Report from Dagstuhl Seminar 19481

Composing Model-Based Analysis Tools

Edited by

Francisco Durán[1], Robert Heinrich[2], Diego Pérez-Palacín[3], Carolyn L. Talcott[4], and Steffen Zschaler[5]

1 **University of Málaga, ES,** duran@lcc.uma.es
2 **KIT – Karlsruhe, DE,** robert.heinrich@kit.edu
3 **Linnaeus University – Växjö, SE,** diego.perez@lnu.se
4 **SRI – Menlo Park, US,** clt@csl.sri.com
5 **King's College London, GB,** steffen.zschaler@kcl.ac.uk

Abstract

This report documents the program and the outcomes of the Dagstuhl Seminar 19481 "Composing Model-Based Analysis Tools". The key objective of the seminar was to provide more flexibility in model-driven engineering by bringing together representatives from industry and researchers in the formal methods and software engineering communities to establishing the foundations for a common understanding on the modularity and composition of modeling languages and model-based analyses.

Seminar November 24–29, 2019 – http://www.dagstuhl.de/19481
2012 ACM Subject Classification General and reference → General literature, General and reference
Keywords and phrases Modelling, Simulation, Semantics, Formal Methods, Software Engineering
Digital Object Identifier 10.4230/DagRep.9.11.97

1 Executive Summary

Francisco Durán
Robert Heinrich
Diego Pérez-Palacín
Carolyn L. Talcott
Steffen Zschaler

Quality properties like performance and dependability are key for today's systems. Several techniques have been developed to effectively model quality properties, which allow analyzing these systems. However, the very different nature of these properties has led to the use of different techniques and mostly independent tools. In addition, different tools and techniques can be used for modelling quality depending on the size and complexity of the systems and the available details. For example, for modeling dependability techniques like Fault Trees, Markov Chains, and Reliability Block Diagrams are available. Similarly, a range of analysis techniques are available, including simulations, using numerical, analytical or graphical techniques, and analytical methods.

Composing Model-Based Analysis Tools, *Dagstuhl Reports*, Vol. 9, Issue 11, pp. 97–116
Editors: Francisco Durán, Robert Heinrich, Diego Pérez-Palacín, Carolyn L. Talcott, and Steffen Zschaler

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

Although it is worth exploring other techniques and methodologies, model-driven engineering (MDE) seems a promising technique to efficiently design and reason about behavior and quality of systems in various domains. Indeed, it has been very successfully applied to improve the efficiency of software development and analysis in various domains.

Moreover, recent innovations, like the Internet of Things, production automation, and cyber-physical systems, combine several domains such as software, electronics and mechanics. Consequently, also the analyses for each of these individual domains need to be combined to predictively analyze the overall behavior and quality. The composition of systems and their analyses is a challenging but unavoidable issue for today's complex systems. Existing MDE approaches to modeling and analysis are not sufficient to compose modular analyses combining domain-specific languages. First attempts towards composable modular models have been developed in recent years, attempting to compose, not only the structure of models and domain specific modeling languages (DSMLs), but also their dynamic aspects (behavior and semantics). These indeed may be good foundations for building composable modular analyses. However, much work remains ahead.

In this Dagstuhl Seminar, we target more flexibility in MDE by discussing how to modularize and compose models and analyses. This provokes questions from the theoretical computer science and formal methods community – for example, on validity, uncertainties, behavior and property protection/preservation/reflection, and termination of analyses. Traditionally, research on these topics is conducted in the formal methods community isolated from the MDE community. A key objective for bringing together representatives from industry and researchers in the formal methods and software engineering communities is to make progress towards establishing the foundations for a common understanding.

2 Table of Contents

3 Introduction

Quality properties like performance and dependability are key for today's systems. Ensuring these properties is a major concern for design engineers. Several techniques have been developed to effectively model and analyze characteristics like performance or failure of systems. However, the very different nature of these properties has led to the use of different techniques and mostly independent tools for their different aspects. For instance, while some of the properties (e.g., performance, reliability and availability) are quantitative, other ones (e.g., confidentiality and safety) are essentially qualitative.

Depending on the size and complexity of the system and the available details, there exists different techniques for modelling quality. For example, for modeling dependability techniques like Fault Trees [29], Markov Chains [11], and Reliability Block Diagrams [5] are available. Similarly, a range of techniques are available for dependability analysis, including simulations, using numerical, analytical or graphical techniques, and analytical methods. Several surveys have analyzed different aspects of the state of the art, including [17, 28, 7, 16, 25, 3, 19, 21, 1]. Although methods and procedures are not standardized for most industries and there are several open questions (exemplified in this report), known techniques both for the modelling and analysis are successfully used in cases such as defense, transportation, and space industries.

Where rigorous and precise methods are required, different formal methods have been used to provide mathematical reasoning, so that once the system's intended behavior is modelled, one can construct a proof that the given system satisfies its requirements. For dependability analysis, we can find proposals using Petri nets, model checking and higher-order logic theorem proving. See [1] for a recent survey on the use of formal methods for dependability modelling and analysis.

In model-driven engineering (MDE), models are created and applied to efficiently design and reason about behavior and quality of systems in various domains. For representing systems in the form of a model, a modeling language (for example defined through a metamodel) is required. Recent innovations like the Internet of Things, production automation and cyber-physical systems combine several domains such as software, electronics and mechanics. To successfully develop reliable systems for these contexts, analyses for the single domains need to be combined to estimate the overall behavior and quality. Usual simulation-based, analytical, or graph-based solvers can then be used to analyze models. An interesting case is the one of executable domain-specific modeling languages (xDSML), on which different techniques can be used for their analysis, including graph-based analyses, which has been extensively used in existing work [20].

The composition of systems and their analyses is a challenging but unavoidable issue for today's complex systems. However, existing approaches to modeling and analysis are not sufficient to compose modular analyses over xDSMLs. First attempts at composable modular models came up in recent years, attempting to compose, not only their structure (metamodels), but also their dynamic aspects (behavior) [8]. These indeed may be good foundations for building composable modular analyses. For example, a trace-based semantics of xDSMLs may lead to composable and decomposable traces, possibly inspired by existing notions such as trace slices or trace superposition.

Furthermore, since models are abstractions of reality, they are not a faithful representation of the system but they contain uncertainties [18, 30]. Identifying and handling these uncertainties is a challenge for the research community [10, 22] that is, at present, only partially addressed. The combination of models from different domains and usage perspectives

may exacerbate the effect of such uncertainties by creating, for instance, model inconsistencies, incoherence, mismatches in granularities of models, mismatches in the underlying assumptions made when creating the different models, etc. The study of the existence, quantification and management of the new uncertainties created during the combination of models is an unaddressed task that should be tackled to trust the results of the subsequent model analysis.

