ADA, THE PROGRAMMING LANGUAGE OF CHOICE FOR THE UPMSAT-2 SATELLITE

Jorge Garrido, Juan Zamorano, Juan A. de la Puente, Alejandro Alonso, and Emilio Salazar

ETSI Telecomunicación, Universidad Politécnica de Madrid (UPM), E28040 Madrid, Spain

ABSTRACT

The proper selection of development mechanisms and tools is essential for the final success of any engineering project. This is also true when it comes to software development. Furthermore, when the system shows very specific and hard to meet requirements, as it happens for high-integrity real-time systems, the appropriate selection is crucial. For this kind of systems, Ada has proven to be a successful companion, and satellites are not an exception. The paper presents the reasons behind the selection of Ada for the UPMSat-2 development, along with the experience and examples on its usage.

Key words: Ada; Ravenscar profile; real time systems; embedded systems; on-board software.

1. INTRODUCTION

The UPMSat-2 project is aimed at developing an experimental micro-satellite mission that can be used as a technical demonstrator and an educational platform. It is being carried out in a collaborative way by several UPM research groups and industrial companies. The STRAST group is responsible for all the on-board and ground software development (de la Puente et al., 2014b). In order to fulfil its technology demonstrator purpose, the UPMSat-2 payload consists of a set of experiments proposed by different Spanish companies, the European Space Agency (ESA), and other UPM research groups. The aim of these experiments is focused on testing new equipment behaviour in space conditions.

The satellite has an external envelope of $0.5 \times 0.5 \times 0.6\,\mathrm{m}$, as shown in figure 1, and a total mass of $50\,\mathrm{kg}$ approximately. The satellite will be set on a low Earth noon sun-synchronous polar orbit (Fortescue et al., 2011), with an expected altitude of $600\,\mathrm{km}$ and a period of $97\,\mathrm{min}$. Energy is provided by solar cell panels located on the sides of the satellite. Radio communications with the ground segment are ensured by a radio equipment operating in the VHF 400 MHz band, with a linear antenna located on the top of the satellite structure.

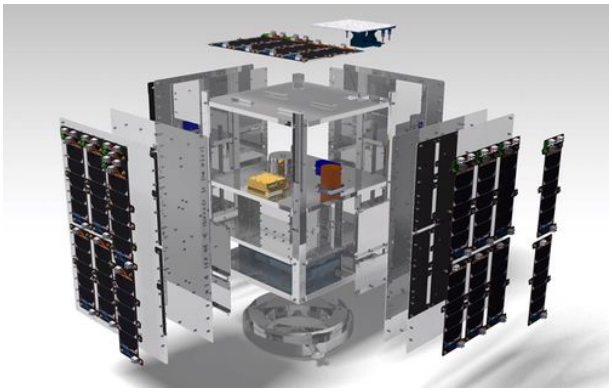

Figure 1. General view of the satellite platform

An on-board computer (OBC) carries out all the data handling, supervision and control functions of the satellite, including the management of the payload experiments. It is based on a LEON3 processor (Gaisler) with 4 MB SRAM, 1 MB EEPROM, timers, analog inputs, and digital I/O.

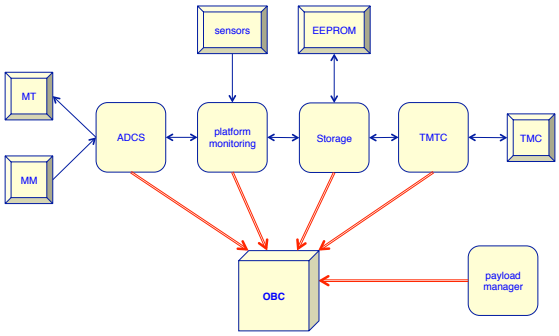

Figure 2. UPMSat-2 on-board software architecture.

The software architecture is shown in figure 2. The on-board software consists of a number of subsystems that control the operation of the satellite.

