applications that use kernels. In the latter case, modular verification would require specifications to be written describing the effects of a kernel.
- Race analysis of data-dependent kernels (to reason about, e.g., accesses of the form $A[B[e]]$ where $A$ and $B$ are shared memory arrays) may require some level of functional verification, even if this is not the ultimate goal.
- It could be interesting and useful to provide a mechanism for programmers to annotate barriers with light-weight assertions, to specify the intended use of a barrier.
- Can static analysis techniques help in understanding performance issues of GPU kernels, such as the possible gains from offloading, or the worst case execution time of a kernel?
- Techniques for cross-checking GPU and CPU implementations of an algorithm could be very useful; this has been explored to some extent through the KLEE-CL project.
- Can abduction be used to infer preconditions for kernels that are sufficient for ensuring race-freedom (or other properties)?

[1] https://code.google.com/p/gpuocelot/

## Participants

- Jade Alglave
University College London, GB
- Adam Betts
Imperial College London, GB
- Albert Cohen
ENS – Paris, FR
- Christian Dehnert
RWTH Aachen, DE
- Dino Distefano
Queen Mary University of London, GB
- Alastair F. Donaldson
Imperial College London, GB
- Jeremy Dubreil
Monoidics Ltd. – London, GB
- Benoit Dupont de Dinechin
Kalray – Orsay, FR
- Ganesh L. Gopalakrishnan
University of Utah, US
- Sebastian Hack
Universität des Saarlandes, DE
- Lee Howes
AMD – Sunnyvale, US
- Marieke Huisman
University of Twente, NL
- Christina Jansen
RWTH Aachen, DE
- Joost-Pieter Katoen
RWTH Aachen, DE
- Jeroen Ketema
Imperial College London, GB
- Alexander Knapp
Universität Augsburg, DE
- Georgia Kouveli
ARM Ltd. – Cambridge, GB
- Alexey Kravets
ARM Ltd. – Cambridge, GB
- Anton Lokhmotov
ARM Ltd. – Cambridge, GB
- Roland Meyer
TU Kaiserslautern, DE
- Cedric Nugteren
TU Eindhoven, NL
- Zvonimir Rakamaric
University of Utah, US
- Oliver Reiche
Univ. Erlangen-Nürnberg, DE
- Philipp Rümmer
Uppsala University, SE
- Ana Lucia Varbanescu
TU Delft, NL
- Sven Verdoolaege
INRIA, FR
- Jules Villard
University College London, GB
- Heike Wehrheim
Universität Paderborn, DE
- Anton Wijs
TU Eindhoven, NL
- Marina Zaharieva-Stojanovski
University of Twente, NL
- Dong Ping Zhang
AMD – Sunnyvale, US

Report from Dagstuhl Seminar 13151

# Drawing Graphs and Maps with Curves

**Edited by**

**Stephen Kobourov[1], Martin Nöllenburg[2], and Monique Teillaud[3]**

1 **University of Arizona – Tucson, US,** kobourov@cs.arizona.edu
2 **KIT – Karlsruhe Institute of Technology, DE,** noellenburg@kit.edu
3 **INRIA Sophia Antipolis – Méditerranée, FR,** Monique.Teillaud@inria.fr

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13151 "Drawing Graphs and Maps with Curves". The seminar brought together 34 researchers from different areas such as graph drawing, information visualization, computational geometry, and cartography. During the seminar we started with seven overview talks on the use of curves in the different communities represented in the seminar. Abstracts of these talks are collected in this report. Six working groups formed around open research problems related to the seminar topic and we report about their findings. Finally, the seminar was accompanied by the art exhibition *Bending Reality: Where Arc and Science Meet* with 40 exhibits contributed by the seminar participants.

**Seminar** 07.–12. April, 2013 – www.dagstuhl.de/13151
**1998 ACM Subject Classification** I.3.5 Computational Geometry and Object Modeling, G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems
**Keywords and phrases** graph drawing, information visualization, computational cartography, computational geometry
**Digital Object Identifier** 10.4230/DagRep.3.4.34
**Edited in cooperation with** Benjamin Niedermann

## 1 Executive Summary

*Stephen Kobourov*
*Martin Nöllenburg*
*Monique Teillaud*

Graphs and networks, maps and schematic map representations are frequently used in many fields of science, humanities and the arts. The need for effective visualization and aesthetically pleasing design is attested by the numerous conferences and symposia on related topics, and a history that is several centuries old. From Mercator's maps dating to the 1500's, to interactive services such as Google Earth, geography and cartography have generated and solved many theoretical and practical problems in displaying spatial data effectively and efficiently. From Euler's visualization of the bridges of Königsberg in the 1700's, to Facebook's social networks, graph drawing has also proven a fertile area for theoretical and practical work. More recent is the notion of highly schematized maps and graphs, with the classic examples of statistical value-by-area cartograms by Raisz and Henry Beck's London Tube map, both dating back to the 1930's.

A key challenge in graph and cartographic visualization is designing cognitively useful spatial mappings of the underlying data that allow people to intuitively understand the

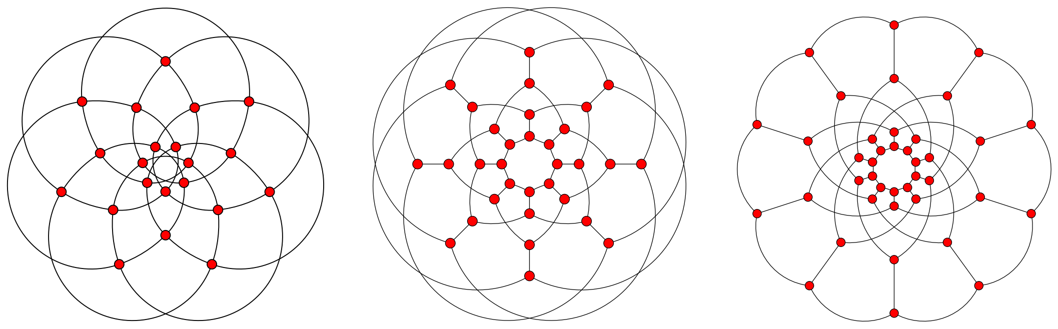


**Figure 1** Lombardi graph drawings [1]: Brinkman graph, Dyck graph, and $F_{40}$ (dodecahedron double cover).

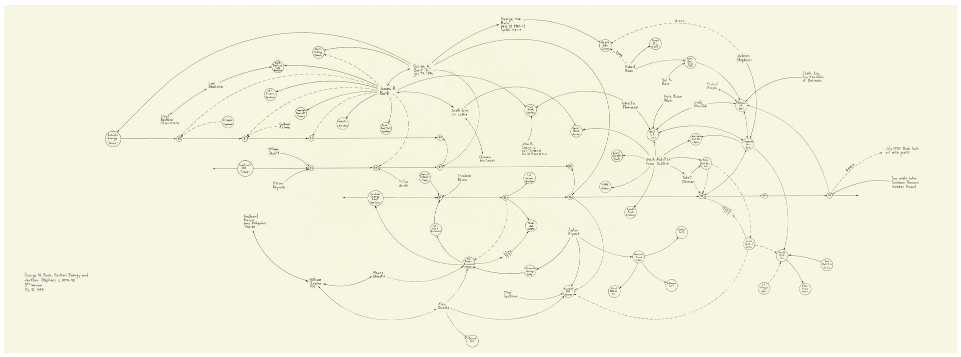


**Figure 2** One of Mark Lombardi's pieces: *George W. Bush, Harken Energy, and Jackson Stevens ca. 1979–90*, 1999. Graphite on paper, 20 × 44 inches. Courtesy Pierogi Gallery and Donald Lombardi. Photo credit: John Berens.

displayed information. Such work draws on the intellectual history of several traditions, including information visualization, human-computer interaction, psychology, cognitive science, graphic design, cartography, and art. The synthesis of relevant ideas from these fields with new techniques can lead to new and better visualizations to help us keep pace with the torrents of data confronting us.

Although a great deal is known, both in theory and in practice, about drawing graphs and maps with straight-line segments, there are few corresponding results about circular-arc drawings in particular, and curve drawings in general. The use of circular arcs in place of straight-line segments opens a new chapter in drawing graphs and maps from both theoretical and practical points of view. Specifically, we are interested in the interplay between practical requirements of drawing with curves, arising in cartography and GIS, and theoretical results in computational geometry and graph drawing. Such work is motivated by perception research which indicates that representing paths with smooth geodesic trajectories aids in comprehension, as well as by the aesthetic appeal of smooth curves; see Fig. 1 and Fig. 2.

## Aims of the Seminar

The main goal of this seminar was to bring together researchers with interests in drawing graphs and maps with curves coming from information visualization, psychology, cognitive science, human-computer interaction, graph drawing, computational geometry, cartography, and GIS. It follows in a tradition of several previous similarly structured Dagstuhl seminars on graph drawing and map visualization. From April 7th to April 12th a group of 34 junior and senior researchers from eight different countries gathered in Dagstuhl. Being a *small* seminar with a target participation of 30 persons, the seminar was fully booked, which shows that this seemingly narrow topic still raises a lot of interest in the different communities. We all came together to discuss open research questions and engage in new collaborations around visualizations that replace straight lines with circular arcs and curves. This topic opens a great deal of theoretical and practical possibilities and with this in mind, the specific aims of the Dagstuhl seminar were:

- To learn about the state of the art of the use of curves in the different research areas. We invited a small number of survey lectures to define a common ground for interdisciplinary work.
- To organize an exhibition of art and visual designs on the common theme of curves contributed by participants and artists, and use this to stimulate discussion.
- To identify specific theoretical and practical open problems that need to be solved in order to make it possible to draw graphs and maps with circular arcs and curves.
- To form smaller working groups around some of the identified problems and to initiate a collaborative research process for finding answers and solutions to these problems.
- To report about the progress made in the working groups in a plenary session for getting feedback and further input from members of the other groups.
- To continue the joint research efforts beyond the seminar week and eventually publish those results.

## Achievements of the Seminar

The achievements in the seminar were numerous and varied. The subsequent chapters of this report summarize the more important ones.

1. On Monday and Tuesday, we enjoyed seven survey lectures; see Section 3 for the abstracts. David Eppstein opened with a broad overview of the use of curves in visualization of graphs and networks. Günter Rote talked about algorithms for approximating polygonal curves by simpler curves and sequences of biarcs. Sylvain Lazard illustrated connections with algebra and geometry when dealing with curves. Jo Wood surveyed the use of curves in cartography and information visualization. Helen Purchase discussed perception theories and empirical studies on the use of curves in visualization, and Maxwell Roberts discussed the question whether curvilinear metro maps have cognitive benefits over traditional straight-line schematic maps. Finally, Monique Teillaud and Michael Hemmer overviewed the history of the open source project CGAL, the Computational Geometry Algorithms Library, and then discussed specific CGAL packages that are relevant for drawing circular arcs and smooth algebraic curves. Beyond the survey and review talks, we also heard a presentation by Wouter Meulemans about the use of curved schematization of geometric shapes, where the results were obtained via a user study of the participants in the seminar.
2. We also had two short impromptu presentations and software demos. In particular, Günter Rote presented an ipelet to transform polygons into splines in the drawing editor ipe. Jan-Henrik Haunert reported about work in progress and showed a demo on morphing polygonal lines so that edge lengths and angles behave as consistently as possible over time.

3. A number of relevant open problems were formulated early in the seminar and six working groups formed around some of the problems. The groups then worked by themselves, formalizing and solving their specific theoretical and practical challenges. Section 4 contains the working group reports summarizing the problem and sketching the current progress made by the groups. Below is a list of the working group topics.
   a. **Smooth Orthogonal Drawings:** What is the complexity of recognizing whether a given 4-planar graph admits a smooth orthogonal drawing of edge complexity 1?
   b. **Confluent Drawing:** What is the complexity of determining whether a given graph has a so-called *strict* confluent drawing?
   c. **Automated Evaluation of Metro Map Usability:** What are good, objective, quantifiable criteria by which curvilinear metro maps can be evaluated? Can such criteria be used so that linear maps can likewise be compared both with each other and also with curvilinear maps?
   d. **Universal Point Sets for Planar Graph Drawings with Circular Arcs:** What can be said about universal point sets for drawing planar graphs if curves are used instead of straight-line segments?
   e. **Labeling Curves with Curved Labels:** How can points on a smooth curve be labeled automatically using curved labels?
   f. **Graphs with Circular Arc Contact Representation:** Which graphs can be represented by contacts of circular arcs?

   The remaining open problems collected during the seminar are listed in Section 5.
4. We had an excellent exhibition entitled “Bending Reality: Where Arc and Science Meet”; see Section 6. This exhibition is the third one in a series of exhibitions that accompany Dagstuhl seminars where aesthetics and art are naturally part of the scientific topics. It was on display from April 8 to April 21, 2013. Moreover, for the first time in Dagstuhl history, this exhibition is made permanently available as a virtual exhibition that can be accessed at **http://www.dagstuhl.de/ueber-dagstuhl/kunst/13151**.

The last three days of the seminar were dedicated to working group efforts. Several of the groups kept their focus on the original problems as stated in the open problem session, while other groups modified and expanded the problems; see Section 4. On the last day of the seminar we heard progress reports from all groups. The results of two of the groups have recently been accepted to international conferences, and we are expecting further research publications to result directly from the seminar.

Arguably the best, and most-appreciated, feature of the seminar was the opportunity to engage in discussion and interactions with experts in various fields with shared passion about curves. The aforementioned exhibition “Bending Reality” helped make the topics of the seminar more visible and raised new questions. In summary, we regard the seminar as a great success. From the positive feedback that we got it is our impression that the participants enjoyed the unique scientific atmosphere at Schloss Dagstuhl and profited from the scientific program. We are grateful for having had the opportunity to organize this seminar and thank the scientific, administrative, and technical staff at Schloss Dagstuhl. We also thank Benjamin Niedermann for helping us to put this report together.

## References

1 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Lombardi drawings of graphs. *J. Graph Algorithms and Applications*, 16(1):85–108, 2012.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 A Brief History of Curves in Graph Drawing

*David Eppstein (University California – Irvine)*

"It is not the right angle that attracts me, nor the straight line, hard and inflexible, created by man. What attracts me is the free and sensual curve—the curve that I find in the mountains of my country, in the sinuous course of its rivers, in the body of the beloved woman."

—Oscar Niemeyer [45]

#### 3.1.1 Early Directions

Hand-generated graph drawings have long used curves, independently of graph drawing research. Examples can be found, for instance, in the works of Listing [38] and Peterson [46]. Mark Lombardi (1951–2000) raised hand-made curved graph drawings to an art form with his diagrams of connections between the actors in global financial, criminal, and political conspiracies, or as he called them, *narrative structures* [31]. Hand-drawn graphs have also been the subject of graph drawing research: Plimmer et al. [48] describe techniques for the automated rearrangement of hand-drawn graphs, taking care to preserve the features that make these graphs look hand-drawn such as the curvature of their edges.

The earliest research on the mathematics and algorithmics of curved graph drawing concerned a drawing style that is now called an *arc diagram*. These are drawings in which the vertices are placed on a line, and the edges are drawn on one or more semicircles, either above or below the line. Drawings with these styles were first used by Saaty [54] and Nicholson [44] as a way to formalize and attempt to solve the problem of minimizing crossing numbers of graphs. Finding the minimum number of crossings of a graph in this style turns out to be NP-hard [41], although if crossing-free diagrams exist they may be found by solving a 2-satisfiability problem once the vertex ordering has been determined [17]. Djidjev and Vrt'o [10] and Cimikowski [8] developed heuristics to reduce the number of crossings in drawings of this type without guaranteeing to find the minimum. For *st*-planar graphs (and also for undirected planar graphs) it is always possible to find drawings in this style in which edges are drawn as smooth curves formed of at most two semicircles (oriented left to right in the *st*-planar case) with at most one crossing of the spine on which the vertices are placed [29, 3].

#### 3.1.2 Uses of Curvature

One use of curvature, in drawings with edges that are curved but that do not have inflection points, is to indicate directionality: if each edge is oriented in clockwise from source to destination, then this orientation will be unambiguously indicated by the curve of the edge without having to use other visual techniques to show it such as arrowheads or line thickness variation [22]. This style is a good fit for arc diagrams [49] but for other layout styles, user studies have shown that it can be confusing to readers [34].

Curved edges may also be used to help edges avoid obstacles that a straight edge would run into. This can already be seen in the work of Peterson [46], who used circular arcs to represent distinct edges connecting the same two vertices in a multigraph. In Sugiyama-style layered graph drawing [58], vertices are assigned to layers and edges are subdivided by dummy vertices so that each connection is between consecutive layers. The dummy vertices (which are used to route edges between the other vertices on each layer) may then be used to guide the placement of spline curves so that the edges may be drawn as smooth curves [28]. A simplified version of this spline calculation by Sander [55] was used by him to win the graph drawing contest at GD 1994 GD. Another obstacle-avoiding drawing style, by Dobkin et al. [11], involves placing the vertices of the drawing first, and then in a second stage of the algorithm routing the edges around the vertices. In their algorithm, each edge is routed first as a shortest obstacle-avoiding polyline, which is then smoothed by a spline curve, and finally the curves are adjusted locally to eliminate any intersections with obstacles caused by the smoothing step. There has also been much related work in motion planning on finding smooth curves for a fixed obstacle-avoiding route; see e.g. Lutterkort and Peters [40].

The two ideas of curvature indicating directionality and curvature to avoid obstacles were combined in PivotGraph [60], a system for displaying graphs whose vertices have other numerical or categorical data associated with them. The vertices are placed into a grid using this additional data as Cartesian coordinates, but this placement would lead to many edges that pass directly through vertices if drawn as straight lines. By using curves (oriented clockwise by directionality) they avoid these bad edge-vertex intersections as well as showing the orientation of the edges graphically. In another visualization system, NodeXL [57], a “polar” layout places vertices on concentric circles; curved edges are used to connect vertices on consecutive circles without passing through the inner circle, cf. [1].

### 3.1.3 Focus + Context

Sarkar and Brown [56] suggested interactive *fisheye views* of graphs that could be used to simultaneously zoom in on a point of interest and showing its surrounding context. The Poincaré model of hyperbolic geometry (with edges drawn as circular arcs) automatically has this effect [37] and has the additional advantage that there is a natural way to morph from one focus to another, “maintaining the mental map”. Mohar [42] proves a version of Fáry’s theorem (stating that the existence of non-crossing drawings in which each edge follows a geodesic path) for graphs in the hyperbolic plane or on surfaces of negative curvature.

In later work on highlighting points of interest, Wong et al. [62] use edge curvature to bend edges locally away from a part of a drawing without distorting the vertex placements in the drawing. A related technique called *edge plucking* allows interactive user control of local bending of bundles of edges [61].

### 3.1.4 Edge Complexity

Much research in graph drawing has focused on drawing styles with angular bends but low *curve complexity* (bends per edge). However, graph drawing researchers have long known that bends can be replaced by smoothed curves [16]. Bekos et al. [3] formalize the *edge complexity* of graphs drawn with piecewise-circular-arc smooth curves, as the maximum number of arcs and straight segments per edge. They observe that edge complexity is always within a constant factor of bend complexity but that in many cases it is actually possible to achieve lower edge complexity than bend complexity.

Goodrich and Wagner [30] achieve low edge complexity in a planar graph drawing. They modify the (straight line) planar drawing algorithm of Fraysseix et al. [25] by surrounding each vertex with a protected region of radius proportional to its degree, and placing equally spaced “ports” on the boundary of this region. Spline curves through the ports have constant edge complexity, near-optimal angular resolution, and do not cross. Similar ideas have also been used by Cheng et al. [6] and Duncan et al. [12] for drawings with piecewise circular edges, again achieving high angular resolution.

### 3.1.5 Force-directed Graph Drawing (Spring Systems)

Force-directed methods have long been a mainstay of practical graph drawing. These methods use forces (which may be visualized as springs) to attract adjacent pairs of vertices and repel other pairs. Despite being somewhat slow they are very flexible, as they allow the implementer great freedom of choice in modifying the system to add other forces beyond these basic attractive and repulsive ones. Force-directed methods began with Tutte [59], who showed that springs can automatically generate planar straight line drawings of planar graphs. Other important early research in this area (also using straight edges) was done by Eades [15], Kamada and Kawai [36], and Fruchterman and Reingold [26].

The combination of curves and forces was made by Brandes and Wagner [5], as part of a system for drawing graphs that represent train systems. In these graphs, vertices represent train stations and edges connect consecutive stops on the same train route. The vertex placement is fixed by the geographic position of the stations, and in many cases involves sets of nearly-collinear stations spaced out along the same train line. However, the edges representing express trains skip some of these vertices, representing local train stations, and (if drawn as straight) would overlap the positions of these stations. The solution of Brandes and Wagner [5] is to use force-directed methods, with forces on the control points of splines, to bend these edges outwards. In another application to train systems, Fink et al. [23] use force-directed drawing to schematize train system maps, by replacing paths of degree-two vertices by spline curves. Bending outwards can also be used in 3d to separate edges of geographic graphs from the Earth’s surface [43].

For arbitrary graphs, Finkel and Tamassia [24] place a new vertex in the middle of each edge of a given graph, apply force-directed layout, and then use the new vertices as spline control points. They report that this gives significant improvements in angular resolution and modest improvements in crossings compared to straight line drawings. Similar ideas were used by Chernobelskiy et al [7] to spread out the edges in drawings with circular-arc edges.

### 3.1.6 Edge Bundling

*Edge bundling* is a technique that, as initially developed, was used to simplify drawings of large graphs with a hierarchically clustered vertex structure. This technique groups edges that connect the same two clusters (at some level of the hierarchy) into “bundles” drawn as nearly-parallel curves, making the set of edges both visually distinctive and more compact than if they were all drawn separately. This idea was introduced by Holten [32] based on a physical analogy to electrical wiring bundles; it is also closely related to flow maps for numerical geographic data [47], and since the initial work on this style there have been hundreds of successor papers.

Some of the many variations on bundling that have been considered include:

- Non-hierarchical bundling by modeling edges as springs that attract each other [33].
- A circular vertex layout, with unbundled edges outside the circle chosen to minimize crossings, and with edges grouped into bundles inside the circle using a heuristic that attempts to minimize the total amount of ink used for the drawing [27].

- Edge bundling in Sugiyama-style layered drawing [50].
- Forbidding crossings within the edges of a bundle, and routing the edges of each bundle on parallel tracks resembling metro maps, so that individual edges are easier to follow [4].

For a taxonomy of bundling-related curved edge techniques see Riche et al. [53].

### 3.1.7 Confluent Drawing

Although visually similar to bundled drawing, and often co-cited with it, confluent drawing [9] is semantically very different. In confluent drawing, one represents a graph using *train tracks* (sets of smooth curves meeting at points called *junctions*) rather than drawing the edges as individual curves. Two vertices are adjacent in the underlying graph if and only if they can be connected by a smooth curve through the curves and junctions of the drawing. Thus, each curve carries a set of edges, similar to bundling, but in an unambiguous way.

The graphs with crossing-free confluent drawings form a strict superset of the planar graphs, and include for example the interval graphs [9] and distance-hereditary graphs [20, 35]. A partially ordered set has an upward-planar confluent *Hasse diagram* if and only if its order dimension is at most two [19].

Confluent edge routing (allowing crossings between pairs of confluent tracks) has been combined with Sugiyama-style layered drawing, by finding complete bipartite subgraphs within the sets of edges that connect consecutive layers and replacing these subgraphs by confluent tracks [21]; this style was used for a winning entry in the 2003 graph drawing contest. Confluence has also been used as a way to achieve axis-parallel edges for high-degree planar graphs [52].

### 3.1.8 Lombardi Drawing

Inspired by the art of Mark Lombardi, graph drawing researchers have defined *Lombardi drawing* to be a very strict layout style in which edges must be drawn as circular arcs, meeting at equal angles at each vertex. This allows plane trees to be drawn in balloon style (with their subtrees drawn recursively in disks surrounding the root node) with polynomial area, in contrast to straight line drawing styles for which drawing plane trees with equally spaced edges at each vertex may sometimes require exponential area [14].

Lombardi drawing may also be used for regular graphs and symmetric graphs [13], planar graphs with maximum degree three [18], and some other special cases [39]. However, not every graph has a Lombardi drawing, causing researchers in this area to resort to force-based approximations [7] or multi-arc relaxations [12]. User studies on the aesthetics and usability of force-based Lombardi-like drawings have had mixed results [63, 51] perhaps indicating that additional constraints (such as restricting edges to arcs that cover less than half of their circles) are necessary to make this style more effective.

A related result is that every 3-connected 4-regular planar graph may be represented as the graph of an arrangement of circles [2]; however, the arcs of this representation will not in general be equally spaced around each vertex.

### 3.1.9 Conclusions

Curves have been used very widely in graph drawing: almost every method of graph drawing that has been studied, has been studied with curves. There are still many remaining technical challenges (for instance on the edge complexity needed for 1-planar drawings, or on the existence of outerplanar Lombardi drawings) but the biggest future challenge for curved drawing is more general, and still has not been completely achieved: designing methods for curved drawing that are general (applicable to every graph and not just specialized graph classes), usable (as measured in user studies), and consistently beautiful.

## References

1 Christian Bachmaier, Hedi Buchner, Michael Forster, and Seok-Hee Hong. Crossing minimization in extended level drawings of graphs. *Discrete Appl. Math.*, 158(3):159–179, 2010.

2 Michael A. Bekos and Chrysanthi N. Raftopoulou. Circle-representations of simple 4-regular planar graphs. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 138–149. Springer, 2013.

3 Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov, and Antonios Symvonis. Smooth orthogonal layouts. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 150–161. Springer, 2013.

4 Sergey Bereg, Alexander E. Holroyd, Lev Nachmanson, and Sergey Pupyrev. Edge routing with ordered bundles. In *Graph Drawing 2011*, volume 7034 of *LNCS*, pages 136–147. Springer, 2012.

5 Ulrik Brandes and Dorothea Wagner. Using graph layout to visualize train interconnection data. *J. Graph Algorithms and Applications*, 4(3):135–155, 2000.

6 Christine C. Cheng, Christian A. Duncan, Michael T. Goodrich, and Stephen G. Kobourov. Drawing planar graphs with circular arcs. *Discrete Comput. Geom.*, 25(3):405–418, 2001.

7 Roman Chernobelskiy, Kathryn I. Cunningham, Michael T. Goodrich, Stephen G. Kobourov, and Lowell Trott. Force-directed Lombardi-style graph drawing. In *Graph Drawing 2011*, volume 7034 of *LNCS*, pages 320–331. Springer, 2012.

8 Robert Cimikowski. Algorithms for the fixed linear crossing number problem. *Discrete Appl. Math.*, 122(1-3):93–115, 2002.

9 Matthew T. Dickerson, David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Confluent drawings: visualizing non-planar diagrams in a planar way. In *Graph Drawing 2003*, volume 2912 of *LNCS*, pages 1–12. Springer, 2004.

10 Hristo Djidjev and Imrich Vrt'o. An improved lower bound for crossing numbers. In *Graph Drawing 2001*, volume 2265 of *LNCS*, pages 96–101. Springer, 2002.

11 David P. Dobkin, Emden R. Gansner, Eleftherios Koutsofios, and Stephen C. North. Implementing a general-purpose edge router. In *Graph Drawing 1997*, volume 1353 of *LNCS*, pages 262–271. Springer, 1997.

12 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Maarten Löffler. Planar and poly-arc Lombardi drawings. In *Graph Drawing 2011*, volume 7034 of *LNCS*, pages 308–319. Springer, 2012.

13 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Lombardi drawings of graphs. *J. Graph Algorithms and Applications*, 16(1):85–108, 2012.

14 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Drawing trees with perfect angular resolution and polynomial area. *Discrete Comput. Geom.*, 49(2):183–194, 2013.

15 Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42(11):149–160, 1984.

16 Peter Eades and Roberto Tamassia. Algorithms For Drawing Graphs: An Annotated Bibliography. Technical Report CS-89-09, Computer Science Dept., Brown University, 1989.

17 Alon Efrat, Cesim Erten, and Stephen G. Kobourov. Fixed-location circular arc drawing of planar graphs. *J. Graph Algorithms and Applications*, 11(1):145–164, 2007.

18 David Eppstein. Planar Lombardi drawings for subcubic graphs. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 126–137. Springer, 2013.

**19** David Eppstein and Joseph A. Simons. Confluent Hasse diagrams. In *Graph Drawing 2011*, volume 7034 of *LNCS*, pages 2–13. Springer, 2012.

**20** David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Delta-confluent drawings. In *Graph Drawing 2005*, volume 3843 of *LNCS*, pages 165–176. Springer, 2006.

**21** David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Confluent layered drawings. *Algorithmica*, 47(4):439–452, 2007.

**22** Jean-Daniel Fekete, David Wang, Niem Dang, Aleks Aris, and Catherine Plaisant. Overlaying graph links on treemaps. In *IEEE Symp. on Information Visualization, Poster Compendium*, pages 82–83, 2003.

**23** Martin Fink, Herman Haverkort, Martin Nöllenburg, Maxwell Roberts, Julian Schuhmann, and Alexander Wolff. Drawing metro maps using Bézier curves. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 463–474. Springer, 2013.

**24** Benjamin Finkel and Roberto Tamassia. Curvilinear graph drawing using the force-directed method. In *Graph Drawing 2004*, volume 3383 of *LNCS*, pages 448–453. Springer, 2005.

**25** Hubert de Fraysseix, János Pach, and Richard Pollack. Small sets supporting Fary embeddings of planar graphs. In *20th ACM Symp. on Theory of Computing*, pages 426–433, 1988.

**26** Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

**27** Emden R. Gansner and Yehuda Koren. Improved circular layouts. In *Graph Drawing 2006*, volume 4372 of *LNCS*, pages 386–398. Springer, 2007.

**28** Emden R. Gansner, Stephen C. North, and Kiem-Phong Vo. DAG—a program that draws directed graphs. *Software: Practice and Experience*, 18(11):1047–1062, 1988.

**29** Francesco Giordano, Giuseppe Liotta, Tamara Mchedlidze, and Antonios Symvonis. Computing upward topological book embeddings of upward planar digraphs. In *Proc. Int. Symp. Algorithms and Computation (ISAAC 2007)*, volume 4835 of *LNCS*, pages 172–183. Springer, 2007.

**30** Michael T. Goodrich and Christopher G. Wagner. A framework for drawing planar graphs with curves and polylines. *J. Algorithms*, 37(2):399–421, 2000.

**31** Robert Hobbs. *Mark Lombardi: Global Networks*. Independent Curators, 2003.

**32** Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Visualization and Computer Graphics*, 12(5):741–748, 2006.

**33** Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009.

**34** Danny Holten and Jarke J. van Wijk. A user study on visualizing directed edges in graphs. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 2299–2308, 2009.

**35** Peter Hui, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Train tracks and confluent drawings. *Algorithmica*, 47(4):465–479, 2007.

**36** Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

**37** John Lamping and Ramana Rao. Laying out and visualizing large trees using a hyperbolic space. In *Proc. 7th ACM Symp. on User Interface Software and Technology*, pages 13–14, 1994.

**38** Johann Benedikt Listing. *Vorstudien zur Topologie*. Vandenhoeck und Ruprecht, 1848.

**39** Maarten Löffler and Martin Nöllenburg. Planar Lombardi drawings of outerpaths. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 561–562. Springer, 2013.

**40** David Lutterkort and Jörg Peters. Smooth paths in a polygonal channel. In *Proc. 15th ACM Symp. on Computational Geometry*, pages 316–321, 1999.

41 Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computing*, 39(1):124–127, 1990.

42 Bojan Mohar. Drawing graphs in the hyperbolic plane. In *Graph Drawing 1999*, volume 1731 of *LNCS*, pages 127–136. Springer, 1999.

43 Tamara Munzner, Eric Hoffman, K. Claffy, and Bill Fenner. Visualizing the global topology of the MBone. In *IEEE Symp. on Information Visualization*, pages 85–92, 1996.

44 T. A. J. Nicholson. Permutation procedure for minimising the number of crossings in a network. *Proc. IEE*, 115:21–26, 1968.

45 Oscar Niemeyer. *The Curves of Time: the memoirs of Oscar Niemeyer.* Phaidon, 2000. As quoted at http://en.wikiquote.org/wiki/Oscar_Niemeyer.

46 Julius Peterson. Die Theorie der regulären Graphen. *Acta Mathematica*, 15(1):193–220, 1891.

47 Doantam Phan, Ling Xiao, Ron Yeh, Pat Hanrahan, and Terry Winograd. Flow map layout. In *IEEE Symp. on Information Visualization*, pages 219–224, 2005.

48 Beryl Plimmer, Helen C. Purchase, Hong Yul Yang, Laura Laycock, and James Milburn. Preserving the hand-drawn appearance of graphs. In *Int. Worksh. Visual Languages and Computing*, 2009.

49 A. Johannes Pretorius and Jarke J. van Wijk. Bridging the semantic gap: Visualizing transition graphs with user-defined diagrams. *IEEE Computer Graphics and Applications*, 27(5):58–66, 2007.

50 Sergey Pupyrev, Lev Nachmanson, and Michael Kaufmann. Improving layered graph layouts with edge bundling. In *Graph Drawing 2010*, volume 6502 of *LNCS*, pages 329–340. Springer, 2011.

51 Helen C. Purchase, John Hamer, Martin Nöllenburg, and Stephen G. Kobourov. On the usability of Lombardi graph drawings. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 451–462. Springer, 2013.

52 Gianluca Quercini and Massimo Ancona. Confluent drawing algorithms using rectangular dualization. In *Graph Drawing 2010*, volume 6502 of *LNCS*, pages 341–352. Springer, 2011.

53 Nathalie Henry Riche, Tim Dwyer, Bongshin Lee, and Sheelagh Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proc. Int. Working Conf. on Advanced Visual Interfaces (AVI '12)*, pages 506–513, 2012.

54 Thomas L. Saaty. The minimum number of intersections in complete graphs. *Proc. National Academy of Sciences*, 52:688–690, 1964.

55 Georg Sander. Graph layout through the VCG tool. In *Graph Drawing 1994*, volume 894 of *LNCS*, pages 194–205. Springer, 1995.

56 Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 83–91, 1992.

57 Marc A. Smith, Ben Shneiderman, Natasa Milic-Frayling, Eduarda Mendes Rodrigues, Vladimir Barash, Cody Dunne, Tony Capone, Adam Perer, and Eric Gleave. Analyzing (social media) networks with NodeXL. In *Proc. 4th Int. Conf. Communities and Technologies*, pages 255–264, 2009.

58 Kozo Sugiyama, Shôjirô Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-11 (2):109–125, 1981.

59 William T. Tutte. How to draw a graph. *Proc. London Math. Society*, 13(52):743–768, 1963.

60 Martin Wattenberg. Visual exploration of multivariate graphs. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 811–819, 2006.

61 Nelson Wong and Sheelagh Carpendale. Interactive poster: Using edge plucking for interactive graph exploration. In *IEEE Symp. on Information Visualization*, 2005.

62 Nelson Wong, Sheelagh Carpendale, and Saul Greenberg. EdgeLens: an interactive method for managing edge congestion in graphs. In *IEEE Symp. on Information Visualization*, pages 51–58, 2003.

63 Kai Xu. A user study on curved edges in graph visualization. *IEEE Trans. Visualization and Computer Graphics*, 18(12):2449–2556, 2012.

## 3.2 Algorithms for Curve Approximation

*Günter Rote (FU Berlin)*

**Joint work of** Scott Drysdale, Günter Rote, Astrid Sturm
**Main reference** R.L.S. Drysdale, G. Rote, A. Sturm, "Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs," Computational Geometry, Theory and Applications 41 (2008), 31-47.
**URL** http://dx.doi.org/10.1016/j.comgeo.2007.10.009

This is a survey of algorithms for approximating a curve by a simpler curve, mostly by a polygon with few edges. In the last part I also mention an algorithm of Drysdale, Rote, and Sturm [1] for smooth (tangent-continuous) approximation by biarcs. I discuss the problem definition, the variations and different objectives, and the pitfalls. I also survey the classical algorithms of Douglas and Peucker; Imai and Iri; and of Chan and Chin.

### References

1 R. L. Scot Drysdale, Günter Rote und Astrid Sturm: Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs. *Computational Geometry, Theory and Applications* 41 (2008), 31–47. doi:10.1016/j.comgeo.2007.10.009

## 3.3 Algebraic Curves in Computational Geometry

*Sylvain Lazard (INRIA Grand Est – Nancy, FRA)*

**Joint work of** Yacine Bouzidi, Sylvain Lazard, Marc Pouget, Fabrice Rouillier

I will survey some recent results on the problem of drawing, in a certified way, algebraic curves in the plane given by their implicit equation. I will focus on the problem of computing the topology of such curves and their critical points. The talk will mostly be centered on the presentation of the standard tools and main classes of methods for this problem. I will also present some recent results (joint with Y. Bouzidi, M. Pouget and F. Rouillier) on the problem of computing the critical points of the curve or, more precisely, on the problem of solving bivariate algebraic systems by means of rational univariate representations. I will show that computing such representations can be done in $O(d^8 + d^\tau)$ bit operations modulo polylogarithmic factors, where $d$ is the degree of the input curve and $\tau$ is the maximum bitsize of its coefficients. This decreases by a factor $d^2$ the best known complexity for this problem. I will finally present some experiments and comparisons between state-of-the-art software for computing the topology of plane algebraic curves.

## 3.4 Curved Lines in Cartography and Information Visualization

*Jo Wood (City University – London)*

Calling on examples from the history of cartography and information visualization this talk shows where curved lines have been used to carry information in visual depiction. Examples range from terrain analysis, through sketchy rendering, to flows of people, bicycles and money. The role of metaphor in visual depiction in both maps and information visualization is considered and related to the geometric properties of curved lines.

## 3.5 Some Brief Notes on Perceptual Theories — in Relation to Empirical Studies

*Helen C. Purchase (University of Glasgow)*

In trying to understand how people understand information visualisations, we can use two sources; empirical data collected through experiments, or theories of perception. Of course, the two are necessarily intertwined: data suggests theory, theory is validated by data. Conducting experiments is costly and the results are not easily generalisable outside the experimental parameters. What we would like to be able to do is predict the effectiveness of a visualisation prior to its use, and existing theories of perception and cognition can help us do this.

This presentation introduces some common theories of perception that can help in pre-experiment assessment of visualisation effectiveness. It takes as a starting point a paper by Peebles (2004) where the effect of a commonly-known visual illusion was investigated in data plots used for comparing the performance of local councils with respect to several issues (e.g. education, leisure, housing). This example demonstrates how existing perceptual theories can be used to predict the effectiveness of a visualisation – and how, in this case, the predictions were confirmed through an empirical study (and as a result, a better visualisation was proposed).

Several perceptual theories are introduced in this presentation, with focus on the Gestalt theories, and visual acuity. These theories are then related to graph drawings, showing, for example, that the existance of non-symmetric edges can overcome the preception of symmetric nodes. The relative visual priority of visual features like colour, shape, texture etc., is demonstrated through the process of 'pop-out'.

Finally, an experiment comparing the effectiveness of graphs with straight line edges vs curved edges is presented as an example of a study explicitly addressing the limits of visual acuity. The results, surprisingly, showed better performance with the straight-line graph, suggesting that visual acuity is not as problematic in graph-reading tasks as might be expected.

## 3.6 Do We Need Curvilinear Metro Maps?

*Maxwell J. Roberts (University of Essex)*

The topics of cognitive load and cognitive capacity are related to metro map usage and, in relation to this, the following optimisation criteria are identified: simplicity; coherence; balance; harmony, and topographicity. It is suggested that as long as these criteria are fulfilled, the design ruses do not matter, and that in certain circumstances it may be necessary to design a curvilinear metro map. Three types of such map are identified: freeform Bézier, concentric circles, and Lombardi, each with strengths and weaknesses likely to suit networks with different qualities. Generally, curvilinear maps may be particularly suited to dense, interconnected networks with complex line trajectories, where linear maps have high angular density and poor coherence (e.g., Paris, Tokyo).

## 3.7 Curves in CGAL

*Michael Hemmer (TU Braunschweig) and Monique Teillaud (INRIA Sophia Antipolis – Méditerranée)*

CGAL, the Computational Geometry Algorithms Library, followed the evolution of Computational Geometry, progressively offering more and more curved objects and functionality on them.

The talk first quickly overviews the history of the CGAL open source project, the development process and the contents of the library. It recalls how CGAL solves robustness issues, following the so-called exact geometric computing paradigm pioneered by Chee Yap.

Then packages in CGAL that should be relevant within the context of graph drawing with curves are examined. In particular, support for circles, spheres, triangulations and Voronoi diagrams involving circular arcs, planar arrangements of arbitrary algebraic curves and the algebraic kernel are discussed in more detail.

## 3.8 Curved Schematization – User Study Results

*Wouter Meulemans (TU Eindhoven, NL)*

**Joint work of** Arthur van Goethem, Wouter Meulemans, Bettina Speckmann, Jo Wood

We invited the participants of the seminar to also participate in an online user study related to curved schematization. Curved schematization is the task of obtaining a low-complexity representation of a detailed shape using curves. In particular, we use circular arcs. We had three hypotheses: curved schematization is better than straight-line schematization in terms of (1) aesthetics, (2) visual simplicity, and (3) recognizability. To verify these hypotheses in the user study, we had a single algorithm generate schematizations according to four different styles. One style admitted only straight-line edges; the other three styles used circular arcs. The latter differentiated themselves in the degree of "curviness" (measured as central angle) they strive for. The user study itself consisted of three tasks, one for each hypothesis. For each task, the participant was given a random set of questions about either aesthetics, simplicity, or recognizability. The study had 303 participants. We concluded that the three curved styles were preferred over the straight-line style, but there was no clear preference for any particular curved style. Hence we accept hypothesis (1). In terms of simplicity, the straight-line style and the curved style with low curviness (i.e. using arcs that are almost a straight line) were the best. Therefore we reject hypothesis (2). As for recognizability, we observed that the curved styles performed better than the straight-line style in a range of "medium complexity" (approximately 10 arcs). Thus we accept hypothesis (3).

# 4 Working Groups

## 4.1 Smooth Orthogonal Drawings

*Michael A. Bekos (National TU – Athens)*
*Martin Gronemann (Universität Köln)*
*Sergey Pupyrev (University of Arizona – Tucson)*

### 4.1.1 Introduction

*Smooth orthogonal drawings* were recently introduced in the graph drawing literature as a response to the problem of smoothening orthogonal drawings (of graphs of maximum degree 4), by "replacing" bends with smooth circular arcs. It is expected that such drawings are of improved readability and more aesthetic appeal, since it is widely accepted that bends interrupt the eye movement and require sharp changes of direction. Smooth orthogonal drawings follow the paradigms of traditional orthogonal drawings, in which each vertex is drawn as a point on the plane and each edge is drawn as an alternating sequence of axis-aligned line-segments. However, they also allow for circular arc-segments. We are particularly interested in providing theoretical guarantees about graphs that admit smooth orthogonal drawings of *edge complexity* 1, where the edge complexity of a drawing is given by the maximum number of segments forming any of its edge. Formally, we say that a graph has *smooth complexity* $k$ if it admits a smooth orthogonal drawing of edge complexity at most $k$.

The work of Bekos, Kaufmann, Kobourov and Symvonis [1] was the first on the problem of creating smooth orthogonal drawings of 4-planar graphs. They presented an infinite class of 4-regular planar graphs generated from the octahedron graph that do not admit smooth orthogonal drawings of edge complexity 1. They also proved that biconnected 4-planar graphs admit drawings of smooth complexity 3. For triconnected 3-planar graphs and Hamiltonian 3-planar graphs, they gave algorithms to produce smooth orthogonal drawings of edge complexity 1. Finally, they showed that there exist graphs whose smooth complexity-1 drawings needs exponential drawing area (assuming a quite restricted drawing model that requires fixed embedding and fixed ports).

### 4.1.2 Results

As already stated, not all 4-planar graphs admit smooth orthogonal drawings of smooth complexity 1. What if crossings are allowed in the resulting drawing? The following theorem answers this question in a positive way.

▶ **Theorem 1.** *Any graph of maximum degree 4 admits a (not necessary planar) drawing of smooth complexity 1.*

The input graph is augmented such that it is 4-regular. Based on the work of Peterson [2], a cycle cover decomposition is used to create a book-embedding-like drawing. The cycle cover decomposition guarantees that the port constraints are respected. However, the resulting drawing is not necessarily planar.

Bekos et al. [1] have proven that triconnected 3-planar graphs admit drawings of smooth complexity 1. Their approach is based on canonical ordering, hence, the requirement for triconnectivity. We are able to strengthen this result by using the *BC (Block-Cutpoint)* and *SPQR*-tree data structures to decompose the graph into its bi- and triconnected components, respectively.

▶ **Theorem 2.** *Any 3-planar graph admits a (planar) drawing of smooth complexity 1.*

The proof is constructive and provides an algorithm that exploits the special structure of the BC- and SPQR-trees on 3-planar graphs. The approach uses a slightly modified version of the algorithm in Bekos et al. [1] to deal with the triconnected components. Similar to Eppstein [3] who uses these properties to create Lombardi-style drawings of sub cubic graphs, we are able to maintain smooth complexity 1 while reassembling the components into one drawing. However, the approach relies heavily on the degree restriction, hence, does not easily extend to higher degree.

### 4.1.3 Open Problems & Ongoing Work

While working on the aforementioned problems during the seminar, it seemed tempting to use a book embedding style drawing as a basis for a layout. Especially when considering the Kandinsky model, i.e. one drops the port constraints, the two seem to be related. Besides the results of the seminar, a list of open problems has been compiled.

- What is the complexity of recognizing whether a given 4-planar graph admits a smooth orthogonal drawing of edge complexity 1?
- Is it possible to determine a universal point set so that any 4-planar graph can be embedded on it with a certain smooth complexity (e.g., SC 2 or SC 3)?
- The algorithm that we currently have for drawing any graph of maximum degree 4 with smooth complexity 1 does not take care of the crossings that arise. So, crossing minimization for non planar graphs is of importance.

#### References

1 Michael A. Bekos, Michael Kaufmann, Stephen G. Kobourov, and Antonios Symvonis. Smooth orthogonal layouts. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 150–161. Springer, 2013.
2 Julius Peterson. Die Theorie der regulären Graphen. *Acta Mathematica*, 15(1):193–220, 1891.
3 David Eppstein. Planar Lombardi drawings for subcubic graphs. In *Graph Drawing 2012*, volume 7704 of *LNCS*, pages 126–137. Springer, 2013.

## 4.2 Confluent Drawing

*David Eppstein (University of California, Irvine, US)*
*Danny Holten (SynerScope BV)*
*Maarten Löffler (Utrecht University)*
*Martin Nöllenburg (KIT – Karlsruhe Institute of Technology)*
*Bettina Speckmann (TU Eindhoven)*
*Kevin Verbeek (University of California – Santa Barbara)*

Confluent drawing [1] is a style of graph drawing in which edges are not drawn explicitly; instead vertex adjacency is indicated by the existence of a smooth path through a system of arcs and junctions that resemble train tracks. These types of drawings allow even very dense graphs, such as complete graphs and complete bipartite graphs, to be drawn in a planar way. We introduce a subclass of confluent drawings, which we call *strict* confluent drawings. Strict confluent drawings are confluent drawings with the additional restrictions that between any pair of vertices there can be *at most one* smooth path, and there cannot be any paths from a vertex to itself. Figure 3 illustrates the forbidden configurations. We believe that these restrictions may make strict drawings easier to read, by reducing the ambiguity caused by the existence of multiple paths between vertices.

Our results are as follows:

- It is NP-complete to determine whether a given graph has a strict confluent drawing. To prove this, we reduce from planar 3-SAT and we use the fact that $K_4$ has exactly two strict confluent drawings.
- For a given graph, with a given cyclic ordering of its $n$ vertices, there is an $O(n^2)$-time algorithm to find an *outerplanar* strict confluent drawing, if it exists: this is a drawing in a (topological) disk, with the vertices in the given order on the boundary of the disk (see Figure 4). We define a *canonical diagram* which is a unique representation of all possible outerplanar strict confluent drawings, where some of the faces are marked as cliques. We compute a canonical diagram by first computing its junctions, and then computing the arcs between them. From a canonical diagram we can easily derive an outerplanar strict confluent drawing.
- When a graph has an outerplanar strict confluent drawing, an algorithm based on circle packing can construct a layout of the drawing in which every arc is drawn using at most two circular arcs (again, see Figure 4).

The full version of this abstract will appear (under the title Strict Confluent Drawing) in the proceedings of the 21st International Symposium on Graph Drawing 2013. The most pressing problem left open by this research is to recognize the graphs that have outerplanar strict confluent drawings, without imposing a fixed vertex order. Can we recognize these graphs in polynomial time?

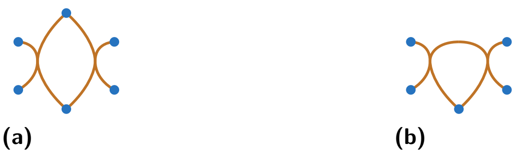


**Figure 3** (a) A drawing with a duplicate path. (b) A drawing with a self-loop.

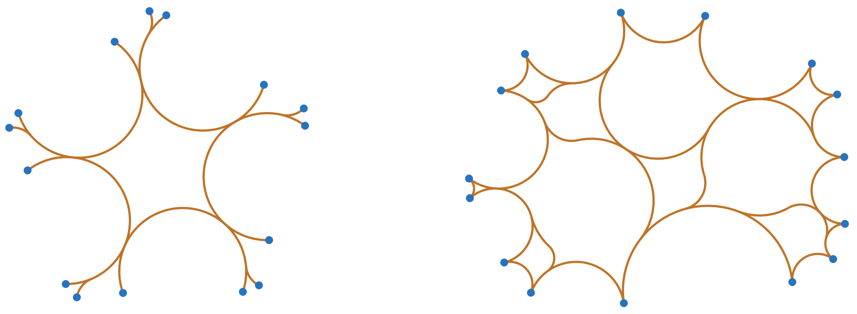


**Figure 4** Two outerplanar strict confluent drawings.

## References

1 Matthew T. Dickerson, David Eppstein, Michael T. Goodrich, and Jeremy Yu Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *Journal of Graph Algorithms and Applications*, 9(1):31–52, 2005.

## 4.3 Automated Evaluation of Metro Map Usability

*Michael Hemmer (TU Braunschweig)*
*Wouter Meulemans (TU Eindhoven)*
*Lev Nachmanson (Microsoft Corp. – Redmond)*
*Helen Purchase (University of Glasgow)*
*Andreas Reimer (Universität Heidelberg)*
*Max Roberts (University of Essex)*
*Günter Rote (FU Berlin)*
*Kai Xu (Middlesex University)*

The problem addressed by this workgroup concerns devising objective quantifiable criteria by which curvilinear metro maps can be evaluated and, subsequently, broadening the applicability of the criteria so that linear maps can likewise be compared both with each other and also with curvilinear maps. In the first instance, the intention was to predict journey planning time differences between maps, as per those revealed in usability studies (for example, comparing freeform Bézier maps with official octolinear designs, for London a curvilinear design is equivalent to the official map, but for Paris a curvilinear design is 50% faster for journey planning than the official map). Ultimately, the development of successful evaluation routines would curtail the need for usability studies.

The methodology adopted was to identify criteria for good design of curvilinear maps, based on exemplars of extreme difference in simplification to line trajectories (a Madrid curvilinear map devised by the first author and one devised by the designers of the current tetralinear official map). From this, the following criteria for good design were identified and attempts made to quantify them independently of each other.

Lower order criteria concern the trajectories of individual lines in isolation of each other. These contribute to the overall simplicity of the design.

- Curve inflections (number and severity).
- Curvature (quantity of change, and severity of change).
- Symmetry of curvature (at Bézier control points).

Higher order criteria concerning relationships between line trajectories. These contribute to the overall coherence of the design.

- Parallel Bézier curves within defined fields.
- Few edge crossings.
- Good angular resolution at edge crossings, especially at line interchanges where the line trajectories are typically part-obscured by the addition of a symbol.

Higher order criteria concerning relationships between line trajectories and stations. These also contribute to the overall coherence of the design.

