Controlling Complexity at McLaren Automotive with Model-Based Design

Matthew Chave
Principal controls engineer - software architecture
Overview

- Why we need to make the most from our models
- Challenges to reusing models and lessons learned
- Six solutions to help you succeed in Model-Based Design
- What it looks like and the scale that has been achieved so far
- What we’re looking forward to
The Journey

- McLaren Technology Centre opens
- McLaren Production Centre opens
- MP4-12C released
- 570S released
- 540C released
- P1 GTR released
- P1 worlds first Hypercar released
- 650S released
- 720S released
- Senna released

Product lines Complexity Timescale
2019
2017
2015
2014
2013
2012
2011
2004
Ahead
Track 2025

- £1.2 billion investment in new products
- 18 new models and derivatives to be launched
- Production to reach 6,000 vehicles per year
- McLaren sportscar and supercar range to be 100 per cent hybrid
- A new Ultimate Series car as a successor to the McLaren P1™
Learning the hard way

What are the problems?

- Customization of the toolchain
- Bespoke software stacks
- Large components
- The MATLAB path, base workspace
- Standardization via documentation
- Manual integration and releases
Getting it right

Six steps to scalability
Getting it right

1. Configure and consult before customizing.
2. Abstract your control systems from the platform.
3. Encapsulate small components within Simulink Projects as micro services.
4. Use a project hierarchy to manage the path, along with data dictionaries.
5. Standardize using templates and deploy using toolboxes.
6. Automate with continuous integration
1. Configure and consult

Instead of jumping into customization

- Align your processes
- Discover existing functionality
- Contribute to future development
- Minimise to keep agile
2. Abstract your control systems from the platform

Instead of implementing bespoke software stacks

- Normal model
- AUTOSAR is abstracted
- Share models across 10 different platforms
- Reuse validated models without rework/porting
3. Encapsulate small components in Simulink Projects

Instead of managing large components

- Performance is up, prioritize reuse.
- Small, meaningful microservices

- Distribute work
- Update in isolation
- Reuse and improve
- Built in parallel
- Lean agile delivery
- Fewer resources
- Coverage
- Released package
Instead of managing the MATLAB path and base workspace

- Use projects and dictionaries in a hierarchy
- Promote common elements
- Project, Model, and Dictionary references.
- Generation and cache access/precedence
- Published data access
- Project manages environment
5. Standardize using templates and deploy using toolboxes.

Instead of relying on documentation

- Custom model advisor checks
- Project and model templates
- Report generation templates.
- Frameworks for test and automation.
- Various utilities for productivity improvements.
- Everyone can contribute

*Minimart, David Sampson
MATLAB Expo 2016
6. Automate with continuous integration
Instead of local/manual integration and releases

- Test upgrades
- Continuously check for regressions
- Gather metrics for quality
- Improve traceability
- And much more
Does it work?
Does it work?

Scalability

- 198 projects and data dictionaries referenced (88 unique).
- 89 models referenced from the largest top model.
- 4420 total blocks within the largest top model
- 7054 data dictionary items.
- 495 Jenkins jobs (1841 including branches)
- 2 weeks to upgrade from 2017b to 2019a.
- Limitations
What does it look like?
What does it look like?
Project and model layout
What are we looking forward to?

- Sub system references (2019b).
- Project and toolbox deployment capabilities (2019b).
- Observers within Simulink test to monitor signals (2019a).
- Adaptive AUTOSAR (2019a)
- Top models without sub model dictionaries (2019a)
- Improved requirements management (2019a)
- Reinforcement learning toolbox (2019a)
Steps to remember

1. Configure and consult before customizing.
2. Abstract your control systems from the platform.
3. Encapsulate small components within Simulink Projects as micro services.
4. Use a project hierarchy to manage the path, along with data dictionaries.
5. Standardize using templates and deploy using toolboxes.
6. Automate with continuous integration