A Real-Time Computer Control Platform for an Experimental Satellite*

Juan A. de la Puente Juan Zamorano Alejandro Alonso
Daniel Brosnan

Grupo de Sistemas de Tiempo Real
y Arquitectura de Servicios Telemáticos,
Universidad Politécnica de Madrid (UPM)

1 Introduction

UPMSat-2 is a project aimed at developing an experimental micro-satellite that can be used as a technology demonstrator for several research groups at UPM, the Technical University of Madrid. The project is led by IDR/UPM, the "Ignacio Da Riva" Institute for R&D activities in the field of space science, microgravity and engineering,[1] and is being carried out with the participation of some other University and industry groups that are active in the Spanish aerospace sector. The objective of this project is the design, construction and qualification, launch and on-orbit operation of a qualified space platform incorporating new technologies and adaptable to different launch services. UPMSat2 has been defined as micro-satellite in the 50 kg class.Its geometrical envelope is a 0.5 m-side cube (figure 1). A 600 km sun-synchronous orbit has been devised for the satellite. A provisional launch date has been set for April, 2013, and the intended life of the mission will be 2 years.

The Real-Time Systems Group (STRAST/UPM[2]) will be responsible for designing and building all the on-board and ground-segment software for the satellite. The group has a long time experience in space-related software projects, and has developed the Open Ravenscar Real-time kernel (ORK) (de la Puente et al., 2000), which supports the Ada Ravenscar profile (Burns et al., 2004). The Ravenscar profile is a subset of Ada tasking features that enables software developers to build predictable, reliable real-time programs using high-level abstractions. It is included in the current Ada standard (ARM05).

The STRAST group is responsible for developing all the on-board and ground segment software for the satellite. This paper is focused on the software architecture of the on-board software, with special emphasis on the software platform aspects. Section 2 describes the on-board computer high-level requirements. Section 3 describes the architecture of the on-board computer system, both at

*This work has been partly funded by the Spanish Ministry of Science, project no. TIN2008-06766-C03 (RT-MODEL).

[1] `www.idr.upm.es`.

[2] `www.dit.upm.es/str`.

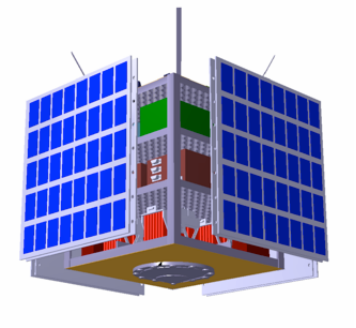

Figure 1: UPMSat2 external view.

the hardware and software levels. Section 4 summarizes the methodological aspects of the on-board software development. Finally, some conclusions and a summary of the remaining work to be done are recorded in section 6.

2 On-Board Computer requirements

2.1 On-board computer functionality

All the computer-related functions on board of the UPMSat2 satellite are executed on a single computer platform, called the On-Board Computer (OBC). Its main functions are:

Attitude determination and control (ADCS). The attitude of the satellite (i.e. its orientation with respect to the Earth surface) is computed from the signals provided by a set of three sensors that measure the intensity of the Earth magnetic field on the platform reference axes (magnetometers).

The attitude is controlled most of the time by means of a passive mechanism, consisting on three sets of permanent magnets that tend to align with the Earth magnetic field. When this is not enough, a set of three magnetic coils (magneto-torquers) are used as actuators in order to adjust the attitude to the required angle values.

Telemetry and telecommand management (TMTC). The satellite communicates with the Earth station by means of two radio links. Telemetry data (i.e. information sent by the satellite to ground) are coded and packaged by the OBC software according to standard protocols, and sent to the communications hardware (TMC) in order to enable the ground operators to monitor the operation of the satellite and its orbital parameters.

Telecommands are produced by the ground equipment and sent by radio to the satellite. Upon reception by the communications hardware, they are forwarded to the OBC for decoding and execution.

