Integrating a Simulink System Target File and MOSAIC for Efficient Model Transfer to SMP and EuroSim

Wim Lammen[(1)], Jeroen Moelands[(1)], David Jaffry[(2)], and Quirien Wijnands[(3)]

[(1)] NLR - Netherlands Aerospace Centre
P.O.Box 90502
1006 BM Amsterdam, The Netherlands
Email: mosaic@nlr.nl

[(2)] MathWorks
2 rue de Paris
92196 Meudon, France
Email: david.jaffry@mathworks.fr

[(3)] ESA/ESTEC
P.O. Box 299
2200 AG Noordwijk, The Netherlands
Email: quirien.wijnands@esa.int

INTRODUCTION

Space systems are usually developed in a collaborative setting. Using their specific tools, various partners contribute during the project lifecycle. In order to reduce the overall costs of System Modelling and Simulation (SM&S), ESA has expressed the need to rationalize simulators and harmonize the related tools (including their interfaces) and methodologies.

Simulation model developers often use MATLAB and Simulink [1] to create and test their dynamic system behaviour models. At the same time, many projects require that the simulation models be executed in real time (e.g. with hardware-in-the-loop and/or human-in-the-loop). For portability and reuse, the models may also be required to comply with the Simulation Modelling Platform standard (SMP [2]/ ECSS-E40-07). SMP is currently in the process stage of becoming a formal ECSS standard (ECSS-E40-07 [3]). A main objective of SMP is to enable simulation models to run in different simulation environments, independent of their modelling environment. The SMP standard has been used in several space projects, especially in the field of operational simulation. A number of target simulation environments are compliant with this standard (e.g. EuroSim [6], SIMSAT, and Basiles). These environments are used for simulator integration and real-time simulation and testing. Model development tools, such as MATLAB and Simulink, Modelica/Dymola, EcosimPro [4], or 20-sim [5] are not out-of-the-box compliant with SMP. Users can export C or C++ code from these modelling tools and adapt code (for each model) manually to make it SMP compliant. However, this is a time-consuming, complicated, and error-prone process.

To reduce development costs, automatic model transfer between model development tools and standards (such as SMP and real-time simulation environments) is essential. The MOSAIC tool [7] automates model transfer from commercial modelling tools such as Simulink, EcosimPro, and 20-sim to the native format of the real-time EuroSim simulation tool; it also automates model transfer to several SMP target platforms such as SimVis, SIMSAT, Basiles, and EuroSim. For more than 15 years, the European space industry has used MOSAIC in a large number of projects. The latest publicly available version of the tool is MOSAIC 10 [8]. MOSAIC 10 transfers Simulink exported code through parsing and by adding interfacing and wrapping code around it and to specific files needed for the target simulation environment.

With Simulink, another model transfer approach is also possible. Simulink Coder and Embedded Coder enables the configuration of the code generation process of a Simulink model in order to fulfill a required format of generated C or C++ code. Using this method, a Simulink model can be exported to SMP-compliant code directly, using a specific System Target File (STF). Such a capability establishes a stronger link between the modelling tool and the SMP standard, with the model transfer step integrated in the modelling tool. A direct mapping can be created between Simulink and SMP features, increasing the information transfer. Complementary, the MOSAIC capability is still needed for handling other input formats as well (e.g., EcosimPro and 20-sim), supporting specific features of target simulation platforms, and optimizing specific SMP related issues such as UUID on a generic level.

During the latest MOSAIC activities, (funded by ESA) the new STF approach and traditional MOSAIC approach have been harmonized and integrated with each other, in order to derive an optimal product for the space community that can be easily maintained. First, a feasibility and analysis study was performed to investigate the possibilities of direct SMP-compliant code generation from Simulink [7]. Second, an integrated version of STF and MOSAIC has been developed, resulting in MOSAIC 11. This version has been tested using standard MOSAIC test models and with the SMP Conformance Suite [10]. MOSAIC 11 is considered an experimental version and has not been released externally. It forms the base for development towards an external release (MOSAIC 12) in which all the envisaged MOSAIC features, as well as features specific to Simulink are included. This paper details the technical aspects of the STF/MOSAIC integration performed for MOSAIC 11, as well as the ongoing developments towards MOSAIC 12.

