MathWorks
Aerospace and Defence Conference '08
April 29-30 | Heritage Motor Centre | Warwickshire, England

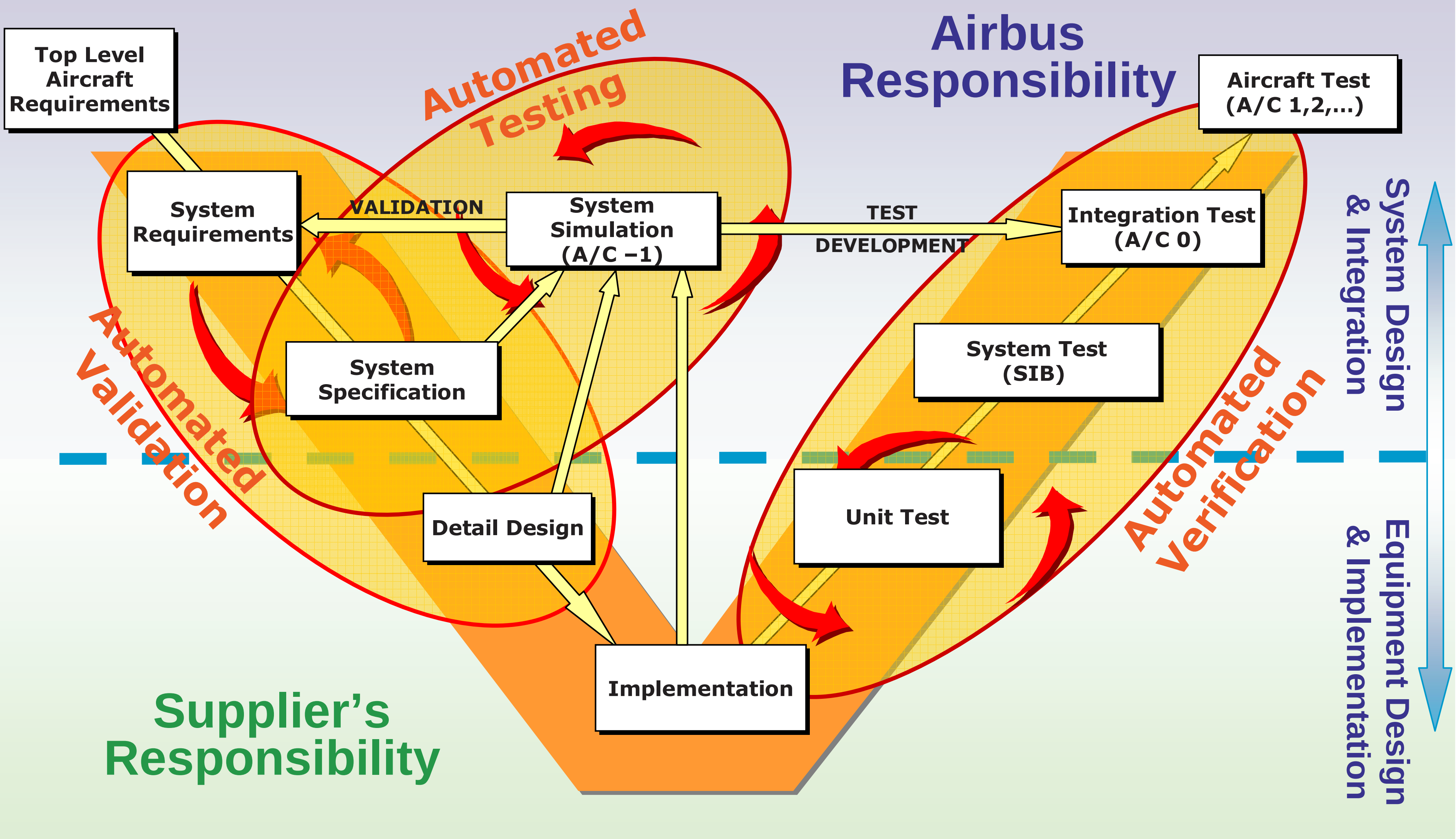
Presented by

Christopher Slack
Fuel Modelling Specialist
Airbus UK

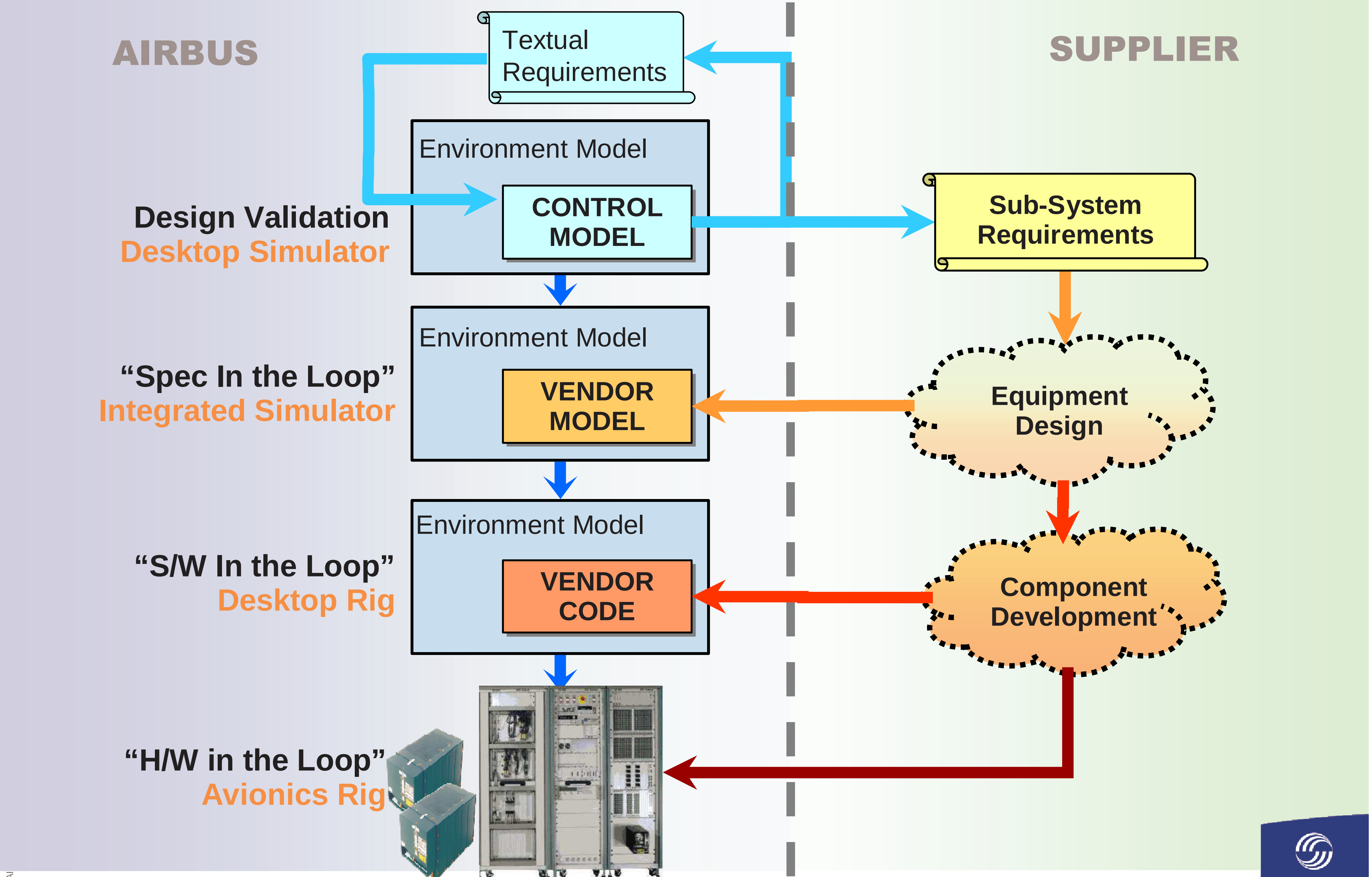
MBD within Airbus-UK Fuel Systems

Opportunities and Experiences

Systems Engineering V-Cycle

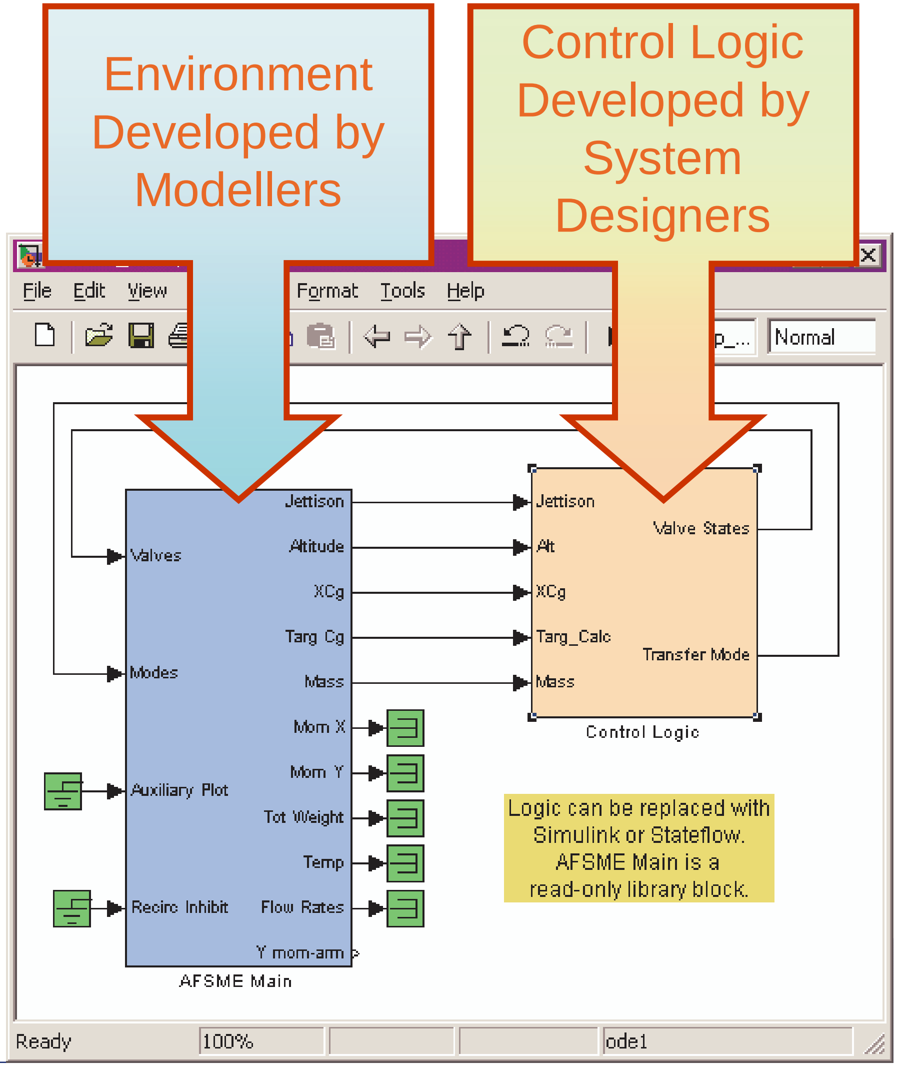


