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 fulfill 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 $0.5 \times 0.5 \times 0.6$ m, as shown in figure 1, and a total mass of $50$ kg approximately. The satellite will be set on a low Earth noon synchronous polar orbit (Fortescue et al., 2011), with an expected altitude of $600$ km and a period of $97$ 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 started 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 behavior. 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).

**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 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 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.](image)

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 written 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