The main subsystems are:

- *Attitude determination and control system (ADCS)*. This subsystem controls the attitude of the satellite based on measurements of the Earth magnetic field. The basic actuators are magnetorquers, although a reaction wheel can be used as an experiment.
- *Platform monitoring*. This subsystem is in charge of periodically checking the structural conditions of the satellite, including housekeeping data such as temperatures and power supply voltages. Housekeeping data are periodically sent to ground by means of telemetry messages.
- *Telemetry and telecommand (TMTC)*. This subsystem handles all the communications between the satellite and the ground segment using the radio equipment.
- *Payload manager*. This subsystem manages the execution of the experiments. Experiments are staterted upon receipt of specific telecommands, and the results are sent to ground through telemetry messages.
- *Storage*. This subsystem manages the use of non-volatile memory to store relevant data, preventing data loss in case of a computer shut down.

The ADCS, platform monitoring, and TMTC subsystems have been assigned criticality level B as per EDCSS-Q-80C (ECSS), and the other subsystems have classified as level C.

2. ON-BOARD SOFTWARE DEVELOPMENT

2.1. Language choice

Ada 2005 has been chosen as the main language for the UPMSat2 on-board software, due to its outstanding characteristics for the development of high-integrity embedded systems (Taft et al., 2006; Barnes, 2008). The most relevant of these properties are:

Strong typing. This feature enables many potential errors to be detected at compilation time. Unlike other strongly typed languages, Ada has a rich set of mechanisms for user-defined types, including subtypes, derived types, and tagged types, thus enabling checking for validity while letting programmers use the best suited type for each purpose. Access types enable safe pointer usage, thus avoiding dangling pointers and other vulnerability sources found in other programming languages.

Reusability Generics and type extension provide a strong basis for reuse. The Ada approach to object-oriented programming, where encapsulation is separated from type extension, defines a flexible programming model that enables programmers to choose different paradigms, depending on the use of extensions, inheritance, and interfaces (Rosen, 1992).

Support for concurrency and real-time. The Ada concurrency model is very complete and powerful, and the extensive support of real-time mechanisms is very helpful for developing complex systems such as those found in on-board software.

Support for high-integrity software development Ada has a rich set of built-in restrictions (ISO/IEC) that can be applied in order to facilitate the use of software validation and verification techniques. The restrictions limit the use of some language features which, while contributing to the expressiveness of the language, may lead to unbounded or indeterministic behaviour. Examples of this are dynamic storage and unbounded type declarations.

Schedulability analysis techniques have a significant role in high-integrity systems verification. The Ada Ravenscar profile (Burns et al., 1998, 2003) restricts tasking to an analysable model that can be implemented by a lightweight kernel. The profile was incorporated into the Ada 2005 standard.

SPARK (Barnes, 2013; AdaCore, d) is a language intended for developing high-integrity systems which is based on Ada. It limits the language in the same sense as the native restrictions, and includes formal annotations that can be used with an extensive set of verification tools.

Hardware interfaces and low-level programming. Representation clauses and interrupt handling are powerful tools for addressing specific interfacing requirements as those arising when accessing hardware devices. Device drivers can thus be written in a safe way while keeping an abstract view of the hardware, and can be made compatible with the Ravenscar restrictions as shown by López et al. (2010).3

Mixed language development Ada supports mixed-language development by means of import and export pragmas, and representation clauses. In the case of the UPMSat2 software this is an important feature in order to integrate the ADCS control algorithm code, which is automatically generated in C from a Simulink model.

Based on above considerations, the UPMSat2 development team decided to use Ada 2005 with the Ravenscar profile tasking restrictions as the main language for the project.

2.2. Development tools

In order to properly support the use of of Ada and the project requirements, the GNAT Pro compilation chain for LEON3 (Ruiz, 2005) has been chosen, together with other Ada related tools, such as the GNAT Programming Studio (AdaCore, a), GNATcoverage (AdaCore, b), and CodePeer (AdaCore, c). The GNAT run-time system includes the Open Ravenscar Real-time Kernel, ORK, developed at UPM (de la Puente et al., 2000; Zamorano and Ruiz, 2003). ORK was designed so as to to keep its implementation as simple as possible, focusing on robustness and predictability.

3. PROJECT EXPERIENCE

The UPMSat2 project is still under development, and the on-board software system has not been completed yet. However, there is already some experience with the software development process and tools.

