A DSML based Approach for simulating on-board Equipments in Space Applications

Sódor, Bálint(1) Tróznai, Gábor(1) Dr. Szalai, Sándor(2)

Affiliation(1)
Wigner Research Centre for Physics, Hungarian Academy of Sciences, HUNGARY
sodor.balint@wigner.mta.hu
troznai@wigner.mta.hu
Affiliation(2)
SGF Ltd., HUNGARY
szalai@sgf.hu

Abstract- The special nature of the spacecraft equipment development drives the necessity of extensively incorporating modelling and simulation into the development process. The utilization of interface level simulation results in earlier feedbacks from the applied concepts and increased reliability of the whole onboard system. On the other hand domains specific modelling languages (DSMLs) ensure the semantically correct and easy to use description of the spacecraft units' behaviour. In the MTA-Wigner FK (earlier KFKI-RMKI) and the SGF ltd. we decided to put together the knowledge gained in the decades on the field of space system development and build a conceptual framework of a platform and mission independent interface level simulator. The fundamental concepts was derived from the lander software simulator (LSS) implemented for the Rosetta mission. In this paper we present the conceptual architecture of a generalized interface level simulation framework for spacecraft onboard equipments along with the concepts of the underlying modelling language.

Introduction

In the MTA-Wigner FK (earlier KFKI-RMKI) and the SGF ltd. we have spent decades developing - along many other onboard systems - Electrical Ground Support Equipments (EGSEs) simulating onboard electrical interfaces for different space missions [5, 9]. For the European Space Agency's (ESA) Rosetta mission we have implemented the simulator of the Philae lander's onboard system (called Lander Software Simulator - LSS). This simulator uses a dedicated modelling framework and a special hardware module to perform signal level simulation of the onboard equipments.

In space system development the interface level simulation of on-board equipments has a special importance due to two main nature of the field. The space system developments are done parallel by a number of different – often geographically separated - development teams. While on the other hand the application of special hardware elements and the lack of maintainability of these systems drives the necessity to ensure the reliability of the system components as well as the whole system.

As the model of an equipment serves as the base of the simulation, one of the key issues is the implementation of the unit model. A well defined and carefully implemented Domain specific modelling language (DSML) provides the opportunity not only to easily build unit models on the necessary level of abstraction but to use formal or semi-formal methods for model verification as well.

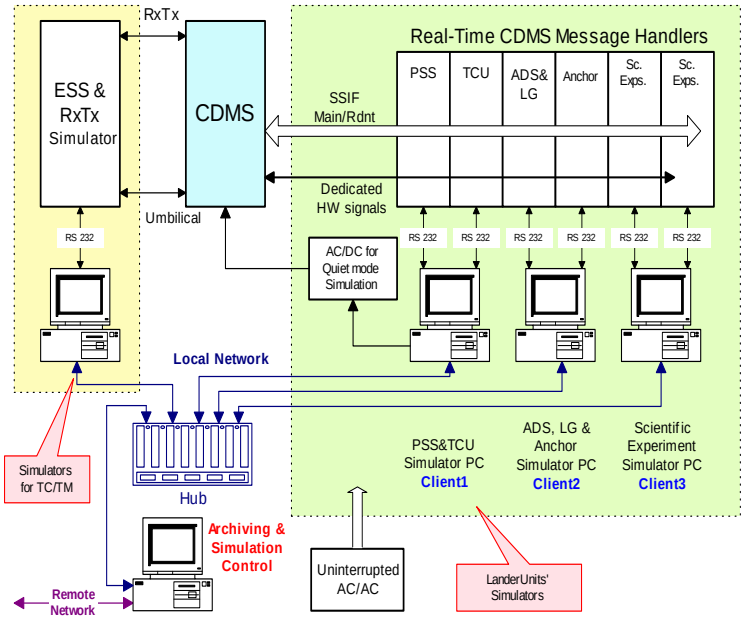
Based on our experiences we have defined a system class for the onboard systems used in space missions. A spacecraft contains a number of scientific instruments and subsystems (units) communicating with a central control and acquisition unit. These onboard units usually contain some internal processing and communication logic implemented on a dedicated processor card or FPGA built up by space classified hardware elements. The maintainability and reparability of onboard systems are strongly restricted only the software components can be updated partly. Due to the low bandwidth and long delay of the communication channels these systems often has to react autonomously for the environmental stimulus's. Based on the above characteristics of the area, our statements about the identified system class are the following:

- These systems are strongly distributed reactive autonomous systems.
- Embedded, fault tolerant and highly reliable – mission critical – systems.
- These systems often have only very limited processing storing and communication resources.

In the following section we introduce the problem statements in the field of onboard system development from our point of view by briefly describing our participation in two actual missions. In later sections we present our conceptual framework designed for solve a number of the introduced problems.

LSS

The Rosetta mission was approved in November 1993 by European Space Agency (ESA). The aim of the mission is to approach and observe the comet Churyumov-Gerasimenko. The space-probe started its 10 years long journey in 2004. It consists two parts one is the orbiter the other is a lander module. The later one contains a highly parallel onboard system built up by 6 subsystems and 9 scientific modules – developed parallel by a number of different groups – and the Command and Data Management Subsystem (CDMS). During the mission our responsibility was to develop the hardware and software components of the CDMS [1,2].



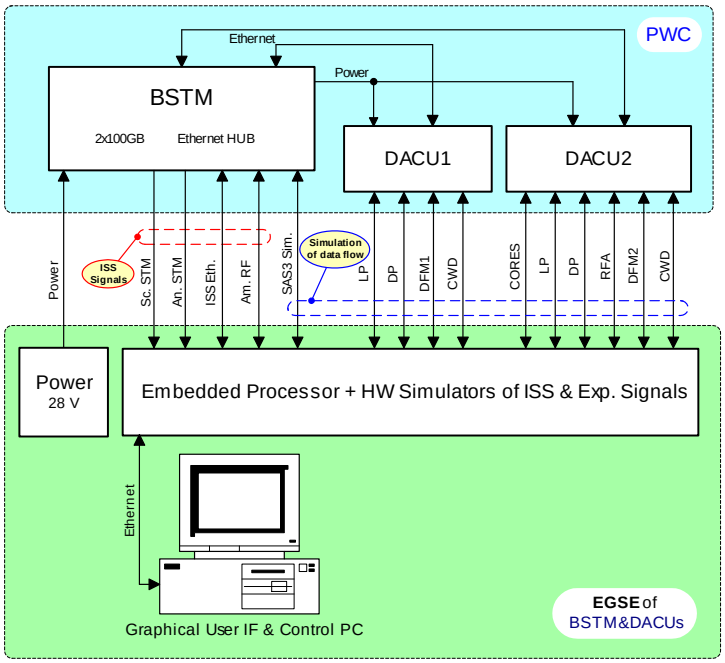
1. Figure: LSS architecture

During the mission we were facing the following main problems:

- We have to develop the central control and processing unit of a highly distributed system built up by many independent reactive units. The development of the CDMS and the other subsystems controlled and monitored by the CDMS is done parallel at the same time.
- Due to the long term of the mission and the reduced maintainability – only the software can be partly updated – we have to ensure the reliability robustness and fault tolerance capability of the CDMS. The fault tolerance examination arise the necessity of an environment where faults can be injected and fault propagation can be investigated. That can be very hard task in a system where not all of the units are ready.
- The CDMS is based on hardware components whit very limited resources.

To provide some solution for the above problems we have prepared the Lander Software Simulator (LSS) which is a parallel hardware in loop simulator system built around a real implementation of the CDMS (figure 1.). The onboard equipments of the Rosetta lander connected to the CDMS are simulated based on their behaviour models. The low level and high speed simulations of the different equipments are accomplished by unique developed hardware elements. These are the Real-Time Message Handlers, which are connected to the simulation system. The distributed simulation system consists of a number of desktop computers running the subsystem simulators and driving the message handlers. The LSS is also responsible for controlling and monitoring the simulation workflow and to provide a global environment for the simulated onboard system as well.

