Altreonic NV.
www.altreonic.com
Tel. +32 16 20 20 59

From deep space to deep sea

Linden, 21 June 2013

Subject: Thesis and project proposal themes

Dear Sirs,

Following the productive and interesting meeting and presentation on 20st June at the Minho University, it is clear that there is a large common ground for our R&D activities.

Please find attached a list of title proposals that can be used:

- for the student projects;
- for MsC thesis work;
- for projects executed as internships in the context of a master thesis.

The subjects cover multiple domains.

For practical contacts, Leonel Braga will act as liason. Altreonic will actively coach the students during the execution.

Best regards,

Eric Verhulst,
CEO/CTO

Thesis and project proposal themes

Altreonic NV is a small innovative company in Belgium focusing on trustworthy systems engineering. Altreonic has developed its own formalized methodology and supporting tools. In cooperation with the University of Minho, Altreonic offers students to possibility to support these efforts with additional research and proof-of-concept implementations. The work can be done as team projects, MsC thesis or internships.

1. OpenComRTOS Designer

OpenComRTOS Designer is a set of tools that support the development of distributed real-time applications (mostly embedded). The developer first models in a visual way the hardware as well as the application topology. Code generators (currently supporting ANSI-C C99) generate application specific template code and datastructures based on metamodels. The application code is linked with the kernel libraries and booted on the target system. Event tracers and profilers allow to analyse the application software. While the OpenComRTOS kernel was formally developed, the application code is still written by the developer. The projects aims at reaching higher level of assurance for the application.

1.1 Use of ADA programming language for safety and certification purposes

Although C is widely used for embedded systems, for high integrity systems, the use of C code is problematic. Therefore, the use of ADA or its subset SPARK is envisioned

1.1.1 Developing an ADA programmer's interface to the OpenComRTOS kernel for application programming of Tasks in ADA o SPARK/ADA.

In this project the OpenComRTOS kernel is kept as is (compiled from C code), but an interface is developed that allows to program application tasks in ADA or the safe subset SPARK. The project encompasses:

- Investigating how C and ADA can be linked.
- Investigating the impact on the semantics.
- Investigating the impact on the supporting tools (code generators, profiling tools, etc.)
- Implementing proof of concept.

1.1.2 Redeveloping the OpenComRTOS kernel in ADA/SPARK. If SPARK is feasible, apply the formal verification.

In this project the goal is to re-implement the OpenComRTOS kernel in ADA or the safe subset SPARK/ADA. The project encompasses:

- Investigating the feasibility and constraints.
- Investigating the trade-offs when using SPARK/ADA
- Implementing proof of concepts of:
    - Code generators (kernel data structures)
    - Kernel service reimplementation
    - Formal verification o the SPARK code.

1.2 Generic RTOS services

In OpenComRTOS Tasks interact through so-called "hubs". A hub is essentially an implementation of a Guarded Action, composed of a synchronization predicate and an action predicate (both are implemented as call-back functions). Using this a wide range of semantics are possible. The goal of this project is to use a generic grammar of interactions as found in coordination languages such as REO. The goal of the project is as follows:

- Investigate the mapping coordination languages to the OpenComRTOS hub concept.
- Develop an orthogonal classification.
- Develop automatic code generation for the services
- Integrate with the OpenComRTOS designer metamodels.
- Proof of concept.

1.3 Formal or specification focused front-end for RTOS applications

The goal of OpenComRTOS Designer (together with the integration in the GoedelWorks portal) is to be able to develop fully trustworthy systems and applications. The weak point in the toolchain is that applications are still developed using error-prone programing. The goal is to use specification oriented front-end tools that allows formal verification as well as code generation

1.3.1 Event B Support

The goal is to use Event-B and Event-B model checkers like Pro-B and Atelier B to specify incrementally the behavior of the sequential parts of the application Tasks followed by C-code generation. The project involves:

- Studying Event-B and the supporting tools.
- Study the C-code generation and its constraints
- Implement a proof of concept implementation.

1.3.2 Alloy support

The goal is to use Alloy and its model checkers to specify the behavior of the sequential parts of the application Tasks followed by C-code generation. The project involves:

- Studying Alloy and the supporting tools.
- Study the C-code generation and its constraints
- Implement a proof of concept implementation

1.3.3 Haskell support

The goal is to use Haskel as a specification language to specify the behavior of the sequential parts of the application Tasks followed by C-code generation. The project involves:

- Studying Event-B and the supporting tools.
- Study the C-code generation and its constraints
- Implement a proof of concept implementation

2 Systems and safety engineering methodology

2.1.1 Formalisation of the ARRL criterion

The ARRL (Assured Reliability and Resilience Level) is a novel, normative criterion develop by Altreonic that allows to classify system components in terms of their support for safety critical systems. The emphasis is on how the component is designed and architected to handle faults. Using the ARRL criterion, rules can be defined on how ARRL level components can be combined to achieve required Safety Integrity Levels for a given system, independently of the application domain. The project entails:

- Studying the problem domain.
- Applying the ARRL criterion to a selected sample of components.
- Investigating how to achieve completeness of the set of properties that define a give ARRL level.
- Investigate how component and their architecture can be analysed and classified according to the ARRL criterion.
- Investigate how ARRL architectures can be formally verified.

2.1.2 Safety extensions to a Virtual Machine for C code.

Part of the OpenComRTOS support is the capability to execute binary compiled tasks in a dynamic way independently of the target processor. This is achieved by linking in a 3 KB Virtual Machine that interprets the binary compiled code. This VM was generated from a generic specification using Haskel. The goal of this project is to enhance the VM with safety support by extending the instruction interpretation with runtime checks that can prevent runtime errors like e.g. memory violations.
The project entails:

- Studying the problem domain.
- Selecting candidates for runtime checking
- Formally verifying the VM implementation (generated from the Haskel specification)
- Comparing a number of approaches
- Proof of concept with a selected set of runtime checks.

2.1.3 Automatic fault-tolerance support for distributed real-time applications

A major advantage of the OpenComRTOS architecture is the separation between hardware topology and application topology. In combination with a very small code size and a general use of packet switching, this makes adding safety and security support relatively simple. The goal of this project is to automate the generation of fault tolerant application architectures by automatic

triplication and voting generation without changing the original application. The project entails:

- Studying the problem domain.
- Develop a suitable code generation strategy.
- Develop a fault tolerant voter architecture.
- Develop and verify fault tolerant link drivers.
- Proof of concept demonstration.

2.1.4 MAST integration

The goal of this project is to add real-time scheduleability analysis to the OpenComRTOS Designer. This will be achieved by using the MAST real-time analysis tool. The project will investigate:

- Needs and constraints of using the MAST tool
- Ways to obtain the profiling information from OpenComRTOS Designer
- Interfacing OpenComRTOS Designer with MAST

2.1.5 Frama-C integration

The goal of this project is to increase the assurance in the application and kernel code written in C. Therefore Frama-C is used to verify and proof the C-code. The project entails:

- Studying Frama-C and its applicability.
- Developing use-case scenarios as a function of Frama-C's capabilities.
- Integrating with the OpenComRTOS design process.
- Developing proof of concept demonstrations.

2.1.6 Verification of drivers

An important element of a real-time software are the drivers that create an interface between the application and peripheral processor or board hardware. However, often these drivers are developed by third parties. They are often overly complex but not formally verified. The goal of this project is develop a methodology that allows to verify and validate third party drivers (written in C). The project entails:

- Studying the problem domain.
- Selecting a few test case drivers.
- Comparing a number of approaches
- Proof of concept with a selected set of drivers.