Large-scale Modelling in Simulink

Andrew Bennett
andrew.bennett@mathworks.co.uk
Overview

- Introduction
- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs
- Conclusion
Overview

- Introduction
- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs
- Conclusion
Model-Based Design:
From Concept to Production

- Model multidomain systems
- Explore and optimize system behavior
- Collaborate across teams and continents

- Generate efficient code
- Explore and optimize implementation tradeoffs
- Model concurrent systems

- Automate testing
- Detect design errors
- Support certification and standards

**RESEARCH**
**REQUIREMENTS**

**DESIGN**
- Environment Models
- Physical Components
- Algorithms

**IMPLEMENTATION**
- C, C++
- VHDL, Verilog
- Structured Text
- MCU, DSP, FPGA, ASIC, PLC

**INTEGRATION**
**TEST AND VERIFICATION**
Large-scale: Too complex for one person to understand it all
Complex projects have Hundreds of files

Models, libraries, data, scripts, C/C++ code, documents, images, …
Large-scale Modelling

- Many customers’ projects are:
  - 1,000,000+ blocks
  - 100,000+ sub-systems
  - 1,000+ project specific tests
  - 100+ model files

- Many customers’ project outputs are:
  - 100,000+ functions within generated code
  - 100,000+ tests and checks
  - 10,000+ pages of generated documentation

- Embrace complexity
  - But how do we manage it?
Caveat(s)

- The best way to implement Model-Based Design varies by:
  - Project aim
  - Project size
  - Incremental versus Absolute Adoption
    - Existing organisation process, versus
    - A clean slate

- This master class focuses on Large-Scale Modelling
- If you are unsure or have questions – please just ask
Overview

- Introduction

- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs

- Conclusion
Simulink Project

- recommended way to manage projects
Variety of options

- Blank project
- From project template
- From a network directory
- From project archive (.zip)
- From project source control repository

- Available from MATLAB desktop menu: “New>”
What is a Simulink Project?

- We often talk about managing models...
- ...when we really mean managing files
  - including MATLAB and Simulink files
- Projects contain a multitude of files
- Files have variety of functions
- Files need to be organised
- Designs need to be componentized
Example – opening from Source Control
Shortcuts accessible from all views
Sorting and grouping by metadata
Simulink Project

- Recommended way to manage projects
- File changes
- Change impact
- Review changes
File dependencies
What’s the impact of a change?
What’s changed?
What’s changed in my MATLAB code?

```matlab
% Force control to return to the MATLAB.
% Only for the purpose of this example.
edit(filename)
```

![MATLAB code screenshot](image_url)
What’s changed in my data files?

File Comparison - NXTWayCtrl_Rev_3.mat vs. NXTWayCtrl.mat

Click on a column header to sort the table

<table>
<thead>
<tr>
<th>Variables in NXTWayCtrl_Rev_3.mat</th>
<th>Variables in NXTWayCtrl.mat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Size</td>
</tr>
<tr>
<td>KK</td>
<td>2x5</td>
</tr>
</tbody>
</table>
What’s changed in my Simulink models
Merge Simulink Models from Comparison Report

- Merge structural and parametric changes
- Hide non-functional changes
Overview

- Introduction

- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs

- Conclusion
Why do you need to partition a design?

- Enable efficient and robust system development
  - Reduce overall design complexity by solving smaller problems
  - Gain performance benefits that scale
  - Reuse components across multiple projects

- Facilitate collaboration
  - Partition algorithms, physical models, and tests
  - Collaborate with teams across organizational boundaries on product development
  - Manage design with source control tools

- Improve elaboration and V&V
  - Eliminate retesting for unchanged components
  - Elaborate components independently through well-defined interfaces
Recommendations for Partitioning Design

- Architecture should:
  - Flow from the requirements
    - High-level and low-level requirements
  - Be extensible
    - For future challenges
  - Address how the team will work together
    - Reduce conflicts

- But there is no silver bullet
  - Experience is key here as well
Model blocks and subsystems

- architect models – wide variety of options suitable for different situations
- Model block
  - Executed as a unit
  - Do not inherit properties, are context independent
- Atomic subsystem
  - Simulink executes all blocks within an atomic subsystem as a unit before executing the next block
  - Inherit properties from their context such as Dimension, Data Type, Sample Time, so are context dependent
- Virtual subsystem
  - Graphical component
  - Execution not affected
## Summary of Componentization Techniques