SIMULINK SYSTEM TARGET FILE

The custom SMP.TLC System Target File is based on the Target Language Compiler (TLC) technology [9]. The TLC is an integral part of Simulink Coder and Embedded Coder. This technology enables a high degree of flexibility to customize the generated ANSI C/C++ code. TLC uses text files with template code that explicitly control the way the code is generated. The TLC provides a complete set of ready-to-use TLC files for generating code. Fig. 1 shows how the TLC is integrated with its target files and Simulink Coder, producing output code and makefiles. The first step in the code generation process is the transformation of the Simulink block diagram into an intermediate file called model.rtw. This file contains a "meta" representation of the model describing the execution semantics of the block diagram in a high-level language. It includes the model-specific information required for generating SMP-compliant code from the Simulink model (e.g. variable names and types). The model.rtw file is passed to the TLC, which uses it in combination with a set of included system target files and block target files to generate the code. The code generated by the TLC is highly optimized, customized, fully commented, and can be generated from linear, nonlinear, continuous, discrete, or hybrid Simulink models.

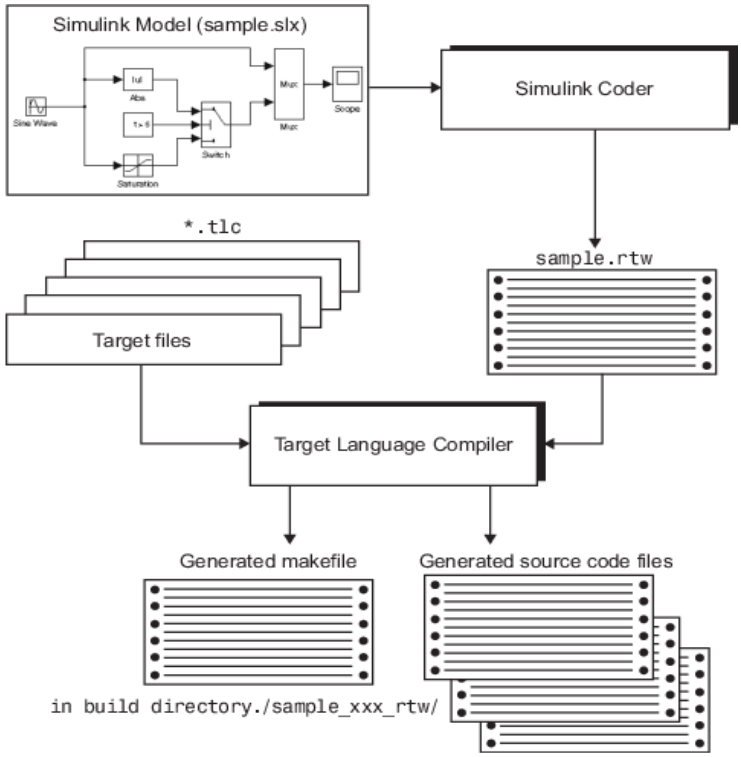

Fig. 1. Integration of the TLC into the code generation process.

The custom SMP.TLC STF is accessible directly from the configuration parameters panel of a Simulink model (see Fig. **2**). As such, it provides a user interface to the SMP code generation process from within Simulink.

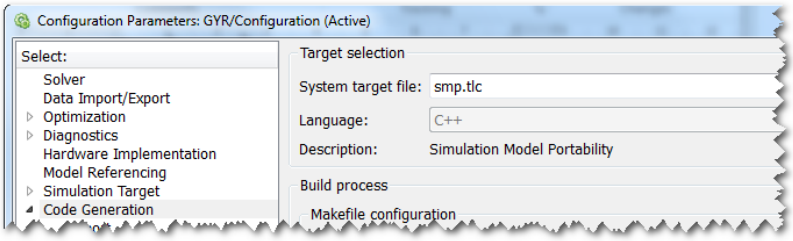

Fig. 2. Configuration Parameters panel with smp.tlc selected.