Our aim is to get more flexibility in MDE by discussing how to modularize and compose models and analyses based on modular models. Of course, composing modular analyses provokes questions from theoretical computer science and formal methods community – for example, on validity, uncertainties, behavior and property protection/preservation/reflection, and termination of analyses. Traditionally, research on these topics is mostly conducted in the formal methods community isolated from the MDE community. Bringing together representatives from industry and researchers in the formal methods and software engineering communities may lead to the establishment of the foundations for a common understanding.

3.1 Main Purpose of the Seminar

MDE has proven to be able to provide a good set of tools and techniques for the development of models and tools for the manipulation of these models. In the context of performance and dependability analysis, tools like Palladio, Modellica or AADL, are good examples of the possibilities. Indeed, a tool like Palladio already provides a modelling language for the analysis of performance, reliability and maintainability of systems [23]. However, it is a monolithic tool, making extension to new properties challenging. Thus, its internal structure eroded over time [27]. Furthermore, there is no way to verify the non-interference between the analyses provided. On the other hand, we have witnessed interesting advances in some of these issues, for example in the fields of graph rewriting, algebraic specification or tree automata. The seminar was organized with the believe that sharing specific problems and advances in some of these fields might lead to fruitful discussions, and possibly to new approaches and alternative views so some of these problems may be tackled.

With this goal in mind, we envision an environment in which xDSMLs for the different quality properties are independently provided, and where one can pick up the desired ones at will. In addition to being able to perform such a composition of models and analysis tools, the combined analysis performed by the composed system would allow us to analyze the tradeoffs between different properties (e.g., performance vs. security). Furthermore, we would like to be able to share the analysis effort between computation resources as much as possible. This led us to interesting questions relevant for research and industry like:

- Given the great costs of such analyses, if two different properties are analyzed using a graph-based simulation, how to combine the simulations so that the performance of the tool can be dramatically improved?
- Can analytical analyses performed on Markov chains or Petri nets be similarly composed?
- What are the limits of modular and composable model analysis?
- Is it possible to identify the uncertainties that spring from a composition of models? Is it possible to quantify the criticality of such uncertainties or inconsistencies? Is it possible to handle these possible uncertainties with robust methods so that the analysis still produces trustworthy results?

Composition of analyses over xDSMLs for different domains is an ambitious goal. In the past, it was assumed to be unattainable as models and analyses for different properties were considered to be too different. Recent research advances, however, lead to the conclusion that

commonalities can be identified, based on which foundational concepts can be elaborated as argued in detail by the following bullet points.

- Advances in behavior-parametrized modular specifications [9] together with advanced support for xDSMLs with graph-based operational semantics give reason for optimism in enabling the specification and composition of modular analyses.
- Research on formalization, measures, and metrics of single quality properties [2, 4, 12] resulted in much deeper and clearer understanding of what the corresponding analyses depend on, which is foundation for modularizing the analyses.
- Research on the analysis of mutual quality impact (e.g., performance [14] and maintainability [24, 13]) between different domains provides starting points for composing modular analyses.
- A first reference architecture for metamodels to tailor quality modeling and analysis [15] is starting point to more generic investigation on the topic.

3.2 Structure and Organization of the Seminar

The organization of a successful seminar poses a number of challenges, which are possibly consequence of:

(1) the selection of participants and their availability to participate,
(2) the attractiveness of the discussions and the topics around which these are going to happen along five days, and
(3) the involvement of the participants in such discussions and in the generation of summaries and documentation.

In order to organize the discussions during the seminar around common interests and challenges, before the seminar, participants were asked to share a short statement on their main interests (related to the topic under discussion). These challenges were analyzed and classified, and served as a starting point for the organization of the discussions around specific topics. The analysis of these challenges and how it led to the different working groups is explained in Section 4.

Although some of the participants knew each other before the seminar, we wanted them to introduce themselves to facilitate the interactions as soon as possible. To avoid spending too much time on introductions – 39 researchers participated in the seminar – we collected one slide from each participant with their core data and set up the presentation so that a new slide was on every two minutes. The presentations covered research interests, background, current research of the participant and topics to discuss at the seminar. Participants were asked to prepare their presentation slides before the seminar.

As discussed in Section 4, the analysis of the challenges, and the posterior discussion led to break-up groups in which the identified topics were discussed. The agenda for the week can be seen in Figure 1. The breakout groups were created to discuss these topics in smaller groups and create first results and plans for follow-up activities during the seminar. The breakout groups were suggested to produce paper projects and follow-up activities like workshop proposals or other community activities. Each breakout group started writing papers during the seminar using Overleaf, Google doc or other collaborative tools.

To share the discussions in the break-up groups with the rest of the participants, and to get feedback from them, there were presentations from each of the groups in the main room on the advances on their discussions. In addition to the consequent discussions that followed these presentations, and given the tight relationships between the different discussions,

<table>
<tr><th>Timeslot</th><th>Monday</th><th>Tuesday</th><th>Wednesday</th><th>Thursday</th><th>Friday</th></tr>
<tr><td>09:00-09:30</td><td>Opening and seminar objectives</td><td>Wrap up and planning of the day</td><td>Wrap up and planning of the day (and reorganizations of groups)</td><td>Presentation of Wednesday results, wrap up and planning of the day (and reorganizations of groups)</td><td>Seminar summary and planning for next steps</td></tr>
<tr><td>09:30-11:00</td><td>Introduction of participants</td><td>Breakout groups</td><td>Breakout groups</td><td>Cross-cut discussions</td><td>Seminar summary and planning for next steps</td></tr>
<tr><td>11:00-11:15</td><td colspan="5">Coffee break</td></tr>
<tr><td>11:15-12:15</td><td>Extended talks</td><td>Breakout groups</td><td>Breakout groups</td><td>Breakout groups</td><td>Seminar summary and planning for next steps / closing</td></tr>
<tr><td>12:15-13:30</td><td colspan="4">Lunch</td><td></td></tr>
<tr><td>13:30-13:45</td><td rowspan="2">Extended talks (13:30-14:00)

Introduction of participants (14:00-15:00)</td><td>Short wrap up of breakout groups</td><td rowspan="2">Excursion</td><td>Short wrap up of breakout groups</td><td rowspan="8">Departure</td></tr>
<tr><td>13:45-15:00</td><td>Breakout groups</td><td>Breakout groups</td></tr>
<tr><td>15:00-15:15</td><td colspan="4">Coffee break</td></tr>
<tr><td>15:15-17:00</td><td>Summary of submitted challenges (15:15-15:45)

Identification of topics for breakout groups and allocation of participants to groups (15:45-17:00)</td><td>Breakout groups</td><td>Excursion</td><td>Breakout groups</td></tr>
<tr><td>17:00-17:15</td><td colspan="4">Coffee break</td></tr>
<tr><td>17:15-18:00</td><td>Breakout groups</td><td>Presentation of results of the day</td><td>Excursion</td><td>Breakout groups</td></tr>
<tr><td>18:00-19:15</td><td colspan="4">Dinner</td></tr>
<tr><td>19:30-</td><td colspan="3">Come together</td><td>Tool demos</td></tr>
</table>

Figure 1 Seminar's schedule.

cross-cutting topics were identified in Thursday's first session (after a list of topics prepared by each group the previous day). These cross-cutting topics were discussed in the following session in groups in which there were representatives of each of the main break-up groups. These participants share these discussions back in their groups in the following break-up sessions.

