# Feasibility of using Operational Simulators for real-time model based diagnostics


Daniele Segneri[(1)], Vemund Reggestad[(2)]

[(1)] *Telespazio VEGA Deutschland GmbH*
*Europaplatz 5*
*D-64293 Darmstadt*
*Email: daniele.segneri@telespazio-vega.de*

[(2)] *European Space Operations Centre*
*Robert-Bosch Strasse 5*
*D-64293 Darmstadt*
*Email: vemund.reggestad@esa.int*


## INTRODUCTION

This paper describes the study and prototyping activities performed in order to assess the feasibility of using operational simulators for real time model based system diagnostic.

The study was run in the following two major steps, both including prototyping:

**Simulation synchronisation to spacecraft using real time telemetry**. During the first phase of this step, the vector defining the state of the spacecraft systems was refined and the approach for synchronisation of the different systems was drafted. In the second phase, a subset of systems was selected for prototyping activities.

**Simulator usage for real-time model based diagnostics**. During this step, automatic algorithms were prototyped aiming at demonstrating the possible usage of simulation together with satellite real time telemetry for real time identification of both failures and false alarms.

## SIMULATION SYNCHRONISATION TO SPACECRAFT USING REAL TIME TELEMETRY

### Spacecraft systems synchronisation approach

The state vector of a satellite system is the minimum set of variables from which the state of the entire system can be derived. Using this definition we can easily state that:

- Only a small subset of the variables in a simulator is part of the state vector.
- Only a small subset of the satellite data reported in telemetry is part of the state vector.
- The state vector is much wider than the data reported in telemetry as it includes dynamic configuration such as OBSW image, orbit data, pre-loaded mission time line content, etc.

The approach to be used for the synchronisation of the state variables deeply depends on their nature and there is no "fit all approach". The following approaches have been identified depending on the nature of the systems and their state vectors:

**Epoch Time and Orbit State Vector**: Setting of the epoch time and of the orbit state vector is natively supported by the SIMULUS simulation infrastructure.

**Attitude and angular rates**: The attitude and the angular rates of the spacecraft are not members of the state vector. They are products of the AOCS/GNC operational mode and of the loaded guidance. To speed up the convergence process they can be set into the dynamics model. The core activities to achieve their convergence are anyhow the achievement of the AOCS/GNC operational mode and the pre-loading of the guidance in the OBSW. Both of these activities can only be performed by running flight control procedures that are specific to the mission.

**Digital status parameters**: These are the status of switches in the electrical network, the status of separable and deployable units and to the status of valves. Most of these status parameters can be mapped to parameters in housekeeping and directly set into the models from the telemetry values of the mapped parameters.

**Analogue status parameters**: Parameters of these kind are, for example, the temperatures of the thermal nodes and the attitude and orbit control parameters. These parameters are usually under control of either OBSW or HW loops. The correct initialisation of these loops is therefore the core activity to achieve their convergence. To speed up the convergence, most of these values can be mapped to parameters in housekeeping and directly set into the models from the telemetry values of the mapped parameters.

**Non observable values**: These values need to be estimated from other parameters, typical example is the state of charge of the battery that can be deduced by the battery voltage, current and temperature.

**Functionally modelled intelligent units**: The synchronisation of these units is mainly a matter of synchronising their operational modes and their dynamical values (telemetry profile, monitoring tables, configurable parameters, OBCPs status, time tagged command queues, etc.). This can often be performed only by running flight control procedures specific to the mission and the unit.

**Emulated intelligent units**: The synchronisation process is partially equivalent to the one applicable to functional models of intelligent units and relies to a large extent on starting from a known status (as close as possible to the final status of the real spacecraft unit) and on playing back flight control and flight dynamics related products (flight control procedures, command stacks, command request files). This task is more complicated for the processors in charge of the spacecraft control as it requires also the synchronisation of all the systems under their control.

