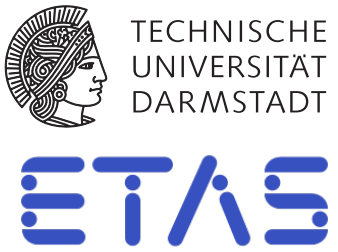
Application of language principles of ABS in deeply embedded systems in the context of ESDL



Masterthesis in Software Engineering (Prof. Dr. Reiner Hähnle)

Background

A large portion of today's software is hidden in systems not directly visible to the end user. An important example in this category is software in automobiles. This software is essential for the operation of the vehicle as well as the diagnosis of the vehicle state. Today's vehicles – not only those in the premium segment – contain a large number of interconnected computation nodes and the software totals several millions lines of code. Due to the application area software in automobiles have some specific characteristics that make them different to "normal" software:

- In many cases there is no dynamic object model during run time, but the object model is static ("the number of wheels of a car / number of cylinders of a combustion engine does not change dynamically"). The greatly simplifies the programming model as aspects of memory management have to be consider only in a very restricted way if even. In addition, the run-time model has a very static structure with regular (cyclic) repeating activation (e.g. of control systems)
- The computation are heavily interrelated either through data flow (same signals) or coupling of the technical process (engine, vehicle), but computation is distributed across several independent computation nodes (electronic control units). Real time and multi-core are additional challenges that are to be mastered by the user of higher-level design methods like model based software engineering and others,.
- High demands with respect to quality and correctness: the more can be proven the better (see also the ISO 26262 for software safety in automobile applications).

ETAS supports the software development engineer to master these challenges of deeply embedded software development in the automotive industry with tools for model-based software development. The programming and modeling language *ESDL* (implemented in the Eclipse eco-system) as well as the widely spread product *ASCET* are two examples of such tools.

On the other hand, there has been progress in area of formal systems in the academia in the last years. Software modelling concepts and languages have been developed for concurrent systems with a special focus on simulation, static analysis and verification. Here we would like to especially mentioned the *Abstract Behavioral Specification Language* (ABS) that comprises an entire tool portfolio (e.g. dead-lock analysis, run-time analysis, formal verification) and is implemented in the Eclipse eco-system.

Task

The central goal of the master thesis is to investigate how the concepts of the ABS language (e.g. modular concept for concurrency, formal semantics, Java code generation, cost analysis, verification, test case generation) can be applied to deeply embedded systems under special consideration of the aforementioned specific characteristics of these systems. We propose an empirical approach with the use of suitable examples that represent the essential language constructs of the ESDL language. The examples then shall be modelled also in ABS. On the basis of these ABS models, it is the task to investigate if substantial statements can be made about the corresponding ESDL program example, e.g. by using analysis based on code generated by the Java Code generator and subsequent analysis performed on generated code. The investigation should comprise also the identification and formalization of desired characteristics of the ESDL model (e.g. reaction time, resource consumption, safety, correctness). These characteristics shall be formally proven on the corresponding ABS examples, or

at least been validated through simulation and generation of suitable test cases. These investigations shall lead in consequence answers to the following questions:

- Is it possible to model the ESDL examples in sufficient detail in ABS, so that the desired characteristics can be ensured (or validated as described above)?
- Which enhancements/adaptations would be required (either on the ESDL or ABS side) to make effective use of the ABS methods in an industrial context for the analysis of deeply embedded software? Can the ABS methods be applied to the deeply embedded software directly?
- How can the ESDL and ABS model languages be coupled so that there is an immediate benefit for a user of the ESDL language? Is it possible to automatically generate an ABS model from an ESDL model so that the ABS analysis capabilities can be made available on the base of the generated ABS model? If so, how would a concept for such a tool coupling look like?

Contact

The thesis requires knowledge and interest in formal methods. The thesis is done in corporation with ETAS (Bosch).

Interested?

- Reiner Hähnle (haehnle@cs.tu-darmstadt.de, Room A204, S2|02)
- Richard Bubel (bubel@cs.tu-darmstadt.de, Room A225, S2|02)