Since the development of tools was present in many of the discussions, and many of the participants have themselves developed tools for the modelling and analysis of systems, a tool demo session was organized. This session was scheduled on the evening of the Thursday, and participants were asked to show their interest in delivering demos. The following tools were demonstrated:

- Shadow Models – incremental model transformation and lifting of error messages back to the original model, by Markus Voelter.
- FASTEN – a stack of DSLs and analyses for (formal) system level specification and assurance (SMV, tabular specification, contract-based design, UI-modeling, requirements specification, GSN, STPA), by Daniel Ratiu.
- MBEDDR – code-level analyses based on CBMC, Model-Driven Code Checking DSLs and analysis based on Spin, feature models consistency based on Sat4J, by Daniel Ratiu.
- ASMETA – a toolset for the Abstract State Machines (model specification, animation, simulation, verification, reviewing, ect), by Patrizia Scandurra and Elvinia Riccobene.
- The GEMOC Studio – a Language Workbench providing generic components through Eclipse technologies for the development, integration, and use of heterogeneous executable modeling languages, by Erwan Bousse, Steffen Zschaler and Benoit Combemale.
- GROOVE – a graph transformation tool for flexibly modelling any system whose states have a graph-like structure, and subsequently exploring and model checking the ensuing state space, by Arend Rensink.

- Palladio: A software architecture simulator for performance and reliability, by Robert Heinrich.
- Timed Rebeca Model Checking Tool – Afra, a tool for model checking and debugging a timed actor-based language, by Marjan Sirjani.
- Horus, for Business Process / Enterprise Modelling, by Arthur Vetter.
- AToMPM and Modelverse, by Hans Vangheluwe.

3.3 Viewpoint Talks

Although most participants might have common interests, each of us was approaching the general problem from different perspectives and using different techniques. Any of us could have told his/her story, but we did not want to spend the whole week with talks from the participants. We wanted to use the time to discuss and find new synergies between participants. We decided however to start sharing the focus with presentations from a selection of participants. From the pre-seminar challenge statements, we picked three of them that were providing three alternative views:

- Arthur Vetter, from KIT (Karlsruher Institut für Technologie), Germany, discussed on the evolution of systems and their analysis,
- Kenneth Johnson, from the Auckland University of Technology, New Zealand, focused on large scale complex systems, and shared his experience in the verification of large scale complex systems, with a cyberphysical perspective, and
- Fiona Polack, from Keele University, UK, presented an interesting discussion about *purpose*.

They were asked to deliver 15-minutes talks on their points of view. Moreover, since we were assuming that each of us works on different projects, for which we use different techniques, formalisms, and tools, they were specifically asked to avoid in their presentations details on these specifics, and try to provide a broader view from their specific perspectives. The goal was not about surveying on what can be done or how can be solved, that was the purpose of the rest of the week, we wanted to have a general discussion on what the problem is and what they would like to get as a result of the seminar.

4 Overview of Challenge Statements

Before the start of the seminar, all invited participants were asked to submit brief challenge statements, summarising what they felt were the key challenges in the area of composing model-based analysis tools. Overall, we received 27 such challenge statements. In preparation of the seminar, recurring themes from these challenge statements were clustered using a mindmapping technology. The resulting clusters can be seen in Figure 2. Through this exercise, we identified the following five top-level challenges:

1. *Composition of formalisms / languages.* Key sub-challenges here were mentioned as
   - Composition of semantics
   - Composition of different formalisms
   - Composition of analysis method vs composition of analysis results
   - Composition in space and time (variants vs versions)

