SESP 2010

Session:	Session: Simulation Technologies (06)
Type:	Concurrent Session
Date:	Tuesday, September 28, 2010
Time:	14:00 - 15:30
Room:	Newton
Chair:	
Co-chair:	
Remarks:	

Seq	Time	Title	Abs No
1	14:00	Push and Pull of SMP2 Compliant Models: New Developments in Automated Model Transfer with MOSAIC from the User Perspective	

Lammen, W.[1]; Moelands, J.[1]; Wijnands, Q.[2]
[1]National Aerospace Laboratory NLR, NETHERLANDS;
[2]ESA/ESTEC, NETHERLANDS

Simulation model developers often use MATLAB/Simulink as their design environment. At the same time many projects require that the simulation models are executed in real-time, e.g. with hardware and/or human-in the-loop. Furthermore, for portability and reuse, the models can be required to comply to the Simulation Model Portability standard (SMP2 and ECSS E40-07). In order to reduce development costs it is widely acknowledged that automatic model transfer between development tools and real-time simulation environments and simulation standards is essential. The model developer can 'push' his or her simulation model to a wider user community, making it compliant to a simulation standard (e.g. SMP2, E-40-07) automatically. Using the same technology a real-time simulation engineer or systems integrator can 'pull' domain specific models into his or her spacecraft simulator. This paper will detail new developments in automated model transfer of (SMP2-compliant) simulation models from the user perspective.

The success of automatic model transfer is based on a continuous interaction with the space community, taking into account user experience. The tool Model-Oriented Software Automated Interface Converter - or for short MOSAIC - automates model transfer between commercial off-the-shelf simulation tools and simulation standards. It has been used by the European space industry for more than 10 years in a large number of projects. In a typical use case spacecraft (sub) system models are developed in Simulink and converted automatically into SMP2-compliant models. In the target simulation environment the models are reconnected with each other (by restoring the dataflow) and integrated with other SMP2-compliant models into a complete spacecraft simulator.

To trigger new developments a questionnaire has been sent recently to (potential) users in the aerospace community that addresses both push and pull aspects of automated model transfer. Several user responses indicate that MATLAB/Simulink is a base environment to develop and maintain the simulation models. Conversion to SMP2 is necessary to be able to exchange the models between different simulation environments. Conversion should be performed as automatically as possible, e.g. integrated into Simulink. Furthermore, users indicate that they would like to integrate the transfer tool into their specific software development environment, e.g. MOSAIC as plug-in to Eclipse /UMF. Possible solutions will be treated in the paper. Further feedback

remarks from users include
- enabling of customisation of the transfer models to cope with exact project needs,
- facilitation of 'round-trip modelling' to be able to (re)store all model changes performed during the model lifecycle,
- one common SMP2 approach, necessary for simulating each tool's code.

The user reactions on the questionnaire provide valuable feedback information that can be used by ESTEC and NLR cooperatively in the process of defining requirements for model transfer of SMP2 models. Details on how the user feedback is treated will be described in this paper.

MOSAIC 8.0
NLR, supported by ESA/ESTEC, has extended the model transfer capabilities resulting in MOSAIC 8.0. Simulation models developed in Simulink and exported with the Real-Time Workshop (RTW) of MATLAB 7.7, can be transferred automatically to four different simulation environments that follow the SMP2 standard: BASILES, EuroSim Mk4, SIMSAT4.2 and SimVis2.2/SIMSAT2.1. User experience has shown that the implementation of the SMP2 standard between the various simulation platforms differs in details (e.g. different conventions of filenames, in-exchangeable catalogue files). This issue is mitigated by facilitating automated transfer to a specific SMP2 compliant simulation tool. Platform-specific configuration files are produced automatically, tightly integrating the transferred model with all supported target platforms.

MOSAIC 9.0
The next MOSAIC release includes model transfer from the EcosimPro to SMP2. With EcosimPro one can model dynamic systems represented by differential-algebraic equations or ordinary-differential equations and discrete events. It is used by leading companies in the aerospace and energy sectors. ESA/ESTEC uses it as a tool for simulation in several fields, including propulsion, environmental control systems, and life support and power systems. Extension of MOSAIC to incorporate EcosimPro model transfer will allow the integration of SMP2-compliant simulation models coming both from Matlab/Simulink and EcosimPro, e.g., for system simulation in ESTEC's Concurrent Design Facility. The support of EcosimPro substantially changes the MOSAIC architecture. The 'one-to-many' porting scheme has to be replaced by a 'many-to-many' porting scheme. This architectural change will be further detailed in the paper. Furthermore it will prepare the transfer tool for supporting additional modelling environments, e.g. Modelica.

