PRISE: An Integrated Platform for Research and Teaching of Critical Embedded Systems

Jérôme Hugues – ISAE/DMIA

Pierre Siron – ISAE/DMIA, ONERA/DTIM

Alexandre Hamez – ISAE/DMIA

Abstract

In this paper, we present PRISE, an integrated workbench for Research and Teaching of critical embedded systems at ISAE, the French Institute for Space and Aeronautics Engineering. PRISE is built around state-of-the-art technologies for the engineering of space and avionics systems used in Space and Avionics domain. It aims at demonstrating key aspects of critical, real-time, embedded systems used in the transport industry, but also validating new scientific contributions for the engineering of software functions. PRISE combines embedded and simulation platforms, and modeling tools. This platform is available for both research and teaching. Being built around widely used commercial and open source software; PRISE aims at being a reference platform for our teaching and research activities at ISAE.

1. Introduction

The Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) is a large public institution of scientific, cultural and professional missions with an expertise in the field of aerospace engineering. ISAE is under the supervision of the General Directorate for Arms of the French Ministry of Defense. It was founded in 2007 as the result of the merging of SUPAERO (founded in 1909) and ENSICA (founded in 1945), two French Graduate Schools (Grandes écoles). It has a 420 permanent staff. Its mission is to provide higher Programs in order to train engineers highly qualified in the aerospace sectors and related fields; dispensing specialized teaching, improvement and updating of knowledge; conducting scientific research and technological development work.

Inside ISAE, the Department of Mathematics, Computer Science and Control (DMIA) concentrates research driven in the multidisciplinary scientific fields of mathematics, computer science and control theory. The DMIA aims to develop methods, techniques and tools that make it possible to understand, analyze, evaluate, control and design, the functional and operational behavior and the performances of complex systems. The complexity of the systems studied by the DMIA encompasses their physical laws, their dynamic behavior, time and resource constraints, large size, and the diversity of their components and spatial distribution of their functions. All these concerns are personified as part of our research on embedded systems.

The building of embedded systems involves the combination of many domains of expertise to cover a large development cycle: from requirements elicitation, functional design and allocation onto a hardware architecture; and then testing, verification and validation activities. As an education and research, ISAE is leading several activities to build its own in-house expertise on such system to advance its teaching programs (engineering curriculum and advanced master programs), but also its research agenda. The PRISE project (Plateforme de Recherche et d'enseignement d'Ingénierie des Systèmes Embarquée, or Pedagogical and Research Infrastructure for Software-intensive Embedded systems) is our research vehicle to meet two complementary objectives so as to provide

1. A platform for our teaching activities on software-intensive embedded systems;
2. A test bench for our research activities, combining modeling, analysis and simulation of complex systems

In this paper, we report on our experience in designing this platform, presenting our initial needs and the actual implementation, as well as future directions. In section 2, we present the requirements for the PRISE platform and how it could be beneficial to our students and research teams. In section 3, we present the Distributed Simulation test case we develop for the analysis of advanced control functions as part of an avionic test bench. In section 4, we present the Star Tracker experiment, an experiment to implement a space-qualified system. Section 5 concludes the paper by providing future works and perspectives.

2. Requirements for the PRISE platform

The PRISE project started 5 years ago as an ISAE internal initiative to build a platform that is representative of good engineering practice in the field of the space and aeronautics domain. Being an applied science and engineering school, ISAE has a strong focus on technological transfer from research to advanced R&D projects to serve the aerospace industry. DMIA application domains cover both space and aeronautics system. PRISE being defined as a support for software-intensive systems, this imply we shall cover activities from early requirement elicitation down to simulation, implementation and analysis (performance, etc.).

The key objective when defining the PRISE platform is that it shall be representative of the many activities, domains involved. Furthermore, it shall support both research and teaching activities. This obviously put a strong pressure on how large such platform could be.

