Integrated Project Management Tool for Modeling and Simulation of Complex Systems

Sergio Tamayo[1], Stacey Gage[2]
MathWorks. Natick, MA 01760

and
Gavin Walker.[3]
MathWorks Ltd. Cambridge, England, CB4 0HH, United Kingdom

In the aerospace industry, engineers rely on fast, accurate results from simulation environments to design and develop ever more complex systems in shorter development times. An integrated approach for the development of complex aerospace systems is presented. This commercial off-the-shelf solution provides tooling to simplify the consolidation of collaborative efforts from diverse research, development, and test groups in the implementation of sophisticated simulation models. The software described in this paper is based in the MATLAB® and Simulink® products and intended to increase the efficiency of the workflow when several teams or individuals are involved. This is achieved through a unified graphical user interface in which the project manager and members of the development team can componentize the different aspects of the project model and assign tasks and files to specific teams or engineers. To better illustrate the scope of this technique, a case study for a complex aerospace model is provided based on the HL-20 personnel launch system.

I. Introduction

THE growing complexity of aerospace systems, combined with the need for increased accuracy and simulation speed, has, in turn, increased the complexity of managing the design and development of these systems. Hence, the workflows of current simulation environments include the collaborative work of individuals and teams delivering specific components in their respective area of expertise. Such areas could include guidance and control, aerospace structures, computational fluid dynamics, and testing. The approaches for the implementation of architectures that support collaborative efforts in software development have proven successful in other areas such as manufacturing[1], social systems[2], and robotics[3], as well as other more specific applications such as aerospace systems[4].

An integrated approach for the development of complex aerospace systems is presented, using a commercial off-the-shelf (COTS) solution that provides tooling to simplify the consolidation of collaborative efforts from diverse research, development, and test groups in the implementation of sophisticated simulation models. The software described in this paper, Simulink® Projects, is based in the MATLAB®[5] and Simulink®[6] products and is designed to increase workflow efficiency when several teams and individuals are involved.

Simulink Projects, shown in Fig.1, supports interactive integrated development and management of simulation models, data files, and script files. Using the unified graphical user interface, the project manager or the team itself can identify the different aspects of the project model and assign tasks and files to specific teams and individuals. The assigned tasks can be as diverse as developing, testing, or verifying aspects of the project. Having a unified user interface to assign and coordinate efforts on large scale simulation models simplifies and improves team collaboration in the design and implementation of those models.

[1] Senior Software Engineer, Control Design Automation Department, 3 Apple Hill Drive, AIAA Member.
[2] Simulink Applications Manager, Control Design Automation Department, 3 Apple Hill Drive, Senior AIAA Member.
[3] Development Manager, Simulink Model Management Team, Control Design Automation Department, Matrix House Cambridge Business Park.

This approach not only improves workflow efficiency via parallelization of tasks, but also provides easy integration to the simulation environment since Simulink Projects is executed within the MATLAB environment, increased flexibility of the simulation environment due to componentization, and simplified hardware deployment via automatic code generation.

This paper is organized as follows. The case study of the HL-20 model example and its componentization are briefly described in Section II. The integration of the HL-20 case study with Simulink Projects is described in Section III. Finally, conclusions are drawn in Section IV followed by a list of references.

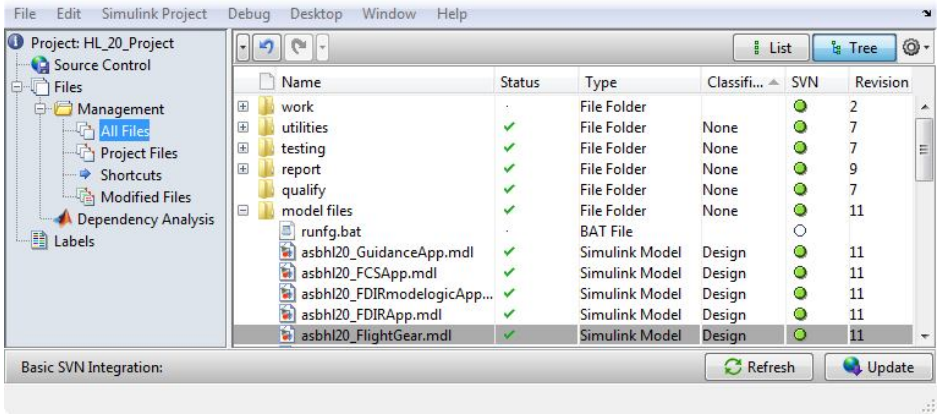

Figure 1. Graphical user interface for Simulink® Projects with the HL-20 files.

