EUROSIM
NEW DEVELOPMENTS ON A PROVEN REAL-TIME SIMULATOR ENVIRONMENT

Robert de Vries(1)

(1) Dutch Space
Newtonweg 1
2333 CP Leiden
The Netherlands
Email: r.de.vries@dutchspace.nl

INTRODUCTION

Originally developed as a simulator-framework on IRIX for Real-Time-purposes, the EuroSim consortium - Atos Origin, Dutch Space and NLR – has recently released its Mk4-version. This new release has added support for the new SMP2 (Simulation Model Portability) model interface standard, support for models written in Java, calibration support, Web interface, support for enumerated values and many small improvements.

EuroSim is a versatile simulation tool which can be used in all phases of the development of spacecraft. It can be used from the feasibility stage up to operations. It is especially useful during testing as it has extensive support for hardware-in-the-loop real-time simulations. It supports models written in C, C++, Java, FORTRAN 77 and ADA, hardware interfaces such as MIL1553, IRIG-B, GPIB, Reflective Memory, etc., configuration control systems like CVS, and much more.

Most of our customers start modelling in MATLAB/Simulink, commonly used in engineering companies and universities. Since 1999 it is possible to transfer MATLAB/Simulink models to EuroSim with MOSAIC. MOSAIC automatically converts model source code that has been generated with the Real Time Workshop (RTW) of MATLAB into model source that can be incorporated in EuroSim, where additional EuroSim specific files are also created automatically.

While further detailing and expanding models, long execution times and other barriers are encountered. At that moment, EuroSim helps in a growing path to further enhancements.

SIMULATION ENVIRONMENT

EuroSim is based on the principle that each simulator can naturally be broken down into an invariant tool part (the simulation environment) and a part that is specific to the subject being simulated: the latter is called the (simulation) model. The model is a software representation of the behavior of a particular real-world system. By means of a careful design of the tool part, it can be used for both small and large simulators, and it can support the portability of simulation models, allowing them to be re-used between projects and to exchange one model for another, e.g. one offering higher fidelity. Having such a tool available will help to reduce the cost associated with simulation and/or allow it to be used more extensively and earlier in a Space project. It will also help to concentrate the effort on model development itself, i.e. the project specific part of the simulation.

The simulation environment is also used to provide interfaces to the rest of the test facility. Figure 1 shows a generic architecture of a simulation facility. It is clear from the picture that there are many interfaces between a simulation infrastructure and the rest of the system. EuroSim provides drivers and interface libraries to support users to integrate the simulator into the entire test facility.

The interface to the models is supported by the EuroSim native interface method, the SMP1 and SMP2 standard. The EuroSim method requires that the models are written in C, Fortran 77 or Ada. These models can be parsed and their interfaces (variables and entypoint functions) can be tagged and annotated in the Model Editor GUI so that they become available for the rest of the toolset (for scheduling, monitoring, recording, etc.). The SMP1 method requires users to manually publish variables by calling publication functions. Models can be written in C or C++. The SMP2 method requires users to define the interface of each model in a SMP2 catalogue editor. From the catalogue, code is generated. One file contains only skeleton code to be completed by the programmer with the actual model code. Models must be written in C++. In Java, variables and entypoint methods are published via reflection. Variables and entypoint methods can be annotated with special EuroSim annotations. The publication code in EuroSim will automatically extract the information from the Java class files and store that in the EuroSim data dictionary. The data dictionary is a global index containing references to everything that is published using any method. The data dictionary is the source of information on a simulator used by all EuroSim tools.
Interfaces to Front-End equipment are supported by device drivers. There are device drivers for PCI-VME bridge cards, Reflective Memory, GPIB, IRIG-B, MIL1553 bus, etc. These drivers allow model developers to directly (MIL1553 bus) or indirectly (Reflective Memory to Front-End Equipment) interface to Physical Equipment under test (e.g. On-Board Computers, sensors, actuators or payloads).

Interfaces to 3D graphics systems or other external software elements are supported by the External Simulator Access protocol. This allows you to drive applications running on the same or on a different computer with data from the simulator. The external application is a network client to the simulator. You can have any number of external clients connecting to a running simulator. This method can also be employed to create custom MMI screens.

Other functions of the environment include recording, monitoring, scripting (real-time and batch), post processing, configuration control, etc.

REAL-TIME FUNCTIONS

In the first place, EuroSim is designed to be a real-time simulation environment. This means that great care has been taken to ensure that models can be executed at exactly the right time. This is done by the built-in real-time scheduler. The tasks to be executed in the scheduler are defined in the Schedule Editor. This is a graphical editor in which the user can specify exactly how the models shall be run. No modification of source code is required. Based on the same simulator executable you can define different schedules for different purposes. The scheduler can schedule tasks periodically, on software event or hardware interrupt.
Some real-time performance figures of EuroSim:
Maximum scheduler frequency: 1000 Hz
Maximum clock jitter: < 50 µs
Maximum interrupt latency: < 200 µs

TRACK RECORD

EuroSim Mk1 was released in 1997 for SGI computers running IRIX. The first project to use EuroSim was the ERA project. It is used as an engineering simulator first, and later reused for the qualification test campaign. It is also reused in the Mission Preparation and Training Equipment (MPTE). In that configuration it is linked to 3D graphics for the training of the cosmonauts/astronauts who have to use ERA in space on the ISS.