PWC



2. Figure: PWC development architecture

The Plasma Wave Complex (PWC) experiment is planned to be placed in the Russian segment of the International Space Station (ISS). It consists of a distributed data acquisition

and control system and 11 scientific subsystems (figure 2.). In the development phase of the Plasma Wave Complex experiment (PWC – Russian name is Obstanovka) our responsibility is to implement the distributed central computers [10]. We have to develop an EGSE system for simulating the interfaces of the science instruments as well. This EGSE provides the simulation and test environment for the central computers. Besides that we faced nearly the same problem classes as with the Rosetta Lander CDMS a few more problems emerged. Among this one of the major was the continuously changing specifications of the scientific instruments and the requirements not only at the early phase of the development but at the later ones as well.

Concept of a general simulation framework

As the introduced problems emerged during the two above mission are not unique ones, a generalized approach is necessary to provide a good mission and platform independent solution. We started to elaborate the concepts of a generalized simulation framework mainly based on the LSS. We have kept and refined the structure of the LSS but performed major modifications on the unit modelling and simulation layer. Following this paper presents the conceptual diagram of our framework later it describes the expected benefits of it.

Architecture

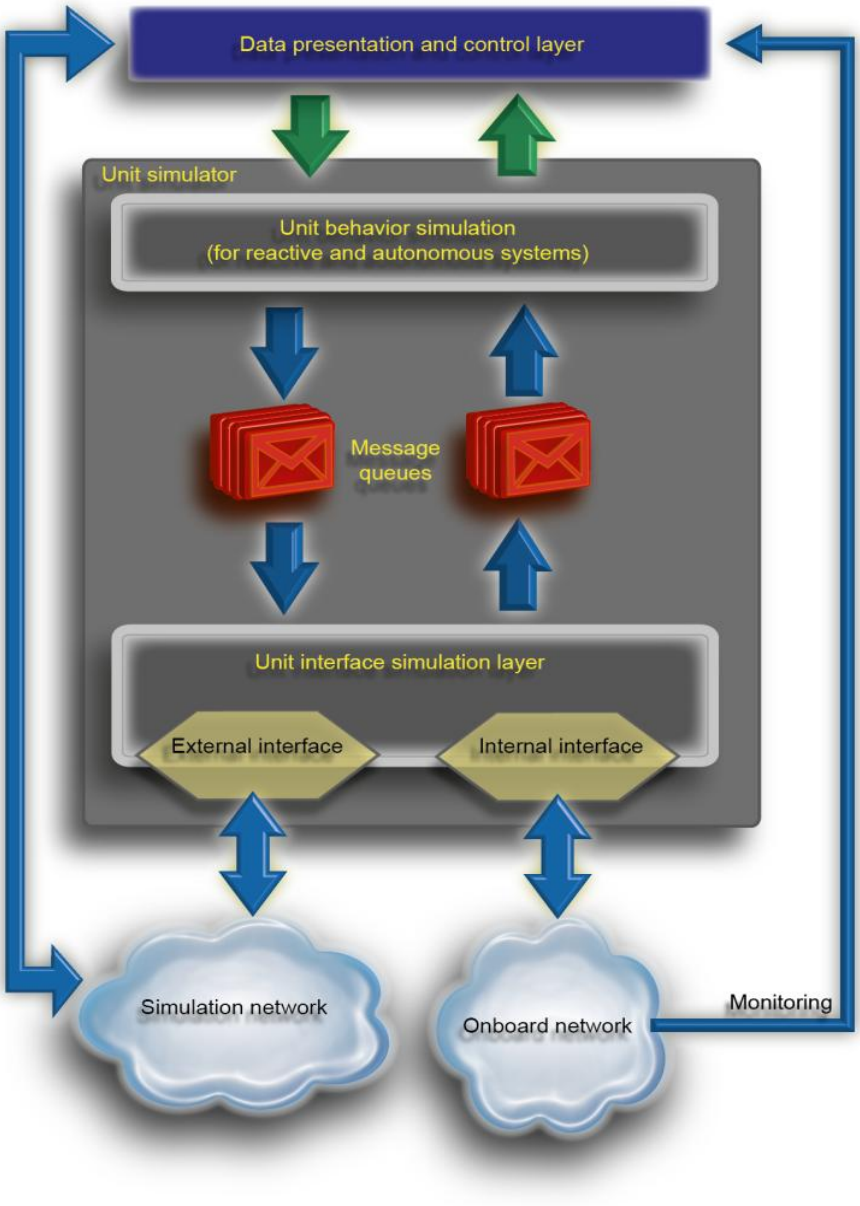
Our concepts of the simulation framework can be described by a 5 layer architecture where all the layers have its own well defined responses (figure 3.). From a bottom-up point of view our system is built up by the following layers:

- Physical layer – unit connectors – responsible for interconnecting the onboard units.
- The unit interface simulator layers main responsibility is to convert the uniform internal communication data structures into the physical layers communication entities.
- The message broker layer – message queues – responsibility is to serve as buffers for the communication messages and to ensure the chronologically correct message delivery.
- The unit simulation layer performs the simulation of the reactive and autonomous behavioural logic of the onboard equipment based on its domain specific model (DSM).
- The topmost layer is the data presentation and control layer which serves as the main interface between the user and the simulation environment.

In our vision one of the main benefits of the above structure is that: starting from the bottom ones the layers are dynamically interchangeable with the real onboard systems in different phases of the development.

Physical layer

The physical layer interconnects the simulated units. It contains the same data and control buses and independent control wires as the onboard system. It provides the identical signals and timing conditions as the fling spacecraft. Despite its main role at the very beginning phase of the development this layer may serve as a transparent communication tunnel for transferring the simulation data structures between the simulated units – for example standard Ethernet network.



3. Figure: Simulation framework architecture

Interface simulation layer

This layer transforms the unified internal communication structures into the message entities used by the physical layer. The upper part of the layer serves as the message broker client. It transfers the messages to and from the message broker layer and handles them to the lower part of the layer which is divided into two modules:

- The internal interface is connected to the physical layer thus it handles the communication between the different onboard units. This part provides the appropriate hardware connectors signal levels and driver circuits.
- The external interface is a logical interface for the control layer and receives the simulated environmental parameters (for example: stimulus for the detector of the scientific unit or the value of an internal temperature sensor and so on...)

The benefit of this layer is that it is independent from the simulated unit. It only depends from the onboard communication interface used during the space mission. Furthermore as nowadays the spacecrafts often uses standardized onboard data and control buses – like space wire, CAN bus, sometimes RS-422 or Mil-1553 connections, ... – and the digital or analogue p2p onboard connections can be generalized this layer supports the mission independency of the simulation framework.

Message broker layer

The message queues are derived from the event queues of the discrete event simulation (DES) concept [6, 7]. It contains the messages transferred between the interface and behaviour simulation layers. Its main responsibility is to maintain the causal order of both the incoming and outgoing messages. Using this layer results in the capability to switch between real-time simulation and virtual time simulation.

Unit behaviour simulation layer

The unit simulation layer is the most complex layer in or simulation environment. As its main responsibility is to run the simulation over the model of onboard equipment, this is the only unit dependent layer. This dependency can be reduced by utilizing the benefits of a well-defined and suitable modelling language used to describe the behaviour of the unit on the proper abstraction level. From our point of view the well-definedness of the modelling language means that the semantics of the basic entities of the language are defined correctly. A suitable language should meet the requirements of the space systems area. Our requirements against the modelling language are the following:

- The language should be capable to describe the behaviour of a reactive autonomous system including the logical interface description and the internal computation logic.
- The semantics of the building blocks should be well defined. Only this can ensure that formal and semi-formal methods can be used for running model checking and correct simulation over the unit model.
- As the behavioural description of an autonomous reactive system is often easier by distinguishing different internal states the language should support state machine based description. The behaviour description in a specific state of the unit should support activities, conditional statements, and the definition of periodical-, sequential- and parallel control flows. Furthermore the activities should be divided into two main classes: internal activities describe the state changes while external activities describe the communication flow of the unit.
- The state machine based description should distinguish between nominal and non-nominal states for support the fault injection and fault propagation investigations.