**Prototyping activities**

The prototyping activities focused on demonstrating the feasibility of synchronising and keeping synchronised the simulator behaviour to the spacecraft behaviour for a complex domain under the control of the OBSW.

The demonstrator was developed using the BepiColombo mission and its thermal subsystem.
The choice of the thermal subsystem was driven by the fact that it implies the synchronisation of both analogue status parameters from telemetry and of OBSW internal configuration parameters of the thermal loops.
The choice of BepiColombo was driven by the high level of fidelity of the Thermal Model used in its operational simulator. This model solves the thermal balance differential equations and is configured using a reduced set of the data used to configure the ESATAN network in ESTEC.

The demonstrator was developed using two installations of the same simulator, one acting as the satellite (Master), the other one acting as the simulator to be synchronised (Slave). As shown in “Fig.1” the master simulator was connected via proxy to the slave simulator for routing of telecommands and telemetries.
As part of the slave simulator a JavaScript based synchroniser was developed having access to both the master and the slave telemetry flow and having access to the slave configuration files.

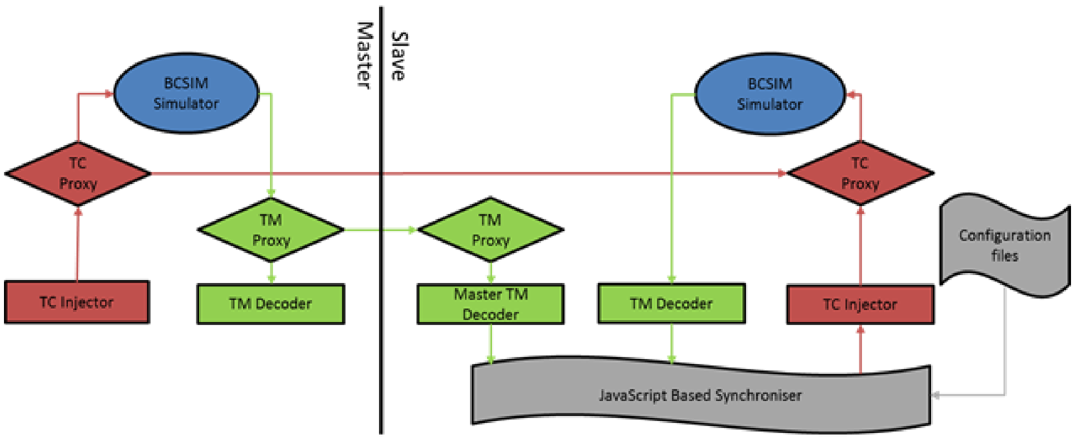



Fig. 1. Synchroniser architecture

The synchronisation of the thermal subsystem was achieved by synchronising both the thermal control loops within the OBSW and the temperatures of the thermal nodes within the thermal network.

At synchronisation time the synchroniser algorithms act on the following items:

**Thermal control loops**: The synchroniser retrieves the status of the loops (enabled/disabled state and control temperature thresholds) from the telemetry of the master simulator using the master TM decoder. It then sets the same status into the slave simulator using flight control procedures.

**Temperatures of the thermal nodes**: The synchroniser uses the slave simulator configuration files to link the thermal control system temperature acquisitions to thermal nodes in the thermal network. It uses the thermal control system telemetry values of the master spacecraft to align the thermal values of the slave spacecraft thermal nodes.

As shown in “Fig.2”, the synchronisation of a two thermal control loops (called MERTIS and MORE) under control of OBSW loops is possible relying on the telemetry values of the satellite and on mission specific operational knowledge. In this figure, the nodes of interest are the slave nodes with (w) and without (w/o) synchronisation process having occurred. Starting from a slave simulator misaligned in terms of thermal behaviour, the synchronisation process leads to a fast alignment of the slave nodes temperature to the master node temperature and to a trend following the master node temperature.