- Adequate distance between stations and edge crossings (a station too close to an edge crossing is the most frequent source of planning errors in usability studies).
- Continuity of station density (a balanced design should not have abrupt changes in station density in nearby regions of the map).
- Horizontal/vertical alignment of stations.

The first objective was to assemble criteria that would predict differences in basic usability between designs: the mean time necessary to plan a journey (station location time is another variable of interest in evaluating usability). Purely aesthetic criteria were not identified. Aesthetic judgement will be related to the usability criteria outlined above, but other aesthetic criteria (such as a preference for straight lines versus curves) may impact only upon user-acceptance of a design (an important consideration, albeit one that is vulnerable to individual differences in preference). Measures of topographical accuracy were also not implemented. These may affect user acceptance of a design, but their prime influence on usability is their potential to affect journey choices: whether an efficient or inefficient journey is planned. Any such algorithm should penalise a map more for distorted relative positions of nearby stations than distant stations. Finally, the relationship between label placement and usability was not considered. These are an important element of map usability, and future work will address criteria for effective placement.

### 4.3.1 Specific Criteria Quantifications

Quantified evaluations of lower order criteria for line curvature were fully implemented on an individual line-by-line basis, derived from plots of curvature versus distance. From this, aspects of poor design could easily be identified visually and numerically. The main challenge now is to weight the relative importance of these.

Parallelism and angular resolution were implemented together, the former on the basis of constancy of distance between lines within a defined distance of each other. Angular resolution was incorporated by not subjecting lines to a parallelism analysis if their crossing was greater than a pre-determined angle.

Station density plots were created which enabled hotspots and coldspots to be identified. Designs were penalised on the basis of distance between these. Station placement was evaluated by identifying x-co-ordinates and y-co-ordinates, and for each of these determining the number of aligned stations.

Scores for various criteria were obtained for a variety of curvilinear and linear Madrid maps (high scores are bad for all measures)

1. total inflections
2. overall extent of curvature (variation of curvature over the whole line)
3. symmetry of curvature (sum of the squares of the vertical jumps in curvature–ie: extreme changes in curvature–over the whole line)
4. "lack of parallelism" penalty (not yet normalised for number of lines)
5. number of edge crossings (not yet normalised for complexity = stations x lines)
6. spacing discontinuity of stations
7. vertical/horizontal mis-alignment of station symbols

Simplified curvilinear map: 1) 19; 2) 1.23; 3) 0.006; 4) 5.76; 5) 42; 6) 0.21; 7) 0.19
Tetralinear (rectilinear) map: 1) NYC; 2) NYC; 3) NYC; 4) NYC; 5) 46; 6) 0.26; 7) 0.11
Hexalinear map: 1) NYC; 2) NYC; 3) NYC; 4) NYC; 5) 47; 6) 0.19; 7) 0.16
Complex curvilinear map: 1) 80; 2) 4.02; 3) NYC; 4) 6.94; 5) 42; 6) 0.28; 7) 0.21
Concentric circles map: 1) NYC; 2) NYC; 3) NYC; 4) NYC; 5) 43; 6) 0.18; 7) 0.18
NYC = not yet computed

### 4.3.2 Future work

Having identified methods of quantification, the next step is to extend them so that relative predictions can be made for linear versus curvilinear maps, and to combine and weight them so that they make predictions that match usability study data. The latter are, unfortunately, relatively course: typically all that can be identified is a relative difference in journey planning time between pairs of maps. This can be mitigated by comparing as may different maps as possible using different design techniques from a variety of cities. Once available data has been modelled, this will then enable future usability studies to be identified so that predictions can be tested and refined. In addition to this, the availability of these criteria will enable particular defects with maps to be identified, and possibly resolved automatically via a simulated annealing process.

## 4.4 Universal Point Sets for Planar Graph Drawings with Circular Arcs

*Patrizio Angelini (Università di Roma III)*
*David Eppstein (University California – Irvine)*
*Fabrizio Frati (The University of Sydney)*
*Michael Kaufmann (Universität Tübingen)*
*Sylvain Lazard (INRIA Grand Est – Nancy)*
*Tamara Mchedlidze (KIT – Karlsruhe Institute of Technology)*
*Monique Teillaud (INRIA Sophia Antipolis – Méditerranée)*
*Alexander Wolff (Universität Würzburg)*

It is well known that every planar graph has a drawing on the plane where vertices are mapped to points of an $O(n) \times O(n)$ integer grid, and edges to straight-line segments connecting the corresponding points. To generalize this result, Mohar (according to Pach [1]) posed the following question: What is the smallest size $f(n)$ (as a function of $n$) of a point set $S$ such that every $n$-vertex planar graph $G$ admits a planar straight-line drawing in which the vertices of $G$ are mapped to points in $S$? Such a point set $S$ is called *universal point set*.

Despite more than twenty years of research efforts, the best known lower bound for the value of $f(n)$ is linear in $n$, while the best known upper bound is quadratic in $n$. The question is listed as problem #45 in the Open Problems Project [1].

Given the topic of our seminar, we are interested in understanding whether drawing edges as circular arcs rather than straight-line segments helps to answer the corresponding question. Our starting point is a result of Everett et al. [2] saying that, for any natural number $n$, there is a universal set of size $n$ if edges are drawn as one-bend polygonal chains.

It turns out that we can get the same result for drawings with circular arcs. More specifically, we prove the existence of a set $S$ of $n$ points on the parabolic arc $\mathcal{P} = \{(x, y) \colon x \geq 0, y = -x^2\}$ such that every $n$-vertex planar graph $G$ can be drawn such that its vertices are mapped to $S$ and its edges are mapped to pairwise non-crossing circular arcs connecting points in $S$. We start with a sequence of $n^2$ points, $q_0, \ldots, q_{n^2-1}$, on $\mathcal{P}$ such that $x_0 \geq 1$ and $x_i \geq 2x_{i-1}$ for $i = 1, \ldots, n^2 - 1$. For our universal point set $S$, we take the $n$ points $p_i = q_{ni}$ with $i = 0, \ldots, n - 1$. We call the points in $q_0, \ldots, q_{n^2-1}$ that are not in the universal point set *helper points*. In the same spirit as Everett et al. [2], we draw $G$ in two steps.

In the first step, we construct a *monotone topological book embedding* of $G$. This is a drawing of $G$ such that the vertices lie on a horizontal line, called *spine*, and the edges are represented by non-crossing curves, monotonically increasing in the direction of the spine. Di Giacomo et al. [3] have shown that every planar graph has a monotone topological book embedding where each edge crosses the spine exactly once and consists of two semi-circles, one below and one above the spine.

In the second step, we map the vertices of $G$ and the crossings with the spine to the points in $q_0, \ldots, q_{n^2-1}$ in the same order as they appear on the spine of the book embedding, so that the vertices of $G$ are mapped to points in $S$ and the crossings with the spine to helper points. Each edge of $G$ is then drawn as a circular arc that passes through two points of $S$ and a helper point between them. By exploring geometrical properties of circular arcs through three points of $q_0, \ldots, q_{n^2-1}$, we show that the constructed drawing is planar.

Our $n$-point universal point set uses an area of $2^{O(n^2)}$. Hence, it would be interesting to investigate whether there exist universal point sets of size $n$ (or $o(n^2)$) for circular arc drawings that fit in polynomial area.

The full version of this abstract will appear (under the same title) in *Proc. 25th Canadian Conference on Computational Geometry 2013*, see also http://hal.inria.fr/hal-00846953.

## References

1 Erik. D. Demaine, Joseph. S. B. Mitchell, and Joseph O'Rourke. The open problems project. Website, 2001. URL cs.smith.edu/˜orourke/TOPP, accessed May 5, 2012.

2 Hazel Everett, Sylvain Lazard, Giuseppe Liotta, and Stephen Wismath. Universal sets of $n$ points for one-bend drawings of planar graphs with $n$ vertices. *Discrete Comput. Geom.*, 43(2):272–288, 2010.

3 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Curve-constrained drawings of planar graphs. *Comput. Geom. Theory Appl.*, 30:1–23, 2005.

## 4.5 Labeling Curves with Curved Labels

*Jan-Henrik Haunert (Universität Würzburg)*
*Herman Haverkort (TU Eindhoven)*
*Benjamin Niedermann (KIT – Karlsruhe Institute of Technology)*
*Arlind Nocaj (Universität Konstanz)*
*Aidan Slingsby (City University – London)*
*Jo Wood (City University – London)*

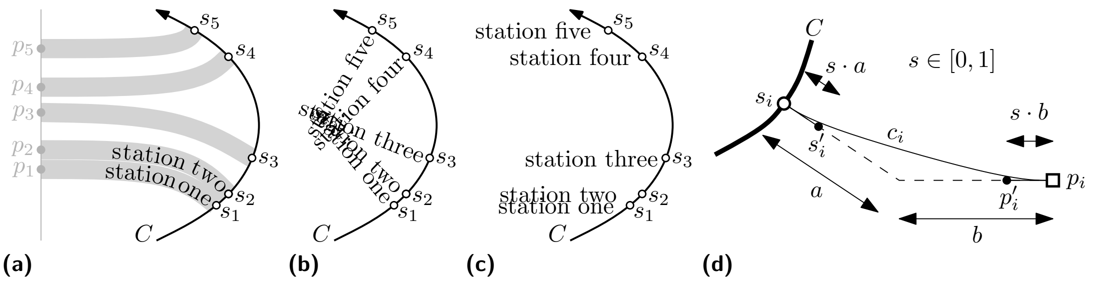


**Figure 5** (a)–(c) Different types of labels for the same metro line $C$. (a) Curved labels connecting stops of curve $C$ with ports to the left of $C$. (b) Using straight labels perpendicular to $C$ yields unavoidable clashes. (b) Straight labels can become cluttered when aligned horizontally. (c) Control points of a curved label.

Laying out metro maps automatically is a challenging question in research that in particular comprises finding an appropriate labeling of the metro lines and their stops automatically. We focus on the problem of labeling already embedded metro lines having a smooth curved shape. To the best of our knowledge we present the first considerations towards labeling smooth curved metro lines automatically. So far only labeling of metro lines represented by polygonal chains has been discussed in previous work, for example see [1, 2, 3, 4]. In order to simplify the problem of laying out metro maps with curves we started with only one metro line and assume that its shape is already given by a directed, smooth, non-self-intersecting curve $C$ in the plane, for example described by a Bézier curve. Further, the stops of the metro line are given by $n$ ordered points $s_1, \dots, s_n$ on $C$, which we hence call *stops*. Going along $C$ from its beginning to its end we assume that these stops appear in the order $s_1, \dots, s_n$. Depending on the specific problem formulation one may assume that the stops are not fixed on $C$, but that we need to find their concrete positions maintaining their order. For each stop $s_i$ we are further given a label that should be placed closely to it. We do not follow traditional map labeling abstracting from the given text by bounding boxes, but we use fat curves prescribing the shape of the labels; see Fig. 5b. We identified the following criteria, which motivated us to consider curved text.

- For aesthetic reasons the labels should integrate into the curved style of the metro map.
- More labels can be placed without intersections when considering curved labels; see Fig. 5a–5c.

Hence, we want to find directed fat curves $c_1, \dots, c_n$ of fixed width such that for each $c_i$, $1 \le i \le n$ it is true that 1. $c_i$ begins at $s_i$, 2. $c_i$ does not intersect $C$, and 3. $c_i$ does not intersect any other curve $c_j$, $1 \leq j \leq n$. We call these curves *curved labels*. To avoid labels describing wavy lines we assume that $c_1, \ldots, c_n$ are cubic Bézier curves. Thus, we need to specify the four control points $s_i$, $s'_i$, $p'_i$ and $p_i$ as depicted in Fig 5d. Using the notion of boundary labeling we call the second endpoint $p_i$ of $c_i$ the *port* of $c_i$. We considered different choices of those control points, e.g., $p_i$ and $s_i$ are given, $s'_i$ is located on the perpendicular line to $C$ through $s_i$, $p'_i$ is located on the horizontal line through $p_i$ and the distance of $s_i$ and $s'_i$ is equals to the distance of $p_i$ and $p'_i$. Thus, in that example only one degree of freedom remains, namely the distance between $s_i$ and $s'_i$. We think that force-directed methods could be a reasonable way to determine those parameters. However, we still need to identify an appropriate system of forces that also respects conditions 1–3 mentioned above.

To tackle the problem from a more algorithmic point of view we make the assumption that the locations of the stops are given and the ports of the curves are located on a vertical line $\ell$ that lies to one side of $C$. To test whether curves $c_1, \ldots, c_n$ exist with respect to condition 1–3, we developed basic algorithms that either follow a greedy strategy or a dynamic-programming approach. For the greedy strategy we assume that the number of stops and ports coincide. Thus, going along $\ell$ from bottom to top the assignment between stops and ports is unique. If the number of ports is greater than the number of stops we apply a dynamic-programming approach using the observation that a curve $c_i$ of a solution partitions the instance into two independent sub-instances. We think that these algorithms can be used for finding an initial solution, which later can be refined with a force-directed algorithm as described above.

For future work we identified the following questions to be answered. How can the approaches be extended if ports lie on both sides of the given metro line $C$? Where should ports be placed? How can the case be handled that several metro lines are given? How to define an appropriate system of forces in order to fine-tune the parameters of the Bézier curves?

## References

1 Maria Angeles Garrido, Claudia Iturriaga, Alberto Márquez, José Ramón Portillo, Pedro Reyes, and Alexander Wolff. Labeling subway lines. In Peter Eades and Tadao Takaoka, editors, *ISAAC*, volume 2223 of *Lecture Notes in Computer Science*, pages 649–659. Springer, 2001.

2 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, May 2011.

3 Jonathan Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):101–114, January 2011.

4 Alexander Wolff. Drawing subway maps: A survey. *Inform., Forsch. Entwickl.*, 22(1):23–44, 2007.

## 4.6 Graphs with Circular Arc Contact Representation

*David Eppstein (University California – Irvine)*
*Éric Fusy (Ecole Polytechnique – Palaiseau)*
*Stephen Kobourov (University of Arizona – Tucson)*
*André Schulz (Universität Münster)*
*Torsten Ueckerdt (University of California – Santa Barbara)*

An *arc* is an open connected subset of a circle. For an arc $a$ the two points in the closure of $a$ that are not on $a$ are called its endpoints. If $a$ has only one endpoint, i.e., it is a circle with one point removed, then this endpoint is counted with multiplicity two. Let $\mathcal{A}$ be an arrangement of disjoint circular arcs in the plane. We say that an arc $a_1$ touches an arc $a_2$, if one of the endpoints of $a_1$ lies on $a_2$. A graph $G$ is represented by $\mathcal{A}$ if there is a bijection between the vertices of $G$ and the arcs in $\mathcal{A}$ with the property that two arcs touch in the arrangement exactly if their corresponding vertices are adjacent in $G$. We call $\mathcal{A}$ a *circular arc contact representation* of $G$, *arc-representation* for short. Notice that $G$ might contain parallel edges but no self-loops. Fig. 6 shows an example of a circular arc arrangement and its associated graph.

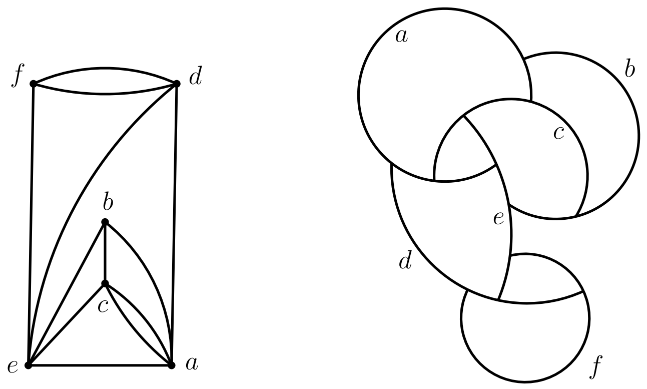


**Figure 6** A graph (on the left) and one of its arc-representation (on the right).

We study the question of which graphs can be represented by an arc-representation. In every arc-representation each arc touches at most two other arcs with its endpoints. Hence, there are at most twice as many edges in $G$ as there are vertices. This observation is not only true for the full arrangement, but also for every subset of it. Furthermore, since the arrangement is non-intersecting, the graph $G$ has to be planar as well. In terms of the following notation, graphs with arc-representation are $(2,0)$-sparse and planar.

▶ **Definition 1** (Lee and Streinu [3]). A graph $G = (V, E)$ is called $(k,\ell)$-*tight* for some natural numbers $k, \ell$, if

1. $|E| = k|V| - \ell$,
2. for every subset $X \subseteq V$ with induced edge set $E[X]$ we have that $|E[X]| \leq k|V| - \ell$.

If only condition 2. is fulfilled we call the graph $(k,\ell)$-*sparse.*

Every planar $(2,3)$-sparse graph has contact representation with straight line segments [1], and therefore an arc-representation. On the other hand, every graph with a segment contact representation is necessarily $(2,3)$-sparse. We claim the following:

▶ **Theorem 2.** *Every planar* $(2, 1)$*-sparse graph has an arc-representation.*

Since graphs with arc-representations are closed under taking subgraphs, it clearly suffices to prove Theorem 2 for $(2, 1)$-tight graphs. Our main tool for the proof is a construction sequence that produces $G$ by sequentially adding a vertex with low vertex degree, while maintaining the planarity and the tightness constraint of the graph. These local operations are called *Henneberg steps* [2]. When introducing a vertex of degree $k + 1$, we have to add $k - 1$ edges, such that tightness constraint remains fulfilled. If the new vertex has degree $k + 1$ the modification is called a Henneberg-$k$ step, H$k$ step for short. In general Henneberg steps do not have to preserve planarity. However, in our setting it suffices to consider planarity-preserving *planar* Henneberg steps.

Every planar $(2, 2)$-tight graph can be constructed by a series of planar H1 and H2 steps starting from a graph with two vertices and a double edge. Similarly, every planar $(2, 1)$-tight graph can be constructed by a series of planar H1 and H2 steps starting from a graph with two vertices and a triple edge. In the $(2, 0)$-tight case we have to allow three Henneberg steps, H1, H2 and H3. Here the base case is any plane graph each of whose components consists of two vertices and 4 parallel edges.

To show that every planar $(2, 1)$-tight graph $G$ has an arc-representation we follow its construction sequence. We start with an arc-representation of the triple edge and then apply the Henneberg steps that lead to $G$. All these steps are carried out geometrically in the arc-representation. This way we finally construct an arc-representation of $G$. The same strategy can be used to show that all $(2, 2)$-tight graphs have an arc-representation. The only difference is the base case.

To carry out the Henneberg steps in the arc-representation we need one more tool. When adding an arc to the arc-representation we have to ensure that we can establish the desired connectivity between the arcs. To guarantee this we enhance the arc-representation with so called *corridors* as spatial certificates for the *circular arc visibility*. We can show that each Henneberg step in the construction sequence maintains a required set of corridors. Some more technical parts of the proposed proof still need to be checked.

Our techniques do not apply for general planar $(2, 0)$-tight graphs yet. The main difference to the previous cases is that during the construction the graph might become disconnected. However, if the construction sequence maintains the connectivity we can prove that the graph has an arc-representation.

### References

1 Muhammad Jawaherul Alam, Therese Biedl, Stefan Felsner, Michael Kaufmann, and Stephen G. Kobourov. Proportional contact representations of planar graphs. In *Proc. 19th Symposium on Graph Drawing (GD '11)*, volume 7034 of *LNCS*, pages 26–38. Springer, 2012.

2 Lebrecht Henneberg. *Die graphische Statik der starren Systeme.* B.G. Teubner, Leipzig, 1911.

3 Audrey Lee and Ileana Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, 2008.

## 5 Open Problems

Apart from the six problems, for which working groups formed during the seminar, we collected several additional open problems during the two open problem sessions. They are briefly summarized in this section.

## 5.1 Drawing $r$-partite hypergraphs

*Günter Rote (FU Berlin)*

A 3-uniform 3-partite hypergraph is a subset of $A \times B \times C$ for three disjoint sets $A, B, C$. It can be drawn by placing points for the vertices in $A, B, C$ on three vertical lines and drawing each triplet (hyperedge) $(a, b, c)$ as a quadratic function (parabolic arc) joining the three corresponding points. The hypergraph can, in principle, be reconstructed uniquely from this drawing, even if different arcs going through a vertex have the same tangent direction. The open question is to define quality criteria for the visual appearance, beauty, clarity, or clutteredness of such a drawing and to see whether one can optimize the drawing by placing the vertices suitably. The problem extends to $r$-uniform $r$-partite hypergraphs by using curves of degree $r - 1$.

## 5.2 Characterization of Planar Lombardi Graphs

*David Eppstein (UC Irvine)*

Lombardi drawings are drawings of graphs that use circular arcs to represent edges and require that every vertex $v$ has perfect angular resolution $2\pi/\deg(v)$, see Section 3.1.8 and [1]. A graph that admits a planar Lombardi drawing is called a planar Lombardi graph. Not every planar graph has a planar Lombardi drawing [12] and it is an open question to characterize the planar Lombardi graphs. In particular, it is unknown whether every outerplanar graph has a planar Lombardi drawing.

### References

1 Christian A. Duncan, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Martin Nöllenburg. Lombardi drawings of graphs. *J. Graph Algorithms and Applications*, 16(1):85–108, 2012.

## 5.3 Small Path Covers in Planar Graphs

*André Schulz (Uni Münster)*

Let $G = (V, E)$ be a graph of a graph class $\mathcal{G}$ with $|E| = m$ edges. A *path-cover* $\mathcal{P} = \{P_1, \ldots, P_k\}$ is a partition of $E$ into edge-disjoint simple paths. The size of the cover is $\sigma(\mathcal{P}) = k$. I am interested in upper and lower bounds for the quantity

$$\operatorname{pc}(G) := \min_{G \in \mathcal{G}} \max_{\mathcal{P} \text{ is path-cover of } G} m/\sigma(\mathcal{P}).$$

In other words, how large is the average path length in the smallest path-cover in the worst case. The graphs classes $\mathcal{G}$ I am interested in are (1) planar 3-connected graphs, and (2) triangulations.

There is a simple observation for an upper bound: In every odd-degree vertex one path has to start/end. By Euler's formula a planar graph can have up to $3|V| - 6$ edges. So when all vertices in a triangulation $G$ have odd degree then $\operatorname{pc}(G) \leq 6 - \varepsilon$. The variable $\varepsilon > 0$ can be made arbitrarily small by considering larger triangulations. The same bound follows from the fact that $\lceil(|V|-1)/2\rceil$ paths have to pass through a vertex of degree $|V|-1$.

▶ Problem 1. Does any 3-connected planar graph (triangulation) $G=(V,E)$ have a path-cover of size $\mathrm{pc}(G) \geq 6 - c/|V|$, for $c > 0$ being a constant.

The problem might be related to the *linear arboricity conjecture*. This conjecture claims that the number of linear forests (disjoint unions of paths) of any graph is either $\lceil\Delta/2\rceil$ or $\lceil(\Delta+1)/2\rceil$, where $\Delta$ denotes the maximal vertex degree of the graph. It was proven for planar graphs by Wu [3].

If the graph is a triangulation it can be decomposed into edge-disjoint simple paths that have all exactly three edges [2]. The same is true for cubic bridge-less graphs [1]. So in both cases we have a lower bound of $\mathrm{pc}(G) \geq 3$ for those graph classes.

▶ Problem 2. Does there exist for every planar 3-connected graph a path-cover in which every path has exactly three edges, with the exception of one path that might be shorter?

These questions are motivated from proving lower bounds for certain problems in graph drawing with circular arcs.

**References**

1 André Bouchet and Jean-Luc Fouquet. Trois types de décompositions d'un graphe en chaînes. In C. Berge, D. Bresson, P. Camion, J. Maurras, and F. Sterboul, editors, *Combinatorial Mathematics Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75, pages 131 – 141. North-Holland, 1983.

2 Roland Häggkvist and Robert Johansson. A note on edge-decompositions of planar graphs. *Discrete Mathematics*, 283(1-3):263–266, 2004.

3 Jian Liang Wu. On the linear arboricity of planar graphs. *J. Graph Theory* 31:129–134, 1999.

## 5.4 Self-approaching Networks on Planar Point Sets

*Fabrizio Frati (University of Sydney)*

A curve $C$ in the plane with end-points $u$ and $v$ is called self-approaching from $u$ to $v$ if, for every three points $a, b$, and $c$ in this order along $C$ from $u$ to $v$, the Euclidean distance between $a$ and $c$ is greater than the Euclidean distance between $b$ and $c$. A drawing $D$ of a graph $G$ is called self-approaching if, for every ordered pair $(u, v)$ of vertices of $G$, there exists a path in $G$ whose drawing in $D$ is a self-approaching curve from $u$ to $v$.

The problem asks whether, for every point set $P$ in the plane, there exists a planar drawing $D$ of a graph $G$ that spans the points in $P$ and that is self-approaching. The problem has been defined by Alamdari et al. [1].

**References**

1 Soroush Alamdari, Timothy M. Chan, Elyot Grant, Anna Lubiw, and Vinayak Pathak. Self-approaching Graphs. In *Proc. 20th Symposium on Graph Drawing (GD '12)*, volume 7704 of *LNCS*, pages 260–271. Springer, 2013.

## 5.5 Improving Curved Drawings with Edge Direction and Curvature Optimization

*Kai Xu (Middlesex University)*

We presented three related problems on improving Lombardi-like drawings:

1. The first is how to choose the edge curve direction to improve the layout. Different criteria can be applied here. Examples are to reduce edge crossing or improve the continuity of paths between nodes.
2. The second one is weighting between angular resolution and other aesthetics criteria. A method that considers different criteria may produce a layout that is better than a method that only focuses on angular resolution. The research question is how to find the right balance, which may vary from graph to graph.
3. The last one is to have both straight and curved edges in a drawing. Curved edges are only used if they can improve certain aesthetic metrics, for example less edge crossings. The research questions are when to use curved edges and how to do this efficiently.

## 5.6 Improving Graph Readability by Spatial Distortion of Node-Link-Based Graph Depictions within Geographical Contexts

*Danny Holten (SynerScope BV)*

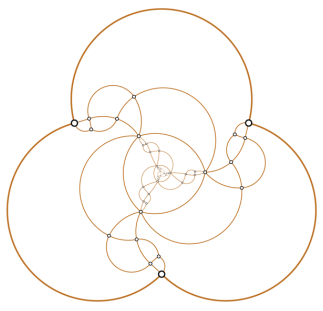
A multitude of layout algorithms exists for the generation of 2D/3D node-link-based graph layouts. Most of these algorithms are fairly unrestricted in the sense that they have a high degree of freedom with respect to the final placement of nodes and/or links within the 2D/3D space into which the layout is embedded. However, there are situations in which additional and sometimes stringent restrictions are applicable to (mostly) node placement, which can severely limit the generation of an optimized and easily readable layout due to the presence of node clutter. A typical example is when node positions are determined by, e.g., locations on a geographical map; this does not really allow for optimized placement of nodes for improved readability through uncluttered positioning. In such cases, (mild) continuous distortions can be applied to the layout to unclutter the graph and, hence, improve readability by utilizing spatial graph sparsity as a measure, e.g., by locally contracting (parts of) the graph where node/link density is low and locally expanding (parts of) the graph where node/link density is high.

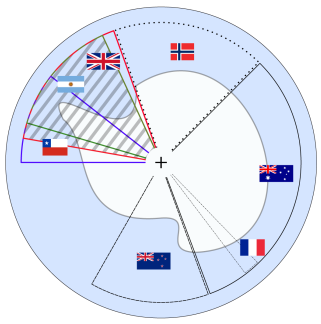
## 6 Exhibition: Bending Reality

*Maxwell J. Roberts (University of Essex)*

Previous seminars concerned with graph drawing and mapping have been staged in conjunction with successful exhibitions which have highlighted the visual nature of the subject matter [10461 – Schematization in Cartography, Visualization, and Computational Geometry – Underground Maps Unravelled; 12261 – Putting Data on the Map – Beyond the Landscape]. Continuing with this tradition, an exhibition was staged in conjunction with this seminar: *Bending Reality: Where Arc and Science Meet*.

The exhibition comprised works directly relevant to the theme of the seminar, demonstrating the use of smooth curves of any kind in order to depict spatial or abstract relationships in the arts and science. The intention was that by showcasing relevant drawings and other visual works this would provide an attractive environment in which to work, and stimulate discussion amongst seminar participants. Exhibits were solicited from seminar members and were reviewed by the exhibition committee. Part of the seminar program comprised an unofficial opening, in which participants toured the exhibits and artists described their own works. In the meantime, the exhibition is available online. Please visit **http://www.dagstuhl.de/ueber-dagstuhl/kunst/13151**.





Schottentor
Taborstraße
Messe
Schottenring
Praterstern
Rathaus
Nestroyplatz
Schweden-
platz
Volks-
theater
Stephans-
platz
Herrengasse
Rochusgasse
Stubentor
ımsquartier
Land-
straße
Stadtpark
Kettenbrücken-
Karlsplatz
2

### Exhibition Synopsis

Nature is curved. From tangled branches to meandering rivers and irregular coastlines, there is barely a straight line to be seen. In centuries past, information designers have sought to emulate this, with tree-like diagrams and twisting connections, but others have objected to the inefficiency of curves. Straight lines are direct, simple and easy to follow, so why take pointless diversions?

Straight lines can play a part in simplifying the world, highlighting routes and connections, but they can only go so far, and sometimes curves may be better suited. Whether these are free-form Béziers or regular geometric patterns, in some circumstances they can be more effective or more aesthetic, and the task of the information designer is to take a flexible approach, identifying the best tools and techniques for the particular task to hand.

This exhibition presents information from a variety of contexts, the linking theme is that the designers have investigated the use of curves in order to show routes, structures, movement, time and connections.

The exhibition was divided into seven sections, as described below. It has now been implemented online, and those artworks that are now part of the online exhibition are listed. All listed exhibits were contributed by seminar participants.

## 6.1 Curved Annotations of the World

The world is a sphere, but even when it is shown as a flat projection, curves are often used, especially to show movement.

- The German war society 1939 to 1945. Exploitation, interpretations, exclusion. Andreas Reimer. 2005
- The Flow of Whisky. Kevin Buchin, Bettina Speckmann, and Kevin Verbeek. 2010
- A Bicycle's Journey. Jo Wood. 2013
- Airline graph after several steps of an edge bundling algorithm. Emden Gansner, Yifan Hu, Stephen North and Carlos Scheidegger. 2011

## 6.2 Curving the World

Curves in the world are rarely simple, whether twisted gnarled branches or a fractal coastline. Even man-made structures such as railway lines can have complex, uncertain trajectories. These can be smoothed and simplified, but sometimes there is a temptation to go further, turning the world into alien geometric shapes.

- O-ZONE III. Arthur van Goethem, Wouter Meulemans, Andreas Reimer, Herman Haverkort, and Bettina Speckmann. 2013
- Curve Limit IV. Arthur van Goethem, Wouter Meulemans, Bettina Speckmann, and Jo Wood. 2013
- Countries in Southeast Asia, with straight edges, Bézier curves, and circular arcs. Arthur van Goethem, Wouter Meulemans, Andreas Reimer, Herman Haverkort, and Bettina Speckmann. 2013
- Circular-arc cartograms of the states of Germany. Martin Nöllenburg. 2013

## 6.3 Early Metro Maps

The most widely used examples of the curves of reality being converted into straight lines are metro maps worldwide. This information design technique is relatively recent, first used in Berlin (1931) and London (1933). Before this, curves on maps generally were used in an attempt to represent reality. However, there are also examples of attempts to simplify, even going as far as using a regular circle.

- London Underground map based upon designs by F.H. Stingemore, 1925-1932. Maxwell J. Roberts. 2009
- Metropolitan Line carriage diagram from the early 1920s. Maxwell J. Roberts. 2012
- Berlin S-Bahn map based upon a 1931 design. Maxwell J. Roberts. 2012
- London Underground map based upon a 1933 design by Henry Beck. Maxwell J. Roberts. 2009
- Paris Metro map based upon a 1939 design by Loterie Nationale, France. Maxwell J. Roberts. 2012
- Moscow Metro schema based upon 1970s designs. Maxwell J. Roberts. 2009

## 6.4 Metro Maps Using Freeform Béziers

Sometimes, straight lines fail to simplify reality. In the case of a complex highly interconnected network, gentle curves smooth away harsh zigzags, potentially revealing the underlying structure of the networks.

- The Madrid Metro: An all-curves design. Maxwell J. Roberts. 2009
- Berlin all-curves U- and S-Bahn network map. Maxwell J. Roberts. 2012
- Automatically Generated Drawing of the London Underground using Bézier curves. M. Fink, H. Haverkort, M. Nöllenburg, M. Roberts, J. Schuhmann, and A. Wolff. 2013
- An all-curves map of the Paris Metro. Maxwell J. Roberts. 2007
- Curvy tube map. Maxwell J. Roberts. 2008
- Subway network of Vienna drawn by a force-Directed method using Bézier curves. A. Wolff, M. Fink, H. Haverkort, M. Nöllenburg, and J. Schuhmann. 2013

## 6.5 Metro Maps Using Concentric Circles

Highly abstract and stylised, this new way of showing networks does not necessarily offer simplified line trajectories, but nonetheless presents a highly organised view of the network, which many people find striking.

- The Vienna U-Bahn in minimalist style. Therese Biedl and Maxwell J. Roberts. 2012
- The Madrid Metro: A concentric circles design. Maxwell J. Roberts. 2013
- A concentric circles map of the Moscow Metro. Maxwell J. Roberts. 2013
- Berlin concentric circles U- and S-Bahn network map. Maxwell J. Roberts. 2013
- A map of the London Underground based upon concentric circles, spokes, and tangents. Smooth corners version. Maxwell J. Roberts. 2013

## 6.6 Curved Relationships

In theory, the display of abstract concepts such as time and relatedness need not be constrained by preconceptions derived from our experience of nature. In practice, use of curves can add an aesthetic dimension, or even assist in the presentation of information.

- Mark Lombardi: Kunst und Konspiration. Film poster. 2012
- The collaboration network between Jazz bands. S. Pupyrev, L. Nachmanson, S. Bereg, and A.E. Holroyd. 2011
- 40 Jahre Universität Konstanz. Ulrik Brandes. 2006
- Poster for 21st International Symposium on Graph Drawing. David Auber. 2013
- Spaghetti à la Wehrli. Mereke van Garderen and Bettina Speckmann. 2013

## 6.7 Mathematical Abstractions

Curves are no stranger to the mathematical world, and sometimes enable combinatorial structures, abstract graphs and other concepts to be visualised more efficiently than with straight lines.

- Iterated Möbius-transformations. André Schulz. 2013
- Random confluent Hasse diagram on 222 points. David Eppstein. 2013

- Flowerpot. Günter Rote. 2013
- Lombardi drawings. C. Duncan, D. Eppstein, M. Goodrich, S. Kobourov, and M. Löffler. 2011
- The Petersen family. David Eppstein. 2010
- Lombardi drawing of the full binary outertree on 1025 vertices (in a non-standard embedding). Maarten Löffler. 2012
- A selection of graphs from the Rome library drawn using a Lombardi force-directed algorithm. Stephen Kobourov. 2011
- Hyperbolic tessellation generated by a Fuchsian group with genus 1 and a period of 3. Jakob von Raumer. 2013
- Covering of the hyperbolic plane. Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. 2013
- The Nauru graph in 3d. David Eppstein. 2008

## Participants

- Patrizio Angelini
Università di Roma III, IT
- Michael Bekos
National TU – Athens, GR
- David Eppstein
Univ. of California – Irvine, US
- Fabrizio Frati
The University of Sydney, AU
- Eric Fusy
Ecole Polytechnique – Palaiseau, FR
- Martin Gronemann
Universität Köln, DE
- Jan-Henrik Haunert
Universität Würzburg, DE
- Herman J. Haverkort
TU Eindhoven, NL
- Michael Hemmer
TU Braunschweig, DE
- Danny Holten
SynerScope BV, NL
- Michael Kaufmann
Universität Tübingen, DE
- Stephen G. Kobourov
Univ. of Arizona – Tucson, US
- Sylvain Lazard
INRIA Grand Est – Nancy, FR
- Maarten Löffler
Utrecht University, NL
- Tamara Mchedlidze
KIT – Karlsruhe Institute of Technology, DE
- Wouter Meulemans
TU Eindhoven, NL
- Lev Nachmanson
Microsoft Corp. – Redmond, US
- Benjamin Niedermann
KIT – Karlsruhe Institute of Technology, DE
- Arlind Nocaj
Universität Konstanz, DE)
- Martin Nöllenburg
KIT – Karlsruhe Institute of Technology, DE
- Sergey Pupyrev
Univ. of Arizona – Tucson, US
- Helen C. Purchase
University of Glasgow, GB
- Andreas Reimer
Universität Heidelberg, DE
- Maxwell J. Roberts
University of Essex, GB
- Günter Rote
FU Berlin, DE
- André Schulz
Universität Münster, DE
- Aidan Slingsby
City University – London, GB
- Bettina Speckmann
TU Eindhoven, NL
- Monique Teillaud
INRIA Sophia Antipolis – Méditerranée, FR
- Torsten Ueckerdt
KIT – Karlsruhe Institute of Technology, DE
- Kevin Verbeek
University of California – Santa Barbara, US
- Alexander Wolff
Universität Würzburg, DE
- Jo Wood
City University – London, GB
- Kai Xu
Middlesex University, GB

Report from Dagstuhl Seminar 13161

# Interface of Computation, Game Theory, and Economics

**Edited by**
**Sergiu Hart[1], Éva Tardos[2], and Bernhard von Stengel[3]**

1 **Hebrew University, Jerusalem, IL,** hart@huji.ac.il
2 **Cornell University – Ithaca, US,** eva@cs.cornell.edu
3 **London School of Economics, GB,** stengel@nash.lse.ac.uk

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13161 "Interface of Computation, Game Theory, and Economics". The workshop was strongly interdisciplinary, on the leading edge of current topics generally connected to algorithmic game theory: Mechanism design and auctions, interactions in networks, social models, and dynamics and equilibrium in games and markets. We summarize these topics, give the talk abstracts, and comment on experiences related to the organization of the workshop.

**Seminar** 14.–19. April, 2013 – www.dagstuhl.de/13161
**1998 ACM Subject Classification** J.4.1. Computer Applications – Economics
**Keywords and phrases** Algorithmic Game Theory, Economics, Internet, Nash Equilibrium, Mechanism Design, Auctions
**Digital Object Identifier** 10.4230/DagRep.3.4.69

## 1 Executive Summary

*Sergiu Hart*
*Éva Tardos*
*Bernhard von Stengel*

The aim of this seminar was to study research issues at the interface of computing, game theory and economics. It facilitated discussions among people working in different disciplines. The majority of participants were academics from computer science departments, and the others (about one third) from other disciplines such as economics or corporate research departments of Google or Microsoft. All have strong cross-disciplinary interests.

Economic transactions on the internet are of ever-increasing importance. In order to execute and support them algorithmically, it is important to understand the agents' incentives on one hand and computational constraints on the other hand. This is studied in approaches to mechanism design and auctions, which formed a large part of the topics of this workshop.

Theoretical and practical issues of *mechanism design* were topics of the following presentations: epistemic implementations with belief levels (Jing Chen), translating agent-provided inputs to optimization (Constantinos Daskalakis), reward schemes (Shahar Dobzinski), the difficulties of allocating more than one good (Sergiu Hart), advertisement exchanges (Vahab Mirrokni), mechanisms for the private supply of a public good (Rudolf Müller), truthfulness versus privacy (Aaron Roth), composing mechanisms (Vasilis Syrgkanis), and allocating indivisible objects (Rakesh V. Vohra).

Aspects of *auctions* concerned “expressiveness” about preferences (Paul Dütting), the approximate optimality of marginal revenue maximization (Jason D. Hartline), improving the design of online advertising auctions (Kevin Leyton-Brown), commitment (Katrina Ligett), inefficiency of multi-unit auctions (Vangelis Markakis), symmetric auctions (Mallesh Pai), interdependent values (Tim Roughgarden), and spectrum auctions (Ilya Segal).

Understanding the interconnectedness of complex economic systems requires models and theories for the underlying *network* structures and their dynamics. Networks were studied with respect to social segregation (Nicole Immorlica), practical market applications (Ramesh Johari), online creation (Thomas Kesselheim), competition (Brendan Lucier), and social contagion (Sigal Oren).

*Social models*, with bridges to mechanism design, were studied in presentations on division protocols (Simina Branzei), randomized social choice (Markus Brill), ranking methods (Gabrielle Demange), power changes in voting games (Edith Elkind), and incentives beyond selfishness (Guido Schäfer).

Achieving and computing an equilibrium in *dynamic models of large interactions* such as games and market models was studied for large aggregative games (Yakov Babichenko), new price updating in markets (Nikhil R. Devanur), payoff queries for games (Paul W. Goldberg), limit processes for evolutionary games (Bill Sandholm), and tournament competitions (Bernhard von Stengel).

The topics were chosen by the presenters, not by the organizers. The rather strong emphasis on mechanism design and auctions (which may have caused one single critical feedback comment on “too much groupthink”) reflects the strong current interest in this area, in line with its economic importance, for example as the source of the riches of Google and other internet search engines.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Best-Reply Dynamics in Large Aggregative Games

*Yakov Babichenko (CalTech, US, babich@caltech.edu)*

We consider small-influence aggregative games with a large number of players $n$. For this class of games we present a best-reply dynamic with the following two properties. First, the dynamic reaches Nash approximate equilibria fast (in at most $cn\log(n)$ steps for some constant $c > 0$). Second, Nash approximate equilibria are played by the dynamic with a limit frequency that is exponentially close to 1 (at least $1 - e^{-c'n}$ for some constant $c' > 0$).

### 3.2 Equilibria of Generalized Cut and Choose Protocols

*Simina Branzei (Aarhus University, DK, simina@cs.au.dk)*

Classic cake cutting protocols – which fairly allocate a divisible good among agents with heterogeneous preferences – are susceptible to manipulation. Do their strategic outcomes still guarantee fairness? We model the interaction among agents as a game and study its Nash equilibria. We show that each protocol in the novel class of generalized cut and choose protocols – which includes the most important discrete cake cutting protocols – is guaranteed to have an $\varepsilon$-equilibrium for all $\varepsilon > 0$. Moreover, we observe that the (approximate) equilibria of proportional protocols – which guarantee each of the $n$ agents a $1/n$-fraction of the cake – must be (approximately) proportional. Finally, we design a generalized cut and choose protocol where all equilibrium outcomes satisfy the stronger fairness notion of envy-freeness.

### 3.3 On the Tradeoff between Economic Efficiency and Strategyproofness in Randomized Social Choice

*Markus Brill (TU München, DE, brill@in.tum.de)*

Two fundamental notions in microeconomic theory are efficiency – no agent can be made better off without making another one worse off – and strategyproofness – no agent can obtain a more preferred outcome by misrepresenting his preferences. When social outcomes are probability distributions (or lotteries) over alternatives, there are varying degrees of these notions depending on how preferences over alternatives are extended to preference over lotteries. We show that efficiency and strategyproofness are incompatible to some extent when preferences are defined using stochastic dominance (SD) and therefore introduce a natural weakening of SD based on Savage's sure-thing principle (ST). While random serial dictatorship is SD-strategyproof, it only satisfies ST-efficiency. Our main result is that strict maximal lotteries – an appealing class of social decision schemes due to Kreweras and Fishburn – satisfy SD-efficiency and ST-strategyproofness.

## 3.4 Epistemic Implementation

*Jing Chen (IAS and Stony Brook, US, jingchen@csail.mit.edu)*

In a setting of incomplete information, we model the hierarchy of the players' beliefs about each other's payoff types in a set-theoretic way. A player's beliefs can be totally arbitrary, and the beliefs of different players can be inconsistent with each other. In single-good auctions, for $k = 0, 1, \ldots$, we define a revenue benchmark $G^k$ on the players' belief hierarchy. Intuitively, $G^k \geq v$ if and only if there exist at least two players "believing that there exists a player ..." ($k$ times) valuing the good at least $v$. We construct an interim individually rational mechanism $M$ that, without any clue about the players' beliefs and their rationality level, virtually guarantees revenue $G^k$ whenever the players happen to be level-$(k+1)$ rational. We also separate the revenue achievable with level-$k$ and level-$(k+1)$ rational players. For every $k \geq 0$, we show that no interim individually rational mechanism can virtually guarantee revenue $G^k$ when the players' rationality level is $k$ instead of $k+1$.

## 3.5 Reductions from Mechanism to Algorithm Design

*Constantinos Daskalakis (MIT, US, costis@csail.mit.edu)*

Algorithmic mechanism design centers around the following question: How much harder is optimizing an objective function over inputs that are furnished by rational agents compared to when the inputs are known? We provide a computationally efficient, black-box reduction from mechanism design (i.e. optimizing over rational inputs) to algorithm design (i.e. optimizing over known inputs) in general Bayesian settings. As an application of our reduction, we extend Myerson's celebrated auction to the multi-item setting.

## 3.6 A Ranking Method Based on Handicaps

*Gabrielle Demange (Paris School of Economics, FR, demange@pse.ens.fr)*

Ranking methods are a fundamental tool in many areas. Popular methods are based on the statements of "experts" and aggregate them in some way. As such, there is a variety of meaningful ranking methods, more or less adapted to the environment under consideration. We introduce and characterizes a new method, called the handicap-based method. The method assigns scores to the items and weights to the experts. Scores and weights form an equilibrium for a relationship based on the notion of handicaps. The method is, in a sense that we make precise, the counterpart of the counting method in environments that require intensity-invariance. Intensity-invariance is a desirable property when the intensity of the experts' statements has to be controlled. Otherwise, both the counting and handicap-based methods satisfy a property called homogeneity, which is a desirable property when cardinal statements matter, as is the case in many applications.

## 3.7 Tatonnement Beyond Gross Substitutes? Gradient Descent to the Rescue

*Nikhil R. Devanur (Microsoft, Redmond, US, nikdev@microsoft.com)*

**Main reference** Y.K. Cheung, R. Cole, N.R. Devanur, "Tatonnement beyond gross substitutes?: Gradient descent to the rescue," in Proc. of the 45th Annual ACM Symp. on Symposium on Theory of Computing (STOC'13), pp. 191–200, ACM, 2013.
**URL** http://dx.doi.org/10.1145/2488608.2488633

Tatonnement is a simple and natural rule for updating prices in exchange (Arrow-Debreu) markets. We define a class of markets for which tatonnement is equivalent to gradient descent. This is the class of markets for which there is a convex potential function whose gradient is always equal to the negative of the excess demand and we call it Convex Potential Function (CPF) markets. We show the following results. CPF markets contain the class of Eisenberg Gale (EG) markets, defined previously by Jain and Vazirani. The subclass of CPF markets for which the demand is a differentiable function contains exactly those markets whose demand function has a symmetric negative semi-definite Jacobian. We define a family of continuous versions of tatonnement based on gradient descent using a Bregman divergence. As we show, all processes in this family converge to an equilibrium for any CPF market. This is analogous to the classic result for markets satisfying the Weak Gross Substitutes property. A discrete version of tatonnement converges toward the equilibrium for the following markets of complementary goods; its convergence rate for these settings is analyzed using a common potential function. Fisher markets in which all buyers have Leontief utilities. The tatonnement process reduces the distance to the equilibrium, as measured by the potential function, to an $\varepsilon$ fraction of its initial value in $O(1/\varepsilon)$ rounds of price updates. Fisher markets in which all buyers have complementary CES utilities. Here, the distance to the equilibrium is reduced to an $\varepsilon$ fraction of its initial value in $O(\log(1/\varepsilon))$ rounds of price updates.

This shows that tatonnement converges for the entire range of Fisher markets when buyers have complementary CES utilities, in contrast to prior work, which could analyze only the substitutes range, together with a small portion of the complementary range.

## 3.8 Shared Resource Management via Reward Schemes

*Shahar Dobzinski (Weizmann Institute, Rehovot, IL, dobzin@gmail.com)*

**Joint work of** Dobzinski, Shahar; Ronen, Amir;

We study scenarios in which consumers have several options of using a shared resource (e.g., truck operators that can drive either in peak or off peak hours). Our goal is to design reward schemes that, in equilibrium, minimize the cost to society and the total sum of rewards. We introduce a simple reward scheme which does not require any knowledge of the private values of the consumers, yet its cost in equilibrium is always within a factor of $\sqrt{\alpha}$ of the cost of the optimal scheme that has complete knowledge of the consumers' valuations. Here, alpha is the ratio between the costs of the worst and best alternatives. We show the optimality of our scheme in various settings by providing lower bounds.

## 3.9 Expressiveness and Robustness of First-Price Position Auctions

*Paul Dütting (EPFL Lausanne, CH, paul.duetting@epfl.ch)*

It has been argued that increasing the expressiveness of an auction increases the quality of the outcomes it is able to support. Intuitively, more expressive auctions should allow agents to more accurately reveal their preferences in the presence of uncertainty. We study this issue in the context of a position auction in which valuations are one-dimensional but the designer is uncertain about the relative values of the positions. In this setting, efficient equilibria may fail to exist for simplified auctions that solicit only a single bid from each agent, but existence can be restored by increasing expressiveness. In particular, we show this to be the case for a generalized first-price (GFP) auction. In addition to the existence of an efficient Bayes-Nash equilibrium, the GFP auction is robust to varying assumptions about the information available to agents while second-price and VCG auctions are not. Technically, our main result is interesting because the Bayes-Nash equilibrium is constructed for a multi-dimensional bid space. The structure of the equilibrium bids moreover provides an intuitive explanation for why first-price payment rules may be able to support equilibria in a wider range of scenarios than second-price payment rules.

## 3.10 Dynamic Coalitional Games

*Edith Elkind (Nanyang Technical University, Singapore, SG, eelkind@ntu.edu.sg)*

**Joint work of** Elkind, Edith; Pasechnik, Dmitrii V.; Zick, Yair;
**Main reference** E. Elkind, D.V. Pasechnik, Y. Zick, "Dynamic weighted voting games," in Proc. of the Int'l Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'13), pp. 515-522, IFAAMAS, 2013.
**URL** http://dl.acm.org/citation.cfm?id=2485003

We define and study dynamic weighted voting games – weighted voting games where the weight of each player and the quota may change as a function of time. We investigate computational aspects of such games under the assumption that all weights and the quota are given by polynomials with integer coefficients. We focus on two types of algorithmic questions: computing a given solution concept at a particular point in time, and checking that a certain function of the game (e.g., the Shapley value of a given player or the value of the least core) remains within given bounds during a particular time interval. We provide pseudopolynomial algorithms for both types of problems, for a variety of solution concepts. We then use our results to investigate the changes in power distribution in the Council of the European Union over the next 50 years.

## 3.11 Payoff Queries

*Paul W. Goldberg (University of Liverpool, UK, P.W.Goldberg@liverpool.ac.uk)*

I give an informal introduction to "payoff query" algorithms, where an algorithm can specify a pure profile of a game (with initially unknown payoffs) and get told the payoffs for that pure profile. Given a class of games, and a solution concept, the challenge is to figure out the query complexity of solving an initially-unknown game from that class.

## 3.12 Two(!) Good To Be True

*Sergiu Hart (Hebrew University, Jerusalem, IL, hart@huji.ac.il)*

How to sell goods optimally? While the mechanism-design literature has solved this problem neatly when there is only one good, the multiple goods case turns out to be extremely difficult, mathematically and conceptually. Much of what is true for one good does not extend to multiple goods. We will try to explain the difficulties, show what can go wrong, and then present some universal approximation results. The talk is essentially self-contained; no background in mechanism design is necessary.

## 3.13 The Simple Economics of Approximately Optimal Auctions

*Jason D. Hartline (Northwestern University, Evanston, US, hartline@eecs.northwestern.edu)*

The intuition that profit is optimized by maximizing marginal revenue is a guiding principle in microeconomics. In the classical auction theory for agents with quasi-linear utility and single dimensional preferences, Bulow and Roberts (1989) show that the optimal auction of Myerson (1981) is in fact optimizing marginal revenue. In particular Myerson's virtual values are exactly the derivative of an appropriate revenue curve. We consider mechanism design in environments where the agents have multi-dimensional and non-linear preferences. Understanding good auctions for these environments is considered to be the main challenge in Bayesian optimal mechanism design. In these environments maximizing marginal revenue may not be optimal, and furthermore, there is sometimes no direct way to implementing the marginal revenue maximization mechanism. Our contributions are twofold: we give procedures for implementing marginal revenue maximization in general, and we show that marginal revenue maximization is approximately optimal. Our approximation factor smoothly degrades in a term that quantifies how far the environment is from an ideal one (i.e., where marginal revenue maximization is optimal).

