E-40-07, a New Standard for Simulation Model Portability and its Implementation in SIMULUS

Nuno Sebastião, ESA/ESOC (esa.int)
Nicola Di Nisio, Terma (terma.com)
→ Introduction to the E-40-07 standard

Simulation Model Portability
  Portability across missions
  Portability across phases of the same mission
  Portability of binary models

Implementation of the standard in SIMULUS
  The Model Integration environment
  The Runtime Environment
ECSS: European Cooperation for Space Standardisation
Participated by ESA and the European Space Industry

E-40-07 is being drafted and promoted by the ECSS (ecss.nl)
It is based on the existing SMP2 technical specification
From now onwards we will only identify it with the term SMP2

The goal is to promote simulation model portability
Portability across missions
Portability across phases of the same mission
Exchange of binary models

It follows a Model Driven Architecture (MDA) approach
Platform Independent Models and mappings to C++ and Java
E-40-07 Standard being finalised for public review (Oct 2008)
   It will be the next release following SMP2 v1.2

Fully supported by SIMSAT 4 (Version 1.2)
Partially supported by Eurosims, Basiles, SimTG

Industrial validation conducted by:
CNES (Prime)
Spacebel
Thales Alenia Space
EADS Astrium
Ellidiss
A Metamodel-based representation method for reusable simulation model (Winter Simulation Conference 2007)

“We have developed a initial CMM implementation inspired by Simulation Model Definition Language (SMDL) of Simulation Model Portability 2 (SMP2) standards”

Yonglin Lei, Lili Song, Weiping Wang, Caiyun Jiang
School of Information System and Management
National University of Defence Technology, China
Introduction to the E-40-07 standard

→ **Simulation Model Portability**
  - Portability across missions
  - Portability across phases of the same mission
  - Portability of binary models

Implementation of the standard in SIMULUS
  - The Model Integration environment
  - The Runtime Environment
ESA runs a number of missions each year

Models developed for a mission can be reused and adapted for another mission

   Re-use is currently achieved by relying on the detailed knowledge of the implementer, rather than on formalised interfaces

Different kinds of simulators have to be procured for each mission at different stages (real-time, SVF, operational, etc.)

   Re-use within a mission is the exception rather than the rule, this is where significant financial gains can be made from the standard Terma and EADS Astrium achieved this goal in the ATV Operational Simulator (Toulouse, France) and the SVF(Les Mureaux, France)
By portability across missions we mean the possibility to reuse models from a simulator built for mission A in a simulator for mission B (same kind of simulator).

Different requirements on mission B will likely require the customisation of some models.

Typically some models are more reusable “as is” than others:

- Environmental models
- Thermal models
- Electrical Network Models
- TM/TC tool-kit
- Spacecraft Dynamics Model
- Processor Emulators
- IEEE 1553 bus model
- Ground Models
Other models need adaptations from mission to mission

- AOCS
- Reaction Control Subsystem
- Radio Frequency Subsystem
- Data Handling Subsystem

Portability across missions requires that all missions share the same standards and architecture.

ESA/ESOC has implemented reuse across missions internally, for the Operational Simulators

With the E-40-07 we want to enable such a reuse at a broader level, in the European industry and ecosystem of national space agencies. This will open up competition even further and drive costs down.
Along the phases of a mission different simulators are built

- Software Validation Facilities (Numerical benches)
- Real-time simulators (Avionic benches, HIL simulations, etc.)
- Precise Flight Dynamics Simulators (PFDS)
- Operational Simulators (OpSim)

PFDSs are built in parallel with the OpSims

- The AOCS is modelled very accurately
- The Environment is modelled very accurately

The AOCS and the Environment modelling of PFDSs can be reused pretty much “as is” for an OpSim.

- Typically the teams working on PFDSs and OpSims are different, but the first could work for the second.
SVFs are close to OpSims and this is where most of the intra-mission reuse is expected to happen

An SVF is built 1 or 2 years before the OpSsim
SVF models far from the OBSW are typically thinner, so when reused in an OpSim they need to be extended/refined
The CDMU of an SVF is a good match for an OpSim, nowadays we have a processor emulator and an high-fidelity modelling of what is around the on-board computer in both cases

But SVFs can gain from OpSims too!
OpSim models are more realistic and help to validate and debug the OBSW at a later stage

Terma and EADS Astrium exercised both scenarios for ATV
All those simulators are procured by different entities and built by different teams, a common playground is needed to enable broader reuse and also to unlock some markets.

