Flexible Hardware In the Loop Configuration in Spacecraft Test Benches

M. Neefs[(1)], M. Haye[(2)]

[(1)]*Dutch Space B.V.*
P.O.Box 32070, 2303 DB Leiden, The Netherlands
Email: m.neefs@dutchspace.nl

[(2)]*Dutch Space B.V.*
P.O.Box 32070, 2303 DB Leiden, The Netherlands
Email: m.haye@dutchspace.nl

INTRODUCTION

In recent ESA science programs new development approaches have been adopted in order to optimize spacecraft integration and validation efficiency. A coherent set of test facilities is developed supporting the different phases in the verification program. The spacecraft simulator is an important common component present in all test configurations. The use of a single simulator throughout the project maximizes reuse of test experience and simplifies validation of system performances. Characteristics measured in test bench configurations can easily be compared with expected results derived from earlier simulations. However deployment of a single simulator in a range of test configurations is not trivial. A flexible solution can only be achieved by careful design of the simulator and its external interfaces from the very start of the program. This paper describes the approach chosen in the Real Time Simulator and Avionics SCOE as developed by Dutch Space for the Gaia program. The paper will focus on the mechanisms used to configure the EuroSim [1] based test system for the different Hardware In the Loop (HIL) configurations used in the Gaia AVM and PFM verification campaigns.

GAIA TEST BENCH OVERVIEW

The Gaia mission will provide scientists a very precise three-dimensional map of the stars in our galaxy. In order to reach its goals the Gaia instrument and the Gaia avionics must meet stringent accuracy requirements. The actual performances of the Gaia spacecraft are validated using high fidelity simulators and front end equipment that are integrated in various test benches. Fig. 1 shows an overview of the main test benches used in the Gaia development program.

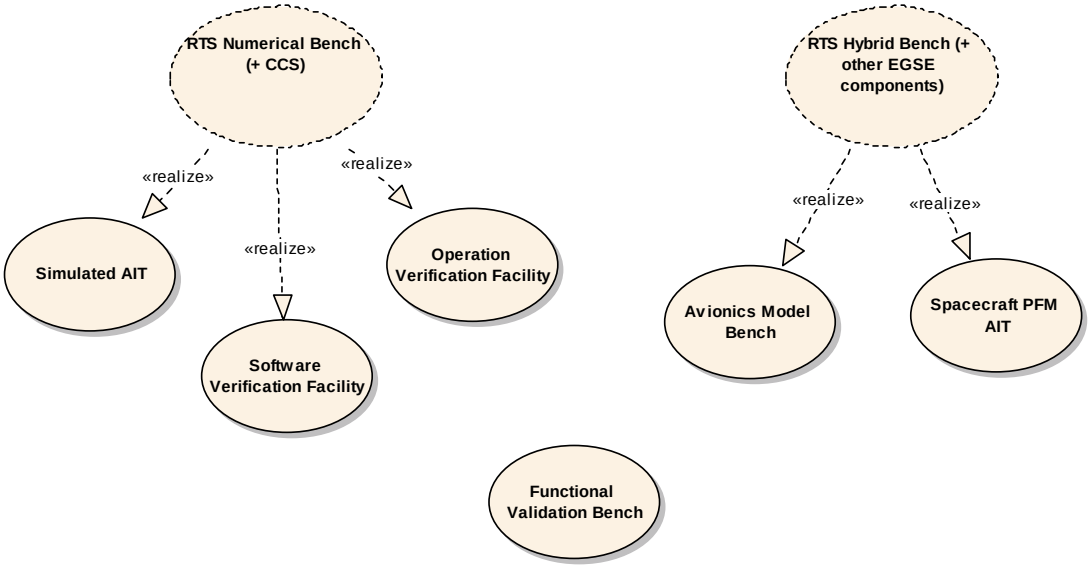

Fig. 1. Test bench overview

The Gaia test benches as shown in of Fig. 1 are:

- Functional Verification Bench (FVB). This pure numerical facility is used to test the (initial) AOCS algorithms.
- Software Validation Facility (SVF). This pure numerical facility is used to test and debug the Central Software (CSW).
- Operation Validation Facility (OVF). This pure numerical facility is used for the development of test procedures for Spacecraft Operations.
- Simulated AIT (SimAIT). This pure numerical facility is used for the development of test procedures for Spacecraft PFM AIT.
- Avionics Model Bench (AVM). This facility is used for the integration and test of the Gaia Avionics.
- Spacecraft PFM AIT. This facility is used for the integration and test of the Gaia spacecraft.

The RTS is a component in all these benches, except for the FVB. However, many FVB models will be reused and integrated in the RTS, enabling a direct comparison of simulation done on the FVB and the RTS.