## 3.14 An Analysis of One-Dimensional Schelling Segregation

*Nicole Immorlica (Northwestern University, Evanston, US, nickle@eecs.northwestern.edu)*

We analyze the Schelling model of segregation in which a society of $n$ individuals live in a ring. Each individual is one of two races and is only satisfied with his location so long as at least half his $2w$ nearest neighbors are of the same race as him. In the dynamics, randomly-chosen unhappy individuals successively swap locations. We consider the average size of monochromatic neighborhoods in the final stable state. Our analysis is the first rigorous analysis of the Schelling dynamics. We note that, in contrast to prior approximate analyses, the final state is nearly integrated: the average size of monochromatic neighborhoods is independent of $n$ and polynomial in $w$.

## 3.15 The Engineer as Economist: The Design of Online Market Platforms

*Ramesh Johari (Stanford University, US, ramesh.johari@stanford.edu)*

Markets are an ancient institution for matching those willing to provide a good or service with those who want it. Physical markets were typically slow to evolve, with simple institutions governing trade, and trading partners generally facing a daunting challenge in finding the "right" partner.

Online marketplaces dramatically disrupt this tradition. Such markets – from eBay, to Google's sponsored search auction, to online labor markets such as oDesk and TaskRabbit – can rapidly respond to evolving market trends, and "engineer" in fine grained ways the interactions of their participants with the platform. Further, the traditional difficulty of *finding* even one trading partner has been replaced with a new difficulty: how to narrow down a plethora of choices? Motivated by this new landscape, this talk will discuss some of the challenges that engineers face in designing and implementing emerging online marketplaces.

## 3.16 Online Independent Set Beyond the Worst-Case: Secretaries, Prophets, and Periods

*Thomas Kesselheim (Cornell University, US, kesselheim@cs.cornell.edu)*

We investigate online algorithms for maximum (weight) independent set on graph classes with bounded inductive independence number like interval and disk graphs with applications to, e.g., task scheduling and spectrum allocation. In the online setting, it is assumed that nodes of an unknown graph arrive one by one over time. An online algorithm has to decide whether an arriving node should be included into the independent set. Unfortunately, this natural and practically relevant online problem cannot be studied in a meaningful way within a classical competitive analysis as the competitive ratio on worst-case input sequences is lower bounded by $\Omega(n)$. This devastating lower bound holds even for randomized algorithms on unweighted interval graphs and, hence, for the most restricted graph class under consideration.

As a worst-case analysis is pointless, we study online independent set in a stochastic analysis. Instead of focussing on a particular stochastic input model, we present a generic sampling approach that enables us to devise online algorithms achieving performance guarantees for a variety of input models. In particular, our analysis covers stochastic input models like the secretary model, in which an adversarial graph is presented in random order, and the prophet-inequality model, in which a randomly generated graph is presented in adversarial order. Our sampling approach bridges thus between stochastic input models of quite different nature. In addition, we show that the same performance guarantees can be obtained for a period-based input model that is inspired by practical admission control applications.

Our sampling approach yields an online algorithm for maximum independent set on interval and disk graphs with competitive ratio $O(1)$ with respect to all of the mentioned stochastic input models. More generally, for graph classes with inductive independence number $\rho$, the competitive ratio is $O(\rho^2)$. The approach can be extended towards maximum weight independent set by losing only a factor of $O(\log n)$ in the competitive ratio with $n$ denoting the (expected) number of nodes. This upper bound is complemented by a lower bound of $\Omega(\log n/\log^2 \log n)$ showing that our sampling approach achieves nearly the optimal competitive ratio in all of the considered models. Furthermore, we generalize our analysis to address several practically motivated extensions of the independent set problem, e.g., arrival and departure times of nodes or edge-weighted graphs capturing SINR-type interference conflicts in wireless networks.

## 3.17 Revenue Optimization in the Generalized Second-Price Auction

*Kevin Leyton-Brown (University of British Columbia, Vancouver, CA, kevinlb@cs.ubc.ca)*

We consider the question of how to optimize revenue in advertising auctions without departing from the generalized second-price. We consider several different GSP variants (including squashing and different types of reserve prices), and how to set their parameters optimally. Our main finding is that unweighted reserve prices (i.e., where each advertiser has the same per-click reserve price) are dramatically better than the quality-weighted reserve prices that have become common practice in the last few years. This result is extremely robust, arising from theoretical analysis as well as multiple computational experiments. Our work also includes one of the first studies of how squashing and reserve prices interact, and of how equilibrium selection affects the revenue of GSP when features such as reserves or squashing are applied.

## 3.18 Preplay Commitment in First-Price Auctions

*Katrina Ligett (CalTech, Pasadena, US, katrina@caltech.edu)*

We study a variation of the standard single-item sealed-bid first-price auction wherein all bidders know one another's valuations, and one bidder (the leader) publicly commits to a (possibly mixed) strategy before the others submit their bids. We formulate the auction as a two-stage Stackelberg game, and study the impact of commitment on the utilities of the bidders and the auctioneer. For the case where the leader's valuation is the highest or the second highest (including, e.g., when there are only two bidders), we characterize the commitment that maximizes the expected payoff of the leader. In this case, both the leader and the bidder with the highest valuation among the other bidders strictly benefit from the commitment—each obtains an expected payoff higher than that achieved at a Nash equilibrium of the standard first-price auction. For an important variant of our model where the leader's commitment is restricted to be a discrete random variable (and thus a credible commitment may be more practically implemented), we characterize the leader's optimal commitment as a solution to an optimization problem. There, we study the extent to which a discrete-valued commitment can approximate the maximum expected payoff achievable under committing to arbitrary mixed strategies.

## 3.19 A Model of Bertrand Price Competition in Networks

*Brendan Lucier (Microsoft Research, Cambridge, US, brlucier@microsoft.com)*

We study scenarios where multiple sellers of a homogeneous good compete on prices, where each seller can only sell to some subset of the buyers. Crucially, sellers cannot price-discriminate between buyers. We model the structure of the competition by a graph (or hyper-graph), with nodes representing the sellers and edges representing populations of buyers. We study equilibria in the game between the sellers, prove that they always exist, and present various structural, quantitative, and computational results about them. We also analyze the equilibria completely for a few cases. Many questions are left open.

## 3.20 On the Inefficiency of Standard Multi-unit Auction Formats

*Vangelis Markakis (Athens University of Economics and Business, GR, markakis@gmail.com)*

**Joint work of** de Keijzer, Bart; Markakis, Vangelis; Schaefer, Guido; Telelis, Orestis;

We study two standard multi-unit auction formats for allocating multiple units of a single good to multi-demand bidders. The first one is the Discriminatory Price Auction, which charges every winner his winning bids. The second is the Uniform Price Auction, which determines a uniform price to be paid per unit. Variants of both formats find applications ranging from the allocation of bonds to investors, to online sales over the internet, facilitated by popular online brokers.

For these multi-unit auction formats, we consider two bidding interfaces: (i) standard bidding, which is most prevalent in the scientific literature, and (ii) uniform bidding, which is the most widely used interface in practical applications. We evaluate the economic inefficiency of the two formats for both bidding interfaces, by means of upper and lower bounds on the Price of Anarchy for pure equilibria and mixed Bayes-Nash equilibria. Our results for bidders with submodular valuations improve upon bounds that have been obtained recently in [Markakis, Telelis, SAGT 2012] and [Syrgkanis, Tardos, STOC 2013]. Moreover, we also consider for the first time bidders with subadditive valuation functions and obtain constant upper bounds there as well.

## 3.21 Mechanism Design Problems in Ad Exchanges and Budget Constraints

*Vahab Mirrokni (Google, New York, US, mirrokni@google.com)*

I will give a survey of mechanism design problems motivated by ad exchanges and budget constraints in online advertising. For each problem, I will present preliminary/known results and pose open problems and research directions. Some topics that are discussed are auctions in the presence of intermediaries, optimal revenue-sharing double auctions, and pareto-optimal polyhedral clinching auctions.

## 3.22 Optimal Mechanism Design for the Private Supply of a Public Good

*Rudolf Müller (Maastricht University, NL, r.muller@maastrichtuniversity.nl)*

We study the problem of finding the profit-maximizing mechanism for a monopolistic provider of a single, non-excludable public good. This problem has been well studied for the case when agents' signals are independently distributed, but the literature is almost silent about the case of general joint distributions. Our model covers the most general setting, namely, we allow for correlation in the signal distribution as well as for informational externalities. We investigate the problem from an automated mechanism design perspective, meaning that we want to understand the algorithmic complexity of finding the optimal mechanism when we are given a finite set of signal profiles and their distribution.

We show that the optimal deterministic, ex-post incentive compatible, ex-post individual rational mechanism can be computed in polynomial time by reducing the problem to finding a maximal weight closure in a directed graph. Node weights in the graph correspond to conditional virtual values. When valuations are independent and independently distributed, the constructed mechanism is also optimal among all Bayes-Nash implementable and interim individual rational mechanisms. In contrast, for dependent valuations strictly higher profit can be achieved if one allows for interim individual rationality or Bayes-Nash implementability. By invoking techniques due to Cremer and McLean [1988], we show that optimal deterministic, interim individual rational, ex-post implementable mechanisms still can be found in polynomial time if the joint distribution of signals satisfies certain regularity conditions. Finally, we demonstrate that our techniques can be adapted for the excludable public good problem as well.

## 3.23 Selection and Influence in Cultural Dynamics

*Sigal Oren (Cornell University, US, sigal@cs.cornell.edu)*

One of the fundamental principles driving diversity or homogeneity in domains such as cultural differentiation, political affiliation, and product adoption is the tension between two forces: influence (the tendency of people to become similar to others they interact with) and selection (the tendency to be affected most by the behavior of others who are already similar). Influence tends to promote homogeneity within a society, while selection frequently causes fragmentation. When both forces are in effect simultaneously, it becomes an interesting question to analyze which societal outcomes should be expected.

In order to study the joint effects of these forces more formally, we analyze a natural model built upon active lines of work in political opinion formation, cultural diversity, and language evolution. Our model posits an arbitrary graph structure describing which "types" of people can influence one another: this captures effects based on the fact that people are only influenced by sufficiently similar interaction partners. In a generalization of the model, we introduce another graph structure describing which types of people even so much as come in contact with each other. These restrictions on interaction patterns can significantly alter the dynamics of the process at the population level.

For the basic version of the model, in which all individuals come in contact with all others, we achieve an essentially complete characterization of (stable) equilibrium outcomes and prove convergence from all starting states. For the other extreme case, in which individuals only come in contact with others who have the potential to influence them, the underlying process is significantly more complicated; nevertheless we present an analysis for certain graph structures.

## 3.24 Symmetric Auctions

*Mallesh Pai (University of Pennsylvania, US, mallesh@econ.upenn.edu)*

Real-world auctions are often restricted to being symmetric (anonymous and nondiscriminatory) due to practical or legal constraints. We examine when this restriction prevents a seller from achieving his objectives. In an independent private value setting, we characterize the set of incentive compatible and individually rational outcomes that can be implemented via a symmetric auction. Our characterization shows that symmetric auctions can yield a large variety of discriminatory outcomes such as revenue maximization and affirmative action. We also characterize the set of implementable outcomes when individual rationality holds in an ex post rather than an interim sense. This additional requirement may prevent the seller from maximizing revenue.

## 3.25 Mechanism Design in Large Games and Differential Privacy

*Aaron Roth (University of Pennsylvania, US, aaroth@cis.upenn.edu)*

We study the design of mechanisms satisfying two desiderata – incentive compatibility and privacy. The first, requires that each agent should be incentivized to report her private information truthfully. The second, privacy, requires the mechanism not reveal "much" about any agent's type to other agents. We propose a notion of privacy we call Joint Differential Privacy. It is a variant of Differential Privacy, a robust notion of privacy used in the Theoretical Computer Science literature. We show by construction that such mechanisms, i.e. ones which are both incentive compatible and jointly differentially private exist when the game is "large", i.e., there are a large number of players, and any player's action affects any other's payoff by at most a small amount. Our mechanism adds carefully selected noise to no-regret algorithms similar to those studied in Foster-Vohra and Hart-Mas-Colell. It therefore implements an approximate correlated equilibrium of the full information game induced by players' reports.

## 3.26 Optimal Ex Post and Prior-Independent Auctions with Interdependent Values

*Tim Roughgarden (Stanford University, US, tim@cs.stanford.edu)*

We study optimal and approximately-optimal mechanism design questions in the interdependent values model, which generalizes the standard setting of independent and private values. We focus our attention on ex post incentive compatible and individually rational mechanisms, and develop an analog of Myerson's optimal auction theory that applies to many interdependent settings of interest. We demonstrate two applications for specific interdependent settings: First, a parallel result to the well-known optimality of the second-price auction with reserve for i.i.d. bidders, where the English auction replaces the second-price one. Second, we identify good prior-independent auctions – auctions with near-optimal expected revenue across a wide range of priors – for certain interdependent value settings.

## 3.27 Large Deviations and Stochastic Stability in the Large Population Limit

*Bill Sandholm (University of Wisconsin, Madison, US, whs@ssc.wisc.edu)*

In this talk I will give an overview about new tools for equilibrium selection in games in the framework of stochastic evolutionary game theory. In this project we investigate stochastic stability theory from a new angle. We elaborate on the precise role of the parameters in this game theoretic model, which are the population size and the level of noise in the agents' updating decisions. Stochastic stability theory is concerned with understanding the long-run properties of the stochastic evolutionary game dynamics when these parameters are taken to their respective limits separately, or simultaneously. For each possible way of taking limits, we present the appropriate technique to understand the long-run of the game dynamics. This requires a novel and interesting combination of various mathematical techniques, such as large deviations theory and optimal control. We also discuss the computational problem of stochastic stability in simple game settings.

## 3.28 Altruism and Spite in Games

*Guido Schäfer (Guido Schäfer, CWI, Amsterdam, NL, G.Schaefer@cwi.nl)*

In most game-theoretical studies it is assumed that the decision makers base their decisions on purely selfish grounds. This assumption is in stark contrast with a large body of research in experimental economics and the social sciences, which suggest that decision makers are often motivated by other-regarding preferences such as altruism, spite or fairness. Very little attention has been given to the analysis of the impact of such alternative behaviors. In this talk, we review some recent advances in the study of the inefficiency of equilibria when players are (partially) altruistic or spiteful and highlight a few counter-intuitive results.

## 3.29 Heuristic Auctions and U.S. Spectrum Repurposing

*Ilya Segal (Stanford University, US, ilya.segal@stanford.edu)*

We examine a novel class of procurement auctions for single-minded bidders, in which the auctioneer selects a set of bids to be accepted subject to complicated feasibility constraints that preclude optimization-based winner determination. (This setting is inspired by the U.S. Federal Communication Commission's problem of buying out a subset of broadcast TV licenses to clear spectrum for broadband use while retuning the remaining stations into the remaining TV spectrum subject to interference constraints and a possible budget constraint.) Instead, we propose a class of computationally feasible "greedy deferred-acceptance heuristic" auctions for calculating both a feasible set of winning bids and "threshold" payments which induce strategy-proof bidding. The calculation iteratively rejects the "highest-scoring" bid that could still be feasibly rejected, with a bidder's score based on its bid value and possibly on the bids already rejected. (The latter dependence could be used to ensure that the total "threshold payments" satisfy the auctioneer's budget constraint, or that the winners do not get excessive payments.) This class of "deferred acceptance" heuristic auctions differs from the previously studied "deferred rejection/instance acceptance" heuristic auctions. In particular, we show that deferred-rejection heuristic auctions with threshold payments: (1) are equivalent to clock auctions with descending bidder-specific prices in which bidders who haven't quit are acquired at their final clock prices; (2) are (weakly) group strategy-proof, and so are the corresponding clock auctions for any information disclosure policy; (3) are outcome-equivalent to their paid-as-bid counterparts with the same allocation rule under full information: A paid-as-bid heuristic auction has a Nash equilibrium with the same outcome as its threshold-auction counterpart, which is a unique outcome surviving iterated deletion of weakly dominated strategies. In contrast, the Vickrey auction generally fails properties (1)–(3), except when bidders are substitutes, in which it can be implemented as a heuristic or clock auction.

## 3.30 Composable and Efficient Mechanisms

*Vasilis Syrgkanis (Cornell University, US, vasilis@cs.cornell.edu)*

Online markets require simple, and well-designed systems that work well even if users participate in multiple ones in parallel. Traditional mechanism design has considered mechanisms only in isolation, and the mechanisms it proposes tend to be complex and impractical. In contrast, players typically participate in various mechanisms that are run by different principals (e.g. different sellers on eBay or different ad-exchange platforms) and coordinating them to run a single combined mechanism is infeasible or impractical. Even the simple and elegant Vickrey auction loses some of its appeal when not in isolation: when the overall value of each player is a complex function of the outcomes of different Vickrey auctions, the global mechanism is no longer truthful.

We initiate the study of efficient mechanism design with guaranteed good properties even when players participate in multiple different mechanisms simultaneously or sequentially. We define the class of smooth mechanisms, related to smooth games defined by Roughgarden, that can be thought of as mechanisms that generate approximately market clearing prices. We show that smooth mechanisms result in high quality outcome in equilibrium both in the full information setting and in the Bayesian setting with uncertainty about participants, as well as in learning outcomes. Our main result is to show that such mechanisms compose well: smoothness locally at each mechanism implies efficiency globally.

## 3.31 Rounding and the Allocation of Indivisible Objects

*Rakesh V. Vohra (Northwestern University, Evanston, US, r-vohra@kellogg.northwestern.edu)*

The problem of allocating indivisible objects arises in the allocation courses, spectrum licenses, landing slots at airports and assigning students to schools. We propose a technique for making such allocations that is based on rounding a fractional allocation. Under the assumption that no agent wants to consume more than $k$ items, the rounding technique can be interpreted as giving agents lotteries over approximately feasible integral allocations that preserve the ex-ante efficiency and fairness properties of the initial fractional allocation. The integral allocations are only approximately feasible in the sense that upto $k-1$ more units than the available supply of any good is allocated.

## 3.32 Equilibria in the Challenge Tournament

*Bernhard von Stengel (London School of Economics, stengel@nash.lse.ac.uk)*

Arad and Rubinstein (2013) describe a challenge tournament where $n$ players have a binary choice and play a round-robin tournament where they score against each other randomly for a stake that depends on their choices. The player with the highest total score wins, with ties resolved randomly. They conjecture that for $n > 3$ the only equilibrium is that all players take the riskier choice. We propose an elementary proof of this conjecture based on a dominance argument.

## 4 Further Participants and Session Chairs

In addition to the speakers listed above, the following researchers participated in this Dagstuhl workshop, often co-authors of the given presentations:

- Giorgos Christodoulou (University of Liverpool, GB)
- Michal Feldman (Hebrew University, Jerusalem, IL) – session chair Thursday afternoon
- Felix Fischer (University of Cambridge, GB)
- Yannai A. Gonczarowski (Hebrew University, Jerusalem, IL)
- Penélope Hernández (University of Valencia, ES) – session chair Friday morning
- Martin Höfer (MPI für Informatik – Saarbrücken, DE)

- Max Klimm (TU Berlin, DE)
- Elias Koutsoupias (University of Oxford, GB) – session chair Tuesday afternoon
- Jeffrey MacKie-Mason (University of Michigan, US) – session chair Tuesday morning
- Hervé Moulin (Rice University – Houston, US) – session chair Wednesday morning
- Dimitrii V. Pasechnik (Nanyang Technical University, Singapore, SG)
- Rahul Savani (University of Liverpool, GB)
- Michael Schapira (Hebrew University, Jerusalem, IL)
- Éva Tardos (Cornell University, US) – session chair Monday afternoon
- Berthold Vöcking (RWTH Aachen, DE) – session chair Thursday morning
- Jens Witkowski (Universität Freiburg, DE)

## 5 Open Problems

Two open problem sessions were held on Tuesday and Thursday afternoon after the afternoon talks. Of the discussed research questions, the following is published here (the open problem is stated at the end of the following exposition; the topic was not presented in a talk but in the open problem session).

### 5.1 Fair Division of a Max-Flow

*Hervé Moulin (Rice University, Houston, US, moulin@rice.edu)*

The problem of fairly dividing a max-flow has a long history, going back to the work of Megiddo, Brown, Hall and Vohra, and others. In all applications there is a distribution network that is capacity constrained (capacities may be on nodes or edges), and agents need to be allocated different amounts of the commodity based on objective or subjective needs. While standard models of flows in networks are typically concerned with the optimization of an exogenous objective, there is a substantial subset of that literature where the goal is to not only maximize the quantity distributed, but also to ensure that the distribution is equitable. As an example, Brown (O.R., 1979) discussed a sharing problem motivated by the equitable distribution of coal during a prolonged coal strike. Other settings where similar problems come up is in distributing aid or relief during a catastrophic event (e.g., food during a famine).

The classical papers of Megiddo and Brown call a max-flow fair if it equalizes the allocation of shares between the relevant agents (located on some nodes or edges); they typically use the lexicographic ordering to select the fairest max-flow, and design efficient algorithms to find it. Crucially, they take an ethically neutral view of the structure of the graph: if we can give equal shares of the flow to all agents, we should do it, ignoring the differences in their connectivity altogether. This is appropriate in contexts where agents should not be held responsible for their connections/capacities, such as the distribution of relief supplies mentioned above. It is not appropriate when agents should be held responsible for their connections in the network, so that a "better connected", "more central", agent ought to receive a greater share of the resources. Think of the distribution of a workload between contractors with different abilities to perform certain jobs.

The formal property of "consistency" captures the influence of the network structure on the final division of the flow: each agent has a claim only on the subflows that he is connected to, and the division of a given subflow relies on the residual claims of the relevant agents (total claim minus shares in other subflows; as in the bargaining model of Kleinberg and Tardos). Consistency is a key component of several other axiomatic fair division models.

Jay Sethuraman and I have developed a fairly complete model for the case of bipartite graphs but extending the approach to a general flowgraph raises several conceptual difficulties that I would like to discuss with the workshop participants.

## 6 Organizational Issues

The following comments about the organization of the workshop and lessons learnt may be of general interest.

### 6.1 Invitations

By far the most time-consuming organizational issue was whom to invite and when. In the end, the seminar had 48 participants, which were well accommodated (very few junior people sharing rooms) while a second smaller seminar took place in parallel at Schloss Dagstuhl.

The selection of invitees was difficult because of the broad scope of the workshop across economics and computer science.

In order to explain the workshop to economists, Sergiu Hart as the organizer closest to economics sent a separate email to them explaining the "by invitation only" and funding model. In addition, we asked for a subjective probability of attendance in order to estimate the number of participants. Both were very helpful.

One suggestion we have is the possibility to "*overbook*" the seminar even if confirmation numbers are relatively high (within small limits, of course). The number of people confirming after the first round of invitations was higher than expected. This left almost no room for a second round of invitations until a few months before the start of the workshop when people had to decide (and, as expected, some could not come after all). At this point (about six months before the meeting), it would have been good if we had been allowed to invite one or two senior researchers that we had not invited in the first round or which were suggested to us by colleagues, because in the end we did have the space and at worst could have asked more people to share rooms. For one person the eventual invitation came too late. However, the high caliber of those who had confirmed continued to make the workshop very attractive. The overall coordination with the Dagstuhl service team (Annette Beyer) was excellent.

In the end, we were happy with the selection that we could make: If in doubt, preferring an early-career researcher to a more established one. In addition, PhD students of the organizers could fill last-minute open places (in the feedback for the workshop, one participant suggested not to allow this because of a perceived restriction of the diversity of the group, but we think this is one of the privileges the organizers should enjoy, and because it allows to use free spaces with flexibility).

Moreover, the suggested "affirmative action" of Dagstuhl was successful in the sense that we had a relative large number of female participants (10 out of 48). In addition, three of them had a baby or small child looked after by the nanny of the Center, so families were well accommodated. Dagstuhl is exemplary here!

## 6.2 During the Workshop

We wanted to have a workshop with sufficient free time for informal talks and collaboration. For that purpose, we wrote about two months before the workshop an email to the effect of "if you really want to give a talk, please send us your suggested topic". This was deliberately not very encouraging and resulted in 12 suggested topics, not enough to fill the workshop.

We separately encouraged those whom we knew as particularly good speakers to propose a topic.

A second, more short-term email resulted in over 20 further suggestions, so that we had to make a choice but in the end accommodated nearly everyone.

We allocated 32 talks (with the abstracts above) of 30 minutes each to the following slots (morning + afternoon): 1+5 (Monday), 5+3 (Tuesday), 5+0 (Wednesday), 5+3 (Thursday), 5+0 (Friday), plus 2 open problem sessions at the end of the afternoons of Tuesday and Thursday.

Monday morning started with a "lightning talk" session where every speaker sent in advance one or two PDF slides about themselves and introduced themselves for at least one minute but no longer than two minutes. This worked very well and was considered a success. One suggestion is a large *countdown clock* (maybe in software) that would have helped to put the speakers more at ease that they use their time properly, because a talk time of at most two minutes is unfamiliar to most people. Only the very junior participants talked about themselves not long enough (i.e., less than one minute). The length restriction allowed the lightning talks to proceed in two sessions of about one hour each; five minutes per talk would have been too much.

In fact, the lightning talk already provided information about the quality of a speaker. One suggestion was to let participants *vote* on the permitted time of a talk for each speaker after the lightning talks, which could work as follows: Standard slot speaking time is 20 minutes, but, say, 10 slots of 30 minutes are also available, and everybody would vote for their most preferred speakers for those longer slots. In that way, winning a longer slot time would be an honour, rather than a shorter time an embarrassment. (The full vote count would not be made public, only the winners.)

Because of the lightning talks, Monday had one scientific talk in the morning and five afternoon talks. Every other day had five morning talks, and the only other afternoon talks were three each (plus an open problems session) on Tuesday and Thursday from 16:00, so that a large part of the afternoon and every evening was free for personal collaboration. This was highly appreciated.

In retrospect, we could have possibly been more selective and dictatorial in reducing or lengthening individual talk times, or inviting longer tutorials (apart from the suggested vote after the lightning talks). On the other hand, the equal talk length allowed us to make these decisions at the workshop with ease (with a Google Docs spreadsheet for joint editing). Only the Monday schedule was decided shortly in advance of the meeting. At the workshop, we could also discuss the content of some presentations with those participants who wanted advice, which was helpful. In addition, the topics followed in relatively random order, which gave each day variety and allowed everyone to listen regularly to a topic close to their interests.

All talk topics and most abstracts were sent to us ahead of time, but only about half of them were uploaded as system-wide materials. As suggested in the feedback for the workshop, we should have made this *mandatory* for being allowed to give a talk, for the benefit of the participants. On the other hand, people attended most talks, despite of limited advance information.

There was no need for working groups or parallel sessions because the group was largely interested in and listening to all topics.

The free Wednesday afternoon was used for a local hike, and we were very lucky to have good weather throughout the workshop. Most took part and liked it. A small number of participants would have preferred a more exciting sightseeing tour of, for example, the Völklinger Hütte, but the long bus journey and the fact that a few had already seen it made us decide against it. The site office was very helpful with advice and local maps.

The quality of talks was very high, in line with our expectations and selection of the participants.

## Participants

- Yakov Babichenko
CalTech, US
- Simina Branzei
Aarhus University, DK
- Markus Brill
TU München, DE
- Jing Chen
Institute of Advanced Study – Princeton, US
- Giorgos Christodoulou
University of Liverpool, GB
- Constantinos Daskalakis
MIT, US
- Gabrielle Demange
Paris School of Economics, FR
- Nikhil R. Devanur
Microsoft – Redmond, US
- Shahar Dobzinski
Weizmann Inst. – Rehovot, IL
- Paul Dütting
EPFL – Lausanne, CH
- Edith Elkind
Nanyang TU – Singapore, SG
- Michal Feldman
The Hebrew University of Jerusalem, IL
- Felix Fischer
University of Cambridge, GB
- Paul W. Goldberg
University of Liverpool, GB
- Yannai A. Gonczarowski
The Hebrew University of Jerusalem, IL
- Sergiu Hart
The Hebrew University of Jerusalem, IL
- Jason D. Hartline
Northwestern University – Evanston, US
- Penelope Hernandez Rojas
University of Valencia, ES
- Martin Hoefer
MPI für Informatik – Saarbrücken, DE
- Nicole Immorlica
Northwestern University – Evanston, US
- Ramesh Johari
Stanford University, US
- Thomas Kesselheim
Cornell University, US
- Max Klimm
TU Berlin, DE
- Elias Koutsoupias
University of Oxford, GB
- Kevin Leyton-Brown
University of British Columbia – Vancouver, CA
- Katrina Ligett
CalTech – Pasadena, US
- Brendan Lucier
Microsoft Research New England – Cambridge, US
- Jeffrey MacKie-Mason
University of Michigan, US
- Vangelis Markakis
Athens University of Economics and Business, GR
- Vahab Mirrokni
Google – New York, US
- Hervé Moulin
Rice University – Houston, US
- Rudolf Müller
Maastricht University, NL
- Sigal Oren
Cornell University, US
- Mallesh Pai
University of Pennsylvania, US
- Dimitrii V. Pasechnik
Nanyang TU – Singapore, SG
- Aaron Roth
University of Pennsylvania, US
- Tim Roughgarden
Stanford University, US
- William H. Sandholm
University of Wisconsin – Madison, US
- Rahul Savani
University of Liverpool, GB
- Guido Schäfer
CWI – Amsterdam, NL
- Michael Schapira
The Hebrew University of Jerusalem, IL
- Ilya R. Segal
Stanford University, US
- Vasilis Syrgkanis
Cornell University, US
- Éva Tardos
Cornell University, US
- Berthold Vöcking
RWTH Aachen, DE
- Rakesh V. Vohra
Northwestern University – Evanston, US
- Bernhard von Stengel
London School of Economics, GB
- Jens Witkowski
Universität Freiburg, DE

Report from Dagstuhl Seminar 13162

# Pointer Analysis

**Edited by**

**Ondřej Lhoták[1], Yannis Smaragdakis[2], and Manu Sridharan[3]**

1 **University of Waterloo, CA,** `olhotak@uwaterloo.ca`
2 **University of Athens, GR,** `yannis@smaragd.org`
3 **IBM TJ Watson Research Center – Yorktown Heights, US,** `msridhar@us.ibm.com`

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13162 "Pointer Analysis". The seminar had 27 attendees, including both pointer analysis experts and researchers developing clients in need of better pointer analysis. The seminar came at a key point in time, with pointer analysis techniques acquiring sophistication but still being just beyond the edge of wide practical deployment. The seminar participants presented recent research results, and identified key open problems and future directions for the field. This report presents abstracts of the participants' talks and summaries of the breakout sessions from the seminar.

**Seminar** 14.–19. April, 2013 – www.dagstuhl.de/13162
**1998 ACM Subject Classification** F.3.2 Logics and Meanings of Programs: Semantics of Programming Languages: Program Analysis, D.3.4. Programming Languages: Processors: Compilers
**Keywords and phrases** pointer analysis, points-to analysis, alias analysis, static analysis, programming languages
**Digital Object Identifier** 10.4230/DagRep.3.4.91
**Edited in cooperation with** Gogul Balakrishnan

## 1 Executive Summary

*Ondřej Lhoták*
*Yannis Smaragdakis*
*Manu Sridharan*

The Dagstuhl seminar on *Pointer Analysis* brought together experts in pointer analysis and researchers building demanding clients of pointer analysis, with the goal of disseminating recent results and identifying important future directions. The seminar was a great success, with high-quality talks, plenty of interesting discussions, and illuminating breakout sessions.

### Research Context

Pointer analysis is one of the most fundamental static program analyses, on which virtually all others are built. It consists of computing an abstraction of which heap objects a program variable or expression can refer to. Due to its importance, a large body of work exists on pointer analysis, and many researchers continue to study and develop new variants. Pointer analyses can vary along many axes, such as desired precision, handling of particular language features, and implementation data structures and optimizations. Given the subtle implications of these design choices, and the importance of low-level details often excluded from conference-length papers, it can be difficult even for pointer analysis experts to understand the relationship between different analysis variants. For a non-expert aiming to use pointer analysis in a higher-level client (for verification, optimization, refactoring, etc.), choosing the right analysis variant can be truly daunting.

Pointer analysis is a mature area with a wealth of research results, at a temptingly close distance from wide practical applicability, but not there yet. The breakout application of precise analysis algorithms has seemed to be around the corner for the past decade. Although research ideas are implemented and even deployed in limited settings, several caveats always remain. These include assumptions about client analyses (i.e., the pointer analysis algorithm is valid only under assumptions of how the information will be used), assumptions about the analyzed program (e.g., that some language features are absent or that their presence does not affect the analysis outcome), assumptions about modularity (e.g., that the code to be analyzed constitutes the whole program), etc. The right engineering packaging of pointer analysis algorithms as well as a convenient characterization of their domain of applicability are still elusive.

In this light, the seminar aimed to emphasize the relationship of pointer analysis algorithms with client analyses, as well as practical deployment issues. The seminar brought together researchers working on pointer analysis for various programming languages with researchers working on key analysis clients. Our main goals were (1) to deepen understanding of the relationships between existing pointer analysis techniques, and (2) to gain a better understanding of what pointer analysis improvements are required by clients, thereby setting an exciting agenda for the area going forward.

## Seminar Format

Our seminar employed a somewhat unusual format for participant talks, intended to encourage a deeper discussion of each participant's work. Each participant was alloted a 40-minute slot to present their work, consisting of 20 minutes of presentation and 20 minutes of discussion. The presentation and discussion times in each slot were enforced using a chess clock: when a question arose during a talk, the clock was "flipped" to discussion time, and after the discussion, it was flipped back to speaker time. (The times were not very strictly enforced; in some cases, the audience would "donate" time to the speaker to complete his/her presentation.) This format had two key benefits:

- It enabled discussion to freely occur during the talk, removing the worry that the speaker would have no time left to complete his/her presentation.
- It encouraged the audience to ask more questions, in order to "use up" the alloted audience time.

Overall, the format was very successful in encouraging good discussion, and most participants enjoyed it.

In addition to talks, we held four 90-minute breakout sessions. The session topics were proposed by participants before and during the seminar and voted on by participants. The sessions were scheduled two at a time, and participants could choose which session to attend. The discussions held in these sessions were quite illuminating, and are summarized in Section 4 of this report. Finally, the last half-day of the seminar was spent on additional discussion of the breakout session topics, and on an initial effort to collectively improve the Wikipedia article on pointer analysis.[1]

[1] See http://en.wikipedia.org/wiki/Pointer_analysis.

## Seminar Results

Recent advancements in pointer analysis have come from several different directions:

- Formulations (CFL, Datalog)—highly-complex analyses have been specified in terms of consise specifications, by utilizing declarative notations.
- Greater precision—interesting analyses that maintain finer-grained abstractions while maintaining scalability have been invented.
- Optimizations—data structures such as BDDs have been used to make complex analyses feasible.
- Demand-driven, refinement—the analysis problem has been specialized effectively when pointer information only needs to be computed for select program sites.
- Partial programs—analyses have been formulated to work without fully analyzing all libraries, or even all application code.

Such advances were discussed in detail during many participant talks in the seminar, and in the breakout sessions.

Recent work in pointer analysis has been driven by new clients for the analysis and by new programming languages. Along with ongoing use of pointer analysis in traditional optimizing compilers, recent years have seen many other clients emerge that require effective pointer analysis, e.g., in the areas of program verification and bug finding, refactoring, and security. These clients were well-represented by seminar attendees, who gave many interesting talks on novel uses of pointer analysis (particularly in the security domain). The rich exchanges between researchers building novel clients and those with pointer analysis expertise were one of the most valuable aspects of the seminar. Additionally, one breakout session covered the difficulties in designing an effective general pointer-analysis API that is suitable for a wide variety of clients.

Mainstream programming has been transitioning to increasingly heap-intensive languages: from C-like languages to object-oriented languages like Java and C#, and more recently to scripting languages like JavaScript and Ruby. As languages become more heap-intensive, the need for effective pointer analysis is greater, motivating continuing work in this area. The seminar talks covered a wide and diverse set of languages, each with its own considerations. A few talks covered pointer analysis for higher-level languages such as JavaScript and MATLAB. Such languages are becoming increasingly popular, and they are very heap-intensive compared to C-like languages, motivating the need for better pointer analysis. A couple of talks presented techniques for control-flow analysis of functional languages like Scheme. While the pointer analysis and control-flow analysis communities often use similar techniques, the relationships between the techniques is often obscured by differing terminology and presentation styles. The presentations on control-flow analysis and the corresponding discussions were helpful in bridging this gap.

The seminar included a good deal of discussion on practical issues with pointer analysis, including evaluation methodologies and issues arising in real-world deployments. A key theme that arose from these discussions was the need for pointer analysis to be at least partially unsound to be useful in practice, and how this need for unsoundness has not been explained properly in the literature. Analyses that made soundness compromises for practicality were deemed "soundy," a tongue-in-cheek term that caught on quickly among participants. Recently, some seminar participants presented a well-received PLDI Fun and Interesting Topics (FIT) talk on the notion of "soundiness," and several participants have agreed to collectively co-author a publishable document on the topic.

## Conclusions

Overall, the *Pointer Analysis* Dagstuhl seminar was a great success. The seminar brought together 27 researchers from both academia and industry (including Google, IBM, Microsoft, NEC), with a good mix of junior and senior researchers. There were many interesting talks, with deep discussion facilitated by the chess clock time maintenance. The seminar facilitated interaction between pointer analysis experts and researchers building novel clients (a key goal for the seminar from the beginning), and also between researchers working on analyses for a variety of languages. Breakout sessions enabled further discussion of certain particularly interesting topics. In particular, there were invaluable discussions of many practical issues that often get short shrift in conference papers. These discussions sparked the notion of "soundiness," which may have broader impact via a future publication.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Does it have to be so hard?

*José Nelson Amaral (University of Alberta, CA)*

The brightest and most capable students are the ones that undertake research in the area of pointer and reference analysis. Yet, often they take a long time to graduate, and sometimes they produce fewer research results than students that undertake research in different areas. Moreover, often the outcome of their research is incomplete and unsatisfying. On the other hand, many of the papers published in the area — even the best ones that appear in the top venues — leave readers and reviewers with a sense of a good work that is incomplete. In this discussion we look at several shortcomings in currently published papers in pointer and reference analysis. Then we will talk about some undertakings by the community that could change this situation, making research in analysis more rewarding and productive for students and practicioners, and accelerating the speed of innovation in this area.

### 3.2 Scalable and Precise Program Analysis at NEC

*Gogul Balakrishnan (NEC Laboratories America, Inc. – Princeton, US)*

**Joint work of** Balakrishnan, Gogul;Ganai, Malay; Gupta, Aarti; Ivancic, Franjo; Kahlon Vineet, Li, Weihong; Maeda, Naoto; Papakonstantinou, Nadia; Sankaranarayanan, Sriram; Sinha, Nishant; Wang, Chao
**Main reference** G. Balakrishnan, M. K. Ganai, A. Gupta, F. Ivancic, V. Kahlon, W. Li, N. Maeda, N. Papakonstantinou, S. Sankaranarayanan, N. Sinha, C. Wang, "Scalable and precise program analysis at NEC," in Proc. of 10th Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD'10), pp. 273–274, IEEE, 2010.
**URL** http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5770960

In this talk, I will briefly present the program analysis tools that we have developed at NEC Labs, and describe how pointer analysis is used in these tools. Specifically, I will talk about Varvel and ARC++. Varvel is tool for finding bugs in C and C++ programs, and is based on static analysis and model checking. ARC++ is a tool to find bugs in C++ programs based on user-specified error patterns.

#### References

1 G. Balakrishnan, M. K. Ganai, A. Gupta, F. Ivancic, V. Kahlon, W. Li, N. Maeda, N. Papakonstantinou, S. Sankaranarayanan, N. Sinha, and C. Wang. Scalable and precise program analysis at nec. In *FMCAD*, pages 273–274, 2010.
2 F. Ivancic, G. Balakrishnan, A. Gupta, S. Sankaranarayanan, N. Maeda, H. Tokuoka, T. Imoto, and Y. Miyazaki. Dc2: A framework for scalable, scope-bounded software verification. In *ASE*, pages 133–142, 2011.
3 P. Prabhu, N. Maeda, G. Balakrishnan, F. Ivancic, and A. Gupta. Interprocedural exception analysis for C++. In *ECOOP*, pages 583–608, 2011.
4 J. Yang, G. Balakrishnan, N. Maeda, F. Ivancic, A. Gupta, N. Sinha, S. Sankaranarayanan, and N. Sharma. Object model construction for inheritance in C++ and its applications to program analysis. In *CC*, pages 144–164, 2012.

## 3.3 Challenges in vulnerability detection for the Java runtime library

*Eric Bodden (TU Darmstadt, DE)*

**Joint work of** Bodden, Eric; Hermann, Ben; Lerch, Johannes

In this talk I will discuss a recent client analysis that we use to detect vulnerabilities in the Java runtime library. I will discuss challenges this analysis poses in terms of managing calling contexts and different analysis directions. In particular, it currently appears challenging to synchronize with each other a forward and backward analysis in such a way that they both only consider common calling contexts.

## 3.4 Precise Heap Reachability by Refutation Analysis

*Bor-Yuh Evan Chang (University of Colorado – Boulder, US)*

**Joint work of** Blackshear, Sam; Chang, Bor-Yuh Evan; Sridharan, Manu
**Main reference** S. Blackshear, B.-Y.E. Chang, M. Sridharan, "Thresher: Precise Refutations for Heap Reachability," in Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI'13), pp. 275–286, ACM, 2013.
**URL** http://doi.acm.org/10.1145/2462156.2462186

Precise heap reachability information that can be provided by a points-to analysis is needed for many static analysis clients. However, the typical scenario is that the points-to analysis is never quite precise enough leading to too many false alarms in the client. Our thesis is not that we need more precise up-front points-to analyses, but rather we can design after-the-fact triage analyses that are effective at refuting facts to yield targeted precision improvements. The challenge that we explore is to maximally utilize the combination of the up-front and the after-the-fact analyses.

We have investigated refutation analysis in the context of detecting statically a class of Android memory leaks. For this client, we have found the necessity for an analysis capable of path-sensitive reasoning interprocedurally and with strong updates–a level of precision difficult to achieve globally in an up-front manner. In contrast, our approach applies a refutation analysis that mixes a backwards symbolic execution with results from the up-front points-to analysis to prune infeasible paths quickly.

## 3.5 Precise and Fully-Automatic Verification of Container-Manipulating Programs

*Isil Dillig and Thomas Dillig (College of William and Mary)*

One of the key challenges in automated software verification is obtaining a conservative yet sufficiently precise understanding of the contents of data structures in the heap. A particularly important and widely-used class of heap data structures is containers, which support operations such as inserting, retrieving, removing, and iterating over elements. Examples of containers include arrays, lists, vectors, sets, maps, stacks, queues, etc.

In this talk, we will describe a sound, precise, scalable, and fully-automatic static analysis technique for reasoning about the contents of container data structures. This technique is capable of tracking position-value and key-value correlations, supports reasoning about arbitrary nestings of these data structures, and integrates container reasoning directly into a heap analysis, allowing, for the first time, the verification of complex programs that manipulate heap objects through container data structures. More specifically, we will describe a symbolic heap abstraction that augments a graph representation of the heap with logical formulas and that reduces some of the difficulty of heap reasoning to standard logic operations, such as existential quantifier elimination and satisfiability. I will present experimental results demonstrating that our technique is very useful for verifying memory safety in complex heap- and container-manipulating C and C++ programs that use arrays and other container data structures from the STL and QT libraries.

## 3.6 The End of Pointer Analysis?

*Julian Dolby (IBM TJ Watson Research Center, Hawthorne, USA)*

Pointer analysis means computing an approximation of the possible objects to which any program variable may refer; it has traditionally been done by conservatively approximating all possible data flow in the program, resulting in a conservative approximation of the objects held by any variable. This has always been a bit fake—no tools soundly approximates all possible reflective and JNI behavior in Java, for instance—but even the comforting illusion of soundness has become unsustainable in the world of framework- and browser-based Web applications. The frameworks are built on ubiquitous complex reflective behavior, and the browser appears as a large, complex, poorly-specified native API; the frameworks and the applications themselves are written in JavaScript, the lingua franca of the Web, the dynamic nature of which gives pointer analysis no help. Whether this world can be analyzed soundly is perhaps technically still an open problem, but the prognosis seems grim at best.

We have been exploring deliberately unsound analyses which make no attempt to approximate all possible data flow in a program; certain constructs are ignored not because they are unimportant, but simply because they are too hard. The tradeoff is now between how much can we ignore and still provide useful information versus how little can we ignore and still be tractable in practice. The good news so far is that there appear to be good tradeoffs, at least for a range of applications supporting IDE services. I will discuss recent and ongoing work in providing key information for IDE services: callgraphs and smart completions.

## 3.7 The Business of Pointer Analysis

*Samuel Z. Guyer (Tufts University)*

Over the past few years I have had the opportunity to work with a company that relies on pointer analysis as part of its core business – finding security vulnerabilities in software. I have worked closely with them to select and implement algorithms from the literature, and I have been able to see how these algorithms work (or don't work) at an industrial scale. Some of the most interesting issues I have encountered, however, are not related to the question of "Does it work on real software?" In this talk I will describe some of the challenges of deploying sophisticated analyses for commercial purposes. They are important and research-worthy problems that have not, to my knowledge, received much attention in the academic community.

## 3.8 Pointer analysis for dynamic information flow control

*Christian Hammer (Universität des Saarlandes)*

Dynamic information flow control is a powerful technique to ensure that confidential data cannot leak illicitly, and that untrusted data must not be used for trusted computations. However, since the standard security policy noninterference is a 2-trace-property, it cannot be enforced soundly and precisely by looking at one execution trace alone. One must either use conservative approximations or resort to static analysis about variables that might be modified in an alternative branch. This talk will present these challenges and how pointer analysis for a dynamic language like JavaScript, while challenging, is imperative to improve the precision of dynamic information flow.

## 3.9 Pointer Analysis Meets MATLAB

*Laurie J. Hendren (McGill University, CA)*

**Joint work of** Hendren, Laurie J.; Lameed, Nurudeen, Doherty, Jesse; Dubrau, Anton; Radpour, Soroush
**URL** http://www.sable.mcgill.ca/mclab

MATLAB is a dynamic array-based language commonly used by students, scientists and engineers. Although MATLAB has call-by-value semantics and no explicit pointers, there are many flow analysis problems that are similar to pointer analysis. In this talk I discussed why it is important for our research community to work on programming languages like MATLAB. I then outlined the key flow analysis problems that need to be solved and gave a summary of what my research group has accomplished and what we plan to work on in the future (http://www.sable.mcgill.ca/mclab).

## 3.10 The Approximations vs. Abstractions Dilemma in Pointer Analysis

*Uday Khedker (Indian Institute of Technology, Mumbai, India)*

Given the vital importance of pointer analysis and the inherent difficulty of performing precise pointer analysis for practical programs, a large fraction of pointer analysis community has come to believe that compromising on precision is necessary for scalability and efficiency. This is evidenced by the fact that a large number of reported investigations in pointer analysis involve a significant amount of engineering approximations.

We find it hard to accept this assumption as the final inevitability. We believe that a lot could be gained by exploring a science of pointer analysis that tries to build clean abstractions. In our opinion, this is a road less travelled in pointer analysis. Without undermining the engineering efforts, we propose that a search for approximations should begin only after building clean abstractions and not before it. The talk describes our efforts in this direction.

## 3.11 Incomplete Program Analysis

*Ondřej Lhoták (University of Waterloo, CA)*

**Joint work of** Lhotak, Ondrej; Ali, Karim; Naeem, Nomair

Points-to analyses requiring the whole program are inefficient, hard to get right, non-modular, and fragile. This severely hinders development of client analyses and practical adoption. I discuss two possible solutions: the access path abstraction, which does not require the analysis to know about all allocation sites in the program, and the separate compilation assumption, which enables sound yet precise analysis of an application without analysis of its separately compiled libraries.

## 3.12 Challenges in Pointer Analysis of JavaScript

*Benjamin Livshits (Microsoft Research – Redmond)*

This talk two specific challenges that arise in the process of doing – or attempting to do – static analysis for JavaScript programs. The first is that JavaScript programs on the web are not static – far from it – they're in fact streaming. As such, the notion of whole program analysis needs to be reevaluated and perhaps discarded in favor of incrementality. Incremental pointer analysis for JavaScript is addressed in the Gulfstream project.

The second challenge is that of analyzing programs surrounded by complex *environments*, such as the DOM API, or node.js. Understanding the surrounding frameworks is both challenging and necessary. This is the topic of this talk and an upcoming FSE 2013 paper.

## 3.13 Comparing Different Points-To Analyses

*Welf Löwe (Linnaeus University – Växjö)*

Comparing the accuracy of different points-to analysis approaches is important for us as the research community: it allows us to focus on successful approaches and to drop the less successful ones. However, comparing accuracy only works if the analyses are either strictly conservative or strictly optimistic. Unfortunately, only few such analyses exist in practice; most of them are conservative only on a subset of the languages they are designed for and, hence, neither conservative nor optimistic in general. Practical issues add to the problem of comparability: analyses are defined for different languages and versions and run-time systems thereof, and there are no commonly accepted standard benchmarking suites nor accuracy metrics defined. This makes it often impossible to take two research publications and reliably tell which one describes the more accurate points-to analysis.

In this talk, we discuss theoretical and practical issues with comparing points-to analyses and we suggest a methodology on how to benchmark them. We then and argue for a Gold Standard, i.e., a set of benchmark programs with known exact analysis results. Such a Gold Standard would allow assessing the exact accuracy of points-to analysis. Since such a Gold Standard cannot be computed automatically, it needs to be created semi-automatically by the research community. We suggest a methodology on how this could be achieved.

## 3.14 Towards a Quantitative Understanding of Heap Structure and Application to Analysis Design

*Mark Marron (Microsoft Research, Redmond, USA)*

This talk looks at two related questions (1) what kinds of heap structures and sharing relations appear in object-oriented programs and (2) how can this information be used to guide the design of a heap analysis. I will show results which indicate that in practice the heap is a relatively simple structure where the vast majority of sharing (aliasing) and shapes that are present can be described by a small number of simple concepts that are closely related to standard programming idioms. I will also outline a hybrid shape/points-to analysis, which is both precise and computationally lightweight, that was designed based on these quantitative results. These initial results demonstrate the potential for leveraging empirical data during the design, or evaluation, of a heap analysis and demonstrate the potential for further work on the quantitative characterization of the heaps that appear in real-world programs.

## 3.15 Control-flow analysis of higher-order programs

*Matt Might (University of Utah, US)*

Control-flow analysis of higher-order programs and pointer analysis share much in common with each other. This talk serves as a tutorial on the basic method in higher-order control-flow analysis–the modern formulation of Shivers k-CFA. It discusses one common enhancement–abstract garbage collection. The talk concludes with cultural differences between the control-flow analysis and pointer analysis communities, noting the years- or decades-long lag between problem discovery, decidability, tractability, feasibility and evaluation in the CFA community.

## 3.16 Inference and Checking of Context-sensitive Pluggable Types

*Ana Milanova (Rensselaer Polytechnic, US)*

We develop a framework for inference and checking of pluggable types, also known as type qualifiers. The framework allows us to formulate context-sensitive pluggable type systems (e.g., Ownership, Immutability, Taint, others) and infer and check types on large Java codes. The key novelty is 1) support for context sensitivity, and 2) a scalable inference engine, which allows type inference with zero or small number of user annotations. We formulate two analyses, traditionally powered by pointer analysis: 1) purity inference and 2) taint analysis, as type inference problems in our framework, and discuss our results.