Re-use of models across mission phases will be anyhow challenging.

SMP2 is our **first step** to facilitate this process.

A reference architecture for all simulators is the **second step**.

Collaboration among vendors and sections of various organisations is the third and **definitive step**.
A binary model is distributed without source code, in one or more binary files.

It allows third parties to provide models without giving away their IPRs (special algorithms, optimised computations, etc.). This is a “must have” for intra-mission reuse scenarios, where different companies are expected to exchange models.

SMP2 guarantees the portability of binary models through
- Clear packaging and boot-strapping rules
- Explicitation of all the interfaces needed to operate with the models
- Explicitation of all the interfaces consumed by the model
Introduction to the E-40-07 standard

Simulation Model Portability
- Portability across missions
- Portability across phases of the same mission
- Portability of binary models

→ Implementation of the standard in SIMULUS
  - The Model Integration environment
  - The Runtime Environment
SIMULUS supports the full simulation life cycle:

MIE (Model Integration Environment), to design, assemble, code-generate and package SMP2 models and simulators
SIMSAT, a simulation infrastructure capable of loading SMP2 models

It also provides the following generic models

Generic models for modelling the Ground Segment elements
Generic models for modelling the Space Segment Emulator for ERC32 and MIL-STD-1750

Additional generic models will be provided in the future for many spacecraft subsystems and devices, all fitting an intended Reference Architecture.
Supporting the full simulation life cycle
Delivered as a set of Eclipse RCP plug-ins

Catalogue Editor
Assembly Editor
Schedule Editor
Package Editor
Semantical Validator
Code Generator
A Catalogue is a collection of models and types

Skeleton models can be generated from their catalogue description

You only need the catalogue description along a third party binary model to be able to integrate and use it in your own simulator.
Assemblies are organised in a hierarchical fashion tree of model instances.

Assemblies define:
- models instances
- interface links
- inter-model events links
- field links
- initial values for fields

The Link Editor helps connecting models through the various kinds of links.
Schedule events against Simulation Time
Epoch Time
Mission Time
Zulu Time

Analyse time-slips with the Schedule Analyser
class SolarPanel:
    virtual public ::Smp::IDynamicInvocation,
    virtual public ::Smp::Mdk::Management::ManagedModel,
    virtual public ::Smp::Mdk::Management::EntryPointPublisher
{

    // Default constructor.
    SolarPanel();

    // Constructor setting name, description and parent.
    SolarPanel(::Smp::String8 name, ::Smp::String8 descriptor)

    // Virtual destructor that is called by inherited classes.
    virtual ~SolarPanel();

    // Request for publication.
    void Publish(::Smp::IPublication *receiver) throw ( ::):

    // Perform custom configuration steps.
    void Configure(::Smp::Services::ILogger* logger) throw

    // Connect model to simulator.
    void Connect(::Smp::ISimulator *simulator) throw ( ::Smp::)

    ::Smp::Int32 result;

    // MARKER: OPERATION BODY: START

    result = 10;

    // MARKER: OPERATION BODY: END
    return result;
}

// MARKER: OPERATION BODY: START

result = 20;

// MARKER: OPERATION BODY: END
return result;

SMP2 Model → C++ Code

Code Merging

Model updates are fed back into customised code.
User-friendly conflict management.
The SIMSAT runtime is delivered in two parts
   C++ Kernel where the models are executed
   MMI that is used to control the Kernel

Multiple users can connect to a simulation
   The MMI and the Kernel communicate through CORBA

The SIMSAT Kernel is simulation standard agnostic
   Supports native SIMSAT models and services
   Supports SMP1 models (SMI)
   Supports SMP2 models (v1.2)
   Any new standard that the future will bring us ...
The MMI is delivered as a set of Eclipse RCP plug-ins:

- Data Display
- Logger Viewer
- Commander
- Schedule Viewer
- Schedule Analyser
- Property Grid
- Recorder
- Simulation Tree
- Status Viewer

SIMSAT – the runtime environment
SMP2 Specifications
https://projects.de.terma.com/simsat40/docs/EXT/SMP2/smp2-12/

ECSS – European Cooperation for Space Standardisation
http://ecss.nl

Winter Simulation Conference
http://www.wintersim.org

Nicola Di Nisio, Terma, Project Manager and Software Analyst
nin@terma.com

Nuno Sebastião, ESA/ESOC, OPS-GI, Technical Officer
nuno.sebastiao@esa.int

References
Q & A