Model Based Design – Supplier Involvement



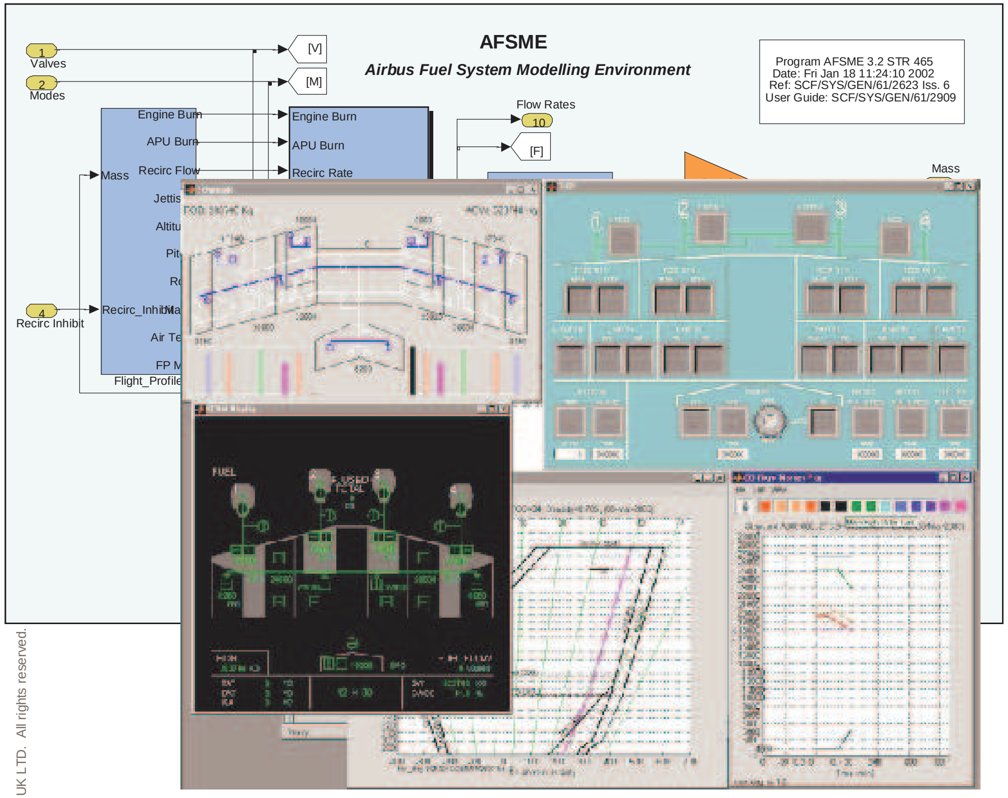
Model Based Design - In Practice

- Rapid Prototyping of Control System Requirements.
  - Normal and Failure Operating Modes
- Simulink/Stateflow Application
  - Platform Independent
  - Exploits DCT
- Control Logic separated from Aircraft Environment
  - System Designers focus on
    - Control Functions
    - HMI
    - Robustness & Validation