Elicitation of high-level requirements

Hence, we defined the following sets of requirements, for which we provide a rationale, and an early analysis on the impact on the design of PRISE. The main requirement is that PRISE is both a software and hardware platform. It has to gather all required elements in an integrated manner. From these high-level requirements, we derived the following considerations:

- [R1] The platform shall support most steps of the typical V cycle for engineering of embedded functions.
  - Rationale: The V cycle is a recurrent pattern in our teaching and research activities, each member of our team covers particular subsets of this cycle. This fosters collaboration in our teams.
- [R2] The platform shall integrate tools to support activities
  - Rationale: Tools increasingly support this cycle: requirements engineering (Reqtify, DOORS, KAOS, etc.), modeling (SysML, UML, AADL), languages (Simulink, C, Ada), RTOS (RTEMS, VxWorks, etc.), analysis tools (model checking, scheduling analysis).
  - The key difficulty is to define proper levels of integration and cooperation between tools. We illustrate this aspect in the next sections.
- [R3] The platform shall cover both space and avionics domains, this implies two sub platforms with convergent tool support and hardware
  - Rationale#1: Hardware are radically different, with different technologies being involved: avionics rely more on PowerPC CPUs and AFDX networks; while space systems prefer LEON3/4 processors and SpaceWire networks.
  - Rationale#2: Although having representative hardware is a stringent requirement, it has a significant cost, but it is a key requirement to demonstrate on an actual test bench that provides a complete coverage of the V cycle up to deployment.
  - Fortunately, there are some similar concerns when modeling systems (e.g. Simulink, UML tools) or analysis (model checking, scheduling).

Implementation of PRISE

From these requirements, we defined PRISE around three pillars:

Embedded platforms: these platforms cover the low-end part of the V cycle, providing all support for running and testing software: real-time operating systems (RTEMS, Xenomai, FreeRTOS, VxWorks), analysis (WCET

computation, code quality); hardware platform for the space (LEON boards from AeroFlex Gaisler) and avionics (PowerPC SBC8349 reference boards). These hardware elements rely on field buses (SpaceWire, CAN and Ethernet) for interworking.

Simulation platform: it is based on the IEEE HLA standard, for testing software on-board systems in a system-level simulation, integrating software in a physical environment for assessing its performance. A typical example is an autopilot controlling a plane in a flight simulator.

Modeling tools: a set of tools has been selected and combined for the engineering of these functions, enabling a systemic approach, using the OMG SysML and SAE AS2-C AADL standards, down to its implementation. This covers analysis (schedulability analysis, model checking for time and safety properties, dimensioning), code generation following industry standards: (ECSS-E-40B, RT-POSIX, etc.).

In addition to industry leading tools such as SCADE Studio, Matlab Simulink or DOORS, ISAE also develops and maintain its own set of tools. Over the years, the PRISE team at ISAE developed and maintains several open-source tools with academic partners to cover all these steps: TASTE/Ocarina for AADL model manipulation, in partnership with ESA, TTool, for SysML models with Telecom ParisTech; CERTI with ONERA for HLA. These tools are used to model representative use cases that are deployed either on the simulation or embedded platforms.

Usage of the PRISE platform

Mastering the complete set of platforms and tool chains is a key asset of the PRISE platform and its team at ISAE. It allows us to deploy PRISE for Research and Teaching at ISAE, and perform new experiments.

Teaching: student in our engineering and master programs can have a full view of either particular functions of an embedded onboard function (embedded platforms), the global system (simulation), or associated modeling and analysis tools. We deployed successfully PRISE in teaching activities, covering more than 50 students/year, and 200 hours or lab or project sessions. PRISE is used for validating research contributions for the engineering of complex software, and published in various conferences..

Research: PhD candidate and researchers at ISAE can test new functions, or interconnection between tools and platforms to ease the engineering of critical complex software. It also helps defining or refining part of these tools for better usability, coverage of functions and increase its maturity and TRL level. We discuss these experiments in the next sections for two case studies: simulation of avionics systems; engineering of a spatial subsystem.

