# Simulation for all components, phases and life-cycles of complex space systems

Fernand Quartier, Frédéric Manon

Spacebel, Technoparc 8, Rue Jean Bart, 31670 Labège, France
fernand.quartier@spacebel.be
Centre National d'Etudes Spatiales, 18, Avenue Edouard Belin, 31401 Toulouse, France
frederic.manon@cnes.fr

**Abstract.** This paper describes the use and evolution of discrete event simulators and models throughout CNES, its various space system developments, disciplines and related life-cycles and teams. Simulators and models are built in the first place to ensure that the organization improves it competences in a number of key areas. It presents how a careful federation of means, know-how and models using a bottom-up approach, will meet one day the top-down System of Systems approach.

**Keywords:** Discrete event simulators, life cycle, multi-disciplinary, functional simulation, space systems

## 1 Introduction

In a large engineering enterprise, such as Centre National d'Etudes Spatiales (CNES), there are many simulators used and developed. The most demanding is the operational simulator as it has to be representative for a satellite as seen from the ground and because it is used in many verification and qualification chains for control centres, mission control centres and payload control centres. For those qualifications, the real satellite is only used rarely as incurs very expensive operations with many constraints, while introducing risks on damage and planning. Moreover, testing with real satellites still has limited representativity and fault injection is even more cumbersome. Nowadays, the operational simulators *fly* many months before the satellite is launched.

The significant efforts to develop such large operational simulators have not only led to a better understanding of the problematic and to better technical solutions, as described in subsequent sections. It equally triggered the awareness of the value of models that contain part of the company's memory and its patrimony and a means of communication and specification of behaviour. The validation and qualification of models takes often much more resources than the development itself, so that reuse is much more rewarding than traditional reuse of software components. But most importantly, models and simulators are creating some sort of biotope that allow improv-

ing key competences and facilitates cooperation between people having various expertise and project roles.

## 2 Operational Simulators

### 2.1 Main Requirements

Operational simulators have the following key requirements:

- From the point of view of operators, the simulator should be indistinguishable from the real satellite
- Causality must be respected and all runs must be reproducible
- Failure, fault and reproducible noise injection without changing models
- Fine control and visibility on internals (introspection)
- Formal and automated procedures for model and simulator validation
- Save/restore of context to allow bypassing operational test lead-in times of several days
- Perennity guarantees for 15+ years: Linux, mainstream PC's, Open source versus COTS, heritage/reuse of 15 years

### 2.2 Content and Performance Requirements

- Independent models in C, Fortran, Matlab, Scilab, object format (industrial secret).
- Start script based model instantiations and connection of model variables without compilation (using naming database)
- Computer emulators are loaded with the production version of the ROM images (1750, ERC32, LEON, …)
- Performance: minimum is guaranteed real-time, 3 to n times real-time for increased productivity

Although the main content of an operational simulator revolves around its computer simulator, many disciplines are present: on-board software, command and control, guidance and attitude, mechanics, thermal, electric, power …

As an example, the Pleiades operational simulator contains:

- 200 models, model frequencies of 1 to 128 Hz
- 7 processor emulators, globally up to 80 million of OBSW instructions/sec
- Up to 200 events in scheduler
- 10.000 events per simulated second

Its performance is:

- minimum 2 times real-time
- 10 times real-time preferred (possibly with models that support reduced representativity)

– 100.000 events per executed second

The Argos study simulators contain 100.000 models and manage 200.000 events in the scheduler. They run 5 to 500 times the real-time speed, executing 500.000 events per second.

Large simulators tend to have separate specialized teams to

- Develop and validate models, covering various disciplines (mechanics, thermal, power, dynamics, …)
- Configure, integrate and validate simulators for the specific needs
- Deploy simulators for use in the various operational chains and execute the needed scenarios

### 2.3 Life Cycles

The life of a satellite simulator has many dimensions as can be seen in the pictures below.

