Simulating Heterogeneous Constellations

Nicola Di Nisio[1]
ESA/European Space Operations Centre, Darmstadt, Germany

Christian Bodemann[2]
Vega Space GmbH, Darmstadt, Germany

and

Sonia Toribio[3] and Christian Schurig[4]
ESA/European Space Operations Centre, Darmstadt, Germany

The Galileo Constellation Simulator (CSIM) was developed to support training and FOP validation in the Galileo project. Started as a very demanding project on the industry side, with required support for the simulation of a single family of 30 spacecraft, each with different configuration and OBSW patch level, it evolved into an heterogeneous constellation simulator. Initially it will have to support two different families of spacecraft, from two different primes. Later on a third family will be added, possibly quite different from the first two. The current version of the CSIM, meant to serve the first phase of the project (In Orbit Validation) is now a stable product, validated both formally and by comparison with experience gained on the real spacecraft. The two families of spacecraft already defined in the program share a large percentage of components, and this may reflect in possible high source code reuse in the development of the CSIM to support the second family of spacecraft. However this might come at the expenses of breaking existing trusted and validated functionality and thus a whole range of trade-offs can be considered in many areas of the CSIM code and development. Possibilities range from the two extremes of maximum software reuse to complete code branching. In the first case we maximise engineering and revalidation efforts, minimising maintenance costs, while in the second case we can expect the contrary. This paper and the presentation will evaluate the possible solutions, with pros and cons and then indicate the selected one, and the criteria leading to that decision. On the user side, of particular interest is the configuration of a constellation for specific training and validation scenarios. A constellation editor is provided, along with tools to update Epoch and Orbit of each spacecraft, but other techniques used during the first IOV launch will be reviewed.

I. Introduction

THE Galileo Constellation Simulator (CSIM) was developed to support training and FOP validation in the Galileo project.

Started as a very demanding project on the industry side, with required support for the simulation of a single family of 30 spacecraft, each with different configuration and OBSW patch level, it evolved into a heterogeneous constellation simulator. Initially it will have to support two different families of spacecraft, from two different primes. Later on a third family will be added, possibly quite different from the first two.

[1] Data System Manager, HSO-G, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany
[2] Project Manager, Space, Europaplatz 5, 64293 Darmstadt, Germany.
[3] Procurement and AIV Manager, HSO-G, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany.
[4] Data System Manager, HSO-G, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany.

II. CSIM IOV Version

The current version of the CSIM, meant to serve the first phase of the project (In Orbit Validation) is now a stable product, validated both formally and by comparison with experience gained on the real spacecraft.

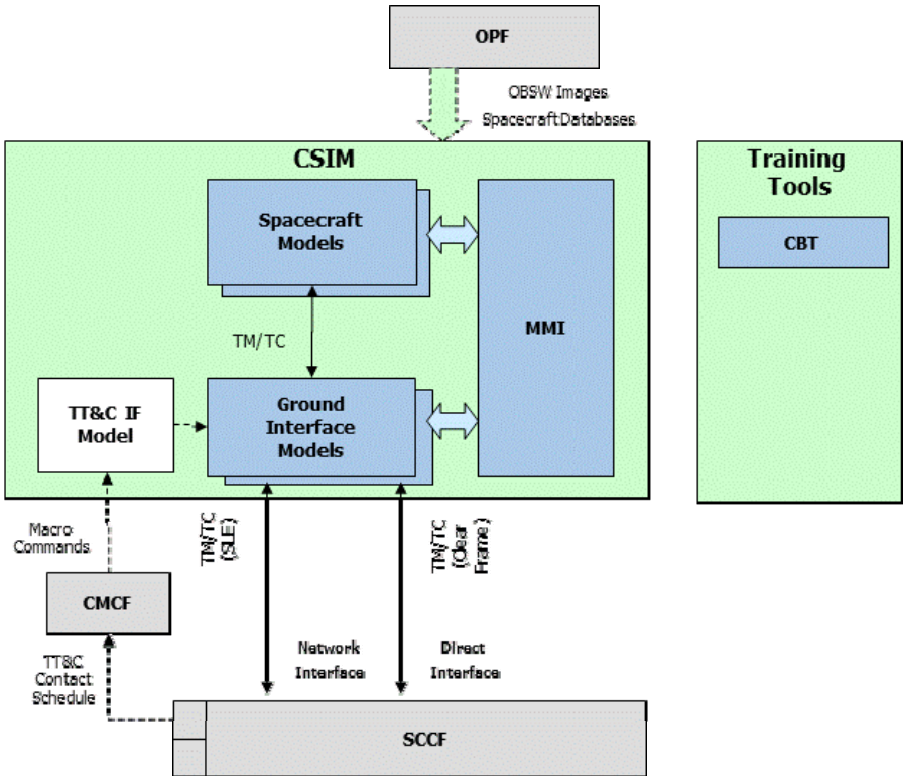

Figure 1: CSIM IOV Context

The CSIM constellation simulator is meant to support ground segment validation and operations.
Its main features are:

1) It models all spacecraft in and out of ground contact
2) Receives telecommands (TC) from the SCCF and responds appropriately
3) Generates payload and platform housekeeping telemetry (TM)
4) Models on-board mass memory (stores TM when out of ground station contact)
5) Runs the real on-board software image
6) Models predefined failure cases
7) Supports Galileo IOV phase (4 spacecraft)

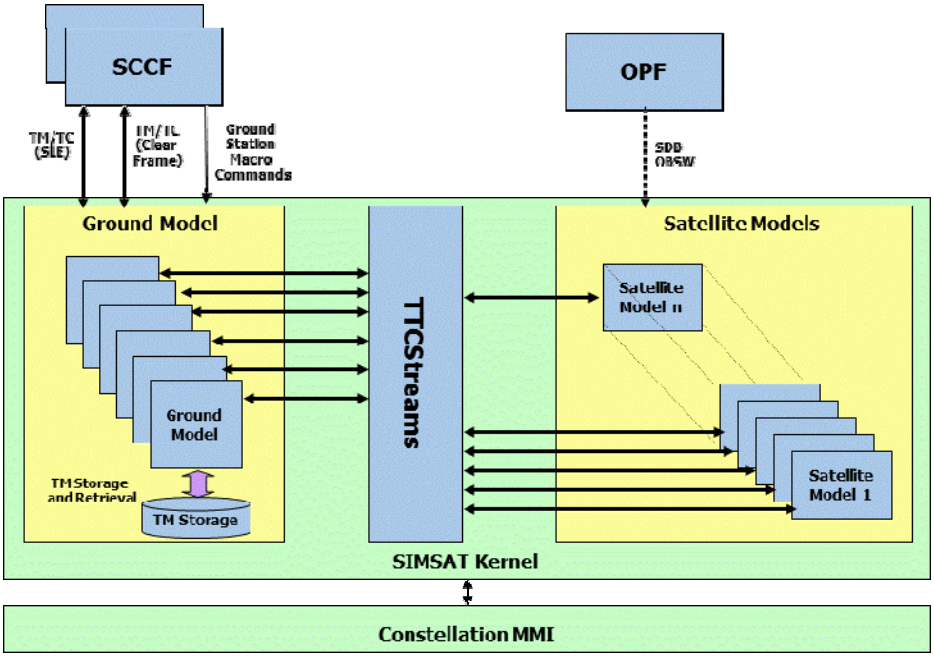

Figure 2: CSIM IOV Architecture

III. CSIM FOC Version

With the start of the FOC phase of the Galileo project several issues had to be considered on the simulator. One issue, known from the beginning, is the fact that there will be different families of spacecraft, which will be very similar, but different. The other issue arising early in the FOC phase of the FOC simulator update is the fact that the IOV satellites are not identical. It turned out that IOV one and two are using different batteries and payload (NSGU) from different suppliers. This made it necessary to find a solution for the two families and for differences within the families..

A. Differences within a spacecraft family

For accommodating two different payload models (NSGU) within the IOV spacecraft family, the following issues had to be considered:

1) IOV Spacecraft 1 & 2 have a NSGU from one supplier
2) IOV Spacecraft 3 & 4 have a NSGU another supplier
3) The command set is different between the two NSGU units
4) Their functionality is very similar
5) Support and lifecycle updates for the two NSGU units are assumed to be different

Therefore the question was with regards to the simulator models: shared code or complete code branching? It was decided to have a complete code branching. The reason was that the models have two aspects, the TM/TC related functionality and the behaviour related functionality. The NSGU behaviour related functionally is limited as the purpose of the simulator is not to model payload functionality. The TM/TC related functionality is completely dependent on the databases related to the spacecraft's. It was therefore the conclusion that having a complete code branching is more convenient for operations as for example a database update for IOV Spacecraft 1 & 2 would not have any impact on the IOV Spacecraft 3 & 4 spacecraft. The drawback is then that if a problem is found in the behaviour functionality, it has to be fixed on both sides.

The CSIM checks at start-up the database used and then loads the corresponding NSGU model.

B. Two spacecraft families

The two families of spacecraft already defined in the program share a large percentage of components, and this can potentially result in possible high source code reuse in the development of the CSIM to support the second family of spacecraft. However this might come at the expenses of breaking existing trusted and validated

functionality and thus a whole range of trade-offs can be considered in many areas of the CSIM code and development. Possibilities range from the two extremes of maximum software reuse to complete code branching. In the first case we maximise engineering and revalidation efforts, minimising maintenance costs, while in the second case we can expect the contrary.

