Master's Thesis Projects:
Concurrent and Distributed Software Systems

This document provides a short introduction to the research area of the theses that I am/will be supervising. For the current list of projects, see:

https://github.com/lmkr/mscprojects

The Research Area

The increasing use of handheld devices, embedded computers, wireless communication and cloud computing means that still more software development projects are concerned with *concurrent software systems*, i.e., systems consisting of software components and processes that communicate, synchronise, and share resources. This trend is expected to accelerate in the future in the context of pervasive and ubiquitous computing systems and in the visions of the Internet of Things (IoT). Since software is required to support increasingly advanced use of information technology, it becomes still more complex, and it becomes challenging to ensure correctness and reliability. One of the main reasons for this is that concurrent software systems (by nature) is executed on a number of communicating devices, and it is difficult for the developers to foresee the complete behaviour of the entire system when these components and devices communicate. Errors in concurrent software may in the best case be harmless and without major implications. This is unfortunately not always the case. In mission critical software such as control systems for heavy industry equipment (e.g., in the oil and gas industry) or in software supporting financial transactions (e.g., in the banking sector) errors in software can have both significant human and financial costs.

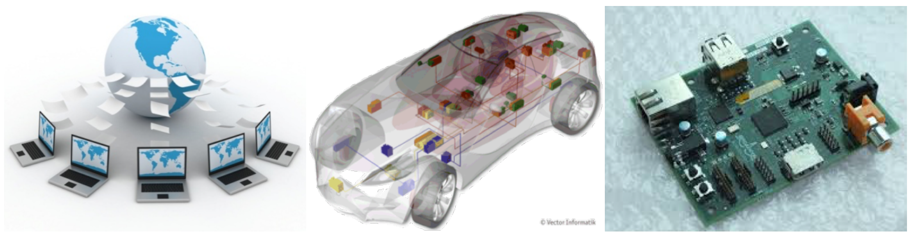

Figure 1: Examples of concurrent software systems

The engineering of software for concurrent systems is therefore a challenging discipline and has made specification and validation techniques, computer tools, programming languages and environments for requirement engineering, design, validation, and implementation an active and prominent area of research. A main approach to validation of concurrent software systems is based on graphical modelling languages and state space exploration techniques for software verification. This approach relies on the construction of executable models of concurrent software systems, and the construction of such models is becoming a still more important discipline in software development. The process of constructing a model of a concurrent software system to be developed is typically done in early phases of software development. The main motivation behind the construction of models is that it is obviously preferable to correct as many design errors and other shortcomings prior to the implementation and deployment of the software system.

The fact that the constructed models are executable means that they can be manipulated, executed and analysed using computer tools, and hence it is possible to explore the system at a very early stage of development. Furthermore, the models make it possible to validate the correctness of software using computer tools, i.e., automatically verify and test that the software system works as intended, and identify errors, omissions, and limitations prior to implementation. State space exploration and model checking techniques represent one of the most promising approaches to computer-aided verification of software systems. The basic idea underlying state spaces is to compute all reachable states and state changes of the system and represent these as a directed graph where the nodes represent states and arcs represent occurring events. State spaces can be constructed fully automatically and from a constructed state space it is possible to automatically answer a large set of verification questions concerning the behaviour of a system.

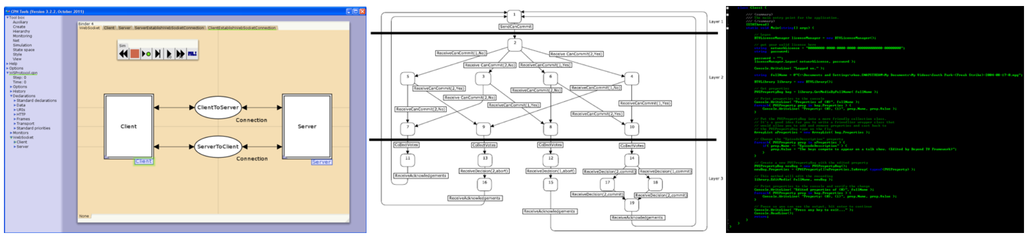

Figure 2: Models of concurrent software systems – verification and code generation

Master's Thesis Projects

Master's thesis projects are offered within the following research areas. Concrete projects are developed in cooperation with the students starting from initial ideas and subject to research interests and background skills of the student. It is also possible to define projects spanning one or more of the four areas.