II. Case Study: HL-20 personnel launching system

The HL-20 lifting body[7] was originally designed by NASA as a personnel launching system (PLS) that would be launched into orbit by means of either the Space Shuttle Orbiter or a booster rocket. For re-entry, the lifting body would use an on-board propulsion system and then execute a nose-first maneuver. The lifting body was designed to carry up to 10 people. A scaled model can be seen in Fig. 2.

Figure 2. HL-20 personnel launching system.[8]

A comprehensive simulation model of the HL-20, shown in Fig. 3, includes several different modules, such as an environment module, a simulation interface module, and a guidance and control module. The simulation interface module includes an interface between the simulation software and FlightGear[9], a third-party open source flight simulation application. The interface to FlightGear software supports bidirectional communication between the simulation model and the animation software. FlightGear provides renderings of geographical locations that are accompanied by topography surfaces that can be sent back to the simulation environment in the form of a variable such as the altitude above ground. The guidance and control module has evolved from an inner glideslope feedback controller, and several other modules such as fault detection and a heading control system to reject lateral wind gust disturbances[10] have been added.

The HL-20 model[11] comprises more than 5,500 individual blocks. It has undergone a process of componentization following a strategy for enterprise-wide modeling[12]. This modeling strategy is aimed at the improvement of the efficiency of developing an enterprise-wide design, defined as a model within an organization

that is too large for one person to know all the details. Hence, the model is linked with the organization such that a top-level team that shares knowledge of all the different components distributes the modeling and development tasks among different independent work teams that are dedicated to a single component of the model.

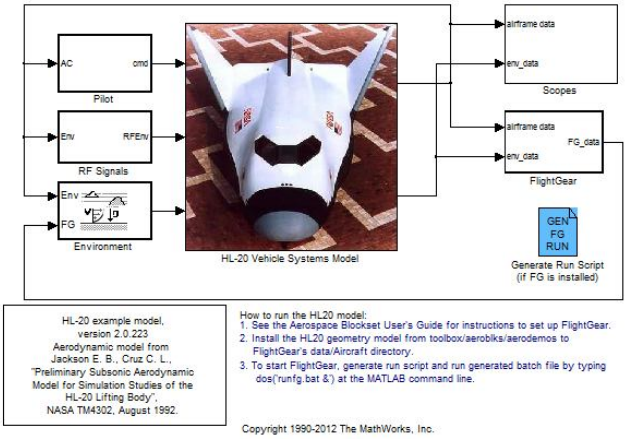

Figure 3. HL-20 top level model.

Efficient collaboration on an enterprise-wide model can be improved if control over the software environment is exercised, and a common path to the files is provided. Control over the software environment can be achieved with startup files that determine environment variables such as the MATLAB path, confirm or change compiler settings, provide access to shared utilities, and so on. For the case study, a startup file that defines the different libraries and loads data has been created.

Enterprise-wide modeling can be hampered when different teams rely on physical documents to store information regarding the model and its performance. With Model-Based Design[12], communication between teams is done using the models themselves and report creation can be generated automatically using up to date data from the models themselves. For the case study, an automatic report generation file[14], was created so it could be used for analysis among the teams.

The enterprise-wide modeling strategy dictates that the design components of a model be architected so that they can be individually developed and tested via automation. When the testing is performed automatically, the problems with integration of new or improved components are mostly limited to interface and system level issues. Stand-alone component models can be integrated directly into the top-level model. Furthermore, since stand-alone component models run independently, they can be integrated to the test harness in a straightforward manner. For the case study, it was found that specific portions of the flight control system were best developed in independent models. Hence the guidance and fault detection models were integrated into the HL-20 model using Simulink Model Reference blocks. The required variables for each of these flight control models have been extracted from the main data file to enable development and testing on the individual models. Furthermore, they can be implemented in such a way that they can be reused in the model, such as in redundancy systems.