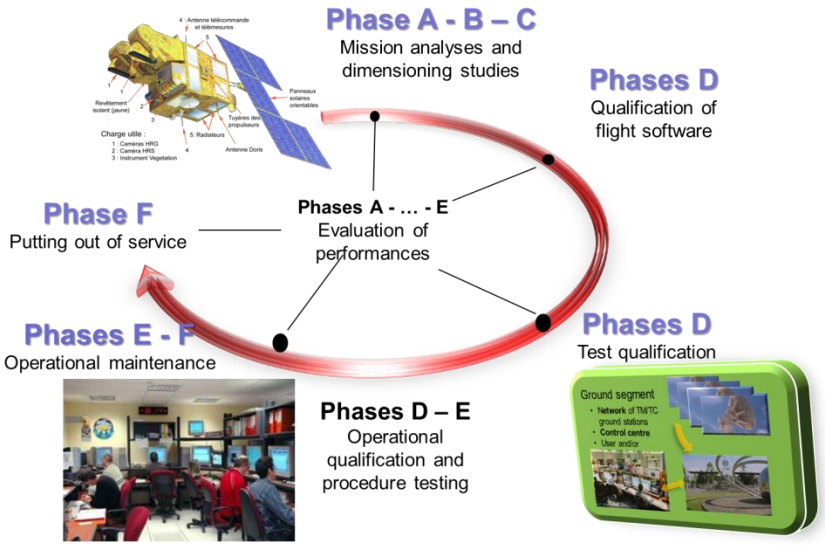


**Fig. 1.** Life cycles of a spacecraft

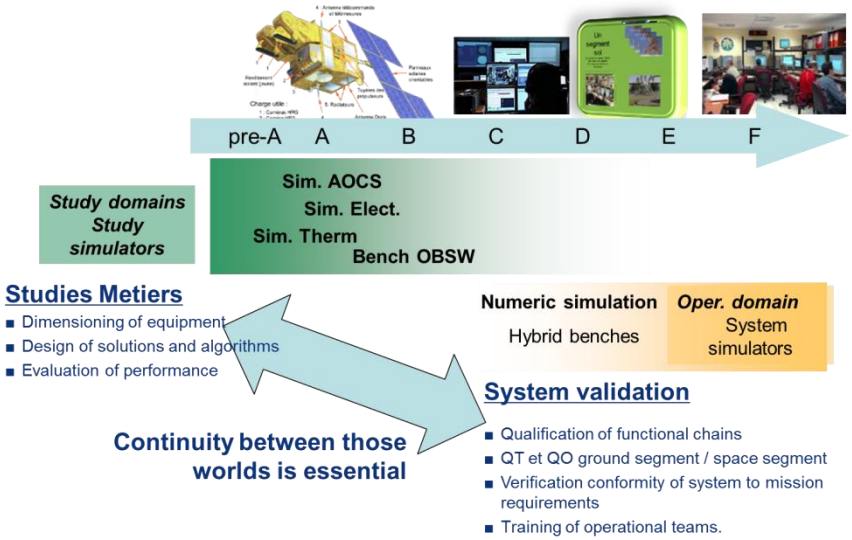


**Fig. 2.** CNES main simulator needs during a spacecraft life cycle

Other dimensions along the project phases are:

- Instruments that range from simple acquisition subsystems to complex instruments, such as GPS, star trackers, Gyros, …
- The several space platforms for the various product lines (mini-satellites, micro-satellites)
- Within operational phases, different configurations of the simulators are used, called variants. Typically, representativity and scope is reconfigured as to provide optimal performance for the tests at hand.

All these dimensions need a well thought out approach for testing, validation, configuration management and maintenance.

### 2.4 Integration with Other Components

Obviously, operational simulators need to have flexible interfaces to connect with the control and operational centres. It must be possible to route those interfaces directly or via the receiving station, through real RF equipment or through Spacelink simulators when representativity is paramount.

Co-simulation with other specialized simulators, such as Saber, is achieved through the use of standard interfaces, such as HLA. In the long run, hardware-in-the-loop will be needed for some components such as instruments and payloads.

## 3 Towards Better Use and Continuity of Means.

The development of operational simulators is on a crossroad where many project phases, disciplines, models and people come together. Nevertheless, it was observed that the re-use of models, know-how and tools was far from optimal. So it was further investigated.

### 3.1 Identified Problems and Barriers

The main identified problem is due to the partitioning barriers caused by the many dimensions of the life cycles, project teams, disciplines, platforms.

Building a simple discrete event simulator is not that complex, so that there are many such simulators developed throughout the company. As usually with software, those simple simulators evolve quickly to more complete in-house products and test environments. The more they evolve, the less the models tend to be reusable and the more difficult it becomes to move to a common platform.

### 3.2 BASILES

To improve the situation, in a first phase, BASILES (BAncs SImulateurs et Logiciels d'Etude de Satellite) has been created. It is a common simulation platform to promote models and simulation reuse among space programs and among the different simulators that are created during the lifecycle of a project. BASILES provides a methodology and a standard for CNES simulators.

First of all, BASILES is a simulation framework to develop, configure and run simulators. It allows representing complex systems using discrete event simulation. It contains the simulation kernel in charge of time and events handling, logger service, integrators, processor emulator management, distributed simulation handling, etc.

Concerning the development of a new simulator, BASILES features help to easily develop prototypes with basic programming knowledge in a short period of time and with a good level of accuracy. Models are simply configurable. Concerning the execution of a simulator, BASILES provides a great number of self-functionalities to interact and introspect the models and simulation.

Finally, BASILES is also a model library in order to share and reuse models and simulators among space programs.

In order to extend its user base, CNES accepts to attribute licenses of the product to other industries, thereby stressing the system more to achieve quicker full maturity and to expose the product to new user requirements and ideas.