3. Distributed Simulation of Embedded Systems

In this section, we present one of the facets of the PRISE platform: the simulation of avionics system based on the HLA simulation standard. We present the context of this project, and the hardware and software elements we combined in PRISE to support this project.

Interoperability Standard

Distributed computing paradigm proposes a high performance solution thanks to advances in network technologies. Different programs located on several computers interact all together in order to achieve a global common goal. However, designers and developers of distributed software applications have to face several problems such as heterogeneity of the various hardware components as well as both operating systems and communication protocols. Development of middleware standards allows to consistently facing these problems. The term middleware describes a software agent operating as an intermediary between distributed processes. This software must be considered in the domain of interoperability; it is a connectivity software that enables the execution of several interacting applications on one or more linked computers. This interoperability allows the building of more complex systems; the reusability of components for many studies and the flexibility by making easy the replacement of a component by a new one.

Distributed Simulation and Distributed Application

Simulation is a well-established technique used in the man-machine system area for training, evaluation of performance and research. Flight simulation re-creates how an airplane flies in its environment; it models the dynamic behavior of the flight vehicle under the action of aerodynamic, thrust and gravity forces, accordingly to the external environment characteristics (air density, wind, turbulence...). To achieve this goal, a flight simulator consists of different components. The essential one still remains the mathematical description of the aircraft and its environment; the more accurate the model, the more realistic and reliable the simulation will be. Then a digital computer running a real-time operating system computes this model. The simulation can finally be completed with input organs (e.g. yoke-pedal systems, joysticks), display screens, cockpit-like environment and mechanical devices reproducing the aircraft motion (e.g. Stewart platform).

We claim that the choice of a distributed standard and its underlying middleware is an important requirement to obtain a high fidelity, valid and scalable real-time flight simulation. This choice implies which operating system, which programming language and which hardware could be used for compliance with the middleware. Many studies and integration simulations are elements of the industrial process but the different models are proprietary (and sometimes certified) as well as the Run Time Infrastructure. We choose the HLA standard [1].

HLA is a standard for distributed simulations, initially proposed by the US Department of Defense, then accepted as an IEEE open standard (IEEE 1516). Reuse and interoperability are fundamental goals of HLA. Complex simulations (called federations) are realized by interconnecting simpler simulators (called federates). These federates actually connect to a simulation kernel called RTI (runtime infrastructure). Federates manage common objects and exchange information as defined in the HLA specifications.

The HLA specifications consist of a set of rules about federates and federations, an object model for the federates (and particularly for public objects and interactions appearing in the federations), and a set of services required to manage objects and interactions. Some of these services have to be provided by the federates to the RTI, others have to be provided by the RTI to the federates. Particularly, federates publish and subscribe to relevant object and interaction classes, so as to be able to create such entities or to receive information about them. Another characteristic is the time management services, which allow deterministic, and reproducible distributed simulations.

We also claim that the distributed simulations look like to the real applications, because they have in common:

- The same modularity requirements.
- Very similar parts of codes.
- The use of a middleware and specific execution platforms.
- Real time constraints.

Hardware and Software Platform

To study new embedded system concepts and techniques, we need a special hardware (computers and network) and software (operating system and middleware) environment.

