MATLAB EXPO 2016

Simulink for Teams: High-Productivity Workflows

Gavin Walker, Development Manager
Who am I?

- Read engineering at Cambridge University, specializing in systems & control
- PhD & post-doc also in CU Engineering Dept
  - Achievable Responses of Vehicle Suspension Systems
  - Consultancy for F1 team on suspension systems
  - Part of team that developed the “inverter”
  - Been using Simulink since 1994
- MathWorks
  - 7 years as a consultant, working on complex simulation problems, control & model management systems
  - 10+ years heading a development team, focussed on enabling Model-Based Design in Teams
Vision

Create tools that make it easy for teams to collaborate on Model-Based Design

Collaborate

Integrate

Analyse
Overview

- Creating solid foundations for Model-Based Design in a team
- Ways to share information around a team
- Components. How, where, when.
Getting started with a legacy project
Familiar?
Experienced users might try…

- Group by type in MATLAB
- Look for largest model?
- Load all the data files you can find
- Keep pressing “run” until something works
- Not an ideal process…
Instead: create a Simulink Project from those files
What did we see?

- First, a Simulink Project: what’s that?
  - Dependency analysis
  - Shortcuts being created
**Simulink Project**

Foundation technology supporting efficient Model-Based Design in teams

- Enables sharing of work
- Ensure a project is complete (files, data)
- Consistent environment across a team (MATLAB path, data, slprj location, ...)
- Support different entry points & sharing best practice via shortcuts
- Sharable meta-data about project contents via labels
- Part of Simulink (first shipped in R2011b)

Supports advanced users

- Impact analysis: what’s the impact of a change?
- Rich APIs to script & customize
What did we see?

- First, a Simulink Project: what’s that?
  - Dependency analysis
  - Shortcuts being created
What is dependency analysis?
What is dependency analysis?

- Static analysis of all the files in your current project
- Provide information on how the different files depend upon each other
- What type of dependency
  - Model Reference, Library link, model/block callback, Data Dictionary, MATLAB code dependencies, C-code header files, ...
  - Long & ever growing list

Static analysis:
Models & code do not need to “work” (do not need to simulate, run)
What is dependency analysis?

- Show model structure
- List products required
- Highlight issues
What did we see?

- First, a Simulink Project: what’s that?
- Dependency analysis
  - Shortcuts being created
Simulink Project Shortcuts
Simulink Projects Shortcuts

- Make it easy for *any* engineer (not just engineer who created the project) to:
  - Find important files
  - Find & execute important or common operations
    - Make the top-level model in the project a shortcut
  - All debuggable
- Optionally set shortcuts to run at project startup or shutdown
  - Provides formal mechanism for running initialization scripts
  - Makes it easier to ensure the symmetric shutdown scripts are called
Demo

1. Automate loading of data when the project opens using shortcuts
2. Explore the existing utilities in the project that are not instantly accessible
Using Simulink Projects to Create a Consistent Cross-Team Environment

- We saw Project Shortcuts being used to
  - Check MATLAB version, compiler version, etc.
  - Set (and re-set) the MATLAB path
  - Make available (the right version of) company utilities
  - Load data
  - Highlight entry points into the Project

- Benefits:
  - Everyone on the team has the same environment
  - New team members can get started more quickly
  - Less wasted time debugging discrepancies
Simulink Project: A Foundation Technology

Set up the MATLAB Path for this project
No more issues with unexpected shadowed models
Set default model template
Set team preferences
Set "slprj" location
Shortcuts to key functionality
Simulink Project: A Foundation Technology

Robustly configure the team environment.
For everyone.
Automatically.
Integration with Source Control
Managing versions of files

Short aside…
How do people share & manage projects?

At an SAE webinar on “Model-Based Engineering”, asked:

Q: “How do you manage the files and data within your projects?”

1. Named folders (“project_v1”, “project_v2”, etc.)
2. Source Control tool
3. Application Lifecycle Management (ALM) tool
How do people share & manage projects?

