54,000 Employees

€49.2 billion Annual revenue*

6,726 Backlog

400 Operators

17,287 Aircraft sold

60 Produced monthly

10,561 Delivered

25,000+ Daily flights
System Development Perimeter and Interfaces

Fuel Pumps
Flame arrestor
Burst Discs

Engines

Cockpit displays

Electrical Interfaces

Fuel Gauging & Control Circuit

Gauging Fuel Probes

Compensators Densitometers

Level Sensors Temp Sensors

Refuel Panel

Refuelling

Flammability Reduction

Pipes, Couplings

Valves
Towards Full MBSE
System Requirement Authoring, Validation and Verification
Model Based Design Lifecycle

**Model Translation for Software Development**

**Desktop Simulator**
- Validation of Requirement and Verification of detailed design

**Aircraft 0**

**Fuel Virtual Integration Platform**
- to support system tests

**Matlab Simulink for Fuel Management SSRD Req and detailed design**

**Systems**

**Multi-systems**

**Architecture Design**

**Detailed Design**

**Simulation**

**Integration Testing**

**AMO & FSP for Test Rig and A/CO**

**Tank Modelling:**
- A/C - Output for Gauging & Fuel Management
- Output for Wing design (stress, load, etc.)

**Mission Distribution**

- Number of Mission in 200 nm block
- Range 1000's NM

**Desktop simulator**

**Tank Modelling:**
- Output for Gauging & Fuel Management
- Output for Wing design (stress, load, etc.)

**Mission Distribution**

- Number of Mission in 200 nm block
- Range 1000's NM

**Desktop simulator**

**Fuel Virtual Integration Platform**
- to support system tests

**AMO & FSP for Test Rig and A/CO**

**Mission Distribution**

- Number of Mission in 200 nm block
- Range 1000's NM

**Desktop simulator**

**Fuel Virtual Integration Platform**
- to support system tests

**AMO & FSP for Test Rig and A/CO**

**Mission Distribution**

- Number of Mission in 200 nm block
- Range 1000's NM

**Desktop simulator**

**Mission Distribution**

- Number of Mission in 200 nm block
- Range 1000's NM

**Desktop simulator**
Model Based Design - In Practice

Develop models to specify system functionality
- Describes behavioural & functional aspects

Details become the System (and Sub-System) Requirements
- Exercise the model to Validate Requirements

Delivered to Fuel System Supplier
- Model contains Requirements and intent
- Model execution provides system understanding
- Minimal Work to turn into Code
- Separate layer for independent validation
MBSE – Functional System Requirements

- MATLAB/Simulink/Stateflow Application
- Development of Control System Reqts
  - Normal and Failure Operating Modes
  - Crew Procedures
- Control Logic separated from Aircraft Environ
  - System Designers focus on
    - Control Functions
    - HMI
    - Robustness & Validation
  - Specialist Modellers focus on:
    - Aircraft & Environmental Simulation
    - Physics (Fuel, Thermal)
    - Auto-Test Capabilities
MBSE – Stateflow for Requirements Authoring

- Aircraft Fuel System Statecharts:
  - Linked Requirements
    - System Requirements Documents Cascade
    - Requirements Database (DOORS)
  - Separate Chart for each Major A/C Function
    - Allows for collaborative development
  - Transition booleans calculated externally
    - Input from Simulink
    - Stateflow graphical function
  - Driven behaviour of Stateflow logic separated from driving conditions
    - Allows easier readability and testing
Model Based Design - Reuse

• Integrated Desktop Simulator
  – Requirements & Environment Model
  – AutoCode using Simulink Coder
  – Optional Interfaces to Cockpit Display & Flight Warning

• OCASIME, VIP & Aircraft -1
  – Entire Software Simulation
  – Interfaces Identical to Full Flight Simulator

• Aircraft-0 (Iron Bird)
  – Cockpit Avionics & Displays
  – Integrated of Real & Simulated Systems
  – Virtual Hosting of Supplier’s Code

• Full Flight Simulator
  – Single model for all platforms
  – Training Flight and Ground Crews
Model Development Test/Verification/Validation Cycle

- Model V&V has to go through several loops
  - When the model is the requirements, the distinction between “Model Verification” and “Requirements Validation” is somewhat blurred

If a test fails, what is at fault?
- the requirement?
- the model?
- the test?
Using Simscape to Model A350 Refuel System
Component export, parameter estimation
And model simplification for Real Time performance
Use of Simscape

- Fuel Design Model Developed in Flowmaster
  - Architecture and Component Performance
  - Spec Model Only - not real-time
  - Cannot produce C-Code or embedded simulations

- Exploiting new SimHydraulics Toolbox

- Mathworks Consultancy
  - Airbus provision of core models and perf data
  - Majority of development by Mathworks
Component Development

- Mapping of Flowmaster components to Simscape/SimHydraulics equivalents
  - Most 1:1 equivalents
  - Some required customisation from base library

- Curve Fitting Toolbox
  - Fit source data to SimHydraulics block equations
  - Saved as Matlab Script for re-use
Library Construction and Parameterisation

• Component Library to customise standard Hydraulic Library Components

• System Library contains “System Level” Components
  – E.g. There are several different type of pumps

• Each System Palette contains Multiple Components
  – System Palette constructed using MATLAB scripts
    – Self Documenting
    – Re-run if design model updated
Model Simplification

- Design Model has ~900 individual Components
  - A reduction of the number of blocks by a factor of 10 can potentially yield a simulation speed improvement by a factor of 1,000.