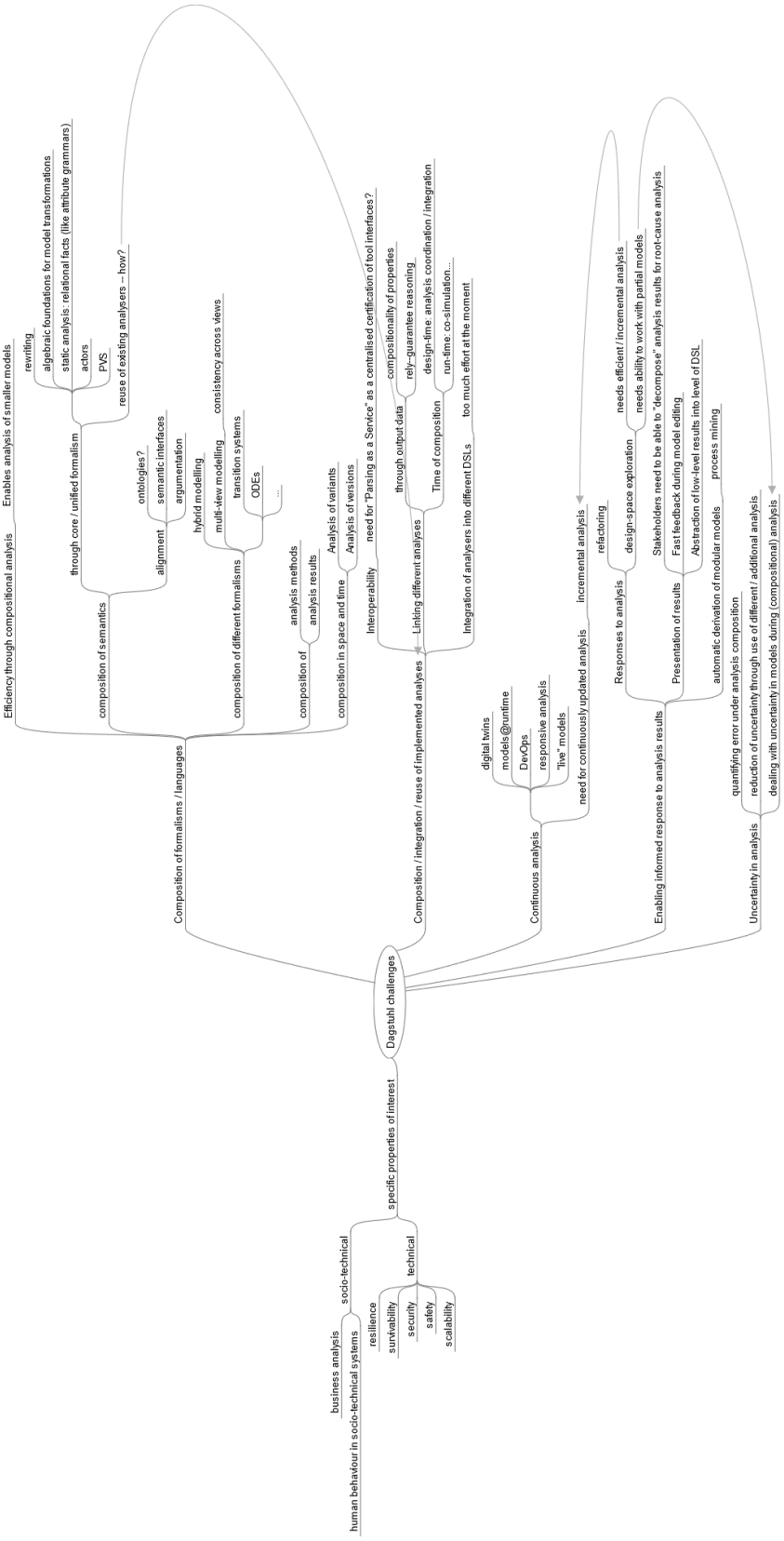

Figure 2 Mindmap of challenge clusters.

2. *Composition / integration / reuse of implemented analyses / tools.* Key sub-challenges here were mentioned as
   - Interoperability
   - Linking different analysers
   - Integration of analysers into different DSLs
3. *Continuous analysis.* Key sub-challenges here were mentioned as
   - Uses in different areas such as digital twins, models@runtime, DevOps, and responsive analysis
   - Achieving incremental analysis
4. *Enabling informed response to analysis results.* Key sub-challenges here were mentioned as
   - Presentation of results
   - Automatically obtaining modular models for efficient analysis
5. *Uncertainty in analysis.* Key sub-challenges here were mentioned as
   - Quantifying error under analysis composition
   - Combining different analyses to reduce overall uncertainty
   - Handling uncertainty / incompleteness in underlying models

To ensure we had correctly interpreted participants' challenge statements and to give an opportunity for all participants to contribute to the list of challenges, we undertook a separate clustering activity with all participants on the first day of the seminar. Here, participants were asked to write their key challenges on a post-it and to then physically cluster them on a pinboard. The resulting clustering can be seen in Figure 3. These clusters were very close to our original clustering and were used as the basis for the formation of breakout groups, with each group discussing one of the challenges in more depth. The following sections provide brief summaries of results provided by each group.

5 On the Relation between Language Composition and Analysis Composition

This group consisted of Carolyn Talcott, Ralf Reussner, Bernhard Rumpe, Hans Vangheluwe, Patrizia Scandurra, Kyungmin Bae, Séverine Sentilles, Narges Khakpour, Mark Hills, and Sofia Ananieva.

The group *On the Relation between Language Composition and Analysis Composition* was concerned with understanding the forms of compositions, especially the compositionality of analyses in relation to composition of underlying models and composition of the system. They also identified the different classes of compositionality and specific conditions of composition.

The group define analysis as answering questions about properties of a system under study. Next to the model of the system, they therefore also need a model of the context (actually the assumptions on the context) and a model of the property we want to analyse. For models, we follow the definition of Stachowiak [26] (i.e., abstraction, isomorphism, and pragmatics). Models can relate to each other via (a) *abstraction / refinement*, (b) *view projection / view merge* and (c) *architectural composition / decomposition*. They do not distinguish between modelling artifacts and models.

The group realised that considering the context of analysis is important for compositionality. Consequently, the definition of *what context is* depends on the kind of analysis. For structural (syntactic) analysis, the context is given by the meta-model / language definition. For behavioural analysis, the context is given through a semantic definition for the