  - Specialist Modellers focus on:
    - Aircraft & Environ Simulation
    - GUI/Panels
    - Auto-Test Capabilities

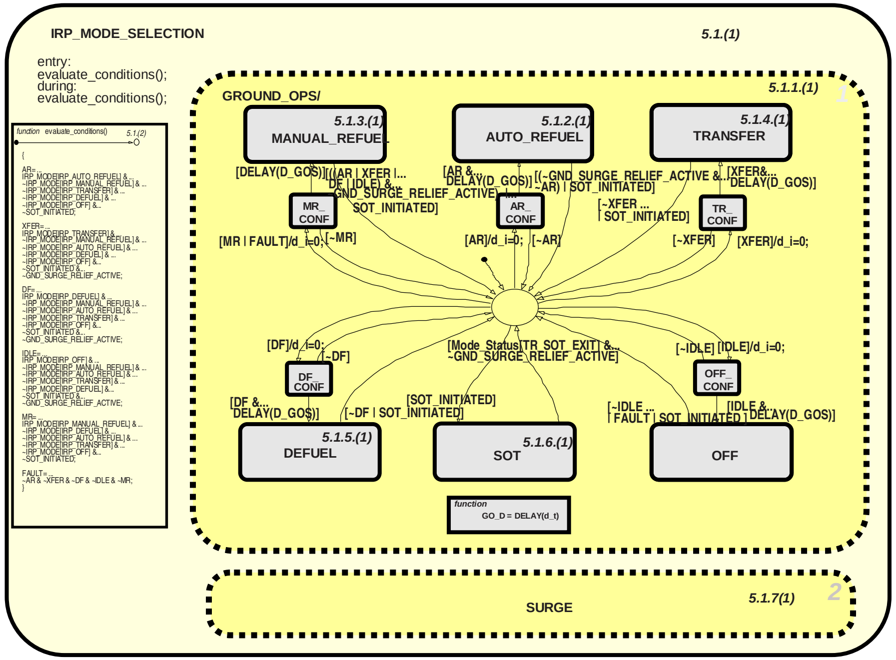


Model Based Design - In Practice

- Statecharts control behaviour
  - Easier than Enabled/Triggered Subsystems
- Enhanced Validation
  - Statechart representation can be clearer and less ambiguous
  - Increases validation confidence



