10451 Executive Summary
Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems
— Dagstuhl Seminar —

Klaus Havelund[1], Martin Leucker[2], Martin Sachenbacher[3], Oleg Sokolsky[4] and Brian C. Williams[5]

[1] Jet Propulsion Laboratory/Caltech, US
Klaus.Havelund@jpl.nasa.gov
[2] Universität Lübeck, DE
leucker@isp.uni-luebeck.de
[3] TU München, DE
sachenba@in.tum.de
[4] University of Pennsylvania, US
sokolsky@cis.upenn.edu
[5] Massachusetts Institute of Technology, US
williams@mit.edu

Abstract. From November 7 to 12, 2010, the Dagstuhl Seminar 10451 "Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems" was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, 35 participants presented their current research and discussed ongoing work and open problems. This executive summary provides an overview of the goals and topics of the seminar.

Keywords. Runtime Verification, Model-based Diagnosis, Planning, Control, Autonomous Systems

1 Introduction

Over the last decade and a half, a phase transition has occurred in the level of processing power that is incorporated in embedded systems. Simultaneously, a phase transition occurred in the scale of problems that can now be solved by automated reasoning methods. This is leading to a revolution in a range of disciplines, including model-based planning and scheduling, verification, diagnosis, and hybrid systems control: each discipline is applying automated reasoning to increasingly complex real-world problems, and is incorporating real-time versions of their respective methods on board embedded systems. These are employed in order to elevate the level at which the embedded system is commanded, to verify correctness of system behavior at runtime, to improve the reconfigurability of the system, and to automatically recover from failure. The future trend is to connect these computationally intensive systems into vast, networked embedded

Dagstuhl Seminar Proceedings 10451
Runtime Verification, Diagnosis, Planning and Control for Autonomous Systems
http://drops.dagstuhl.de/opus/volltexte/2011/2947

systems, such as nation-wide earth observing systems, coastal cabled observatories, or smart power grids.

The objective of this seminar was to catalyze a new field of model-based autonomous, embedded and robotic systems, with the salient characteristic that these devices incorporate a significant level of the above-mentioned online reasoning, based on a system model. A common vision is emerging of systems that combine varied forms of real-time reasoning on models within comprehensive run-time architectures, and that are programmed using new forms of high-level programming languages. However, while many of the appropriate languages and modeling formalisms exist, as well as real-time reasoning algorithms for planning and monitoring, these elements are currently spread amongst several disciplines. This seminar therefore brought together researchers from four complementary disciplines to work towards languages and architectures for model-based autonomy:

- Runtime Verification and Monitoring
- Model-based Diagnosis and Execution
- Continuous Planning and Dispatching
- Control of Hybrid Discrete/Continuous Systems

Each of those four areas contributed a different point of view to the problem: Runtime verification is a field that has emerged from the formal methods/testing community, and focuses on how to specify behavior of and monitor systems executions. Diagnosis focuses on determining what has gone wrong once an error situation has been detected, requiring knowledge about the executing system. Planning focuses on search-based re-configuration of the executing system, essentially generating programs (plans) on the fly. Finally, control is concerned with classical control theory, and in particular in the context of hybrid systems.

The meeting seeked to map out the necessary architectures, languages, formal models, and underlying reasoning methods for predictable robust and autonomous embedded systems. Discussions at the seminar aimed to identify research needs of autonomous systems in terms of capabilities for monitoring, verification, diagnosis, planning and control in the context of compelling applications. At the same time, participants were be able to discuss technical approaches that have emerged in various related research areas, and assess their applicability to this emerging field.

2 Participants and Program

Including the organizers, 35 researchers from the United States, Europe, Australia, and Japan participated in the seminar. Over the course of the week, 33 talks were given. As one of the main goals of the seminar was to more closely bring together four communities – during the seminar, the acronym RvDPC (Runtime verification, Diagnosis, Planning, and Control) was coined for this combination –, the program was kicked off with tutorial talks on hybrid control, runtime verification, model-based diagnosis, and planning. This overview was

then followed by a mix of technical presentations, panels, and breakout discussions. Due to the large number of presentations, we grouped them into blocks of 3-4 talks each, which were then wrapped up by micro-panel discussions; this structure worked out quite well. To foster exchange among the communities and avoid conference-style talks, each presenter was asked to point out open problems, make connections with the other areas of RvDPC, and identify open challenges for the communities.

In the following days, more room was given for discussions, and the participants split into four "multi-community" groups that aimed to shed a light on four different topics:

- Problem A as problem B: What are the similarities and differences between the tasks carried out in each discipline? Are there common underlying techniques that can be shared? How could one problem (for example, diagnosis) benefit from results from another problem (for example, runtime verification)?
- Modeling: Where do models (system models, specifications) come from and how are they built? What kind of models do we need (continuous/discrete, probabilistic/deterministic, temporal/static)? What modeling formalisms and languages are currently used in each area, and how do they possibly overlap?
- What are models used for: How important is the concept of utility (frequently used in planning)? How does run-time information affect this? What qualities should a model have to support autonomous RvDPC? What is the right balance between autonomy and decision support?
- Reasoning: Under what conditions should reasoning be done offline or online? To what extent can problem decompositions be exploited? How important are trade-offs between efficiency and optimality? Is it preferable to model uncertainty explicitly or to use Monte-Carlo approaches?

As a result of these break-out sessions, a number of insights have emerged:

- In terms of autonomy, all the communities are involved in the migration towards increased autonomy. Decision support is an important midway point.
- It was agreed that Timed Hybrid Automata (THA) are a common language/model for discussion between all four of the communities. Possibilities to construct such models include mining from empirical data, and extracting models from design specifications.
- PDDL is a language for describing automata (only PDDL+ describes full timed hybrid automata). What language is preferred for modeling THAs depends on what kind of automaton needs to be constructed and the goals of the reasoning. The choice of language depends on the cost of supporting the reasoning process, which will be different in each of the four communities.
- Utility of both states and actions is very important in planning and plan execution. The perceived utilities might change at run-time. In diagnosis and runtime verification this was less likely to be the case. In runtime verification it would be very hard to dynamically re-instrument code depending on runtime effects. This could be of value but is not currently done. In planning it

can be crucial to have something of value to return if the process is stopped prematurely, so a notion of relative utility of partial solutions is necessary.
- In planning we often need to query models at run-time, to get accurate information about variables that cannot be directly observed. This could benefit from the instrumentation techniques of runtime verification to enable more efficient query-answering.
- There are several ways in which the different areas might help each other. Planning could use better instrumentation from runtime verification. Perhaps planning could help solve sensor placement problems for diagnosis, and might be able to synthesize monitors for runtime verification. It might be possible to use runtime verification techniques to infer properties from diagnoses (the kind of properties that would normally be inferred directly from observation). There are many other such examples of potential synergies.

At the last day of the seminar, a longer discussion was devoted to possible case studies that could be used to demonstrate the unification of RvDPC methods. We identified several demonstration platforms in three different application areas: autonomous systems (smart cars in smart cities/grids, humanoid robots in households, search-and-rescue scenarios, autonomous underwater vehicles), building energy management, and process control. For each area, task groups were formed to further work out these scenarios.