The five benches that include the RTS can be divided into two main configuration types: a so-called numerical bench configuration and a so-called hybrid bench configuration. A numerical bench configuration is a pure numerical (i.e., software) configuration. In this configuration all S/C units are simulated including the on-board computer (CDMU); the CSW is executed by an instruction set emulator. The hybrid bench configuration is a mixture (hybrid) of simulated and "real" S/C units. In this configuration the CDMU hardware is always present (i.e., not simulated) and the other S/C units can be either present or simulated. The simulator and the SCOE in the benches are controlled by automated test sequences running on the Central Checkout System or CCS. A top level overview of the hybrid and numerical benches is shown in Fig. 2.

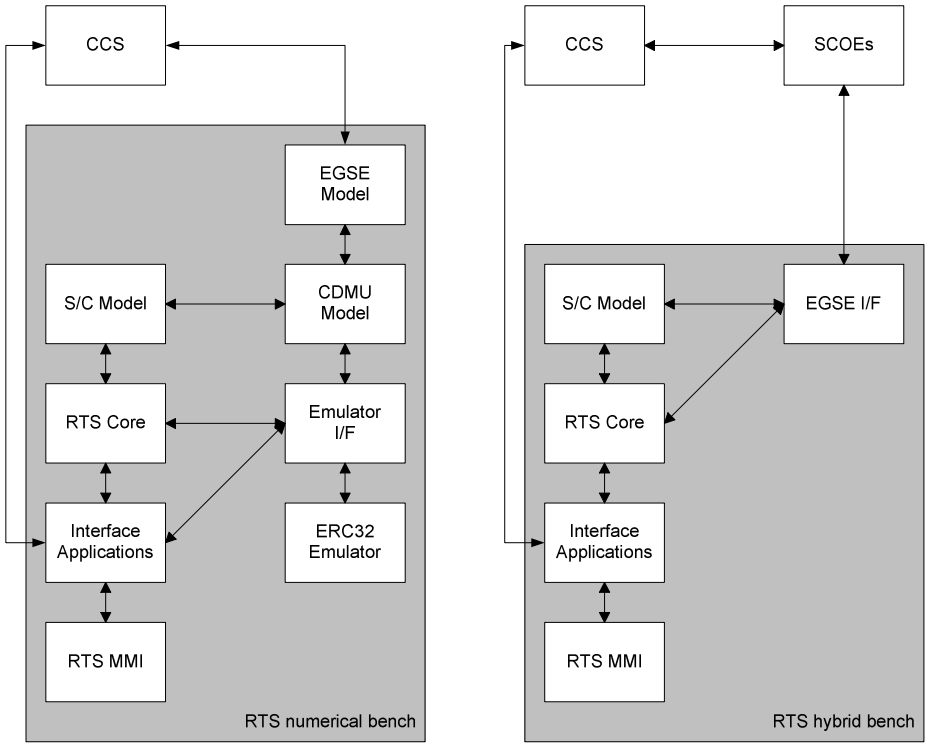

Fig. 2 Numerical and hybrid benches

In hybrid bench configurations the simulator models exchange data with real S/C equipment using various SCOE. In numerical bench configurations all S/C unit are simulated and no SCOE interfaces are needed. The numerical bench SimAIT includes simulation models of the relevant SCOE used in hybrid configurations.

HIL CONFIGURATIONS

As described in the previous section the Gaia RTS supports different combinations of real and simulated spacecraft equipment in the hybrid benches. In the simplest HIL configuration only the on board computer is real and all other spacecraft units and their environments are simulated. During spacecraft integration real units are gradually integrated on the bench, replacing their simulated counterparts. Initially electrical integration is performed; later the units open and closed loop performances are verified. In closed loop tests the sensors are supplied with appropriate stimuli. Actuator commanding and status is acquired to allow the simulator to compute orbit and attitude propagation. The total number of possible test configurations is quite large especially when considering the unit redundancy used in the Gaia design. Changing a single spacecraft unit from simulated to real, requires changing various system configuration settings. The simulation model and its electrical interfaces simulated towards the spacecraft (such as MIL-1553 response generation) must be disabled. If the unit involved is a sensor, another piece of software computing stimuli may need activation. As a consequence electrical interfaces involved in the sensor stimuli generation must be enabled. In some configurations power supply to the real unit must be activated using the PCDU simulation functions present in the EGSE. Furthermore simulation models depending on the specific state of the unit must now retrieve data from the real unit interfaces instead of copying the information from the simulated unit. A similar set of configuration settings must be altered when changing a spacecraft actuator model from simulated to real. To further complicate the matter additional limitations apply when setting up the bench configuration due to functional or interface constraints in flight hardware or EGSE equipment. In the developed Gaia RTS several mechanism have been devised in order to offer the required flexibility while maintaining a single simulator with a common set of simulator models used in all configurations.