In the custom panel 'SMP Specific options', Simulink users can select SMP-specific code generation options, such as the target environment (see Fig. 3).

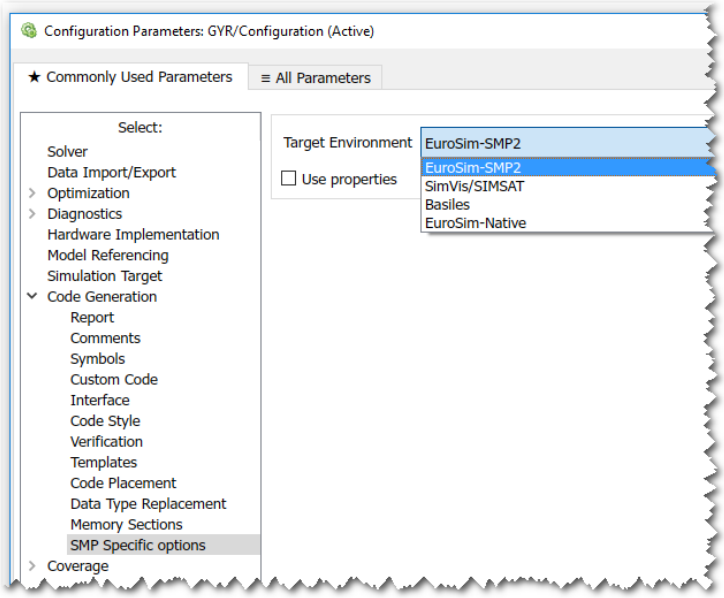

Fig. 3. SMP Specific options panel.

INTEGRATION OF NEW AND TRADITIONAL TRANSFER METHODS

During the feasibility and harmonization study of the new and traditional approaches [7], a division was made between MOSAIC functionality specific to Simulink input models, functionality that was generic for all types of input models, or functionality specific to one of the other input model types (i.e. EcoSimPro and 20-sim). To avoid duplication of code and for ease of maintenance, only functionality specific to Simulink should be removed from the MOSAIC codebase and moved to the STF. Note that the actual generation of files by MOSAIC is a generic functionality, to be retained by MOSAIC's traditional codebase. The Simulink functionality encompasses the analysis of Simulink input models. The complexity of this functionality can be reduced by moving it to STF. A significant advantage of this approach is that the meta-information of a Simulink model is accessed directly, rather than by parsing the exported C code (as was done with the traditional approach). Furthermore, less development effort is expected during updates of MOSAIC to new MATLAB releases. This increases the alignment of MOSAIC with future MATLAB versions.

An integrated transfer tool has been created that enables all porting cases started in Simulink through STF. This enables generic and cases not created in Simulink through the (traditional) MOSAIC GUI or command line version. This is illustrated in Fig. 4.