Libraries are an alternative to Model Reference blocks. Libraries[13] enable the reuse of blocks inside multiple models and involve independent files, which work well with source control software. Libraries for different subsystems of the HL-20 model, such as environment blocks, communication systems, actuator subsystems, and the GPS subsystem were developed.

The enterprise-wide modeling strategy also calls for developing a specific architecture for interfacing each of the blocks to decrease issues with integration. This interface is created by defining the signals, data types, and sample times that should be passed to and from each of the component models. For the case study described in this paper, bus objects were used for interfacing the different components. These bus objects support the specification of the names and characteristics of each of the signals, and are stored in specific data files shared by multiple components. As an example, the AirData bus defines the interface between the air data computer and a redundant avionics bay.

III. Project management for the case study

The solution proposed in this paper incorporates Model-Based Design techniques that provide a seamless transition between the mathematical model and the real life system. Simulink Projects can be linked to source control software enabling all team members to access both the latest working configuration of the files in the project, as well as earlier revisions. This enables early detection of integration issues before the final product is developed.

An additional way to detect integration issues is by performing an analysis of the file dependencies within the project. A graphical view of the results of such a file dependency analysis can be obtained within Simulink Projects. Such a view enables the engineer to see how his or her module fits in the development workflow, and helps the project manager to allocate resources in the development of each subsystem.

Deployment of the aforementioned case study into a project is described next. To improve the efficiency of simulation and testing of an aerospace solution, simulations are often performed in batches. Under these conditions, the inclusion of FlightGear can slow simulations. The ability to not to run this particular component can improve overall simulation time, especially for batch testing. Hence, the HL-20 model was updated to use a variant subsystem block to better accommodate batch testing. The variant subsystem block lets the engineer choose one subsystem from various subsystems for simulation. This selection can be performed without modifying the model itself.

The variant subsystem can be selected from the different variants for the simulation either programmatically or through a graphic user interface. The first variant includes the interface to FlightGear. The second variant does not use the input to the subsystem and generates outputs by using previously saved data stored in a .MAT file. The third and last variant, like the second, does not use the input, but generates outputs based on a specifically designed signal created using the Simulink Signal Builder block. The implementation of this feature is shown in Fig. 4.

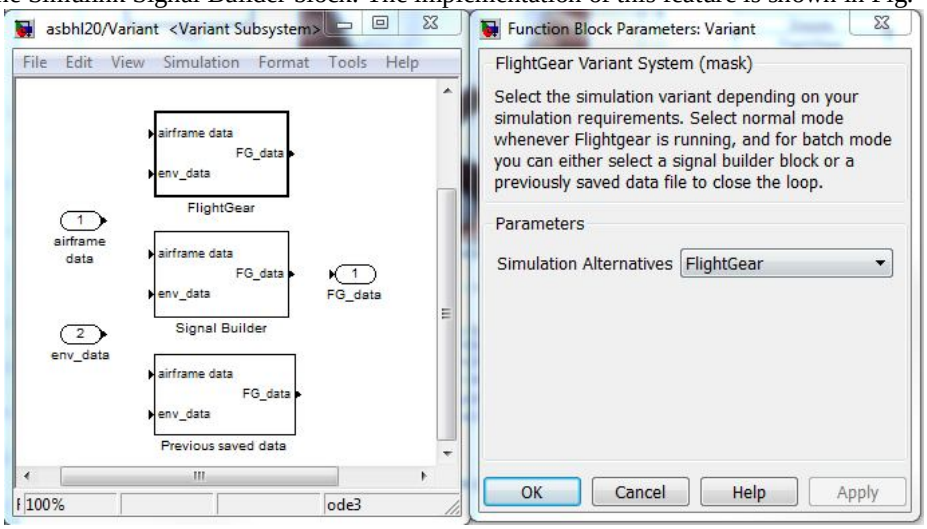

Figure 4. Implementation of a variant subsystem for the HL-20 model.