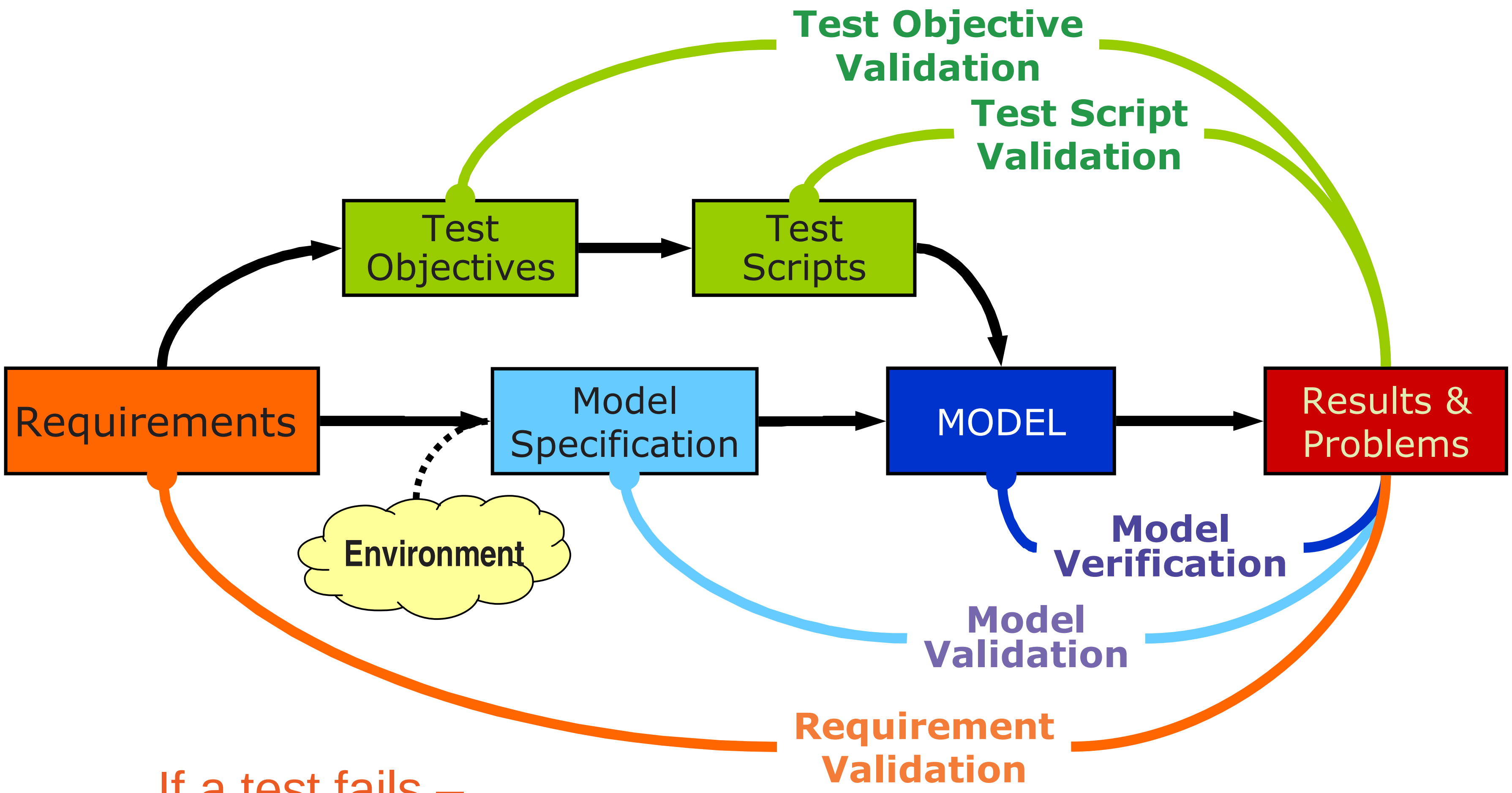
Fuel System Modelling Environment



Control Function Design

Model Development Process

When the model is the requirements, the distinction between "Model Verification" and "Requirements Validation" is somewhat blurred