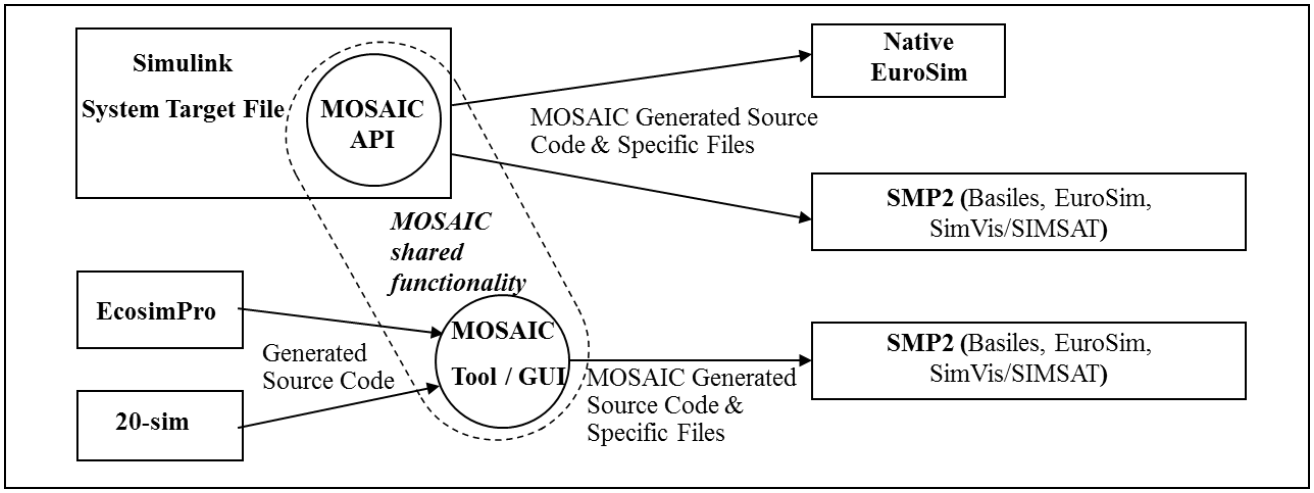

Fig. 4. Porting scheme of the integrated MOSAIC/STF tool.

MOSAIC has been integrated into STF as a dynamic library with a specific Application Program Interface (API). MOSAIC core functionality has been made available as a C++ library. A C++ API was created on top of the MOSAIC model transfer functionality such that the relevant methods are externally exposed and can be invoked programmatically from within the STF. The STF (using TLC files) is based on the MATLAB language and as such, enables the loading of a dynamic C++ library and use of the API that it exports. Fig. 5 illustrates the layered structure of the integrated MOSAIC/STF. The 'base' library contains all MOSAIC core functionality. The left stack represents the integration of the MOSAIC core with STF. For a Simulink use case, the STF covers all specific functionality and interfaces for the end user. It make calls into the MOSAIC library through its API. The right stack represents the MOSAIC part that is invoked in the traditional way for formats not created in Simulink (such as EcosimPro and 20-sim). With this traditional approach, the end user interfaces with the MOSAIC GUI, which calls the command line version of the MOSAIC tool. The layered structure of the right stack was already in place in the implementation of MOSAIC 10. Both the Simulink and traditional approach depends on the MOSAIC library.

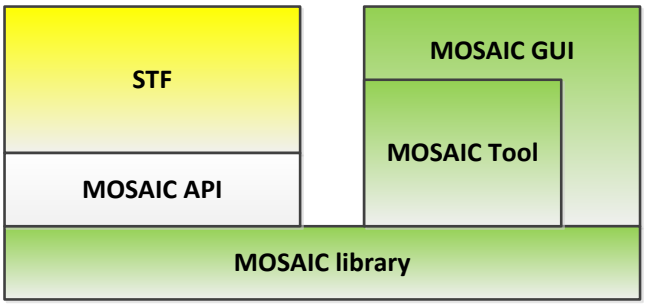

Fig. 5. Illustration of the different layers in the integrated STF/MOSAIC.

In the integrated tool, the conversion of a Simulink model into SMP compliant code appears as one step to the end user, executed by pressing the 'Generate Code' button in the Simulink UI. Underneath, the code generation is performed in two main steps (see Fig. 6)

- The TLC code generates a MAT file containing all metadata information from the Simulink model (e.g. inputs, outputs, parameters)
- Next, the MATLAB hook uses the MOSAIC dynamic library to generate the SMP artefacts and simulation environment-specific files (e.g. configuration files)

Fig. 6 STF/MOSAIC code generation workflow from Simulink

MOSAIC 11 implements the first integrated version of STF and MOSAIC. This version has been tested using standard MOSAIC test models with the SMP Conformance Suite [10]. MOSAIC 11 is considered an experimental version and has not been released externally. It forms the base for development towards an external release in which all the envisaged MOSAIC features as well as Simulink features are included. MOSAIC 11 limitations, with respect to the Simulink use case include:

- Facilitates atomic model transfer only (treating one whole model as a black box)
- Requires Embedded Coder. Specific support of Simulink Coder will be added in MOSAIC's next version.
- Only supports transfer to SMP. Support of native-EuroSim still needs to be added.

The enhancement of MOSAIC 11 is an ongoing project, see the next section for more information.