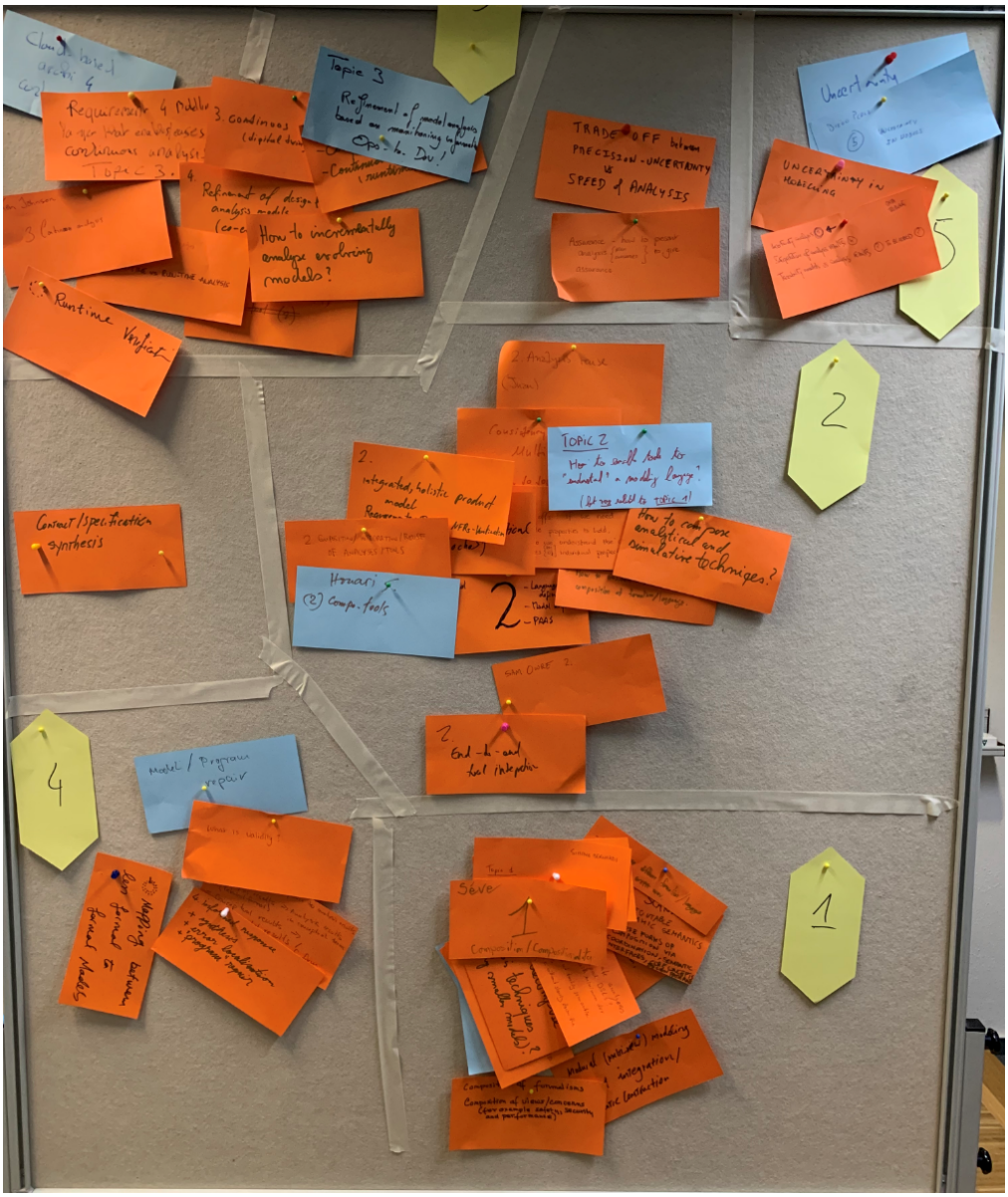

Figure 3 Clustering of participants cards.

system model and the specification of the system and its semantics. For the analysis of extra-functional properties (e.g., performance, reliability or security) the context is given through the model of usage profile, the model of the execution environment, and a model of the external services. If models of different semantic domains are involved, lifting of the analysis results back to the system model under manipulation is harder. Therefore, we need to understand the relationship between the involved contexts for composing analyses.

We require workflows to model the relation between activities using, changing and creating models. A workflow is a partially ordered set of basic activities (either human or computer based) or composed activities which take modelling artifacts as input and produce modelling artifacts as output. An activity can be refined into a workflow. The orchestrated execution of activities forms a workflow. Workflows can lead to variability in space (variants) and time (versions) for modelling artifacts.

Finally, we conceived three classes of composition:
1. System Model Composition (*white-box* composition)
2. Result Composition (*black-box* composition)
3. Analysis Composition (*grey-box* composition)

In analysis composition, we orchestrate the steps of the two analyses to be composed. Mathematically, the three cases are described as follows:
Let A be Analysis,
A_i be Analysis i,
A_i^j be the step j of Analysis i,
S_i be System model i,
C_i be Context model i,
Q_i be Question model i,
$\times$ be Composition operator, and
K be Orchestration model

System Model composition (white-box composition)
$A(S_1 \times S_2, C_1 \times C_2, Q_1 \times Q_2)$

Result composition (black-box composition)
$A_1(S_1, C_1, Q_1) \times A_2(S_2, C_2, Q_2)$

Analysis composition (grey-box composition)
$K((A_1^1(S_1, C_1, Q_1), A_1^2(S_1, C_1, Q_1), \ldots, A_2^1(S_2, C_2, Q_2), \ldots), C_1 \times C_2, Q_1 \times Q_2)$

6 Orchestration of Analysis Tools

This group consisted of Erwan Bousse, Robert Heinrich, Sandro Koch, Daniel Ratiu, Elvinia Riccobene, Markus Voelter, Marjan Sirjani, and Sam Owre.

Sophisticated engineering tools, often based on the principles of Model Driven Engineering (MDE) and Software Language Engineering (SLE), are becoming more and more ubiquitous– i.e., more and more disciplines rely on such tools. These tools become all the more valuable if they provide deep insights into the correctness or fitness for purpose of the created models. At the same time there is a community of analysis tool builders who distill mathematical and logic experience into analysis tools that rely on formalisms such as Satisfiability Modulo Theories (SMT) formulae, transition systems or discrete events. Many of these analyses can be used beneficially in the aforementioned engineering tools if they are suitable integrated. In practice this usually means that user-facing models must be translated to the input formalism of the analysis tool, and the results must be lifted back to the domain level. In addition, there are many use cases where multiple existing analyses must be orchestrated to deliver value in the context of the engineering tool.

The group's vision to tackle these challenges is to enable the definition of an architecture for the integration and or composition of existing analysis tools with a given domain-specific language. At the core of such an architecture is the orchestrator, a component both responsible for interacting with analysis tools, and for interacting with the domain expert willing to perform analyses. This orchestrator follows some orchestration logic that defines which analysis tools should be used for a given analysis task, in which order these tools should be used, and how the analysis results they produced should be combined or exchanged. The orchestrator relies on a set of tool drivers that each defines how to make use of a specific

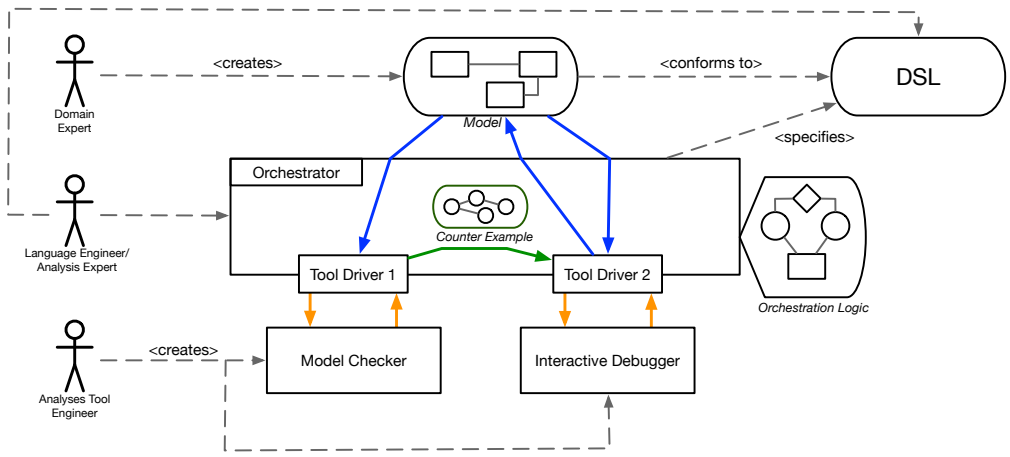

Figure 4 Vision applied to the same example.

analysis tool, including how to translate the domain-specific model into a valid input for the tool, how to lift back the analysis result into a form that makes sense at the abstraction level of the domain model, as well as the protocol to exchange messages and information with the tool. In addition, for such architectures to work, a set of requirements must be satisfied by the considered analysis tools: both the input and output languages of the tool but be explicitly defined (which excludes loosely structured output formats), and the protocol to use these tools must be explicitly defined and exploitable by the orchestrator. Figure 4 illustrates this overall vision with a simple case where a model checker is used to analyse a model, and where the counter-example produced by this model checker is injected in an interactive debugger for further investigation. In summary, the expected outcomes of such a research work are:

- A metamodel allowing one to define:
  - The overall architecture of a particular tool integration and composition scenario,
  - Tools drivers that each knows how to integrate the DSL with an analysis tool,
  - The orchestration logic (eg. the model is model checked when asked by the user, and the produced counter example (if any) is sent to an interactive debugger for investigation).
- A set of requirements for analysis tools, such as :
  - A tool must have a well-defined and explicit input language,
  - A tool must have a well-defined and explicit output language,
  - A tool must have a well-defined and explicit protocol.
- A set of case studies that demonstrates the relevance and applicability of the abstractions defined in this paper.