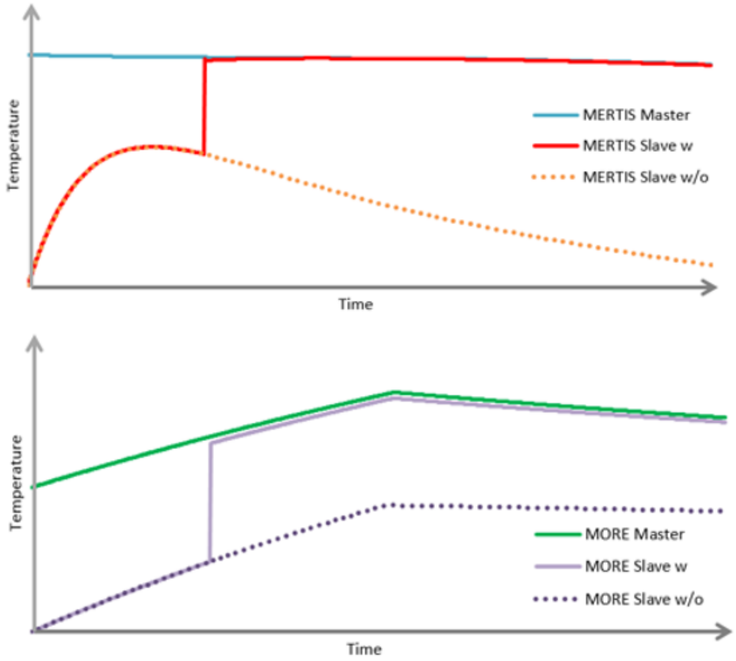



Fig. 2. Synchroniser results

## SIMULATOR USAGE FOR REAL-TIME MODEL BASED DIAGNOSTICS

This phase of the study focused on prototyping a detector capable of identifying failures occurring on the satellite by comparing the telemetry stream of the satellite with an equivalent stream coming from a synchronised simulator. During this phase, the architecture shown in “Fig 1” was extended developing a failure detector as shown in “Fig 3”.

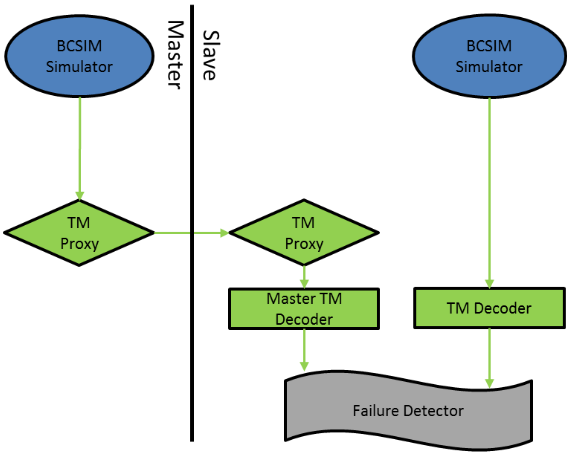
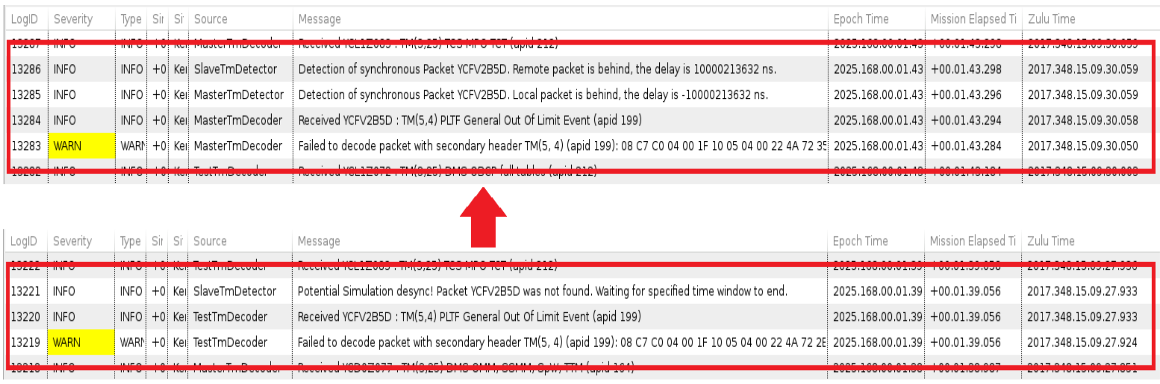