Platform monitoring and control. This function includes periodic operational and navigational data measurements (e.g. temperatures and voltages at various points, orbit and attitude parameters), which are sent to

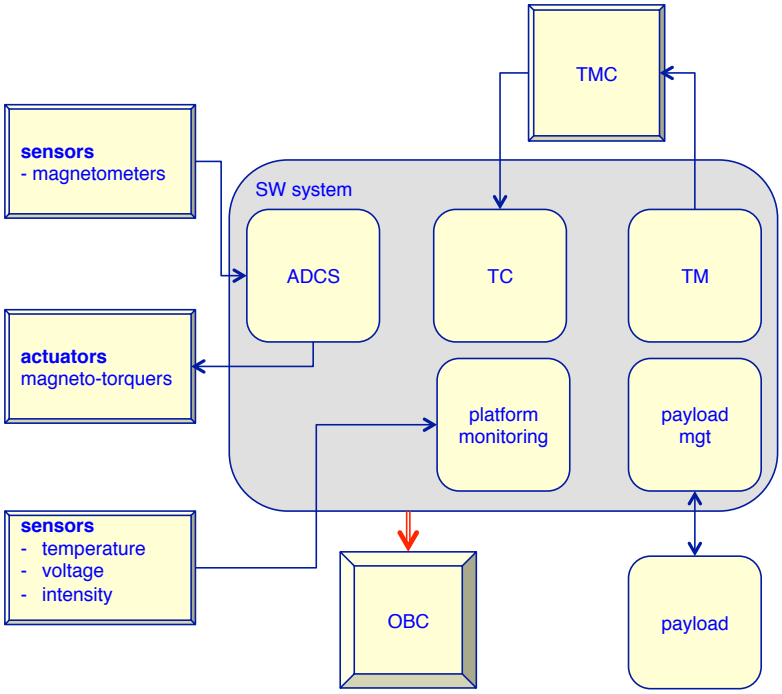

Figure 2: OBC context diagram.

the ground station by telemetry. Monitoring functions also include detecting some possible incorrect situations and signalling them to ground.

Basic platform control functions (e.g. power supply and temperature) are mostly carried out directly in hardware. The computer functionality consists on setting references and changing operation modes when necessary (e.g. low battery voltage).

Fault detection, isolation, and recovery (FDIR). One important function is to detect and recover from faults of the computer system itself. The approach to FDIR in the project is mostly based on constructive mechanisms, e.g. decoupling of software modules, software cheeks and redundancy.

Payload data management. The payload is the part of the satellite that carries out the mission objectives. Possible payloads for the UPMSat2 mission include an Earth observation system and a marine emergency surveillance system.

Figure 2 shows a context diagram of the system including its main functionality.

2.2 Software requirements

Based on the required functionality, a software system specification document (SSS) has been produced, according to the standards commonly used in the European space industry (ECSS40). The document also includes some non-functional requirements, including temporal requirements for some functions,

and the requirement to use Ada and open source software to the largest possible extent.

3 Computer system architecture

The hardware architecture is based on a LEON3 computer[3] implemented on an FPGA, with a clock frequency of 266 MHz. The computer board includes 1 GB of SDRAM, a 1 GB solid state disk, and a number of analog and digital input and output lines.

The software architecture is based on the functional components of the system, as depicted on figure 3.