FURTHER ENHANCEMENTS

During the current MOSAIC 12 activity, the MOSAIC 11 implementation is further extended towards a tool which can be released externally. Moreover, additional MOSAIC enhancements are planned as part of the MOSAIC 12 activity. This includes enhancements of the MOSAIC library which are exposed to users of both the traditional MOSAIC application and the STF, as well as enhancements to the STF. They are described in the following subsections.

Support for Model Regeneration and Integration

Consider the two use cases:

- *Model Regeneration:* A MOSAIC user wishes to transfer an improved version of a model for a second time and automatically reuse it in an existing simulator (which already incorporates an earlier version of the transferred model). (Obviously there are limitations to this model reuse when the model's interface has been changed.) Currently, reuse without tedious manual integration is not possible as the second model version contains different (randomly generated) identifiers for the generated SMP elements, which are referenced from other SMP elements than that exist in the existing simulator. These regenerated identifiers need to be changed manually.
- *Model Integration:* In two separate MOSAIC executions, a MOSAIC user transfers two models (A and B) that share an interface (e.g. a data flow from A to B) and integrates them in the target simulation environment by interconnecting them at the interface. Currently, if models A and B both use the same data type for interfacing, their transferred counterparts end up with two generated versions of the data type that are tightly coupled to their respective models. This results in tedious manual work to "identify" these two types in some way. These versions differ in their identifiers as they are generated at random. Ideally, the two different MOSAIC runs result in two generated models (A and B) that share the types used for interfacing.

These use cases will be supported by MOSAIC 12. Below, the problem is analysed and the envisaged solution is described.

SMP elements use Universally Unique Identifiers (UUIDs) and identifiers (IDs) to reference other SMP elements at design or run time. A UUID, also known as a Globally Unique Identifier or GUID) [13], is a unique 128-bit identifier that takes the form of hexadecimal integers separated by hyphens, following the pattern 8-4-4-4-12 as defined by the Open Group. A UUID should be globally unique; but in the context of SMP, it must be unique within a simulator for the references to function correctly. UUIDs are used to identify SMP type specifications and implementations [12]. An identifier (ID) is an identification string for model elements stored in XML documents. Almost every SMP element has

an identifier that enables it to be a URL target in an SMP document. Therefore, an identifier must be unique in the space of identified elements, i.e. within an SMP document [12]. When transferring a model, MOSAIC (up to version 11) generates unique UUIDs. Moreover, it generates unique identifiers based on a random UUID.

MOSAIC 12 is foreseen to support these two use cases with:

- Generation of a *deterministic* ID based on the element name and the parent's element name (recursively); such that an identifier will still be unique within its file and remain the same for every transfer of the element.
- Generation of a *deterministic* UUID based on the element name and parent (namespace) name (recursively); such that they will be the same for every transfer of the element and also unique for all practical SMP purposes. This is accomplished by taking the concatenation of the parent and element names (which is assumed to be unique in the context of a simulator) and generating a 128-bit MD5 hash [14] from it (the MD5 hash of a given input is the same every time it is calculated, and the MD5 hashes of two different inputs are assumed to be different). Note that both an MD5 and a UUID are 128 bit values. Next, the MD5 hash is modified somewhat to obtain a proper UUID (a so-called RFC4122, name-based version UUID). After this second step, it is still expected that two different inputs result in two different UUIDs. Therefore, the generated UUIDs may be assumed to be unique in the context of a SMP simulator (with a probability of exception of order 10^{-38}).
- Generation of a separate C++ header file, C++ source file, and SMP catalogue file for each generated type; such that SMP specification and C++ code for each type can exist in isolation. Currently MOSAIC generates a single header and source file for all supporting types of a model, and a single catalogue per transferred model (containing the specifications of the model and all supporting types).
- Generation of model-independent names for supporting types. Note that these names are the input for ID and UUID generation. Identical names for two types therefore result in identical IDs and UUID for the type which is identical as far as SMP is concerned.

Changes in generated code result in a transferred model that is identical each time it is generated; existing external references to a generated ID or UUID no longer break.

To support the Model Integration use case, the generated model consists of easily manageable parts (separate files) on the level of individual and model-independent supporting types. Users can easily manipulate these files so that models may share the same set of types. This allows for easy reuse of interface types.