Fig. 3. Failure detector architecture

The algorithms running in the failure detector are based on the assumption that the telemetry produced by simulator (Slave) is representative of a non-failed satellite. Only deviations of the telemetry produced by the real satellite (Master) from the one produced by the simulator are to be considered as indications of possible failures.
This set up is particularly powerful not only for the detection of failures but also for the isolation of false alarms. Typical false alarms are high severity events produced by the satellite under nominal conditions. These may be caused by external conditions such as eclipses and not by failures occurring in the satellite.

A simple algorithm based on a configurable set of packets and their acceptable reception time window (i.e. the acceptable tolerance in reception time between the simulator and satellite) was able to identify and silence false alarms as shown in "Fig. 4" extracted from the simulator log.

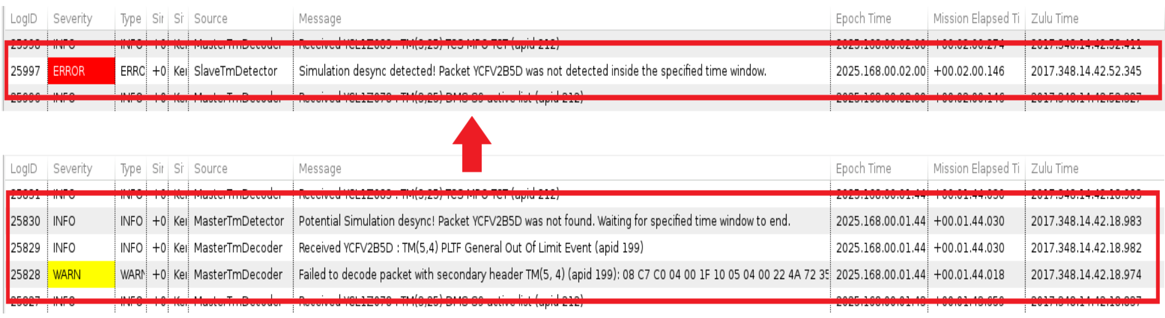


| LogID | Severity | Type | Sir | Si | Source | Message | Epoch Time | Mission Elapsed Ti | Zulu Time |
|---|---|---|---|---|---|---|---|---|---|
| 13286 | INFO | INFO | +0 | Kei | SlaveTmDetector | Detection of synchronous Packet YCFV2B5D. Remote packet is behind, the delay is 10000213632 ns. | 2025.168.00.01.43 | +00.01.43.298 | 2017.348.15.09.30.059 |
| 13285 | INFO | INFO | +0 | Kei | MasterTmDetector | Detection of synchronous Packet YCFV2B5D. Local packet is behind, the delay is -10000213632 ns. | 2025.168.00.01.43 | +00.01.43.296 | 2017.348.15.09.30.059 |
| 13284 | INFO | INFO | +0 | Kei | MasterTmDecoder | Received YCFV2B5D : TM(5,4) PLTF General Out Of Limit Event (apid 199) | 2025.168.00.01.43 | +00.01.43.294 | 2017.348.15.09.30.058 |
| 13283 | WARN | WARN | +0 | Kei | MasterTmDecoder | Failed to decode packet with secondary header TM(5, 4) (apid 199): 08 C7 C0 04 00 1F 10 05 04 00 22 4A 72 35 | 2025.168.00.01.43 | +00.01.43.284 | 2017.348.15.09.30.050 |