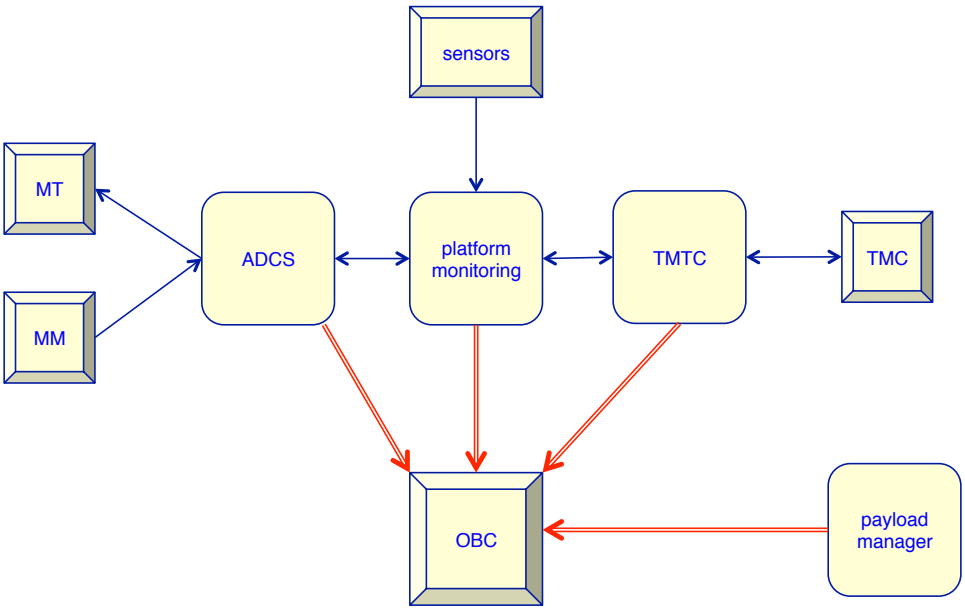

Figure 3: UPMSat2 software architecture.

4 Software development methodology

4.1 Model-driven software engineering

Following previous experience in the ASSERT[4] and CHESS[5] projects, a model-driven engineering approach (Schmidt, 2006; Zamorano and de la Puente, 2010) will be used to develop the on-board software system. Accordingly, the system is first described as a high-level platform-independent model (PIM), which is later transformed to a platform-specific model (PSM), including all implementation

[3]See `www.gaisler.com`.

[4]Automated proof-based System and Software Engineering for Real-Time systems. FP7 IST IP004033. `http://www.assert-project.net`

[5]Composition with Guarantees for High-integrity Embedded Software Components Assembly. Artemis JU grant no. 216882. `www.chess.project.ning.com`.

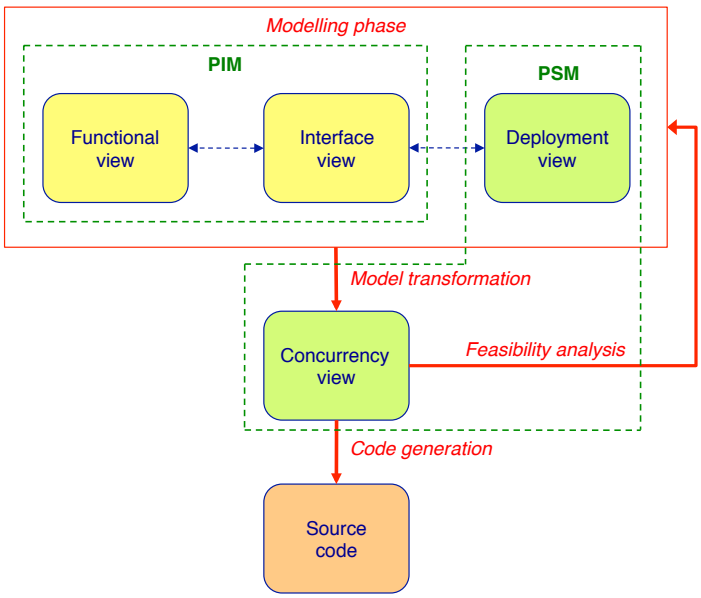

Figure 4: Software development process.

details. The system is implemented by automatically generating code from the PSM. Figure 4 shows the main elements of the development process.

An important aspect of such a process is the emphasis on separation of concerns. The functional aspects of a system are modelled first, as a set of components connected by functional interfaces. Non-functional aspects, such as timing properties, are specified as annotations of the corresponding operations.