The general modelling languages like UML are often lack of correct semantic definitions. We decided to implement a so called domain specific modelling language (DSML) [8] which satisfies all of the above requirements and can be easily maintained and used. The fundamentals of this modelling language have already been introduced [4].

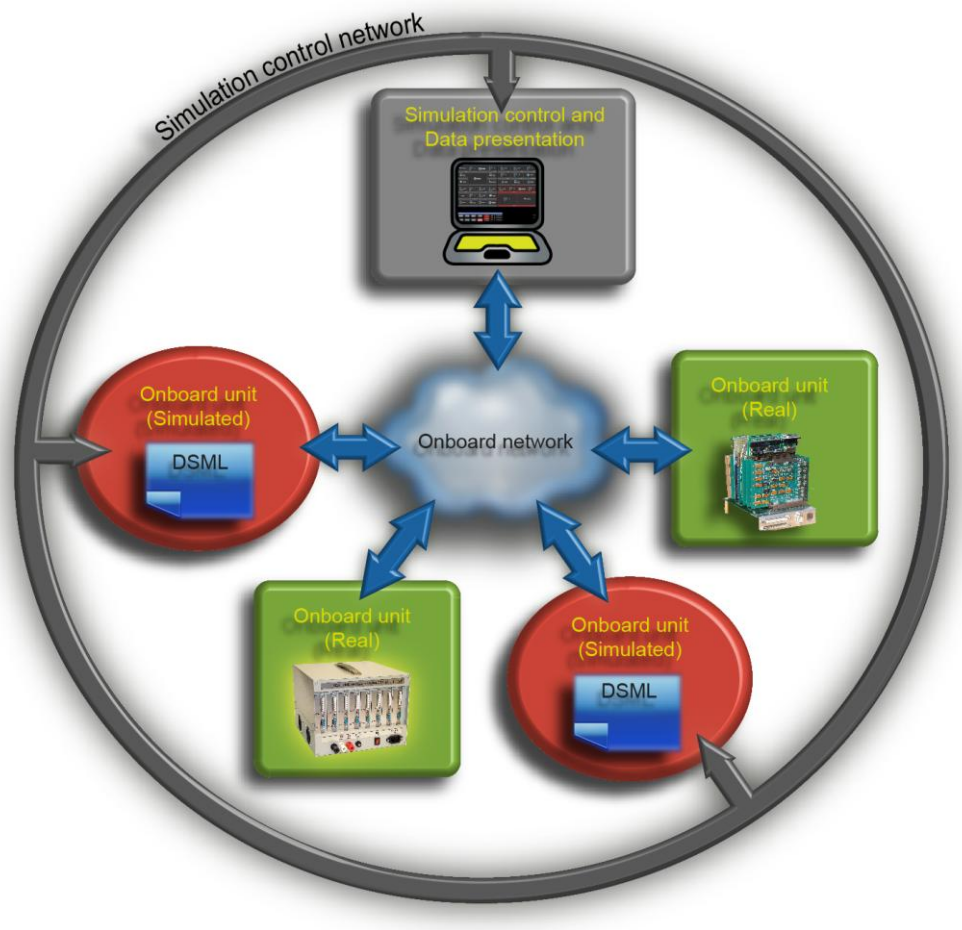
Data presentation and control layer

It serves as the main interface between the user and the system. Its main functions are the following:

- Provide access to the internal states of the units to the user.
- Visualize the current state of the simulation.
- Controlling the simulation.
- Distributing the environment related data among the simulated units.

Possible use-cases

To present our idea and desired expectations of the application of the described framework we provide a few theoretical use-cases in the following. The use-cases are divided by the different phases of a space mission related development.