The file structure described in this text is not unique and it was designed for projects that use Model-Based Design. In the case of the HL-20 project, a top level model named asbhl20 is located in the model files folder, along with model referenced files such as the guidance and flight control system applications, as can be seen in Fig.1.

Within the project, additional folders for simulation model files, model testing, model qualifying, report generation, utilities, and work files were created. The *model files* folder contains all model files required for simulation, such as the top level file (asbhl20.mdl) and the environment subsystem, containing the turbulence and wind models. The *testing* folder includes automated regression test files that preserve previous simulation behavior. Additional tests for automated baseline comparison are included in a *qualify* folder. These baseline comparison tests compare the model results with a set of either flight test data or independently created data, and can qualify iterations of the model. The *report* folder contains files that automatically generate reports based on simulation data. These generated reports can be used to share data between developing teams and external sources such as subcontractors or other teams within the company. A *utilities* folder containing startup and clean up tasks is also included. The startup tasks include scripts that are run whenever the project is started, to set the same initial

conditions for every simulation run. Furthermore, the startup files also specify the paths for the different folders and avoid file shadowing issues. Clean up scripts are run whenever the project is closed and can be used to clear variables from the workspace so they are not improperly used for other tasks. The *work* folder contains files created by the simulation environment to run the simulation.

As mentioned previously, Simulink Projects can be interfaced with source control software. In this case study the built-in support for Subversion®[15] software was used, and a corresponding Subversion server was set up to enable the project files to be shared by the entire team.

To begin work on the shared project, each team member creates a working area, or sandbox, for their version of the project. Once the sandbox is created changes to the original model can be made: in this case, the changes include the addition of a variant model in the top-level model file. Since the variant model is a self-contained component, the block was added to a specific library, and placed in the libraries folder. The changes for the project are visible in the "Modified Files" pane, as shown in Fig 5. Labels can be added to files describing the review status, such as "to review", "accepted", and "rejected", to support a formal, auditable design process. The auditable design process, generally included in the development cycle, usually includes several steps before the changes are integrated, such as an approval from a quality engineer, a fellow member of the team, or a member of the project management team.

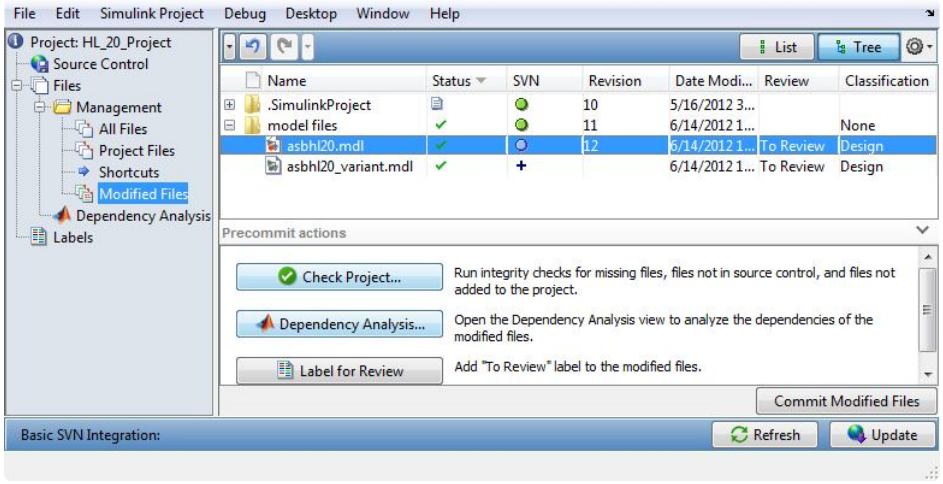

Figure 5. Modified files list pane.

Changes in files in the Modified Files view are tracked using the comparison and merge tools available with MATLAB and Simulink. Fig. 6 shows a Simulink XML Comparison[16] of the submitted model with a previous version. The previous version of the file is shown on the left and the one that includes the changes on the right. Among other minor changes, this view shows that the file in the sandbox includes a variant subsystem.