Use of the MOSAIC API for Third-Party Applications

The MOSAIC API is part of a released MOSAIC product and consists of a C++ header file specifying and documenting the interface. Its functionality is implemented in a dynamic library. Any third-party application may use the MOSAIC API to integrate MOSAIC functionality. This is illustrated in Fig. **7**. Note that the header file and dynamic library are an integral part of the licensed MOSAIC product. The usage of the MOSAIC API requires a valid MOSAIC license key, provided by NLR.

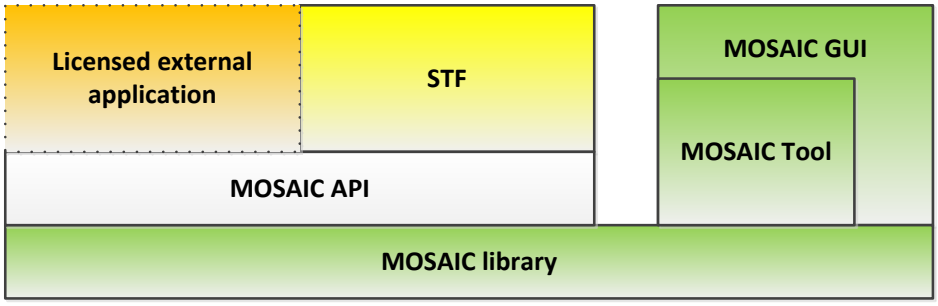

Fig. 7 MOSAIC functionality used from a licensed external application.

The dynamic MOSAIC library is delivered in such a way that all required supporting code is statically linked within it. It can therefore be loaded within Simulink or any external application without adding additional libraries.

Improved Support for Multiple Subsystem Transfer

Consider the following Multiple Subsystem Transfer use case: a MOSAIC user wants to transfer multiple subsystems that are part of the same Simulink system as separate SMP models (e.g. to allow easy replacement in a later stage of the project). The user also wants to obtain a functionally correct SMP simulator that integrates these models just as their counterparts in Simulink. The STF currently supports transfer of an entire Simulink model as a single unit, or the selection of a subsystem that is transferred by MOSAIC—again as a single unit. If multiple subsystems need to be transferred, this can be achieved by multiple independent MOSAIC runs using the STF. In MOSAIC 12, the transfer of multiple subsystems is supported within one STF run (which was already possible with the traditional approach, using MOSAIC 10). In addition, MOSAIC 12 provides support to restore the data flows between the subsystems.

To obtain a functionally correct simulator (when transferring multiple subsystems at the same time), the output models of MOSAIC need to be integrated with each other and optionally, with models originating from another source (e.g. handwritten model source code). This integration is a manual process that involves:

- Reordering the models in the schedule's executing phase.
- Restoring the data flow (from model A to B, for example); by adding source code to transfer outputs of model A to inputs of model B (corresponding to the input and output ports in the original Simulink model); and adding the data flow code to the schedule.

While the scheduling of entry points and data flows is a problem that requires knowledge of the simulation, MOSAIC 12 generates data flow "building blocks" to help users restore data flows. This includes:

- For native EuroSim model transfers, a Model Description (MD) file and a Parameter Exchange (PE) file to describe data flows. Using the EuroSim schedule editor, users can easily schedule the actual data transfers.
- For SMP model transfers, the generation of FieldLinks as part of the Assembly file. Using an SMP model builder, users can easily add these FieldLinks to the SMP Schedule file.

Enhancements of the STF

From the STF point of view, the following list of enhancements has been identified:

- Support of Simulink Coder (complementary to Embedded Coder).
- Updates of the Embedded Coder report with the newly generated MOSAIC files.
- Support of the Model Reference block. The Model block allows inclusion of a model as a block in another model. This feature permits componentization and eases the management and test of important and complex system architectures.
- Support for tuneable parameters with structures. Currently Simulink users cannot generate SMP compliant code when the model contains a tuneable structured parameter. For example, a structured parameter such as "myStruct.data", used in a Gain block when this data is tuneable.
- Model transfer from Simulink to native EuroSim models.
- Multiple model transfer of Simulink submodels and reconnection of data flows between the submodels. See previous chapter for more information.