## 3.17 Pointer Analysis for Refactoring JavaScript Programs

*Anders Møller (Aarhus University, DK)*

Modern IDEs support automated refactoring for many programming languages, but for dynamic languages, such as JavaScript, the tools are still primitive. This talk presents two approaches toward tool supported renaming refactoring for JavaScript: 1) using (almost sound) pointer-analysis for fully automatic refactorings, and 2) using a pragmatic variant of Steensgaard-style analysis for semi-automatic refactorings.

## 3.18 New Search Techniques for Query-Driven Dataflow Analysis

*Mayur Naik (Georgia Institute of Technology)*

A central problem in static analysis concerns how to balance its precision and cost. A query-driven analysis seeks to address this problem by searching for an abstraction that discards program details that are unnecessary for proving an individual query. I will describe our results and experience over the past four years addressing this problem in the context of query-driven dataflow analyses that are parametric in the abstraction. The abstraction is chosen from a large family that allow abstracting different parts of a program with varying precision. A large number of fine-grained abstractions enables an analysis to specialize to a query but poses a hard search problem in practice. Our main result is a set of new search techniques (black-box and white-box approaches, deterministic and randomized approaches, purely static and hybrid dynamic-static approaches) for new problems (minimal abstractions, necessary conditions, impossibility results) that show promise for realistic pointer-related analyses on medium-to-large Java programs from the Dacapo suite.

## 3.19 Sparse Analysis Framework

*Hakjoo Oh (Seoul National University, KR)*

**Joint work of** Oh, Hakjoo; Heo, Kihong; Lee, Wonchan; Lee, Woosuk; Yi, Kwangkeun
**Main reference** H. Oh, K. Heo, W. Lee, W. Lee, K. Yi, "Design and Implementation of Sparse Global Analyses for C-like Languages," in Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI'12), pp. 229–238, ACM, 2012.
**URL** http://dx.doi.org/10.1145/2254064.2254092

In this talk, I present a general method for achieving global static analyzers that are precise, sound, yet also scalable. Our method, on top of the abstract interpretation framework, is a general sparse analysis technique that supports relational as well as non-relational semantics properties for various programming languages. Analysis designers first use the abstract interpretation framework to have a global and correct static analyzer whose scalability is unattended. Upon this underlying sound static analyzer, analysis designers add our generalized sparse analysis techniques to improve its scalability while preserving the precision of the underlying analysis. Our method prescribes what to prove to guarantee that the resulting sparse version should preserve the precision of the underlying analyzer.

## 3.20 Empirical Evaluation of Points-To Analyses

*Erhard Plödereder (Universität Stuttgart, DE)*

**Joint work of** Frohn, Simon; Staiger-Stoehr, Stefan; Plödereder, Erhard

Over the years, we have run several experiments to evaluate the performance and precision of various points-to analyses in the context of global program analyses. The results have significantly influenced the approach taken towards points-to analyses in the Bauhaus program analysis framework. In a first part, this talk reports on the results of a comparative study (work by Simon Frohn) of the three well-known algorithms by Andersen, Das, and Steensgaard. The evaluation showed that the Das algorithm ran in near-linear time and was hardly more expensive than Steensgaard's but produced considerably more precise results. The Anderson algorithm lived up to its worst-case cubic performance, while not improving precision by much over the Das algorithm. Field sensitivity added considerable precision but also added a (small) factor to the execution time. An important insight was that the size of the points-to sets clearly correlated with the size of the test programs, refuting the hypothesis that locality would put constant bounds on the size of these sets. Quantitative results are shown in comparative charts both on execution time and performance metrics of the three algorithms including some variations. The second part of the talk describes the principles of our best-yet algorithm developed by Stefan Staiger-Stoehr in his Ph.D. thesis partly in response to the findings of Simon Frohn. As a flow-sensitive analysis it combines control flow analysis, call graph construction, SSA-construction and points-to analysis. It starts out presuming no data-flow effects from dereferenced pointers and then iterates towards a conservative fix-point using the gradually constructed points-to sets. Strong indirect updates are recognized and exploited. Proven to be of cubic complexity in MOP-accuracy and of bi-quadratic complexity in MOVP-accuracy, the algorithm is efficient enough to allow the analysis of programs of a quarter of a million lines of code in less than one hour, and often in less than 10 minutes. Again, empirical data on execution time and precision are presented, comparing variants that are flow-insensitive, flow-sensitive, and flow-sensitive with strong indirect updates. The data shows considerable reduction of the average size of points-to sets by flow-sensitive analysis but only marginalreductions from strong indirect updates. Execution time measurements were not entirely conclusive between linear and quadratic behavior for tests with up to 250.000 lines of code. Surprisingly, the strong indirect updates make a significant difference on the number of SSA-nodes generated (the algorithm uses Phi-nodes to represent weak updates). Compared to flow-insensitive analysis, up to 25% of SSA-nodes are avoided by flow-sensitive analysis, and up to 40% of strong indirect updates are recognized. A factor of 5 in execution time between the least and the most discriminating analyses makes the added effort well worthwhile, as the SSA-form is subsequently processed by various user-oriented program analyses.

## References

1 Simon Frohn. *Konzeption und Implementierung einer Zeigeranalyse für C und C++*. Diploma thesis, University of Stuttgart, Stuttgart, Germany, 2006

2 Stefan Staiger-Stoehr. *Kombinierte statische Ermittlung von Zeigerzielen, Kontroll- und Datenfluss*. doctoral dissertation, University of Stuttgart, Stuttgart, Germany, 2009

3 Stefan Staiger-Stoehr. *Practical Integrated Analysis of Pointers, Dataflow and Control Flow*. ACM Transactions on Programming Languages and Systems, Vol. 35, No. 1, Article 5, April 2013

## 3.21 Set-Based Pre-Processing for Points-To Analysis

*Yannis Smaragdakis (University of Athens, GR)*

**Joint work of** Smaragdakis, Yannis; Balatsouras, George; Kastrinis, George
**Main reference** Y. Smaragdakis, G. Balatsouras, G. Kastrinis, "Set-Based Pre-Processing for Points-To Analysis," in Proc. of the 28th Annual ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'13), to appear.

We present set-based pre-analysis: a virtually universal optimization technique for flow-insensitive points-to analysis. Points-to analysis computes a static abstraction of how object values flow through a program's variables. Set-based pre-analysis relies on the observation that much of this reasoning can take place at the set level rather than the value level. Computing constraints at the set level results in significant optimization opportunities: we can rewrite the input program into a simplified form with the same essential points-to properties. This rewrite results in removing both local variables and instructions, thus simplifying the subsequent value-based points-to computation. Effectively, set-based pre-analysis puts the program in a normal form optimized for points-to analysis.

Compared to other techniques for off-line optimization of points-to analyses in the literature, the new elements of our approach are the ability to eliminate statements, and not just variables, as well as its modularity: set-based pre-analysis can be performed on the input just once, e.g., allowing the pre-optimization of libraries that are subsequently reused many times and for different analyses. In experiments with Java programs, set-based pre-analysis eliminates 30% of the program's local variables and 30% or more of computed context-sensitive points-to facts, over a wide set of benchmarks and analyses, resulting in an over 20% average speedup.

## 3.22 Pointer Analysis for Probabilistic Noninterference

*Gregor Snelting (KIT – Karlsruhe Institute of Technology, DE)*

**Joint work of** Dennis Giffhorn; Gregor Snelting
**Main reference** D. Giffhorn, G. Snelting, "A New Algorithm for Low-Deterministic Security," Karlsruhe Reports in Informatics 06/2012, revised 2013.
**URL** http://pp.info.uni-karlsruhe.de/publication.php?id=giffhorn13lsod

Information Flow Control (IFC) analyses program source or machine code to discover possible violations of confidentiality (i.e. secret information is leaked to public ports) or integrity (i.e. critical computations are manipulated from outside). IFC algorithms must handle realistic programs in e.g. full Java; they must be provably sound (discover all potential leaks) and precise (produce no false alarms). For full Java, this not only requires flow- context- object- and field-sensitive analysis of explicit and implicit information flow, but also a precise pointer analysis, in particular for nested objects and exception handling. For concurrent programs, all potential leaks exploiting scheduling or interleaving effects must be discovered.

Probabilistic noninterference (PN) is the established technical criterion for IFC of concurrent programs operating on shared memory. IFC and PN algorithms can be based on non-standard type systems, or on program dependence graphs (PDGs). In any case, PN requires precise May-Happen-in-Parallel (MHP) information, which in turn requires precise pointer and alias analysis.

The talk presents basic examples for IFC and precision issues, and sketches PDG-based PN, as recently introduced by Giffhorn & Snelting. It then discusses challenges for pointer analysis in PN and MHP. It is shown that dynamic pushdown networks, as introduced by Müller-Olm et al., allow for lock-sensitive IFC and PN analysis, but require precise must-alias information.

**References**

1 C. Hammer, G. Snelting: *Flow-Sensitive, Context-Sensitive, and Object-sensitive Information Flow Control Based on Program Dependence Graphs.* International Journal of Information Security, Vol. 8 No. 6, 2009, pp. 399–422.

2 D. Giffhorn, G. Snelting: *A New Algorithm for Low-Deterministic Security.* Karlsruhe Reports in Informatics 06/2012, revised 2013, submitted for publication.

## 3.23 Pointer Analysis and Reflection

*Manu Sridharan (IBM TJ Watson Research Center – Yorktown Heights, US)*

**Joint work of** Sridharan, Manu; Dolby, Julian; Fink, Stephen J.; Chandra, Satish; Schaefer, Max; Tip, Frank

Over the last several years, our group at IBM Research has put considerable effort into building industrial-strength pointer analyses for Java and JavaScript programs. For both languages, one of the biggest challenges we faced was handling reflective code, particularly in libraries and frameworks. In my talk, I presented some of the problems we have faced, approaches we have tried, the strengths and weaknesses of these approaches, and some ideas on how to make progress going forward.

**References**

1 Max Schäfer, Manu Sridharan, Julian Dolby, and Frank Tip. Dynamic determinacy analysis. In *PLDI*, 2013.

2 Manu Sridharan, Shay Artzi, Marco Pistoia, Salvatore Guarnieri, Omer Tripp, and Ryan Berg. F4F: taint analysis of framework-based web applications. In *OOPSLA*, 2011.

3 Manu Sridharan, Julian Dolby, Satish Chandra, Max Schäfer, and Frank Tip. Correlation tracking for points-to analysis of JavaScript. In *ECOOP*, 2012.

4 Omer Tripp, Marco Pistoia, Stephen J. Fink, Manu Sridharan, and Omri Weisman. TAJ: effective taint analysis of web applications. In *PLDI*, 2009.

## 3.24 Modular combination of shape abstraction with numeric abstraction

*Xavier Rival (ENS, Paris)*

In this talk, we will discuss techniques to combine in a single static analysis pointer abstract domains with value abstract domains. Such combinations are required whenever pointer information is required in order to discover value (numeric, boolean...) properties and vice versa. We will show an abstract domain combination technique, which allows to build static analyzers in a modular manner, and let domains that abstract different kinds of elements exchange information.

## 3.25 Scaling flow analysis using big-step semantics

*Dimitris Vardoulakis (Google Inc. – Mountain View, US)*

**Joint work of** Vardoulakis, Dimitris; Shivers, Olin
**Main reference** CFA2: a Context-Free Approach to Control-Flow Analysis, ESOP 2010.
**URL** http://www.ccs.neu.edu/home/dimvar/papers/cfa2-lmcs11.pdf

Traditional flow analyses for higher-order languages, such as k-CFA, approximate programs as control-flow graphs. This approximation does not allow precise analysis of function calls and returns; during the analysis, a function may be called from one program point and return to a different one.

I will argue that call/return mismatch is undesirable in a flow analysis, especially in a higher-order setting, where function call and return is the central mechanism for control-flow transfer. Pushdown flow analyses, such as CFA2, offer greater precision than traditional analyses because they can match an unbounded number of calls and returns.

The increased precision of CFA2 is not obtained at the expense of scalability. I will discuss how to implement CFA2 as an abstract interpretation of a big-step operational semantics. The use of big-step semantics minimizes caching of abstract states (and thus memory usage) because it does not require one to remember the analysis results for each sub-expression in the program. It also makes the analysis faster because it requires fewer comparisons between abstract states than summarization-based analyses. I have implemented this analysis for JavaScript and used it to analyze all Firefox add-ons, with promising results.

### References

1 Dimitrios Vardoulakis. *CFA2: Pushdown Flow Analysis for Higher-Order Languages*. PhD dissertation, Northeastern University, August 2012.

# 4 Breakout Sessions

## 4.1 Better APIs for Clients

*José Nelson Amaral (University of Alberta, CA)*

This breakout session was motivated by the observation that it is difficult for a single person/group to design and implement a significant pointer/alias/reference analysis and then also implement meaningful clients to use the analysis and test its effectiveness on creating new opportunities for optimization based on the analysis results.

The initial idea is that if an interface could be defined between the analyses and the clients, then a new analysis could be tested on several existing clients and a new client could try to use several existing analyses.

However, after discussing the initial idea, we realised that we would need not a single interface, but rather two interfaces and a query system as shown in Fig. 1:

- **Client API**: interfaces with the clients and answers the questions that a client may ask
- **Analyzer API**: provides the fact that are generated by the analysis to the query system
- **Query System**: translate analysis facts into answers to client queries.

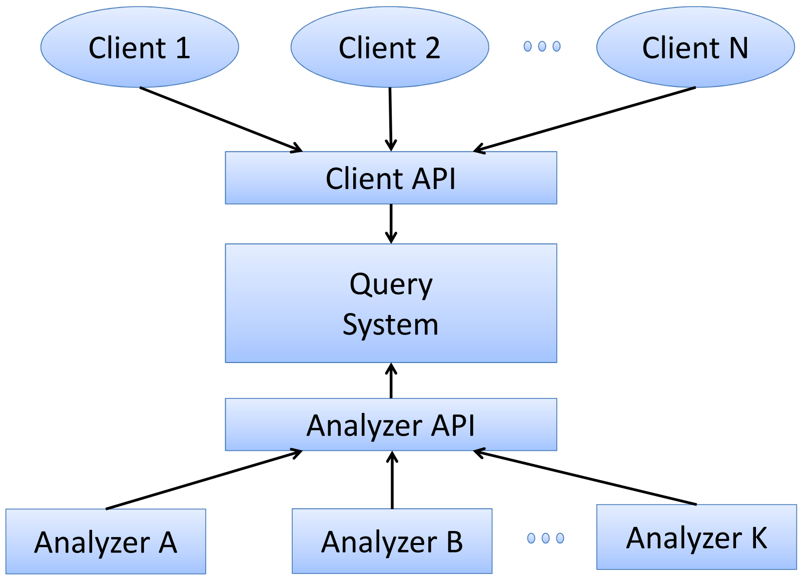


**Figure 1** Interfaces and the Query System for Pointer Analysis Clients.

The ensuing discussion made it obvious that the problem is significantly more complicated than the motivation and initial idea implied: both the analyzer and the client representations are usually tied to some intermediate representation of the code.

### Client API: Examples of Possible Functions

```
must_alias(x, y, pi, cj)
must_not_alias(x, y, pi, cj)
points_to_set(x,pi, cj)
```

where:

`x`, `y`: variables in the program (how to represent abstraction of heap locations?)

`pi`: a point in the program (can we represent a point in a program without tying it to a specific intermediate representation?)

`cj`: a context (how should contexts be represented?)

This simple notation does not address either path sensitivity or field sensitivity.

### Analyzer API

The analyzer API seems to need to be more complex than the client API and we did not look into it in detail.

### Fairness

While such a system could potentially be built to measure the effect of an analysis outcome on clients, it would probably be unwise to use it to measure how fast an analysis works with a client. For instance, if the client specifies a point in the program but the analyzer is flow insensitive, there is an overhead incurred by the framework that would not exist in a direct connection of the client with the analyzer without going through the APIs.

## 4.2 Pointer analyses for open programs (libraries/frameworks)

*Eric Bodden (TU Darmstadt, DE)*

We talked about two possible problem cases, one being the situation where one wants to summarize library information for speeding up client analyses, effectively preventing the library from being analyzed over and over again. The question is how to pre-analyze a library as much as possible, even though nothing is known about possible clients. One idea was to use Class Hierarchy Analysis (CHA) but it was noted that even this would be unsound, as types may be missing. Regular allocation-site-based analyses yield similar problems, also causing call edges to be missed. It was hence suggested to simply proceed as follows:

- perform a points-to analysis on a best-effort basis, persisting the points-to information that can be computed without any client code
- then, as the client code is analyzed, simply load the persisted analysis facts and continue the fixed point iteration from there

We also noted that summaries for points-to analyses might be hard to store and interpret, as they would probably need to involve specifications with complex access paths. Summaries for client analyses may make more sense but to date there is no efficient system for computing such summaries. Another problem with summaries in general is that callbacks create holes which the summary must accommodate for.

The second application scenario was analyzing a library for internal code vulnerabilities that could be exploited by malicious client code. Interestingly, in this scenario any sound analysis must make worst-case assumptions; so if there is a callback that could execute client code then the analysis better assume that this code could call any library code it has a handle to because that's exactly what an attacker could do. A correct but imprecise assumption would hence be that the callback could call any method at all in the library. Of course, this would not lead one anywhere. The art is hence to find out what object handles the client could get access to, and based on this information what functions the client could possibly call. An interesting observation was that in turn the library itself can only call methods it knows about, i.e., methods that are defined in some interface that is part of the library itself (unless the library uses reflection, that is).

## 4.3 Shape Analysis and Pointer Analysis: Working Together

*Bor-Yuh Evan Chang (University of Colorado – Boulder, US)*

This session focused on identifying the characteristics that would typically define an analysis as either a shape analysis or a pointer analysis. While both are static program analyses that infer relationships about memory addresses, they are mostly considered distinct sub-areas today. The intent of this session was first to better understand the technical and cultural differences between these two kinds of memory analyses and then by doing so to explore how to leverage the two perspectives together.

We first summarize the discussion about defining these two kinds of memory analyses. In the following, we try to speak in general terms, as certainly there are specific analysis algorithms that blend characteristics.

An initial, strawman definition of the difference was raised that is oversimplifying but still informative. Pointer analysis infers relationships about "stack pointers." Or more precisely, it infers relationships about static, named addresses (e.g., global and local variables). Shape analysis infers relationships about "heap pointers." Or more precisely, it infers relationships about dynamic, unnamed addresses (e.g., malloced addresses).

Of course, both kinds of analyses can derive relationships about both stack and heap pointers, and the most significant challenges for both arise from dynamically-allocated addresses.

Pointer analysis abstracts the heap of dynamically-allocated memory cells with an up-front, static partitioning. For a standard context-insensitive allocation site-based abstraction, the partitioning is clear: every dynamically-allocated address is bucketed into its allocation site and every partition is named by its allocation site. Adding context-sensitivity still fits this definition even though it seems a bit less clear: the partitions are more refined and not all partitions are necessarily named during the course of the analysis, but the partition to which a concrete memory cell belongs does not change during the course of the analysis. Overall, we called the pointer analysis approach "staticifying the dynamic."

Shape analyses typically vary the heap abstraction during the course of the analysis (e.g., the partitioning of the heap may vary across program points). Though not necessarily a prerequisite, materialization and summarization operations that change the heap partitioning arise from this perspective. Materialization decides how a heap partition should be split, while summarization decides how heap partitions should be merged.

An important point raised was that the differences described above are orthogonal to the distinction between store-based versus store-less abstractions, even though some combinations are less represented in the literature than others.

We identified some cultural differences. A sub-area, often tied to shape analysis, tends to focus on what's possible with no or little loss of precision. For example, it is not uncommon for such analyses to simply halt when a strong update is not possible (i.e., the concrete cell being updated cannot be identified). Another sub-area, often tied to pointer analysis, tends to focus on handling all programs and all language features with a best-effort approach to precision: weak updates must be supported even if we wish to avoid them. An important observation is that these perspectives need not be tied to whether an analysis is deriving shape or pointer information.

In summary, it is oversimplifying to say that "shape analysis is more precise pointer analysis" or "pointer analysis is more scalable shape analysis." These clichés have a kernel of truth, but it seems likely that further advances can be made by borrowing perspectives and ideas from each sub-area.

## 4.4 Practical Aspects of Pointer Analysis

*Manu Sridharan (IBM TJ Watson Research Center - Yorktown Heights, US)*

This breakout session focused on discussion of aspects of pointer analysis typically not discussed in conference papers, such as negative results and low-level details. A number of interesting topics were covered, as summarized below:

- For Java points-to analysis, differences in reflection handling can make a dramatic impact on scalability, which can be non-obvious just from reading a paper.
- Although BDDs are essential in making some straightforward analysis specifications scalable, the participants could find no compelling case for using BDDs in *highly tuned* points-to analysis, at the present time. (This is a delicate balance that may shift in the future, however.) BDDs have high constant-factor overheads that can have a significant performance impact, and other techniques, such as careful writing of Datalog rules or a shared bit-vector repository, can be as effective in eliminating redundancy from the results.
- Cycle elimination seems to yield little to no benefit for Java points-to analysis, as type filters lead to many fewer cycles than for large C programs. However, other opportunities for compression of the constraint graph remain, as discussed, e.g., by Hardekopf and Lin [1].
- The topic of debugging point-to analysis implementations and tracking down the root cause of imprecision was discussed. Suggestions included:
  - standard practices for good software engineering (unit tests and assertions)
  - comparing analysis variants with well-known precision relationships, e.g., ensuring that a context-sensitive analysis is strictly more precise than the context-insensitive version.
  - comparing results from different analysis frameworks; while this may involve significant work due to differing handling of language constructs, the work can be worthwhile. Additionally, it was suggested that a good fuzz testing harness for pointer analyses would be highly useful. For tracking down imprecision, delta debugging was suggested, though this becomes tricky in the presence of multiple files.
- For points-to analysis of JavaScript and other dynamic languages, differing string handling was identified as another obscure source of very large differences in analysis precision and performance.
- Finally, participants listed known production systems making use of sophisticated points-to analysis. Examples included link-time optimization in some commercial compilers, the JavaScript JIT compiler in Mozilla Firefox,[2] Veracode's analysis engine,[3] JavaScript taint analysis in IBM Security AppScan Source Edition,[4] and in an architecture analysis system.

### References

1 Ben Hardekopf and Calvin Lin. Exploiting Pointer and Location Equivalence to Optimize Pointer Analysis. In *SAS*, 2007.

[2] https://wiki.mozilla.org/IonMonkey
[3] http://www.veracode.com
[4] http://www-03.ibm.com/software/products/us/en/appscan-source/

## Participants

- Jose Nelson Amaral
University of Alberta, CA
- Gogul Balakrishnan
NEC Lab. America, Inc. – Princeton, US
- Eric Bodden
TU Darmstadt, DE
- Bor-Yuh Evan Chang
University of Colorado – Boulder, US
- Isil Dillig
College of William and Mary, US
- Thomas Dillig
College of William and Mary, US
- Julian Dolby
IBM TJ Watson Res. Center – Hawthorne, US
- Samuel Z. Guyer
Tufts University, US
- Christian Hammer
Universität des Saarlandes, DE
- Laurie J. Hendren
McGill University, CA
- Uday Khedker
Indian Institute of Technology – Mumbai, IN
- Ondrej Lhotak
University of Waterloo, CA
- Benjamin Livshits
Microsoft Res. – Redmond, US
- Welf Löwe
Linnaeus University – Växjö, SE
- Mark Marron
Microsoft Res. – Redmond, US
- Matt Might
University of Utah, US
- Ana Milanova
Rensselaer Polytechnic, US
- Anders Moeller
Aarhus University, DK
- Mayur Naik
Georgia Inst. of Technology, US
- Hakjoo Oh
Seoul National University, KR
- Erhard Plödereder
Universität Stuttgart, DE
- Xavier Rival
ENS – Paris, FR
- Yannis Smaragdakis
University of Athens, GR
- Gregor Snelting
KIT – Karlsruhe Institute of Technology, DE
- Manu Sridharan
IBM TJ Watson Research Center – Yorktown Heights, US
- Bjarne Steensgaard
Microsoft Res. – Redmond, US
- Dimitris Vardoulakis
Google Inc. – Mountain View, US

Report from Dagstuhl Seminar 13171

# Customizing Service Platforms

**Edited by**

**Luciano Baresi[1], Andreas Rummler[2], and Klaus Schmid[3]**

1 **Politecnico di Milano, IT,** luciano.baresi@polimi.it
2 **SAP Research Center – Dresden, DE,** andreas.rummler@sap.com
3 **Universität Hildesheim, DE,** schmid@sse.uni-hildesheim.de

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13171 "Customizing Service Platforms". The aim of the seminar was to bring together researchers from different areas of academia and industry that are related to the seminar topic and typically do not intensively interact with each other. These communities are Product Line Engineering, Software Architecture, Service Engineering, and Cloud Computing.

The ambition of the seminar was to work on the topic of "Customization of Service Platforms", which is related to all of these areas, in a synergistic and cooperative way to identify new research challenges and solution approaches. As part of the seminar, we identified a number of key areas which provided the basis for highly interactive working groups.

**Seminar** 21.–26. April, 2013 – www.dagstuhl.de/13171
**1998 ACM Subject Classification** D.2.2 Design Tools and Techniques, D.2.11 Software Architectures, D.2.13 Reusable Software
**Keywords and phrases** Service-Oriented Architectures, Service Platforms / Cloud Computing, Product Line Engineering, Variability Management
**Digital Object Identifier** 10.4230/DagRep.3.4.114

## 1 Executive Summary

*Luciano Baresi*
*Andreas Rummler*
*Klaus Schmid*

### Background

Service-orientation has become a major trend in computer science over the last decade. More recently cloud computing is leading into the same direction: a virtualization of resources and service offerings. Especially cloud computing is getting very significant attention by companies. While the initial idea in service orientation was to have the relevant services standardized and distributed across the internet, we also see that an increasing amount of customization must be done to really meet customer needs. As in traditional system development, one size fits all is not enough.

This seminar focused on the notion of service platforms, a concept including, but not limited to, cloud computing. A service platform is a combination of technical infrastructure along with domain-specific or business-specific services built according to the service-oriented development paradigm. Especially the latter in practice often requires significant customization in order to be practically useful. Providing such customizations on a massive scale cost-effectively is an extremely demanding task. This is a lesson that has been learned hard by a number of companies in traditional software engineering. As a consequence the concept of product line engineering was conceived.

The focus of this seminar was to explore the range of different approaches towards customized service offerings in current —- and future —- service-based environments. In particular, it was a goal to address the potential for a combination of service-orientation with product line engineering ideas. In this regard, this seminar was the first of its kind.

## Diversity of Topics

The expected diversity of inputs that was desired for the seminar was well achieved. This is shown by the diversity of individual presentations summarized in chapter 3. Also the working groups that were established had participants from multiple communities. These working groups discussed the following topics:

**Quality Assurance and Validation in the Context of Customization:** Here, a broad range of different problems and techniques could be identified, related both to problems of varying of the object of the quality assurance as well as to the variation of the expections (qualities).

**Mobility Devices and Customization:** This working group focused particularly on the difficulties that arise from a mobile context with a lot of variation over time and limited resources.

**Architecting for Platform Customization:** Architectures are fundamental to any software system, so this group addressed what architectural techniques are important to create customizable platforms.

**Energy-Aware Customization:** Here, the focus was on the issue of energy-awareness and, in particular, energy-efficiency, which is particularly relevant to mobile platforms. By adequate customization, this can be improved for a platform.

**Customizing Service Platforms for Cloud Computing:** Modern cloud computing environments pose new challenges and provide new opportunities for customizing service platforms. It turned out that the cloud context provides a number of very special problems and technologies for addressing them.

**Customizing Service Platforms for Agile Networked Organizations:** The organizational context of service platform needs to be taken into account as well as a platform needs to fit to the relevant business context. Hence customization needs to be done on both levels in a synchronized manner.

**Binding time aspects of service platform customization:** This working group focused on when (i.e., in which lifecycle phase) the customization is done, as this has significant impact on the details of the technologies that can be used.

## Reflections on the Format

A main goal of the seminar was to have a significant portion of the time for discussion. In order to achieve this, we decided to not require presentations from everyone associated with a long introduction round. Rather, we decided to ask everyone for a poster to present her- or himself and describe the personal interest and relation to the topic. Overall this novel approach was well received by the participants. The poster walls were set up in the coffee break area outside the room. (Thanks to everyone at Dagstuhl for their support.) This allowed for a casual browsing of the posters in every coffee break during the seminar. Each poster also had a picture of the participant, this also helped to get to know each other.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Imperative versus Declarative Process Variability: Why Choose?

*Marco Aiello (University of Groningen, NL)*

**Joint work of** Aiello, Marco; Groefsema, Heerko; Bulanov, Pavel

Variability is a powerful abstraction in software engineering that allows managing product lines and business processes requiring great deals of change, customization and adaptation. In the field of Business Process Management (BPM) the increasing deployment of workflow engines having to handle an increasing number of instances has prompted for the strong need for variability techniques.

The idea is that parts of a business process remain either open to change, or not fully defined, in order to support several versions of the same process depending on the intended use or execution context. The goal is to support two major challenges for BPM: re-usability and flexibility. Existing approaches are broadly categorized as Imperative or Declarative. We propose Process Variability through Declarative and Imperative techniques (PVDI), a variability framework which utilizes temporal logic to represent the basic structure of a process, leaving other choices open for later customization and adaptation. We show how both approaches to variability excel for different aspects of the modeling and we highlight PVDI's ability to take the best of both worlds. Furthermore, by enriching the process modeling environment with graphical elements, the complications of temporal logic are hidden from the user. To show the practical viability of PVDI, we present tooling supporting the full PVDI lifecycle and test its feasibility in the form of a performance evaluation.

### 3.2 My View on Customizing Service Platforms

*Luciano Baresi (Polytechnic University of Milano, IT)*

A service platform is a set of related services supplied by the same provider under a common umbrella and together with some shared qualities of service. Platforms as a service are a special class of the more general concept.

My interests in the customization of service platforms come from different motivations. Since I have been working on services at application level for years, moving to platforms provides a nice complement. The work done on eliciting the requirements for (self-)adaptive service applications easily fits the customization problem since it helps one understand what the platform is supposed to provide, and how it should be tailored to the different needs and situations. Similarly, the work done on adapting service compositions may provide interesting insights towards the definition of suitable customization means for the different service platforms. In these days, I am also interested in mobile applications and in the frameworks (platforms) that provide the bases to implement them. Since the resources of these devices are still limited, the customization of these platforms may help preserve them, and thus it may help the user keep the device (e.g., a smartphone) alive longer.

I am still a bit concerned, or confused, about the use of many different terms, like customization, adaptation, evolution, and maintenance, to mean similar and possibly related concepts, but I am very interested in how the diverse services, and the infrastructure that operates them, can evolve in the different phases of the platform's life-cycle. Run-time changes, and the correctness of the new platform, are also particularly intriguing.

## 3.3 Dynamic Product Lines using the HATS framework

*Karina Barreto Villela (Fraunhofer IESE, DE)*

Typical Software Product Lines (SPL) approaches do not focus on dynamic aspects, and the reconfiguration of products occurs mainly statically at development time. Dynamic Software Product Lines (DSPL) enable a product to be reconfigured dynamically at runtime, which can be understood as the transformation of a product into another valid product without any kind of interruption in its execution. The reconfiguration, in this context, takes place without the need to halt the system, recompile and redeploy. From a technical perspective, dynamic reconfiguration is a challenging task due to reasons such as ensuring that dynamically updated systems will behave correctly or ensuring that no state data is lost. Moreover, from the Product Line (PL) perspective, not all technically possible changes in a running system are valid and make sense. In order to preserve the consistency of the PL products when reconfigured at runtime, there must be a way to restrict the adaptations that can be performed at runtime.

Fraunhofer IESE has added support for runtime product reconfiguration to ABS (an abstract but executable modeling language developed in the HATS project), by adding a dynamic representation of the possible product reconfigurations at runtime and a state update element responsible for data transfer, and by using the metaABS tool developed by the University of Leuven, which allows deltas to be applied at runtime.

## 3.4 Quality-Aware Product Configuration

*Karina Barreto Villela (Fraunhofer IESE, DE)*

The configuration of concrete products from a product line infrastructure is the process of resolving the variability captured in the product line according to a company's market strategy or specific customer requirements. Several aspects influence the configuration of a product, such as dependencies and constraints between features, the different stakeholders involved in the process, the desired degree of quality, and cost constraints. Fraunhofer IESE has developed a quality-aware configurator in which the user specifies the key product features and its quality concerns and cost constraints, and the configurator gives close to optimal configurations based on the user's input. The configurator is based on the assumption that the selection of a feature has an impact in the quality attributes of the final product, as well as the interaction among the selected features. This work included the integration of COSTABS (a static performance analyzer developed by the University of Madrid) to provide performance annotations to features and the first steps towards a reusable security feature model, which includes security feature implementations in ABS (an abstract but executable modeling language developed in the HATS project).

## 3.5 Customization of existing industrial plants to achieve modernization

*Deepak Dhungana (Siemens AG-Wien, AT)*

Industrial plants are complex and costly software-intensive systems that are operated over long periods of time. The modernization of plants with new technologies can significantly increase productivity while at the same time reducing energy consumption and environmental impact. Unfortunately, it is a daunting task to find out which new technologies are appropriate for an existing plant and to calculate the modernization costs and time. This process in practice today relies mainly on the experience and knowledge of key employees and is not well defined. Currently, there is no standardized method or tool for systematically eliciting customer requirements and plant data. In our ongoing work, we are developing methods and tools to support planning the modernization of complex industrial plants, which need to be adapted to meet new customer requirements and environmental constraints. In this project we first analyzed the current modernization process based on concrete scenarios and examples (e.g., improvement of a cooling system in a steel plant, in order to reduce the water consumption) to clearly define the requirements for tool development. The next step was to develop tools supporting the modeling of expert knowledge, the definition and formalization of modernization goals, as well as the definition of available resources. The optimization with regard to global constraints and objectives like productivity, quality, and economic impact is a complex task. We thus develop tools for capturing and modeling expert knowledge with the goal to assist in the selection of modernization packages based on customer requirements and in the creation of sales offers. The tools further support optimizing selected modernizations, for example, by comparing different modernization scenarios. The tools are flexible to allow their use within industrial plants in various domains.

## 3.6 Forward Recovery for Web Service Environments

*Peter Dolog (Aalborg University, DK)*

In web service environments there are web services which are often long running. In this situation, typical properties known from transactional management in databases, such as atomicity or isolation, are relaxed. This impacts on the transactions so that when some of the participants fail, they cannot easily undo outcomes of the web services participating in such transactions. We have studied this problem and designed an environment where we allow for forward recovery which means we allow for replacing failed web services with different ones which can deliver the work required to finish the transactions. The candidate web services are selected based on features which have been defined similarly as in product lines methodology, with mandatory and optional features. We compare and rank suitability of services according to matching between required feature model and those provided. The score is higher if there are more optional features satisfied with provided candidate service.

## 3.7 Customizing Service Platforms

*Holger Eichelberger (University of Hildesheim, GE)*

Customization of service-based systems is current practice in industry to meet the needs of customers in a qualitative and timely manner, e.g., to introduce novel functionality, to optimize the quality of service (QoS), or to realize integrations with existing systems. While many customizations can be implemented using service-oriented mechanisms such as (late) service bindings, several situations require the customization of existing services or the underlying service platforms, e.g., to develop domain-specific platforms. Here, systematic customization of services and service platforms can lower development effort and increase reusability.

Software Product Line Engineering (SPLE) is an industry best practice to achieve systematic customization in software systems. The key idea of SPLE is to focus on the differences (called variabilities) among similar systems. However, the existing methods and techniques for describing and realizing variabilities must be refined or extended to provide adequate support for service-based systems, including heterogeneity, open-world scenarios and runtime dynamicity which are common in service orientation. Thus, current challenges in customizing services and service platforms are: a) variability modeling for heterogeneous environments and, moreover, for entire software ecosystems, b) unified approaches to variability realization in service-based systems (in contrast to current individual and unrelated techniques), and for both, variability modeling and instantiation support for c) openness and extensibility and d) runtime variability.

Currently, we work on methods and techniques for addressing the challenges sketched above, in particular on

- Large-scale variability modeling, in particular in terms of the INDENICA variability modeling language (IVML), a textual language which provides concepts for variability modeling, runtime variabilities, openness, extensibility and QoS constraints.
- Generalizing and unifying the implementation of variabilities. Currently, we work on designing and realizing a common Variability Implementation Language (VIL).
- Increasing the flexibility of variability instantiations by separating the binding of variabilities and their functional code so that even the binding can vary according to properties of variabilities (meta-variability), e.g., to flexibly shift the binding time (currently applied in physical manufacturing of embedded systems).
- Observing the resource consumption of individual software parts at runtime, including services, components and variabilities. Our approach called SPASS-meter is designed for quality assurance for SPLE and, in particular, for supporting and simplifying the development of resource-adaptive software systems.

Future work is planned in particular on a) dynamic software product lines based on resource measurements and enhanced meta-variability, b) quality and resource aware variability modeling and c) large-scale variability modeling as well as supporting techniques.

## 3.8 SPASS-Meter – Monitoring Resource Consumption of Services and Service Platforms

*Holger Eichelberger (University of Hildesheim, GE)*

Monitoring the resource consumption of a system supports the operationalization of quality requirements, supports quality assurance, provides a basis for the estimation of energy consumption and supports the realization of (resource-aware) adaptive systems. Currently, resource consumption is typically measured on application level, on operating system level or, in contrast, on the level of individual classes. As also expressed in discussions during this seminar, there is a clear need to provide such measurements also for units within programs such as individual application services, for the underlying service platform or for technical services within the service platform.

In this talk, we present SPASS-meter, a novel monitoring approach, which enables the observation of resource consumptions for user-specified semantic units of software systems such as services, components or variabilities. In SPASS-meter, these semantic units are defined in the so-called monitoring scope specification, including the individual resources to be monitored as well as the monitoring depth, i.e., whether dependent functionality in related services, the service platform or in libraries shall be considered or not. SPASS-meter aggregates the resources consumption of these semantic units at runtime and allows comparing the consumptions with those on application and system level. Currently, SPASS-meter supports the monitoring of Java applications and, in particular, of Android Apps. As monitoring tools such as SPASS-meter execute additional code for probing and analysis, they cause a certain memory overhead. We conclude that the monitoring overhead created by SPASS-meter is reasonably small compared to the overhead of recent tools such as OpenCore or Kieker, in particular regarding the provided flexibility and functionality of SPASS-meter.

## 3.9 On-the-Fly Computing – Individualized IT Services in Dynamic Markets

*Gregor Engels (University of Paderborn, GE)*

Due to a steadily increasing market and budget pressure, the development and maintenance of IT systems have to become more efficient and more effective in the future. The traditional approach of software procurement by employing expensive and inflexible standard IT solutions or by purchasing individually developed software systems is obviously not a solution in the future. The new approach of cloud-based services allows an on-demand usage of software solutions and might be a first step in the direction of a more efficient and effective procurement of IT solutions. Combining this with the paradigm of service-oriented architectures, individualized IT services might be composed and used to fulfill certain business demands.

This service-oriented paradigm combined with the idea of deploying services in the cloud was one of the motivating starting points of the Collaborative Research Center

(CRC) 901 On-The-Fly Computing (OTF Computing), which is funded by the Deutsche Forschungsgemeinschaft (DFG) and conducted at the University of Paderborn since 2011.

The objective of CRC 901 – On-The-Fly Computing (OTF Computing) – is to develop techniques and processes for automatic on-the-fly configuration and provision of individual IT services out of base services that are available on world-wide markets. In addition to the configuration by special OTF service providers and the provision by what are called OTF Compute Centers, this involves developing methods for quality assurance and the protection of participating clients and providers, methods for the target-oriented further development of markets, and methods to support the interaction of the participants in dynamically changing markets.

In order to reach these objectives, computer scientists from different areas like software engineering, algorithms, artificial intelligence, distributed systems, networks, and security cooperate with scientists from the economics department, who are experts in organizing world-wide markets.

The CRC 901 is structurally divided into three scientific project areas: Project area A is devoted to the algorithmic and economic basic principles for the organization of large, dynamic markets. It concerns on the one hand the algorithmic procedures for the organization of large nets in general and for the interaction from participants in nets in particular; and on the other hand the economic concepts for incentive systems to the control of participants in markets.

Project area B investigates procedures for the modeling, composition and quality analysis of services and service configurations with the goal of an on-the-fly development of high-quality IT services.

Project area C develops reliable execution environments for the On-The-Fly Computing, and is concerned with questions of the robustness and security of markets, the organization of high-grade heterogeneous OTF Compute Centers and the execution of configured services by such Centers. In addition, there is an integrated application project which is concerned with optimization systems for supply and logistics networks and acts on a long-term basis as an application domain for the work of the SFB.

More detailed information about the CRC 901 can be found at http://sfb901.uni-paderborn.de/sfb-901.

## 3.10 Multi-level Service Management

*Sam Guinea (Politecnico di Milano, IT)*

Due to the growing pervasiveness of the service paradigm, modern systems are now often built as Software as a Service, and tend to exploit underlying platforms (Platform as a Service) and virtualized resources (Infrastructure as a Service). Managing such systems requires that we are aware of the behaviors of all the different layers, and of the strong dependencies that exist between them. This way we will be able to perform run-time customization and ensure that the functional and non-functional aspects of the overall system are always preserved, even in the wake of profound changes in the stakeholders' requirements and in the context of execution.

To accomplish this we are studying how to apply the traditional MAPE-K (Monitoring

– Analysis – Planning – Execution) control loop to such multi-level systems. We advocate that this will require novel data collection, analysis, planning, and execution mechanisms. Indeed we will need to collect runtime data from multiple levels at the same time, and be able to correlate them to build more detailed information of what is actually occurring inside the system. To this extent we have developed the Multi-layer Collection and Constraint Language. It allows us to define how to collect, aggregate, and analyze runtime data in a multi-layered system. We also present ECoWare, a framework for event correlation and aggregation that supports the Multi-layer Collection and Constraint Language, and provides a dashboard for on line and off-line drill-down analyses of collected data. Our initial empirical assessment shows that the impact of the approach on runtime performance is negligible.

In the future we will further pursue this research by evaluating our results in concrete real-world examples, through the collaboration with key cloud-based industrial partners. We will also study how the understanding that we gather of the system at runtime can be used to plan coordinated recovery actions at multiple levels. Indeed, we expect that the most cost-effective customization solutions would require coordinated intervention at multiple levels.

## 3.11 Customizable Reliability and Security for Data-Centric Applications in the Cloud

*Waldemar Hummer (TU Vienna, AT)*

Service-oriented computing (SOC) has become a prevalent paradigm for creating loosely coupled distributed applications and workflows. In parallel to SOC, Event-Based Systems (EBS) in various fashions (e.g., data stream processing) are gaining considerable momentum as a means for encoding complex business logic on the basis of correlated, temporally decoupled event messages. More recently, advanced virtualization and resource allocation techniques advocated by Cloud computing have further shaped the implementation possibilities of SOC and EBS. Clouds have proven to be an ideal environment for flexible and elastic applications which provide scalability, resource optimization, and built-in support for multi-tenancy. Ongoing trends in the area of Data-as-a-Service (DaaS) have spurred further research efforts towards robust data processing services, leveraging the benefits of the Cloud.

Distributed computing systems in general, and applications in the Cloud in particular, are often burdened with stringent requirements concerning reliability and security, dictated by business objectives (e.g., cost-benefit tradeoffs), contractual agreements (e.g., service level agreements, SLAs), or laws. Customized support for reliability and security in service platforms is a key issue. One approach to reliability is software testing, which attempts to identify and avoid software-induced faults in the first place. A second important aspect of reliability is adaptability and fault-tolerance, which involves different runtime challenges such as fault detection, isolation, or recovery. Additionally, due to the multi-tenancy inherently encountered in Cloud environments, security and access control play a crucial role for application provisioning. Consideration of these aspects in the software development and validation process requires precise knowledge about the type and nature of potential threats to reliability.

Within our work we tackle the aforementioned challenges and present novel techniques for reliable and secure provisioning of data-centric service platforms and applications in the Cloud. We strive for a robust, scalable, and secure execution environment for applications to integrate services and data from a plurality of sources, generating added value for service consumers. During the development phase, applications are systematically tested for incompatibilities and integration issues. At runtime, platforms should leverage Cloud virtualization to ensure reliability and efficiency (elastic scaling, minimal resource allocation, optimized load distribution). Moreover, customized security policies need to be enforced to assure responsibilities and avoid unauthorized access.

## 3.12 Adaptation in complex service ecosystems

*Christian Inzinger (TU Vienna, AT)*

Our current research deals with customization through adaptation of complex service ecosystems operating in dynamic environments such as cloud computing systems. Based on our work on fault detection and identification we model monitoring and adaptation behavior of complex applications in a unified manner to allow for optimized deployment of necessary control infrastructure.

## 3.13 A Library for Green Knowledge

*Patricia Lago (VU University Amsterdam, NL)*

**Joint work of** Lago, Patricia; Gu, Qing

In spite of the investments in green ICT, industry and research both lack reusable green practices including operational actions to re-green ICT, metrics, and examples of achieved results. Such green action can include optimizations in customized cloud provisioning, but also reusable patterns for engineering software exploiting service oriented principles.

Another problem is the lack of alignment between economic impact and environmental effect in green practices. If green practices do not lead to an explicit (and significant) reduction of costs (hence increase in revenues) they are nice but not part of the business strategy of the company.

To address these two problems, in this project an online-library for green practices has been built. This library provides a collection of 258 reusable green ICT practices with explicitly documented environmental effects and economic impacts, based on which companies are able to select and justify green ICT practices that fit best their needs.

While green practices so far mainly focus on non-software related actions, research is maturing toward energy efficient and environmental sustainable software service engineering. Future optimizations (green actions) will hopefully focus on how to achieve green services and how to combine them in greener service-based applications.

### 3.14 Cloud Computing as a Service Platform for Mobile Systems

*Grace Lewis (SEI, USA)*

Cloud computing infrastructures are used by organizations to provide access to large public data sets such as maps and images from mobile devices, and to host mobile applications outside of the enterprise to support front-line employees such as sales personnel. An additional use case that is at the intersection of mobile and cloud computing is to use the cloud to perform computation-intensive activities on behalf of mobile devices such as is currently done by Apple Siri, Google Glass, and the coming soon Apple iWatch. The latter use case is the one that is of interest from the perspective of customizing service platforms. This presentation discusses cloud computing as a service platform for mobile systems in the context of cyber-foraging – the leverage of external, nearby resource-rich surrogates to augment the capabilities of resource-limited mobile devices. It presents two types of cyber-foraging – code/computation offload and data staging – as well as the challenges of customizing surrogates as service platforms.

### 3.15 Cloudlet-Based Cyber-Foraging

*Grace Lewis (SEI, USA)*

Cloudlet-Based Cyber-Foraging is a strategy for extending the computation power of mobile devices by offloading resource-intensive computation to cloudlets – discoverable, generic servers located in single-hop proximity of mobile devices. We present the basic of cloudlet-based cyber-foraging in addition to future work in this area to address system-wide quality attributes beyond energy, performance and fidelity of results.

### 3.16 Customizing Platforms by Higher-Order Process Modeling: Product-Lining, Variability Modeling and Beyond

*Tiziana Margaria (University of Potsdam, GE)*

**Joint work of** Margaria, Tiziana; Steffen, Bernhard; Neubauer, Johannes

(Business) Process modeling languages like BPMN2 are static in the sense that they determine at modeling time which activities may be invoked at runtime and where. We overcome this limitation by presenting a graphical and dynamic framework for binding and execution of (business) process models. This framework is tailored to integrate

1. ad hoc processes modeled graphically,
2. third party services discovered in the (Inter)net, and
3. (dynamically) synthesized process chains that solve situation-specific tasks, with the synthesis taking place not only at design time, but also at runtime.

Key to our approach is the introduction of type-safe stacked second-order execution contexts, that allow for higher-order process modeling. Tamed by our underlying strict service-oriented notion of abstraction, this approach is tailored also to be used by application experts with little technical knowledge: users can select, modify, construct and then pass (component) processes during process execution as if they were data. The approach has been applied to a concrete, realistic (business) process modeling scenario: the development of Springer's browser-based Online Conference Service (OCS).

The most advanced feature of our new framework allows one to combine online synthesis with the integration of the synthesized process into the running application. This ability leads to a particularly flexible way of implementing self-adaption, and to a particularly concise and powerful way of achieving (re-)configuration via variability not only at design time, but also at runtime.

## 3.17 Platform Architectures

*Nenad Medvidovic (USC – Los Angeles, USA)*

The talk explores different views of service platform from the perspective of architectural style and architectural building blocks (specifically, connectors). An argument is made that a platform in this context is a middleware platform or a framework. Customization, then, boils down to customizing the middleware or framework. These are software systems in their own right and suffer from many architectural problems common to software systems. Grid service platforms are presented as an example case study. A number of open issues are identified as research challenges.

## 3.18 Variability Modeling & Management

*Nanjangud C. Narendra (IBM India – Bangalore, IN)*

Our work is motivated by the need to improve productivity of software development solutions, in particular, SOA-based solutions, in the IT services industry. Traditional approaches have involved the development of solutions from scratch in every customer engagement. To that end, we have developed the Variation Oriented Engineering (VOE) approach towards developing reusable SOA-based solutions, by modeling variations in those solutions as first-class entities. Currently our work has spanned the following topics:

- Variation Oriented Service Design for deriving variants from Business Process specifications
- Automated change impact propagation
- Variability Modeling for determining legal variants
- Variant and Version Management in Business Process Repositories

We foresee the following challenges in Variability Management:

- Formalizing Variability via algebraic approaches
- Integration with Business Process Management
- Lifecycle-based approach towards Variability Management
- Variability Management at runtime

Our future work will comprise (but not be limited to) the following:
- Variability Algebra
- Integration with adaptive workflow
- Variability at runtime
- Integrating variability into service ecosystems

## 3.19 Customized Mashups with Natural Language Composition

*Cesare Pautasso (University of Lugano, CH)*

End-User Development (EUD) is an emerging research area aiming at empowering non-technical users to somehow create or design software artifacts. Web mashups provide a high potential for EUD activities on the Web. Users on the Web can tap into a vast resource of off-the- shelf components in order to rapidly compose new, custom-made, lightweight software applications called mashups. In this presentation we have demonstrated JOpera (http://www.jopera.org) a visual service composition tool for Eclipse and NaturalMash a natural mashup composition tool that combines WYSIWYG, programming by demonstration and constrained natural language within a live programming environment that lets end users interactively specify the behavior of custom-made mashups that are built on-the-fly.

More information:
- S. Aghaee, C. Pautasso, Live Mashup Tools: Challenges and Opportunities, accepted at the First ICSE International Workshop on Live Programming (LIVE 2013), San Francisco, USA, May 2013.
- S. Aghaee, C. Pautasso, EnglishMash: usability design for a natural mashup composition environment, 4th International Workshop on Lightweight Integration on the Web (ComposableWeb2012) at ICWE 2012, Berlin, Germany, July 2012

## 3.20 Challenges of offering customizable domain-specific business processes as a service

*Manuel Resinas Arias de Reyna (University of Sevilla, ES)*

**Joint work of** Resinas Arias de Reyna, Manuel; Ruiz Cortés, Antonio

The growing demand of business–driven IT systems as well as the rise of Software as a Service (SaaS) has led to the creation of a category of SaaS known as Business Process as a Service (BPaaS). In them, service users can access a set of domain-specific processes, customize them according to their needs and enact them in the cloud. In this scenario, several challenges arise. On the one hand, current mechanisms to manage the variability in business processes should be extended to allow the customization not only of the control flow, but also of other perspectives of the process such as the organizational or the performance perspective. On the other hand, compliance with regulations, best practices and internal policies is a key aspect in organizations nowadays and may vary significantly from one organization to another.

Therefore, BPaaS must provide their users with mechanisms to ensure their processes are customized and enacted according to the regulations that are relevant for the users. Our current work focus on facing these challenges leveraging the work we have done on business process compliance management systems and on models and techniques for the management of process performance indicators and human resources

## 3.21 Customization of Large, Complex Systems

*Klaus Schmid (University of Hildesheim, GE)*

The major thrust of our work is on the customization of large, complex systems and in particular software ecosystems. We are in particular using product line engineering technologies to perform the necessary kinds of customizations. A particular challenge in the service platform is the need to support a range of very different artifacts and the need also to go to later binding times like initialization time or runtime. This requires on the hand a complex coordination among individual customizations to support the integrated customization. On the other hand it requires different techniques to address the later binding times.