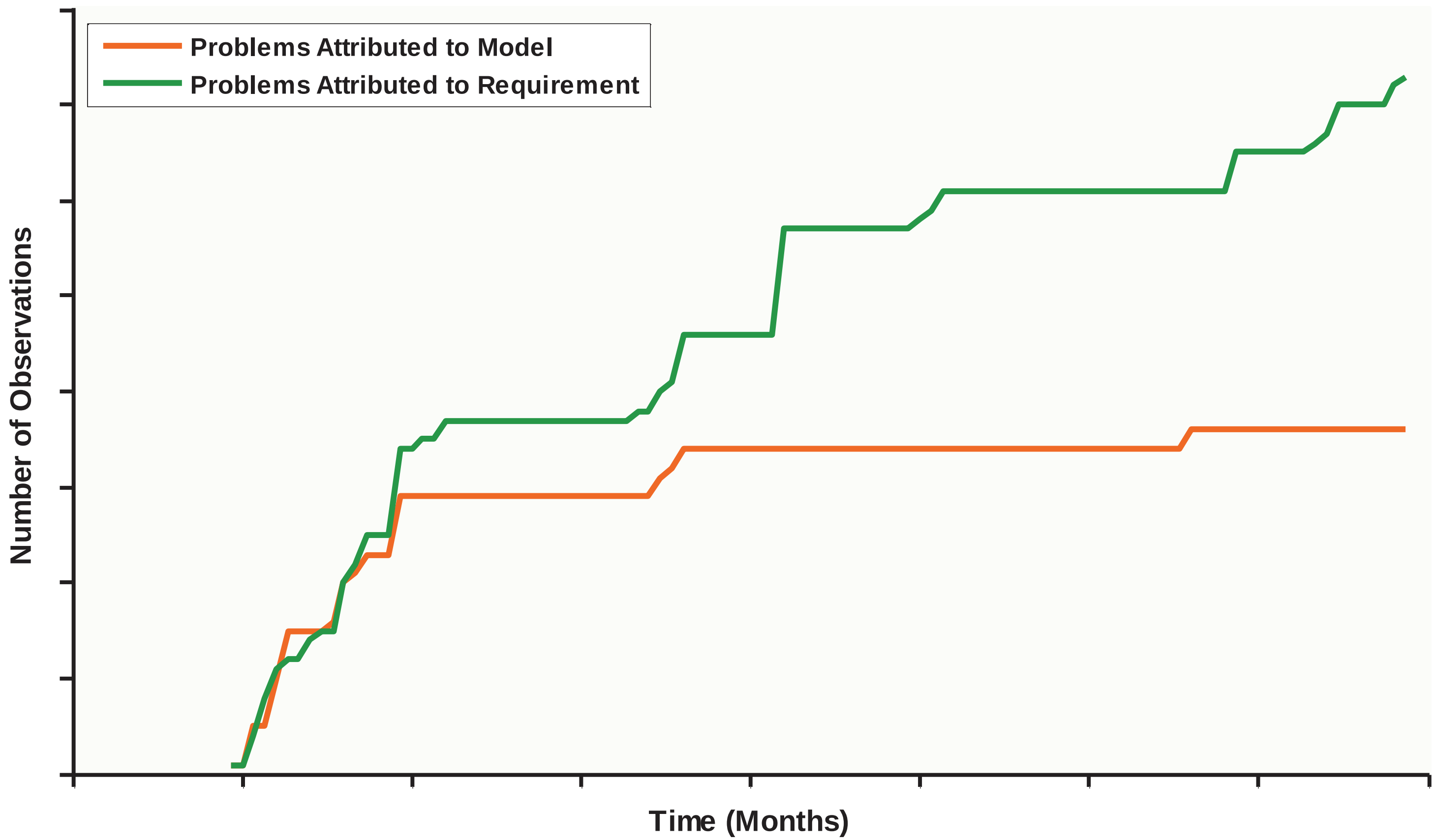
If a test fails –

is model, the requirement or the test at fault?

Model & Requirement Validation

- Typical Model Development Cycle
  - As model matures, tends towards Requirements Validation

Statistics of Model/Requirement Validation