7 Continuous Model Analysis (CMA)

This group consisted of Christel Baier, Olivier Barais, Kenneth Johnson, Dániel Varró, Arthur Vetter, Marc Zeller.

The current pressure to ensure that software systems remain available, dependable, preferment at all times despite changes in their operating context, the addition or evolution of features, the increasing integration with other systems has led to the implementation of so-called continuous deployment techniques (i.e. DevOps) and self-healing mechanisms. Behind such mechanisms lies the need to continuously analyse a system against a number of properties. The combined use of a number of model-based analysis tools on abstract representations (models) of a running system has therefore become common.

Several challenges arise then:

1. How to understand and reason about this composition of analysis tools?
2. How to orchestrate these analyses?
3. How to minimize the analyses to be performed each time there is a change in the system specification, in the context of the system's execution or on the system itself?
4. How to validate compositionality of CMAs?
5. Could we use the same analysis tool at runtime and at design time?
6. What happens when we have uncertain knowledge of the system? What is the minimal set of information to carry out reusable analysis?

The fields of application are very diverse:

- Incremental verification of system component models at runtime
- DevOps pipeline
- Safety-critical systems development
- Self-adaptive systems

The working group sought to define a general conceptual framework for continuous model analysis. They keep to simple mathematical concepts to describe behaviours and relationships between modules. Notions of timing and change are important.

A key challenge is to formulate and validate compositionality of CMAs

CMA modules:

- System meta-model (the set S)
- System meta-model Requirements (the set Q)
- Models (Models of systems, requirements) (the set M) and their properties (the set P)
- Analysis tasks (the set A)
- Results output from the analysis (the set V)
- Monitoring and notifications from the actual system + requirements

Mathematically,

- $\alpha : S \to M$ modelling process from system meta-models to their semantic meaning. Will be input for analysis
- $\beta : Q \to P$ modelling process from system meta-model requirements to their properties. Will be input for analysis
- α and β are used in many ways to form analysis tasks in A. Let the process be modelled by $\gamma : [S \to M] \times [Q \to P] \to A$. This may simply be $a = ((s), (q))$ where verification analysis task a is comprised of a model and property.
- The analysis is modelled by $\rho : A \to V$ which takes as input an analysis task and computes some sort of result value(s) in V

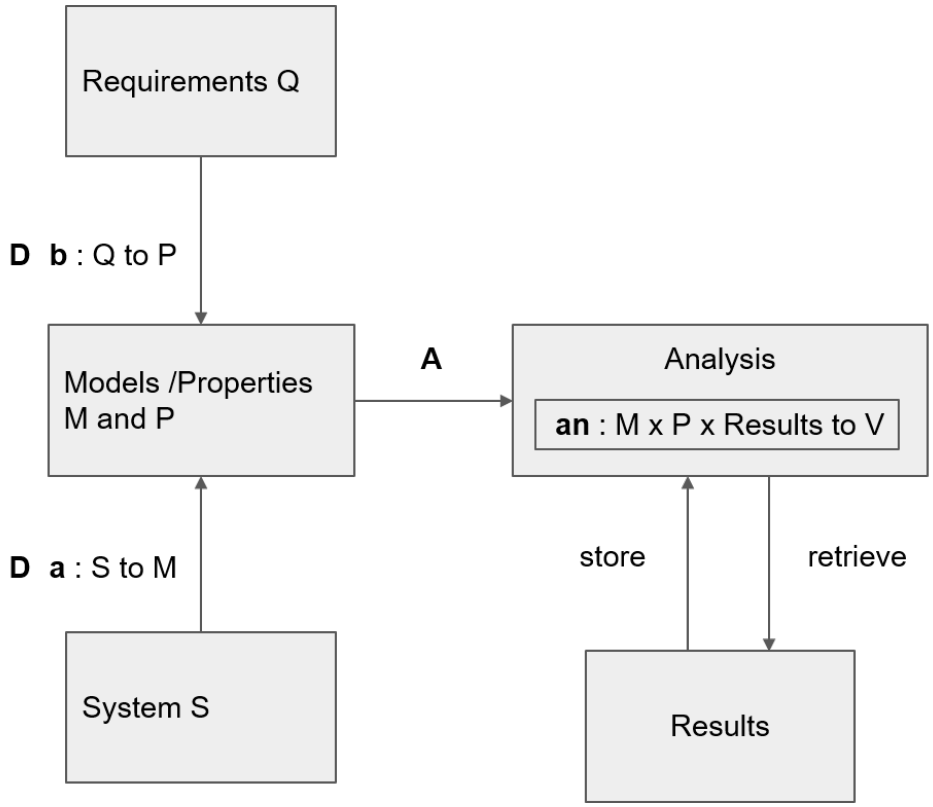

Figure 5 General conceptual framework for continuous model analysis (CMA).

- Note that we need to extend ρ to take as input all existing analysis tasks and results. We can extend this by $\rho : [A \to V] \times A \to V$

A key technical challenge is to minimise the amount of computation that needs to be performed.

8 Creating Value from Analysis Results

This group consisted of Steffen Zschaler, Houari Sahraoui, Esther Guerra, Martin Gogolla, Francisco Durán, and Juan de Lara.