MODEL DATA AND DATA EXCHANGES

The Gaia simulator contains of a collection of models each with their own input and output parameters. As part of the EuroSim simulator development, the model interface is established. This means that for each model a list of variables and functions is to be provided which must be visible to the simulation kernel. After compilation, all identified variables become part of the EuroSim run-time variable database. The identified functions will become accessible by the EuroSim scheduler. GUI tooling is available to specify this interface. The EuroSim run-time database that contains all EuroSim accessible variables and functions is called the data dictionary. The accessible functions are called entrypoints. The mechanisms to come to an interface between EuroSim and the models however are not enough to come to an integrated simulator. What is still missing is infrastructure to exchange data between models.

An essential consideration is to ensure that the model itself does not have knowledge about the origin and destination of its input, respectively, output values. If a single simulator is to be used in all configurations the data flows between models must be configurable at test bench initialization time. Only at simulator startup a configuration parameter will determine if a model gets its command parameters from the real CDMU via the Avionics SCOE or from a Simulated CDMU running in a numerical bench configuration. The data exchanges should not be hard coded in the model or in the integrated simulator executable.

The EuroSim environment offers building blocks to implement the data exchange infrastructure. It is based on the presumption that data flows between models are handled via simulation variables. The infrastructure is based on the so-called “Model Description” facility in combination with the “Variable Exchange” facility. The model description facility is used to identify for a model entrypoint, by means of a “model description file”, what the input and output variables are. During the simulator build process, EuroSim will generate a so-called datapool, that contains copies of all the specified input and output variables (see Fig. 3 on the next page). In addition, EuroSim will generate, for each model entrypoint defined in the model description file, an entrypoint to copy input variable values from the datapool to the model and an entrypoint to copy output variable values from the model to the datapool. Henceforth, we will refer to the original model variables as “local” model variables, as opposed to the “datapool” model variables. Note that both these variable types are part of the data dictionary. The local variables are exposed to EuroSim by the model developer as explained before. The datapool variables are generated and exposed to EuroSim by the model development tooling automatically.

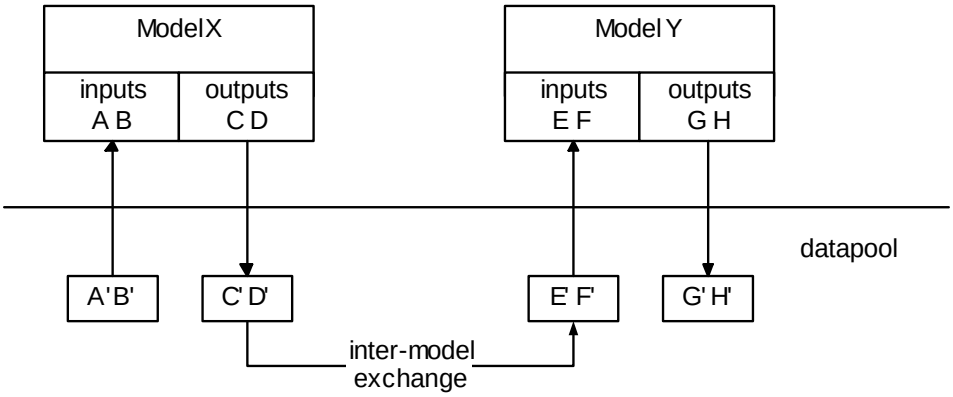

Fig. 3. Datapool and parameter exchange

The inter model exchange facility is built on top of the datapool concept. In a variable exchange file the mapping between model inputs and outputs is established by defining groups of data exchanges between variables inside the datapool. E.g., a group "get_model_Y_inputs" can be created which defines for model Y input variables from which other variables they are derived. For each defined exchange group, EuroSim will create an entrypoint that performs the copy of the data. The creation of the exchange entrypoints is done dynamically at the start of a simulation session. Hence, data exchanges can be changed without the need to recompile the simulator. Note that the parameter exchange is performed on the datapool variables and not on the "local" model variables.

Using the model description and variable exchange facilities, a typical sequence for executing an integration step becomes:

1. Execute the variable exchange entrypoint ("get_model_Y_inputs"). This updates the inputs to model Y in the datapool, E'F', with output parameters from other models. In this case C'D' from ModelX.
2. Execute the function that copies the model Y inputs from the datapool to the model Y local variables. In this example, copy E'F' to EF.
3. Execute an integration step of model Y.
4. Execute the function that copies the model Y outputs from the model Y local variables to the datapool. In the example copy GH to G'H'.