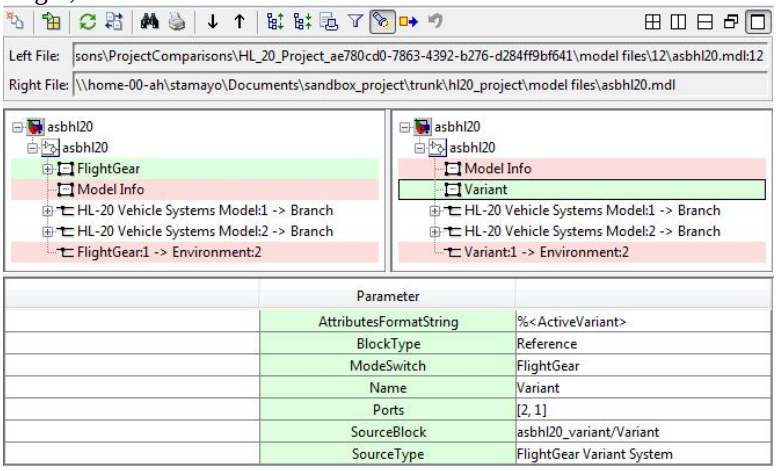

Figure 6. Comparison tool with Simulink Projects.

A dependency analysis was performed upon the project using Simulink Projects. The results of this analysis are shown in Fig. 7. The different components of the top level file are shown along with each of the model reference files, libraries, and MAT-files. As can be observed, there is a clear dependency between the model reference files and the asbhl20GlobalChildrenData.mat file, which contains the bus object AirData, which in turn determines the interface between the model reference files. Since this file determines the interface between different teams, it is suggested that once changes are made to this file, it is submitted for approval to members of the other teams so updates can be done on each of the components that they develop to avoid affecting the integrity of the simulation. This procedure proves especially important when the project teams are in different locations and communication issues are a concern.

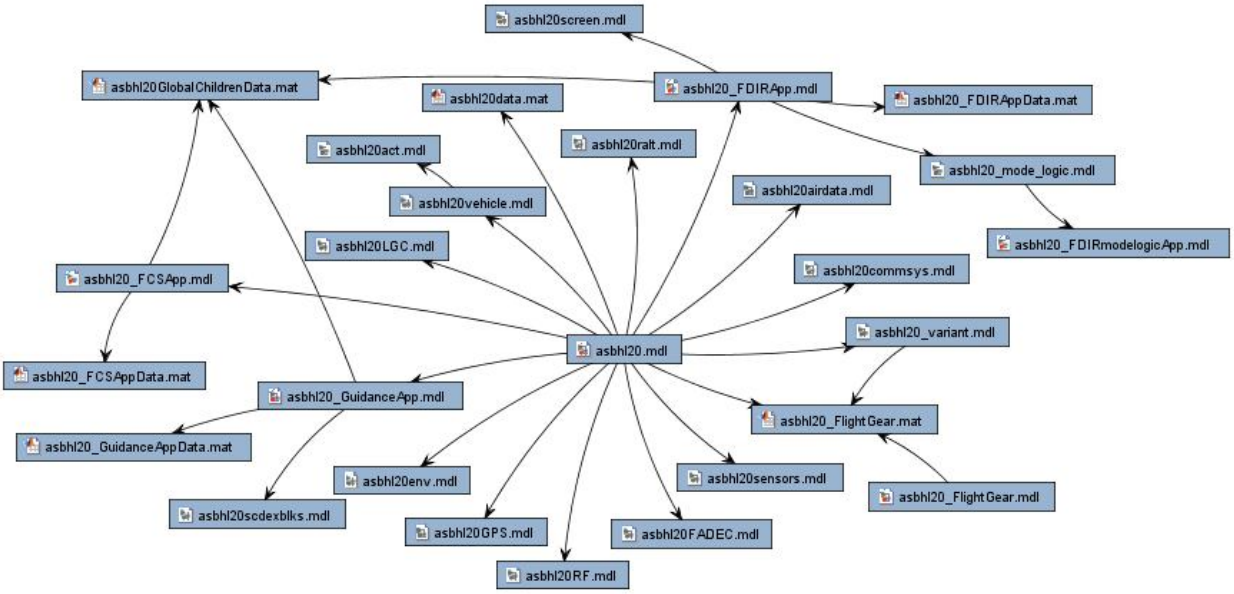

Figure 7. Dependency analysis for the HL-20 project.

