Dr Calum Brown  
Principal System Engineer  
SELEX Galileo  

“From Concepts to Chips and Beyond:  
Implementing Model-Driven Engineering Workflows Within SELEX Galileo”
During the next 30 mins I shall:

• Present a view of the paradigm shift from traditional to Model Driven Engineering that SELEX Galileo have been actively implementing for around 10 years (I have personally been involved since 2003)
• Discuss some sample model-based workflows and applications, for firmware and software designs and highlight how they helped us become more efficient, faster and less prone to error
• Share some of the challenges we have encountered, and how we overcame them
• For those challenges we have not yet overcome, present our current thinking on them
• Present a list of “dos” and “don’ts” for Model Based workflows, from the SELEX Galileo perspective which you are free to do with as you wish
SELEX Capabilities

- **Avionic equipment and systems**
- **Airborne surveillance and ISTAR solutions**
- **Electronic Warfare**
- **Tactical UAS and target drones**
- **Airborne radar and advanced targeting solutions**
- **Tactical and land ISTAR and EO solutions**
- **Aircraft/Mission/Battlespace and simulation solutions**
- **Space attitude sensors, robotics, earth observ, scientific payloads**
- **Sensor components (I/R, RF, lasers, optics) based on proprietary technologies**
**Model Driven Engineering at SELEX Edinburgh**

<table>
<thead>
<tr>
<th>Year</th>
<th>Project Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>Motor control</td>
</tr>
<tr>
<td>2004</td>
<td>Comms link</td>
</tr>
<tr>
<td>2005</td>
<td>UAV Imaging Suite</td>
</tr>
<tr>
<td>2006</td>
<td>Fast Jet Imaging Suite</td>
</tr>
<tr>
<td>2007</td>
<td>High perf. mirror servo</td>
</tr>
<tr>
<td>2008</td>
<td>Fire Control radar</td>
</tr>
<tr>
<td>2009</td>
<td>Surveillance radar</td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td></td>
</tr>
</tbody>
</table>
A simplified ‘V’ Diagram

Modelling & Algorithm Development
- Analysis & Requirements Capture
- Simulation Analysis & Design
- Sub-system Model Design

Modelling Repository

Test & Qual
- System Execution and Evaluation
- System Integration and Test
- Sub-system Model Integration and Test

Firmware, Software, And Hardware Implementation
- Sub-system Model Implementation & Unit Test Framework
MDE workflows

Defining an MDE workflow:

• A workflow must be designed, just like any other part of the system
• Workflows will usually contain elements of simulation and of implementation
• Workflows which use models as their basis have been consistently proven within our organisation to be more efficient, cheaper to implement and less prone to error
• Workflows should be flexible enough to respond to both innovation and unexpected events
• There is no one ‘right’ MDE workflow, there is however a mindset that will ensure success!

“If you’re failing to plan, you’re planning to fail”
Product design – the bad old days

- Static requirements
- “Big Bang” (high risk!) integration
- Cost of fixing errors increases to right
- Incompatible with previously presented ‘V’ diagram
A new, more integrated workflow

- Teams are constantly interacting using the model as a reference
- Interactions are inherently bi-directional
- Errors and mistakes much more likely to be discovered early in the lifecycle
A simplified ‘V’ Diagram

Modelling & Algorithm Development
- Analysis & Requirements Capture
- Simulation Analysis & Design
- Sub-system Model Design

Test & Qual
- System Execution and Evaluation
- System Integration and Test
- Sub-system Model Integration and Test

Firmware, Software, and Hardware Implementation
- Sub-system Model Implementation & Unit Test Framework
When designing a workflow, the following requirements should be taken into account:

- It should be capable of being used by all stakeholders on a project to communicate on both detailed and abstract levels
- It should enable vertical integration to the greatest possible extent
- It should not require the habitual use of ‘workarounds’
- It should not require data to be manually re-entered within the flow
- It should not use proprietary or non-standard data formats
- It should not suffer from ‘versionitis’
- It should directly facilitate, or be able to be integrated with
  - requirements tools (e.g., DOORS),
  - configuration management tools (e.g., Dimensions, SVN)
  - Reporting capability (e.g., Simulink Report Generator)
  - V&V capability
    - Static & Functional Analyses
    - PIL / HIL / SIL / FIL etc.
The following slides shall demonstrate several workflows which are currently being used within SELEX Galileo:

- A highly formal Simulink workflow which is used on a number of firmware-based projects within the business
- A Simulink based workflow, which is used to provide work products into a pre-existing Software (C++) design process
- A lightweight workflow used to target low-cost DSP-enabled MCUs directly from within the Simulink environment
Example Workflow 1 – Firmware Centric

Workflow Background

• Primarily used to target DSP designs to Xilinx FPGAs by firmware engineering discipline
• Has recently been extended to including targeting capability to the new generation Xilinx Zynq architecture (which includes s/w running on dual processor ARM Cortex A9 cores)

• Tools used:
  • Matlab / Simulink
  • Xilinx System Generator
  • (and more recently, HDL coder)
  • Xilinx ISE

• Examples of applications:
  • Airborne imaging for UAV and Fast-Jet applications
  • Aircraft self-protection suite
  • Fire Control and Surveillance radar
**Sample 6-Stage Process**

1. **Golden Reference Model**
   - Matlab scripts/functions
   - Initial, high-level capture of the system performance using Matlab scripts and functions.
   - Functionality of subsequent stages is compared against this reference by means of automated testbenches.
   - Use of correct code structuring at this stage is essential.

2. **Simulink Model**
   - Floating-point Embedded Matlab

3. **(Optional) Autocode ANSI C**
   - PIL
   - Optionally auto-code and test a Processor-In-the-Loop implementation of Simulink model.
   - Allows an upper bound on execution times (as little to no optimisation will have been carried out on the generated code) and could inform the software/firmware partitioning decision.

4. **Software Design**
   - C/C++
   - DSP elements will be auto-coded from Simulink HDL Coder or a vendor-specific tool such as Xilinx System Generator.
   - Autocode (tailoring currently required) for software elements

5. **Firmware Design**
   - SysGen/HDL Coder
   - Firmware and software designs are tested in isolation of each other.
   - Traceability against requirements demonstrated via PIL and FIL using the Golden Reference model.
   - This stage currently subject of ongoing work within SELEX

6. **Full software validation**
   - Conduct a full-system hardware-in-the-loop test involving both processors and FPGAs.
   - Compare to Golden Reference Model to ensure that the required functionality has been achieved
Application – Stabilised Mirror Motor Controller

Captured data presented to model

Matlab used for frequency domain analysis and Test vector abstraction

- All design / analysis and (component) testing done in Matlab / Simulink
Application: Range Doppler Processing

Captured Data

Simulated Data

Data Captured from Xilinx FPGA based hardware

Data Captured from Simulink Simulation

- Analysed in MATLAB
- Pre-existing FPGA code co-simulated in Simulink with Modelsim
  - FPGA functional errors identified and fixed
  - “stunning” agreement between models (3rd dimension and colour maps intentionally obfuscated in pictures above)
  - Use of MathWorks tools expedited quick understanding of a previously unknown effect which had been identified during flight trials
Workflow Background

- Targeting a tightly coupled navigation solution S/W to Curtiss Wright COTS boards

- Tools used:
  - Simulink
  - Embedded Matlab
  - Stateflow
  - Simulink Coder > C++
  - Rhapsody

- Examples of applications:
  - Lightweight fire-control radar
  - High performance E-Scan fire-control radar
  - Airborne integrated imaging and SAR/GMTI radar suite
Navigations Systems Workflow