To distinguish between the various data copy actions the following terminology is used. An exchange is the generic name for copying of data between two variables. Copies between datapool and local model variables are referred to as intra-model exchanges; copies between datapool members, as defined in the parameter exchange file, are referred to as inter-model exchanges. At first glance, the datapool concept may seem overly complicated. Why not directly copy data between the "local" model variables. There are two reasons for the intermediate step via the datapool.

The first reason is that the used scheme enables a better schedulability of the models. Let us assume that models X and Y are two time consuming models. Because model Y uses variable input from model X the two models cannot be executed simultaneously on a multiprocessor/multicore computer, because only after model X has finished, it is guaranteed that its output variables are consistent (e.g., a normalized quaternion). As long as model X is executing there is the risk that it is just updating its output variables while model Y is copying them. With the datapool concept, the time consuming models X and Y can be executed simultaneously, only the short duration data exchanges have to be scheduled mutually exclusive. The second reason is that the design can be easily combined with a generic error injection facility for model variables.

RESPONDER MODELS

Within the simulator the responder models are models that implement the interfaces between simulated spacecraft units and spacecraft hardware interfaces. Again the main reason for the responder models is to hide the actual HIL test configuration of other units from the simulated units themselves. As an example the MIL-1553 responder model provides the unit models access to the MIL1553 interfaces present in Gaia. Any model simulating a RT on the MIL-1553 bus receives its commands via this responder model. For the numerical bench, the command data will be generated by another model, the simulated Bus Controller of the CDMU. In case of the hybrid bench the data is

received via the MIL-1553 interface function part of the Avionics SCOE. For the unit model itself this is completely transparent. The model does not need to be aware of the fact that the CDMU is real or simulated at all.

Another advantage provided by the use of the responder models is the possibility to determine the timing of actual S/C interface updates independently of the execution of the simulation models. For instance updating the MIL-1553 RT data may only be allowed in certain predetermined major/minor cycles on the Gaia MIL-1553 AVM bus. However these 64 HZ cycles are in principle not synchronized to the simulation time in the simulator itself. The responder model allows to specify updating of S/C interfaces relative to the AVM bus cycles.

So hiding these interface details in a single set of responder models has the advantage that:

- Equipment models do not need to know interface details between numerical and hybrid bench, so that the same model can be used for both configurations.
- Data exchanges over the external interface can be scheduled independently from equipment simulation. This allows flexibility when designing these data exchange protocols.

Based on the required S/C interface types, the following responder models are implemented in the Gaia RTS:

- MIL-1553 responder model.
  Implements the interfaces with the two spacecraft MIL-1553-STD-B buses.
- SpaceWire responder model.
  Implements the interfaces with the four spacecraft SpaceWire links.
- PacketWire responder model.
  Implements the interfaces with the eight spacecraft PacketWire links.
- Discrete I/O responder model.
  Implements the so-called discrete spacecraft interfaces such analogs SHP etc.

Apart from implementing the external spacecraft interfaces, the MIL-1553, SpaceWire, and PacketWire responder models have two additional functionalities:

- Record the bus/link traffic to file and give the RTS user run-time access to “any message” on the bus/link
- Optionally simulate autonomously the interfaces based on simple scenarios configured by the user.

MODEL MODES

Another mechanism used to support flexible simulator configuration are the so called “model modes”. Each model can run in three modes: disabled, simulated or HIL. Disabled means that the entire equipment model is not scheduled, including any data exchanges. Simulated means that the equipment model is completely simulated in software. HIL means that the real equipment is in the loop. Data exchanges are automatically selected according to the mode of the equipment model. The model mode is configured at initialization time and does not change at run-time. A set of configuration variables define for each equipment what its mode is. This set of parameters is used to enable/disable parameter exchanges between models, whether or not model entry points are to be scheduled etc. In case the real equipment is in the loop, the simulation model of that equipment is disabled and the HIL model is enabled. The HIL model is responsible for the stimulation and monitoring of the real equipment. Only the required set of Avionics SCOE interfaces is enabled when a specific unit is configured to be in HIL mode, all other electrical interface are automatically disabled.

An example will be used to illustrate this approach and to highlight the complexity that is hidden from the end user by the use of model modes and responder models. The Gaia Avionics includes two Fine Sun Sensor (FSS) that produce voltages depending the orientation of the sensor relative to the sun. These voltages are acquired by he CDMU via a separate Electrical Interface Unit or EIU. The EIU is connected to the CDMU via a SpaceWire link. Both the EIU and the FSS may be real or simulated. In case the FSS is real (HIL mode) it can be stimulated via a test connector on the unit.

