Jumpstarting Team Collaboration in Model-Based Design

Saurabh Mahapatra
MathWorks
3 Apple Hill Drive
Natick, MA 01760
saurabh.mahapatra@mathworks.com

Joachim Schlosser
MathWorks
Adalperostraße 45
85737 Ismaning, Germany
Joachim.Schlosser@mathworks.de

Soeren Roeder
Marc Segelken
MathWorks
Adalperostraße 45
85737 Ismaning, Germany
soeren.roeder@mathworks.de
marc.segelken@mathworks.de

ABSTRACT

In this paper, we introduce a new tool that enables collaboration in Model-Based Design. To accomplish this, we outline a recipe within the context of tool capabilities that will motivate organizations large and small, to jumpstart team initiatives while lowering costs.

Categories and Subject Descriptors

K.6.1 [**Management of Computing and Information Systems**]: Project and People Management – *systems analysis and design, systems development.*

General Terms

Algorithms, Management, Documentation, Design, Standardization.

Keywords

Simulink, Projects, Subversion, Source Control, File Comparison Merging, Documentation

1. INTRODUCTION

With advances in computing technologies, recent trends in Model-Based Design show a focus on building higher fidelity models that result in better and more robust designs. The result has been the proliferation of projects that involve multidisciplinary groups of engineers that must collaborate to realize these models. Associated with these workflows are inherent complexities arising out of engineers having to organize, manage, and revision control files that contain algorithm implementations, data, utilities, and associated report artifacts. Consequently, the complexity faced by the engineer is three-fold: design, file and configuration management. The evolution of tools specific to these three tasks has led to a disjointed workflow where the engineer's attention is divided among these tasks resulting in diminished productivity and effectiveness. Many teams in firms with infrastructure and budgetary constraints are finding the team collaboration concept elusive and inaccessible.

In this paper, we propose a tool that seeks to alleviate the above issues. A design-centric approach to the problem is undertaken in which the file and project management tasks are exposed to the engineer from within the design tool itself. By providing flexibility to connect the design tool to various source control tools via an authoring API, the amount of the latter tool's exposure for common tasks engineers perform can be managed, while other critical project management tasks still remain with the configuration management specialist. Using Simulink as an exemplary environment for Model-Based Design, we address these issues with a new tool called Simulink Projects.

2. SIMULINK PROJECTS

The motivation behind the Simulink Projects tool was to provide an environment within Simulink®[1] that supports the adoption of best practices such as component-based modelling, peer review workflow, simplified configuration management, and integration with source code management (SCM) tools. Standardization of workflows among all the engineers working on the same project using the tool would improve consistency of results and efficiency.

2.1 Tool Architecture

The architecture of the tool was guided by two main requirements from the design engineer's perspective– bring structure to file management tasks specific to the Simulink design environment and improve accessibility to configuration management. To transition the engineer from an individualized to a collaborative design environment requires that tool capabilities be extensible with an out-of-the-box experience for tasks that require specialized knowledge. The tool design is modularized according to the mapping shown in Figure 1 that allows teams to choose the capabilities based on desired design environment characteristics.

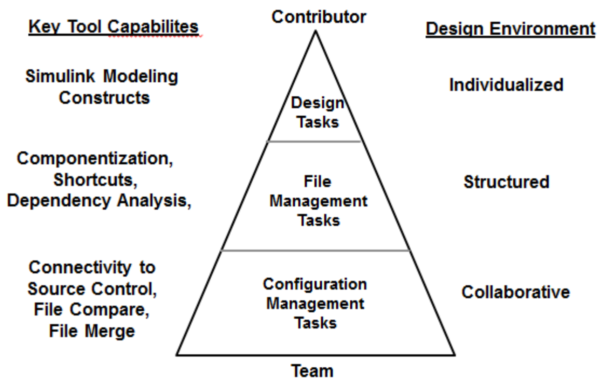

Key Tool Capabilites
Contributor
Design Environment
Simulink Modeling Constructs
Design Tasks
Individualized
Componentization, Shortcuts, Dependency Analysis,
File Management Tasks
Structured
Connectivity to Source Control, File Compare, File Merge
Configuration Management Tasks
Collaborative
Team

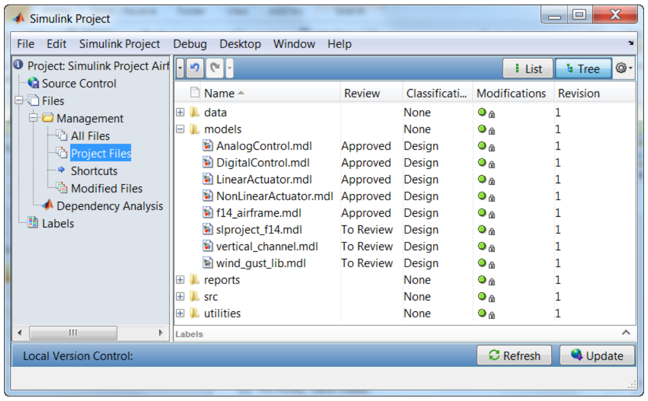
Figure 1. Mapping between project tasks and design environment mapping that motivated key capabilities design.