- Reduction Strategies
  - Reduce multiple serially Connected Pipes/Bends/Losses to a single Equivalent pipe/loss combination

- Design Optimisation toolbox
  - Established Equivalent Parameters

- Reduction in the number of components
  - Pipe components reduced from 290 to 60
  - Total Components reduced from 900 to 170
  - So would expect ~120 x speed-up
Model Reduction

• During Refuel or Defuel, certain valves are not in use
  – Fluid network behind those closed valves do not contribute to pressure/flow calculations
  – Therefore can be removed

• This can be repeated for each combination of tank that needs to be studied.
  – The reduced model can be constructed automatically with MATLAB scripts that analyze network topology.
Simscape Summary of Results

• Two system-level models of the Defuel system created in SimHydraulics
  – One complete: all components required to model the system behaviour included
  – One simplified: all “isolated” components located behind closed valves removed
• Performance of the simplified model sufficient for real-time
  – Tested with Simulink Real Time on industrial PC
• Performance of the complete model not sufficient for real-time implementation, despite simplifications made.
  – Depends on the solver chosen to a large extent
  – Improves substantially from with later Simscape versions
  – Near real-time performance in exploiting Simscape local solver
• New blocks and demos added to SimHydraulics as a direct result of this work
Code Efficiencies and Performance Enhancement
Fuel Temperature Prediction Software
Fuel Temperature Prediction for Airlines
An Exercise in Code Efficiencies

- Low outside temperatures with long exposure times
- Fuel temperature may drop close to or below freezing point
  - Software written in MATLAB
  - Predict fuel temperatures given Flight Profiles & Global Air Temperatures.

Comparison of Predicted and Measured Fuel Temperature

Temperature (°C)

Distance (nm)
**Fuel Temperature Prediction for Airlines**

*An Exercise in Code Efficiencies*

**Intended Usage**

- **Flight Plan**
- Fuel Temperature Prediction
- Cold Fuel?  
  - Y
  - N
  - Fly Route

**Run-Time ~ 40 seconds (reasonable)**

**Actual Usage**

- Flight Plan
- Fuel Temperature Prediction
- Other Safety & Economic Factors
- Calculate “Best” Route
- Fly Route

**Up to 50 potential Routes**

**Run-Time ~ 50 * 40 seconds = Half Hour...BAD!**
Using MATLAB Profiler to Identify Code Efficiency Bottlenecks

- Exploit MATLAB Profiler
- Built into MATLAB
  - profile on ; run program ; profile viewer
- Creates timing profiles of every function called
- Look at the “Self Time” for time spent within function
- Profile Report highlights most expensive L.O.C.
- Iterative process to increase code efficiencies.
Code Optimisation Strategies

- Equation Vectorisation
- Loop Unrolling
- Switch…case statements
  - Reduce volume of code inside each “case”
- Use c-mex for time-critical functions
  - Check target platforms
- Minimise Globals
  - Very slow in MATLAB
- Reduce calculations inside for loops
  - Pre-calculate invariant parts of equations

Run-Time Improvements of Fuel Temperature Prediction Module

Average Simulation Time per Flight (s)

Target Time
Keeping Track of Mathworks Release Cycles

Industry Model Testing
Industry Model & Code Testing

• Aircraft Development 3-5 years, Mathworks upgrades every 6 months
  – One solution to reduce cost of (continuing) upgrade cycles
• Testing infrastructure utilising customer models and MATLAB scripts
  – Release Compatibility
  – Performance

Win-Win Situation:
• Value to Customers
  – Reduced product upgrade cost
  – Increased productivity
  – Early knowledge of regression
• Value to Mathworks
  – Compatibility testing
  – Performance testing
  – Increased tool adoption

Simple 4-step process

| Establish NDA | Package Models/Scripts | Send Package | Review and Act on Results |
Summaries and Lessons Learnt
Lessons Learnt

• Deployment of MBSE
  – As much about Competences as Technologies
  – Skillsets & Mindsets
  – Integration of Functional & Non-Functional models

• Model Build Reveals Emergent Properties
  – Validation for free
  – System difficult to model will be difficult to build/test

• Validation/Verification Testing
  – A test that is more complex than that being tested is probably wrong
  – Easy to be caught in the trap of “Test for Success”
    – Testing for intentional but not unintentional behaviour
  – Automated Test/Analysis allows regression testing
  – Formal Proof more thorough than test scripts

• System Designers Focus on Designing the System
  – The System Model is the System Requirements
  – Extra functionality required to exercise the model are not requirements
  – Need to clearly identify what are requirements and what are the extras

• Model Architecture
  – Must match System Architecture
  – Also conducive to multi-team development

• Easy for Designers can be Difficult for Simulators
  – Engineers can be very “ingenious”
  – Break downstream processes
    – Model exchange with suppliers
    – Automatic code generators
  – Require adherence to Style Guidelines and Design Patterns
Thank you