<table>
<thead>
<tr>
<th>Modeling requirement</th>
<th>Virtual SS libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development process requirements</strong></td>
<td></td>
</tr>
<tr>
<td>Functional hierarchy</td>
<td>---</td>
</tr>
<tr>
<td>Component reuse</td>
<td>---</td>
</tr>
<tr>
<td>Concurrent development</td>
<td>+++</td>
</tr>
<tr>
<td>IP protection</td>
<td>---</td>
</tr>
<tr>
<td>Incremental code gen</td>
<td>---</td>
</tr>
<tr>
<td>Unit testing</td>
<td>---</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
</tr>
<tr>
<td>Incremental loading</td>
<td>---</td>
</tr>
<tr>
<td>Simulation speed</td>
<td>+++</td>
</tr>
<tr>
<td>Memory</td>
<td>---</td>
</tr>
<tr>
<td><strong>All Simulink features supported</strong></td>
<td>✔️✔️✔️</td>
</tr>
</tbody>
</table>
# Summary of Componentization Techniques

<table>
<thead>
<tr>
<th>Modeling requirement</th>
<th>Virtual SS libraries</th>
<th>Atomic SS libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development process requirements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional hierarchy</td>
<td>---</td>
<td>✔✔✔</td>
</tr>
<tr>
<td>Component reuse</td>
<td>---</td>
<td>+++</td>
</tr>
<tr>
<td>Concurrent development</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>IP protection</td>
<td>---</td>
<td>+++</td>
</tr>
<tr>
<td>Incremental code gen</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Unit testing</td>
<td>---</td>
<td>+++</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incremental loading</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Simulation speed</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Memory</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>All Simulink features supported</strong></td>
<td>✔✔✔</td>
<td>✔✔✔</td>
</tr>
</tbody>
</table>
# Summary of Componentization Techniques

<table>
<thead>
<tr>
<th>Modeling requirement</th>
<th>Virtual SS libraries</th>
<th>Atomic SS libraries</th>
<th>Model blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development process requirements</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional hierarchy</td>
<td>---</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Component reuse</td>
<td>---</td>
<td>+++</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Concurrent development</td>
<td>+++</td>
<td>+++</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>IP protection</td>
<td>---</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Incremental code gen</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Unit testing</td>
<td>---</td>
<td>+++</td>
<td>✓✓✓</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incremental loading</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Simulation speed</td>
<td>+++</td>
<td>+++</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Memory</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
</tr>
<tr>
<td><strong>All Simulink features supported</strong></td>
<td>✓✓✓</td>
<td>✓✓✓</td>
<td>+++</td>
</tr>
</tbody>
</table>

- **✓✓✓** Very well suited
- **+++** Provides support
- **---** Limited or no support
# Summary of Componentization Techniques

<table>
<thead>
<tr>
<th>Modeling requirement</th>
<th>Virtual SS libraries</th>
<th>Atomic SS libraries</th>
<th>Model blocks</th>
<th>Atomic sub charts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development process requirements</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional hierarchy</td>
<td>---</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Component reuse</td>
<td>---</td>
<td>+++</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Concurrent development</td>
<td>+++</td>
<td>+++</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>IP protection</td>
<td>---</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Incremental code gen</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Unit testing</td>
<td>---</td>
<td>+++</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incremental loading</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Simulation speed</td>
<td>+++</td>
<td>+++</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>Memory</td>
<td>---</td>
<td>---</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
</tr>
<tr>
<td>All Simulink features supported</td>
<td>✓✓✓</td>
<td>✓✓✓</td>
<td>+++</td>
<td>+++</td>
</tr>
</tbody>
</table>
## Software Component Modularity: Semantics for sub-dividing large projects

<table>
<thead>
<tr>
<th>Feature</th>
<th>Simulink Models</th>
<th>Stateflow Charts</th>
<th>MATLAB Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component File Partitioning</td>
<td>Model Block</td>
<td>Atomic Sub-charts</td>
<td>MATLAB-file (.m)</td>
</tr>
<tr>
<td>Incremental update/ code generation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Simulink Coder Function Partitioning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Simulink Coder File Partitioning</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Variant Support</td>
<td>Yes</td>
<td>Simulink</td>
<td>Simulink</td>
</tr>
<tr>
<td>IP Protection</td>
<td>Protected</td>
<td>Simulink</td>
<td>coder.allowpcode</td>
</tr>
</tbody>
</table>
Why think about model interface definitions?

- Signal propagation is a powerful Simulink capability, but can increase component verification effort:
  - Data types & sample times can change behaviour
  - Interface expects a sample rate X but gets Y.
  - Verify N instances rather than one

- Under specified interfaces result in integration and rework issues.
  - Components can’t be just “dropped” into system model.
  - Integration cost is expensive.
Defining interfaces

- Bus editor
- Model explorer
- Block parameters dialogue
Simulink Data Objects (SDO)

- Provide precise control over attributes
  - impacting simulation and code (e.g. initial value)
  - affecting only code (e.g. storage class)

- Apply to Parameters, States, Data Stores and Signals
  - Simulink.Signal
  - Simulink.Bus
  - Simulink.Parameter
Using Simulink Data Objects

- When to use data objects
  - When precise control over data attributes is needed
  - For defining component interfaces