The software is being developed using a model-driven approach (de la Puente et al., 2014b). The production version of the software is being developed using the TASTE toolset (Perrotin et al., 2012) and the AADL(Feiler, 2012), ASN.1 (ITU, 2008), and SDL (ITU, 2011) as modelling languages. Simulink (Mathworks, 2013) has been used to model the attitude control algorithm, from which sequential C code has been automatically generated. The code is integrated into skeleton Ravenscar Ada code generated by TASTE (de la Puente et al., 2014a).

Figure 3 shows a high-level view of the software architectural model.

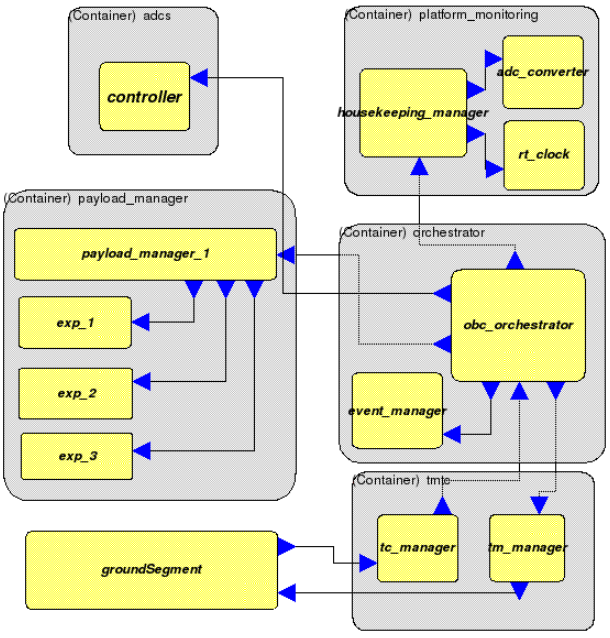

Figure 3. AADL model of the on-board software system.

An alternative experimental version for a partitioned, IMA platform has also been developed using UML-based tools (Salazar et al., 2014). The implementation approach is similar, with functional C code being integrated into Ada code skeletons generated from the high-level models.

The resulting source code has been fed into the compilation chain. Some functional modules have been directly written in Ada using GPS. Ada code, both handwritten and tool-generated, and C code generated from Simulink, have been integrated using the GPS environment. The interfacing features of Ada have been used extensively for this purpose, making it rather simple to build a multi language system.

The GNAT compilation chain has been used to produce a LEON3 executable code image, which can be loaded into the OBC hardware, and then tested and debugged, using the GRMON monitor (Gaisler Research, 2013).

We are currently in the process of verifying all the software subsystems, for which purpose the CodePeer tool is being of great help. The real-time behaviour is being analysed using WCET and response-time analysis tools (Garrido et al., 2012).

4. CONCLUSIONS

The main features that support the choice of Ada as the main programming language for the UPMSat2 project have been analysed. We believe that this choice is extensible to other small satellite projects, in which the robustness and tool availability can lead to highly reliable software systems at a reasonable cost.

The experience with several specific features of the language has been addressed, with promising preliminary results.

REFERENCES

AdaCore. *GPS User's Guide*, 2014a.

AdaCore. *GNATcoverage Users Guide*, 2014b.

AdaCore. *CodePeer User's Guide*, 2014c.

AdaCore. *SPARK 2014 Reference Manual*, 2014d.

John Barnes. *Ada 2005 Rationale*. Number 5020 in Lecture Notes in Computer Science. Springer-Verlag, 2008. ISBN 978-3-540-79700-5.

John Barnes. *SPARK - The Proven Approach to High Integrity Software*. Altran, 2013.

Alan Burns, Brian Dobbing, and Georges Romanski. The Ravenscar tasking profile for high integrity real-time programs. In Lars Asplund, editor, *Reliable Software Technologies — Ada-Europe'98*, volume 1411 of *Lecture Notes in Computer Science*, pages 263–275. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64536-8. doi: 10.1007/BFb0055011.

Alan Burns, Brian Dobbing, and Tullio Vardanega. Guide for the use of the Ada Ravenscar profile in high integrity systems. Technical Report YCS-2003-348, University of York, 2003.

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.