For the hardware part, we are using four real-time nodes with Opteron 6-core processors, two Graphical HP station computer with Intel Xeon processors and high performance GP-GPU, an Ethernet Gigabit switch on a dedicated network and also two input organs (Yoke/Throttle/Pedal systems). The operating system-id is Linux Red Hawk [2], its kernel has been configured to be strictly compliant with POSIX Real time standard [3]. A distributed clock technology allows the distribution of the same clock reference to each node

For years, the French Aerospace Laboratory (ONERA) has been developing his own Open-Source middleware RTI compliant with HLA standard called CERTI [4], running under several operating systems including Linux and Windows. We will use this RTI for interconnecting each component of the simulator. This RTI is recognizable through its original architecture of communicating processes. In our case, a key benefit of this architecture is to

master the implementation of the used RTI and thus to facilitate the integration of changes in the source code to ensure temporal predictability of CERTI. Initial results, providing some answers about the suitability of CERTI to face real-time constraints, came from ONERA/CNES satellites formation flying studies [5]. These studies have shown that CERTI (in its original version) is able to manage multiple real-time federates with short period of time.

Although this is a specific a dedicated platform, this platform is made of generic elements and open-source software. It can be easily replicated and the cost is affordable for the study of realistic avionics systems.

Simulation Architecture

In the context of PRISE, we developed a particular federation that represents a full airplane, at a reduced complexity level. The PRISE HLA Federation SDSE is composed of 11 federates, each representing a specific part

