Presentation of a Model Integration Concept for Real Time Simulation

Oliver Ellis[1*]

[1]Institute of Flight Systems, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Braunschweig, Germany; [*]*oliver.ellis@dlr.de*

Abstract

Developing and integrating a model for a simulation system is a challenge. This process can be broken down to a consecutive progression of steps. During a modelling step, the model designer sets up a project and creates the model, thereby using a designated tool. In the next step, the model must be transformed into a format that is readable for a targeted simulator so that it can be executed. Thereby it may be necessary that the compiled model is compatible with provided dependencies that possibly be needed to be exported. Between these two steps, may include intermediate steps.

This underlying schema can also be identified at the work with the Air Vehicle Simulator (AVES) at the German Aerospace Center (DLR) in Braunschweig. AVES can run models on two targets. Both the execution with Windows and with the Unix-like real-time operating system QNX is possible. To deploy a model in AVES, it is first designed with MATLAB / Simulink. With MATLAB Coder C++ source code is generated from the Simulink model. At this stage, the generated source code is now linked against the exported dependencies that are provided by the AVES operation team and then build to a target application. This target application can now be used in the simulator.

This model integration process is subject to efforts for standardisation in order to keep it as uniform as possible. A first attempt involves a prepared MATLAB-script as a template that automates the process from the Simulink model to the target application as much as possible. The modellers can copy the file and edit it by setting configurations like a project name, step time and IP addresses. However, two problems are revealed here. Model designers may be inexperienced in software development and prefer a solution that enhances the user experience in the model integration process. Therefore, extended support might be necessary. For both the designer and the AVES team, this can disrupt the workflow. The other problem is that designers may change the script so that the support is then complicated. This set of individual scripts decreases the uniformity.

To address the above discussed problems the development of the AVES Model Builder is initiated. The AVES Model Builder provides a GUI over which a model designer handles the model integration process. Over the GUI the designer is able to create a profile at first. A profile is a JSON-file that contains individual information of the designed model which include the location of the Simulink model, the project name and a list of dependencies. The profile is then stored in the folder of the Simulink project file. So the AVES Model Builder can load a model by reading the profile. During the configuration the designer can choose for which target the project should be build and in which MATLAB version the Simulink model is designed. Any further MATLAB-code can be implemented in an initial script, that is referenced in the profile. The AVES Model Builder generates at some point in the process a standard MATLAB-script that references to these additions. This way a uniform method is established. Moreover, by eliminating the necessity to directly edit a template script through establishing a GUI, the user experience should be improved.

For the integration, the AVES Model Builder provides aside from Load Profile and Configure Profile four options as buttons, export dependencies, generate code from the Simulink model, build a target application and run all. The first three options correspond to the model integration steps discussed earlier. Run all groups these three options together so that with one click all three steps are run in one step. Export dependencies downloads libraries that are linked against the generated source code. Generate code opens a MATLAB console to invoke MATLAB Coder for the source code generation. In the process the MATLAB console runs the earlier mentioned generated standard script. The build target option is responsible for building the

target application. This step is comprised of intermediate steps which may vary depending on the target. A build for Windows include the build automation software CMake that generates a Microsoft Visual Studio project. Then the AVES Model Builder invokes the Microsoft build system MSBuild to compile the source code. In case for a QNX target build, a project template is used. The AVES Model Builder copies the template to its designated location and inserts information from the profile. Then the QNX build system mkbuild is invoked. In the end, a binary file is available that can be deployed into the simulator.

The AVES Model Builder also provides a headless mode, so it can be used with arguments in the command line. This provides the possibility to invoke it for continuous integration.

At the time of submission of this work, the development is still ongoing and further features yet to be implemented. In the current state the AVES Model Builder is a local application that is started by the model designer. However, for this software to work, it requires other software to be installed locally. Because of the strict licensing policy of Mathworks for MATLAB, Simulink and MATLAB Coder, it is a problem that a designer may not have a local installation on their working computer. Therefore, for future development it is planned that the AVES Model Builder provides an option for client functionality to communicate with a designated build server that consolidate the necessary software at a central spot. The model integration process is then executed remotely. Furthermore, it is planned to implement features for model validation. Also smaller features like versioning and other comfort features are scheduled. In the longer term, the AVES Model Builer is intended to become platform agnostic so that it is usable for other simulation systems than AVES.

In conclusion this work has identified the potential to enhance the user experience for the model integration process. Uniformity is created by by creating a profile that contains the necessary information. The AVES Model Builder then generates a standard script that references to the profile information. It furthermore executes the three steps of exporting dependencies, the generating code and building a target application that can be run in a simulation. The established GUI is an improvement in comparison to editing a script. It also provides a headless mode for the usage in the command line so it is usable for continuous integration. To address the strict licensing policy of Mathworks, it is planned to integrate a client / server solution to consolidate the needed resources. All in all, this project is still a work in progress.