Majority use COTS tools for managing work & sharing information

- Source control
- Application Lifecycle Management (ALM)

Surprise was number just using the file system

- Doesn’t scale well
- Doesn't support team work
- So why were they doing it?
Source Control Integrations

Published API

Simulink Project (R2011b) & MATLAB Current Folder Tool (R2014b)

Custom Integration

Published Java API + SDK for integration with other tools

Other source control tools
Source Control Integrations

Perforce Announces MATLAB and Simulink Integration with P4D

P4SL for Simulink® Extends Perforce’s Powerful Version Management and Collaboration to Scientists and Engineers Using MATLAB and Simulink for Embedded Systems Software Development

Web search for “p4sl”
Supports R2014a onwards
Source Control Integrations

Microsoft Team Foundation Server (TFS) integration available now from MathWorks File Exchange

Supports R2014a onwards

And more…
Demo: Add our project to Source Control

- Just press a button, browse to repo and done.
- Can see revision info
- Best: get a new view of the project: **Modified Files**
Working out what has changed; getting it peer reviewed
Support for team source control workflows

Part of Simulink Report Generator
Simulink File Comparison & Merge

- Merge desired changes back into main development branch
  - High level: bring in entire subsystems & Stateflow charts
  - Fine grained: individual blocks & parameters
Simulink File Comparison & Merge

- Create detailed report of changes
  - Word doc editable for peer review workflows
Source Control integration is not Essential… Just useful

- Can continue to manage your files using your source control tool directly
- All information about the project is stored in simple XML files
- Just check them in together with your design files

- Get all the benefits of a project apart from the Modified Files view
Quick recap of what we’ve seen so far…

- Simulink project: foundation for efficient team work
  - Dependency analysis
  - Shortcuts

- Integration with source control
  - Support for peer review of changes
  - Diff & merge support
Organizing files in a large project
Pick a folder structure that works for your team

- Creating a folder structure helps partition files into logical areas
- Helps keep the project maintainable
- My recommendations
  - Few/no files in the root folder
    - If something needs to be found easily, just make it a shortcut
  - Group by component, not task
- Allows for separation of tests from models/data
- Easy to ensure test code and models are not being relied upon by your design
- But…
  - If it’s not working, just change it!
Making your workflows mistake-proof

“I don’t want to ever have to write code to configure the MATLAB environment for my project”

Customer quote from 2014
Code to set the MATLAB Path

```matlab
% Examples of code to set the MATLAB path
addpath('C:\work\EXPO\gettingStarted\ReadyToRefactor\Components')
addpath([pwd '\Data'])
addpath(fullfile(pwd, 'Libraries'));
project = simulinkproject;
addpath(project.RootFolder, 'Tests');
```

(Only the code using the project root folder is recommended!)
Simulink Project: MATLAB Search Path Management

Automatically manage your MATLAB search path with your projects.

- No longer need to write MATLAB code run as startup shortcuts to manage MATLAB’s search path
- Project path folders automatically
  - Added to MATLAB search path on project start
  - Removed from the MATLAB search path on project close.
- Path is added in an OS neutral way

» sldemo_slproject_airframe

MATLAB EXPO 2016
Mistake-proofing within a project: Refactoring

- One of the models in my project has the wrong name
  - “lift_door.slx” should be “lift_doors.slx”

- What’s the impact of fixing this on
  - This Simulink file
  - Models that use it
  - MATLAB scripts & callbacks that use it
  - The source control repository
  - ?
Impact analysis supports refactoring

1. Renamed “lift_door”
2. Three other files required modification
3. Add new file to project; remove old file
4. Source control rename (add/remove)
Creating Simulink Components
Simulink Architectural Components

- Virtual subsystem
  - Graphical component – The contents are flattened to the level of the parent system during execution.

- Atomic subsystem
  - Simulink executes all blocks as a unit before executing the next block
  - Context dependent so inherits properties such as dimensions and data types from the parent model

- Model block
  - Executed as a unit
  - Context independent so doesn’t inherit properties from parent model
Component selection strategy

- Virtual and Atomic Subsystems
  - When scalability is not an issue
  - When the atomic subsystem boundary is acceptable
  - During early development of the system

- Model Reference
  - When scalability is needed
  - When hard interfaces are critical
  - To enable concurrent teamwork and unit testing

- Library Components
  - Reused utility functions
Some “Project Level” requirements for handling large models

- Many engineers must work on different aspects of the same model simultaneously
  - Combine (& recombine) different engineers’ work
  - Ensure testing is performed consistently
  - Ensure a good audit trail is kept

- Projects often have a (very) long life-span
  - Control bug-fixes and updates
  - Reuse components from Project X on Project Z?