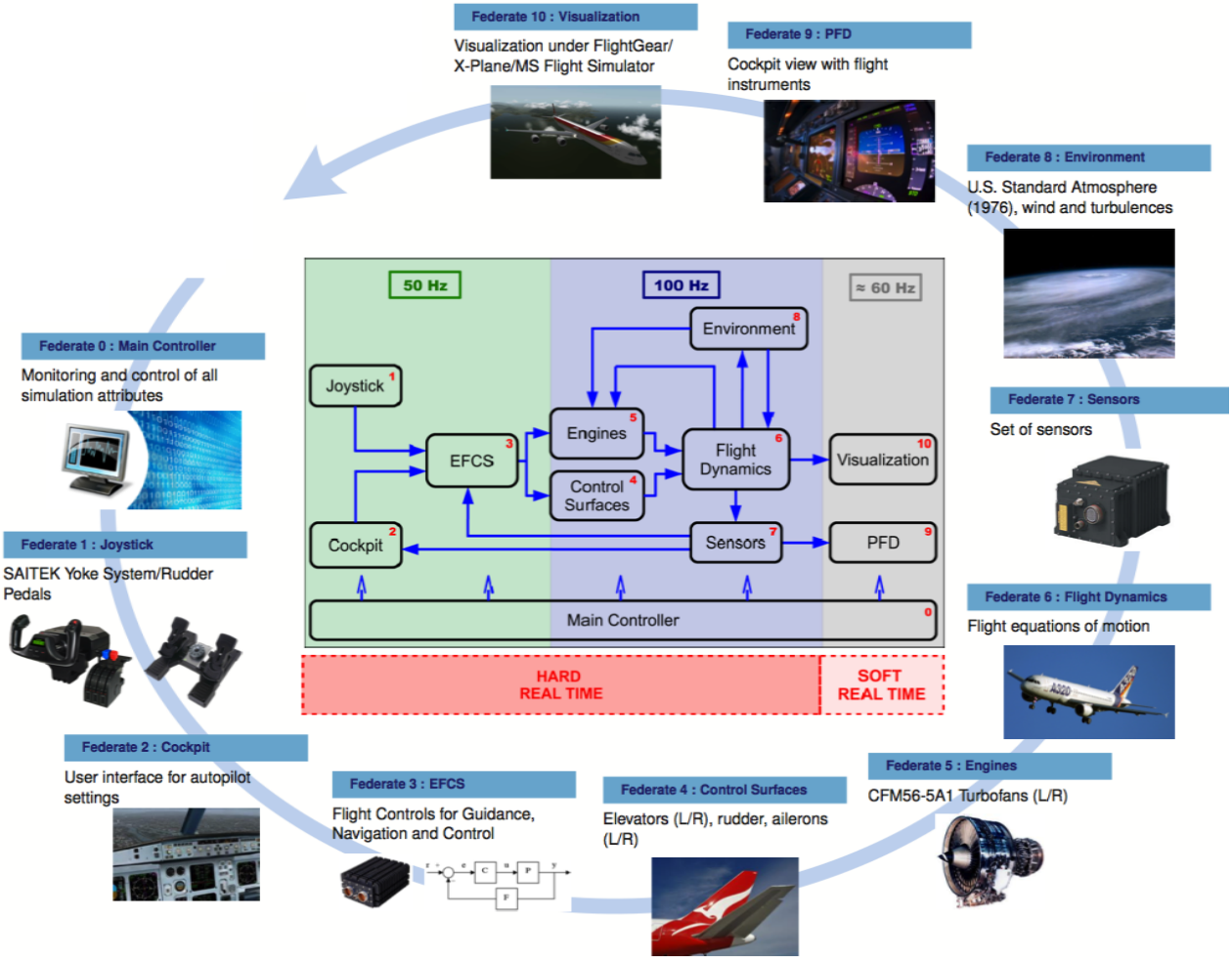

Figure 1: SDSE Architecture

of the aircraft environment (Fig. 1).

New federates will be incorporated accordingly to the evolution of the requirements. For now, the federates go as follows:

- Federate 0: Main controller monitors and controls all simulation attributes;
- Federate 1: Joystick relays pilot inputs coming from the yoke/pedals device;

- Federate 2: Cockpit emulates user interface for autopilot settings via a touch screen;
- Federate 3: EFCS (Electronic Flight Control System) encompasses all flight controllers and automatic pilot functions;
- Federate 4 Control surfaces simulates the five different control surfaces of the airplane: left/right ailerons, left/right elevators, rudder;
- Federate 5 Engines simulates two CFM56-5A1 turbofans;
- Federate 6 Flight Dynamics simulates the flight mechanics equations;
- Federate 7 Sensors simulates twenty different sensors of various kinds;
- Federate 8 Environment implements models of the US standard atmosphere (1976) and of different turbulences/winds (Dryden, Von Karman, windshear) that could occur during the flight;
- Federate 9 PFD (Primary Flight Display) is a cockpit view with flight instruments;
- Federate 10 Visualization shows the aircraft in a virtual environment (FlightGear visualization engine, but as well Microsoft Flight Simulator or X-Plane).

Real Time Constraints

Modern flight simulation techniques and implementations often result in many sophisticated and complex calculations that require a high level of computing power. Several flight simulator applications often require their services to be delivered with respect to a given instant of time (deadline). This issue constitutes the problematic of real-time systems which are defined as systems in which the correctness of the system not only depends on the logical results of computation, but also on the time at which these results are produced [6]. Real-time systems are broadly classified into two categories based on the nature of the deadline, namely, hard real-time systems, in which the consequences of not executing a task before its deadline may be catastrophic and soft real-time systems, in which the utility of results produced by a task with a soft deadline decreases over time after the deadline expires. Examples of typical hard real time systems are flight control and nuclear plant control. Telephone switching system and image processing applications are examples of soft real-time systems. Figure 1 shows that our application is concerned by both types of real-time system characteristics: hard real-time for the control logic of the plane, soft real-time for some display GUIs.

The calculation of Worst Case Execution Time (WCET) is a key issue for successfully schedule processes. Calculation of the WCET should take into account specific calculations made by the federate. WCET were estimated for each federate. Moreover, Worst Case Transit Time (WCTT) was calculated for each message through CERTI middleware.

Current CERTI version does not provide any service or mechanism to ensure a real-time behavior of a simulation (federation). To manage every part of federation and to be compliant with formal techniques and scheduling techniques, different methods were added to the CERTI API (for Linux Operating System). The different implemented services also ensure a correct predictability for CERTI communications (WCTT) and federate computation (WCET).