A further challenge is the overall size and complexity of the platforms, which may often give rise to many thousand variation points.

## 3.22 Service Networks for Development Communities

*Damian Andrew Tamburri (VU University Amsterdam, NL)*

Communities of developers have rapidly become global, encompassing multiple timezones and cultures alike. In previous work we investigated the possible shapes of communities for software development. In addition, we explored mechanisms to uncover communities emerging during development. However, we barely scratched the surface. We found that development communities yield properties of dynamic change and organic evolution. Much work is still needed to support such communities with mechanisms able to proactively react to community dynamism. We argue that service-networks can be used to deliver this support. Service-networks are sets of people and information brought together by the internet.

The missing keystone is to support social communities with an innovative and pro-active mechanism operating through services. The research hypothesis that drives the work in this paper is quite simple and equally intriguing: social communities of developers can be supported by a global network of software and socio-technical services, spanning different organisations, sites, timezones and cultures. The result is a service-network that blends the internet of services with large-scale, adaptable choreographies to deliver a powerful and scalable solution that adapts to the changes of a community. On one hand, software services are pieces of software operating under a service-dominant logic. These pieces of software collaborate together across the web using standard protocols, to deliver complex, adaptable functionality (e.g. cloud-based functionalities such as GoogleDocs). Much literature in service sciences provide ways to identify, monitor and adapt software services On the other hand, socio-technical services are hybrid human and software services, i.e. services that explicitly mediate the collaborative work of people within a social community, e.g. by fostering relevant community aspects or by increasing situation awareness of community members.

http://www.dagstuhl.de/mat/Files/13/13171/13171.TamburriDamianAndrew.Paper.pdf

## 3.23 Customizing Science Gateway Platforms via SaaS Approach

*Wenjun Wu (Beihang University – Beijing, CN)*

A Science Gateway is a computational web portal that includes a community-developed set of tools, applications, and data customized to enable scientists to run scientific simulations, data analysis, and visualization through their web browsers. Because scientists always have different requirements for their data processing pipeline, science gateway developers have to cope with the customization of GUI, workflow, applications and runtime environment. So the research problem is how to effectively support multi-tenant customization in science gateway platforms.

The talk introduces a SaaS framework to enable customization of life science gateway through four levels: GUI, workflow, bio-application and workspace.

It allows users to rapidly generate Web GUI and deploy their pipelines in heterogeneous environments

## 3.24 Service-based Platform Integration and Customization

*Uwe Zdun (University of Vienna, AT)*

In service-based integration, platform customization, and similar areas, our research group addresses the following challenges: understand and support architecture and design decision making; link architectures, designs, and implementations; automate recurring tasks; base these solutions on time-proven architectural knowledge; provide empirical evidence. Our work and interests in this area concerns

- reusable decision models and corresponding tools
- design patterns; model-driven techniques (MDD) to bridge between architectural decisions and designs
- view-based architecture for service platform
- MDD generators
- full traceability
- empirical studies

### 3.25 Customizing Service Platforms — new or have we seen this before?

*Frank van der Linden (Philips Medical Systems – Best, NL)*

I have the feeling that, although the problems are new, I have seen this before. Over the year people have struggled with customization or variability at higher levels of abstraction. The initial programming languages tamed the variability into a few constructs: if, case, while, ... and goto. When the programs became complex, functions and subroutines were introduced. This added parameters and recursion to the palette of variability. Separate compilation added IFDEF. Again systems became complex, and again variability needed to be tamed. Functions were grouped into object classes, and related data into objects. This added inheritance to the mechanisms variability. A next step added configurations of object classes into components.

Each time, the new concept reduced the choices of how variability can be used. Special configurations were supported others were not. Sometimes a new mechanism was introduced, but it also kept the programs comprehensible, because the mechanism provided abstraction and hiding of internal details. Presently configurations of components are combined into services. This provides, again, abstraction and hiding of internal details. The situation is somewhat different because now it is apparent that services can and will be provided by different providers. This was also the case for previous mechanisms, as there are third party libraries, object frameworks, and COTS. However, the services structure is getting complex, and we cannot track, control or trust all the code will be executed. As in previous times we have to apply the mechanisms we have used before – variability management, negotiation, service configuration modeling, reduction of configurations that will be allowed to those for which the trust can be assessed. This still needs partially to be done, and that is the goal of this seminar.

# 4 Working Groups

### 4.1 Quality Assurance and Validation in Customizable Service Platforms

*Deepak Dhungana (Siemens, AT)*

*Participants:* Deepak Dhungana, Waldemar Hummer, Georg Leyh, Frank van der Linden, Antonio Ruiz Cortés

#### 4.1.1 Introduction

With the increasing importance of service oriented applications in many businesses and new application fields such as cloud computing, many new challenges are arising in this area. The initial effort required for customization of a service platform or associated services is already very high, but at the same time the nature of these applications requires them to consider customizations at runtime, too. Apart from the technical difficulties related to customization of the functionality, we see serious needs to discuss the impact of developing and deploying customizable services on the quality of the overall system.

#### 4.1.1.1 Quality Assurance

Software quality assurance is often defined as a means of monitoring the software engineering processes and methods used to ensure quality. Quality assurance therefore needs to consider both how such services can be developed or deployed and the runtime monitoring to ensure required quality.

Service based applications (SBA) are described by functional and non-functional properties. Non-functional properties include some QoS dimensions such as accuracy, coverage, network-related QoS, performance, reliability, robustness, and scalability. Our discussion in this section is focused on the relationship between customizability of applications and the impact on the quality attributes.

#### 4.1.1.2 Open Questions

We summarize the key issues related to customization of SBA and quality of the overall system.

- What is the effect of customizability (having a flexible and customizable platform) on the quality of the system? What kind of activities is additionally (as opposed to fixed/monolithic environments) needed during development and runtime to ensure required quality?
- What support can be provided to different stakeholders (which roles do they have?) in customizing a service based application to achieve the required quality?
- How can variability of the quality attributes of a SBA be defined, so that the customization processes and tools can deal with them?

### 4.1.2 Case Studies

In oder to demonstrate the industrial need for "quality-aware" service based applications, we present two case studies.

#### 4.1.2.1 Case Study 1 – Image Processing Service Platform

The first case study is about an image processing service platform at Philips medical systems. The platform provides image processing services to clients that are, in general, hospitals. Dependent of the hospital infrastructure, but also to the terminal capabilities, more or less processing can be done at the client side. A solution for this is to split the service platform in several abstraction layers and provide services in between the abstraction layers. Dependent on the situation, the client can take one or more top level abstraction layers and the server provides the remainder. Note, that the client platforms are not in control of the service provider and can be very diverse. The hospital pays for the services provided. It is the business model that determines how this is done, and will not be part of this description.

This all needs to be done in an environment where several quality concerns are important. These are related to performance, latency, throughput, but also to security, privacy and legal rules. All these quality requirements may vary dependent on the customer. In addition, the user expects personalization.

In summary, the customization deals with personalization, quality requirements, and the level of abstraction that will be provided.

#### 4.1.2.2 Case Study 2 – Logistics Management Platform

The second case study is about a platform for logistics, e.g., Warehouse Management solutions or solutions for spare parts logistics. Usually, the platform vendor provides (delivers) many variants of the platform based on domain specific service platforms. The goal is to reuse the service platforms for different customers, however, different users usually have a different prioritization of qualities (e.g., a small warehouse may prefer a very cost-efficient solution, which may be based on open source software, while a big warehouse needs a solution that is available 24/7). Therefore, we need to customize the prioritization of qualities.

In spare parts logistics, the platform usually makes heavy use of existing legacy systems installed at the end-users site. Sometimes, different legacy systems are deployed e.g. in different countries. Here we need a uniform customization, even if the services use different legacy platforms / data.

### 4.1.3 Identified Challenges

Based on the case studies, we have identified the following challenges related to quality assurance in customizable service platforms.

#### 4.1.3.1 Quality Monitoring

Many service platforms use resources from existing legacy systems. Those systems usually have no formal SLAs. To model qualities of systems that include legacy systems, at least the quality monitoring for legacy systems is necessary. The monitoring information can be used instead of a formal SLA, e.g. to calculate the overall availability of the SBA.

#### 4.1.3.2 Quality(-Driven) Adaptation

As seen in case study 2, for reusing domain specific service platforms it is necessary to adapt the prioritization of different qualities. Since the qualities of the SBA need to be adjusted, a check whether these qualities can be achieved with the existing base services needs to be made. If the base services cannot be used to achieve the necessary quality, other services must be checked for conformance.

#### 4.1.3.3 Quality Assurance in the Development Lifecycle

Quality assurance has to be performed at several stages in the development lifecycle. Already at development time software engineering has to provide the right mechanisms to do quality assurance. In addition, certain development patterns may be used to ensure a level of quality assurance upfront. A candidate service needs to be tested. For instance, it has to become clear whether it performs according to its SLA (see below). This action is similar to acceptance tests for COTS integration in the software. However, run-time testing needs to be added, as the service provision may change unexpectedly. In this case a fault tolerance mechanism needs to be added. In the case that a service fails to perform with the right quality, this should be repaired. There are several options for this: renegotiation, adding a similar service to the configuration, or escalating the fault to the higher level – i.e. announce on-conformance to the own SLA to the client.

#### 4.1.3.4 SLA Modeling

An SLA (Service Level Agreement) is the interface of a service towards its clients. Based on the SLA a client decides whether to use the service. An SLA describes the function that is provided; in addition it describes the level of relevant qualities the function is delivered. Today a SLA is usually static, which means that for each quality a certain quality range is given. Note, that this implies that there is a metric to measure the quality level. Sometimes a few configurations are offered that the client might select.

At the client side the situation is less simple. Quality levels might be related by conditions, such as: if quality A is less than level U then quality B must be more than level V. Also qualities might have priorities: If one of the quality levels needs to be dropped than C is the last one to choose from. The client also might have quality requirements that cannot be measured, e.g. data quality (such as: does the picture show the right features).

This all asks for a model to incorporate all the qualities that are relevant for the client and this needs to be used for SLA management. A model will describe a region in a multi-dimensional space, where qualities are dimensions. The region is the level of acceptable quality. Note that in this model cost is just one of the quality dimensions.

#### 4.1.3.5 SLA Management

SLA management deals with negotiating and monitoring the services provided. Negotiation deals with finding a service configuration that provides the service fitting in the acceptable region defined by the model. Note that we may need a configuration, as it might be the case that a single service cannot provide the right SLA; e.g. if two services both have an availability of 98%, using them both for 50% of the time we can get an availability level of 99%. Note that finding the right configuration can be posed as an optimization problem. In the case that a standard modeling language exists for SLA, it might be expected that service providers might offer more complex service offerings, which makes negotiation more complex as well.

The importance of temporal awareness is rising in SOA solutions. Temporal awareness refers to managing service demands and offers which are subject to validity periods, i.e. their evaluation depends not only on QoS values, but also on time. For example, the QoS of some web services can be considered critical in working hours (9:00 to 17:00 from Monday to Friday) and irrelevant at any other moment. Until now, the expressiveness of such temporal–aware specifications has been quite limited. This issue also makes negotiation and monitoring more complex.

After negotiation the SLA management is not finished. At run-time SLA monitors the services for several reasons:

- Does the service work according to its SLA? If not, then fault management needs to be incorporated to manage the fault, which might lead to re-negotiation, changing the service configuration, or escalation to the client.
- Establish levels of qualities that are not mentioned in the SLA, or that might be difficult to measure

### 4.1.4 Possible Solutions

Some possible solutions to deal with the identified challenges could be summarized as follows. However, these are rather open issues, that need to be elaborated further and are topics for future research.

#### 4.1.4.1 SLA Management Facilities

There are a number of activities that may be performed one or more times during the SLA management lifecycle and the operation of a SLA-aware service platform such as:

- Prior to advertising an offer (quality guaranteed by the service provider) and issuing a demand (customer quality requirements), they both should be checked for consistency, i.e. to check that they do not have any internal contradictions.
- Checking whether an offer fulfills a given demand, i.e. checking them for conformance a.k.a compliance.
- Finding the optimal offer out of a set of offers conformant to a given demand, mandatory if we want to automatically create the optimum SLA.
- Checking whether an SLA has been violated.
- Finding out all the SLAs which are 'outside the law' defined by a set of governance policies.
- Finding out the set of SLAs that may be violated during a given time window with a cost below a given amount.
- Giving explanations about why an offer or demand is not consistent, why there is no possibility to reach an agreement, why the SLA has been violated, etc.

The degree of difficulty of implementing these facilities depends on the degree of expressiveness of the SLA model used. Furthermore, implementing some of these facilities may lead to NP-hard problems, especially if features such as conditional terms, temporal-awareness, non-linear selection criteria are allowed.

These activities could be organized by using a catalogue of common operations a.k.a facilities (this approach has been widely accepted in the Automated Analysis of Software Product Lines). In this catalogue it would be possible to distinguish between basic and composite operations (only if it is possible to define it as a combination of basic operations) as well as to provide a reference implementation.

#### 4.1.4.2 Support during development

The issue of customization and service quality assurance influences all phases of the service engineering lifecycle, including design, implementation, validation, deployment, and runtime. We categorize this lifecycle into development phases (design, implementation, validation) and runtime phases (deployment, execution time). To assure quality in service environments, it needs to be clearly understood which quality parameters are influenced by different parts of the lifecycle.

During the development phase, all aspects related to quality assurance need to be collected and encoded in a multi-dimensional quality model. The quality model should capture the requirements, goals, and risks associated with different quality assurance scenarios. The quality model then serves as the basis to derive measurable quality metrics as well as potential actions to take in case of quality issues (e.g., SLA violations). We distinguish between static consistency checks and dynamic conformance checks. Static consistency checks are required to determine whether the quality model can generally satisfy important properties such as soundness, completeness or satisfiability. Dynamic conformance checks are employed to determine, for concrete instantiations of service platforms, whether the current state corresponds to the target quality requirements and goals.

The development phase is particularly important for quality assurance, for two main reasons. Firstly, certain quality characteristics like correctness, availability, consistency, or fault tolerance need to be modeled and systematically tested for. Secondly, all capabilities required to assure quality at runtime (e.g., monitoring, optimization, adaptation) need to be accounted for during the development phase.

#### 4.1.4.3 Runtime Support

During runtime, one of the key concerns is to monitor the conformance of the service platform to customized quality metrics, in order to timely react to occurring SLA violations or potential upcoming quality issues. In recent years, event-based monitoring based on the complex event processing (CEP) paradigm has emerged as the key technology to support loosely coupled monitoring infrastructures. CEP leverages the concept of complex events, which aggregate and correlate streams of raw events to provide higher-level knowledge about the current quality and health status of a system, i.e., service platform. Efficiency and non-intrusiveness are among the core research questions related to monitoring of service quality.

Certain quality problems allow timely correction by adapting the system within the current phase of the provisioning lifecycle. For instance, a service platform which is configured to react to load bursts should be able to acquire new computing resources as soon as a quality degradation is measured at runtime due to high request load. However, if elasticity is not correctly implemented and runtime monitoring detects that the acquired resources are not released once the load decreases, the problem needs to be escalated and fixed in previous phases, re-iterating through the development phases of design/implementation/validation. As of today, the research community still lacks a deep understanding of how to support this type of escalation by systematically modeling the connections between quality metrics and different development lifecycle phases.

#### 4.1.4.4 Domain-Specific Solutions

In recent years, research and industry have experienced the emergence of new paradigms related to service platforms and service-oriented computing.

Arguably one of the most important trends is Cloud Computing, which introduces advanced virtualization and resource allocation techniques, providing for a new class of applications with a high degree of dynamism, scalability, and elasticity. These elastic applications often require non-trivial re-configurations, because once the elasticity state is changed (i.e., a resource gets added or removed), the connections and dependencies on potentially many other resources need to be updated and re-configured. To ensure that the transitions between elasticity states function properly, comprehensive testing and verification efforts need to be undertaken. Moreover, quality agreements in Cloud Computing are inherently related to multiple tenants. To avoid redundant detection and enforcement efforts, quality assurance mechanisms should be efficiently tailored to multiple tenants, for instance by grouping together tenants with similar or overlapping quality requirements.

A second important trend is the increasing integration of humans in the service delivery process. Novel concepts like crowd-sourcing or human-provided services (HPS) are finding adoption in service-based applications. Since humans operate distinctly different from machines, advanced quality characteristics and metrics such as trust, skills, or reputation need to be taken into account.

### 4.1.5 Open Issues

Some issues need further discussions and more industrial cases to support their practical relevance.

#### 4.1.5.1 Variability in SLA Definitions

Currently, it is not possible to describe variability in SLAs. In addition, it is not possible to check SLAs that contain variability information for conformance with other SLAs. E.g., a variable SLA may state that it has a const / availability variation point. 99% availability with 0.01 EUR per call for low budget, 99.99% availability with 0.05 EUR per call for high availability customers. Three services are available: Service A with 99.999% availability and 0,03 EUR / call, Service B with 99.9% availability and 0,005 EUR per call, Service C with 99,99% availability and 0,01 EUR per call.

Service A would partially comply (only high availability customers), Service B would partially comply (only low budget customers), Service C would fully comply to the variable SLA. A formal language and calculus for this kind of problems is currently missing.

#### 4.1.5.2 Systematic Decision Support

Quality assurance, in customizable service platforms, requires decision making in complex multi-dimensional spaces. This implies that automatic decision support is requested. It is needed in static and run-time SLA management during negotiation and configuration selection to decide which offering fits best to the request. Decision support needs optimization algorithms to execute its ask.

The decision making has to find matches of regions in multi-dimensional spaces, where certain dimensions are described as ranges, others are described as a probability distribution, and others are even not really measurable, but are based on human "expert" based decisions. In addition there are relationships between different dimensions, and there are dimensions that have higher priority than others. Optimization algorithms exist, but they often are restricted to certain kinds of input only.

The main issue here is to find algorithms that deal with the complexity above. In addition, it needs to be made clear in which form the regions in the spaces can be described. This latter point is related to SLA modeling, but modeling needs to address the issue of decidability as well.

The output of decision support will be the best offer that fits the requirement, but it should also indicate margins between the solution and the best fit. This margin should be expressed in such a way that it can be understood by the human client.

#### 4.1.5.3 Integrated Tooling for SLA Languages and Models

Probably it makes no sense to design a universal (domain-independent) language to describe SLAs. In fact, the WS-Agreement specification identifies up to nine regions where for each and every region a DSL (Domain Specific Language) must be provided by the SLA editor. Thus, it is possible to have an unbounded number of different WS-Agreement compliant DSLs. Probably, this circumstance may explain at a given extent why the WS-Agreement has not been (widely) used.

However, it does not seem reasonable to implement the SLA management facilities (see above) from scratch to the management of the SLAs of each SBA, especially for very expressive SLAs. Therefore, there is an important room for improvement in this issue.

#### 4.1.5.4 Bootstrapping

In this document we have discussed various aspects of quality assurance, including modeling, monitoring, adaptation, decision support, and more. An additional complexity dimension is the question of how quality requirements can be introduced into an (existing) platform in the first place. Assume that the current state of quality provided by a service platform is expressed as X, and that the provider plans to achieve the customized quality level Y. First, the delta between X and Y needs to be identified. In order to enact the required changes, a semi-automated procedure can be employed to identify the required steps to be taken. For high-availability applications the key concern is that these quality adaptation steps should be enforced with the shortest possible downtime. Moreover, since these steps may involve non-trivial re-configurations and adjustments within the platform, a critical aspect of the adaptation procedure is to maintain correctness and not to introduce any new issues (e.g., bugs, misconfigurations).

#### 4.1.5.5 Maturity Model for Customizability

In general terms, a maturity model can be viewed as a set of structured levels that describe how well the behaviors, practices and processes of an organization can reliably and sustainably produce required outcomes. In this sense, a maturity model can be used as a benchmark for comparison and as an aid to understanding – for example, for comparative assessment of different organizations where there is something in common that can be used as a basis for comparison. In the case of the CMM, for example, the basis for comparison would be the organizations' software development processes. (from Wikipedia)

Assuming that service platforms have a considerable number of customization points that may crosscut different perspectives implies that the customization degree may vary significantly among service platforms. Moreover, the existing dependencies among customization degree and other quality factors such as maintainability increase the complexity of this issue.

In these circumstances having a maturity model to assess the service platform's customization capabilities could be really useful for both customers and providers.

## 4.2 Mobility and Service Platform Customization

*Grace Lewis (SEI, USA)*

*Participants:* Luciano Baresi, Schahram Dustdar, Sam Guinea, Grace Lewis, Tiziana Margaria, Andreas Rummler, Karina Villela, Wenjun Wu, Uwe Zdun

Mobile computing is transforming the way in which people interact with the world and with each other far beyond the simple use of a smartphone as a communication device. In recent years, there has also been a rapid explosion of mobile devices and sensors that are not only pervasive but often interconnected. Mobility and ubiquity therefore create a potential for mobile devices to (1) become service platforms for local applications as well as service platforms for other nearby mobile devices and (2) extend their computational capabilities by taking advantage of service platforms in the cloud. This working group explores both options.

### 4.2.1 Mobile Devices and Customization

We define mobile device as any device that is battery-powered, has wireless capabilities, and a small form factor. Examples of mobile devices therefore include smartphones, tablets, wearable devices (e.g., Google Glass, iWatch), devices built on top of small single-board computers (e.g., RaspberryPi), sensors, drones, sports devices (e.g., FitBit), medical devices, and navigation devices.

Examples of design-time and runtime customization for service platforms deployed on mobile devices to serve local applications as well as other mobile devices include:

- Sensors to enable/disable
- Exploitation of context-awareness to adjust sensor sampling rates to drive energy efficiency or data quality/precision
- Data consumed and provided
- Definition of swarming behavior for groups of mobile devices
- Location of deployed sensors
- User interaction (e.g., touch, gesture) or action patterns based on specific applications or content
- Add/remove computation or change algorithms
- Communication mechanisms to enable/disable (e.g., WiFi, Bluetooth)

As can be seen in the previous examples, it is difficult to differentiate between services that are provided by the operating system, services that are provided as part of a more traditional service platform, and the applications themselves. This is probably due to the small form factor but also because mobile applications are tied to a specific platform/OS and typically make use of sensors that are part of the platform.

### 4.2.2 Mobility and Service Platforms

There are two areas related to mobility and service platforms that are interesting from a customization perspective:

1. Mobile device as a service platform: These are mobile devices that act as mobile limited-resource service platforms that can exploit on-board sensors and humans for data collection or collections of devices combining to form a single platform.
2. Cloud computing as a service platform for mobile systems: Mobile devices can use service platforms in the cloud in multiple ways:
   - Mobile device as a data collection platform that is uploaded to a service in a surrogate or the cloud
   - Surrogates as a service platform for multiple mobile platforms
   - Mobile devices connecting directly to services in the cloud

### 4.2.3 Sample Scenario: First Responders

There are multiple scenarios that would benefit from customizable mobile service platforms:

**First Responders:** Personnel operating in emergency situations can use a variety of mobile devices to support dynamic mission activities such as situational awareness, exploration of unsafe areas and medical assistance.

**Asset tracking:** Mobile devices can be attached to any asset to determine location, usage patterns, or environment characteristics. An example of an asset is a container that can be tracked from origin to destination, varying the data sampling and rate according to location.

**Smart homes/cities:** Data collected and potentially pre-processed by mobile devices that are spread throughout homes or cities can help in task automation, emergency detection and response, surveillance,

**Remote locations:** Mobile devices can be located or dispatched to locations where it is difficult, impossible or dangerous for a human to go to.

The first responder scenario is of particular interest from a customization perspective because of the dynamic nature of the environment as a disaster or emergency situation evolves from panic to medical attention to supplies to connecting people.

Imagine a scenario in which a bomb detonates in a very public place leaving lots of people trapped and hurt. In addition, the bomb damages the communication network. In this situation, first responders are equipped with smartphones and tablets with sensors and communication mechanisms that are particular to the type of emergency and network situation and can receive and install services on-the-fly. Disposable surrogates that can execute expensive computations and have access to the cloud are deployed in strategic locations. Robots with mounted cameras are sent in to explore the damaged areas and throwable cameras are used to create a picture of the damaged areas. Surveillance cameras in the area are automatically configured to capture high-resolution video of the affected areas.

As the scenario unfolds, first responder mobile devices use contextual information to adjust sensor sampling rates to extend battery life. Nearby mobile devices create an adhoc network to deal with the damaged network and combine to form a single platform where each devices performs certain tasks according to their capabilities. These mobile devices also reconfigure (manually or automatically) as deployed devices gather information about the context.

### 4.2.4 Open Challenges

Satisfying the scenario that was just described requires addressing a number of open challenges related to the customization of mobile service platforms:

- Challenges introduced by mobility: The main challenge is ephemerality — they don't last long and you don't know when they will fail. Because of limited resources, platforms and applications have to be very efficient in terms of energy and bandwidth usage and have to deal with unstable connection.
- Privacy, Security and Trust: Mobile devices that act as customizable service platforms have many issues related to data privacy and trust in new features.
- Multiple stakeholders have different concerns that could be in conflict — users, mobile peers, providers (platform, apps, network, storage), government agencies, and certification organizations.
- Multiple Roles: In a multi-platform scenario, devices may play multiple roles at different points in time — consumer, provider, host. Switching between roles requires, in addition, continuous discovery and coordination.
- Business models: Creating a business model that motivates users to add services on-the-fly is a challenge. There may be the need for third-party certifiers that certify that services do what they say they do.

## 4.3 Architecting for Platform Customization

*Damian Andrew Tamburri (VU University Amsterdam, NL)*

*Participants:* Florian Rosenberg, Cesare Pautasso, Damian Andrew Tamburri, Leonardo Passos, Nenad Medvidovic, Manuel Resinas Arias de Reyna, Patricia Lago, Peter Dolog, Gregor Engels, Nanjangud C. Narendra, Klaus Schmid[1]

The group discussed the architectural implications and underpinning of platform customization problems. First, the group explored the architecture decision to adopt certain technologies as opposed to others, for adaptation and architecture flexibility. These technologies include: REST vs JSON/RPC vs SOAP/RPC vs MQ. These decisions are revisited when needs arise for (re-)integration of platforms for networked organizations. These scenarios require a customization cutting across service definitions. The following discussions rotated around the following research questions:

**RQ 1:** What is customization?
- Depends on the context (service consumer, provider, platform)
- From consumer perspective it is: "Service selection, service configuration, platform configuration/constraint"
- From consumer/provider perspective it is: "Negotiable, flexible billing/accounting model"
- From platform: "Resource allocation, platform service selection"

**RQ 2:** How is customization expressed at the interface/service abstraction level?
- Request/job submission metadata/constraints
- Platform feature selection, activate/deactivate services for which you will be charged
- Product lines?

**RQ 3:** What are the platform mechanisms to allow customization?
**Strategy pattern:** one abstract interface with multiple implementations
**Extension point:** plugin additional implementations
**Architectural patterns for customization** *such as?*
**Tuning/controller component for self-adaptive architectures**

**RQ 4:** How can you design an architectural style for service-oriented architectures that facilitates customization?
- Loose coupling -> easier to customize
- Granularity of services: small -> easy to recompose
- Formalize customization with a customization algebra

**RQ 5:** How are existing services/platforms customizable? How are customization done today? — How is this unique to services/service platforms? How are different "features" of the platform customized and exposed?

- Separate branches in the code, then compile and deploy separate branches to enable different customizations
- One branch with feature toggles and turn toggles on and off at runtime through configuration

[1] Further information can be found at http://ep.sonyx.net:9000/dagstuhl

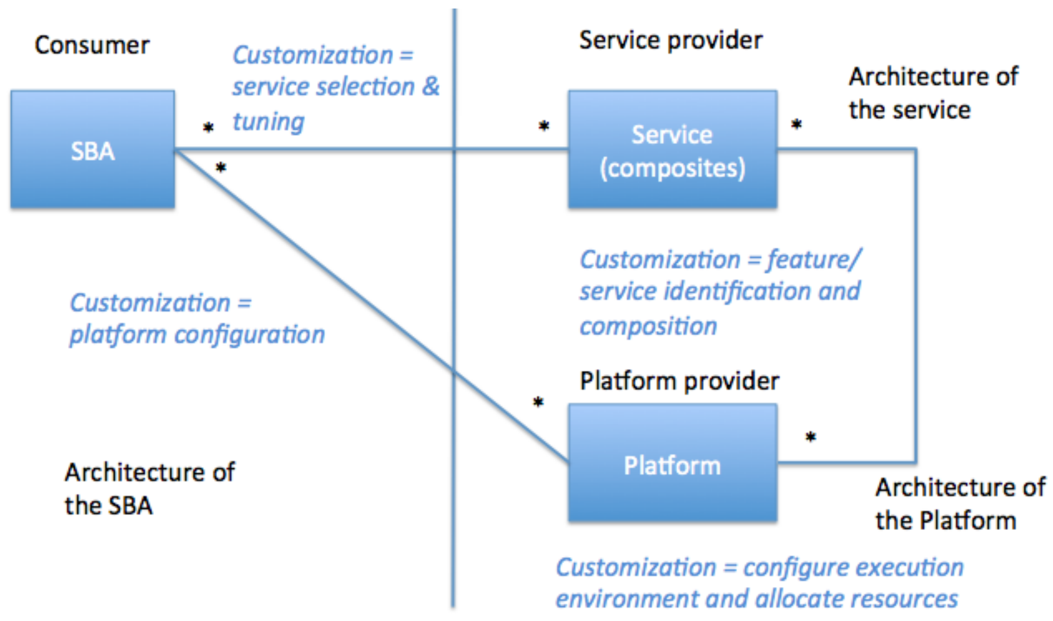

Consumer
SBA
Customization = service selection & tuning
Service provider
Service (composites)
Architecture of the service
Customization = feature/ service identification and composition
Customization = platform configuration
Platform provider
Platform
Architecture of the SBA
Architecture of the Platform
Customization = configure execution environment and allocate resources
*

**Figure 1** Architecture of the SBA.

- UI-level composition of widgets
- #ifdef
- if (config)
- Interface o = Class.new(config)
- Dynamic discovery and composition as a form of customization
- AOP for Devops languages

What are typical binding times:
- Very early (design time, select-integrate-test)
- Early (compile, deploy)
- Late (deploy, runtime)
- Very late (runtime after failure)

It is important to include the customization context as shown in Figure 1.

## 4.4 Energy-Aware Customization

*Patricia Lago (VU University Amsterdam, NL)*

*Participants:* Patricia Lago, Luciano Baresi, Sam Guinea, Grace Lewis, Marco Aiello, Holger Eichelberger, Nenad Medvidovic, Antonio Ruiz Cortez, Jacek Serafinski, Wenjun Wu

The group discussed what energy-aware customizations from the platform level up to the application level should entail. Requirements include measure, platform self-optimization, and mapping of the elements that belong to an energy context, both within and across levels. The unanimous conclusion was that:

1. past research in optimizations driven by scarcity of resources could be partially applicable for reducing energy consumption, *and that*
2. new research is needed due to the complexity of the current IT contexts, and due to the fact that energy efficiency requires tradeoffs between optimization costs and energy savings.

While promising, this area needs much more research in the future.

## 4.5 Customizing Service Platforms for Cloud Computing

*Cesare Pautasso (University of Lugano, CH)*

*Participants:* Florian Rosenberg, Waldemar Hummer, Christian Inzinger, Cesare Pautasso, Manuel Resinas, Peter Dolog, Klaus Schmid

Service Platforms for Cloud Computing are highly customizable. In this working group we have analyzed the variability points for Infrastructure-as-a-Service (IaaS) offerings and performed a practical comparison of concrete public cloud platforms (Amazon EC2, Microsoft Azure, Google Compute) and also the OpenStack framework for private clouds.

The variability points concern:

- the mechanism used for the compute, storage, and network abstraction offered by the platform
- the possibility of configuring image flavours (and how these flavours are defined)
- the way images are managed and whether it is possible to bring-your-own customized images for deployment
- the possibility to customize images upon their activation
- the structure of the metering and billing model of the provider (whereas most providers differ in the way they charge for using their infrastructure, it is not always possible to negotiate customized billing agreements with a provider)
- the mechanism used for offering elastic scalability (and whether it is possible to customize it with specific policies)
- the set of security mechanisms that can be enabled and the corresponding policies
- the presence of explicit "geographic regions" and how these are represented.

## 4.6 Customizing Service Platforms for Agile Networked Organizations

*Damian Andrew Tamburri (VU University Amsterdam, NL)*

*Participants:* Uwe Zdun, Georg Leyh, Karina Villela, Gregor Engels, Tiziana Margaria, Andreas Rummler, Deepak Dhungana, Nanjangud Narendra, Frank van der Linden, Damian Andrew Tamburri

The group explored the challenge of understanding how to customise service platforms to allow IT-intensive organizations to network with each other. Organizational networking

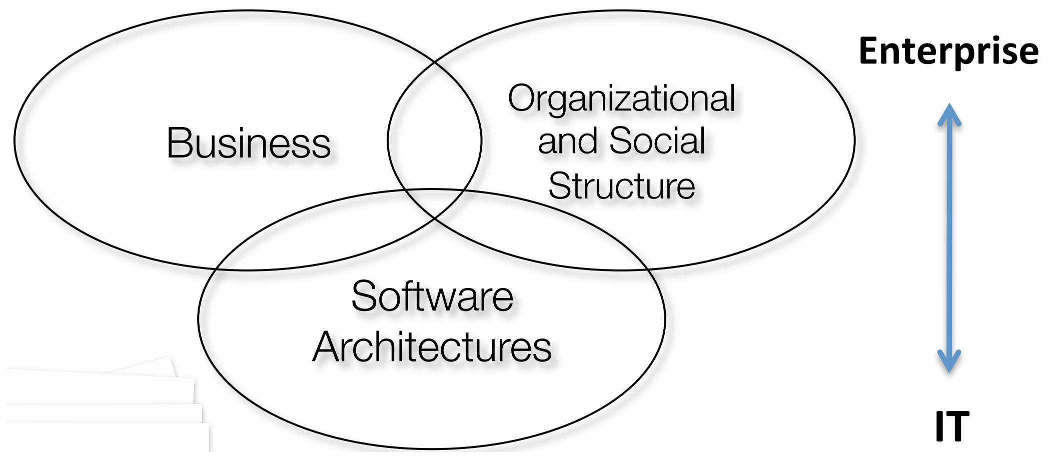


**Figure 2** Relationship of Business, Organization, and Architecture in Networked Organizations.

scenarios include any situation in which a set of organizaitons partner up to achieve shared business goals. Similarly, organizations need to network when one organization subsumes or merges with another one. Making this transition into an "agile", i.e., adaptable, smooth, and painless transition is still a big challenge.

The group agrees that the customization problem at hand is a "Business-IT alignment" problem, since the agile networked organization stems to align business drivers with IT-Architecture and vice versa. Also, the problem is strongly affected by market speed, key driver for organizational dynamism. The organizational artefact, which is being adapted in the process of organizational networking, is the organizational and social structure emerging between networked organizations.

To evaluate strategies and approaches currently adopted by companies to tackle this problem, the group evaluated industrial scenarios of networked organizations during companies merging.

*SCENARIO 1:* when Philips Corp. acquires companies, architects from both companies use software architecture as a brokering artefact to decide which IT infrastructure in either company needs adaptation to the other one. The "best" architecture between the two companies is used as a basis for integration/customization to support the networked organization. This process also requires to "decorate" the software architecture with business (e.g. business processes, business requirements, etc.) and organizational/social structure information (roles, responsibilities, locations, governance guidelines, etc.). Reference frameworks exist within Philips to drive this process but are currently an industrial secret. *NEED: research into networked organizations creation and governance.*

*SCENARIO 2:* when IBM acquires a company, they integrate the company's software products into IBM's product portfolio via a process known as "blue washing". This makes the creation of the networked organization much simpler and smoother, but has the potential to create social and organizational integration issues. *NEED: research into understanding and mitigation of social aspects for networked organizations.*

The group agrees that additional requirements come from the intense and increasing presence of service-based and cloud-based technologies. The problem of networked organizations in the cloud is still an open problem. Also, architecting for networked organizations is still an open problem. Both problems require the definition of architectural viewpoints and reference frameworks to specify and analyse software architectures from four perspectives:

1. Business
2. IT
3. Organization
4. Organizational Process

## 4.7 Binding time aspects of service platform customization

**Customizing Service Platforms - Development time vs. Compile time vs. Runtime**

*Holger Eichelberger (Universität Hildesheim, DE)*

*Participants:* Marco Aiello, Christian Inzinger, Jacek Serafinski, Holger Eichelberger

This group discussed temporal aspects of the customization of service platforms, in particular, the role of (self-)adaptation as a customization technique. This section summarizes the main findings of the group discussions in terms of (agreed) terminology, examples and scenarios for customizations at different points in time as well as research challenges.

### 4.7.1 Main Questions and Terminology

Several questions and terminology issues were discussed:

- What is the system being configured, i.e., what shall be subject to configuration? In this discussion, the group considered traditional systems (software product lines), service platforms, as well as service ecosystems.
- What is the semantics of time with respect to a customization? The group discussed two alternatives:
  - **Application time:** This denotes the point during the software production process when the customization is (actually) applied.
  - **Binding time:** The latest point in time when the decision for a customization must be made. This complies with the typical notion of binding time in Software Product Lines. In particular, a customization may be applicable at multiple binding times.

  During the discussion, the group adopted the binding time semantics.
- Which types of customizations are relevant and can those types be (partially) ordered? The group identified the following types:
  - **Configuration:** Simple as well as sophisticated mechanisms to configure a system, its settings or its code. Although pre-runtime configuration is frequently applied in traditional software product lines, configuration techniques may also be applied at runtime.
  - **Adaptation:** An external mechanism defines the configuration of the system at runtime. This does not imply that the adaptation mechanism itself can also be configured (through appropriate configuration approaches).
  - **Self-adaptation:** The system itself determines and applies appropriate configurations during its runtime. Thus, the adaptation mechanism itself may be subject to configurations within the same system.

  The group agreed that the given sequence expresses a (partial) order of increasing flexibility (and typically also complexity) of the underlying configuration mechansisms. While some work, e.g., [1, 2, 4, 7], make the distinction between adaptive and self-adaptive

systems, the group finally decided to focus only on two types, namely *Configuration* and *Self-adaptation*.
- Can a combination of customization type and the time aspect be used to classify existing systems and their applied customization mechanisms, in particular with respect to the timing aspects discussed by the group? The implied space can be considered as a coordinate-system with two axis, namely customization time and binding time. As discussed, the customization type depends on the (binding) time aspect, i.e., (self-)adaptation can only be applied during runtime of the system, e.g., for startup, initialization, or runtime customizations.

## 4.7.2 Examples / Scenarios

The group identified several examples and scenarios for applying customizations at different (binding) times. In particular, the group focused on service-based systems, service platforms, and service ecosystems. These examples were classified as shown below:
- **Configuration at development time:** Decisions in architecture, manual customizations in code, the build process, etc. The group considered this as a typical approach to the customization of systems and did not discuss specific scenarios.
- **Configuration at compile time:** This is most frequently applied in traditional software product lines. However, customization at compile time is not uncommon in service-oriented systems as well [3]. In particular, the group discussed the approach of the SAS-LeG project (Software As Service for the varying needs of Local eGovernments)[2], where (static) customization at compile time is applied to customize services for the needs of several municipalities in the Netherlands.
- **Configuration at deployment time:** In the INDENICA warehouse management case study [5], customizations are applied prior to or at deployment time so that the customized service platform and the services become active at deployment.
- **Configuration at startup time:** Software in this category reads configuration files and binds the customization with specific values at that point in time, i.e., during early runtime. Examples for traditional software systems are Apache `httpd`[3] or relational database management systems such as `MySQL`[4]. Further, all three service-based case studies in the INDENICA-project [5], namely the warehouse management system, the yard management system or the remote management system also rely on startup time mechanisms.
- **Configuration at runtime:** One particular example is the INDENICA warehouse management case study [5], where the user may determine and change the actual binding of customizations at runtime (although these capabilities are introduced at compile time).
- **Adaptation at runtime:** Here, in particular semi-automated or automated mechanisms such as adaptivity managers may change the actual customization of a system at runtime. This is for example applied in the SM4ALL project (Smart Homes for All)[5] on an embedded middleware for pervasive and immersive environments in order to enable a continuous adaptation of sensors, devices, services and appliances to user context and habits. Further, the virtual service platform researched in the INDENICA-project[6], which

[2] http://www.sas-leg.net/web/
[3] http://httpd.apache.org/
[4] http://www.mysql.com
[5] http://http://www.sm4all-project.eu/
[6] http://www.indenica.eu

integrates the INDENICA case studies mentioned above, contains an adaptation manager [6], which affects the configurations of the integrated customizable platforms, in particular the remote management system.

- **Customization at late runtime:** This class encompasses systems, which enable (open) customizations during runtime, which may even extend the system. Basically, this class includes multi-tenant systems or concepts relized by user-programmable systems such as `Cloud9`[7] or the `Heroku`[8] ecosystem.

The group assigned the examples and scenarios listed in this section to the two-dimensional coordinate system sketeched above. All examples and scenarios in this section are located below the bisecting line, which separates static from dynamic customizations. According to our findings, there is a clear relationship between binding time and flexibility, i.e., the later the binding time, the more flexibility is supported by the customizations, while extremely dynamic and flexible customizations are not applied at early binding times such as design or compile time.

### 4.7.3 Challenges

During the discussion we identified the following challenges:

- **Quantify the trade-offs among different (binding) times** in order to determine (relative) benefits, impacts or even risks. This quantification may happen in terms of metrics such as costs, revenue or downtime. Such tradeoffs enable the objective selection among available binding times for an individual customization opportunity, to support (risk) mitigation strategies or even to support temporal relationships across applied customizations, e.g., to trace failures to (combinations of) customizations applied at different points in time.
- **Support understandability of managing customizations at different points in time**: Are the actual capabilities sufficient for modeling and managing customizations which apply at different points in time? How can the different roles in the software development process be supported in understanding the effect (and the impact) of temporal customization aspects, in particular in the dynamic and open environment of service platforms?
- **Ensure semantically meaningful configurations when temporal aspects become customization alternatives.** This includes consistency issues (avoid selecting the wrong service, the wrong service interface or wrong data) or means to show the correctness of configurations.
- **Combine platform evolution and service platform lifecycle management with temporal configuration aspects.** How can upgrades of services and service platforms with temporal customizations be upgraded? How can staged upgrades be supported? How can integrity and consistency of service bindings and data be guaranteed?
- **Analyze the mutual influence of temporal customization aspects on the openness of service platforms.** By construction, service platforms support open-world scenarios, e.g., services can be discovered and bound at runtime. Do temporal aspects introduce another dimension of openness? How can openness be considered in the challenges state above, for example, how do customizations interrelate with openness and

[7] https://c9.io
[8] https://www.heroku.com

extensibility, e.g., in terms of customizable extension bundles with own development lifecycle?

- **Analyze the impacts of temporal aspects in customizing multi-tenant environments**, e.g., with respect to tenant-specific isolation mechanisms (regarding resource usage, data and tenant-specific functions) or mapping of functionality or resources to physical infrastructures. Shall (temporal) customization be available for tenant-specific extensions (e.g., as part of a development introduce new temporal customization aspects such as a replication of binding times (a "development" time as part of runtime) or can this be considered as views (the system is still at runtime while the tenant has its own time scale).

### References

1 N. Abbas. Towards autonomic software product lines. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, SPLC '11, pages 44:1–44:8, New York, NY, USA, 2011. ACM.

2 N. Abbas, J. Andersson, and D. Weyns. Knowledge evolution in autonomic software product lines. In *Proceedings of the 15th International Software Product Line Conference, Volume 2*, SPLC'11, pages 36:1–36:8, New York, NY, USA, 2011. ACM.

3 H. Eichelberger, C. Kröher, and K. Schmid. Variability in Service-Oriented Systems: An Analysis of Existing Approaches. In *Conf. on Service-Oriented Computing (ICSOC'12)*, pages 516–524, 2012.

4 D. Garlan, B. Schmerl, and S.-W. Cheng. Software architecture-based self-adaptation. In Y. Zhang, L. T. Yang, and M. K. Denko, editors, *Autonomic Computing and Networking*, pages 31–55. Springer US, 2009.

5 INDENICA project consortium. Description of Feasible Case Studies. Technical Report Deliverable D5.1, 2011. http://www.indenica.eu [validated: April 2013].

6 INDENICA project consortium. Report Describing a Framework for Deployment, Monitoring & Controlling of Virtual Service Platforms. Technical Report Deliverable D4.1, 2012. http://www.indenica.eu [validated: April 2013].

7 M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

## 5 Open Problems

Open problems were described throughout the previous sections, in particular, in the working group summaries.

## Participants

- Marco Aiello
University of Groningen, NL
- Luciano Baresi
Polytechnic Univ. of Milan, IT
- Karina Barreto Villela
Fraunhofer IESE – Kaiserslautern, DE
- Deepak Dhungana
Siemens AG – Wien, AT
- Peter Dolog
Aalborg University, DK
- Schahram Dustdar
TU Wien, AT
- Holger Eichelberger
Universität Hildesheim, DE
- Gregor Engels
Universität Paderborn, DE
- Sam Guinea
Politecnico di Milano, IT
- Waldemar Hummer
TU Wien, AT
- Christian Inzinger
TU Wien, AT
- Patricia Lago
Free Univ. of Amsterdam, NL
- Grace A. Lewis
Carnegie Mellon University, US
- Georg Leyh
Siemens AG – Erlangen, DE
- Tiziana Margaria
Universität Potsdam, DE
- Nenad Medvidovic
USC – Los Angeles, US
- Nanjangud C. Narendra
IBM India – Bangalore, IN
- Leonardo Passos
University of Waterloo, CA
- Cesare Pautasso
University of Lugano, CH
- Manuel Resinas Arias de Reyna
University of Sevilla, ES
- Florian Rosenberg
IBM TJ Watson Res. Center – Yorktown Heights, US
- Antonio Ruiz Cortés
University of Sevilla, ES
- Andreas Rummler
SAP Research Center – Dresden, DE
- Klaus Schmid
Universität Hildesheim, DE
- Jacek Serafinski
NextDayLab Sp. z o.o. – Poznan, PL
- Damian Andrew Tamburri
VU – Amsterdam, NL
- Frank van der Linden
Philips Medical Systems – Best, NL
- Wenjun Wu
Beihang University – Beijing, CN
- Uwe Zdun
Universität Wien, AT

Report from Dagstuhl Seminar 13181

# VaToMAS – Verification and Testing of Multi-Agent Systems

**Edited by**

**Alessio R. Lomuscio**[1], **Sophie Pinchinat**[2], **and Holger Schlingloff**[3]

1 **Imperial College London, GB,** `A.Lomuscio@imperial.ac.uk`
2 **IRISA – Rennes, FR,** `sophie.pinchinat@irisa.fr`
3 **HU Berlin, DE,** `hs@informatik.hu-berlin.de`

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 13181 "VaToMAS – Verification and Testing of Multi-Agent Systems".

**Seminar** 28. April–03. May, 2013 – www.dagstuhl.de/13181
**1998 ACM Subject Classification** I.2.11 Distributed Artificial Intelligence, F.3.1 Specifying and Verifying and Reasoning about Programs, I.2.4 Knowledge Representation Formalisms and Methods
**Keywords and phrases** Model checking, Specification-based testing, Multi-agent systems, Controller synthesis, Temporal logic
**Digital Object Identifier** 10.4230/DagRep.3.5.151
**Edited in cooperation with** Bastien Maubert

## 1 Executive Summary

*Alessio R. Lomuscio*
*Sophie Pinchinat*
*Holger Schlingloff*

Multi-agent systems (MAS) are distributed computing systems in which the individual components, or agents, interact with each other by means of communication, negotiation, cooperation etc., in order to meet private and common goals. The agent model finds applications in a variety of key applications of high-impact to society including web-services, autonomous vehicles, and e-government. But if MAS are to deliver on their promise to drive future applications, they need to be reliable.

MAS are typically specified and reasoned about by a variety of modal formalisms, including a variety of different logics. There are presently several, compartmented communities tackling questions pertaining to the correctness of MAS: researchers in model checking, model based testing, and controller synthesis. There presently is very little personal interaction among the scientists from different communities. The aim of this seminar was to bring these communities together, get exposure to each others' solutions to similar aims, and ultimately enhance their future interaction.

The topics concentrated on the intersection of the fields:

- Model checking of temporal-epistemic logic, alternating logics, and BDI logics
- Model based test generation for embedded systems
- Controller synthesis for self-organizing systems

In model checking, usually a model of the system and a property to be verified are given. In model based test generation, the goal is to construct a test suite from a model which establishes confidence in a certain property. In synthesis, a property and a model of computation are given, from which a strategy (a system model) is to be built. Both the test generation and the controller synthesis problem are closely related to model checking – in order to check the satisfiability of certain alternating time temporal logic (ATL) formulas in a model, one needs to construct a strategy for the participating agents.

The purpose of the seminar was to establish a common understanding of the problems in the different technologies of these application areas. It was expected that increased interaction between these three fields would stimulate new results and techniques of both theoretical relevance and practical usefulness.

Besides survey talks (60 minutes) on common technologies, attendees gave short contributions (30 minutes) and lightening presentations (15 minutes) on current research results and discussion rounds on open problems and research agendas. Additional technical sessions, including software demos, were organised spontaneously by the attendees for two of the evenings.

Attendees also contributed to the seminar by taking part in the lively discussions organised on topics of importance in the area. These were held in some of the afternoons but also at during informal occasions outside the usual seminar hours such as after dinner. This helped bridge some of the gaps between the subdisciplines and rectify some misconception about each other's work.

Specifically, research topics of the seminar included:

- Logics and specification formalisms for MAS
- Verification and model checking for interactive systems
- Model-based testing for MAS
- Explicit, symbolic, and SAT-based algorithms for module checking
- Test case generation and synthesis
- Synthesis of winning strategies for games

The goals of the seminar were

- to obtain a common understanding of base technologies and intersections between these topics
- to collect a state-of-the-art picture of recent research results in the fields
- to confront methods from model checking and test generation for MAS
- to clarify terminology, research agendas and open problems
- to define a set of benchmark problems for verification and testing of MAS
- to bring together different communities formerly not interacting

The research topics were also discussed in relation with embedded systems applications such as:

- Verification of cyber-physical systems
- Validation of autonomous robots

It was felt that the seminar helped the participants to reach a common and shared understanding on the roles of logic, verification and testing as well as their interplay in the context of multi-agent systems

## 2 Table of Contents

# 3 Overview of Talks

## 3.1 The Social Laws Paradigm for Coordinating Multi-Agent Systems

*Thomas Ågotnes (University of Bergen, NO)*

**Joint work of** Ågotnes, Thomas; van der Hoek, Wiebe; Wooldridge, Michael

Social laws (or normative systems) have emerged as a natural and powerful paradigm for coordinating such systems, exposing the whole spectrum between fully centralised and fully decentralised coordination mechanisms. A social law is, intuitively, a constraint on the behaviour of agents, which ensures that their individual behaviours are compatible. In a standard multi-agent state transition diagram (where transitions are labelled with the name of the agent effecting the transition), a social law is simply a labelling of the transitions saying whether or not they are "legal", or "desirable". An illegal transition could, for example, correspond to a component malfunctioning. Different social laws give rise to different global system properties, assuming that the social law is complied with. Such properties can be specified and verified using temporal modal logic and model checking. In the talk I will introduce and motivate the idea of social laws, and show how many key social laws verification problems can be solved using model checking. Key questions involve the notion of compliance. First, when is it rational for an agent to comply? Since the properties of the resulting system depend on compliance of all the agents in the system (think of the social law "drive on the right side of the road"), this requires a game theoretic analysis. Second, how can we identify the most important agents/components in the system, the agents whose compliance is crucial for the proper functioning of the system? I will talk about some resulting decision problems, combining logic, game theory, voting theory and complexity theory.