Aviation Authorities View of MBD

Interpretation of ARP4754…

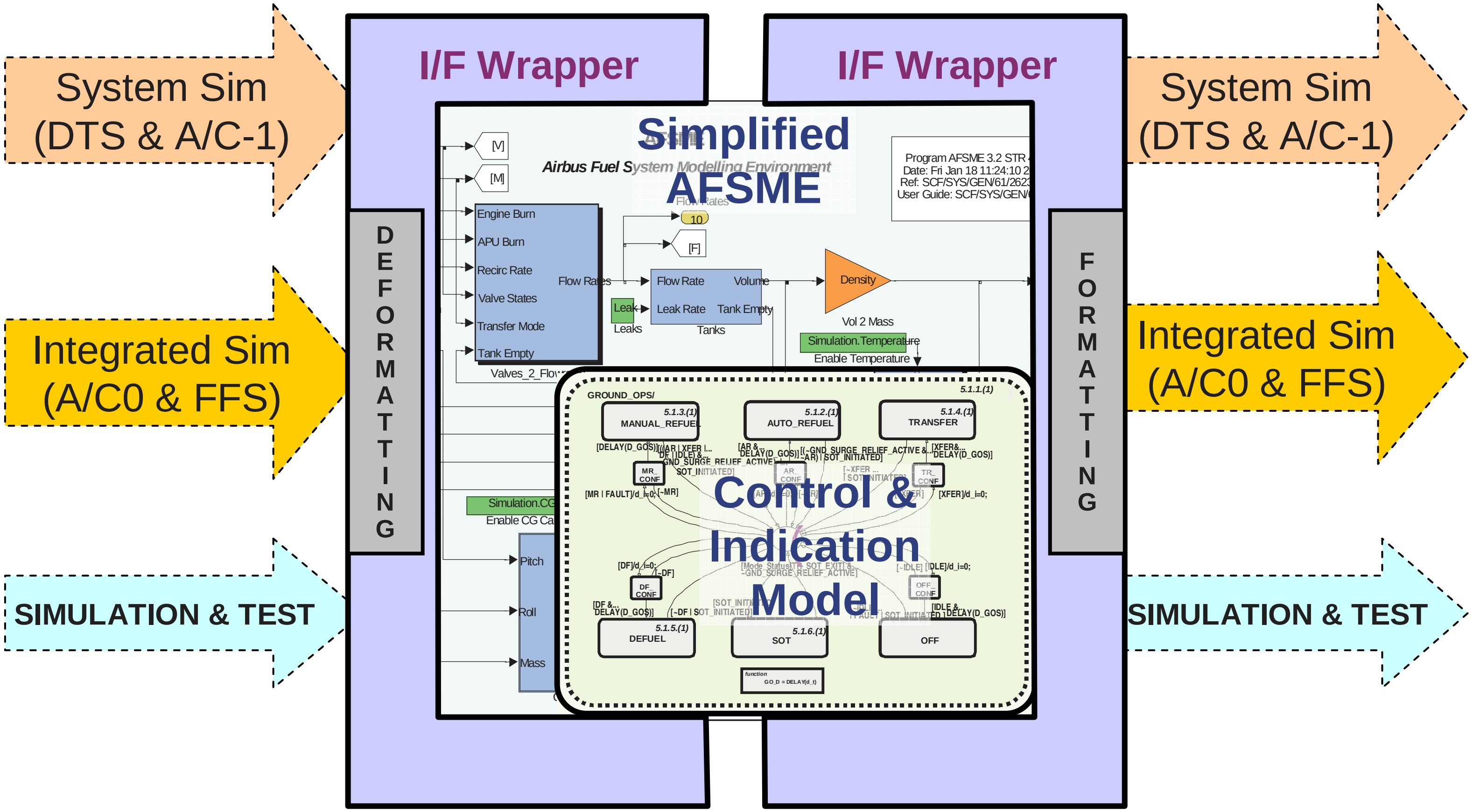
"The complexity of specification written with formalised language raises the need for higher level specification description containing all the requirements implemented in the formalised specification"