We first implemented functions in CERTI that allow using affinity mechanism. CPU affinity is a scheduler property that assigns a process (federate, RTIA or RTIG) to a given set of CPUs on the system. The Linux scheduler will honor the given CPU affinity and the process will not run on any other CPU. Another interface allows now the management of priority and scheduler for CERTI processes (including federates, RTIAs and RTIG). Modification of priority relies then on the choice of real-time scheduling algorithms under POSIX/Linux: two real-time algorithms, SCHED FIFO and SCHED RR, are intended for time-critical applications that need accurate control over the way in which runnable processes are selected for execution. Finally, we also use the mlockall() system call on each federate, their respective RTIA and the RTIG processes to disable memory paging in the calling process.

Perspectives

The SDSE project has required the mastering of many aspects: from the realistic implementation of avionics code and environment models to realistic interfaces, the extension of HLA and CERTI distributed simulation to real time. This simulator looks like to a flight simulator, built with a fully different and interoperable technology.

SDSE illustrates how one single platform can help in many missions of ISAE:

- For training it mimics the components of a real system (at the global and detailed levels).
- For teaching, we plan to study other federations with a set of autonomous and collaborating embedded avionics and ground systems, illustrating challenges in simulation of complex systems.
- For new researches, it will be easy to change the version of an existing federate by another (if its WCET is compliant with the global approach) in this modular and flexible architecture. It will also be easy to add new federates with an evolution of the global object model of the federation (HLA FOM) and a new real time analysis. We hope also that the defined FOM becomes a reference FOM for this research domain.

4. Qualification of a space system

In this section, we introduce a second test case built at ISAE: the design and implementation of a star tracker subsystem. We introduce the mission objectives of this system, its design elements and the qualification steps on our roadmap to integrate it to a mission to be launched in 2015.

Star Tracker, mission and objectives

ISAE is involved in several research projects for the space domain in conjunction with ESA and CNES, and their prime contractors, from mission analysis, definition of high-definition imaging sensor, control command laws and their implementation. PRISE plays a natural role in this setting by supporting hardware/software design and implementation activities to prototype some of the elements.

A Star Tracker (STR) is an optical device that measures the position of star using a camera. From the picture and a dictionary of stars, an algorithm may determine the attitude (or orientation) of a spacecraft. This valuable information may in turn be used to control the position of the satellite through the Attitude Orbital Control System (AOCS). Hence, a Star Tracker exemplifies an embedded system with a significant computation part to be embedded in a harsh environment in space.

ISAE was involved in the ESEO and ESMO projects prior to their cancellation by the funding agencies. The role assigned to ISAE students was to define and implement such Star Tracker, but also the accompanying qualification materials. This project is currently being rescheduled in the context of the EU FP7 project QB50, and would ultimately lead to the STR being embedded in a CubeSat in Q2 2015.

We decided to make this project part of PRISE, since we already supported many of the building blocks required: embedded processor platforms, (LEON3, a space-qualified variant of the SPARC processor), real-time operating systems (RTEMS, selected by ESA as its reference operating system) and communication stacks (CAN bus).

In addition, since ISAE is also in charge of some early testing and qualification materials, PRISE can provide additional support, as we need also to implement a test bench for this STR. We defined a solution based on a dark chamber, optical devices (lenses, camera) and a tablet serving as a display of the stars (see Figure 2). This bench also relies on many elements part of PRISE: network, space boards and associated tools.