4. Figure: Simulation environment

Early phase

During the design phase of the development the modelling language can be used to formalize the specifications and fundamental concepts of the system. After the preparation of the model the simulation layer can be used for checking and verifying the behaviour of the applied concepts. If other development teams are preparing they models the whole framework can be used to test and simulate the collaboration of the different units. At this point not any hardware components are required. The simulation layer runs the unit model while the messaging and interface simulation layers provides a quasi transparent communication service over a public network (such as the Internet). One benefit – among

many other – of this use-case is that the load on the communication infrastructure can be measured in very early stages of the development.

Mid-phases of the development

At later phases when the on-board communication infrastructure is already defined using the appropriate interface simulators the interface model of the unit can be presented based on the model prepared in earlier phases. As the interface simulators are unit dependent equipments no hardware development is necessary for presenting an interface level model for the on-board instrument.

Later in the hardware and software implementation phase the framework can be used as a testing environment if the other participants provide the model for their units. It is supported by the frameworks ability to dynamically replace a unit model with its actual implementation (figure 4.).

Conclusion and further work

So far this paper presented our concept of a simulation framework which utilizes domain specific modelling for aiding the whole system development process in space missions. The framework has many and more aspects which have to be investigated in the future. Although we have implemented parts of some layers – we have elaborated the basics of our DSML – there is still some works to do. We have to modify the implementation of our simulation layer and we have to examine different technologies to select the most efficient ones.

References

[1] G. Tróznai, A. Baksa, S. Szalai, B. Sódor: Rosetta Lander Software Simulator, 57th, International Austronautical Congress Valencia, Spain – 2-6 October 2006, IAC-06-D1.P1.11

[2] G. Tróznai, A. Baksa, S. Szalai, B. Sódor: Spacecraft Simulator for Philae, 9th, International Workshop on Simulation for European Space Programmes, Noordwijk, The Netherlands – 6-8 November 2006

[3] B. Sódor, G. Tróznai, Cs. Lipusz: Implementing Data Presentation Layer In Testing And Simulation Environments using XML 58th, International Austronautical Congress Hyderabad, India – 24-28 September 2007, IAC-07-D1.I.11

[4] B. Sódor, A. Baksa, A. Balázs, S. Szalai, G. Tróznai: New Approach to Modelling Spacecraft Modules. 58th, International Astronautical Congress Glasgow, Scotland 29-september – 3-october 2008

[5] Balajthy K., Lipusz Cs., Sódor B., Dr. Szalai S.:A földi ellenőrző berendezésekben alkalmazott programozási technikák. Híradástechnika, 2008/04 pp. 35-40

[6] Fishman, G. S.: Principles of Discrete Event Simulation. Wiley, New York, 1978.

[7] William Delaney, Erminia Vaccari: Dynamic Models and Discrete Event Simulation. Dekker INC. 1988.

[8] Kelly, S. and Tolvanen, J-P.: Domain-Specific Modeling: Enabling full code generation. John Wiley & Sons, ISBN 978-0-0470-03666, 2008, 427 p.

[9] Cs. Lipusz, A. Balázs, B. Sódor, G. Tróznai, J. Sulyán, K. Balajthy, Z. Pálos, S. Szalai: Examples of Reuseability of Concepts and Development Results. International Workshop on Managing Knowledge for Space Missions, Passadena – 17-19 July 2007.

[10] S.I. Klimov, V.E. Korepanov, Y.V. Lissakov, O.V. Lapshinova, I.V. Sorokin, S.A. Belyaev, G.A. Stanev, K. Georgieva, B. Kirov, M.P. Gough, H.S.C.K. Alleyne, M. Balikhin, J. Lichtenberger, Cs. Ferencz, L. Bodnár, K. Szegő, S. Szalai, J. Juchniewicz, H. Rothkaehl, K. Stasiewicz: Space Weather Workshop: Space Weather Applications Pilot Project. 16-18 December 2002, ESTEC, Noordwijk The Netherlands, Proceeding