From comparing the schematics of the IOV and the FOC satellite components it is quite obvious that the architectures of the two families of spacecraft are more less identical. However, even if the components are produced by the same suppliers, they are controlled by different on-board software and using different TM and TC databases.

For the maximum software reuse scenario the following was considered:

1) It would maximise engineering as a shared code base architecture would be needed. This solution was already implemented for the Rosetta Mars Express and Venus Express Mission.
2) It would maximise revalidation efforts as by using a shared code base all IOV validation would have to be repeated in order to provide the evidence that the modifications have not impacted the IOV family functionality.
3) It would minimise eventually the maintenance costs. This was the case for Rosetta Mars Express and Venus Express simulators. However, contrary to Galileo's case, all three spacecraft were produced by the same supplier.

For the code branching scenario the following was considered:

1) It would minimise engineering as the IOV spacecraft models could be taken as a template and only the differences in behaviour and TC and TM functionality have to be implemented.
2) It would minimise revalidation efforts as the IOV family functionality would not be impacted. The major difference with respect to the IOV constellation simulator would be that there are two spacecraft model classes instead of one and that at the start of the simulator it has to be made sure that the first 4 spacecraft models are IOV based and the other ones FOC based.
3) It would maximise eventually the maintenance costs as problems which affect both families have to be implemented and validated twice.

Of course some operational needs have to be considered in the evaluation as well. The IOV spacecraft simulator has to be usable by operations at any time. There are two launches to be supported during the development period. And the FOC simulator models have to be available for supporting of SVT and SCTC.
The evaluation of the pro and contra showed that:

1) The majority of the differences between the two spacecraft families are in the TC and TM functionality. It was concluded that the effort for implementing a shared code base is much higher than the benefits in the maintenance phase can be. It has at this point also to be noted that the behaviour functionality is supplier dependent. Each of the two spacecraft families has its own lifecycle. The benefit of a shared code base could easily turn into a disadvantage as modifications on one family could cause problems on the shared models.
2) The validation of the IOV spacecraft family was a big effort. Having a shared code base would require that every single requirement has to be revalidated.
3) The IOV simulator was needed for the first launch and will be needed for the second launch. Implementing a shared code base without affecting Operations would mean that the FOC simulator would not be available before all IOV requirements have been revalidated. This could have a major schedule impact.
4) An historical analysis of the problems of the IOV simulator showed that the majority of the them were related to commanding models and generation of telemetry. This means that by selecting the code branching the effort would be higher in the maintenance phase, but hopefully not significantly.

At the end the following solution was selected:
There will be an IOV spacecraft model and a FOC spacecraft model. Both will be integrated in the simulator. This means all infrastructure software will be shared while the spacecraft models are branched. The reason for this was that:

1) Suppliers use very different TM and TC definitions
2) IOV part can be kept usable for operations
3) Only limited revalidation effort required for IOV

In a first step the FOC spacecraft models will be derived from the IOV spacecraft models and the FOC simulator will include this spacecraft model quite early for SVT tests. The reason for this was that:

1) The behaviour functionality of the two spacecraft families is nearly identical. This means the effort for implementing the differences is moderate.
2) The revalidation effort for the CSIM simulator, including the second spacecraft, is limited to regression testing for the IOV spacecraft and some modifications to the constellation tests.
3) FOC spacecraft simulator models can be used in SVT and SCTC.

In a second step the FOC spacecraft simulator model will be iteratively updated and the configuration user interface and constellation editor will be improved for operational needs. The reason for this is that:

1) The FOC database and on-board software are late and are delivered in an iterative fashion.
2) Operational needs for constellation setup are changing as the constellation is getting more heterogeneous.

On the user side, of particular interest is the configuration of a constellation for specific training and validation scenarios. A constellation editor is provided, along with tools to update Epoch and Orbit of each spacecraft, but other techniques used during the first IOV launch will be reviewed.

IV. Presenters

Christian Bodemann was born in 1964 and graduated in aerospace engineering at the University of Braunschweig in 1994.
In 1995 he started working in the space business at Intelsat in Washington DC and joined VEGA in Darmstadt in 1997. In VEGA he worked in the last years as project manager on several ESA simulators such as Rosetta, Mars Express, Venus Express, Lisa-Pathfinder and CSIM IOV. Currently he is the project manager for the CSIMFOC constellation simulator.

Nicola Di Nisio was born in 1972 and graduated in Computer Science at the University of L'Aquila in 1996.
In 1998 he started working in the Space business, with Dataspazio S.p.A. in Rome, on the development of Ground Segment Software, mainly Mission Control Systems and Off-Line Processing Systems for Science Data.
In 2001 he joined Terma and in 2004 moved to development and maintenance of Simulation Infrastructure Software for ESA, most notably SIMSAT 4, Simulus 5 and UMF, of which he was the project manager.
In 2009 he joined ESA to take up his current role of Data System Manager for the Galileo Constellation Simulator (CSIM).