- Plus many other company-specific requirements
Component-Based Modelling

- Picking the right size for components is key:
  - Large enough to
    - Contain enough to be useful
    - Be likely to be re-used
  - Small enough to
    - Be worked on by just one engineer at a time
    - Be likely to be checked out for editing for relatively short periods of time

- Getting this right can reduce the effort of merging the work of many engineers
  - Combining updated components is easier than merging Simulink models

- Refer back to the physical system where possible
  - A model with inputs of a measured current and an absolute force might not be right…
- This subsystem has same name as parent model
- Probably not the best name
- What is it..?
Demo

- Refactor into a new Model Reference
- Advisor helps automate/mistake proof the process
- Dependency analysis helps ensure we don’t “lose” this new component
- Refactoring support for the rename
- Find dependencies helps work out why there are some other components with poorly chosen names (like “lift_intertia”)
Enhance “New Simulink Model”…
Model Templates

Build models using design patterns that serve as starting points to solve common problems

- Use shipped templates to get started with building models or create custom templates to from a Simulink model
  - Avoid problem of corrupting original file when creating a new model
- Avoid repetitive tasks when starting out to build a new model
- Enforce a standard process for building models for the entire team or organization
Create templates relevant to the current project

- Templates registered via the MATLAB path
- Straightforward to add a template relevant to the current project
  - And remove it later when the project is closed (or path removed)
- Common uses for templates
  - Specific configuration set (e.g. fixed step, hardware set up, etc)
  - Company copyright
Managing Parametric Data
Handling model design data

How can I tackle these challenges of using the MATLAB workspace?

- Mixed with other MATLAB data
- Lack of organization: hierarchy, componentisation
- Ease of change detection
- What was the source of the data?
- Data conflict – multiple files writing to same variable name
Data Dictionary provides:

- Clarity of base workspace
- Avoids risk of running with incorrect data, or data clash
- Organised data hierarchy
- Ease of data re-use
- Integrated change management
Data Dictionary API

Automate the creation and editing of data dictionaries with MATLAB scripts

- Migrate models to use data dictionaries
- Import & export data
- Create, delete, and reassign data
- Save or discard changes to entire dictionaries or individual entries
Demo

- Start to migrate data out of base workspace to Data Dictionaries
- Model Properties > Data
Other types of component
Projects can reference other projects

Componentize large modelling projects

- Develop reusable components using projects
- Flexible referencing:
  - Relative
  - Absolute
- Extract folders to referenced projects
- Deep hierarchies are supported

» sldemo_slproject_airframe_references
Simulink Protected Model

- Sometimes you don’t want to share all your IP
  - Supplier, customer
- Sometimes it’s useful to not expose internals of models
  - Robust transfer between different groups
- Simulink Protected model can help here
Simulink Protected Model

Simulation binary

Generated code

SLXP

Encryption

Export

SLX

Model Web view

Requires Simulink® Coder™
Creating Protected Model

- **File**
  - New
  - Open...
  - Close
  - Save
  - Save As...
  - Source Control
  - Export Model to...
- **Create Protected Model: Controller**
  - **Description**
    - Create a protected model (.slxp) that allows read-only view, simulation, and code generation with optional password protection.
  - **Allow user of protected model to**
    - [ ] Open read-only view of model
    - [X] Simulate
    - [X] Generate code
  - **Generated code content type:** Obfuscated source
  - **Enter password (optional)**
Using Protected Models

- Add Model block that points to SLXP
Using labels to share & store information
Using Labels to Add Information to the project

- Done lots of work to understand what the different parts are
- Wouldn’t it be nice to record that so others don’t have to repeat this?
- What are labels?
- Apply some labels to the project
Simulink Project Labels

Easily add, modify and view labels attached to a file.

- Easily see and edit label data for all labels attached to a file.
- Use drag and drop to add labels.
- Easily switch between single-valued labels.
Note on Metadata

- What do we mean by metadata?
  - Wikipedia: “Data about data”
  - MathWorks: “Data about files”

- Data that is about the file, not (necessarily) part of it. For example:
  - FuelType = Gas, Diesel
  - ReleaseStatus = Research, Prototype, Production, Sunset
  - SecurityClassification = Unclassified, Protected, Restricted, Confidential
  - FileClassification = Design, Derived, Artefact
  - TestedWith = R2010b, R2011a, R2011b, ...
  - Coverage Metric = 84%

- Metadata can change without the file it relates to *having* to change.
“What is the impact of changing the supervisory control model?”