CONCLUSIONS

From the latest MOSAIC development activities, it can be concluded that the Simulink approach using STF has been proven feasible and useful for the realization of an integrated MOSAIC product which can be released to the user community. MOSAIC 11 (not released externally) forms the base version for development towards an external product release (MOSAIC 12), in which all envisaged MOSAIC features, as well as Simulink features are included. MOSAIC 12 features a tight integration with Simulink, while transfer of EcoSimPro and 20-sim models is still achieved via the traditional approach.

Moreover, MOSAIC 12 covers advanced use cases. Regeneration of code from an improved version of a model results in seamless integration with an existing simulator containing an older model version (assuming that the model interface is still the same). Automatic model integration at the interface-type level is achieved by "identification" of these (shared) types during MOSAIC transfer. Integration of multiple subsystems is supported by generation of building blocks for restoring data flows.

The tight integration with Simulink is realized through a specific STF that invokes MOSAIC core functionality through an API, in order to generate SMP compliant code. A significant advantage of this approach is that the meta-information of a Simulink model is accessed directly, rather than by parsing the exported C code. Furthermore, less development effort is expected during updates of MOSAIC to new MATLAB releases. This would increase the alignment of MOSAIC with future MATLAB versions.

The dynamic library which is part of the MOSAIC 12 release not only achieves integration with Simulink but also allows any licensed third-party tool to integrate MOSAIC model transfer functionality in the future. The traditional MOSAIC GUI and command line tool are built on top of the same underlying MOSAIC library. No model transfer functionality is duplicated, which reduces maintenance cost.

These new MOSAIC developments (funded by ESA and developed jointly by NLR and MathWorks Consulting) provide the space community with easy-to-use and up-to-date tooling for automated model transfer that is flexible to new input and output formats and can be easily maintained. MOSAIC is now also available through the MathWorks Connections Program [15].

Besides SMP, other model integration standards, such as Functional Mock-up Interface (FMI) [16] will be investigated as well. Such investigation, in conjunction with MOSAIC development, fits in with the high-level objective to reduce the simulator development costs by effective rationalization of simulators and harmonization of simulation tools.

REFERENCES

[1] http://www.mathworks.com/
[2] SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099, Issue 1, Revision 2, 2005/10/28, ESA/ESOC, Darmstadt, Germany
[3] http://ecss.nl/standards/ecss-overall-documents-status/
[4] http://www.ecosimpro.com/
[5] http://www.20sim.com/
[6] EuroSim Mk5.2 Software User Manual, http://www.eurosim.nl/support/manuals/manual_5_2/pdf/SUM.pdf
[7] Wim Lammen, David Jaffry, Jeroen Moelands and Quirien Wijnands, Connecting MATLAB to the SMP2 Standard, Harmonizing new and traditional approaches for automatic model transfer, Proceedings SESP 2015, ESTEC, Noordwijk, The Netherlands, NLR-TP-2015-493.
[8] W.F. Lammen, “Automated model transfer from MATLAB R2014a/Simulink, EcosimPro and 20-sim to ESA's Simulation Model Portability SMP2 standard and the real-time simulation engine EuroSim, MOSAIC Release 10.0: User Manual”, NLR-CR 2014-420.
[9] http://www.mathworks.com/help/releases/R2014a/rtw/tlc/what-is-the-target-language-compiler.html
[10] SMPCS Software User Manual v1.0 - SPB- SMPCS-874-SUM-001-15.01.2010.pdf
[11] W.F. Lammen, “MOSAIC 11.0: User Manual”, NLR-CR 2015-524, February 2016.
[12] SMP2.0 MetaModel, EGOS-SIM-GEN-TN-0100, Issue 1, Revision 2, 28/10/2005, ESA/ESOC
[13] UUID specification, Network Working Group, https://www.ietf.org/rfc/rfc4122.txt, July 2005.
[14] MD5 specification, Network Working Group, https://www.ietf.org/rfc/rfc1321.txt, April 1992.
[15] https://mathworks.com/products/connections/product_detail/mosaic.html
[16] https://www.fmi-standard.org/