- Best practices for using Simulink bus signals and bus objects
  - Make buses virtual except for model reference component boundaries
  - Pass only required signals to each component
    - The interface should specify exactly what is used by that component
  - Use a rigorous naming convention for bus objects
    - Since they are stored in the base workspace

- More details, see:
  - mathworks.com/support/tech-notes/1800/1822.html
Model-reference and non-virtual buses

- Organise design into separate files
- Model componentisation
  - Algorithm
  - Parameters
- Ensures
  - Multiple people can work on project at the same time
  - Clean interfaces
  - Well structured generated code
Model Parameter Partitioning

- Parameters can be used to specify:
  - The plant (ratings/size/performance)
  - Aspects of the control system (gains/time constants/look up tables)
  - Data logging
  - Software interfaces
    - Input and Output structures
    - Runtime tunability
  - Test scenarios
Summary – Model architecture

- Model algorithm
  - Store in multiple Simulink files (*.slx)
- Model parameters or Simulink Data Objects
  - Store in MATLAB-files (*.mat or *.m)
- Use tools for investigating dependencies, modifications and their impact
Overview

- Introduction
- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs
- Conclusion
Iterate Design

- Modelling standards
- Design verification
- Desktop profiling and performance
- On-target profiling and performance
Simulink V&V: Model Advisor

- What is the Model Advisor?
  - Model Advisor is a tool that automatically checks models using predefined checks

- What are checks?
  - Checks are MATLAB functions that are code to test a specific aspect of the model.

- What can I do with the Model Advisor?
  - Run existing checks and generate reports
  - Create custom checks
What are modelling standards?

- Design requirements
- Data/file management
- Readability/appearance
- Code generation

(DO-178B, IEC 61508)
and the benefits...

Consistency

Interoperability

Matching standards

Standards compliance
Guideline checks for Industry standards

- MathWorks guidelines for **ISO 26262**
  *Road Vehicles – Functional Safety*

- MathWorks guidelines for **DO-178B/C**
  *Safety critical guidelines for Aerospace and defense*

- MathWorks guidelines for **IEC 61508**
  *Safety critical guidelines for the automotive and process automation industries*
How do I use existing modelling standards checks?

Model Advisor interface

- By Product
- By Task
  - Code Generation Efficiency
  - Frequency Response Estimation
  - Managing Data Store Memory Blocks
- Managing Library Links And Variants
- Model Referencing
- Modeling Guidelines for MISRA-C:2004
- Modeling Physical Systems
- Modeling Signals and Parameters using RTW
- Modeling Single-Precision Systems
- Modeling Standards for DO-178C/DO-331
- Modeling Standards for EN 50128
- Modeling Standards for IEC 61508
- Modeling Standards for ISO 26262
- Modeling Standards for MAAB
- Requirements Consistency Checking
- Simulation Accuracy
- Simulation Runtime Accuracy Diagnostics
- Simulink Code Inspector compatibility checks
- Simulink Model File Integrity
- Upgrading to the Current Simulink Version

Model

Run Selected Checks

Show report after run

Report
Types of Checks

Checks that verify….

1. Block configuration settings
2. Block modeling patterns
3. Data in the base workspace
4. Model configuration settings
5. Informer checks
6. Links to third party tools from the model
Advisor Highlighting
Desktop performance

- Uniform entry point for desktop simulation performance:
  >>> performanceAdvisor('modelName')
Assessing Target Performance
Concurrent Execution
Desktop concurrent execution profiling & On-target concurrent execution profiling
Overview

- Introduction

- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs

- Conclusion
MATLAB Automation

- The recommended way to manage outputs
  - Code generation
  - Test results
    - Desktop
    - SIL
    - PIL
  - Coverage metrics

- Write utilities for Simulink project – accessible by all
Overview

- Introduction
- For large-scale modelling, recommendations that work well:
  - Manage projects
  - Architect models
  - Iterate design
  - Manage outputs
- Conclusion
Where can I find more information so that I can do this on my own?

- Download Tech kit and demo…
  - MATLAB Central submission

- Read these relevant papers in the tech kit

- Watch webinar
  - https://www.mathworks.co.uk/company/events/webinars/wbnr62877.html

- Watch videos on Simulink Projects
Can I get on-site help to ramp my team up on how to use these tools?

- **Product-focused Training Services** can teach you how to:
  - Get started with Model Based Design
  - Understand options and tradeoffs for architecture and components

- **Project-focused Consulting Services** can help you with:
  - Process assessment
  - Configuration management and adaptors
  - Collaborative Model-Based Design

Work with your account manager to identify topics of interest and customize services to meet your needs.
What have we reviewed today?

For large-scale modelling, the best way to:

- Manage projects – Simulink Project
- Architect models – Model Componentization
- Iterate design – Advisors, Simulink V&V, Profiling
- Manage outputs – MATLAB Automation
  - Generated Code
  - Documentation
  - Test metrics
  - Write utility functions for project
Questions?

Andrew Bennett
andrew.bennett@mathworks.co.uk