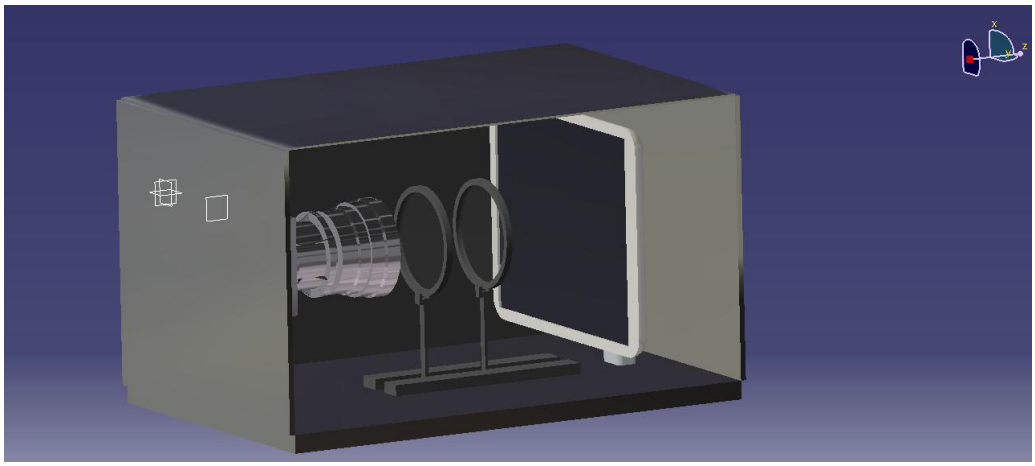

Figure 2: Star Tracker system in its test bench

Implementation of the STR and its test bench

The implementation of the STR is currently an ongoing project performed by our students, with support from professors. Projects in the previous years led to the full design of the system. The objective of this year project is to complete the implementation and evaluate its performance using the test bench. These steps involve following the ECSS-E-40B standard for the software part.

PRISE helps in many dimensions of the project, by providing a set of tools already integrated

- Hardware: reference implementation of LEON3 processors: GR-XC3S-1500 and GR-RASTA from AeroFlex Gaisler, Spacewire field bus for connecting the cameras to the board, as well as Android based tablets;
- Software: Real-Time Operating Systems (RTEMS), WCET analysis tools (Bound-T), scheduling tools (Cheddar), modeling and simulation tools (Matlab Simulink)

One obvious advantage of PRISE in this setup is that all tools are already setup and installed for our student. This greatly reduces the configuration and time for getting familiar with the system.

The implementation and qualification of the STR software revolve around many activities covered by typical activities of embedded systems implementation

1. Definition of the STR algorithm: ISAE designed an innovative method for computing the attitude, based on an optimized search algorithm [7];
2. This algorithm is implemented as Matlab functions, from which we derive automatically generated C code;
3. C code is then integrated in a real-time task sets supported by RTEMS from OAR Corp. [8];
4. Finally, validation and verification activities are supported by a combination of tools for Worst Case Execution Time analysis (Bound-T, [9]), Cheddar for scheduling analysis [10] and GNU gcc binutils for memory consumption and code coverage analysis.

The test bench is made of the following elements:

- A dark chamber to mimic deep space environment, without pollution fro Earth-level luminosity
- An optical bench to hold the camera, lenses and an Android-based tablet that will display the stars
- A simulator to display a portion the deep space sky, using a star catalog and a simulation engine to reproduce the dynamic of the satellite, and move the picture of the sky accordingly. Kinematics simulation of the satellite is done using the Orekit toolkit [11].

By combining software and hardware for the STR, and its associated test bench, students have all the material required to perform a full qualification of the STR logic prior to its integration to a CubeSat platform to be launched as part of the QB50 project. This project proposes to several universities to have their CubeSat be part of launch

scheduled in Q2 2015. This project is a great incentive for our students, as it provides a concrete opportunity to turn their knowledge into a real flying system.

This project is part of a long-term initiative to help at both a teaching vehicle, allowing future students to refine some elements (algorithm, architecture, analysis), or perform benchmarks. For research activities, it provides ISAE with a subsystem of a real satellite, fully qualified to later experiment with new design methods and compare them with the typical development approach presented here. Examples are system engineering activities driven by OMG SysML and the TTool environment we develop, or generation from Architectural Description Language AADL.

5. Conclusion and perspectives

In this paper, we presented the PRISE project, a federation of projects to support the engineering of software-intensive embedded systems. PRISE has been defined to cover extensively the various steps of the V-cycle, and is fully equipped by research and industry tools, but also a full range of hardware blocks.