Juan A. de la Puente, Jorge Garrido, Juan Zamorano, and Alejandro Alonso. Model-driven design of real-time software for an experimental satellite. In Edward Boje and Xiaohua Xia, editors, *Proceedings of the 19th IFAC World Congress*, pages 1592–1598. IFAC-PapersOnLine, 2014a.

Juan A. de la Puente, Juan Zamorano, Alejandro Alonso, Jorge Garrido, Emilio Salazar, and Miguel A. de Miguel. Experience in spacecraft on-board software development. *Ada User Journal*, 35(1):55–60, March 2014b. ISSN 1381-6551.

ECSS. *ECSS-Q-ST-80C Space Product Assurance — Software Product Assurance*. European Cooperation for Space Standardization, March 2009. Available from ESA.

Peter Feiler. *Architecture Analysis & Design Language — SAE AADL AS5506B*. SAE, 2012.

Peter Fortescue, Graham Swinerd, and John Stark. *Spacecraft Systems Engineering*. Wiley, 4 edition, 2011.

Gaisler Research. *GRMON User's Manual*, 2013. Available at http://www.gaisler.com/doc/grmon.pdf.

Gaisler. *LEON3 - High-performance SPARC V8 32-bit Processor. GRLIB IP Core User's Manual*. Gaisler Research, 2012.

Jorge Garrido, Daniel Brosnan, Juan A. de la Puente, Alejandro Alonso, and Juan Zamorano. Analysis of WCET in an experimental satellite software development. In Tullio Vardanega, editor, *12th International Workshop on Worst-Case Execution Time Analysis*, volume 23 of *OpenAccess Series in Informatics (OASIcs)*, pages 81–90. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012. ISBN 978-3-939897-41-5.

ISO/IEC. *TR 15942:2000 — Guide for the use of the Ada programming language in high integrity systems*, 2000.

ITU. *Abstract Syntax Notation One (ASN.1)*, 2008. Recommendations ITU-T X.680–683.

ITU. *Specification and Design Language – Overview of SDL-2010*, 2011. Recommendation ITU-T Z.100.

Jorge López, Ángel Esquinas, Juan Zamorano, and Juan A. de la Puente. Experience in programming device drivers with the Ravenscar profile. *Ada User*, 31 (2), June 2010.

Mathworks. *Simulink*, 2013. URL `www.mathworks.com/products/simulink`.

Maxime Perrotin, Julien Delange, Andre Schiele, and Thanassis Tsiodras. TASTE: A real-time software engineering tool-chain overview, status, and future. In Iulian Ober and Ileana Ober, editors, *SDL 2011: Integrating System and Software Modeling*, volume 7083 of *Lecture Notes in Computer Science*. Springer, 2012.

J. P. Rosen. What orientation should Ada objects take? *Commun. ACM*, 35(11):71–76, November 1992. ISSN 0001-0782. doi: 10.1145/138844.138849. URL `http://doi.acm.org/10.1145/138844.138849`.

José F. Ruiz. GNAT Pro for on-board mission-critical space applications. In Tullio Vardanega and Andy Wellings, editors, *Reliable Software Technologies — Ada-Europe 2005*, volume 3555 of *LNCS*. Springer-Verlag, 2005.

Emilio Salazar, Alejandro Alonso, and Jorge Garrido. Mixed-criticality design of a satellite software system. In Edward Boje and Xiaohua Xia, editors, *Proc. 19th IFAC World Congres*, pages 12278–12283. IFAC-PapersOnLine, 2014.

S. T. Taft, R. A. Duff, R. L. Brukardt, E. Plöedereder, and P. Leroy, editors. *Ada 2005 Reference Manual. Language and Standard Libraries. International Standard ISO/IEC 8652:1995/Amd 1:2007*. Number 4348 in Lecture Notes in Computer Science. Springer-Verlag, 2006. ISBN 978-3-540-69335-2.

Juan Zamorano and José F. Ruiz. GNAT/ORK: An open cross-development environment for embedded Ravenscar-Ada software. In Eduardo F. Camacho, Luis Basañez, and Juan Antonio de la Puente, editors, *Proceedings of the 15th IFAC World Congress*. Elsevier Press, 2003. ISBN 0-08-044184-X.