This document gives a rough schedule and an outline of proposed talks but does not yet assign talks to sessions. Note that we have allowed 10-15 minutes per talk (including questions) along with some extra time in each session to allow things to be reasonably flexible. Please see the next pages for talk titles and abstracts.

Monday

9.00-9.30	Welcome and Introductions
9.30-12.15	Talks

- Kristin Yvonne Rozier
- Cindy Eisner
- Cyrille Artho
- Gordon Pace
- Sylvain Halle
- Torben Scheffel
- Bernd Finkbeiner and Hazem Torfah
- Wolfgang Ahrendt

12.15-14.00	Lunch
14.00-15.30	Discussion
15.30-16.00	Coffee
16.00-17.30	Discussion
18.00	Dinner

Tuesday

9.00-12.15	Talks

- Julien Signoles
- Nobuko Yoshida
- Giles Reger
- Dmitriy Traytel
- Dejan Nickovic
- Borzoo Bonakdarpour and Cesar Sanchez
- Domenico Bianculli
- Adrian Francalanza

12.15-14.00	Lunch
14.00-15.30	Discussion
15.30-16.00	Coffee
16.00-17.30	Discussion
18.00	Dinner

Wednesday

9.00 - 10.30	Concluding Discussions
10.30-10.45	Coffee
10.45-12.00	Closing

An initial effort has been made to group talks into broad topics with the intention that talks in the same topic will be presented on the same day. This grouping is not fixed.

Monday Morning

Cyrille Artho
Domain-Specific Languages with Scala, and model-based testing as an example
Domain-Specific Languages (DSLs) are often classified into external and internal DSLs. An external DSL is a stand-alone language with its own parser. An internal DSL is an extension of an existing programming language, the host language, offering the user of the DSL domain-specific constructs as well as the constructs of the host language. In this presentation, we will give a brief overview of the concepts and also look at an internal DSL used for model-based testing with the tool "Modbat".

Cindy Eisner
PSL: The good, the bad and the ugly
For a specification language to be suitable for formal verification by model checking, it must have sufficient expressive power, its semantics must be formally defined in a rigorous manner, and the complexity of model checking it must be well understood and reasonable. In order to allow widespread adoption in the industry, there is an additional requirement: behavioral specification must be made easy, allowing common properties to be expressed intuitively and succinctly. But while adding syntax is simple, defining semantics without breaking important properties of the existing semantics is surprisingly difficult. In this talk I will discuss various extensions to temporal logic incorporated by PSL, their motivation, and the subtle semantic issues encountered in their definition. I will emphasize where we succeeded, where we were less successful, and point out some features that are still missing.

Gordon Pace
Automata-Based Formalisms for Runtime Verification
In this talk, I will be looking at a number of visual, graph-based formalisms we have used in various projects and domains, with a particular focus on their use in runtime verification. A discussion of their use in different domains. I will be mentioning DATEs (the formalism used by Larva), ppDATEs (that used by StaRVOOrs), contract automata (used to formalise contracts), policy automata (used for social network privacy policies) and more. A discussion of our exploration above and below this layer will ensue - outlining our experience with controlled natural languages (above) and guarded-commands (below).

Wolfgang Ahrendt
Trace Focussed and Data Focussed Specification: Complementary, Competing, or To Be Unified?
In Runtime Verification, as well as in Model Checking, there is a strong focus on traces, often traces of events of some kind. In Deductive Verification, as well as in Runtime Assertion Checking, the focus on properties of the data, at specific points in the execution. Is the difference really motivated by what either communities consider important system properties, or rather by what the respective technologies are good at checking? To which extent should specification formalisms make a pre-choice? I will mention ppDATE as one specific attempt to combine both aspects on the specification level. With that, I hope to trigger a more general discussion about the role and integration of trace focussed and data focussed specification.

Sylvain Halle
Stream Transducers + Runtime Parser = A Do-It-Yourself Specification Language

Torben Scheffel
TeSSLa
The Temporal Stream-based Specification Language (TeSSLa) operates on asynchronous real-time streams. It was first created for specifying properties about programs running on multi core systems and it is currently used and developed in the COEMS project. This talk will show the basic motivation for TeSSLa, the basic operators of TeSSLa, application areas and examples.