The group focused on activities taking place once analysis has been completed: how are analysis results turned into value for the users of the analysis? Discussions were driven by diverse example cases of specific analyses, ranging from static analysis of programs written by novice programmers via business-process deadlock analysis to analysis of object churn. As a result of discussing these different analyses, the group identified a conceptual model of three pathways for result usage:

1. *Result presentation.* Analysis results can first be used to help developers or domain experts explore the system model further—for example to identify the root cause of a problem or to better understand a scientific hypothesis. Challenges the group identified in this pathway include:
   a. Lifting low-level analysis results back up to the domain level so that they can be understood by domain stakeholders.

   b. Selecting what analysis results are most important / useful to present in a given situation–this is closely connected to the original purpose for which the analysis was undertaken.
   c. Enabling users to drill down, possibly interactively, into the analysis results–for example to undertake root-cause analysis.
   d. Interpretation of the results by the end user–this may develop over time as users learn to correlate result presentation and their own understanding of the system and its properties.
2. *Result exploitation.* Analysis results can secondly be exploited to directly improve systems or their specifications / models. Examples of this include model / program repair, refactoring, or refinement. The changes to a system or its specification, in response to some analysis, can be done automatically (*e.g.,* search-based software engineering) as well as manually. Challenges identified in this pathway include:
   a. How far can this be automated for different properties of interest?
   b. Is there a generic automatic mechanism or does each property require its own mechanism?
   c. Can we learn automated exploitation mechanisms by observing how expert domain users respond to different types of analysis results?
   d. Is it possible to undertake repair or similar in relation to multiple properties of interest at the same time (*i.e.,* can repair be composed)?
3. *Analysis improvement.* Analysis results can be used to improve the analysis itself. For example, by asking it to focus on a particular aspect of the system model in more detail or by learning an analysis from a set of expert-provided examples of inputs and expected results. Challenges identified in this pathway include:
   a. How to enable users to understand the analysis results and provide suitable feedback to the learning algorithm (see also Pathway 2)?
   b. How to model such feedback so that it can be effectively used for improving the analysis?
   c. How to automate the learning process; to which extent is this even possible?

Overarching these pathways, there is a challenge of how to choose properties, analysis pathways, and combinations thereof to form an overall argument of fitness-for-purpose of the system as a whole. Goal-Question-Metric, safety cases, goal-oriented modelling (e.g., [6]) appear to have building blocks for answering this challenge, but as far as the group members were aware there is currently no integrated approach.

The group then proceeded to identifying similarities and differences between the example cases and how they covered each of the three pathways. This resulted in a detailed feature model, which can serve as the starting point of a systematic survey of analysis techniques, planned as one of the next steps to be undertaken by the group.

9 Composition of Models and Analysis affect Uncertainty

This group consisted of Simona Bernardi, Michalis Famelis, Jean-Marc Jézéquel, Raffaela Mirandola, Diego Perez-Palacin, Fiona Polack, and Catia Trubiani.

This group discussed the modelling and management of uncertainty, beginning by revisiting the various definitions and taxonomies of uncertainty in the literature. Following that, they discussed how uncertainty is localized in modelling and analysis, identifying four phases: model Definition, model Construction, QoS Analysis, and Validation.

To exercise the existence and location of uncertainties in the four phases, they constructed a simple example consisting of a set of models for a file sharing system based on a Peer-To-Peer protocol (PtPp). The group elicited uncertainty in the different modeling views of the protocol (state, class, deployment, and object diagrams, performance models) and discussed the effect of composition to the various uncertainties. To better understand this, they created a model (flow chart) of the process of development and performance analysis of PtPp to get awareness of the occurrence of uncertainty over time.

After that, the group explored, using a few scenarios, how uncertainty flows through the process model, also speculating that this kind of analysis could lead to a re-conceptualization of DevOps as an iterative approach for the reduction of some types of uncertainty.

In the context of the seminar at large, the group realized that there is a need for more popularisation in our community of existing theories, taxonomies, nomenclature about uncertainty.

The group agreed that more research is needed in deepening the understanding of the occurrence and evolution of uncertainty in the creation of systems, especially by exploring uncertainties in different kinds of quality analysis such as reliability, availability, and security. Finally, they planned joint publications and further academic events, starting with the submission of a Vision paper and a Workshop proposal to the MODELS 2020 conference, respectively.

10 Conclusions and Next Steps

This report summarized the structure, organisation and outcome of the Dagstuhl seminar 19481. We reported about discussions, group work and results of the seminar. Before the seminar, participants were asked to share a short statement on their main interests from which we identified topics for breakout groups. The breakout groups were created to discuss these topics in smaller groups and create first results and plans for follow-up activities during the seminar. We had five breakout groups that worked on the topics relation between language composition and analysis composition, orchestration of analysis tools, continuous model analysis, creating value from analysis results, and composition of models and analysis affect uncertainty, respectively.

The breakout groups individually produced plans for paper projects and follow-up activities like workshop proposals or other community activities. As a joint result of the seminar we agreed to produce an edited book summarising the discussions at the seminar, bringing together the thinking of the community, and demonstrating, through case studies, some of the important challenges and exemplary solutions in the field.

References

1 Waqar Ahmad, Osman Hasan, and Sofiène Tahar. Formal dependability modeling and analysis: A survey. In Michael Kohlhase, Moa Johansson, Bruce R. Miller, Leonardo de Moura, and Frank Wm. Tompa, editors, *9th International Conference Intelligent Computer Mathematics, CICM*, volume 9791 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2016.

2 Steffen Becker, Lucia Happe, Raffaela Mirandola, and Catia Trubiani. Towards a methodology driven by relationships of quality attributes for QoS-based analysis. In Seetharami Seelam, Petr Tuma, Giuliano Casale, Tony Field, and José Nelson Amaral, editors, *ACM/SPEC International Conference on Performance Engineering, ICPE*, pages 311–314. ACM, 2013.

3 Simona Bernardi, José Merseguer, and Dorina C. Petriu. Dependability modeling and analysis of software systems specified with UML. *ACM Comput. Surv.*, 45(1):2:1–2:48, 2012.

4 F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner. Architecture-based reliability prediction with the palladio component model. *IEEE Transactions on Software Engineering*, 38(6):1319–1339, 2012.

5 Marko Čepin. *Reliability Block Diagram*, pages 119–123. Springer, 2011.

6 Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. The Kluwer international series in software engineering. Kluwer Academic Publishers Group, Dordrecht, Netherlands, 1999.

7 Liliana Dobrica and Eila Niemelä. A survey on software architecture analysis methods. *IEEE Trans. Software Eng.*, 28(7):638–653, 2002.

8 Francisco Durán, Antonio Moreno-Delgado, Fernando Orejas, and Steffen Zschaler. Amalgamation of domain specific languages with behaviour. *J. Log. Algebr. Meth. Program.*, 86(1):208–235, 2017.

9 Francisco Durán, Fernando Orejas, and Steffen Zschaler. Behaviour protection in modular rule-based system specifications. In Narciso Martí-Oliet and Miguel Palomino, editors, *Recent Trends in Algebraic Development Techniques, 21st International Workshop, WADT, Revised Selected Papers*, volume 7841 of *Lecture Notes in Computer Science*, pages 24–49. Springer, 2013.