## 3.2 (Really) Dynamic Modal Logics

*Carlos Areces (Universidad Nacional de Cordoba, AR)*

Different Dynamic Modal Logics have been investigated in the literature (e.g., Propositional Dynamic Logics). Interestingly, the semantics of most of these logics is actually *static*: the model over which a formula is evaluated never changes. We are interested in logics with operators that can actually alter the model during evaluation. We will present a number of these operators, discuss their expressive power, and the complexity of the model checking and satisfiability problems.

## 3.3 From Control Theory to Game Theory via LTLKc

*Guillaume Aucher (INRIA Bretagne Atlantique – Rennes, FR)*

Supervisory control theory as initiated by Ramadge and Wonham has been developed independently from game theory. In this talk, we show that it is possible to embed infinite two-player games with imperfect information and safety objective into an extension of supervisory control theory with infinite words. In order to do so, we use an epistemic temporal logic as a lingua franca, and we reformulate the problems of supervisory control theory and infinite games with imperfect information into this logic. As a result of this embedding, we are able to compute and characterize completely in terms of model checking problems the set of winning strategies in a two-player game with imperfect information and safety objectives.

This talk is based on a paper available as a technical report at the HAL archive http://hal.inria.fr/.

## 3.4 On Decentralized Runtime Verification Techniques

*Ezio Bartocci (TU Wien, AT)*

Most of computing devices produced nowadays are embedded systems employed to monitor and control physical processes: cars, airplanes, automotive highway systems, air traffic management, etc. In all these scenarios, computing and communicating devices, sensors monitoring the physical processes and the actuators controlling the physical substratum are distributed and interconnected together in dedicated networks. The formal verification of these systems consists in proving that their execution satisfies a given specification of what their possible behaviors should be. When a system model is available and has a finite number of states, an answer is provided by applying the classical Model Checking technique. In this case, if the system is found to violate the property, a counterexample in the form of a system execution is generated and it can be used to debug the system model or property. The main drawback of this technique is that usually the number of states of a model grows exponentially in the number of its parameters. Furthermore, often the models are not available, leading us to consider them simply as black-boxes where only input and output can be observed. In this case, monitoring their execution provides a still valid alternative verification technique. Decentralized runtime verification refers to a monitoring technique, where each component must infer, based on a set of partial observations, if the global property is satisfied. In this talk I will present an overview of the problem, the available techniques, some of my research results and points of discussion.

## 3.5 Automatic Verification of Equational Knowledge in MAS

*Ioana Boureanu (EPFL – Lausanne, CH)*

Security protocols can be executed concurrently, their participants and their runs describing a MAS. However, as we may know, the knowledge of these agents is subject to the cryptographic abilities and, roughly speaking, to the cryptographic material that they hold. We would like to be able to describe this knowledge in a formal way. Per se, this is not novel. The element of originality is two-fold: 1) being able to treat many primitives, i.e., beyond simple encryption/decryption and, namely, the full class of subterm convergent equational, cryptographic theories; 2) finding a solution that lends itself to automation and then to unsupervised verification. We introduce a modality called rewriting knowledge that operates on local equational congruences. We discuss the conditions under which its interpretation can be approximated by a second modality called empirical knowledge. We report an implementation of a technique to verify this modality inside the open source model checker MCMAS. We evaluate the approach by verifying MAS models of electronic voting protocols automatically extracted from high-level descriptions.

## 3.6 Protocol Descriptions to Interpreted Systems

*Ioana Boureanu (EPFL – Lausanne, CH)*

**Main reference** I. Boureanu, A. Lomuscio, “PD2IS Tool”. May, 2013.
**URL** http://www.dagstuhl.de/mat/Files/13/13181/13181.BoureanuIoana2.ExtAbstract.pdf

PD2IS (Protocol Descriptions to Interpreted Systems) is a toolkit that we developed to generate MAS models upon standard security protocol semantics, e.g., we embedded the multiset rewriting semantics into IS models. The input to PD2IS is a file designating a CAPSL (Common Authentication Protocol Specification Language) protocol description together with some additional parameters to define a MAS instantiation. The output is an interpreted system in ISPL (Interpreted Systems Programming Language). PD2IS systematically generates a full taxonomy of propositions and temporal- epistemic formulae corresponding to expressions of the CAPSL goals. PD2IS automatically inserts these temporal-epistemic formulae in the ISPL file for the model under generation. MCMAS is called for each ISPL file produced by PD2IS. MCMAS returns the calls either by certifying that the specifications are satisfied or by returning detailed counterexamples. These are used by PD2IS to report details of the attack found on the protocol (i.e., the failure of one or more of formulae corresponding to the goals). PD2IS was used together with MCMAS and proved effective in verifying authentication, key-establishment (e.g., the Clark-Jacobs, SPORE libraries), e-voting protocols (FOO’92, Okamoto protocols), against classical security specifications (secrecy, authentication, vote-privacy, receipt-freeness, coercion-resistance, etc.), as well as novel, intrinsically epistemic security requirements, like attack-detectability.

## 3.7 Alternating Epistemic Mu-Calculus: Fixed-point Abilities under Incomplete Information

*Nils Bulling (TU Clausthal, DE)*

**Main reference** N. Bulling, W. Jamroga, "Alternating epistemic mu-calculus," in Proc. of the 22nd Int'l Joint Conf. on Artificial Intelligence (IJCAI'11), pages 109–114, Barcelona, Spain, July 2011.
**URL** http://ijcai.org/papers11/Papers/IJCAI11-030.pdf

The alternating-time temporal logic ATL is a well-known logic for reasoning about strategic abilities of agents. An important feature that distinguishes the variants of ATL for imperfect information scenarios is that the standard fixed-point characterizations of temporal modalities do not hold anymore. In this talk, I present the alternating epistemic mu-calculus [1]. The logic allows to capture abilities that could not be expressed in ATL. The new kind of ability allows agents to always recompute their strategy while executing it. Thus, the agents are not assumed to remember their strategy by definition. I will also briefly address the model checking problem and show that the verification of such abilities can be cheaper than for all the variants of "ATL with imperfect information" considered so far.

### References

1 Nils Bulling and Wojciech Jamroga. Alternating epistemic mu-calculus. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 109–114, Barcelona, Spain, July 2011.

## 3.8 Combining quantitative and qualitative strategic reasoning. Part II: some comparisons and preliminary results

*Nils Bulling (TU Clausthal, DE)*

**Joint work of** Bulling, Nils; Goranko, Valentin
**Main reference** N. Bulling, V. Goranko, "How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract)," in Proc. of the 1st Int'l Workshop on Strategic Reasoning, EPTCS, Vol. 112, pp. 33–41, 2013.
**URL** http://dx.doi.org/10.4204/EPTCS.112.8

In this talk I take our framework on quantitative and qualitative reasoning [1] (presented in Part I) up and propose the logic Quantitative ATL*. I present some preliminary model checking results and briefly discuss related work.

### References

1 Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *Proceedings of the 1st International Workshop on Strategic Reasoning*, Electronic Proceedings in Theoretical Computer Science, pages 33–41, Rome, Italy, March 2013.

## 3.9 Using AJPF to generate models of agent programs for input into other Model Checkers

*Louise A. Dennis (University of Liverpool, GB)*

AJPF is a program model checker for reasoning about agent systems programmed in BDI style languages. Following recent work from Raimondi et al., this has been adapted so that it can be used to generate a model of the system under test, and that model then used as the input for a more traditional model-checker. This presentation will give a quick overview of the AJPF adaptations, look at some preliminary results and discuss possible uses of the system.

## 3.10 Verifying Autonomous Systems

*Michael Fisher (University of Liverpool, GB)*

**Joint work of** Fisher, Michael; Dennis, Louise

As the internal architectures of autonomous systems become more complex, the need for their activities to be both understandable and explainable is increasing. This, combined with the development of agent model-checking techniques, is beginning to allow practical autonomous systems to be verified. Beyond straightforward analysis of functional requirements of autonomous systems, it is increasingly important to (logically) specify and verify both legal and ethical aspects. In this talk, we describe the problems (and partial solutions) associated with these aspects.

## 3.11 Resolution for Temporal Logics of Knowledge

*Michael Fisher (University of Liverpool, GB)*

**Joint work of** Fisher, Michael; Dixon, Clare; Nalon, Claudia

I will briefly outline our work on automated proof methods, particularly clausal resolution methods, for deciding temporal logics of knowledge, belief, etc. The short talk will also touch on current implementation technology for such systems.

## 3.12 Information values in multi-agent bargaining scenarios

*Tim French (University of Western Australia – Nedlands, AU)*

This talk will discuss ongoing work investigating the role information and uncertainty has in multi-agent bargaining scenarios. Game theoretic analyses of bargaining and auctions are well-established. We are interested in the interaction between the bargaining process and epistemic state of the agents involved. In one direction we may consider the question of releasing sufficient information to enable agents to find a stable bargaining solution. In the other direction we can consider the problem of determining and quantifying the amount of information an agent acquires by participating in a bargaining process. We will describe some related work and some broad goals in terms of assigning value to information, and evaluating economic processes with respect to the relative uncertainty of the participating agents.

## 3.13 The synthesis and actuation of informative events

*Tim French (University of Western Australia – Nedlands, AU)*

This talk will describe some recent and ongoing work in the area of synthesising information updates, and executing informative updates through message passing systems. Particularly, James Hales has recently completed work on the synthesis of uniform action models to realise a given epistemic property within the multi-modal K. We will discuss this result, extensions and applications that it may have.

## 3.14 Combining quantitative and qualitative strategic reasoning. Part I: framework

*Valentin Goranko (Technical University of Denmark, DK)*

There are several traditions in studying strategic abilities of agents to achieve objectives in multi-player games, coming from game theory, logic and computer science. Game theory studies rational behavior of players, relevant for their achievement of quantitative objectives: optimizing payoffs (e.g., maximizing rewards or minimizing cost) or, more generally, preferences on outcomes. On the other hand, logic mainly deals with abilities of players for achieving qualitative objectives of players: reaching or maintaining game states with desired properties, e.g., winning states or safe states. Put as a slogan, the former tradition is concerned with how a player can become maximally rich, or pay the least possible price, in the game, while the latter tradition – with how a player can achieve a state of 'happiness', or avoid reaching a state of 'unhappiness', in the game. Studies in computer science have involved both quantitative and qualitative objectives, usually considered separately, but there is an increasingly active recent trend to consider games with combined objectives. We propose a logical framework, first presented in [1], combining the two traditions by enriching concurrent game models with payoffs for the normal form games associated with the states and with guards on the actions available to players in terms of their payoffs. Respectively, we propose a quantitative extension of the logic ATL* enabling the combination of quantitative and qualitative reasoning. In the Part I of this presentation I introduce and discuss the framework. Part II of the talk, presented by Nils Bulling, discusses the model-checking problem for the Quantitative ATL* on concurrent game models with payoffs, mentions some decidability and undecidability results and some related work.

**References**

1 Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *Proceedings of the 1st International Workshop on Strategic Reasoning*, Electronic Proceedings in Theoretical Computer Science, pages 33–41, Rome, Italy, March 2013.

## 3.15 A few remarks about related work in Pretoria

*Stefan Gruner (University of Pretoria, ZA)*

I am currently not working in the field of multi-agent systems (MAS). In a very short statement on the last day of the seminar I mentioned that I had only two older publications in the context of MAS, namely [1] [2]; however the 'agents' in those publications need not be 'intelligent' in the sense of AI. With my main interest in software engineering methodology I participated in the seminar's discussion sub-group on the topic of a to-be-developed MAS-specific theory of software testing that must reach beyond the theory of testing for classical (non-MAS) software systems.

**References**

1 Bilel Derbel, Mohamed Mosbah, Stefan Gruner. *Mobile Agents implementing Local Computations in Graphs.* Lecture Notes in Computer Science 5214, pp. 99-114, Springer-Verlag, 2008.

2 Stefan Gruner. *Mobile Agent Systems and Cellular Automata.* Journal for Autonomous Agents and Multi-Agent Systems 20/2, pp. 198-233, Springer-Verlag, 2010.

## 3.16 Yet Another Modal Notation for Strategy Contexts

*Dimitar Guelev (Bulgarian Academy of Sciences – Sofia, BG)*

I highlight an extension of ATL with knowledge [3] and both knowledge and contexts [2]. It admits some interesting instances of a proof rule which was introduced for $\mathrm{PDL}^{\cap}$ [1]. To make the transition to strategy-context enabled semantics, just one truly specific axiom, which is taken from [4], appears necessary. Comparison with some other existing ATL-based notations for strategy contexts shows them to be of the same expressive power in the case of complete information.

Dimitar P. Guelev was partially supported by Bulgarian NSF Grant DID02/32/2009. He is grateful to the organizers, and especially to Bernd-Holger Schlingloff for his careful guidance.

**References**

1 Philippe Balbiani and Dimiter Vakarelov. Iteration-free PDL with Intersection: a Complete Axiomatization. *Fundam. Inform.*, 45(3):173–194, 2001.
2 Dimitar P. Guelev and Catalin Dima. Epistemic ATL with Perfect Recall, Past and Strategy Contexts. In Michael Fisher, Leon van der Torre, Mehdi Dastani, and Guido Governatori, editors, *CLIMA*, volume 7486 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 2012.
3 Dimitar P. Guelev, Catalin Dima, and Constantin Enea. An Alternating-time Temporal Logic with Knowledge, Perfect Recall and Past: Axiomatisation and Model-Checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.
4 Dirk Walther, Wiebe van der Hoek, and Michael Wooldridge. Alternating-time Temporal Logic with Explicit Strategies. In Dov Samet, editor, *TARK*, pages 269–278, 2007.

## 3.17 The grand game of testing

*Yuri Gurevich (Microsoft Research – Redmond, US)*

We present testing as a game between the Programmer and the Tester. While the game is usually more cooperative than antagonistic, Tester's goal is different from that of Programmer. We discuss when the game is over and address Myers's paradox: "The number of uncovered bugs in a program section is proportional to the number of discovered bugs in the section."

## 3.18 Managing Policies and Trust

*Yuri Gurevich (Microsoft Research – Redmond, US)*

**Joint work of** Blass, Andreas; Guido de Caso; Gurevich, Yuri
**Main reference** A. Blass, G. de Caso, Y. Gurevich, "An introduction to DKAL," Microsoft Research Tech Report MSR-TR-2012-108.
**URL** http://research.microsoft.com/en-us/um/people/gurevich/Opera/216.pdf

With the advent of cloud computing, the necessity arises to manage policies and trust automatically and efficiently. In a brick-and-mortar (B&M) setting, clerks learn unwritten policies from trustworthy peers. And if they don't know a policy, they know whom to ask. In the B&M-to-cloud transition, the clerks disappear. Policies have to be explicit and managed automatically. The more challenging problem yet is how to handle the interaction of the policies of distrustful principals, especially in federated scenarios where there is no central authority. The DKAL project (Distributed Knowledge Authorization Language) was created to deal with such problems. The new language, new logics and tools keep evolving. We discuss the current state of the project.

## 3.19 Logics for Multi-Agent Systems

*Andreas Herzig (Université Paul Sabatier – Toulouse, FR)*

I classify various logics for MAS according to the epistemic and the action dimension. I highlight problematic aspects of each of the standard accounts, including the frame problem, strategy contexts and uniform strategies.

## 3.20 Concepts, Agents, Strategies... and Coalitions. ATL goes (monadic) first order

*Wojtek Jamroga (University of Luxembourg, LU)*

**Main reference** W. Jamroga, "Concepts, Agents, and Coalitions in Alternating Time," in Proc. of the 20th European Conf. on Artificial Intelligence (ECAI'12), Frontieres in Artificial Intelligence and Applications, Vol. 242, pp. 438–443, IOS Press, 2012.
**URL** http://dx.doi.org/10.3233/978-1-61499-098-7-438
**URL** http://icr.uni.lu/wjamroga/papers/atl+dl12ecai.pdf

I consider a combination of the strategic logic ATL with the description logic ALCO. In order to combine the logics in a flexible way, I assume that every individual can be (potentially) an agent. I also take the novel approach to teams by assuming that a coalition has an identity on its own, and hence its membership can vary. In terms of technical results, I show that the logic does not have the finite model property, though both ATL and ALCO do. I conjecture that the satisfiability problem may be undecidable. On the other hand, model checking of the combined logic is decidable and even tractable. Finally, I define a particular variant of realizability that combines satisfiability of ALCO with model checking of the ATL dimension, and I show that this new problem is decidable.

### References

1 Wojciech Jamroga. *Concepts, Agents, and Coalitions in Alternating Time.* Proceedings of the 20th European Conference on Artificial Intelligence ECAI 2012, pp. 438–443, IOS Press, 2012.

## 3.21 ATL with strategy contexts – part 1

*François Laroussinie (Université Paris-Diderot – Paris, FR)*

ATLsc is an extension of ATL with strategy contexts: the agents are committed to their strategies during the evaluation of formulas. This makes a huge difference with standard ATL. In this first talk, we will discuss the expressive power of ATLsc. In particular we will give several examples of properties that can be expressed with this formalism.

## 3.22 ATL with strategy contexts – part 2

*Nicolas Markey (ENS Cachan, FR)*

**Joint work of** Laroussinie, François; Markey, Nicolas

This second talk (the first one is proposed by François Laroussinie) will focus on the decision procedures for ATLsc and its complexity (for model checking and satisfiability). For this we will use an extension of CTL with quantifications over atomic propositions (QCTL). Indeed we will see that (1) verification problems for ATLsc (or Strategy Logic) can be naturally reduced to a problem over QCTL, and (2) decision procedures can be described in a simpler way for QCTL.

## 3.23 Uniform Strategies

*Bastien Maubert (Université de Rennes 1, FR)*

**Joint work of** Maubert, Bastien; Pinchinat, Sophie
**Main reference** B. Maubert, S. Pinchinat, L. Bozzelli, "The Complexity of Synthesizing Uniform Strategies," in Proc. of the 1st Int'l Workshop on Strategic Reasoning, EPTCS, Vol. 112, pp. 115-122, 2013.
**URL** http://dx.doi.org/10.4204/EPTCS.112.17

We study a general notion of uniform strategies that subsumes several existing notions of strategies subject to some uniformity constraints, like for example in games with imperfect information or model checking games for dependence logic. We present a logical language to specify such uniformity constraints. This language is basically LTL augmented with a knowledge-like operator $R$, where $R\varphi$ means that $\varphi$ holds in all related plays. One particularity of this work concerns the semantics of the $R$ modality. Instead of choosing a specific relation over plays, like synchronous perfect-recall for example, we allow for any binary rational relation. This class of relations is very general, and in particular it contains all relations classically used in games with imperfect information and logics of knowledge and time (perfect/imperfect recall, synchronous/asynchronous...). Rational relations are recognized by finite state transducers, which allows us to study the decidability and complexity of synthesizing uniform strategies for different subclasses of rational relations. Our results imply the decidability of the model checking of LTLKn with asynchronous perfect recall, and more generally we have that the strategy problem in games with a winning condition expressed in LTLK is decidable as long as the relation that represents the knowledge is rational.

## 3.24 A Poor Man's Technique for Reasoning About Knowledge

*Stephan Merz (LORIA – Nancy, FR)*

Representing and reasoning about knowledge is fundamental for modeling and analyzing multi-agent systems. Several logics have been proposed that combine epistemic and temporal or dynamic operators, and that support reasoning about the state of knowledge of agents based on the information they observe, e.g. resulting from updates due to point-to-point communication or broadcasts. Specialized model checkers such as DEMO [3] implement efficient procedures for evaluating formulas written in logics such as Dynamic Epistemic Logic (DEL, [1]). We report on an experiment on encoding the semantics of DEL directly in a constraint solver and using that encoding for solving the well-known "Sum and Product" puzzle [2]. The models are written in TLA$^+$ and are evaluated using the constraint solving techniques implemented in ProB [4]. The running times compare very favorably with those reported for DEMO, indicating that general-purpose solvers may in certain cases be quite competitive, at least for prototyping verification engines for new logics.

**References**

1 H. van Ditmarsch, W. van der Hoek, B. Kooi. Dynamic Epistemic Logic. Synthese Library 337. Springer (2007).
2 H. van Ditmarsch, J. Ruan, R. Verbrugge. Sum and Product in Dynamic Epistemic Logic. J. Log. Comput. 18(4): 563-588 (2008).
3 Jan van Eijck. DEMO – A Demo of Epistemic Modelling. In: Interactive Logic. Proc. 7th Augustus de Morgan Workshop (J. van Benthem, D. Gabbay, B. Löwe, eds.) Texts in Logic and Games 1:305–363 (2007).
4 M. Leuschel, M. J. Butler: ProB: an automated analysis toolset for the B method. STTT 10(2): 185-203 (2008)

## 3.25 Reasoning About Strategy

*Aniello Murano (University of Napoli, IT)*

**Joint work of** Mogavero, Fabio; Perelli, Giuseppe ; Sauro, Luigi; Vardi, Moshe
**Main reference** F. Mogavero, A. Murano, M.Y. Vardi, "Reasoning About Strategies," in Proc. of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10), LIPIcs, Vol. 8, pp. 133–144, Schloss Dagstuhl, 2010.
**URL** http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2010.133

In open systems verification, to formally check for reliability, one needs an appropriate formalism to model the interaction between agents and express the correctness of the system no matter how the environment behaves. An important contribution in this context is given by modal logics for strategic ability, in the setting of multi-agent games, such as ATL, ATL*, and the like. In this talk, I will introduce Strategy Logic as a powerful logic framework for reasoning explicitly about strategies in multi-agent concurrent games. As a key aspect, SL uses first-order quantifications over strategies, where strategies are not glued to a specific agent, but an explicit binding operator allows to bind an agent to a strategy variable. This allows agents to share strategies or reuse one previously adopted. In this talk, I will discuss about the model checking and the satisfiability decision problems for SL and show that they are undecidable and NonElementarySpace-hard, respectively. This negative result has successfully spurred us to investigate syntactic fragments of SL, strictly subsuming ATL*, with elementary decision problems. In this talk I will present some of these fragments and discuss their properties.

## 3.26 Bounded model checking for LTLK

*Wojciech Penczek (Siedlce University of Natural Sciences and Humanities, PL)*

We present a novel approach to verification of multi-agent systems by bounded model checking for Linear Time Temporal Logic extended with the epistemic component (LTLK). The systems are modelled by two variants of interpreted systems: standard and interleaved ones. Our method is based on binary decision diagrams (BDD). We describe the algorithm and provide its experimental evaluation together with the comparison with another tool. This allows to draw some conclusions concerning which semantics is preferable for bounded model checking LTLK properties of multi-agent systems.

## 3.27 Abstract planning using genetic algorithms

*Wojciech Penczek (Siedlce Univ of Natural Sciences and Humanities, PL)*

**Joint work of** Niewiadomski, Artur; Penczek, Wojciech; Skaruz Jaroslaw
**Main reference** A. Niewiadomski, W. Penczek, J. Skaruz, "Towards automated abstract planning based on a genetic algorithm," Technical Report 1026, ICS PAS, Warsaw, 2012.
**URL** http://artur.ii.uph.edu.pl/papers/rep1026.pdf

The lecture is based on joint work [1], which discusses a new approach based on nature inspired algorithms to an automated abstract planning problem, which is a part of the web service composition problem. An abstract plan is defined as an equivalence class of sequences of service types that satisfy a user query. Intuitively, two sequences are equivalent if they are composed of the same service types, but not necessarily occurring in the same order. The objective of our genetic algorithm (GA) is to return representatives of abstract plans without generating all the equivalent sequences. The lecture presents experimental results, which show that GA finds solutions for very large sets of service types in a reasonable time.

### References

**1** A. Niewiadomski, W. Penczek, and J. Skaruz. Towards automated abstract planning based on a genetic algorithm. Technical Report 1026, ICS PAS, 2012. http://artur.ii.uph.edu.pl/papers/rep1026.pdf.

## 3.28 Tools for MAS verification: where do we go next?

*Franco Raimondi (Middlesex University – London, UK)*

A number of software tools are available for MAS verification, and their performance has improved by orders of magnitude in the past decade. However, most tools have their own input language and often specialize in one verification technology, or only support checking a specific type of property. As a result, the adoption of MAS verification tools is still very limited in the "real world". In this presentation we suggest a different approach to MAS verification: tools should be moved closer to real systems by means of (re-usable) connectors and libraries, and should be seen as components of a more general framework. We provide an example of this framework using the Brahms modelling language for MAS, and various model checkers to perform verification.

## 3.29 Doomsday Equilibria for Omega-Regular Games

*Jean-François Raskin (Université Libre de Bruxelles, BE)*

Two-player games on graphs provide the theoretical framework for many important problems such as reactive synthesis. While the traditional study of two-player zero-sum games has been extended to multi-player games with several notions of equilibria, they are decidable only for perfect-information games, whereas several applications require imperfect-information games. In this paper we propose a new notion of equilibria, called doomsday equilibria, which is a strategy profile such that all players satisfy their own objective, and if any coalition of players deviates and violates even one of the players objective, then the objective of every player is violated. We present algorithms and complexity results for deciding the existence of doomsday equilibria for various classes of $\omega$-regular objectives, both for imperfect-information games, as well as for perfect-information games. We provide optimal complexity bounds for imperfect-information games, and in most cases for perfect-information games.

## 3.30 Specification based testing in an institutional setting

*Markus Roggenbach (Swansea University, UK)*

**Joint work of** Phillip James; Faron F Moller; Hoang Nga Nguyen; Markus Roggenbach; Steve Schneider; Helen Treharne
**Main reference** F. Moller, H.N. Nguyen, M. Roggenbach, S. Schneider, H. Treharne, "Defining and Model Checking Abstractions of Complex Railway Models Using CSP||B," in Proc. of the Haifa Verification Conference (HVC'12), LNCS, Vol. 7857, pp. 193–208, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-39611-3_20
**Main reference** P. James, M. Trumble, H. Treharne, M. Roggenbach, S. Schneider, "OnTrack: An Open Tooling Environment for Railway Verification," in Proc. of the 5th Int'l NASA Symp. on Formal Methods, LNCS, Vol. 7871, pp. 435–440, Springer, 2013.
**URL** http://dx.doi.org/10.1007/978-3-642-38088-4_30

It is becoming common industrial practice to utilize Domain Specific Languages (DSLs) for designing systems. Such DSLs offer constructs native to the specific application area. Formal methods often fail to be easily accessible for engineers, but designs formulated in DSLs are open for systematic and, possibly, automated translation into formal models for verification.

In this talk, we show that DSLs also allow abstractions to be formulated at the domain level. We demonstrate on the example of the Railway domain that (1) such abstractions exists over the boundary of different specification languages (CSP, CSP||B, CASL) and (2) and demonstrate by the means of our tool OnTrack how to support & automatize such abstractions.

## 3.31 Efficient Testing of Software Product Lines

*Ina Schaefer (TU Braunschweig, DE)*

**Joint work of** Schaefer, Ina; Malte Lochau; Sascha Lity
**Main reference** M. Lochau, I. Schaefer, J. Kamischke, S. Lity, "Incremental Model-Based Testing of Delta-Oriented Software Product Lines," in Proc. of the 6th Int'l Conf. on Tests and Proofs (TAP'12), LNCS, Vol. 7305, pp. 67–82, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-30473-6_7

Testing software product lines by considering each product variant in isolation is impracticable due to the high number of potential product configurations. Therefore, applying reuse principles also to test artifacts in a concise way for efficient testing is essential. In this talk, I address this open issue by presenting a novel, model-based SPL testing framework based on reusable test models and incremental test suite evolution. Test artifacts are incrementally evolved for every product variant by explicitly considering commonality and variability between two subsequent products under test. I illustrate the framework by means of an automotive case study and compare our experimental results with alternative SPL testing strategies with respect to efficiency improvements.

## 3.32 On the specification and analysis of contracts (normative texts)

*Gerardo Schneider (University of Gothenburg, SE)*

Our general aim is to be able to provide a formal language to specify and analyze normative texts in general, and electronic contracts in particular. In this talk we introduce the idea of a framework where normative texts could be translated into a Controlled Natural Language (CNL) and then into a formal language, in order to be analyzed both statically and at runtime. As a step towards such an ambitious aim, we present AnaCon, a framework where a restricted version of normative texts are written in a CNL and automatically translated into the formal language CL using the Grammatical Framework (GF). In AnaCon such CL expressions are analyzed for normative conflicts (i.e., whether there are conflicting obligations, permissions and prohibitions) by the tool CLAN which gives a counter-example in case a conflict is found. We finally discuss research challenges and future directions in the area.

## 3.33 Flatland logic

*François Schwarzentruber (ENS Cachan Brittany extension – Rennes, FR)*

There are possible needs in applications such as video games for reasoning about knowledge and perception of agents. Unfortunately the behaviour of artificial agents is still nowadays often described using imperative languages such as JAVA (or script languages such as Lua). But the use of knowledge programs is a high-level option. We propose a grounded variant of Dynamic epistemic logic called Flatland logic where we can express properties about what

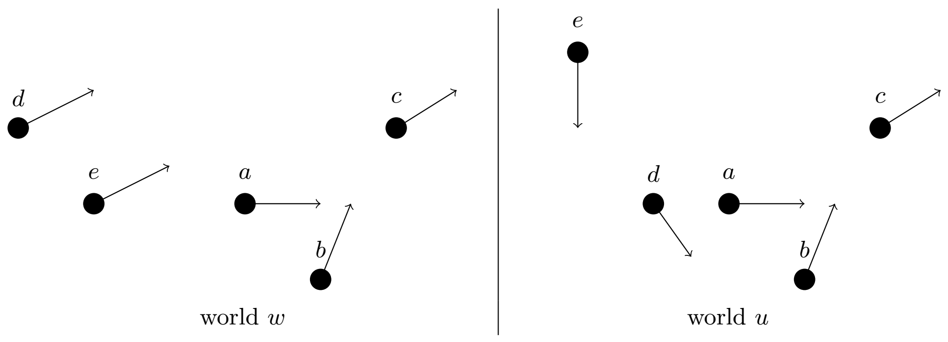


**Figure 1** Two possible 2–dimensional worlds that are indistinguishable for agent $a$.

agents perceive and know about the world. The semantics is built on top of a Kripke model. A possible world in the Kripke model is a mapping that assigns a location to each agent in the space. An agent sees a half-space. Two worlds are indistinguishable for agent $a$ if, and only if, agent $a$ sees the same thing in both worlds. Figure 1 shows two possible worlds $w$ and $u$ that are indistinguishable for agent $a$. For instance, agent $d$ sees $e$ in world $w$ and agent $d$ does not see $e$ in world $u$. Agent $a$ knows that agent $b$ sees agent $c$.

We then give results about its axiomatisation and the complexity of its model checking and satisfiability problems. In the one-dimensional case, agents are placed on a line. The one-dimensional case is axiomatized and the corresponding model checking and satisfiability problems are both PSPACE-complete. Concerning the two-dimensional case, we do not know whether there exists a finite axiomatization. We know that both the corresponding model checking and satisfiability problems are decidable but we do not know the exact complexity. There are many open issues concerning implementation and expressiveness of the logic.

## 3.34 Before the announcement

*Hans van Ditmarsch (LORIA – Nancy, FR)*

This concerns ongoing (but so far very tentative) work with Andreas Herzig and Philippe Balbiani. The well-known logic of public announcement models the effect of public events on information states: $[\varphi]\psi$ stands for 'after public announcement of $\varphi$, $\psi$ is true'. Suppose you want to go in the other direction: $[\varphi]^c\psi$ stands for 'before the announcement of $\varphi$, $\psi$ was true'. What is the logic of that? For example, $[p]^c p$ is valid: before (truthful) public announcement of $p$, $p$ must already have been considered possible. This logic is quite different from the history operators in the work of Joshua Sack, where one goes back one step in time, given history-based structures. Instead, in this 'before the announcement' logic, any structure with 'more uncertainty' than the current state can have led, after an announcement, to that current state of information. So in that sense the $[\varphi]^c$ operator has aspects of propositional quantification. Dual to the 'refinement quantifier' in 'Refinement Modal Logic' by Bozzelli et al., we seem to be looking in this case for a 'simulation quantifier': $[\varphi]^c\psi$ is true in state $s$ of model $M$ now, if $\psi$ is satisfied in any *simulation* of $(M, s)\ldots$ plus something else. Simulation quantifiers would be more proper for 'before the event' logic, the dual of event model (action model) logic. The case of 'before the announcement' is more restricted.

## 3.35 Scaling up Test Data Generation

*Ramanathan Venkatesh (Tata Consultancy Services – Pune, IN)*

**Joint work of** R Venkatesh; Anand Yeolekar; Divyesh, Unadkat
**Main reference** A. Yeolekar, D. Unadkat, V. Agarwal, S. Kumar, R. Venkatesh, "Scaling Model Checking for Test Generation using Dynamic Inference," in Proc. of the 2013 IEEE 6th Int'l Conf. on Software Testing, Verification and Validation (ICST'13), pp. 184–191, IEEE, 2013.
**URL** http://dx.doi.org/10.1109/ICST.2013.29

Structural test coverage criteria are very effective in finding bugs and also required by standards such as DO 178B. Model checkers can be used to generate test data to achieve the required coverage but model checkers unfortunately do not scale up to industry size code. To address this problem of scalability we combine dynamic analysis with model checking. We employ dynamic analysis to determine a pre-/post- condition pair for complex functions. Then we use a model checker after replacing complex functions by their pre-/post- conditions. This technique has given us much better scalability than using plain model checkers.

## 3.36 Show me your friends and I tell you who you are

*Karsten Wolf (Universität Rostock, DE)*

For an open system $S$ (e.g. a MAS), and a desired property $\varphi$, a $\varphi$-partner is another open system such that the composition $S + P$ satisfies $\varphi$. For many properties $\varphi$, we can compute a finite characterization of the (typically infinite) set of $\varphi$-partners. It can be used for several interesting applications:

- Safe exchange of S by S' is possible if partners(S) is a subset of partners(S'); we can decide that using the finite characterization
- From the set of partners, test cases may be selected; the characterization offers some notion of partner coverage
- Correction of P in a collaboration with S is possible: Select, among the partners of S, the one that is most similar to P; the finite characterization helps to reason about the infinite set of candidates.

In the talk, we sketch the work we have done and address some challenges in the MAS area, e.g. declarative representations of internal state.

## 3.37 Synthesis of Knowledge-Based Program Implementations

*Ron van der Meyden (University of New South Wales – Sydney, AU)*

Knowledge-based programs are a representation of agent behaviour, in which agent's actions are conditioned on formulas expressing properties of the agent's knowledge. This provides a useful level of abstraction that yields protocol descriptions that are independent of assumptions about the environment in which the protocol runs, and disentangles the question of what the agent needs to know in order to perform its task from the questions of how it obtains and represents that knowledge. On the other hand, knowledge-based programs are more like specifications than like executable code. To execute a knowledge-based program in a concrete environment it is necessary to determine conditions on the agent's local state that are equivalent to the knowledge properties.

The talk reviewed early work on the topic, which studied the computational complexity of finding implementations, and then addressed the question of how implementation may be found in practice. Two approaches were discussed: a partially automated approach based on epistemic model checking [3, 1, 2] and current work that aims to develop a fully automated approach based on symbolic representations of knowledge [4].

**References**

1 Omar I. Al-Bataineh and Ron van der Meyden. Epistemic model checking for knowledge-based program implementation: An application to anonymous broadcast. In Sushil Jajodia and Jianying Zhou, editors, *SecureComm*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 429–447. Springer, 2010.

2 Omar I. Al-Bataineh and Ron van der Meyden. Abstraction for epistemic model checking of dining cryptographers-based protocols. In Krzysztof R. Apt, editor, *TARK*, pages 247–256. ACM, 2011.

3 Kai Baukus and Ron van der Meyden. A knowledge based analysis of cache coherence. In Jim Davies, Wolfram Schulte, and Michael Barnett, editors, *ICFEM*, volume 3308 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2004.

4 Xiaowei Huang and Ron van der Meyden. Symbolic synthesis of knowledge-based program implementations with synchronous semantics. pages 121–130, 2013.

## 4 Working Groups

### 4.1 Case Study Description: Elevators

*Working Group 2*

Real-Life setting: Elevators in skyscrapers with decentralized control. We assume that there is a group of $n$ elevators serving $m$ floors. All lift controllers (and, hence, the strategies of all lifts) are essentially the same, except for the following configuration parameter: Not all elevators stop at all floors (the elevators can travel at different speeds). Passengers arrive at the different floors at an unpredictable rate.

#### 4.1.1 What can passengers do?

**Variant 1:** Passengers have a "call" button to call for an elevator.

**Variant 2:** Passengers have "up" and "down" buttons indicating the direction where they want to go.

**Variant 3:** Passengers have a "request" button for each floor indicating the floor they want to reach.

In Variant 1) and 2), each elevator has an internal set of buttons where passengers can indicate where they want to go.

**Variant 4:** On each floor, there is a sensor sensing how many people are waiting at that floor.

**Variant 5:** There may be a time-dependent scheme according to which elevators are being requested, e.g., before 9 am it may be mainly upwards and after 5 pm it may be mainly downwards.

**Variant 6:** Certain requests may have to be served with high priority, e.g., the president arriving at the park deck always has to find a lift waiting for him. Other high-priority calls may be for firefighter mode and emergency journeys.

### 4.1.2 What can elevators do?

Each lift knows its current position/direction and notices all passenger requests. The elevators have to communicate and negotiate which requests to serve in which order. Each elevator has a decentralized control of its own and can communicate with the others:

**Variant 1:** via broadcast,

**Variant 2:** to each other lift individually,

**Variant 3:** to "adjacent" lifts (where the adjacency relation is to be defined).

Each elevator can communicate the following information:

- its own position,
- its own direction,
- maybe also its current plan about floors to serve.

The lift may be able to predict the calls or may move to a "standard" position when idle.

### 4.1.3 Goals:

The overall goal is to serve all passengers "fairly" (task: define what that means), with a minimal number of moves (or equivalently, a minimal amount of energy) and/or a minimal waiting time. An additional requirement is that there should not be a single request not served for a certain amount of time, e.g., 10 min.

Goals for the individual elevator are:

- It could be lazy, e.g., want to move as little as possible;
- or it could be eager, trying to serve as many requests as possible;
- or optimize the travel according to some other criterion.
- Variant: There may exist malicious elevators, e.g., an elevator may be faulty or send false messages.

Each elevator should be able to reason about his decisions, e.g., there are cost and reward functions to support the argument. A cost function could be, e.g., the number and distance of moves, fast travel is more expensive than slow travel, etc. A reward could also be given for each passenger served.

## 4.2 Case Study Description: Access Control

*Dimitar P. Guelev*

### 4.2.1 Description:

An automated conference review system, for being well understood infrastructure in academia.

**Agents:** The system designer, the program chairs, the programme committee members, the authors of submissions, some external reviewers. These groups need not be all disjoint.

**Actions:** Those which take place through the automated system, subject to history-dependent permissions, include appointing PC members, submitting papers, assigning papers to review, uploading reviews, various queries, etc.

**Goals:** The *system designer* (see also Section 4.2.3 *Challenge* below) must select permission conditions which implement common express requirements on the reviewing process such as *anonymity of reviewing* and avoid recognized forms of *conflict of interest*, e.g., an author cannot interfere in the reviewing of her own submissions, and, unless a PC member, can follow only her own submissions; a PC decision normally requires a full set of reviews, etc. Overly restricted access may cause unwanted *unease*.

For (groups of) agents who are modelled *in the system*, goals include:
1. The PC want to collect the due sets of reviews and reach decisions within the promised deadlines.
2. Everyone is interested in influencing the conference programme.
3. Everyone is interested in broadening their grasp on the reviewing process as much and as soon as possible.
4. (Un)deniability.

### 4.2.2 Comments:

The scenario is partly *collaborative* and partly *competitive*. Goal 1 is collaborative, but may face shortage of external subreviewers and thus fail *independent reviewing*. Goals of group 3 are typically competitive and relate to inferred knowledge, which can jeopardize, e.g. anonymity. Goals 2 are typically in the reach of PC members, with the appointment of subreviewers in their powers. Goal 4 is primarily competitive; its collaborative aspects are by far less trivial.

### 4.2.3 Challenge:

Analyses must assist the designer in choosing, maintaining and evolving implementations of the system such that:

- each agent, possibly jointly with other agents, can achieve his or her legitimate goals by following preferably simple guidelines, and
- no schemes for reaching illegitimate goals (as understood for peer-reviewed conferences) are conceivable.

The scenario has been used previously, to illustrate research on and test algorithms for verifying access control systems in [1, 3, 2] and elsewhere, in a form that enables clear cut correspondence with the technical matter therein.

**References**

1 Dimitar P. Guelev, Mark Ryan, and Pierre-Yves Schobbens. Model-Checking Access Control Policies. In Kan Zhang and Yuliang Zheng, editors, *ISC*, volume 3225 of *Lecture Notes in Computer Science*, pages 219–230. Springer, 2004.

2 Masoud Koleini, Eike Ritter, and Mark Ryan. Model Checking Agent Knowledge in Dynamic Access Control Policies. In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 448–462. Springer, 2013.

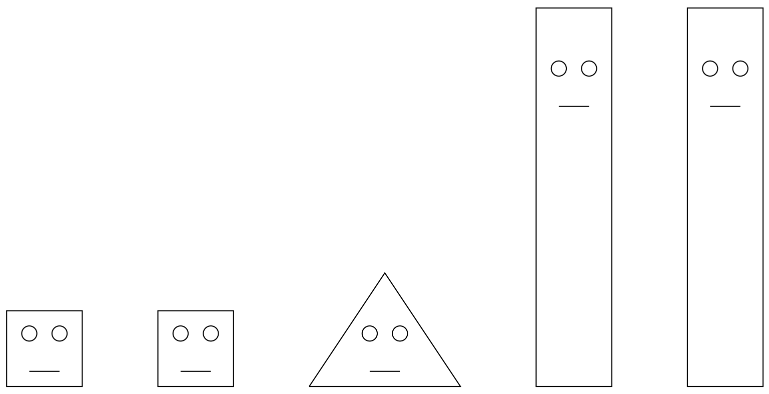
3 Nan Zhang, Mark Ryan, and Dimitar P. Guelev. Synthesising verified access control systems through model checking. *Journal of Computer Security*, 16(1):1–61, 2008.

## 4.3 Case Study Description: Earthquake Rescue Mission

*Working Group 1*

### 4.3.1 Description of the example

An earthquake occurs in a city, making many casualties, and incidentally destroying all the Internet connections. Robots are sent for a rescue mission.



Instead of sending a single type of robots that would be able to achieve all the necessary tasks (it would be too big/complicated, and some tasks may require opposite features), it seems reasonable to use robots of various kinds, with different capabilities/roles (communications, observations/detections/perception, path clearing, lifting, etc.), and also to send multiple robots of each kind, in order to parallelize the rescue mission.

Relevance for using MAS techniques:

- Dangerous environment: not safe to send humans do the job
- Poor communications: no centralized approach
- Diversity of tasks to perform
- Time pressure

### 4.3.2 Modeling as a MAS

Each robot is an autonomous agent. We describe the features of such an agent.

**Actions**

- observe

  - send/receive messages
  - move
  - lift
  - clean
  - collect

**Imperfect information** The robots may have a map in memory, but it may be inaccurate (the earthquake modifies the area) or local (limited to what they observe/learn from other robots). Sensors may be unreliable (robots may be damaged). This leads to:
  - Uncertainty concerning its own location
  - Uncertainty concerning the health of human casualties

**Goals**
  - One main goal: rescue people.
  - Subgoals, possibly dynamically chosen: look for human beings, route finding and cleaning, establish communications, agreements among a coalition of robots, inform other robots (position of a victim. . . ), find human assistants. . .

**Human interaction** Robots should be able to interact with humans they meet during the rescue (basic health diagnosis, take orders from survivors. . . ).

**Testing and verification** Difficult to define the goal (to rescue the maximum number of victims?)

## 4.4 Case Study Description: Examples from dialog/social systems

*Working Groups 3 and 4*

We explore the possibility of testing the four following application domains with MAS techniques: dialog systems, social networks, stability and resilience analysis in social systems and stock markets.

Tasks:
- Find existing benchmarks
- For each topic achieve the following.
  - Develop a concrete example
  - Give a description taking the following features into account

Uncertainty:
- more than 1 agent
- strategies involved
- autonomous control
- objectives and intentions
- communication
- information (K&B)
- human interaction

### 4.4.1 Previous Benchmarks & Competitions

What is a benchmark? A concrete scenario in which the agents compete against each other with well defined goals.

Can we borrow from Benchmarks for Security Protocols? They are so big and so real that it is hard to test the proposal. Possibility of having good enough simulations to test systems against them.

**Trading Agent Competition:** Designed to promote and encourage high quality research into the trading agent problem. Currently uses two scenarios: TAC Classic, a "travel agent" scenario based on complex procurement on multiple simultaneous auctions, and TAC SCM, a PC manufacturer scenario based on sourcing of components, manufacturing of PC's and sales to customers.
http://tac.sics.se/page.php?id=1
Comments/Critics: Use of machine learning.

**General Game Playing Description:** General Game Playing is a project of the Stanford Logic Group of Stanford University, California, which aims at creating a platform for general game playing. The games are defined by sets of rules represented in the Game Description Language. In order to play the games, players interact with a game hosting server that monitors moves for legality and keeps players informed of state changes. Since 2005, there have been annual General Game Playing competitions at the AAAI Conference. The winner of the competition is awarded with US$10,000.
http://en.wikipedia.org/wiki/General_game_playing

**Robocup Rescue Description:** The intention of the RoboCupRescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive systems in future.
http://www.robocup.org/robocup-rescue/

**Generating Instructions in Virtual Environments – GIVE:** Systems should guide a human user to navigate a virtual environment to a certain location. The systems generate natural language instructions, but they can monitor the actions taken by the user. The user cannot make queries, only navigate the environment.
http://www.give-challenge.org/research/
Comments/Critics: No real use (yet) of knowledge or beliefs. Main concern "grounding" (connected to common knowledge, but never made explicit). But systems need to include some modelling of the user. All systems include a planning component. No SAT or Model Checking involved. Reasoning is compiled in the actions taken.

General qualities of existing agents:

- High connectivity
- Low level of reasoning
- General decisions for the definition of examples.
- Qualitative vs. Quantitative
- Specify agents
- Specify actions
- Specify goals

We now describe five proposals of concrete examples for the domains of applications considered.

### 4.4.2 Property Market

Description: The general framework is that of buying and selling houses. Buyers have preferences (certain area) and limitations (amount of money).

**Agents:**
- Buyers,
- Sellers,
- Estate agents,
- Banks

**Goals:**
- Sellers want to get higher price
- Buyers have a private utility
- Banks want to give a loan to the most reliable buyer

**Actions:**
- Announce a price
- Bidding for a house
- Securing a loan (for a Buyer)
- Give a loan (for a Bank)
- Negate a loan (for a Bank)

**Type:**
- Qualitative Information: Give the largest possible loan to prospective worthy agents.
- Quantitative Information: Location, price

**Prediction:**
- Creation of a bubble
- Equilibria Happiness

**Comments:**
- No clear separation between buyers and sellers (a buyer might be a seller)
- State agent misinformation
- Intervention: the state intervenes to fix problems (change payoff, include taxes). They can fix the problem or make it harder.

### 4.4.3 Concert Going

Description: A concert is being announced in a given location. People can announce whether they expect to go or not (e.g., via their facebook pages). Other agents (e.g., hotel and restaurants in the area of the concert) have access to the information and can use it for their planning. (e.g., do I go or not in the end?, how much food do I order, how much do I charge for parking?).

**Agents:**
- Concert goers
- Restaurants
- Hotels

**Goals:**
- Restaurants do not run out of food
- Concert goer: I finally go only if some other people actually go.

**Actions:**
- Announce going to the concert
- Check

**Comments:**
- Quite Complex
- Multi-Agent planning problem
- Friends provide network: the network is not accessible to everybody

Perspective of the city hall: Cooperatively make the system reach some goal.

### 4.4.4 Stock Market

Description: Only one stock

**Agents:** Finite number of agents not determined.
**Actions:**
- Sell
- Buy

**Aim:** Have a system that prevents certain problems from occurring. Intervention. Relation to Thomas Ågotnes' ideas on social laws.
**Comments:** Mainly qualitative, no probabilities. But includes quantities (price of the stock).

### 4.4.5 Book Prices

Description: book buying/selling, with the constraint (legal obligation) that the prices must remain below a certain threshold. Assumption: unlimited supply of books.

**Agents:**
- Book sellers
- Book buyers

**Goal:** Sell the books for the best possible price
**Actions:**
- Buy a book
- Raise the price of the book
- Look at the price of a book

**Beliefs:** They don't have access to the actual demand
**Comment:** A step towards the complex one.

### 4.4.6 Really asymmetric example

Description: The set up of the GIVE challenge.

**Agents:**
- Human following instructions
- System giving instructions

**Actions:**
- Navigation actions (for the User)
- Instruction generation (for the System)
- User Monitoring (for the System)

## 4.5 Case Study Description: Avionic scenario

*Franco Raimondi*

### 4.5.1 Description of scenario and Modeling as MAS

An aeroplane is composed of a number of agents, such as one or more (human) pilots, autopilot, Traffic Collision Avoidance System (TCAS), etc. Also, a number of sensors are present, and each agent has access to a subset of these. Sensors may be noisy and could fail. A number of actuators are present as well, aeroplanes are coordinated by Air Traffic Controllers (ATC) on ground. ATCs have access to sensors, actuators, and other agents (such as the ground equivalent of TCAS). Figure 2 represents the aeroplane diagram, and Figure 3 represents the ATC diagram (100 means "a good amount of", and 1000 means "a lot").

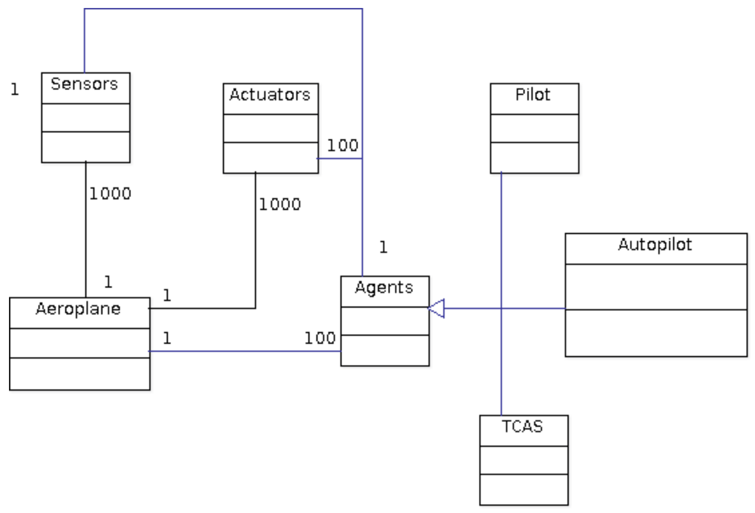


**Figure 2** Diagram of an aeroplane.

### 4.5.2 State of the art

Some tools are available to model similar scenarios. See

1) J. Hunter, F. Raimondi, N. Rungta, R. Stocker, A Synergistic and Extensible Framework for Multi-Agent System Verification, to appear in Proceedings of AAMAS 2013

2) N. Rungta, G. Brat, W. Clancey, C. Linde, F. Raimondi, Chin S. and M. Shafto, Aviation Safety: Modeling and Analyzing Complex Interactions between Humans and Automated Systems, in Proceedings of ATACCS 2013

Papers available at http://www.rmnd.net/publications/

These papers use Brahms, see http://www.dagstuhl.de/Materials/Files/07/07122/07122.SierhuisMaarten.Other.pdf

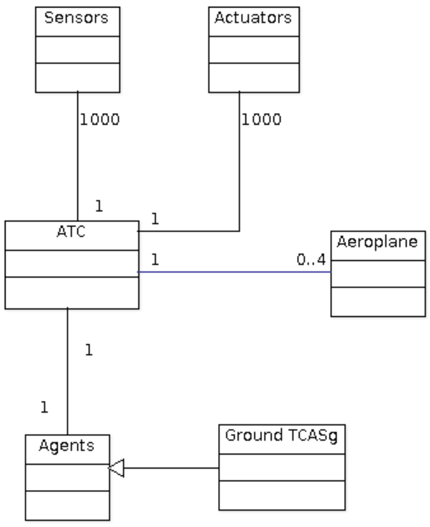


**Figure 3** Diagram of an ATC.

### 4.5.3 Relevant requirements for verification

There are a number of possibilities. Check that safety rules are respected even in presence of noisy sensors:

- Autopilot needs to be engaged while landing in fog, and pilot has a strategy to achieve this
- What is the probability of reaching a certain state?
- If the autopilot is not engaged, the pilot knows it
- If the pilot believes that a sensor is not functioning, then the pilot has a strategy to deploy a backup sensor, etc.