Concurrency and real-time behaviour is added at the PSM level, which implements the PIM by embedding functional behaviour in containers that exhibit pre-determined properties. Examples are periodic and sporadic tasks, and protected and unprotected passive data objects. A key aspect of the PSM is that Ada source can be generated for all kinds of containers (see e.g. Pulido et al., 2007; Panunzio and Vardanega, 2011). Another important point is the availability of an execution environment that can guarantee that the behaviour exhibited by the PIM model is really implemented at the platform level. GNAT/ORK+ is an example of such an execution environment (de la Puente et al., 2008).

5 Functional model of the UPMSat2 ADCS

The above ideas have been applied to the development of the attitude determination and control subsystem of the satellite. In order to keep up with the general practice of control engineers, the attitude dynamics and the controls system have been modelled using Simulink,[6] as part of the platform-independent model. This high-level model has been used to develop the ADCS code by carrying out the following steps:

[6] `www.mathworks.com/products/simulink`.

- C sequential code for the control algorithm has been automatically generated from the dynamic model using the appropriate Simulink toolkit.
- Sequential code is embedded into a cyclic container at the PSM level, using TASTE tools.[7]
- Final code for the full subsystem is generated from the PSM, again using TASTE tools.
- The code is uploaded to the computer board for testing. The Simulink hardware-in-the loop facilities can be used to test the subsystem and validate its operation against a dynamic model of the satellite attitude.

6 Conclusion

The UPMSat2 project is an excellent occasion for our group to apply recent research on real-time software engineering to a real space mission. Although the scope and size of an academic system is limited, it is still a real development which will give us the opportunity to show that the ORK+ technology and modern software engineering methods can be successfully applied in such a demanding kind of applications.

We expect to devote the next months until the launch date to completing the development of the software, and to carry out all the necessary verification and validation activities that will ensure that the software is correct in its final form.

Acknowledgments

The authors wish to acknowledge the collaboration of the rest of the UPMSat2 team, especially Eduardo Serantes, Manuel Rodríguez (RETEMSA), and Ángel Sanz, Gustavo Alonso, Ali Ravanbakhsh, and Assal Farrahi (IDR/UPM).

References

ARM05. *ISO/IEC 8652:1995(E)/TC1(2000)/AMD1(2007): Information Technology — Programming Languages — Ada.*

Alan Burns, Brian Dobbing, and Tullio Vardanega. Guide for the use of the Ada Ravenscar profile in high integrity systems. *Ada Letters*, XXIV:1–74, June 2004. ISSN 1094-3641. doi: http://doi.acm.org/10.1145/997119.997120. URL `http://doi.acm.org/10.1145/997119.997120`.

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.

[7] Downloadable from `www.assert-project.net/-TASTE-`.

Juan A. de la Puente, Juan Zamorano, José A. Pulido, and Santiago Urueña. The ASSERT Virtual Machine: A predictable platform for real-time systems. In Myung Jin Chung and Pradeep Misra, editors, *Proceedings of the 17th IFAC World Congress*. IFAC-PapersOnLine, 2008.

ECSS40. *ECSS-E-ST-40C Space engineering — Software*. European Cooperation for Space Standardization, March 2009. Available from ESA.

Marco Panunzio and Tullio Vardanega. Charting the evolution of the Ada Ravenscar code archetypes. In *Proceedings of the 15th International Real-Time Ada Workshop*, 2011. to appear on ACM SIGAda Ada Letters.

José Pulido, Juan A. de la Puente, Matteo Bordin, Tullio Vardanega, and Jérôme Hugues. Ada 2005 code patterns for metamodel-based code generation. *Ada Letters*, XXVII(2):53–58, August 2007. Proceedings of the 13th International Ada Real-Time Workshop (IRTAW13).

Douglas C. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2), 2006.

Juan Zamorano and Juan A. de la Puente. Design and implementation of real-time distributed systems with the assert virtual machine. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1 –7, sept. 2010.