10 Naeem Esfahani and Sam Malek. Uncertainty in self-adaptive software systems. In Rogério de Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw, editors, *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, volume 7475 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2013.

11 W. Gilks. *Markov chain Monte Carlo*. 2005.

12 Robert Heinrich. *Aligning Business Processes and Information Systems – New Approaches to Continuous Quality Engineering*. Springer, 2014.

13 Robert Heinrich, Sandro Koch, Suhyun Cha, Kiana Busch, Ralf Reussner, and Birgit Vogel-Heuser. Architecture-based change impact analysis in cross-disciplinary automated production systems. *Journal of Systems and Software*, 146:167 – 185, 2018.

14 Robert Heinrich, Philipp Merkle, Jörg Henss, and Barbara Paech. Integrating business process simulation and information system simulation for performance prediction. *Software and Systems Modeling*, 16(1):257–277, 2017.

15 Robert Heinrich, Misha Strittmatter, and Ralf Heinrich Reussner. A layered reference architecture for metamodels to tailor quality modeling and analysis. *IEEE Transactions on Software Engineering*, 2019.

16 Anne Immonen and Eila Niemelä. Survey of reliability and availability prediction methods from the viewpoint of software architecture. *Software and Systems Modeling*, 7(1):49–65, 2008.

17 Allen Johnson and Miroslaw Malek. Survey of software tools for evaluating reliability, availability, and serviceability. *ACM Comput. Surv.*, 20(4):227–269, 1988.

18 M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society*, (63):425—464, 2001.

19 S. Shanmugavel L. Venkatesan and C. Subramaniam. A survey on modeling and enhancing reliability of wireless sensor network. *Wireless Sensor Network*, 5(3):41–51, 2013.

20 Antonio Moreno-Delgado, Francisco Durán, Steffen Zschaler, and Javier Troya. Modular dsls for flexible analysis: An e-motions reimplementation of palladio. In Jordi Cabot and Julia Rubin, editors, *Modelling Foundations and Applications – 10th European Conference,*

ECMFA 2014, volume 8569 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2014.

21 K. Pardeepkumar, P. Dahmani, and D. Narula. RAM analysis of some process industries: A critical literature review. *Int. J. Mech. Eng. & Rob. Res.*, 3(3), 2014.

22 Diego Perez-Palacin and Raffaela Mirandola. Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In Klaus-Dieter Lange, John Murphy, Walter Binder, and José Merseguer, editors, *ACM/SPEC International Conference on Performance Engineering, ICPE*, pages 3–14. ACM, 2014.

23 Ralf H. Reussner, Steffen Becker, Jens Happe, Robert Heinrich, Anne Koziolek, Heiko Koziolek, Max Kramer, and Klaus Krogmann. *Modeling and simulating software architectures: The Palladio approach*. MIT Press, 2016.

24 Kiana Rostami, Robert Heinrich, Axel Busch, and Ralf H. Reussner. Architecture-based change impact analysis in information systems and business processes. In *2017 IEEE International Conference on Software Architecture, ICSA*, pages 179–188. IEEE Computer Society, 2017.

25 S. Saraswat and G. Yadava. An overview on reliability, availability, maintainability and supportability (RAMS) engineering. 25(3):330–344, 2008.

26 Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.

27 Misha Strittmatter, Georg Hinkel, Michael Langhammer, Reiner Jung, and Robert Heinrich. Challenges in the evolution of metamodels: Smells and anti-patterns of a historically-grown metamodel. In Tanja Mayerhofer, Alfonso Pierantonio, Bernhard Schätz, and Dalila Tamzalit, editors, *Proceedings of the 10th Workshop on Models and Evolution*, volume 1706 of *CEUR Workshop Proceedings*, pages 30–39. CEUR-WS.org, 2016.

28 Kishor Trivedi and Manish Malhotra. Reliability and performability techniques and tools: A survey. In B. Walke and O. Spaniol, editors, *Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen*, pages 27–48. Springer, 1993.

29 W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook. Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission, 1981.

30 W. Walker, P. Harremoës, J. Rotmans, J. van der Sluijs, M. van Asselt, P. Janssen, and M. Krayer von Krauss. Defining uncertainty: A conceptual basis for uncertainty management in model-based decision support. *Integrated Assessment*, 4(1):5–17, 2003.

Participants

- Sofia Ananieva
FZI – Karlsruhe, DE
- Kyungmin Bae
Postech – Pohang, KR
- Christel Baier
TU Dresden, DE
- Olivier Barais
IRISA – University of Rennes, FR
- Simona Bernardi
University of Zaragoza, ES
- Erwan Bousse
University of Nantes, FR
- Benoit Combemale
University of Toulouse, FR
- Juan De Lara
Autonomous University of Madrid, ES
- Francisco Durán
University of Málaga, ES
- Michalis Famelis
Université de Montréal, CA
- Martin Gogolla
Universität Bremen, DE
- Esther Guerra
Autonomous University of Madrid, ES
- Robert Heinrich
KIT – Karlsruhe, DE
- Mark Hills
East Carolina University – Greenville, US
- Jean-Marc Jézéquel
IRISA – University of Rennes, FR
- Kenneth Johnson
Auckland University of Technology, NZ
- Narges Khakpour
Linnaeus University – Växjö, SE
- Sandro Koch
KIT – Karlsruhe, DE
- Raffaela Mirandola
Polytechnic University of Milan, IT
- Sam Owre
SRI – Menlo Park, US
- Diego Pérez-Palacín
Linnaeus University – Växjö, SE
- Fiona A. C. Polack
Keele University – Straffordshire, GB
- Daniel Ratiu
Siemens AG – München, DE
- Arend Rensink
University of Twente, NL
- Ralf H. Reussner
FZI – Karlsruhe, DE
- Elvinia Riccobene
University of Milan, IT
- Bernhard Rumpe
RWTH Aachen, DE
- Houari Sahraoui
Université de Montréal, CA
- Patrizia Scandurra
University of Bergamo – Dalmine, IT
- Séverine Sentilles
Mälardalen University – Västerås, SE
- Marjan Sirjani
Mälardalen University – Västerås, SE
- Carolyn L. Talcott
SRI – Menlo Park, US
- Catia Trubiani
Gran Sasso Science Institute, IT
- Hans Vangheluwe
University of Antwerp, BE
- Dániel Varró
McGill University – Montreal, CA
- Arthur Vetter
KIT – Karlsruher Institut für Technologie, DE
- Markus Völter
Völter Ingenieurbüro – Stuttgart, DE
- Marc Zeller
Siemens AG – München, DE
- Steffen Zschaler
King's College London, GB