- **Modelling languages.** Modelling languages for model-based development of concurrent software systems, in particular the Coloured Petri Nets (CPN) modelling language and associated computer tools support (CPN Tools and the ASAP model checking platform). The projects can span from the development of graphical editors and tools (using, e.g., the Eclipse Platform) to extensions of the modelling language and supporting simulation engines.
- **Algorithm engineering**. Development, implementation, and experimental evaluation with space and time efficient algorithms and data-structures for state space exploration and model checking, in particular distributed and external-memory model checking and the sweep-line method.
- **Practical applications.** Case studies on practical applications of modelling, simulation, state space exploration, and supporting computer tools for the development of concurrent software systems. Example areas include recent IETF (Internet Engineering Task Force) protocols for sensor networks and mobile systems, protocols and middleware for Internet-of Things and cloud computing, and cooperation with companies in the Bergen region.

Recent and Ongoing Master's Projects

- Vegard Veiset: An Approach to Semi-Automatic Code Generation for the TinyOS Platform using Coloured Petri Nets. Completed June 2013.
- Matias Vinjevoll: Investigating the use of Scala as Inscription Language and Implementation Language for Coloured Petri Nets. Completed June 2013.
- Uma Behara: Experimental Evaluation of State Space-based Merthods for Automated Analysis of Business Process Models. With Karabin AS. Completed June 2014.
- Endre A. Vestbø: Domain-specific Verification of Formal Workflow Models. Completed August 2014 (with Yngve Lamo).
- Marius Wold: Design and Evaluation of a Mobile Multi-Player Version of the SimSubsea Educational Game. Completed June 2015.
- Jens-Marius Hansen: Model-Driven Techniques for Evaluation of Responsiveness in Mobile Applications. January 2016.
- Erlend Rommetveit. Model-driven Development of a CPN editor. Expected completion, January 2016.
- Andreas Lilleskare. State Space Exploration with the Sweep-Line Method. Expected June 2017.

Selected Background Literature

Copies of papers/articles can be obtained by contacting Lars Kristensen.

- K. Jensen and L.M. Kristensen. Coloured Petri Nets: A Graphical Language for Formal Modelling of Concurrent Systems. Accepted for Communications of the ACM, ACM Press, 2015 www.cpntools.org
- K. Jensen and L.M. Kristensen. *Coloured Petri Nets – Modelling and Validation of Concurrent Systems*. Springer-Verlag, July 2009. home.hib.no/ansatte/lmkr/cpnbook
- S. Evangelista and L.M. Kristensen. A Sweep-Line Method for Buchi Automata-based Model Checking. In Vol. 131, Issue 1 of Fundamenta Informaticae, IOS Press, 2014
- K. Simonsen and L.M. Kristensen. Implementing the WebSocket Protocol based on Formal Modelling and Automated Code Generation. In Proc. of IFIP Conference on Distributed Systems and Interoperable Systems. Vol. 8460 of LNCS, pp. 104-118, Springer, 2014.
- L.M. Kristensen and K. Simonsen. Application of Coloured Petri Nets for Functional Validation of Protocol Designs. In Transactions on Petri Nets and Other Models of Concurrency, LNCS Vol. 7480, pp. 56-115, Springer, 2013.
- V. Veiset and L.M. Kristensen. Transforming Platform Independent CPN Models into Code for the TinyOS Platform: A Case Study of the RPL Protocol. In Proc. of Petri Nets and Software Engineering, 2013.

Contact

Lars M. Kristensen, Høgskolen på Vestlandet

Room E503,

E-mail: lmkr [at] hvl.no / Web: home.hib.no/ansatte/lmkr

List of current proposed project: https://github.com/lmkr/mscprojects