3. ACCELERATING TRANSITION INTO A TEAM-BASED ENVIRONMENT

The primary use case for the tool was to enable design engineers familiar with Simulink but having minimal file and configuration management knowledge, transition into a team-based environment easily. In this section, we outline the key concepts using the tool to jumpstart this process.

3.1 Componentize the Design

For a graphical modeling language such as Simulink, complex implementations can be abstracted out and componentized as block subsystems that can exist as separate design files which enables parallel development. There are two mechanisms to accomplish this-block libraries and model reference. The former is a single file that is a palette containing subsystems reusable in a design while the latter is a separate model in itself. For lightweight components, library subsystems offer a quick and easy way to make reusable components accessible to all team members without deterioration in performance. However, since each library subsystem instance is a graphical copy in memory, frequent use of these can result in simulation performance deterioration. This is precisely what model reference technology addresses by accessing the same block in memory for multiple calls. In a similar vein, the concept of variants where it is possible to choose among multiple algorithm choices for a single component is also supported by the above technologies.



Name	Review	Classificati...	Modifications	Revision
data		None	●	1
models		None	●	1
AnalogControl.mdl	Approved	Design	●	1
DigitalControl.mdl	Approved	Design	●	1
LinearActuator.mdl	Approved	Design	●	1
NonLinearActuator.mdl	Approved	Design	●	1
f14_airframe.mdl	Approved	Design	●	1
slproject_f14.mdl	To Review	Design	●	1
vertical_channel.mdl	To Review	Design	●	1
wind_gust_lib.mdl	To Review	Design	●	1
reports		None	●	1
src		None	●	1
utilities		None	●	1

Figure 2. Simulink Projects user interface showing the componentized design files with folder hierarchy

It is a best practice to discuss model architecture, criteria for componentization and file ownership early in the project. This would help establish contractual rules of engagement among the team members. For example, a project charter that grants exclusive component ownership would entail team members to respect component boundaries.

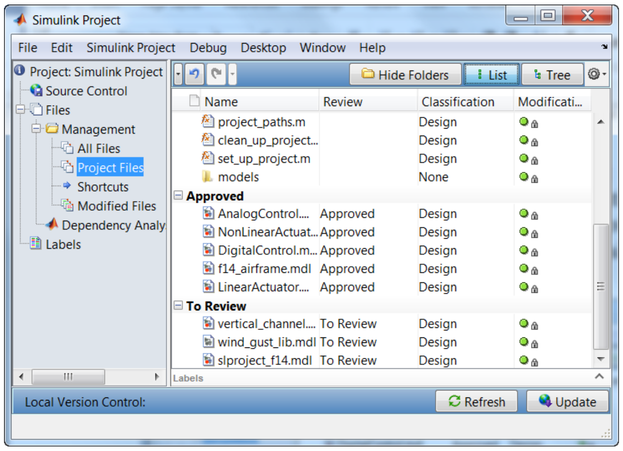
Besides the file hierarchy, it is also important to consider directory structure as an extension of componentization as it provides abstraction for separating categories of files. The Simulink Projects user interface containing the design files and the folder structure is shown in Figure 2.

3.2 Manage Project-Level Utilities Centrally

As the project evolves, knowledge is created by team members in the form of utilities such as automating environmental setup, report generation, regression testing or model standards checking. Also, accessing key files in a large scale project can be a daunting task. The lack of a unified collaborative environment increases the risk of loss of this knowledge and impedes accessibility. Simulink Projects addresses this by providing a mechanism for users to aggregate these utilities and key files centrally in a special view called as “Shortcuts”. It is also possible to assign special actions to certain shortcuts to execute at project startup or shutdown. To transition an existing project into this environment requires an audit of all utility and key files. For an ongoing project, these shortcuts when created would be made available to all since this information is contained in the project definition files contained in the source control repository.

3.3 Use Metadata to Manage Files

It is important to note that additional file attributes can be created based on its use in different contexts and workflows. For example, file review status is an attribute that exists as different engineers work on the file. Other possibilities include ownership, requests for peer review, or file classification. Representing information associated with files in such a manner helps create additional categories which can be used to manage the complexity of a project.



Name	Review	Classification	Modificati...
project_paths.m		Design	●
clean_up_project...		Design	●
set_up_project.m		Design	●
models		None	●
Approved			
AnalogControl....	Approved	Design	●
NonLinearActuat...	Approved	Design	●
DigitalControl.m...	Approved	Design	●
f14_airframe.mdl	Approved	Design	●
LinearActuator....	Approved	Design	●
To Review			
vertical_channel....	To Review	Design	●
wind_gust_lib.mdl	To Review	Design	●
slproject_f14.mdl	To Review	Design	●

Figure 3. Simulink Project Labels shown as columns that have been grouped by review status.