The next project was ATV Rendez-Vous Predevelopment. EuroSim was used to simulate the dynamic behaviour of ATV as well as several critical sensors. The on-board software prototype was running on a separate SPARC single-board computer. The OBC and the (simulated) sensors were connected via a MIL1553 bus. The simulator was connected to EPOS to test the final docking phase. A connection with the GPSLAB GPS Satellite system signal generation system was used to test the performance of the real GPS receiver.

The ATV Test Facilities is the largest test facility to use EuroSim. There are 16 simulators in use to verify the ATV, 5 more simulators are used for operations and training. The simulators are used for software validation, integration testing, qualification and more. The simulator models were mostly developed in Matlab/Simulink and converted to C using Real-Time Workshop using MOSAIC.

The Herschel-Planck ACMS test facility uses 10 simulators for the verification and validation of the ACMS. The simulator models were mostly developed in Matlab/Simulink and converted to C using Real-Time Workshop.

The Generic AOCS simulator runs on EuroSim. The simulator models can run under Matlab/Simulink and EuroSim when real-time performance is required. The simulator is developed in-house.

There are a couple of EuroSim simulators used for the test facilities of various Galileo ground systems. For one of the facilities models in Java are used. These simulators contain ‘models’ that produce and consume network packets. They simulate missing systems in the Galileo ground segment network.

The F16 eCATS embedded training simulator uses a EuroSim simulator to simulate virtual threats to an F16 fighter pilot during an actual flight. The threats are presented on the on-board systems of the plane (radar). The pilot should then take the correct evasive actions. The actions of the pilot are recorded for later playback and evaluation. The advantage of this approach is that it much more realistic (G forces) than training in flight simulators with moving base.
SMP2 SUPPORT

EuroSim offers support for the new SMP2 simulation model portability standard. EuroSim offers a catalogue editor that allows users to define model interfaces. Model interfaces consists of variables, entrypoint, interfaces, etc. A screenshot of the catalogue editor is shown in Figure 5.

The SMP2 Catalogue files are integrated into the standard EuroSim Model Editor. The build process of EuroSim has been adapted to automatically generate the publication source code and some header files. The user only needs to generate a skeleton implementation file once. This file needs to be completed by the user with the implementation of the model. SMP2 models must be implemented in C++, as this is currently the only available standard mapping.

The SMP2 interface dictates a standard interface of the simulator infrastructure that can be used by the models. This interface is completely implemented except for some optional elements.
CALIBRATION

Calibration is used when interfacing with external hardware such as A/D converters or MIL1553 buses. Data on these interfaces cannot directly be used as engineering values, but must be converted first. This process is called calibration.

EuroSim offers a graphical user interface to manually enter the values. The resulting calibration curve is shown to quickly check that the data is correct. The file format of the calibration data is XML. The format specification is open so that users can convert calibration data easily from any format into EuroSim format.

The calibration library using the calibration data files can be called by models to perform the actual calibration calculations.

![Calibration Editor](image)

**Figure 6 Calibration Editor**

WEB INTERFACE

There are sometimes situations when a specialist is needed to diagnose a problem in a test and this specialist is not on-site. For these situations a remote observer capability for EuroSim has been created. The observer application runs on a web browser. The standard secure HTTP protocol is used to communicate between the site with the simulators and the specialist.

The architecture of the system is show in Figure 7. The server communicates with the clients using the HTTPS protocol (HTTP over SSL). The server uses the EuroSim protocol for communication with the simulators. Instead of letting the server connect directly to the simulator, the monitor sets up connections to the simulator and the server, after which it does nothing more than forward data between the two. The monitor is installed on the same network as the simulators it has to watch. The server can request the monitor to scan its local network for simulators, and request a connection to a simulator.

![Web Interface Architecture](image)

**Figure 7 Web Interface Architecture**
SCRIPTING INTERFACES

Since Mk2rev4 an interface to automate simulation runs has been available for the perl scripting language. In this release this interface has been made available to three more (scripting) languages. These languages are:

- python
- tcl
- java

It is now easier than before to integrate EuroSim into existing automation frameworks of customers. The interface code is automatically generated by SWIG (Simplified Wrapper Interface Generator). Support for languages that are supported by SWIG is therefore easy to achieve.

The automation of simulation runs makes it easy to run regression tests or duration tests. The scripting is currently heavily used in the ATV Test Facilities and for the ERA Simulation Facility. It is also used for automated testing of EuroSim itself.

JAVA MODELS

Java models can now simply be integrated into EuroSim by adding them to the Model Editor tree. Java models are published automatically into EuroSim by using the reflection mechanism. This java publication routine simply traverses the object tree to determine the member variables and methods to be published. The publication routine also looks for EuroSim specific annotations. These annotations are used to provide additional information regarding the variables and entrypoints.

Variables and entrypoints can have a description, variables can also have a unit, a minimum and maximum value. Variables and entrypoints can only be accessed through the datadictionary. Data exchange between models written in Java and models written in C, C++, etc. can be achieved using the parameter exchange/data pool mechanism present in EuroSim since Mk3rev2. Figure 8 shows an example of how the publication mechanism looks.

```java
import esimJava.*;

class model
{
    @eurosim(description="some variable", unit="m", min="0", max="10")
    public int var = 2;

    @eurosim(description="another variable")
    public double other = 3.1415;

    model(int x, int y, String name)
    {
    }

    @eurosim(description="this is an entrypoint")
    void compute()
    {
        double x = var * other;
    }
}
```

Figure 8 Example java model

CONCLUSION

EuroSim is continuously evolving to accommodate new user requirements and standards. It has been in use now for more than 10 years in large projects and must be considered mature. New developments allow EuroSim to be used for more applications than before. It is clear that EuroSim is a well established tool in the space simulation application area and beyond.