- Effectively states that a model is only an implementation of unwritten requirements.

- We need a model and textual requirements in order to sufficiently define and validate a system in compliance with ARP4754
  - Non-Functional Requirements difficult to model
    - Performance / Integrity / Reliability

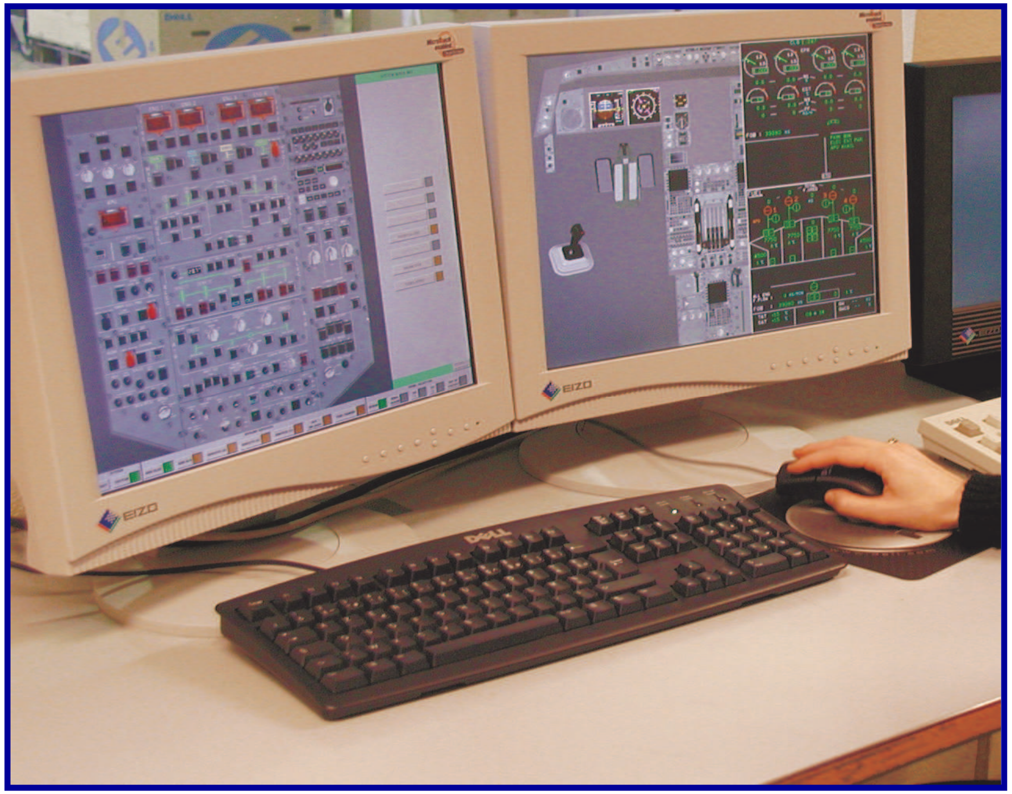
Model Re-Use – Interface Simulation

- Simulation Platforms have different interfaces
  - Pre-Formatted or Formatted ARINC429/AFDX/CANBUS
  - Includes data for simulation (e.g. Fault Injection)
- Provide Common "Core Model" with specific interfaces