| LogID | Severity | Type | Sir | Si | Source | Message | Epoch Time | Mission Elapsed Ti | Zulu Time |
|---|---|---|---|---|---|---|---|---|---|
| 13221 | INFO | INFO | +0 | Kei | SlaveTmDetector | Potential Simulation desync! Packet YCFV2B5D was not found. Waiting for specified time window to end. | 2025.168.00.01.39 | +00.01.39.056 | 2017.348.15.09.27.933 |
| 13220 | INFO | INFO | +0 | Kei | TestTmDecoder | Received YCFV2B5D : TM(5,4) PLTF General Out Of Limit Event (apid 199) | 2025.168.00.01.39 | +00.01.39.056 | 2017.348.15.09.27.933 |
| 13219 | WARN | WARN | +0 | Kei | TestTmDecoder | Failed to decode packet with secondary header TM(5, 4) (apid 199): 08 C7 C0 04 00 1F 10 05 04 00 22 4A 72 2E | 2025.168.00.01.39 | +00.01.39.056 | 2017.348.15.09.27.924 |

Fig. 4. False alarm detection

The same algorithm was able to isolate a real failure as shown in "Fig. 5" extracted from the simulator log.



| LogID | Severity | Type | Sir | Si | Source | Message | Epoch Time | Mission Elapsed Ti | Zulu Time |
|---|---|---|---|---|---|---|---|---|---|
| 25997 | ERROR | ERRC | +0 | Kei | SlaveTmDetector | Simulation desync detected! Packet YCFV2B5D was not detected inside the specified time window. | 2025.168.00.02.00 | +00.02.00.146 | 2017.348.14.42.52.345 |

| LogID | Severity | Type | Sir | Si | Source | Message | Epoch Time | Mission Elapsed Ti | Zulu Time |
|---|---|---|---|---|---|---|---|---|---|
| 25830 | INFO | INFO | +0 | Kei | MasterTmDetector | Potential Simulation desync! Packet YCFV2B5D was not found. Waiting for specified time window to end. | 2025.168.00.01.44 | +00.01.44.030 | 2017.348.14.42.18.983 |
| 25829 | INFO | INFO | +0 | Kei | MasterTmDecoder | Received YCFV2B5D : TM(5,4) PLTF General Out Of Limit Event (apid 199) | 2025.168.00.01.44 | +00.01.44.030 | 2017.348.14.42.18.982 |
| 25828 | WARN | WARN | +0 | Kei | MasterTmDecoder | Failed to decode packet with secondary header TM(5, 4) (apid 199): 08 C7 C0 04 00 1F 10 05 04 00 22 4A 72 35 | 2025.168.00.01.44 | +00.01.44.018 | 2017.348.14.42.18.974 |

Fig. 4. Failure isolation

## CONCLUSIONS

The performed studies and prototypes showed that:

- The synchronisation of simulation behaviour to satellite behaviour is possible also for complex systems under control of the OBSW.
- The synchronisation of simulation behaviour to satellite behaviour is often a mission-specific task. A general infrastructure could be developed having access to all the simulation, satellite and ground elements. Nevertheless a relevant part of the algorithms will have to be developed as mission-specific.
- A simulation synchronised to the real satellite during its entire lifetime is a very valuable tool not only for failure identification but also for false alarms isolation. This set up would provide a valuable contribution for the reduction of the flight control team effort.
- The set up with a master simulator playing the role of the real satellite was very successful given its flexibility in terms of scenarios that can be set up and played.

## WAY FORWARD

The next steps that can be foreseen are in multiple dimensions and involve:

- Extension of the simulation infrastructure to allow their real-time connection to the operational systems for telemetry and telecommands.
- Continuation of prototyping activities for the synchronizer algorithms to exploit the synchronization of complex scenarios up to a full spacecraft.
- Continuation of prototyping activities in the failure detector algorithms to exploit failure reasoning.

The above tasks should allow to develop a minimum viable product capable to demonstrate the added value of the concepts in an operational setup.

## REFERENCES

[1] SIMULUS-NG Consortium OPS-GI, “Summary of Concepts of SIMULUS NG Infrastructure” unpublished.