Bernd Finkbeiner and Hazem Torfah
LOLA

LOLA is a specification language and stream processing engine for monitoring temporal properties and computing complex statistical measurements. Lola combines the ease-of-use of rule-based specification languages with the expressive power of heavy-weight scripting languages or temporal logics previously needed for the description of complex stateful dependencies. The language comes with two key features: template stream expressions, which allow parameterization with data, and dynamic stream generation, where new properties can be monitored on their own time scale. We give an overview on the development and the current state of our tool in addition to a series of applications

Kristin Yvonne Rozier
Specification: The Biggest Bottleneck in Formal Methods and Autonomy
Advancement of increasingly AI-enhanced control in autonomous systems stands on the shoulders of formal methods, which make possible the rigorous safety analysis autonomous systems require. Formal methods are highly dependent on the specifications over which they reason; not only are specifications required for analysis, but there is no escaping the "garbage in, garbage out" reality. Correct, covering, and formal specifications are thus an essential element for enabling autonomy. However, specification is difficult, unglamorous, and arguably the biggest bottleneck facing verification and validation of aerospace, and other, autonomous systems. This talk will examine the outlook for the practice of formal specification, and highlight the on-going challenges of specification, from design-time to runtime system health management. We will pose challenge questions for specification that will shape both the future of formal methods, and our ability to more automatically verify and validate autonomous systems of greater variety and scale.

Tuesday morning

Julien Signoles
E-ACSL, an Executable Behavioural Interface Specification Language for C Programs
This talk introduces E-ACSL, a behavioral specification language for C programs. It is based on a typed first order logic whose terms are pure C expressions extended with a few specific keywords. Every construct may be executed at runtime. Among others, it provides assertions, contracts, invariants, data dependencies and ghost code. It is powerful enough to express most functional properties of C programs and encode other properties such as LTL properties and information flow policies.

Nobuko Yoshida
The Go Language

Giles Reger
What is parametric trace slicing good for?
Parametric trace slicing is an approach for parametric runtime monitoring that was introduced by tracematches and JavaMOP and later extended by QEA. In this talk I will briefly discuss what it is good for and, perhaps more interesting, what it is not good for.

Dmitriy Traytel
Why my dream language is almost MFODL

Dejan Nickovic
Specification languages for Cyber-Physical Systems
Continuous and hybrid behaviors naturally arise from cyber-physical systems (CPS). In this talk, we will present a brief overview of the specification languages that were designed to tackle CPS-specific properties. We will mainly focus on Signal Temporal Logic (STL) and Timed Regular Expressions (TRE), but will also present their syntactic and semantic extensions. We will discuss what are the strength and weaknesses of these languages and in which situations they should or should not be used.

Borzoo Bonakdarpour and Cesar Sanchez
Asynchronous HyperLTL

HyperLTL is a temporal logic for expressing a subclass of hyperproperties. It allows explicit quantification over traces and inter-trace Boolean relations among traces. The current semantics of HyperLTL evaluate formula by progressing a set of traces in a lock-step synchronous manner. In this talk, we will present our recent work on relaxing the semantics of HyperLTL to allow traces to advance with different speeds. While this relaxation makes the verification problem undecidable, the decidable fragment is expressive enough to express most commonly used security policies. Our new semantics has also application in model-based runtime monitoring.

Domenico Bianculli
The Tale of Dr Jekyll and Mr Hyde in Pattern-based Specification Languages
This talk presents two specification languages, SOLOIST and TemPsy. Both are based on property specification patterns and have been defined in the context of an industrial collaboration. One extends a temporal logic with new modalities; the other defines a domain-specific language that can express a limited subset of temporal properties.

Adrian Francalanza
Two To Tango: A Pair of Specification Languages for Runtime Monitoring
The choice of a specification language is ultimately determined by its intended use. In this talk we motivate the need to employ two specification languages to be able to study the problem of monitorability. In particular, we will outline why we chose a variant of the modal-mu calculus on the one hand, and a process calculus on the other to understand formally what can and cannot be monitored for at runtime. We will also overview how the choice of two formalisms can be used to asses the correctness of a monitor that is entrusted with checking the execution of a system against a specification.