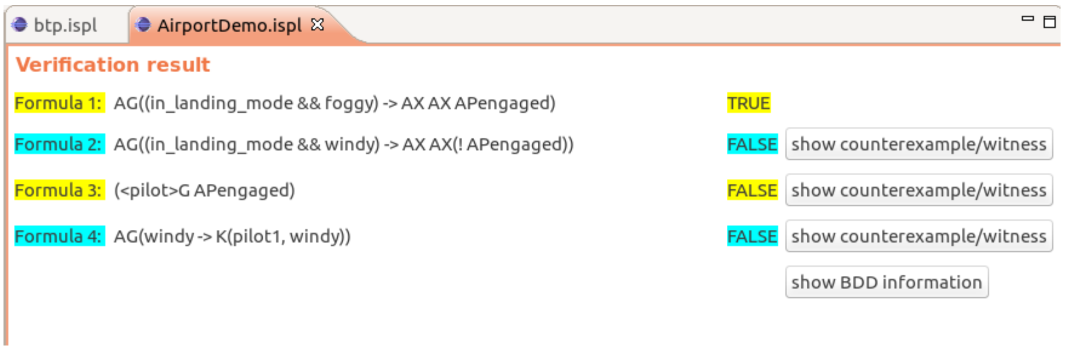
Other possibilities include the encoding of interesting properties from existing procedures or from proposed standards (see NextGen and SESAR, available respectively at http://en.wikipedia.org/wiki/Next_Generation_Air_Transportation_System and http://en.wikipedia.org/wiki/Single_European_Sky_ATM_Research for examples of Air Traffic Management systems).

### 4.5.4 Existing tools

The source code of a very simple example using MCMAS (supports CTLK+ATL) is provided below. The scenario encodes take-off and landing of an aeroplane in random weather conditions (fog, wind, clear). If the weather is foggy, the autopilot should be engaged while landing, but it should be disengaged in case of wind. It is possible to check these properties and an epistemic one such as "if it is windy, the pilot knows it" (false, because a noisy wind sensor is modelled). Also, it is possible to check ATL properties such as "the pilot has a strategy to keep the autopilot on" (false because the aeroplane could be stuck at gate). Figure 4 shows some examples of formulas model checked on the scenario example using MCMAS. When a formula is not true on a model, MCMAS can give an example of behaviour that does not verify it (see Figure 5).

```
Agent Environment
    Vars:
        aeroplane1_status : {atgate, taxiing, takingoff, climbing,
            enroute, descending, landing};
        weather_conditions : { clear, wind, fog};
    end Vars
    Actions = {wind,clear,fog,none};
    Protocol:
        weather_conditions=clear: {clear};
        weather_conditions=wind : {wind};
        weather_conditions=fog : {fog};
        Other : {none};
    end Protocol
    Evolution:
        weather_conditions = clear if Action=clear;
        weather_conditions = wind if Action=wind;
        weather_conditions = fog if Action=fog;
        aeroplane1_status=taxiing if aeroplane1_status=atgate;
        aeroplane1_status=takingoff if aeroplane1_status=taxiing;
        aeroplane1_status=climbing if aeroplane1_status=takingoff;
        aeroplane1_status=enroute if aeroplane1_status=climbing;
        aeroplane1_status=descending if aeroplane1_status=enroute;
        aeroplane1_status=landing if aeroplane1_status=descending;
    end Evolution
end Agent

Agent pilot1
    Lobsvars = {aeroplane1_status};
    Vars:
        perceived_weather : {clear,wind,fog};
    end Vars

    Actions = {pushback,takeoff,engageAP,disengageAP,none};

    Protocol:
        perceived_weather=clear and !(Environment.aeroplane1_status=taxiing or
                                Environment.aeroplane1_status=atgate): {engageAP,disengageAP};
        perceived_weather=wind and !(Environment.aeroplane1_status=taxiing or
                               Environment.aeroplane1_status=atgate) : {disengageAP};
        perceived_weather=fog and !(Environment.aeroplane1_status=taxiing or
                              Environment.aeroplane1_status=atgate) :  {engageAP};
        Other : {none};
    end Protocol
    Evolution:
        perceived_weather=clear if (Environment.Action=clear or Environment.Action=wind);
        perceived_weather=fog if (Environment.Action=fog);
        perceived_weather=wind if (Environment.Action=wind);
    end Evolution
end Agent

Agent autopilot1
    Lobsvars = {aeroplane1_status};
    Vars:
        engaged : boolean;
    end Vars

    Actions = {none};
    Protocol:
        Other : {none};
    end Protocol
    Evolution:
        engaged = true if pilot1.Action=engageAP;
        engaged = false if pilot1.Action=disengageAP;
    end Evolution
end Agent
```

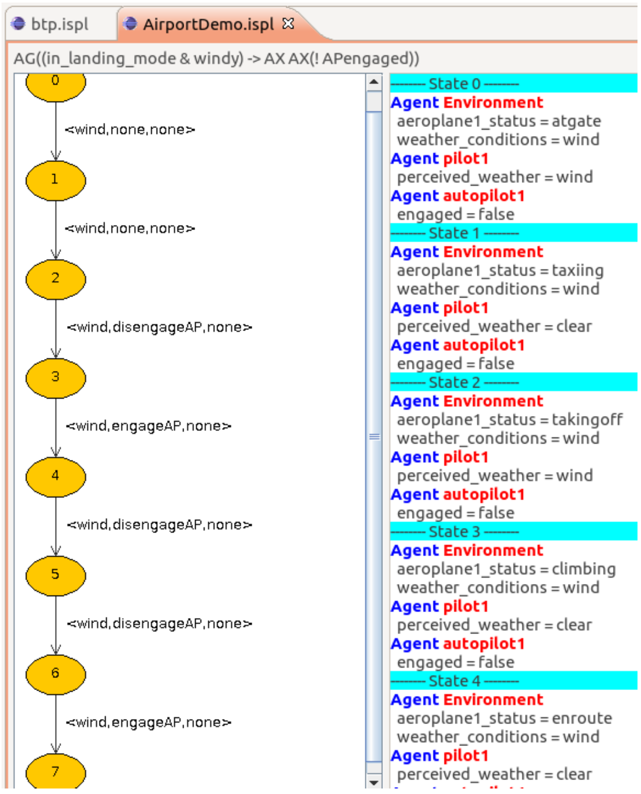
```
Evaluation
    windy if Environment.weather_conditions=wind;
    foggy if Environment.weather_conditions=fog;
    APengaged if autopilot1.engaged = true;
    in_landing_mode if (Environment.aeroplane1_status=descending) or
                       (Environment.aeroplane1_status=landing);
end Evaluation

InitStates
    (Environment.aeroplane1_status=atgate) and autopilot1.engaged=false;
end InitStates

Groups
    pilot = {pilot1};
end Groups

Fairness
    -- in_landing_mode;
end Fairness

Formulae
    AG((in_landing_mode and foggy) -> AX(AX(APengaged)));
    AG((in_landing_mode and windy) -> AX(AX(!APengaged)));
    <pilot>G(APengaged);
    AG(windy -> K(pilot1,windy));

end Formulae
```



Verification result

| | | | |
|---|---|---|---|
| Formula 1: | AG((in_landing_mode && foggy) -> AX AX APengaged) | TRUE | |
| Formula 2: | AG((in_landing_mode && windy) -> AX AX(! APengaged)) | FALSE | show counterexample/witness |
| Formula 3: | (<pilot>G APengaged) | FALSE | show counterexample/witness |
| Formula 4: | AG(windy -> K(pilot1, windy)) | FALSE | show counterexample/witness |
| | | | show BDD information |

**Figure 4** Model checking some formulas on the example scenario.



AG((in_landing_mode & windy) -> AX AX(! APengaged))

```
0
  <wind,none,none>
1
  <wind,none,none>
2
  <wind,disengageAP,none>
3
  <wind,engageAP,none>
4
  <wind,disengageAP,none>
5
  <wind,disengageAP,none>
6
  <wind,engageAP,none>
7
```

```
------- State 0 -------
Agent Environment
 aeroplane1_status = atgate
 weather_conditions = wind
Agent pilot1
 perceived_weather = wind
Agent autopilot1
 engaged = false
------- State 1 -------
Agent Environment
 aeroplane1_status = taxiing
 weather_conditions = wind
Agent pilot1
 perceived_weather = clear
Agent autopilot1
 engaged = false
------- State 2 -------
Agent Environment
 aeroplane1_status = takingoff
 weather_conditions = wind
Agent pilot1
 perceived_weather = wind
Agent autopilot1
 engaged = false
------- State 3 -------
Agent Environment
 aeroplane1_status = climbing
 weather_conditions = wind
Agent pilot1
 perceived_weather = clear
Agent autopilot1
 engaged = false
------- State 4 -------
Agent Environment
 aeroplane1_status = enroute
 weather_conditions = wind
Agent pilot1
 perceived_weather = clear
```

**Figure 5** Counterexample for Formula 2.

## 4.6 Discussion about the testing of MAS

*Working Group 7*

**Main questions:**
- Commonalities between testing and model checking for MAS ⇒ not discussed
- Do we need special techniques for testing autonomous system?
- What are the testing goals for MAS?

**General considerations:**
- Output not deterministically predictable: no "right" or "wrong" of individual decisions.
- Goals and reasoning must be included in the test harness.
- Agents must be able to explain and give rationale for decisions (i.e., no black-box testing is possible)
- Difference to verification? ⇒ considers only a selected number of runs
- Test oracle problem becomes harder (maybe even undecidable)

**Completeness of testing? How to do the testing?**
- Test case problem: find a scenario with a well-defined reaction philosophical issue: "highly intelligent" and "defective" cannot be easily distinguished.
- Often "just" random experimentation is done
- Make (random?) experiment and check whether the behaviour can be explained by the overall goals
- Differentiate between high-level goals (on the level of agent groups) and low-level goals (on the level of individual agents)
- Low-level goals are probably easier to check or verify
- Coverage criteria are hard to define.
- Maybe a new coverage criterion. "Strategy coverage" must be defined. Meaning: cover all strategies or cover all strategic decisions? This is different from path coverage (all path with all strategies).

**How to deal with the change of strategies?**
- Is a meta-strategy just another strategy?
- Example: cellular-automata game with cats, birds and dogs [(C): Stefan Gruner and his students in Pretoria]
- Can learn strategies
- Evolving strategies are a challenge for MAS
- Testing knowledge and belief versus testing strategies?
  ⇒ Probably knowledge and belief are already represented in the strategy.
- Is there a theory of strategy testing?
- Apparently not (yet); e.g. ioco cannot be transferred. Maybe there is a connection to model checking here: evaluation of the quality of testing just by statistical evaluation ⇒ Simulation environments with random testing coverage of environment vs. coverage of SUT.
- We need a test strategy for testing strategies :-)
- We need suitable criteria for this purpose. One such criterion could be "Robustness": if the environment changes only little, the agent's behaviour should typically also change only little.

**What about learning systems?**
- MAS vs. von Neumann's cellular automata

- Programs writing programs?
- Distinguishing a highly intelligent and a defect system can be hard
- Machine learning on two levels:
  - parameter optimization
  - strategy learning
- Learning systems are out of scope at the moment; how about adaptive systems?

OUTLOOK: For the meta theory (Methodology) of testing, can we learn anything from the social sciences or from pedagogics? How do sociologists or school-teachers do their "experiments" with people or pupils? Scientificness of the Methodology must be kept in mind (e.g. Mario Bunge)

## 4.7 Discussion concerning logics for knowledge, time and strategies

*Working Group 5*

| **Logics** | **Extensions** | **Limitations** | **Open questions / Possible solutions** |
|---|---|---|---|
| Epistemic Logic | Temporal<br>DEL<br>Memory<br>Synchronous/Asynchronous | Unrealistic w.r.t resources<br>Models are about uncertainty<br>$\rightarrow$ Awareness?<br>Knowledge de dicto/de re | Alternative semantics |
| Strategic Logics<br>ATL | Imperfect information<br>Recall<br>Type of strategy<br>Explicit strategies/actions<br>Quantitative aspects | Complexity/Undecidability<br>Where are the<br>"killer" applications?<br>ATL + dynamics<br>Mixed strategies/<br>probabilistic settings | Security protocols<br>e-voting<br>Plant controller<br>Modelling (translating<br>from semi-formal<br>to formal)<br>More tools<br>Encoding real problems<br>Connection between<br>ATL and game theory |
| DEL | Connection to<br>LTL + knowledge | Perfect recall unavoidable<br>Asynchronous semantics<br>for DEL | DEL with imperfect<br>recall<br>DEL for planning |

## Participants

- Thomas Ågotnes
University of Bergen, NO
- Carlos Areces
Universidad Nacional de Córdoba, AR
- Guillaume Aucher
INRIA Bretagne Atlantique – Rennes, FR
- Alexandru Baltag
University of Amsterdam, NL
- Ezio Bartocci
TU Wien, AT
- Ioana Boureanu
EPFL – Lausanne, CH
- Nils Bulling
TU Clausthal, DE
- Louise A. Dennis
University of Liverpool, GB
- Michael Fisher
University of Liverpool, GB
- Tim French
The University of Western Australia – Nedlands, AU
- Valentin Goranko
Technical Univ. of Denmark, DK
- Stefan Gruner
University of Pretoria, ZA
- Dimitar Guelev
Bulgarian Academy of Sciences – Sofia, BG
- Yuri Gurevich
Microsoft Res. – Redmond, US
- Andreas Herzig
Paul Sabatier University – Toulouse, FR
- Wojtek Jamroga
University of Luxembourg, LU
- François Laroussinie
University Paris-Diderot, FR
- Alessio R. Lomuscio
Imperial College London, GB
- Nicolas Markey
ENS – Cachan, FR
- Bastien Maubert
INRIA Bretagne Atlantique – Rennes, FR
- Stephan Merz
LORIA – Nancy, FR
- Aniello Murano
University of Napoli, IT
- Wojciech Penczek
Siedlce University of Natural Sciences and Humanities, PL
- Sylvain Peyronnet
GREYC – Caen, FR
- Jerzy Pilecki
Polish Academy of Science, PL
- Sophie Pinchinat
IRISA – Rennes, FR
- Franco Raimondi
Middlesex University, GB
- Jean-François Raskin
Université Libre de Bruxelles, BE
- Markus Roggenbach
Swansea University, GB
- Ina Schaefer
TU Braunschweig, DE
- Holger Schlingloff
HU Berlin, DE
- Gerardo Schneider
University of Gothenburg, SE
- Henning Schnoor
Universität Kiel, DE
- François Schwarzentruber
IRISA – Rennes, FR
- Dmitry Shkatov
University of the Witwatersrand – Johannesburg, ZA
- Ron van der Meyden
UNSW – Sydney, AU
- Hans Van Ditmarsch
LORIA – Nancy, FR
- Ramanathan Venkatesh
Tata Consultancy Services – Pune, IN
- Karsten Wolf
Universität Rostock, DE

Report from Dagstuhl Seminar 13182

# Meta-Modeling Model-Based Engineering Tools

**Edited by**

**Tony Clark[1], Robert B. France[2], Martin Gogolla[3], and Bran V. Selic[4]**

1 **Middlesex University, GB,** t.n.clark@mdx.ac.uk
2 **Colorado State University, US,** france@cs.colostate.edu
3 **Universität Bremen, DE,** gogolla@informatik.uni-bremen.de
4 **Malina Software Corp. – Nepean, CA,** selic@acm.org

## Abstract

Model-based engineering (MBE) is a software development approach in which abstraction via modeling is used as the primary mechanism for managing the complexity of software-based systems. An effective approach to software development must be supported by effective technologies (i.e., languages, methods, processes, tools). The wide range of development tasks that effective MBE approaches must support leads to two possible tooling scenarios. In the first scenario a federated collection of tools is used to support system development. Each tool in the collection provides specialized services. Tool interoperability and consistency of information across the tools are major concerns in this scenario. These concerns are typically addressed using transformations and exposed tool interfaces. Defining and evolving the transformations and interfaces requires detailed low-level knowledge of the tools and thus leads to complex tooling environments that are difficult to configure, learn, use, and evolve. In the second scenario, a single tool is used to support the complete modeling lifecycle. This avoids the inter-tool transformation and consistency problems, but the resulting multi-featured tool is a monolithic entity that is costly to develop and evolve. Furthermore, the large number of non-trivial features can make learning and using such tools difficult.

Successful uptake of MDE in industry requires supporting tools to be, at least, useful and usable. From a tool developer's perspective, there is also a need to significantly reduce the cost and effort required to develop and evolve complex MBE tools. This seminar brings together experts in the areas of MBE, meta-modeling, tool development, and human-computer interactions to map out a research agenda that lays a foundation for the development of effective MBE tools. Such a foundation will need to support not only interoperability of tools or tool features, but also the implementation of high quality MBE tools. The long-term objective is to foster a research community that will work on a foundation that can be expressed in the form of standard tool (meta-)models that capture and leverage high quality reusable MBE tool development experience.

**Seminar** 28. April–03. May, 2013 – www.dagstuhl.de/13182
**1998 ACM Subject Classification** D.2 Software Engineering, H.1 Models and Principles
**Keywords and phrases** meta-modeling, model-based engineering, models, tools, domain specific modeling languages
**Digital Object Identifier** 10.4230/DagRep.3.4.188
**Edited in cooperation with** Dustin Wüest

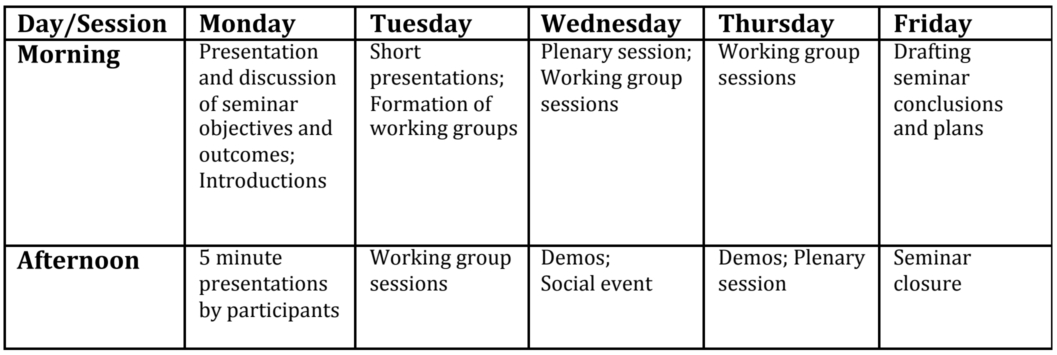
## 1 Executive Summary

*Tony Clark*
*Robert B. France*
*Martin Gogolla*
*Bran V. Selic*

The 33 participants at the Meta-Modeling Model-Based Engineering Tools (M$^3$BET) Dagstuhl Seminar were brought together to explore how model-based engineering (MBE) techniques can be used to improve the quality of software modeling tools. The participants included expert researchers, practitioners and tool developers in the software/system modeling and the human computer interaction communities. The discussions aimed to answer the following question: Can MBE techniques be used to produce more effective MBE tools, and, if so, how should it be done?

The vision underlying the seminar is one in which technologists create tool models that specify desired tool features, and tool modeling frameworks that are used to analyze, compose, transform, simulate and otherwise manipulate the tool models. In the vision, tool developers will use tool models and frameworks to produce useful, usable and cost-effective software modeling tools.

### Seminar Organization



| Day/Session | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| Morning | Presentation and discussion of seminar objectives and outcomes; Introductions | Short presentations; Formation of working groups | Plenary session; Working group sessions | Working group sessions | Drafting seminar conclusions and plans |
| Afternoon | 5 minute presentations by participants | Working group sessions | Demos; Social event | Demos; Plenary session | Seminar closure |

**Figure 1** The Final Seminar Program.

The final seminar program is given in Figure 1. On the first day the seminar objective and outcomes were presented. The seminar objectives, as presented on that day, was to better understand the "what, why, and how" of tool models, and initiate work on (1) languages for tool modeling, (2) MBE methods and technologies for tool development, and (3) tool modeling frameworks.

The planned outcomes were (1) reports on the results of group discussions, (2) a research roadmap for achieving the tool modeling vision, (3) potential solutions for achieving the vision, and (4) initiation of new research collaborations among participants.

To help shape and initiate the discussions, the organizers proposed the following as an initial set of breakout group topics:

**Tool capabilities** – The intent was that discussions in this group would focus on identifying the software tool capabilities that should be captured in tool models, and on how these capabilities could be captured in tool metamodels. This covers discussions on (1) how metamodels can be used to describe tool capabilities in a manner that supports generation of high-quality tool components, (2) the utility and feasibility of defining tool metamodels, (3) potential benefits associated with and purposes served by a tool metamodel, and (4) challenges associated with developing an effective metamodel (i.e., a metamodel that is fit-for-purpose).

**Tool qualities** – Discussions in this group would aim to answer questions about desirable tool qualities (e.g., What issues determine tool adoption and why?). This includes key questions related to, for example, usability/human factors, scalability, interoperability, as well as non-technical but important considerations related to organization goals, culture, and processes.

**Tool ecosystems** – A tool framework can be thought of as a technological ecosystem that involves both tools as well as tool users. Discussions in this group would seek answers to questions such as: What are the features of a good tools framework? Are there candidate frameworks available? If so, are they sufficient or do they need to be extended?

**Tool development methods** – Discussions in this group would focus on answering the following questions: How can MBE be applied to the development of MBE tools? What types of languages are suitable for describing tools? How can tool quality issues be addressed by such methods?

## Working Groups

During the discussions on group topics it was decided to base the groups on tool concerns and issues that the participants had some stake in. It turned out that the concerns and issues that arose from the discussions were mostly derived from those underlying the groups proposed by the organizers.

The concerns and issues were then clustered into two groups based on participant interests. Group A consisted of usability, utility, and broader non-technical concerns (e.g., designing tools that support how developers work and think, designing and performing usability studies, adapting tool features to user expertise and desired level of formality, marketing/business/cultural concerns). Group B consisted of the more technical concerns, for example, concerns related to tool development methods, scalability, support for separation of concerns, tool quality assessment and benchmarking.

Group B concerns were further grouped into two categories: Composition, and Methods and Quality concerns. The Composition concerns included issues related to tool, language, and model composition, and the use of multi-models with heterogeneous semantics.

Three Working Groups, each focusing on one of the above concern groups, were formed on the seminar's first day. A summary of the discussions that took place in each group is included in the report.

## Summary and Future Work

One of the major insights gained during the seminar was that a good understanding of the utility of tool models and the identification of appropriate forms of tool models/metamodels requires one to first address more fundamental tool development and assessment concerns. On hindsight, this should not have been a surprising result; effective tool models would have to capture significant tool development and assessment experience and knowledge and thus such experience and knowledge needs to be distilled first. The seminar provided a good forum for discussing and organizing the experience and knowledge of the participants.

Post-seminar collaborations that will utilize these results to develop an initial set of tool models/metamodels were initiated at the seminar.

In addition to the planned collaborations on tool models, the participants also agreed to engage in the following post-seminar activities:

- Publications: The following publications are planned
  - A special issue of the Software and System Modeling (SoSyM) journal that will include articles that focus on the concerns and issues discussed at the seminar.
  - A paper that discusses problems associated with current software modeling tools.
- Workshops: Workshops in which participants will discuss and develop tool models/metamodels will be held at conferences such as MODELS 2014 and ICSE 2014.

## 2 Table of Contents

## 3 Overview of Talks

### 3.1 Towards Truly View-Based Software Engineering Environments

*Colin Atkinson (Universität Mannheim, DE)*

**Main reference** C. Atkinson, D. Stoll, C. Tunjic, and J. Robin, "A Prototype Implementation of an Orthographic Software Modeling Environment", VAO 2013, Montpellier 2013.
**URL** http://swt.informatik.uni-mannheim.de

Ensuring the coherence of all the data, artifacts and relationships generated during a software development project has always been an important aspect of software engineering, even in the days when software systems were typically written in a single language, but with modern development languages, paradigms and processes maintaining artifact coherence has arguably become one of the biggest challenges in software engineering. Without a continual campaign of "management" activities (e.g. configuration management, change management, version management, release management, traceability management etc.) the coherence of software engineering artifacts inevitably degrades over time. At the concrete level of individual projects this coherence challenge is often experienced as an integration problem – more specifically, as the problem of integrating data, tools, and processes within a single environment. The software industry therefore has tended to tackle this challenge with "integration" or "interoperability" solutions (e.g. the Open Services for Lifecycle Collaboration (OSLC)). While such initiatives undoubtedly alleviate some of the immediate problems faced in using traditional tool suites, they do not really address the underlying problems. In particular, the number of pairwise consistency relationships that need to be maintained grows rapidly with the number of tool/views involved (n(n-1)/2) where n is the number of tools).

This talk takes the position that in order to address the root causes of the aforementioned problems, the next generation of software engineering environments and method should be based on the following three principles. The first principle is that the information edited and portrayed within the "artifacts" should be regarded as "views" derived from a single centralized description of the system a "Single Underling Model (SUM)". The most important consequence of this shift in perspective from artifact/tool centric handling of information to view/SUM centric handling of information is that no lifecycle management activities must be carried out at the artifact level (as is typically the case today). In other words, there must be no versions/variants of artifacts (e.g. code modules, PIMs etc), there must only be versions/variants of the system (and its components) as described in the SUM.

The second principle is that the best modeling architecture to realize such an approach is a multi-level-model approach that allows languages and services to be applied uniformly across multiple classification levels. Such an architecture is often referred to as an Orthogonal Classification Architecture (OCA). It is important to stress that the term "model" here is intended in a more general sense than the typical types of diagram used in model-driven development (i.e. class diagram or state diagrams etc.). In this context the term "model" refers to any form of information representation including all traditional software engineering artifacts such code and models of the kind commonly used in model-driven development.

The final principle is that a software engineering environment based on such an infrastructure should be "view" agnostic in the sense that it should be based on a method or development "paradigm" that is not driven by, or biased towards, any particular kind of view. In particular, this means that no particular kind of view such as "code" or "PIMs" should dominate the development process. This will make it possible to merge today's advanced software engineering paradigms such as model- driven development, aspect-oriented development, product-line engineering and agile development in a much more fundamental way than has so far been possible.

## 3.2 Meta-Models for Model Based Engineering Tools

*Tony Clark (Middlesex University, GB)*

Model Based Engineering claims to address issues including efficiency and quality of the Software Engineering process through the use of abstraction in terms of models. Models represent an aspect of a system and allow the engineer to design, analyze and manipulate the system aspect without needing to deal with implementation details that might otherwise impede progress.

Successful Model Based Engineering relies on tools that construct, analyze and transform models. There are many such tools, some free and come commercial, that are currently available. Unfortunately, the usability of such tools is generally viewed to be less than ideal, and has been claimed to be part of the reason why MBE has not become more widespread within the SE industry.

The lack of MBE tool usability may be attributed to various factors. These include the sheer size and complexity of typical MBE tool functionality where menus proliferate and reach many levels of nesting. It is often the case that a tool tries to support many (in some cases all) the actors in the software development life-cycle leading to tools that, for each individual user, offer a vast array of redundant features. Furthermore, such tools are often developed by software engineers who are not experts in design issues, leading to awkward or unintuitive interactions.

A proposal to address this problem is to apply MBE principles to the development of MBE tools. A meta-language for tool development would allow features such as design principles to be expressed and captured for families of tools. The resulting tool models could be processed by tool engines that make the development of smaller tools more economically attractive. MBE techniques such as slicing, transformation and merging could be applied to tool models thereby allowing tool fragments to be reused so that building larger tools is achievable by smaller groups, and can benefit from earlier tool verification efforts.

Over time, it may be possible to produce a standard model for expressing MBE tools leading to a marketplace in terms of meta-frameworks that process tool-models and in terms of tool fragments. Companies can have proprietary implementations of a meta-framework that produce a product-line of MBE tools tailored for a specific domain.

## 3.3 MDE and SLE Cross Fertilization

*Benoit Combemale (IRISA – Rennes, FR)*

Domain Specific Modeling Languages (DSMLs) are widely adopted to capitalize on business domain experiences. Consequently, DSML implementation is becoming a recurring activity, whether newly defined language or by specialization of a more general language (e.g., UML). To be effective, a DSML requires an associate dedicated modeling environment including tools such as editors, simulators, and code generators. In practice, the implementation of such a modeling environment is a complex and time consuming task. However, it is commonly realized from scratch for each DSML, even for languages which share some concepts and could share bits of tool support. A major obstacle is that there is no opportunity to capitalize the work, even if the different modeling languages and their respective tools are very closed or should be coordinated. In this talk, we explore the challenges associated to the cross-fertilization of Software Language Engineering, Global Software Engineering and Model Driven Engineering to support a Coordinated Model-Driven Language engineering. In particular, we focus on a uniform approach for language reuse, variability, composability and coordination.

## 3.4 Human Activity Modeling of Human Modeling Activities

*Larry Constantine (University of Madeira – Funchal, PT)*

Human Activity Modeling (hAM) is a systematic formalism based on activity theory. Activity theory is a well-established framework for describing and understanding human activities of all kinds. The systematic notation of hAM provides simplified conventions that support user experience and interaction design and builds a shared vocabulary that bridges to software engineering. Recent work extends the range of applications to include social sciences research and organization analysis. In this presentation, hAM is applied recursively to the activities that analysts, designers, and engineers carry out when using models.

## 3.5 Abstract Modeling for Interaction Design

*Larry Constantine (University of Madeira – Funchal, PT)*

Recent work has extended Canonical Abstract Prototypes (CAPs) to support the design of large, complex multi-modal, multi-media, multi-channel systems and services. The resulting dynamic, distributed CAPs (ddCAPs) accomplish a separation of concerns that facilitates fuller specification of dynamic and behavioral aspects of user interfaces at an abstract level. The notation of ddCAP models provides a simpler, more effective platform for communication between interaction designers and software engineers.

## 3.6 Lessons from Human Tool Use

*Larry Constantine (University of Madeira – Funchal, PT)*

Findings from ethnographic and social science investigations of tool use are reviewed for concrete guidelines for design and construction of better modeling tools. Informed by activity theory inquiries into how humans organize and use tools in complex, skilled activities, a dozen specific recommendations are presented for enhancing user performance in modeling and use of modeling tools.

## 3.7 Towards More Reliable Tools

*Catherine Dubois (ENSIIE – Evry, FR)*

Tools are fundamental when using formal methods and when applying a model based engineering approach. And they are many MBE tools. However a question arises: do you trust your tools ? The position I try to defend is that we can adopt a formal approach to verify and/or develop the MBE tools. In particular a formalized semantics of a modeling language is necessary to build tools in order to provide them with a basis as solid as possible. Several researchers have tackled this problem, using Coq, Isabelle or Maude, either to provide sound mathematical foundations for the study and the validation of MDE technologies or to verify some model transformations. However it is a challenging task (and hard work) to develop such formalizations in proof assistants like Coq or Isabelle. We can notice that modeling languages often share some notions or components. Consequently a research perspective could be to develop a framework allowing to define the semantics of these common component together with a way of combining them in order to define formally the mechanized semantics of a modeling language and thus verify tools and model transformations. DSL, aspects, variability etc. are, according to me, ingredients of such a framework.

## 3.8 Megamodels in Model-Based Engineering Tools

*Michalis Famelis (University of Toronto, CA)*

I briefly introduced the ideas of my research group regarding the use of typed megamodeling to improve model management in model-based engineering tools. Through the explicit management of models, their metamodels, and relations and transformations between them, we aim to create support for ecosystems of modular, reusable tools. Typed megamodels can be used at tool runtime for online tool extension and reconfiguration, while providing semantic consistency for tasks such as language extension and adaptation. Tooling tasks can also be better supported by leveraging the megamodel type system, by taking into account type casting, polymorphism, type safety, etc. I illustrated these ideas using the Model

Management Tool Framework (MMTF), an Eclipse-based pluggable infrastructure developed at the University of Toronto. Megamodels can also be used to support collaboration and knowledge transfer among multiple users of tools. I illustrated this using the example of Modelepedia, a semantic-wiki companion to MMTF.

## 3.9 Demo Abstract: Typed Megamodeling with the Model Management Tool Framework

*Michalis Famelis (University of Toronto, CA)*

Model management has emerged as an important approach to addressing the complexity introduced by the use of many interrelated models in Software Engineering. In this tool demonstration, I highlighted the capabilities of the Model Management Tool Framework (MMTF) – an Eclipse-based pluggable infrastructure for rapidly developing model management tools. MMTF differs from other model management frameworks through its emphasis on interactive model management, graphical megamodeling and strong typing. Using examples from automotive software development, I illustrated how MMTF can be used as a runtime typed megamodel to enhance the capabilities of modeling tools.

## 3.10 Towards Empathetic Tool Design

*Robert B. France (Colorado State University, US)*

There is no doubt that research in model driven (software) development (MDD) has advanced the state-of-art in software modeling and yielded revealing insights into the challenging problems that developers of complex software-based systems face. Yet, sustainable use of MDD technologies in industry is still elusive. While many practitioners have some appreciation of the benefits associated with proper use of MDD approaches, technologists (tool architects and developers) have fallen short of providing the types of tools needed to fully support MDD practices. Specifically, a pesky point of friction is the limited number of usable MDD tools that are available today.

Practitioners understandably judge the effectiveness of MDD approaches in terms of tool quality: If the tool is bad then the approach it supports is perceived as being bad. While this perception may be logically flawed, it is understandable; tools are at the front-end of the battle to manage software complexity through MDD approaches, and thus their failure to support MDD practices and the work styles of MDD practitioners has a significant impact on how well MDD practices can be supported in a sustainable manner. In other words, tools should make the power of MDD readily accessible to developers.

Poor tool support can contribute to accidental complexity. This gives rise to the perception that MDD adds complexity to the software development process. In many cases, tools are too heavyweight, that is, they have many features that support a broad spectrum of development activities, in addition to a complex infrastructure for managing the features and for extending the tool with new features. The problem with such broad-spectrum tools is that it can take considerable effort to, for example, (1) identify and get to the subset of features needed to support a particular activity, (2) learn how to effectively use a desired feature, and (3) learn how to use a combination of features to support development workflows that cut across different development phases (e.g., to support coordinated use of models at the requirements, architecture, and detailed design stages). This effort is a significant contributor to the accidental complexity associated with many heavyweight MDD tool environments.

The above problems stem from a lack of attention to tool usability. Rather than build tools that fit the working style of MDD practitioners, it seems that tool developers/technologists arbitrarily impose particular working styles on tool users. The problem stems from a failure on the part of the tool developers to perform usability studies in which MDD practitioners at different skill levels perform their work using the tools and provide feedback on how well tools support their work. More specifically, tool developers need to practice what is sometimes called empathetic design, where tool design activities are centered on the practitioners that technologists seek to influence. MDD technologists need to put practitioners at the "front and center" of their tool design activities. Currently, technologists work with a model of practitioners that is often not validated, that is, they may be working with flawed models of practitioners. It is thus not surprising that tools often do not fit the working styles of practitioners and thus are perceived as adding to the complexity of developing software-based systems.

There is a need for MDD tools that amplify a modeler's skills and good working habits. The antithesis is that a tool should not force a practitioner to change good working styles, nor get in the way of skillful use of modeling concepts. If a tool imposes a particular working style on practitioners it should be done in a manner that makes the benefits clearly apparent to practitioners (i.e., the effort required to use a tool should be significantly offset by gains in productivity or product quality). Both researchers and technologists are needed to realize this vision of usable tools. Researchers need to conduct studies of how practitioners at various skill levels work. Such empirical results can help technologists build more accurate models of their target audience. Technologists also need to continually evaluate the usability of their tools using representative sets of practitioners, and to evolve their tools accordingly.

In closing, I would also like encourage technologists to consider incorporating features that practitioners can use to develop their modeling skills (i.e., not just support current practices or amplify skills). This can take the form of, for example, suggestions for improving models based on modeling "bad smells" and on the detection of anti-patterns in models. Such tools would be particularly useful in classrooms where next-generation modelers are molded.

## 3.11 Flexible Modeling

*Martin Glinz (Universität Zürich, CH)*

**Joint work of** Glinz, Martin; Wüest, Dustin
**URL** http://www.ifi.uzh.ch/rerg/research/flexiblemodeling.html

In my research group at the University of Zurich, we are working on flexible modeling, developing the FlexiSketch tool as a major constituent of our research. We try to understand what's the essence of a flexible modeling tool that makes it a must-have thing. We also investigate the problem of metamodeling by end- users, empowering them to metamodel without knowing it. Future work will investigate how to organize and navigate large collections of sketches, text fragments and model fragments.

## 3.12 Some Challenges and Expectations on Tool Support for Model-Based Engineering

*Martin Gogolla (Universität Bremen, DE)*

This position statement is formulated from the viewpoint of having some experience in various fields like algebraic specification, temporal logic, entity-relationship modeling, graph transformation, and syntax and semantics of modeling languages. Furthermore, our considerations are influenced by a current work focus on OCL (i.e., a textual expression language) as part of UML (i.e., a modeling language with graphical and textual syntax being able to be used as a general purpose and domain specific language and allowing the characterization of design time and runtime phenomena). Our previous work concentrated on validation and verification techniques for assuring model properties and profits from the development of an academic UML and OCL tool during recent years which is currently used within a larger German project.

Key challenges for MBE tool support include (a) bridging the gap between modeling and programming, (b) efficiently and automatically transforming descriptive models into efficient programs (which may be viewed as prescriptive models) respecting properties assured for source models, (c) enabling communication between modeling tools which have different capability focus, and (d) developing benchmarks (test suites) for models and for modeling tools in order to compare functionality, efficiency, usability, exchangeability, learnability (and other criteria) of modeling techniques and tools. As possible outcomes of future work we identify (a) formal comparison criteria for and modeling features of validation and verification modeling tools and their quality assurance, (b) connecting validation and verification modeling tools with modeling tools having different focus (e.g., transformation tools, code generation tools, performance analysis tools, process management tools, teaching tools), and (c) formal criteria for rating the capabilities of modeling tools.

## 3.13 MBE Tools and Evolving Environments

*Lars Hamann (Universität Bremen, DE)*

Years back I worked as a “non-MBE” developer, but also in this past, models were all around. However, they were only used for specific domains during different development phases, like for example, the relational model of an information system. Unfortunately, these models containing so much information ended up as a historical artifact pinned at a wall. The modeling tools used to generate these artifacts were all well suited to create models in their particular domain, but considered a small part of their surrounding world only. Either, they transformed (called export those days) their model into something with lesser abstraction loosing all the semantic of the higher abstraction or they simply could export it to another (established) tool for the same domain (to be honest, to import data from an established tool was the more common case, because the user should use the importing tool).

In present days, import and export are called model transformation and common data formats like EMF, support transferring models. However, tools still consider only a small part of their surrounding world (call it ecosystem if you like). They define their requirements on the ecosystem in a self-serving way ignoring changes to their outside or, in my opinion, much more important they require modification to the outside. This makes it nearly impossible to compose a tool-set of specialized tools. One future challenge in making MBE-based tools more accepted is to build tools in a more solidary way. To relate the aforementioned observations to a possible meta-model of model-based engineering tool, a possible future meta-model for MBE-tools should support this composition of evolving and not evolving tools. A simple, but important feature would be to integrate versioning of tools and their used meta-models. In a strong sense, it could require a tool to provide model-transformations between versions.

## 3.14 Make Good Languages

*Øystein Haugen (SINTEF – Oslo, NO)*

If Model-Based Engineering Tools should be meta-modeled, then their success will be dependent on the means for meta-modeling. Meta- modeling is dependent on meta-languages such as MOF (standardized by OMG), but experience shows that even extremely well qualified and competent people fail to be able to express their intent in MOF. MOF is not a good domain-specific language for describing languages which is what it should have been to make meta-modeling simpler. We need a better standardized meta-language such that those tools and techniques that are built on the meta-models will have better quality, be more precise and unambiguously interpreted. Let good people make good languages for good tooling.

## 3.15 Engineering Tool Integration: Patterns

*Gabor Karsai (Vanderbilt University, US)*

Model-driven development of systems and software implies that many, different types of models are used. Models can represent anything and everything about the system being constructed, and arguably there is no single model-based engineering tool that can cover all aspects of the process. The MDE community has developed various techniques, standards, and tools for model representation, manipulation, management, transformation, etc. and the problem of model integration has been recognized and appreciated. Model-based engineering tools support an engineering process, where different types of models are used for different purposes. Unless the seamless integration of such models is solved, model-based processes and tools will always be difficult to use and will meet resistance from the practitioners of the traditional, document-oriented processes. A couple of requirements for model integration are straightforward. For model integration we need well-defined *interfaces* between modeling languages, although the precise nature of such interfaces is a subject of research at this point. Arguably mapping between model elements in different modeling languages is a part of the solution, but the interface is probably more complex. A specific model element (*unit of* *knowledge*) must be entered during the engineering process only once, and all derived and dependent elements must be related to that single instance. Models must undergo semantics-preserving transformations – for analysis, code generation, verification, testing, etc., purposes. Frequently transformations must be bi-directional: for instance, analysis models should be derived from design models, but the results of the analysis must be translated back into the language of the design. There has been several model and tool integration patterns developed since model-driven development started. Hub-and-spoke, point-to-point translation, and fine-grain linking of model elements across models are the three major examples. It is a major research challenge to make such model integration approaches usable and scalable. Arguable, the most challenging aspect is the tool semantics: if models are to be translated (or linked) between two tools, there has to be some semantic mapping between the metamodels of these tools. The precise, yet actionable specification and efficient implementation of this semantic mapping remains a continuing research problem. Another challenging aspect is the specific process support: engineering processes are often designed and *tooled* for a specific product family. In other words, the model integration and its executable manifestation should be customizable to specific processes and tools – we need reusable model integration frameworks that allow this. Finally, there are several implementation-related issues: efficient and robust synchronization of models among the tools, distributed version control, support for collaborative work, and usability are the main issues in this area.

## 3.16 MetaEdit+: Faster Meta-Modeling by Design

*Steven Kelly (MetaCase – Jyväskylä, FI)*

Empirical research has shown that using UML does not lead to a significant increase in productivity over coding: plus or minus 15%. MDA tool manufacturers claim 20-40% productivity increases with their tools. Using Domain-Specific Modeling languages in MetaEdit+ has shown productivity improving to 500-1000% of that for coding, consistently across a number of different cases and domains. The design and implementation of the DSM language and generators has been found to take only 2-3 person weeks with MetaEdit+. Recent research by Eclipse modelling project committers (tinyurl.com/gerard12) showed that implementing a modelling language is 10-50 times faster with MetaEdit+ than with other graphical language workbenches, both commercial (RSA, Obeo) and open source (GMF, GME). As yet, it is an open research question as to which features in MetaEdit+ contribute most to this difference.

## 3.17 Demo Abstract for MetaEdit+

*Steven Kelly (MetaCase – Jyväskylä, FI)*

MetaEdit+ is a modeling, meta-modeling and generation tool targeting the creation and use of Domain-Specific Modeling languages. MetaEdit+ applies DSM to itself: the GOPRR language used for defining languages is itself a Domain-Specific Language designed from scratch for this task. In the demo we showed how with MetaEdit+ you can quickly and easily incrementally build a language – including abstract syntax, concrete syntax and transformational semantics: draw models in the language, and automatically generate full code for the modeled system.

## 3.18 Finding Tool Paradigms

*Thomas Kühne (Victoria University – Wellington, NZ)*

In tool design there is a tension between serving users in ways they ask for and prescribing paradigms that the tool designers believe are in the best interest of the user.

While it may take considerable effort on behalf of users to adapt to imposed paradigms, it may be an investment that will produce significant pay-off. An unsatisfied tool user should therefore not only ask "What is wrong with the tool?" but also "What is wrong with me?". Conversely, tool builders need to ask themselves not only "What can I build?" but also "What should I build?" trying to answer the latter question by thoroughly analysing what users need rather than what they want.

## 3.19 Towards Greater Adoption of MDE by Industry Practice

*Vinay Kulkarni (Tata Consultancy Services – Pune, IN)*

**Introduction.** Model-driven software engineering has been around since mid-90s. Launch of OMG's MDA in 2000 generated widespread interest in MBSE. Today it can justifiably be said that model-driven development has proved beneficial in certain niche domains if not all. There is ample evidence of models being used in many ways viz., as pictures, as documentation aids, as jump-start SDLC artefacts, as primary SDLC artefacts etc [1, 2]. When used as primary SDLC artefacts, models shift the focus of software development from coding (in terms of the desired implementation technology primitives) to specifications at a higher level of abstraction [3]. These specifications can be kept independent of implementation technology platforms and hence can be targeted for delivery into multiple technology platforms through platform-specific code generation. Being closer to the problem domain, models present a more intuitive mechanism for domain experts to specify requirements. Being more abstract, size of application specification in terms of models is significantly smaller than its implementation and hence easier to review manually. Being rich in structure, models are amenable for constraints to be specified such that certain errors can be altogether eliminated and certain errors can be detected at modeling stage itself. Model-based code generators can be made to encode best coding practices and standards so as to deliver uniformly high code quality in a person-independent manner. Moreover, it is possible to bootstrap model- driven development so that model-based code generator implementation can be generated from its specification in model form [4]. Intuitiveness demand on models dictate they be domain-specific. Since there can be infinitely many domains with each domain possibly ever- expanding, it is impossible to think of a universal modeling language that can effectively cater to them all. Furthermore,

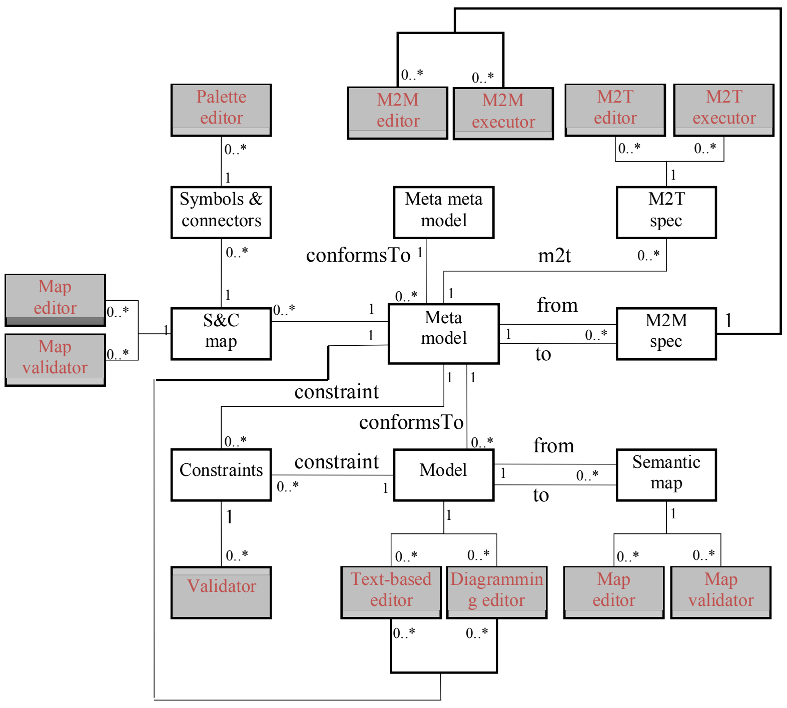


**Figure 2** Meta-Model of Modeling Language Engineering Platform.

models are purposive and hence it is impossible to conceive a single modeling language that can cater to all possible purposes. Therefore, multiplicity of modeling languages is a reality. Separation of concerns principle makes the need for a cluster of related modeling languages (one for each concern in a domain) and a mechanism to relate the separately modeled concerns (say to compose a unified model) apparent. The need to relate otherwise separate models demands expressibility of one in terms of the other. Thus emerges the need for a common language capable of defining all possible modeling languages of interest. There are multiple stakeholders for a model each possibly having a limited view being presented in the form a suitable diagramming notation. From the above discussion, it follows there could be as many diagramming notations as there are modeling languages. And thus emerges the need for a language to define all possible visualizations of a model. For models to be used as primary SDLC artefacts, there needs to be an execution engine for the models – say an interpreter or a transformer to (say) text format that is executable e.g. a programming language. Plus, separation of concerns leading to a cluster of relatable models indicates the need for transforming one model into another and another and so on. Therefore, it can be justifiably claimed that minimal capabilities to comprehensively address the development aspect of software engineering using model driven techniques are: A language to define all possible modeling languages, A language to define all possible visualizations of a model, A language to specify transformation of one model into another, and A language to specify transformation of a model into text artefacts. As a result, with models being the primary SDLC artefacts, software development gets transformed into language engineering endeavour wherein the focus is on defining the most intuitive and expressive modeling language[s] for a given purpose and the necessary execution machinery.

**Modeling language engineering platform.** Figure 1 describes a meta model for configurable extensible modeling language platform. Meta objects coloured grey can be viewed as tool implementations that can be plugged into the platform through a well- defined handshake. The platform should come with inbuilt meta meta model (which is read-only) and model-editing primitives such as InstantiateMetaObject, ConnectTwoMetaObjectsWithAnAssociation, AnnotateMetaObjectWithProperties and corresponding primitives for modification and deletion for the meta objects and associations in the meta meta model. The platforms should also provide a symbols and connectors palette which users can extend further. Primitives for mapping symbols and connectors with user-defined meta model entities should be available out- of-the-box. Platform should enable users specify constraints both at meta model and model levels. It should be possible to plug in suitable validators through well-defined handshake. To ensure easy extensibility, the platform should not make any assumptions about implementation of the tool being plugged in. The platform-to-tool handshake protocol should be externalized, well-defined and stateless. Much of the core technology to implement such a platform is already available. For instance, Eclipse [5] can provide the backbone plug-in architecture for the platform. Eclipse's eCore is a good starting point for the reflexive meta meta model. Text-based [meta] model editors can be realized with little modification, if at all, to the various model editors available. OMG QVT [16] and OMG MOFM2T [7] should suffice as specification languages for model-to-model and model-to-text transformation respectively. Both have many implementations available – licensed as well as freeware variety. In OCL [8], there exists a sophisticated declarative mechanism to specify model constraints. However, it is possible to imagine a situation where a new constraint specification language seems appropriate. Therefore, platform should have the capability to define another constraint specification and execution mechanisms. Enabling diagramming based model editors is a relatively more involved task in absence of externally stated semantics. Plus, there is dependence on presentation manager for rendering the graphics. Ideal solution, that avoids technology lock-in, is to generate platform-specific implementation of the modeling language engineering platform from its specification in model form. This may not be as far out in the future as it seems. A step in this direction has already been accomplished through model-based generation of model-based code generators [4]. Though to be used essentially by knowledgeable users the platform needs to pass minimal usability standards. A graphical user interface providing access to the relevant information through minimal 'clicks' and presenting it in uncluttered manner is the minimal requirement. Ideally, model content accompanying the platform should come organized in the form of a searchable repository. It should be possible to organize the content in the form of components ensuring modularity, high internal cohesion, and explicitly stated coupling. It would be desirable for the component abstraction to support family or software product line concept leading to compact models that can be configured easily [9, 10, 11]. However, the proposed architecture has a significant limitation in absence of a mechanism for specifying semantics (of the models), as a result, the onus of ensuring semantic correctness would be entirely on implementers of the model processing infrastructure. Thus, modeling language engineering platform of Fig. 1 can be viewed as the minimal tooling infrastructure needed for improving productivity of current MDD practitioners. Also, its existence is likely to make MDD enthusiasts to 'take the plunge' so to say. International standards do exist for the key constituents namely i) meta meta model in MOF [12], ii) model to model transformation in QVT [6], and iii) model to text transformation in MOFM2T [7]. Eclipse's plug-in architecture [5] has become defacto standard. The high level of standardization should help develop MDD community for and around the proposed platform. Development (and continuous maintenance) of the proposed platform using open source community model seems the best approach.