“What tests do I need to run to verify those changes?”

All accessible from command-line API for full automation
More options for automation
Why Automate?

- Automated Processes Get Done
  - Regularly (if needed)
  - Repeatably
  - Can be done by anyone

- Corollaries
  - Manual processes are often infrequently done
  - Can be subject to variation
  - Perhaps only one person can do them
How can Automation in Simulink Project help?

- Now anyone can rebuild the S-Functions
  - (or run the tests; generate code; publish the reports; import & validate test data; …)
  - Even at 8:34pm on a Friday night; on a testing trip; …

- Groups help provide structure
  - Group by type; or by job function (project manager group; testing group)
Automation Options in Simulink Projects

- Build-in “best practice” support
  - Project Checks
  - Growing list of our own “gotchas”
Batch Operations on Project Files

Execute project-wide operations authored in MATLAB on batches of files

- Execute a MATLAB function on some or all of the files in a project from the GUI for automation tasks
- Filtering supports selection of files
- Results are displayed in the Batch Job View

```matlab
sldemo_slproject_batchjobs
```
Some Example Batch Jobs

- Very small amount of code required
- Common patterns
  - Is this a file of type X?
  - Does this file have a label from category X with value Y?
Some Example Batch Jobs

1. Run MATLAB code checker on all my .m and .mlx files
2. Check the company copyright is added to all my files
3. Position all my Simulink models so they fit on the screen of my laptop
4. Export all my Simulink files to an earlier release of MathWorks tools
5. Generate a Bill of Materials
6. …

Email gwalker@mathworks.com if you want any of these
Simulink Project API

- Easily access information from the project
- Add, remove, inspect files & labels

If under source control,
- See source control information for files
- Get the list of modified files

New in R2016a:
Source control information
More options for sharing
Most Common Challenge in Sharing Work

“It works on my computer, just not on yours…”

Common causes:

- Incomplete set of files
- Different environment
  - (software versions, MATLAB path, …)
- Wrong data loaded
- What do I do to get started?
Sharing work outside source control

Simulink Project has built in capabilities for sharing

- GitHub
  - Collaborative sharing
  - Expect to make changes together

- Archive file
  - Fast sharing of “what I’m doing now”
  - “Delivery” workflows:
    - Send a package of work
    - Work independently
    - Receive a package of work back
How much to share?

- Typically don’t want to share all my project with a supplier or customer
- Reduce to the minimum to
  - Avoid sharing IP I want to keep in-house
  - Keep it simple

- Create “Export Profiles” to manage which files are exported from project
  - Uses project labels to set up exclusion rules
  - Set up many profiles for different workflows
    - Sharing to supplier (share only what is needed)
    - Share to customer (shield my IP)
    - Share to HIL rig (no tests, doc, requirements)
    - Etc.
Bringing it all together
We have seen

- **Simulink Projects**
  - Foundation technology for model-based design in a team
  - Allows creation & sharing of a consistent team environment
  - Supports automation of common (& uncommon) tasks

- **Links to source control**
  - Tools for efficient peer reviews of changes
  - Add source control information to your process via the project API

- **Options for components**
  - Model Reference, libraries, Data Dictionaries
  - Templates for sharing team settings
Thank you!

System Modelling, Project Management, and Simulation

System-level modelling and simulation requires flexibility in modelling style and testing frameworks. Through the example of a system-level aircraft model, discover how the latest capabilities of Simulink are used to ease modelling, testing, and project management tasks.

Technology Focus: System-level modelling, simulation

Key Products: Simulink, Stateflow
Back-up Slides
Settings for Robust File Management
Setting up Simulink Preferences for Teamwork

- Use start-up scripts to confirm the project environment
- Ensure Simulink Preferences are set appropriately
- Recommend settings shown below

1. Turn on change notification
2. Turn on auto-save
3. Turn on cross-version backup file
4. Notify when opening old models
5. Refuse to load models from a future version