2 14:30 Hybrid (Real Time) Test Benches Based on RT Linux

Chatelais, D.[1]; Kircher, B.[2]
[1]Astrium (SAS) France, FRANCE;
[2]EADS Astrium GmbH, GERMANY

The use of Linux for simulation is state of the art. Linux has become a crucial operating system for several use cases. And since the standard Linux kernel is real time capable there is growing market supporting real time applications.

Most of Astrium's non real time simulations are executed on Linux whereas the real time simulations (hybrid simulation interfacing hardware in the loop) are still based on specific real time operating systems. These specialised real time operating system are very effective but most of them are very adjusted for a dedicated set of use cases and the usage of these operating systems is normally not very handy compared to a normal, well-known system like Linux.

Since there is a growing demand on powerful, flexible hybrid simulations within the current running and planned satellite projects the time is right to review the currently used real time platforms.

Why not using Linux for the real time simulations, too?

Real time Linux or Linux for embedded systems is a promising technical field opening a lot of powerful options for real time simulation. To investigate the possible use of RT Linux for hybrid test benches several Astrium R&D projects were started and several prototypes were developed. The question was: Is RT Linux comparable to the currently used real time operating system, is RT Linux meeting the needs for hybrid simulations, is RT Linux capable of being integrated into the available test processes and what are the advantages respectively drawbacks of a RT Linux based simulation. The tests were performed using actual simulation code and available hardware I/O devices to stimulate and monitor the RT Linux based simulation under "real" test bench conditions.

The paper will present the inclusion of RT Linux as simulation platform for real time satellite test facilities.

3 15:00 SimArch: A Layered Architectural Approach to Reduce the Development Effort of Distributed Simulation Systems

Gianni, D.[1]; D'Ambrogio, A[2]; Iazeolla, G.[2]
[1]European Space Agency, NETHERLANDS;
[2]University of Rome TorVergata, ITALY

Distributed Simulation (DS) has gained popularity for the advantages of increased computational capabilities, simulator reusability and distributed execution. Factors affecting the effectiveness of the DS approach are the skills and the extra-effort that the development of a DS system requires with respect to the equivalent Local Simulation (LS) system. In the case of HLA, the most prominent technology for DS, the extra-effort has been estimated to be up to 60% of the effort required for the development of the equivalent LS system. In addition, the coding of extra 3.5KLOC per federate is generally required.

In this paper, we present SimArch, a layered simulation architecture that considerably reduces the above extra-effort by transparently obtaining the DS system from the equivalent LS one. SimArch defines five layers, each introducing a more abstract set of simulation services on top of the underlying layer, the bottom one being the distributed computing infrastructure. There are three main advantages for adopting the SimArch layered approach. The first advantage is that the simulation model is decoupled from the specific execution environment, and thus it can be reused across several simulation execution platforms. The second advantage is that the layers' implementations can be easily modified or replaced to accommodate custom deployment requirements (i.e., local or distributed deployment, performance optimization for given simulation workloads, etc.). Finally, the third advantage is that simulation developers only deal with the high level simulation services, and therefore can focus on the model description rather than being involved in the typical intricacies of the DS systems implementation. As a consequence of this last advantage, the use of SimArch practically eliminates the development extra effort and avoids the coding of the additional 3.5KLOC per federate. SimArch bottom layer (Layer 1) provides a DES (discrete-event simulation) abstraction on top of the distributed computing infrastructure conventionally identified by Layer 0. Such bottom layer does not belong to SimArch, and therefore the service interfaces between Layers 1 and 0 are not defined. In the case of the current HLA-based Layer 1 implementation, such interfaces are subsets of the RTI-Ambassador and

FederateAmbassador interfaces for the communication between Layers 1 and 0 and between Layers 0 and 1, respectively. However, as highlighted by the above described second advantage, the architecture can be easily ported on other DS infrastructures without affecting the already developed SimArch-based simulation systems. The upper layer (Layer 2) deals with the simulation components synchronization and communication, transparently for local and distributed environments. The distributed version of this layer uses in turn Layer 1, to achieve global time synchronization, and provides communication with the remote simulation components. The adjacent upper layer (Layer 3) provides the implementation of the primitives (i.e., components defining the simulation language) and the model configuration services. Finally, the top layer (Layer 4) corresponds to the simulation model definition, which can be obtained through the invocation of the simulation language primitives. Currently, SimArch is provided with jEQN, a language for modelling Extended Queueing Networks - a standard formalism for modelling and analyzing system performance. However, other simulation languages can be plugged into SimArch facilities.

In this paper, we illustrate SimArch architectural principles and how SimArch can be used to effortlessly develop a DS system. In particular, we consider a communication network model, which can represent a computer network within a Ground Segment System or a Satellite-based Communication between two ground-based systems. We also show details of the DS experiments held between the University of Rome TorVergata (Italy) and Georgia Institute of Technology (Atlanta, US). Finally, we illustrate how SimArch has also been specialised to support Agent-based Simulation.