Input

Pre-existing Model

Nav Simulator

Model Library

Nav Solution

Matlab Environment

Functionally Correct Stimulus

Simulink Environment

Real Model Interfaces

Rhapsody Model

Real Software Interfaces

Target Hardware

Real Hardware Interfaces

Target Implementation

Target Implementation

Simulated Scenarios or Replay

Functional Model

Deployable Model

Deployable Model

Host C++ Instantiation

• Proof of Concept
• Performance Prediction
• H/W Selection

• Functional Test
• Peer Review
• Static Model Analysis

• Timing Estimation
• Code Gen Verification

• Timing Analysis
• System Test
• Performance Analysis

Input

Pre-existing Model

Nav Simulator

Model Library

Nav Solution

Matlab Environment

Functionally Correct Stimulus

Simulink Environment

Real Model Interfaces

Rhapsody Model

Real Software Interfaces

Target Hardware

Real Hardware Interfaces

Target Implementation

Target Implementation

Simulated Scenarios or Replay

Functional Model

Deployable Model

Deployable Model

Host C++ Instantiation

• Proof of Concept
• Performance Prediction
• H/W Selection

• Functional Test
• Peer Review
• Static Model Analysis

• Timing Estimation
• Code Gen Verification

• Timing Analysis
• System Test
• Performance Analysis
Deployable Model Architecture & Key Aims

• Model Architecture
  – Top Level Simulink referenced models
  – Data driven front-end and service oriented design of back-end
  – Principal real-time constructs provided by Rhapsody
  – Moding and scheduling performed using Stateflow

• Key Aims
  – Have a single evolving model which addresses the various project needs:
    • Performance analysis model
    • Real-time embedded implementation
    • Ground replay facility
  – Let the software designers & embedded software tools do what they’re good at
  – Let the algorithm designers & simulation/analysis/visualisation tools do the same
Example Workflow 3 – Low cost / Lightweight

Workflow Background

- Used to target DSP designs to Microchip PIC32 MCU
- Capability was developed in partnership with MathWorks
- Tools used:
  - Matlab / Simulink
  - SELEX PIC32 Custom Blockset
  - Microchip MPLAB IDE
- Examples of applications:
  - Condition monitoring
  - Servo control
  - Communications
Workflow Background

• Workflow developed in partnership with the MathWorks as part of early-phase product development work

• Initial proof-of-concept (RS-422 link and embedded PI controller) demonstrated in approximately 10 person-days

• Blockset provides high level (but very configurable) access to ‘hard’ PIC32 DSP functions

• Extremely low cost solution (couple of $’s UPC for top-end PIC32)

• Workflow subsequently developed within another company division and now on use on high profile, high volume DSP application

(this workflow is modified from www.kerhuel.eu/wiki/Simulink)
MDE workflows

An ideal MDE workflow is:

- Not the preserve of a select few engineers
- Not focused only on modelling, or only on implementation
- Never more complex than it needs to be to achieve the workflow objective
- Capable of providing Intellectual Property protection where appropriate
- Easy to use: it’s surprising how many people are looking for an excuse to switch off

Most importantly:

- A workflow is no substitute for experience and (current) domain knowledge – MDE is NOT push button in all but the most trivial cases, it is a way of thinking
Do’s and Don’ts for MDE, part 1: The Top 3 Do’s

Do:

- Tailor an approach to MDE which applies to your particular project
- Keep the models as simple as possible – complexity for its own sake causes mistakes
- Implement continuous quality management features in your models
Do’s and Don’ts for MDE, part 2: The Top 3 Don’ts

Don’t:

• Model for the sake of it – some things are, literally, not worth the effort
• Assume that a MDE workflow will simply give you answers, or executable code. Domain knowledge is still required – anyone who says otherwise is naïve at best
• Believe tool vendors claims without verifying everything via pilot projects, offline evaluations or independent corroboration.
The following people have contributed material to this presentation:

- **Head Office Engineering Function**
  - Ian Alston
  - Dez Cass

- **Software Engineering (Basildon)**
  - Dave Everett

- **Processing Techniques Group (Edinburgh)**
  - Tom Pitchforth
  - Andy Nicol

- **Sightline Control and Navigation Group (Edinburgh)**
  - Brian Donaghy
Ask the easy ones now, send the harder ones to:

calum.brown@selexgalileo.com