It is therefore a best practice to identify these attributes by creating entity relationship models and converting them into a tabular form such as relational database where each entry record corresponds to a file. In Simulink Projects, this is made possible through the concept of “Labels” which are predefined tags that can be associated with a file. As shown in Figure 3, each user can see a tabular view of a desired set of labels in column view. It is possible to define complex filters using regular expressions that can return names of files matching complex criteria.

3.4 Choose a Representative Source Code Management Workflow

Standardizing the workflow consisting of design, file and configuration management tasks across the entire team is critical for project success.

In Figure 4, we show a popular representative workflow that by many companies who have adopted Model-Based Design. The development work starts off with the creation of a local copy on the design engineer’s machine, checking out of required files to lock them, design with modeling, testing through simulation, and review by peers of the new changes. These changes are then submitted into the central repository where regression tests and additional tasks such as automatic code generation and builds are created.

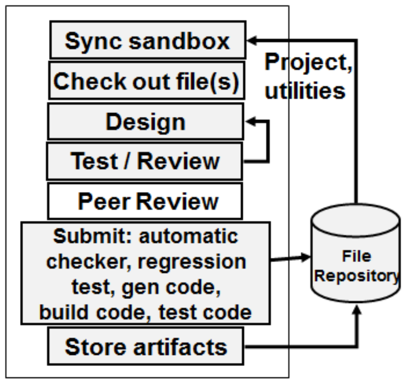


Figure 4. A popular representative source code management workflow.

Artifacts such as reports containing design information, test pass/fail results are derived from the model files and checked into the repository. Although, the sequencing of the above steps may seem logical, there are cases where they do not necessarily hold true. For example, DO-178B workflow requires a check-in of files prior to a peer review. In Simulink Projects tool, all the above actions are made available to the user through file context menus or views. For example, the “Modified Files View” shows all the files awaiting check-in that can be labeled for peer review, and analyzed for missing files.

3.5 Consider Using an Open Source Control Tool

In recent times, the popularity of open source code management (SCM) tools such as Subversion has increased accessibility and lowered costs. Based on anecdotal evidence gathered over several years, the authors would like to claim a trend towards Subversion as the preferred choice for groups working on a tight budget or practicing lean methods. This popularity can be attributed to software engineers who have shared their positive experiences with their modeling and simulation engineers. Newer releases of Subversion are providing features comparable to commercial vendors. This trend offers several opportunities for teams to jumpstart collaboration. Without having to trial and adopt a commercial SCM tool which can take several months, workflows can be prototyped to gain early insight into its efficacy. To facilitate this, Simulink Projects offers an out-of-the-box connectivity to Subversion. It also provides a Java-based software development kit (SDK) that can be used to create connectivity to other source control tools. Such modular tool architecture where design and file management features can be decoupled from those of configuration management allows Subversion to be replaced with any desired SCM tool without disrupting the existing workflow.

3.6 Ensure Project Integrity by Running File Dependency Checks

For large scale systems development, the use of componentization will necessitate the need for file dependency analysis as risk of missing files can disrupt the project. The inbuilt Dependency Viewer™ in Simulink Projects generates an architectural view of the file dependencies, checks for missing files and suggests fixes.

3.7 Conduct Peer Reviews

As the size of the team increases, establishing a feedback mechanism through peer review is essential for project. Several elements of the peer review process can be easily accomplished through the automated reporting tool called the Simulink Report Generator® integrated into Simulink Projects. Here are some examples:

- There will be a need to examine and resolve differences between conflicting versions of the same file. Rollback of changes may be required if necessary. Conflicts can be resolved efficiently within Simulink Projects using the comparison of a XML-based hierarchy representation of two Simulink model file versions.
- Team meetings involving architectural discussions were done through the web-based rendition of Simulink models called “Web Views”.
- System Design Description (SDD), requirements, model advisor, and test reports can be automatically generated from any model as artifacts archival.

3.8 Choose an Optimal Workflow for File Merge

It is important to mention that contractual rules for component ownership be established early in the process. There are two options: exclusive and mutual ownership. In exclusive ownership scenario, each component is exclusively worked on by only one user at a time. This avoids the costs associated with merging changes from multiple users. However, this can slow parallel development down as users can work on the same file serially. This limitation is removed in the mutual ownership scenario but incurs the cost of merging changes. Thus, the componentization and ownership discussion would require a tradeoff between development and merging costs.

3.9 Reuse Project Elements for a Future Project

There are several aspects of a project that may be transferable to other projects inside the organization. This transfer results in standardization of several aspects of the development process that helps build organizational memory. Opportunities abound in this area such as reusing design architecture, utilities, libraries, folder structure, and legacy code. This idea is made possible through the use of “Templates” where these elements can be added and shared with the organization.

4. CONCLUSION

In this paper, a new tool enabling collaborative development in Model-Based Design using the Simulink design environment was discussed. A recipe outlining key considerations to jumpstart the adoption process was provided. In the future, the maturity of open source tools coupled with engineers’ education on collaboration will help bring about synergies never realized before.

5. REFERENCES

[1] MathWorks, *Simulink User’s Guide*, June 2012.