Model Re-Use - Simulation Platforms

- Desktop Simulator
  - Requirements & Environment Model
  - Integrated with Flight Warning & Cockpit Display Models
  - AutoCode using SF Coder & RTW



Model Re-Use - Simulation Platforms

- Aircraft -1
  - Realistic Cockpit Mock-Up
  - Simulated Avionics
  - Interfaces Identical to Full Flight Simulator

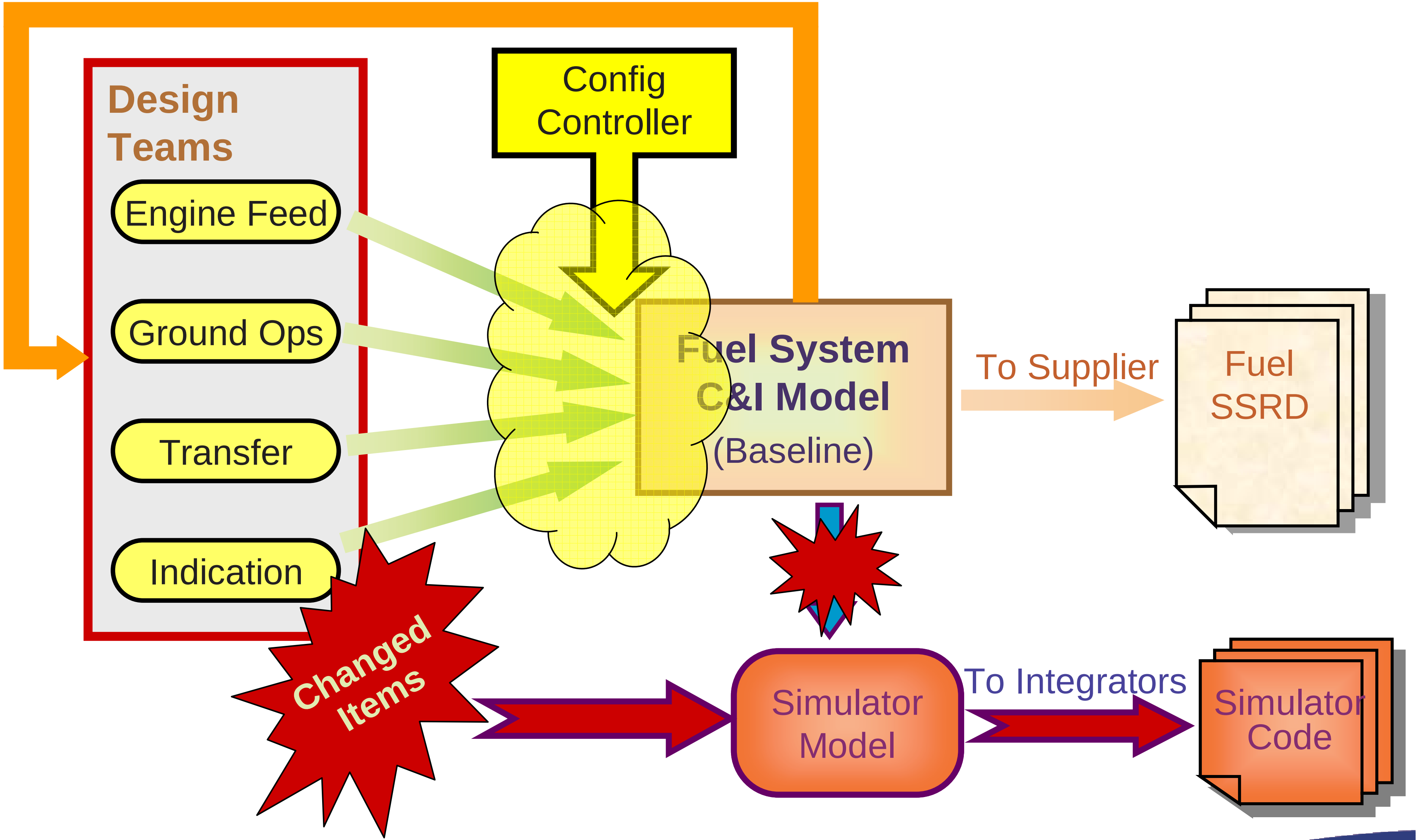
Model Re-Use - Simulation Platforms

- Aircraft Zero (Iron Bird)
  - Cockpit Avionics & Displays
  - Real or Simulated Avionics Equipment (Interchangeable)
  - