The ASSERT Virtual Machine Kernel: Support for preservation of temporal properties *

Juan Zamorano Juan A. de la Puente José A. Pulido
Santiago Urueña
Universidad Politécnica de Madrid (UPM), Spain

Contact author: *José A. Pulido*
Address: *ETSI Telecomunicación UPM*
Ciudad Universitaria s/n, E 28040 Madrid
Telephone: *+34 913367366 ext. 3019*
Fax: *+34 913367333*
E-mail: *pulido@dit.upm.es*

1 Introduction

The ASSERT Project[1] is aimed at defining new software engineering methods and tools for the development of critical embedded real-time systems in the aerospace domain. One of its main achievements is a new model-driven software process, which is based on the concept of property-preserving model transformations. Functional models developed with appropriate tools for the application domain are embedded in *containers* defining component interfaces and non-functional (e.g. timing) properties in a platform-independent set of notations. The resulting model is then automatically transformed to a platform-specific model using deployment information on target computer nodes, communication channels, and software platforms. Finally, source code for each computer node is automatically generated from the platform-specific model.

The key element of the ASSERT process is that non-functional properties must be preserved during all phases of model transformations. In order to ensure that properties are preserved in model transformations and that the different views of each model are consistent with each other, a common meta-model has been defined which provides a formal basis to the whole process. This meta-model is called the Ravenscar Computational Model (RCM).

2 The ASSERT Virtual Machine

The ASSERT Virtual Machine (VM) is the execution platform on which ASSERT applications run. It is based on the Ravenscar Computational Model (RCM), and only accepts software entities which are valid under this model. To this purpose, application-level software is built in such a way that all functional code and data is encapsulated into *VM-Level Containers* (VMLC), which are the run-time code entities that operate on top of the VM.

In order to guarantee the required real-time behaviour, the ASSERT VM is bound to a GNAT cross-compilation system that only accepts code which is legal in terms of the underlying computational model, i.e. the RCM. Since this model is based on the Ada Ravenscar Profile [ISO05], an Ada 2005 [ISO07] compilation system is used. Accordingly, VMLC are coded in Ada 2005 restricted by the Ravenscar profile. However, the functional code can be writen in Ada, C or C++. This compilation system is hosted

*This work has been partially funded by the Spanish Ministry of Education, project no. TIC2005-08665-C03-01 (THREAD), and by the IST Programme of the European Commission under project IST-004033 (ASSERT).

[1] Automated proof-based system and software engineering for real-time systems, see `www.assert-project.org`.

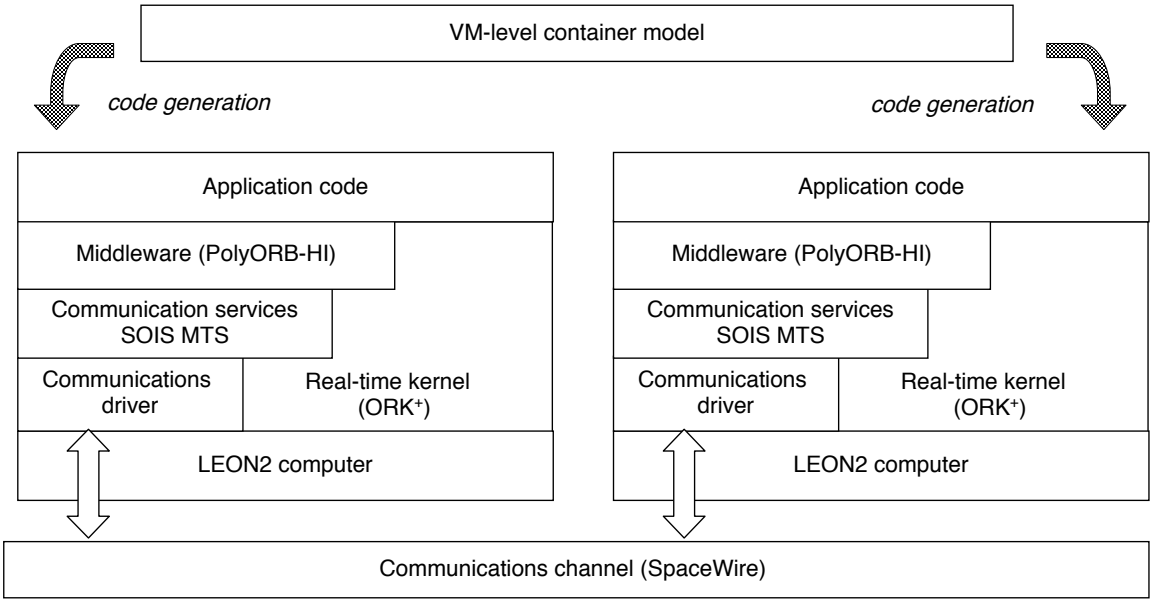

Figure 1: ASSERT Virtual Machine Architecture.

on a PC-compatible computer with a GNU/Linux operating system, and is targeted to the SPARC V8 architecture.

The ASSERT VM provides run-time services that support the timely execution of VMLC. It also supports predictable communication and distribution on a network of computer nodes. In order to properly implement these functions, it has been designed with a layered architecture including the following subsystems (see figure 1):

- A real-time kernel providing concurrent execution, scheduling, synchronization, and timing services to Ada Ravenscar tasks. The kernel also provides basic support for developing device drivers.
- A network access layer containing communication drivers.
- A communications subsystem providing a SOIS MTS transport service.
- A middleware layer providing distribution transparency and higher-level services for distributed applications.

3 The Real-Time Kernel

The VM real-time kernel is an evolved version of ORK[dlPRZ00], a small, high performance real-time kernel that provides restricted tasking support as defined by the Ada Ravenscar profile. The kernel has been designed for efficient support of Ada tasking constructs, but can also be used with C programs, using a C interface package that is provided for this purpose. The current version, ORK+[UPRZ07], includes support for the new Ada 2005 timing features, and is integrated with the GNAT GPL 2007 compiler. The full package including the compiler and the kernel is called GNAT for LEON. The integrated kernel is also known as the GNAT for LEON Bare Board Kernel.

The kernel is not intended to be directly used from Ada programs. Instead, an interface to the GNU Ada Runtime Library (GNARL) is used so that Ada tasking constructs can be directly used by the application programmer [GB94]. The parts of GNARL which are dependent on a particular machine

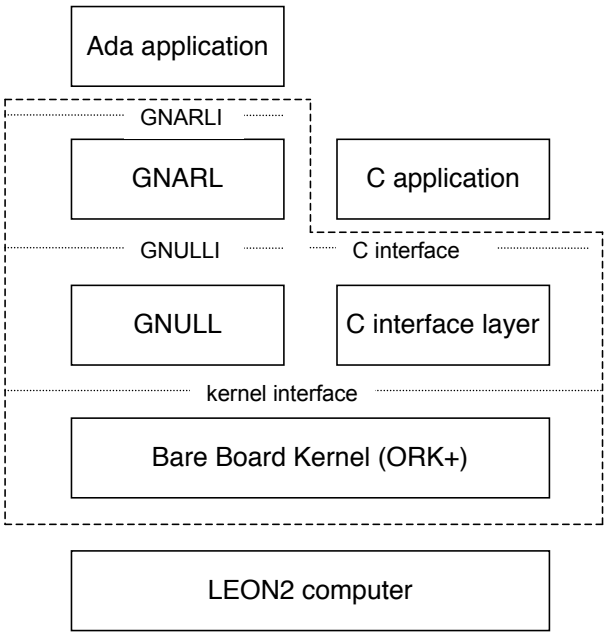

Figure 2: Architecture of the GNAT for LEON run-time system.

and operating system are known as GNULL, and its interface to the platform-independent part of the GNARL is called GNULLI. GNULL is built on top of the GNAT for LEON Bare Board Kernel (figure 2).

GNAT for LEON uses a restricted version of GNARL with reduced size and complexity which was developed by AdaCore[2] with certification in mind and just to support the Ravenscar Profile. A special version of GNULL is also used, which interfaces directly with the bare board kernel.

The restricted version of GNARL has been enhanced by UPM with the Ada library packages needed to support execution time clocks and timers, group budgets, and timing events: `Ada.Execution_Time`, `Ada.Execution_Time.Timers`, `Ada.Execution_Time.Group_Budgets`, and `Ada.Real_Time.Timing_Events`. As a result, the new Ada 2005 features are only available to Ada programs.

The kernel provides the following functionality:

Thread management: Only static Ada tasks declared at the library level are accepted. Consequently, the associated data structures can use only statically allocated memory.

Thread scheduling: The scheduling of threads is performed according to the `FIFO within priorities` and the `ceiling locking` methods [ISO07, D2.3].

Thread synchronization: Only Ravenscar-compliant synchronization is supported. The only synchronization operations provided by the kernel are unconditional suspend and resume.

Time management: Time is represented as a 64-bit integer number of ticks. A tick is a real-time interval during which the clock value remains constant which is four times the period of the LEON2 input clock.

Execution time monitoring: Every thread has an execution-time clock that computes the amount of CPU time it has consumed. Execution-time is also represented as a 64-bit integer number of ticks. The execution-time clock of the running thread takes into account the elapsed time from the last context switch.

[2] `http://www.adacore.com/`

Group budgets: Groups of threads can have one group execution-time budget associated to them. This feature is supported on top of execution-time monitoring of single threads.

Absolute alarms: Absolute alarms can be set with a precision of one tick by absolute delays, execution-time timer of the running thread or timing events. Absolute alarms are implemented in a way that provides a high precision with a low overload.

Interrupt handling: Interrupt handlers are called directly from the hardware, and are executed as if they were directly invoked by the interrupted thread. However, a preamble is executed before the protected procedure handler and an epilogue after it. It must be noticed that the Ravenscar profile makes possible this simple and efficient implementation.

Configuration: The kernel has configuration parameters that enable it to be tailored to different applications. The configurable parameters are: size of the interrupt stack, frequency of the processor clock, available memory space, and range of priorities.