Fig. 4 on the next page provides the relevant combinations of simulated and real equipment in the loop for one of the fine sun sensors and the EIU unit.

When both units are simulated, the FSS1 model retrieves the S/C attitude and sun vector from the dynamics model and calculates its voltage outputs. These voltage outputs are used as inputs by the EIU model. The EIU model then puts the voltage in a SpaceWire packet and forwards it to the SpaceWire responder model. The SpaceWire responder model will exchange real SpaceWire packets with the CDMU via the Avionics SCOE.

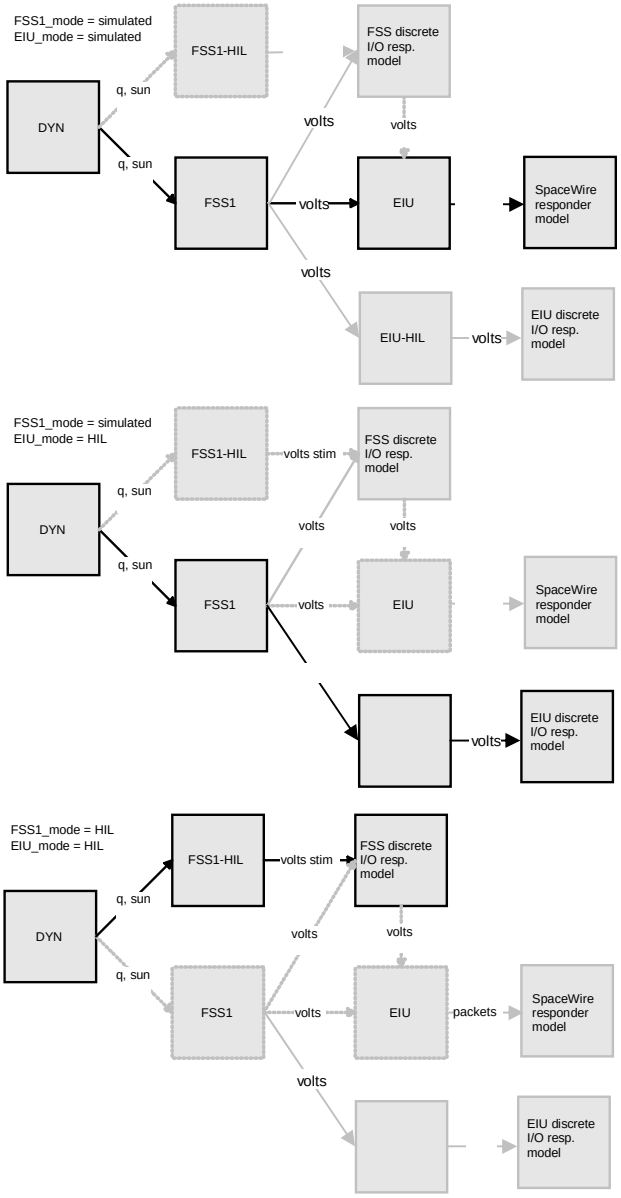

Fig. 4 Model modes at work. Grey lines and boxes indicate disabled models/exchanges

When the FSS1 unit is simulated and the real EIU equipment is present, the FSS1 model and the EIU-HIL model is activated. This model retrieves the S/C attitude and sun vector from the dynamics model and calculates its voltage outputs. These voltages are copied to the EIU-HIL model, which will send them via the EIU discrete I/O responder model to the Avionics SCOE for stimulation of the EIU unit. When both the real FSS1 unit and the real EIU equipment are present, the FSS1-HIL model is activated. This model retrieves the S/C attitude and sun vector from the dynamics model and calculates the voltage outputs. These voltages are copied to the FSS discrete I/O responder model that drives the Avionics SCOE.

CONCLUSION

Considering the described complexity it is clear that the simulation and test infrastructure should support the operator in the definition of the HIL configuration used in a specific test session. Leaving the configuration a manual task would lead to large risks on human errors with potentially serious consequences. In Gaia a flexible configuration mechanism has been developed that allows HIL configuration by changing one single parameter per unit. The mechanism defines different model modes for unit simulation models. Dynamic entry point activation and configurable data exchanges are used to implement the required flexibility in the simulator configuration. The described HIL mechanism has been build on top of existing functions available in the EuroSim real time simulator environment. However the concepts behind the mechanism are generally applicable for any real time simulator used in spacecraft test benches.

References

[1] EuroSim, http://www.eurosim.nl