### 3.3 SMP2

For several years now, the European Cooperation for Space Standardization (ECSS) has taken the initiative to develop the SMP standard (Simulation Model Portability). The aim of this standard is to allow models to be portable among different simulation

infrastructures. Interfaces are specified by SMP independently of simulation infrastructures.

BASILES evolved to the SMP standard and all new developments are SMP based.

### 3.4 Study Simulators

One of the other families of discrete events simulators is MACSIM, basically used as study simulator, and having a large patrimony of existing models. Study simulators tend to be developed starting with few and relatively basic models in an incremental and iterative way: the developer improves or refines the model, runs it, validates it and restart improving it. The MACSIM environment has been successfully integrated in BASILES.

### 3.5 Hardware in the Loop

Ideally, many of the models should be replaceable by hardware equipment, although this adds significant constraints. This allows using real equipment, to raise the level of representativity and expose the used models to a broader range of environments.

Such operations have been successfully performed integrating real payloads with the simulator via Mil 1553. The new Nosyca balloon flight computer has been integrated with BASILES via a number of interfaces. In that case, BASILES became a test bench, environment simulator and controller of the Nosyca flight computer.

### 3.6 Software Validation Facilities

BASILES has been augmented with non-intrusive flight software gdb debugging capabilities on the used processor emulators. That means that breakpoints can be set on specific instructions or data accesses. When such a breakpoint is hit, the clock of the processor and the simulator is frozen and the gdb interface is warned and normal debugging can take place. All external BASILES interfaces remain functional and time progress continues when the processor is released by the debugger.

### 3.7 Towards New Generation of Modular Real-time Benches

A demonstrator has been build that shows the distributed real-time capability of modern systems. It runs BASILES simulators on different mainstream PC's running standard Linux connected via HLA. Measurements have shown that all simulators were capable of generating output with a time precision and jitter that is better than 50 µseconds.

It is believed that test systems will become more modular and cheaper. In fact, many of the typical test systems are based on huge acquisition and driving front-ends, along with custom interfaces and uncommon processors and real-time RTOS with

specialised drivers. Such complex equipment creates a major constraint on re-use, maintenance and perennity.

For the Nosyca system, interfaces have been made using a series of small micro-controllers such as the PIC32, complemented with the needed connectors and small interface logic and shaping. These little 50€ low power boards (power via USB), with a 20 $cm^2$ footprint, contain a 80 MHZ CPU, significant memory and interface variety, including Ethernet. Because the microcontrollers are dedicated to one single function, they are simple, while in many cases, specific interface FPGA's can be avoided as the microcontroller can achieve a time resolution well below the µsecond. An approach that uses multiple small systems is better manageable than huge complex and hierarchical systems.

### 3.8 Defining Simulator Strategy. at Day One of Each Project

From the many experiments and domains BASILES has been used in, it became clear that a complete simulator planning is better studied by the very beginning of each space related project. As has been shown in the Argos and SMAR project, a first global system simulator allows for better dimensioning of many components of the system and helps to create a common understanding of the project.

## 4 On-going Developments and R&D

### 4.1 Thematic

There are several R&D projects and investigations going concerning microscopic traffic simulation (one model per car), Software Validation Facilities for Proba and MTg, missile test planning, FDIT management, TDM space communication and improved thermal simulation.

Indeed, precise thermal simulation used to be extremely processing hungry. CNES is in the process of developing fast thermal simulation technology that will allow simulating the thermal behaviour of major satellite components with a precision of a couple of degrees.

### 4.2 Parallel Processing

Parallel processing of several processor simulators has been proven as an important performance gain. Using the theory of "separability", developed at CNES, we are in a good starting point to engineer the parallelization. Currently, a methodology is being developed to detect model dependencies and allow for parallelization by configuration, without changing the models. This step can be taken when the normal non-parallel simulator is validated.

Another form of parallel work under investigation is the running of a simulator in parallel with the real system. The use would be twofold:

- In a first phase, to dynamically validate (and improve) the simulator versus the real system.
- In a second phase, to compare the real system against the simulator as to warn the operator when something is out of limits. Obviously, such system could have a far more refined warning capability than existing supervision systems.

A frequent context save of such systems would allow to jump backwards in time for deeper investigation of out of limit behaviour and perform what-if scenarios based on a saved context.

### 4.3 Processor Emulators

Processing emulators are the critical path in operational simulators, so significant efforts are devoted to them.

Current emulators decode each instruction to be executed, which limit their speed to around 70 MHz.

One trail concerns the dynamic translation or Just in Time compilation of flight software. It has been demonstrated that such emulators have the capability to reach 500 MHz emulation capability.

Another trail concerns the emulation of multi-core processors exploiting the multiple cores of the PC.

Another domain being investigated concerns the emulation of the space variant of ARINC 653 (also called TSP and IMA). In IMA, application layers are isolated in partitions that are time sliced by a hypervisor. Such partitions could probably simulated in parallel as by design, they have much fewer interdependencies.