IV. Conclusion

A COTS tool based in MATLAB and Simulink, called Simulink Projects, that supports the management of files and data associated with Model-Based Design strategies for the aerospace industry was presented. This COTS tool enables a more efficient workflow when performing large-scale modeling tasks by making it easy to partition and share work, in a standardized environment, across the whole team.

Improvements in the workflow are achieved in part by reducing the number of steps for a new team member to get started with an existing project via startup shortcuts that can provide a standard working environment across a team. Easy discoverability and management of all project files coupled with tight integration with tools for version control, peer-reviews such as file labeling, and comparison tools reduces the learning curve to use this streamlined workflow.

Componentization techniques were identified as a key factor in improving development efficiency on a large scale model. These techniques work well with configuration management tools, which improve management of multiple files. Furthermore, it was shown that componentization issues can be identified with the use of dependency analyses in Simulink Projects.

A case study based on the HL-20 personal launching system was presented, including the workflow that was required for introducing changes, implementing a version control system, and performing automatic report generation and dependency analyses.

References

[1]McLean, C., Leong, S., and Riddick, F. "Architecture for Modeling and Simulation of Global Distributed Enterprises," *Proceedings of the ASIM 2000 Conference*, February, 2000.

[2]Scerri, D., Drogoul, A., Hickmott, S., and Padgham, L. "An Architecture for Modular Distributed Simulation with Agent-Based Models," *Proceedings of the 9th Conference on autonomous Agents and Multiagent Systems,*AAMAS, Toronto, Canada, May 2010.

[3]Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. "MASON: A Multiagent Simulation Environment," *SIMULATION - Transactions of The Society for Modeling and Simulation International*, Vol. 81, No. 7, July 2005, pp. 517-527.

[4]Schum, W., Doolittle, C., and Boyarbo, G., "Modeling and Simulation of Satellite Subsystems for End-to-End Spacecraft Modeling," *Modeling and Simulation for Military Applications, Proc. Of SPIE*, Vol. 6228, May 5, 2006.

[5] MATLAB, Software package, Ver. 7.14. The MathWorks, Inc., Natick, MA, 2012.

[6]Simulink, , Software package, Ver. 7.9. The MathWorks, Inc., Natick, MA, 2012.

[7]Jackson, E., Cruz, C. "Preliminary Subsonic Aerodynamic Model for Simulation Studies of the HL-20 Lifting Body," NASA, TM4302, August 1992.

[8]Schultz, James. "The HL-20 experimental aircraft mock-up". *NASA Langley Research Center – Multimedia Repository.* NASA, October 22, 1990.

[9]FlightGear, Software package, Ver. 2.0. FlightGear. Minneapolis, MN, 2011.

[10]Glass, J. "Tuning Multi-Loop compensators to meet time and frequency domain requirements", *MathWorks Aerospace and Defense Conference*, June 14th, 2006, Reston VA.

[11]Aerospace Blockset, Ver. 3.9, The MathWorks, Inc., Natick, MA, 2012.

[12]Aberg, R., Gage, S. "Strategy for successful enterprise-wide modeling and simulation with COTS software". *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Providence, RI, Aug. 16-19, 2004

[13]Walker, G., Friedman, J., Aberg, R. "Configuration Management of the Model-Based Design Process," *SAE World Congress & Exhibition*, Detroit, MI. April 2007.

[14]MATLAB Report Generator, Software package, Ver. 3.1.2. The MathWorks, Inc., Natick, MA, 2012.

[15]Subversion, Software package, Ver. 1.7.5. Apache Software Foundation. Delaware, 2012.

[16]Simulink Report Generator, Software package, Ver. 3.1.2. The MathWorks, Inc., Natick, MA, 2012.