4 Properties ensured by the kernel

The real-time kernel features described in the previous section allow to support systems compliant with the correctness by construction paradigm (with regard to real-time behaviour). The thread management module ensures:

Periodic activation: The real-time kernel ensures periodic activation of periodic elements.

Sporadic activation with minimum inter-arrival time: Every sporadic element is suspended until the specified minimum interarrival time has expired in order to avoid unexpected too close activations.

Moreover, the Ravenscar computational model defines a restricted tasking model which exclude all those constructions that introduce temporal indeterminism, hence a system to be run on top of the real-time kernel is suitable for static analysis. The most important implication is that deadlines can be guaranteed at design time by static analysis thanks to the response-time analysis theory. MAST+ is a timming analysis tool, derived from MAST [GGPD01], which has been enhanced to perform such analysis according to the ASSERT process characteristics. Its main advantage consist of integrating a sensitivity analysis [PV07], so it does not return a simple boolean response (a task set is feasible or not), but modify some real-time parameters in order to make feasible a set that initially was not.

This static analysis process can be enhanced also by means of WCET analysis tools such as RapiTime©, an analysis tool that provides a solution to the problem of determining worst-case execution times (WCET) for software components running on advanced microprocessors.

Finally, in spite of the static analysis performed, to enforce the real-time behaviour, Virtual-Machine Level Containers (VMLC) include a set of mechanisms intended for run-time detection of temporal faults [PUZd07] provoked by an unexpected behaviour of periodic and sporadic activities, such as:

Deadline overruns detected by timing events: A timing event is set at the very moment a deadline expires. If the associated activity finishes its execution in time, it cancels the alarm (in the case of sporadic activities), or reprogram it for the next period (in the case of periodic activities).

Execution-time overruns detected by execution-time timers: At the beginning of every activity, an execution-time timers is armed with its Worst Case Execution-Time (WCET). If the activity attempts to consume too much CPU time, the timer budget is exhausted releasing an alarm.

5 Conclusions

The ASSERT Virtual Machine is an specialized execution-platform for high-integrity real-time systems designed conforming to the ASSERT process. It has been designed conforming to several principles:

- The Ravenscar Computational Model, based on well-known principles that ensure a predictable, analysable timing behaviour.
- A software development process based on models an model transformations preserving the real-time properties of the system.
- Inclusion of a set of features that guarantee at run-time the preservation of the properties assumed at design-time.

This platform together with the process has been successfully used in several industrial-grade projects, with LEON2 hardware platforms and real software, including sophisticated functions used in space missions.

This paper has only covered temporal isolation, however, plans for the near future include to add spatial isolation by means of novel techniques applicable to the compilation toolchain that do not need a Memory Management Unit (MMU) and do not add overhead to the run-time.

References

[dlPRZ00] Juan A. de la Puente, José F. Ruiz, and Juan Zamorano. An open Ravenscar real-time kernel for GNAT. In Hubert B. Keller and Erhard Plödereder, editors, *Reliable Software Technologies — Ada-Europe 2000*, number 1845 in LNCS, pages 5–15. Springer-Verlag, 2000.

[GB94] E. W. Giering and T. P. Baker. The GNU Ada Runtime Library (GNARL): Design and implementation. In *WADAS '94: Proceedings of the eleventh annual Washington Ada symposium & summer ACM SIGAda meeting on Ada*, pages 97–107, New York, NY, USA, 1994. ACM Press.

[GGPD01] Michael González Harbour, J. Javier Gutiérrez, J. Carlos Palencia, and José María Drake. MAST modeling and analysis suite for real time applications. In *Proceedings of 13th Euromicro Conference on Real-Time Systems*, pages 125–134, Delft, The Netherlands, June 2001. IEEE Computer Society Press.

[ISO05] ISO/IEC. *TR 24718:2005 — Guide for the use of the Ada Ravenscar Profile in high integrity systems*, 2005. Based on the University of York Technical Report YCS-2003-348 (2003).

[ISO07] ISO/IEC. *Ada 2005 Reference Manual. Language and Standard Libraries. International Standard ISO/IEC 8652:1995/Amd 1*, 2007. Published by Springer-Verlag, ISBN 978-3-540-69335-2.

[PUZd07] José A. Pulido, Santiago Urueña, Juan Zamorano, and Juan A. de la Puente. Handling temporal faults in Ada 2005. In Nabil Abdennadher and Fabrice Kordon, editors, *Reliable Software Technologies — Ada-Europe 2007*, number 4498 in LNCS, pages 15–28. Springer-Verlag, 2007.

[PV07] Marco Panunzio and Tullio Vardanega. A metamodel-driven process featuring advanced model-based timing analysis. In Nabil Abdennadher and Fabrice Kordon, editors, *12th International Conference on Reliable Software Technologies — Ada-Europe 2007*, number 4498 in LNCS, pages 128–141. Springer-Verlag, 2007.

[UPRZ07] Santiago Urueña, José Antonio Pulido, José Redondo, and Juan Zamorano. Implementing the new Ada 2005 real-time features on a bare board kernel. *Ada Letters*, XXVII(2):61–66, August 2007. Proceedings of the 13th International Real-Time Ada Workshop (IRTAW 2007).