**Need to shift the focus.** Focus of modeling community as regards software-intensive systems for enterprises has so far been restricted to achieve platform independence and uniform code quality through model-based code generation. As a result, what gets modeled can at best be said as abstract description of the desired implementation of application under consideration. The very purpose of these models is automatic derivation of the desired implementation through code generation wherein the details regarding design strategies, implementation architecture and technology platform are filled in [3]. The models can also help in computing impact of a change and reflect the change into implementation with minimal side-effects thus optimizing code generation as well as testing effort. Thus, current model- driven development infrastructure is at best equivalent to a language compiler in code-centric development. However, support available there in terms of analyzers, debuggers, profilers etc is missing. The higher level of abstraction makes these shortcomings even more striking in case of model driven development. Ever-increasing penetration of internet and rapid advance of technology are subjecting enterprises to increased dynamics. As a result, enterprise IT systems need to be agile in adapting to the changes in their operating environment. For instance, business applications need to conform to new regulatory compliances such as Sarbane-Oxley [13], HiPAA [14] etc; global banks need to cater to recently opened up developing economies; insurance companies need to introduce right-fit products from time to time; financial services organization decides to cut down total IT cost through infrastructure virtualization; and so on. Only very few of these changes are crisply evident e.g. Sarbane-Oxley compliance. Majority of the changes need deep understanding of the current state of the enterprise IT systems and analysis capabilities to decide what change to introduce where and when. Having decided what needs to be done, comes the how part. However, modeling community has so far focused solely on building the right kind of business applications from their high level specifications – a subset of how part above. The harder problem of identifying what change needs to be introduced where and what would be the likely benefits of this adaptation is largely ignored. In absence of relevant information, management is left to resort to ill-informed decision making. Business- critical nature of enterprise systems means heavy price to pay for wrong decisions. It is not surprising to find that vast majority of enterprise transformation projects are either abandoned early in project life cycle or greatly exceed estimated budget and hardly ever deliver the desired returns on investment [15]. Therefore, modeling community needs to focus on what to model and not how to model. For instance, enterprise wide consistent data views is an important requirement of all enterprises. Current solution is to formulate this as a schema integration problem which adequately addresses static (or structural) aspect but forces one to rely on the program control flows for dynamic aspect. The latter is principally due to lack of suitable enterprise models. The models should help capture core properties such as 'Every credit entry must have a debit entry' in double book accounting. Then these models can be analyzed for such properties. Ability to come up with the right models and the ability to analyze them for properties of interest, we think, will provide a better handle on identifying what needs to change where. Simulation (or interpretive execution) capability of models, say on a representative input data, will help, say, get a more realistic feel of likely benefits. Thus, use of models and model-based techniques can bring more certainty to hard problems such as enterprise transformation [15].

## References

1 John Hutchinson, Mark Rouncefield and Jon Whittle. Model-Driven Engineering Practices in Industry. ICSE'11 pp 633–642.

2 Hailpern, B. and Tarr, P. Model driven development: the good, the bad and the ugly. IBM Systems Journal, 2006, Vol 45, Issue 3, pp 451–461.

3 Vinay Kulkarni, R. Venkatesh, Sreedhar Reddy. Generating Enterprise Applications from Models. OOIS Workshops 2002: 270–279.
4 Vinay Kulkarni, Sreedhar Reddy. An abstraction for reusable MDD components: model-based generation of model-based code generators. GPCE 2008: 181–184.
5 Eclipse – http://www.eclipse.org
6 Query, View and Transformation – http://www.omg.org/spec/qvt
7 MOF Models to Text Transformation language – http://www.omg.org/spec/MOFM2T
8 Object Constraint Lanaguage – http://www.omg.org/spec/OCL/2.0/
9 D E Parnas. Designing software for ease of extension and contraction. ICSE 1978: 264–277.
10 Vinay Kulkarni. Raising family is a good practice. FOSD 2010: 72–79.
11 Vinay Kulkarni, Souvik Barat. Business Process Families Using Model-Driven Techniques. Business Process Management Workshops 2010: 314–325.
12 Meta Object Facility – http://www.uml.org/mof
13 The Sarbanes-Oxley act – http://www.soxlaw.com
14 HiPAA – http://www.hipaa.com
15 W B Rouse. Enterprise Transformation: understanding and enabling fundamental change. John Wiley and sons. 2006.
16 Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT) – http://www.omg.org/spec/QVT/

## 3.20 Servicing IT Needs of Future Enterprises

*Vinay Kulkarni (Tata Consultancy Services – Pune, IN)*

We have delivered large business applications using model driven development approach supported by home-grown standards compliant MDD toolset. In our view the basic technological pieces for supporting model-driven development are in place. Many tools with a varying degree of sophistication exist. Other important aspects such as usability, learnability, performance need to be improved which in essence is a continuous process. However, full potential of model-driven development cannot be realized in absence of ready-to-use models supported by domain ontologies providing the semantic and reasoning basis. This aspect is poorly addressed at present. Inability to translate benefits accruable due to MDE in quantitative terms seems to be a serious stumbling block for adoption of model-driven approach.

Focus of MDE community has been on developing technologies that address how to model. Barring the domain of safety-critical systems, these models are used only for generating a system implementation. Rather, modelling language design/definition is influenced very heavily by its ability to be transformed into an implementation that can be executed on some platform. Modern enterprises face wicked problems most of which are addressed in ad hoc manner. Use of modelling can provide a more scientific and tractable alternative. For which, modelling community needs to shift the focus on analysis and simulation of models. Results from random graphs, probabilistic graphical models, belief propagation and statistics seem applicable here. We believe, it is possible to model at least a small subset of modern complex enterprises so as to demonstrate that model is the organization.

## 3.21 Methods, not Models

*Stephen J. Mellor (Evesham, UK)*

**Main reference** Pick any "methods" book you like from the Eighties and Nineties.

Model-based engineering depends, unsurprisingly, on models. What tends to be emphasized less is how we go about building those models, and how we go about determining the target into which we shall transform those models. In other words, MBE depends on methods.

A *method* is a defined, but flexible, definition of an approach to building models and their subsequent translation into something else, most importantly, executables. However, since the advent of the UML, methods have taken a back seat to notation. Latterly, with increased interest in DSLs and technology for transforming (any) models, methods have been deprecated further to the point that they have all but disappeared. (Save, perhaps, agile "methods", though these can mostly be distilled into social techniques and coding strategies.)

As a consequence, we have tools (MBE all its forms), but little or no guidance on how to use them. Improving the tools so that they produce better code or have better interfaces is clearly of value–and may even induce more people to use them–but without ways to use the tools properly so as to grind though a customer problem and truly understand it, the tools are doomed to fail. We need methods, not models.

## 3.22 Computer Automated Multiparadigm Modeling for the Design of Software Systems in a Physical World

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

**Joint work of** Mosterman, Pieter J.; Biswas, Gautam; Vangheluwe, Hans; Zander, Justyna; Denckla, Ben; Zhao, Feng; Nicolescu, Gabriela; Hamon, Gregoire; Bouldin, Don; Rucinski, Andrzej; Ghidella, Jason; Friedman, Jon
**URL** http://msdl.cs.mcgill.ca/people/mosterman/publications.html

To provide computational support for multiple paradigms in engineering of software systems it is essential that the semantics are well defined in a computational sense, for which semantic anchoring holds great promise. Further value derives from a theoretical formulation vs. a computational implementation. Models of a physical world often rely on abstractions and so it is essential to have these approximations well defined and analyzable so that they can be consistent within a design stage as well as between design stages. Moreover, model exchange such as via model repositories derives much benefit from computationally formulated semantics. With graphical models, the transformations between design stages as well as for semantic anchoring requires efficient graph transformation methods which in addition enables domain-specific languages. Models are very promising to (re)solve system integration issues. For system-level analyses, domain-specific languages for modeling the physics requires consideration of (i) how the physics domain constraints can be formalized, (ii) what good semantic domains are, and (iii) how the information domain interplays with the physics.

**References**

1 Ben Denckla and Pieter J. Mosterman. Stream- and state-based semantics of hierarchy in block diagrams. In *Proceedings of the 17th IFAC World Congress*, pages 7955–7960, Seoul, Korea, July 2008.
2 Pieter J. Mosterman. Implicit Modeling and Simulation of Discontinuities in Physical System Models. In S. Engell, S. Kowalewski, and J. Zaytoon, editors, *The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 35–40, Dortmund, Germany, September 2000.
3 Pieter J. Mosterman and Gautam Biswas. Formal Specifications from Hybrid Bond Graph Models. In *Proceedings of the Qualitative Reasoning Workshop*, pages 131–142, Cortona, Italy, June 1997.
4 Pieter J. Mosterman, Don Bouldin, and Andrzej Rucinski. A peer reviewed online computational modeling framework. In *Proceedings of the CDEN 2008 Conference*, CD– ROM. paper ID pmo131763, Halifax, Canada, July 2008.
5 Pieter J. Mosterman, Feng Zhao, and Gautam Biswas. An ontology for transitions in physical dynamic systems. In: *Proceedings of the National Conference on Artificial Intelligence* (AAAI-98), pages 219–224, Madison, WI, July 1998.
6 Pieter J. Mosterman, Jason Ghidella, and Jon Friedman. Model-based design for system integration. In *Proceedings of The Second CDEN International Conference on Design Education, Innovation, and Practice*, pages CD– ROM: TB–3–1 through TB–3–10, Kananaskis, Alberta, Canada, July 2005.
7 Pieter J. Mosterman and Hans Vangheluwe. Guest editorial: Special issue on computer automated multi- paradigm modeling. *ACM Transactions on Modeling and Computer Simulation*, 12(4):249–255, 2002.
8 Pieter J. Mosterman and Justyna Zander. Advancing model-based design by modeling approximations of computational semantics. In *Proceedings of the 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 3–7, Zürich, Switzerland, September 2011.
9 Gabriela Nicolescu and Pieter J. Mosterman, editors. *Model-Based Design for Embedded Systems*. CRC Press, Boca Raton, FL, 2009. ISBN: 9781420067842.
10 Justyna Zander, Pieter J. Mosterman, Gréegoire Hamon, and Ben Denckla. On the structure of time in computational semantics of a variable-step solver for hybrid behavior analysis. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, September 2011.

## 3.23 The Original Sin – Using Current Computers for Modeling Software

*Pierre-Alain Muller (University of Mulhouse, FR)*

Computers are supposed to be able to do anything, I guess, and this is why computer scientists one day got the idea that computers could be used for developing software as well. And this might be wrong.

My new thesis is that current computers are not capable of supporting the informal, unstructured, emerging, global, creative process that goes on when people think about the software which should be developed for a given (often partially unknown) purpose.

This is also the reason why programmers do not model. They are using wonderfull computerized tools for programming, and naturally stop to use computers when they “feel” that computers are not appropriate anymore.

If modeling could be automated by computers with the same kind of efficiency than programming, I’m sure that they would use computers (for modeling).

So the real issue is: “can we come up with that ideal computerized system for modeling”? (in other words, the meta-model for model-based engineering tools, which this seminar is all about).

## 3.24 The Other Face of Models

*Leonel Domingos Telo Nóbrega (University of Madeira – Funchal, PT)*

TModels are traditionally used in software engineering for documenting aspects of design and implementation, and, in some cases, to generate the code for the system. Though there remains debate about this type of approaches and the role, qualities and properties that models should have, the existence of models that conforms to well-defined modeling languages allows other types of uses that goes beyond the aforementioned. Visualization techniques can enable the mining of information about the models and give an innovative perspective that will contributes positively to a better understanding, analysis and validation of models.

## 3.25 The MetaSketch Workbench

*Leonel Domingos Telo Nóbrega (University of Madeira – Funchal, PT)*

**Main reference** L. Nóbrega, N.J. Nunes, H. Coelho, “The Meta Sketch Editor,” in G. Calvary, C. Pribeanu, G. Santucci, J. Vanderdonckt (Eds.), Computer-Aided Design of User Interfaces V, ISBN 978-1-4020-5819-6, pp. 201–214, Springer Netherlands.
**URL** http://dx.doi.org/10.1007/978-1-4020-5820-2_17

The MetaSketch is a modeling language workbench that allows the rapid development of modeling languages, creating new possibilities and opportunities in the context of the Model-Based Development approach. The technology behind the MetaSketch is based on OMG standards like MOF 2.0, OCL 2.0 and XMI 2.1, and is specially tailored for creating new members of the UML family of languages. Currently, this workbench comprises a set of tools – the Editor, the Merger, the Comparer, the Visualizer and the Designer – and all of them are in a pre-beta release stage:

- MetaSketch Editor – The editor is the heart of the workbench and is simultaneously a modeling and a metamodeling editor. In a practical sense, this means that the same editor can be used to create the metamodel and the models that conforms to the created metamodel.
- MetaSketch Merger – This tool is only applicable on a metamodel and basically resolves all the Package Merge’s used on the definition of the metamodel. In addition, all hierarchy of package is flattened, resulting in a metamodel with only one package that contains all language constructs definition merged.

- MetaSketch Comparer – Tool to compare easily two models described in XMI.
- MetaSketch Visualizer – This tool was originally developed for improve the navigability of modeling Editor. The base idea is to use any kind of relationship within model's elements to create views different than existing diagrams. Further, it is possible to visualize quantitative data extracted forma models through charts. To query engine to generate the data for the visualization use an Imperative OCL interpreter.
- MetaSketch Designer – This is a work in progress and will allow graphically create the definition of the concrete syntax for each language constructs and type of diagram. Currently, these definitions have to be done directly in the XML files.

## 3.26 Improving the Usability of Formal Verification Techniques Through Customizable Visualizations

*Ileana Ober (Paul Sabatier University – Toulouse, FR)*

**Joint work of** Ober, Ileana; Ober, Iulian; Aboussoror, El Arbi
**Main reference** E.A. Aboussoror, I. Ober, I. Ober, "Significantly Increasing the Usability of Model Analysis Tools Through Visual Feedback," in Proc. of the 16th Int'l SDL Forum on Model-Driven Dependability Engineering (SDL'13), LNCS, Vol. 7916, pp. 107–123, Springer, 2013.
**URL** http://dx.doi.org/10.1007/978-3-642-38911-5_7

One of the in hallmarks of the massive and rigorous introduction of the use of models in software development life cycles, is the openings they offer towards early model based verification and validation. During the last decades, the formal methods communities have developed interesting and powerful theoretical results that allow for advanced model checking and proving capabilities. Their usage is hindered by the complexity of information processing demanded from the modeler in order to apply them and to effectively exploit their results.

One of the research directions that we are about to explore in my research group concerns the definition of mechanisms that decrease the cognitive effort demanded from the user in order to use traditional model checking tools. This consists in defining flexible and carefully selected visual presentations of the feed-backs provided by classic model checking tools used in the context of high- level (UML and SysML) models. This ranges from simple presentation adjustments (highlighting new information or coloring things that change their values) to defining, based on feed-backs from domain experts, new diagrams (such as message exchange diagrams). Offering flexible visualization customizations opens the way to new types of scenario visualizations, improving scenario understanding and exploration. This approach was implemented in our UML/SysML analyzer and was validated in a controlled experiment that shows a significant increase in the usability of our tool, both in terms of task performance speed and in terms of user satisfaction.

Our thesis is that still a lot of effort is needed in order to make available tools and techniques allowing the user to take benefit of the use of modeling and abstraction without requiring. The results we obtained [1], in terms of tool functionalities and user evaluation, make us confident in the feasibility of this objective.

### References

1 El Arbi Aboussoror, Ileana Ober and Iulian Ober. *Seeing Errors: Model Driven Simulation Trace Visualization.* In Proceedings of the 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012, LNCS 7590

## 3.27 Empirical Studies of Expert Software Design (Cuckoo or Canary?)

*Marian Petre (The Open University – Milton Keynes, GB)*

Based on empirical studies of professional software developers, this talk identifies key features of the notations developers actually use for early design, and remarks, from a cognitive perspective, on fundamental challenges arising from the complex nature of 'modeling'. What software designers produce when they're unconstrained has some commonalities and emphasises movement between perspectives, between representations, between levels of abstraction, and between alternatives. Their sketches also preserve indications of provisionality. Similar characteristics – fluidity, ease of annotation, ability to change representation, expressive juxtaposition, escape from formalism, provisionality, indications of history – facilitate design discussions. One of the things that has struck me over time is that we keep finding new contexts, without quite answering the deep questions that cut across them. How can we support: reasoning across levels of abstraction; understanding the consequences of design decisions; deriving operational models; managing the tradeoff between what information a representation makes accessible and what it obscures; reasoning about interactions and emergent behaviours; preserving rationale, provenance and provisionality? From a cognitive perspective, modeling is not one activity, and addressing the needs of one constituent may be problematic for another. Moreover, possibly the most reliable result in the psychology of programming is the impact of individual differences. One size does not fit all.

## 3.28 Bridging the Gap Between Software Models and Analysis (Performance, Reliability, etc.) Models

*Dorina C. Petriu (Carleton University – Ottawa, CA)*

**Joint work of** Petriu, Dorina C.; Alhaj, Mohammad; Tawhid, Rasha
**Main reference** D.C. Petriu, M. Alhaj, R. Tawhid, "Software Performance Modeling," in Proc. of the 12th Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'12), LNCS, Vol. 7320, pp. 219–262, Springer, 2012.
**URL** http://dx.doi.org/10.1007/978-3-642-30982-3_7

Many formalisms and tools (such as queueing networks, stochastic Petri nets, stochastic process algebras, fault trees, probabilistic time automata, etc.) for the analysis of different nonfunctional properties (NFPs) of systems and software (such as performance, reliability, availability, scalability, security, etc.) have been developed over the years. Traditionally, analysis models used to be built "by hand" by the analyst. More recently, in the MBE context, the following approach for the analysis of different NFPs has emerged in literature: a) extend the software model to be analyzed with annotations specific to the respective NFP; b) apply a model transformation to derive an analysis model expressed in the selected formalism from the annotated software model; c) analyze the NFP model using existing solvers; and d) interpret the results and give feedback to the software developers.

In the case of UML-based software development, the extensions required for NFP-specific annotations are defined as UML profiles, which have the advantage to be processed by standard UML tools without any change in the tool support. For instance, two standard UML profiles provide, among other features, the ability to define performance annotations:

the UML Profile for Schedulability, Performance and Time (SPT) defined for UML 1.X versions (OMG, 2005) and the UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) defined for UML2.X versions (OMG, 2009).

An important research challenge is to hide, as much as possible, the analysis formalism and the complexity of the analysis models from the software developers, while at the same time providing analysis results and advice for improvement in the context of the software model. A list of research questions is as follows:

- Model transformation from software model to analysis model: keep the separation of concerns between the platform-independent model (PIM) of the software under development and the underlying platform model. Platform modeling requirements are: provide support for reusable, generic, parametric platform models; provide support for flexible deployment models (because resource allocation is important for many NFPs).
- Bridging the gap between the software model and the analysis model trough trace-links between the elements of the two models. Such trace-links will be used, for instance, for providing analysis feedback to software developers (analysis results, advice).
- Integrate multiple NFP analyses in the software development process (e.g., performance, reliability and security): a) For each NFP, explore the parameter space for different design alternatives, configurations, workload parameters, etc., in order to find the "best solution"; b) Toolset should provide support for an experiment controller for exploring the parameter space; c) Software process issue: how integrate the evaluation of multiple NFP in the software development process.
- Tool interoperability: Input and output format of the analysis tools (many created a long time ago) are tool specific and often unfriendly to automated processing.
- Impact of software model changes on the NFP analysis: it is desirable to be able to propagate changes incrementally, rather than re-building the analysis models from scratch every time the software model is changed.

## 3.29 The Need for Scalable and Flexible Model-Based Engineering

*Louis Rose (University of York, GB)*

I briefly summarised recent work in my research group on improving the scalability of modelling tools and techniques, and on increasing the degrees of flexibility in modelling tools and techniques. I argued in favour of better supporting iterative and incremental development in model-based engineering tools by, for example, favouring dynamic and weakly typed environments. Finally, I spoke about the way in which I believe that model-based engineering does and does not address issues of software maintenance, which remains one of the most costly elements of software engineering.

## 3.30 EuGENia Live: A Flexible Graphical Modelling Tool

*Louis Rose (University of York, GB)*

**Joint work of** Rose, Louis; Kolovos, Dimitrios S.; Paige, Richard F.
**Main reference** L.M. Rose, D.S. Kolovos, R.F. Paige, "EuGENia Live: A Flexible Graphical Modelling Tool," in Proc. of the Extreme Modeling Workshop (XM'12), pp. 15–20, ACM, 2012.
**URL** http://dx.doi.org/10.1145/2467307.2467311
**URL** http://www.di.univaq.it/diruscio/sites/XM2012/xm2012_submission_6.pdf

Designing a domain-specific language is a collaborative, iterative and incremental process that involves both domain experts and software engineers. Existing tools for implementing DSLs produce powerful and interoperable domain-specific editors, but are resistant to language change and require considerable technical expertise to use. I presented EuGENia Live, a tool for rapidly prototyping and designing graphical DSLs. EuGENia Live runs in a web browser, supports on-the-fly metamodel editing, and produces DSLs that can be exported and used to produce an initial implementation in the Eclipse Modeling Framework. EuGENia Live draws on existing work from the fields of flexible modelling and model-driven engineering.

## 3.31 Compositional Model Based Software Development

*Bernhard Rumpe (RWTH Aachen, DE)*

**URL** http://monticore.de

Model based software development promises to strongly improve efficiency and quality in software development projects. However, MBSE has not yet delivered it's promises yet.

In the talk, we examine the current state and problems of MBSE and discuss a number of approaches to tackle those. In particular, we discuss how to make use of models in large development projects, where a set of heterogenous models of different languages needs is developed and needs to fit together.

A model based development process (both with UML as well as a domain specific modeling language (DSML)) heavily relies on modeling core parts of a system in a redundant free form, having compositional generators to early and repeatedly cut code and tests from these models.

We in detail discuss compositionality on models and heterogeneous modeling languages and how it supports agile development as well as reuse of language and tooling infrastructures.

And we show what the current status of compositionality is vs. a bunch of interesting languages given.

We finally demonstrate what has already been achieved in the language workbench MontiCore developed in our group over the recent years.

## 3.32 Formal Verfication and Model-Based Engineering

*Martina Seidl (University of Linz, AT)*

As in traditional hardware and software system development, also in model-based engineering formal techniques find many applications within the development process in order to verify that certain properties of the system under development hold. In the context of the Austrian research project FAME (supported by the Vienna Science Fund under grant ICT10-018; www.modelevolution.org), we currently investigate how different verification techniques may be used to support the evolution process of software models. Therefore, we use various reasoning engines like SAT solvers and model checkers not only to verify that evolution does not violate any constraints but also for calculating correctly evolved models. For example, the satisfiability problem of propositional logic is used to encode consistent merging of concurrently evolved versions of one sequence diagram with respect to a given set of state machines [1]. The SAT solver is used to suggest the modeler a set of correct solutions from which the most convenient approach may be selected. During this work, we encountered several challenges. First of all, we need a concise semantics in order to specify the verification problem. Having an exact formulation of the verification problem, the encoding in the respective formalism almost directly derivable. Second, there is a huge gap between the representation in the modeling language and the input of the verification tool. For example, in the case of a SAT solver, the problem is encoded as a sequence of numbers. In order to make the output interpretable for humans, it has to be translated back to the model representation. In order to highlight the solutions found by the SAT solver, we annotate the solution with colors for directing the human modeler's attention to the particular point in the model where modifications have been performed in order to ensure the consistency. The third major challenge involves the testing of our approach. On the one hand we took manually engineered models from the literature. These models allowed us to test the basic functionality of our approach, but they were too small to evaluate the performance of our approach. Therefore, we implemented a random generator which allows us to generate suitable input models of arbitrary size. These models showed not only valuable to improve the performance of our approach, but it helped us also to find bugs in our implementation which were not triggered by the manually engineered test models. Besides investigating how formal verification techniques support various evolution tasks in model-based engineering, we are also investigating how models can support the development of verification backends like SAT solvers. A model-based testing approach [2] shows to be be extremely valuable for building modern SAT solvers, which consists of strongly optimized code (usually in C) in order to be able to tackle verification problems of practical size. Such systems are too complex to be verified and therefore, much emphasize has to spend on testing their correctness. We propose to describe correct sequences of API calls, input data, and option configurations in terms of state machines which may then be used for grammar-based blackbox fuzzing as well as for delta debugging. Whereas the former technique generates the test cases, the latter reduces an error triggering test case such that manual debugging is feasible. By this means an automatic test chain is realized. In future work, we plan to extend this approach to other solving systems like SMT solvers. To sum up, both model-based engineering techniques and formal verification techniques can benefit from each other and an exchange of ideas and approaches of the different fields may bring valuable insights and advances to both research fields.

**References**

1 Magdalena Widl, Armin Biere, Petra Brosch, Uwe Egly, Marijn Heule, Gerti Kappel, Martina Seidl, Hans Tompits. Guided Merging of Sequence Diagrams. SLE 2012. 164–183.

2 Cyrille Artho, Armin Biere and Martina Seidl. Model-Based Testing for Verification Backends. Accepted for Tests and Proofs (TAP) 2013.

## 3.33 MBE Tools: Effective Automation Where It Helps

*Bran V. Selic (Malina Software Corp. – Nepean, CA)*

**Main reference** B. Selic, "What will it take? A view on adoption of model-based methods in practice," Journal of Software and System Modeling, October 2012, Volume 11, Issue 4, pp. 513–526.
**URL** http://dx.doi.org/10.1007/s10270-012-0261-0

Increased levels of computer-supported automation are considered as one of the key enablers to the higher levels of productivity and product quality promised by model-based engineering (MBE) . However, practical experience with present-day MBE tools indicates that we are still far from this ideal. In fact, if anything, these tools are proving to be an impediment to productivity; they are complex, difficult to learn, and difficult to use. Users of these tools often find themselves in frustrating situations where the tools are blocking their progress, forcing them into workarounds and constrained operating modes and procedures which are not conducive to free expression of ideas.

Much of this can be attributed to the tool designers' poor understanding of their users, how they work and what motivates them. There is often a naïve and implicit image of a "typical user", based on very shallow analysis and a distressing lack of awareness of the complexities involved in human-machine interaction. (This is a problem even when tool developers are users of their own tools – something that, unfortunately, does not happen often enough for MBE tools – since in those cases developers often lose sight of the fact that their users do not have the same deep understanding of a tool's architecture as they do.) In such settings, usability is rarely considered an architectural-level concern, but as something that is merely a presentation issue to be solved by a 'suitable" user interface.

To get the full benefits that we expect from the automation potential of MBE tools, we must not only understand how and where it should be applied, but equally importantly, knowing where it is inappropriate. (Just because something can be automated does not mean it should be.) A key research issue related to this is finding ways to support the transition between an informal and provisional mode of operation, which is conducive to design exploration, and the formal mechanistic world required for computer-based design capture. Ideally, this should not be a one-time one-way transition, which could lead to premature commitment, but a continuous back-and-forth interplay of these modes as design progresses.

## 3.34 Models in Software Engineering

*Perdita Stevens (University of Edinburgh, GB)*

I gave an abbreviated version of a talk I have given in Edinburgh in the series of Hamming Seminars, whose remit is to discuss what were the major problems in a given field. Here is the abstract for the original talk.

A model is (for purposes of this talk) an abstract, usually graphical, representation of some aspect of a software-intensive system. Software engineering has, since its invention, involved the use of models. In recent decades, a perceived need for greater automation of processes in software engineering, the wish to develop software faster and more cheaply, and the drive for higher quality have motivated ever more reliance on models. Languages for modelling and techniques for working with models have struggled to keep pace, hindered in some cases by conflicting requirements arising from the different motivations. We are currently in a dangerous state in which models sometimes endanger rather than support the developers' aims. I will explain some of the problems, and discuss the progress that I think we will/may/should/must/cannot expect to see in the coming years

## 3.35 Tool Building Using Language Composition

*Laurence Tratt (King's College London, GB)*

Language composition allows users to mix languages together in a fine-grained manner. It can complement modelling by giving the possibility to mix together modelling and non-modelling languages. We presented a new language composition editor that allows arbitrary language composition to feel like using a normal text editor. It also allows textual and non-textual languages to be composed and edited together.

## 3.36 Generic Tools, Specific Languages

*Markus Völter (Völter Ingenieurbüro – Heidenheim, DE)*

In my talk, and in the mbeddr tool demo, I introduced an new approach to developing domain-specific software engineering tools. It promises a significant increase in the fidelity of adapting generic tools to specific domains, while significantly reducing the effort of the adaptation. The approach, called Generic Tools, Specific Languages, shifts the focus from the engineering *tool* to the underlying *languages*. Recent advances in language engineering – and the language workbenches that support these advances – enable the modular, incremental extension of languages. Projectional editing supports increased notational freedom, such as tables or mathematical symbols. These two ingredients enable the shift in focus for developing domain-specific engineering tools: a *generic tool*, the language workbench, hosts arbitrary languages and provides IDE support such as syntax coloring, code completion, go-to-definition and find-references as well as model search, refactoring, debugging or visualization. On top of this platform, a set of very *specific languages* adapt the tool to a given domain. Some actual tool adaptations (windows, buttons, context menus) are sometimes necessary, but they are few and far between and are always related to languages or language extensions. mbeddr (http://mbeddr.com) is an example implementation of this approach for embedded software engineering. It builds on top of the JetBrains MPS language workbench (http://jetbrains.com/mps)

## 3.37 The Tool's The Thing

*Jon Whittle (Lancaster University, UK)*

In this talk, I review literature on tools from psychology, sociology, philosophy, management studies and computer science. Tool use has been studied for centuries and MDE researchers should try, as much as possible, to take such literature into account when developing new MDE tools. The talk also presents a recent study by Whittle, Hutchinson, Rouncefield, Burden and Heldal, which carried out 40 interviews will companies applying MDE tools in practice. Interview data was analyzed using a grounded approach to identify emergent themes related to MDE tool use. The result is a taxonomy of 30 technical, social and organizational factors of MDE tools which affect their adoption in practice.

## 3.38 Flexible, Lightweight Metamodeling

*Dustin Wüest (Universität Zürich, CH)*

**Joint work of** Wüest, Dustin; Seyff, Norbert; Glinz, Martin
**Main reference** D. Wüest, N. Seyff, M. Glinz, "FlexiSketch: A Mobile Sketching Tool for Software Modeling," in D. Uhler, K. Mehta, J.L. Wong, (Eds.), Mobile Computing, Applications, and Services, LNICST, Vol. 110, pp. 225–244, Springer, 2013.
**URL** http://dx.doi.org/10.1007/978-3-642-36632-1_13

Current metamodeling tools provide the technical means for constructing all kinds of domain specific languages (DSLs), but they have two disadvantages. First, they are hard to learn and require experts in both the problem domain and metamodeling. Second, they neglect the processes used in practice to come up with good DSLs. Tools are not adjusted to the methods how practitioners create DSLs. Creators of metamodeling tools assume that the users know what they are doing and have the completed DSLs already in their minds when they start to use the tools. In contrast, creating good DSLs is often an iterative process, where engineers interleave between constructing a DSL and creating model examples of it. These models can help to show weak spots in a DSL, and the DSL evolves over time.

We argue for the need of flexible metamodeling tools that enable users with only basic metamodeling knowledge to incrementally build DSLs based on concrete model examples. Domain experts should be able to create modeling languages step by step and repeatedly test them by drawing example models. This also implies that tools must support different level of details for language definitions and do not require a complete language description before it can be tested in a productive environment. At the beginning, it must be possible to define individual elements of a modeling language at different abstraction levels, depending on how much detail is needed or already known. Metamodeling tools should foster creativity and allow trying out different design ideas by providing fast feedback cycles. To achieve this, it must be possible to quickly change a metamodel and try out the new version. It helps if a model editor is generated or updated on the fly.

For being able to come up with better metamodeling support in practice, we have to know i) who the typical users are, ii) how much they know about metamodeling, iii) what methods and processes they use to perform metamodeling in practice, and iv) what their requirements for metamodeling tools are. In addition, we should exploit the possibilities that a tool-supported approach gives for guiding less experienced users in performing metamodeling.

## 3.39 Liberating Software Engineers from the Tyranny of a Strict Modeling Language

*Dustin Wüest (Universität Zürich, CH)*

**Joint work of** Wüest, Dustin; Seyff, Norbert; Glinz, Martin
**Main reference** D. Wüest, N. Seyff, M. Glinz, "Flexible, lightweight requirements modeling with FlexiSketch," in Proc. of the 20th IEEE Int'l Requirements Engineering Conf., pp. 323–324, IEEE, 2012.
**URL** http://dx.doi.org/10.1109/RE.2012.6345826

Creativity and discussing different design alternatives play central roles in early phases of software development. Many modeling tools do not provide adequate support for these phases, because they restrict users in utilizing a particular modeling language and level of detail. Having to think about how to express ideas in a particular language while coming up with the ideas hinders the creative flow. Also, different ideas may be expressed at different levels of detail, while a modeling language typically only supports one level of detail. Therefore, whiteboards and flip charts are still dominant tools for early software design phases. Engineers have to accept the drawback of manually re-creating the information from a whiteboard in a software modeling tool if they want to re-use and refine the information.

We presented FlexiSketch, a flexible and lightweight modeling tool for mobile devices. The tool allows freeform sketching and the creation of nodes-and-edges diagrams. Modeling and metamodeling can be performed in a single environment. Engineers do not have to adhere to a predefined modeling language. Instead, they can quickly sketch different design ideas. Metamodel information can be added to individual diagram elements on demand, which enables a step-wise formalization and refinement of model sketches. This also means that domain specific languages can be built in an iterative, incremental way. Short feedback cycles foster creativity and invite to try out different design alternatives.

### 3.40 Design and Design Thinking

*André van der Hoek (University of California – Irvine, US)*

**Main reference** M. Petre, A. van der Hoek, (Eds.), "Software Designers in Action: A Human-Centric Look at Design Work," Chapman and Hall/CRC, forthcoming Sep 2013.
**URL** http://sdcl.ics.uci.edu/research/calico

Model-driven engineering requires software developers to design. A large body of literature concerns itself with design thinking – ways in which designers work through a design problem. To date, this literature has not been examined from the perspective of software design. This talk makes the case for taking a design thinking look at software design, and highlights some of the dimensions of what such an exploration would look like.

## 4 Overview of Working Groups

### 4.1 Working Group A: Summary of Discussions and Conclusions

*Bran V. Selic (Malina Software Corp. – Nepean, CA)*

**Joint work of** Larry Constantine, Robert B. France, Martin Glinz, Lars Hamann, Steven Kelly, Thomas Kühne, Vinay Kulkarni, Stephen J. Mellor, Pierre-Alain Muller, Leonel Domingos Telo Nóbrega, Marian Petre, Dorina C. Petriu, Louis Rose, Bran V. Selic, Perdita Stevens, André van der Hoek, Jon Whittle, Dustin Wüest

The participants agreed to separate the issues to be addressed into two broad categories. One category dealt primarily with technical issues related to modeling and modeling tools. The second category addressed topics related to the use of modeling tools in context. This included a wide spectrum of concerns, from recognizing end-user business needs, to understanding how domain experts do design, to identifying and overcoming tool usability issues. Fortunately, among the participants were a number of researchers who specialized in studying software design practices and usability, who helped provide the necessary multi-disciplinary perspective.

A general consensus quickly emerged that the flaws of the current generation of rigidly conceived modeling tools were standing in the way of agile design, and thereby impeding broader adoption of models and modeling in industrial practice. One of the root causes of this seems to be lack of support for provisionality in design, that is, the ability not only to tolerate but to actively support the type of informal processes and ambiguous artifacts produced during design exploration. Case studies of industrial projects have shown that design involves a myriad of aspects that are not easily replicated by computer-based automation, including non-verbal communications, analogies, and the use of ad hoc informal notations and terms. Collaboration is a key ingredient in such processes that should be supported by tools, but without imposing undue structure that blocks active discourse. This raises the issue of understanding what should (and what should not) be automated in this process and, if so, how? A closely related challenge is determining how best to transition back and forth between the informal and highly flexible world of creative thinking and the much more formal world of design capture and implementation, or between different purposes of models, such as descriptive and prescriptive uses of models.

From these considerations, the notion of tools that are "fit for purpose" emerged as a useful formulation and desirable objective. This, of course, implies a highly human-centric approach to the design of such tools. There are many different uses of and many different perspectives on models that must be accommodated. One way of achieving this is to focus less on users per se and more on their activities, purposes, and expertise levels (expert users work in very different ways from novices). This means highly adaptable and customizable tools, allowing users to organize them according to the task and situation at hand, rather than based on some preconceived hardwired workflow or, worse yet, on some automated inference strategy built into the tool (experience has shown that such inferences are often wrong and annoying). Workflows should not be interrupted by petty and inflexible formal rules.

Finally, an additional topic discussed in this context was how to create and popularize a modeling culture among software practitioners by teaching the value and use of models early in software engineering curricula.

At the end of the workshop, a number of key research questions were identified that related to the issues described above (NB: many of these questions are closely related and even overlap):

1. Investigate the transitions that occur in model-based development (e.g., between the informal and formal domains, between models for different purposes, between models at different levels of abstraction) and how this can be supported by tools.
2. Identify the various modeling purposes and what would be required to construct corresponding "fit for purpose" tools.
3. Study how to specify and manage the relationships between different models intended for different purposes.
4. Understand how a model intended to support one purpose can be retargeted for a different purpose.
5. Understand the actual and potential end-user value propositions and if and how they can be supported by models and modeling.
6. Collecting evidence of the value of models in practice (industry and research).
7. Study what to (and what not to) automate in modeling. (When? Why?)
8. How can tools help us capture and exploit the provisionality inherent in design.
9. Investigate ways of fostering new generations of model-aware practitioners. (How should we teach modeling? How do we make it relevant?)
10. Understand how model-based engineering fits into modern agile development processes.
11. Investigate and clarify the differences (social, cultural, technical, etc.) between practitioners of "traditional" software development methods and those using models and modeling; Based on that understand why so many practitioners resist model-based methods.

## 4.2 Working Group B.1: Composition Issues in MBE Languages and Tools

*Tony Clark (Middlesex University, GB)*

**Joint work of** Colin Atkinson, Tony Clark, Benoit Combemale, Lukas Diekmann, Øystein Haugen, Gabor Karsai, Steven Kelly, Vinay Kulkarni, Stephen J. Mellor, Pieter J. Mosterman, Dorina Petriu, Bernhard Rumpe, Laurence Tratt, Markus Völter.

The hypothesis of group B.1 is that the problem of complexity (and therefore usability) of Model Based Engineering tools can be addressed through *composition*. Methods and technologies for modularity and composition can help by providing a basis for reuse of tool elements and a basis for understanding otherwise highly complex technologies.

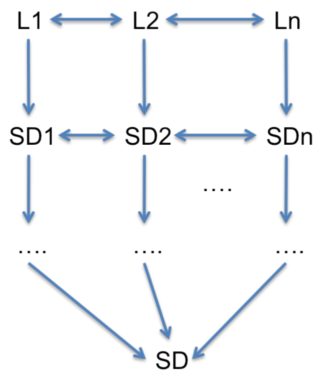
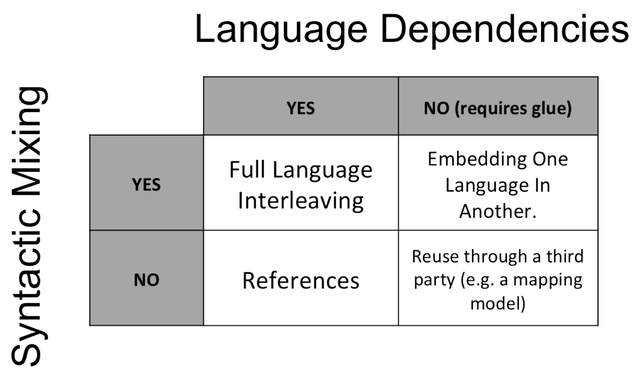


**Figure 3** Finding a Common Semantic Domain.



**Language Dependencies**

| Syntactic Mixing | YES | NO (requires glue) |
|---|---|---|
| YES | Full Language Interleaving | Embedding One Language In Another. |
| NO | References | Reuse through a third party (e.g. a mapping model) |

**Figure 4** Categories of Language Composition.

| | |
|---|---|
| **Syntax** | Includes the abstract and concrete syntax, graphical, textual, mixed. |
| **Semantics** | Includes models of computation, denotation, translational, mathematical, logic, |
| **Implementations** | Code generators, interpreters. |
| **Analytics** | Includes the type system, well-formedness rules, language properties and their definitions, relationships to analysis tools. |
| **Tools** | Editors, debuggers, version control, life-cycle management. |
| **Usage** | Guidelines, development process, documentation, life-cycle. |

**Figure 5** A Proposal for a Composable Language Module.

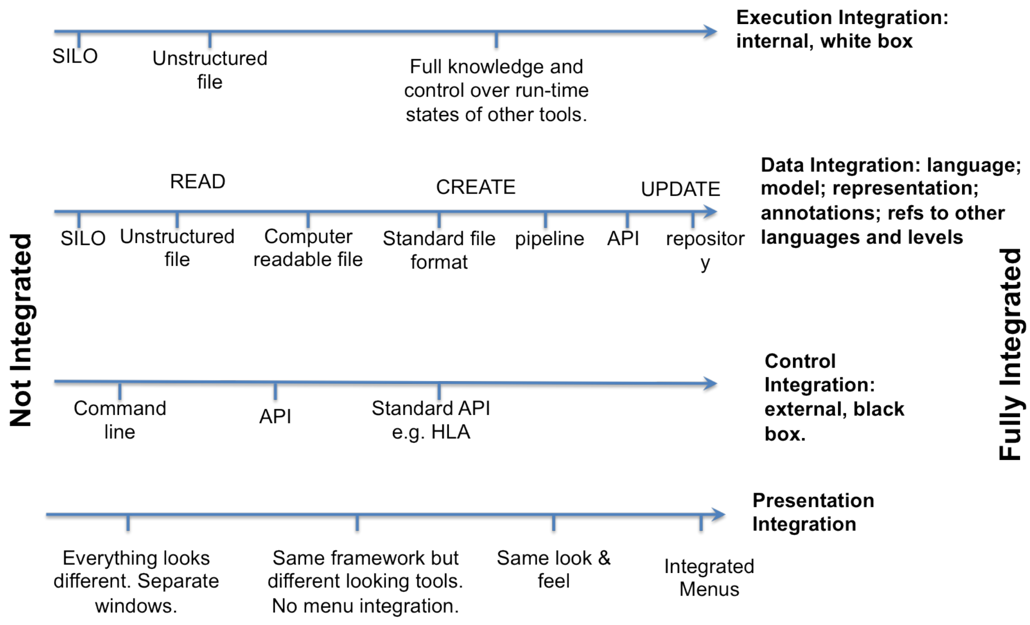

Execution Integration: internal, white box
SILO
Unstructured file
Full knowledge and control over run-time states of other tools.
Data Integration: language; model; representation; annotations; refs to other languages and levels
READ
CREATE
UPDATE
SILO
Unstructured file
Computer readable file
Standard file format
pipeline
API
repository
Not Integrated
Fully Integrated
Control Integration: external, black box.
Command line
API
Standard API e.g. HLA
Presentation Integration
Everything looks different. Separate windows.
Same framework but different looking tools. No menu integration.
Same look & feel
Integrated Menus

**Figure 6** A Spectrum of Tool Composition.

The group identified four different facets of composition that were discussed over several sessions leading to the identification of a series of key research questions:

**Semantic Heterogeneity** arises because MBE languages and tools often consist of multiple sub-components each of which has a specific semantics all of which must be made consistent in order for the tools to work together. Figure 3 (proposed by Bernhard Rumpe) shows an approach to composition which seeks to identify a single common semantic domain SD through multiple refinement transformations of individual language semantic domains. The group also discussed whether it would be possible to produce a small number of agreed models of computation that are reused as a semantic basis for modelling languages. By agreeing such a set of domains, maturity and well-understood relationships between them would evolve over time.

**Language Composition** often occurs when a tool offers the ability to model multiple system aspects, and is key to language reuse. The group spent time trying to identify different types of language composition and produced the categorization shown in Figure 4 (proposed by Markus Völter) where languages can be integrated to different levels. The group also discussed what an ideal composable language module might look like as shown in Figure 5 together with a likely set of composition operators including union and embedding.

**Tool Composition** occurs when building tool-chains or reusing modular tool components. Several aspects of tool composition were identified and described as a spectrum of integration from *not integrated* to *fully integrated* as shown in Figure 6 (proposed by Steven Kelly).

## 4.3 Working Group B.2: Research Questions for Validation and Verification in the Context of Model-Based Engineering

*Martin Gogolla (Universität Bremen, DE)*

**Joint work of** Catherine Dubois, Michalis Famelis, Martin Gogolla, Leonel Nóbrega, Ileana Ober, Martina Seidl, Markus Völter

**Notions 'Validation' and 'Verification' (V&V):** The goal of the using validation and verification techniques is to better understand models and to uncover model properties which are stated implicitly in the model (see Fig. 7). Validation and verification serves to inspect models and to explore modeling alternatives. Model validation answers the question 'Are we building the right product' whereas model verification answers 'Are we building the product right'. Validation is mainly an activity by the developer demonstrating model properties to the client, whereas in verification the developer uncovers properties relevant within the development process. Verification covers testing as well as automatic and semi-automatic proving techniques.

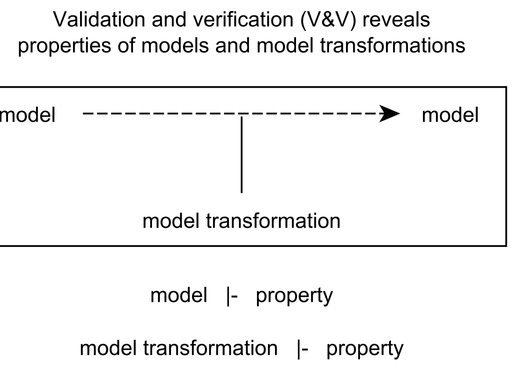

Validation and verification (V&V) reveals
properties of models and model transformations
model
model
model transformation
model |- property
model transformation |- property

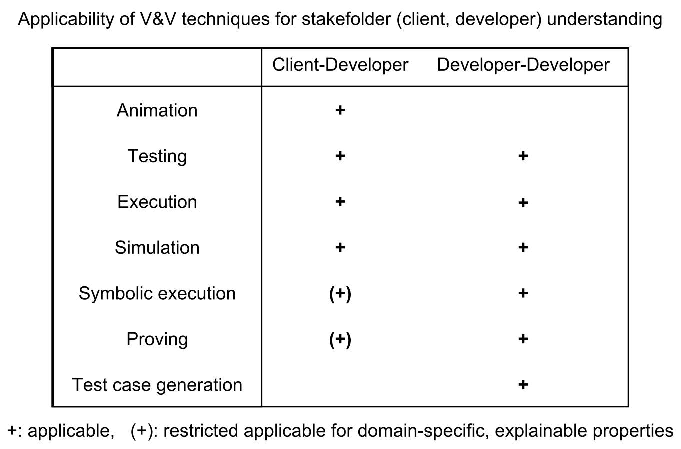
**Figure 7** Validation and Verification of Models and Transformations.

**Relationships between Client, Developer and V&V Engine:** How do we express properties at the level of models in a way understandable to clients? How do we formulate models and properties in a single language transparent to clients? How do we report the validation and verification results and diagnostics in an appropriate form to clients? How do we bridge the gap between formally expressed and verified properties on one side and client attention on the other side? Can modeling language extensions help in making explicit the needs of validation and verification machines?

**Design VS Time-Runtime:** How do we obtain during the validation and verification phase an initial model instanciation on the model runtime level which is determined by the model design time description? How do we obtain consequent runtime instanciations? How do we connect design time and runtime artifacts? How do we deal with the scalability issue in the context of validation and verification tools? How do we handle time and space concerns wrt design time and runtime artifacts? How do we automatically or semi-automatically manage the validation and verification machine configuration?

**Model Transformation:** What verification techniques are meaningful for verifying model transformations? How do we analyse properties like confluence and termination for transformations which are composed from transformation units? How do we analyse correctness of model transformations wrt a transformation contract? How do we infer a transformation contract from a model transformation?

**Methods:** How do we integrate validation and verification in the overall development and modeling process? On the technical level of tool exchange? On the methodological level of using the right technique at the right time for the right task? When are techniques like animation, execution, symbolic evaluation, testing, simulation, proving or test case generation used efficiently during development (see Fig.8)? For which model and model transformation properties can they be employed?



Applicability of V&V techniques for stakefolder (client, developer) understanding

| | Client-Developer | Developer-Developer |
|---|---|---|
| Animation | + | |
| Testing | + | + |
| Execution | + | + |
| Simulation | + | + |
| Symbolic execution | (+) | + |
| Proving | (+) | + |
| Test case generation | | + |

+: applicable, (+): restricted applicable for domain-specific, explainable properties

**Figure 8** V&V Techniques and their Relationship to Stakeholders.

**Informal VS Formal VS Incomplete Modeling:** How do we leverage informal assumptions found in sketches for exploratory validation and verification? Are informal sketches close enough to validation and verification at all? What are appropriate relaxation mechanisms for different degrees of formality? How do we handle incomplete or partial models wrt validation and verification? How do we deactivate and activate model units? How do we handle the exploration of model properties and alternatives?

**Comparison and Benchmarking:** How do we compare existing validation and verification tools employed for modeling wrt functionality, coverage, scalability, expressiveness, executing system (i.e., for models at runtime)? Which criteria are appropriate for comparison? Can the broad and diverse spectrum of validation and verification machines (like B, Coq, HOL/Isabelle, SAT, SMT, CSP solvers, Relational logic and enumerative techniques) be globally compared in a fair way at all?

**Properties:** How do we handle model and model transformation properties relevant in validation and verification like consistency, reachability, dependence, minimality, conformance, safety, liveness, deadlock freeness, termination, confluence, correctness? How do we search for such properties in models and model transformations? What are the benefits and tradeoffs between expressing these properties on more abstract modeling levels in contrast to expressing them on more concrete levels? How do we find the right techniques for uncovering static and dynamic model properties? Which techniques are appropriate for uncovering static modeling language inherent properties, which for static model-specific properties? Which techniques are appropriate for uncovering dynamic generic properties, which for dynamic model-specific properties? Which high-level features are needed in the property description language in order to query and to determine modeling level concepts?

## Participants

- Colin Atkinson
Universität Mannheim, DE
- Tony Clark
Middlesex University, GB
- Benoit Combemale
IRISA – Rennes, FR
- Larry Constantine
University of Madeira – Funchal, PT
- Lukas Diekmann
King's College London, GB & University of Konstanz, DE
- Catherine Dubois
ENSIIE – Evry, FR
- Michalis Famelis
University of Toronto, CA
- Robert B. France
Colorado State University, US
- Martin Glinz
Universität Zürich, CH
- Martin Gogolla
Universität Bremen, DE
- Lars Hamann
Universität Bremen, DE
- Øystein Haugen
SINTEF – Oslo, NO
- Gabor Karsai
Vanderbilt University, US
- Steven Kelly
MetaCase – Jyväskylä, FI
- Thomas Kühne
Victoria Univ. – Wellington, NZ
- Vinay Kulkarni
Tata Consultancy Services – Pune, IN
- Stephen J. Mellor
Evesham, UK
- Pieter J. Mosterman
The MathWorks Inc. – Natick, US
- Pierre-Alain Muller
University of Mulhouse, FR
- Leonel Domingos Telo Nóbrega
Univ. of Madeira – Funchal, PT
- Ileana Ober
Paul Sabatier University – Toulouse, FR
- Marian Petre
The Open University – Milton Keynes, GB
- Dorina C. Petriu
Carleton Univ. – Ottawa, CA
- Louis Rose
University of York, GB
- Bernhard Rumpe
RWTH Aachen, DE
- Martina Seidl
University of Linz, AT
- Bran V. Selic
Malina Software Corp. – Nepean, CA
- Perdita Stevens
University of Edinburgh, GB
- Laurence Tratt
King's College London, GB
- André van der Hoek
Univ. of California – Irvine, US
- Markus Völter
Völter Ingenieurbüro – Heidenheim, DE
- Jon Whittle
Lancaster University, UK
- Dustin Wüest
Universität Zürich, CH