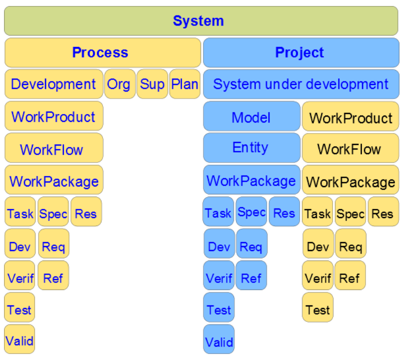
NEW: Ada and SPARK-Ada interface for OpenComRTOS Designer

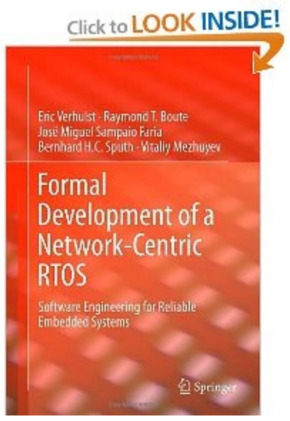
Unique Open Technology Licensing offer from Altreonic for Certifiable Trustworthy Software and Systems

Altreonic offers advanced embedded systems technology under a unique risk-free **Open Technology License**. The licensee receives all supporting design documents, formal models, source code, test suites, etc. and the right to rebrand the software whereby all certification and business risks are seriously reduced. Free yourself from legacy COTS and open source limitations. Two technologies are offered: the formally developed, network-centric **OpenComRTOS™ Designer** and the internet based **GoedelWorks™** portal for supporting certifiable engineering projects. Soon to be integrated. Complemented with engineering services.



GoedelWorks™ reflects our global "Correct by Construction" approach, covering from early requirements capturing till the last line of source code and hardware component. Based on a formalised and clean meta-model, it allows describing any (engineering) process and support users executing it. Out-of-the-box support for ASIL centered IEC-61508, ISO-26262, ISO-13849, ISO-25119, ISO-15998 and IEC-62061, CMMI and automotive SPICE process flow. Integrate your own specific process flows for pre-certification support during the development of the system.

GoedelWorks™ implements the core concepts of any engineering standard: full traceability and configuration management from top level Requirements to the last line of source code.


Click to LOOK INSIDE!
Eric Verhulst · Raymond T. Boute
José Miguel Sampaio Faria
Bernhard H.C. Sputh · Vitaliy Mezhuyev
Formal Development of a Network-Centric RTOS
Software Engineering for Reliable Embedded Systems
Springer

OpenComRTOS™ Designer breaks new grounds in the field of Real-Time Operating Systems. Formally designed and verified it is a 4th generation of the Virtuoso RTOS used by ESA on the Rosetta mission.

Conceptually a scalable communication layer to support heterogeneous multi-processor systems in a transparent way, it runs equally well on a single processor. It runs on small microcontrollers, many-core chips with little memory, parallel DSPs as well widely distributed systems and supports FPGAs. Unique support for distributed priority inheritance. Scalable, yet only requiring about 5 to 15 KB/node. Code is generated from a visual modelling environment. Full qualification package available. A big resource and time saver for management, a joy to use.

ALTREONIC NV
Gemeentestraat 61A B1
B3210 Linden, Belgium

CONTACT
+ 32 16202059
info.request (@) altreonic.com

WEB
www.altreonic.com

Altreonic

Ada and SPARK-Ada interface for OpenComRTOS Designer

Ada has a long history. Originally developed in the late 70's on request of the US DoD, it became available with a certified compiler in 1983. While the language had as goal to improve the quality of software, in its striving to be complete (procedural, object-oriented, modularity, concurrent tasking and many more features), it was complex and fairly heavy to use. Nevertheless, it was and still is the language of choice for large safety critical applications, especially when large teams are involved. Its complexity, the steep pricing for the tools and its lower performance inhibited its wider use. Hence C compilers offering often better performance and more control over the hardware gradually became the compiler of choice even if the language has many safety issues. Ironically, VHDL which is a widely used programming language to develop hardware circuits heavily borrowed from Ada.

Over the years efforts have been made to reduce the complexity and to improve the usability of Ada. For example the Ravenscar profile introduced a large number of restrictions on Ada-95. It is now part of Ada-2005 making it easier to use for hard real-time safety critical applications. Similarly SPARK was defined as a derived language (originally since 1983). SPARK-2014 is based on Ada-2012 and adds a new dimension in formal verification by adding "contracts" as part of the language. Using the GNAT Prover as a back-end of the GNAT compiler it combines for the first time human readable formal specifications with formal verification in a programming language, making it ideally suitable for high integrity systems (as found in safety and security critical applications). The same pre- and post conditions, loop invariants, etc. will become runtime assertions and can be formally verified at compile time using the prover tool.

By adding interfaces for Ada and SPARK-Ada as programming languages that can be used with OpenComRTOS Designer, Altreonic extends this evolution towards a very clean (CSP inspired) concurrent Tasking model with hard real-time capabilities from one to many processor systems. Not all Ada constructs are supported (for similar reasons as given for the Ravenscar profile and SPARK), as OpenComRTOS provides itself a formally developed and verified runtime layer for multi-tasking with inter-task synchronisation and communication and priority based preemptive scheduling as well as including support for distributed priority inheritance using a ceiling protocol. For reasons of safety, OpenComRTOS also favors a model whereby code and datastructures are generated statically at compile time, thereby reducing the risks of runtime error conditions.

As with the standard C interface, the user defines his application as a number of task entities and interaction hubs. The latter can be binary events, counting semaphores, FIFOs, Ports, Resources (acting like an improved mutex), DataEvents, BlackBoards, MemoryBlockQueues, Memory- and Packet Pools. The interaction services using these hub types acts like guarded actions (i.e. synchronisation points) which then enable the predicate actions associated with the hubs.

What sets OpenComRTOS apart is that all these services operate system-wide whereby priorities are also system-wide parameters to schedule everything in order of priority, including when communicating, preserving real-time properties across processor boundaries. When Resource locking is used, priority inheritance works distributed as well and reduces system-wide blocking times.

The user can now program his OpenComRTOS Tasks either in C, C++, Ada or SPARK_Ada and mix them in a single application. When using SPARK this means that he mainly programs the sequential (procedural) segments with the multi-tasking being provided by the OpenComRTOS runtime layer. SPARK allows him to make use of formal verification for various aspects related to data-flow analysis, information-flow analysis, robustness properties and various functional properties expressed as contracts, loop-invariants or loop-variants. These expressions are an integral part of the source code and document the intention of the program. Contrary to e.g. runtime assert statements in C, the properties can be formally proven at compile time based on the source code.

From Deep Space to Deep Sea Altreonic