The investment around PRISE at ISAE is quite important, allowing us to set up a full test bench and representative case studies for both the avionics and space domains. This was done thanks to the full definition of the required building blocks: hardware (high-performance computers, embedded boards, networks) and software covering all relevant steps of the V cycle. PRISE has been implemented by the careful combination of these blocks to provide relevant case studies that serve as basis for our engineering and master programs, but also to support activities in the DMIA department of ISAE.

From a teaching perspective, PRISE has been deployed successfully in our programs to serve as a realistic development and simulation environment, allowing students to better understand the various technical and technological challenges of complex avionics and space systems. PRISE is incrementally refined each year to expand various aspects: software engineering, validation and verification. Increments are chosen based on teaching objectives of the various program delivered at ISAE: understanding of system level behavior, analysis of particular functions, implementation of novel control functions, etc.

From a research perspective, PRISE serves as a platform to validate our contributions in the field of simulation middleware (CERTI project in collaboration with ONERA), modeling and analysis of systems as well as code generation (TTool and Ocarina projects).

Future increments of PRISE will focus on extending the range of our experiments, by investigatin multi-core architectures, safety assessment of complex software and system and the refinement of the modeling and simulation of the environment to interact with the various software blocks already in place.

Acknowledgments

The PRISE project has been funded partly thanks to DGA/MRIS support. PRISE is developed thanks to support from the LIA laboratory at ISAE. All teaching activities supported by the PRISE platform are supported thanks to academic license agreement that provides access to modeling, analysis and simulation tools, and hardware from Tidorum Ltd (Bound-T), Esterel Technologies (SCADE Suite), The MathWorks (Matlab), VwWorks (WindRiver).

We also thank all the students and PhD candidates that were involved in the implementation of PRISE experiments.

References

[1] The Institute of Electrical and Electronics Engineers (IEEE) Computer Society. *"IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification,"* September 2000.

[2] J. Baietto, *"Real Time Linux: The RedHawk Approach,"* Concurrent Computer Corporation White Paper, 2002.

[3] W. M. Corwin, D. C. Locke, and K. D. Gordon, *"Overview of the IEEE POSIX p1003.4 realtime extension to POSIX,"* IEEE Real-Time Syst. Newsl., vol. 6, pp. 9–18, March 1990.

[4] E. Noulard, J.-Y. Rousselot, and P. Siron, *"CERTI : an open Source RTI, why and how,"* Fall Simulation Interoperability Workshop, 2009.

[5] E. Noulard, B. d'Ausbourg, and P. Siron, *"Running Real Time Distributed Simulations under Linux and CERTI",* European Simulation Interoperability Workshop, Edimbourg, June 16-19, 2008.

[6] J. A. Stankovic, *"Misconceptions about real-time computing,"* IEEE Computer, vol. 21, no. 10, 1988.

[7] Jalabert, Eva and Fabacher, Emilien and Guy, Nicolas and Lizy-Destrez, Stéphanie and Rappin, William and Rivier, Guillaume *"Optimization of star research algorithm for ESMO star tracker."* (2011) In: 8th International ESA Conference on Guidance, Navigation and Control Systems - GNC 2011, 05 -10 June 2011, Karlovy Vary, Czech Republic.

[8] RTEMS Real-Time Operating Systems, OAR Corp, http://www.rtems.org/

[9] Bound-T Worst-Case Execution Time analysis tool, http://www.bound-t.com/

[10] F. Singhoff, A. Plantec, P. Dissaux and J. Legrand. "*Investigating the usability of real-time scheduling theory with the Cheddar project."*, Journal of Real Time Systems, volume 43, number 3, pages 259-295. November 2009. Springer Verlag. ISSN:0922-6443.

[11] OREKIT (ORbits Extrapolation KIT), https://www.orekit.org/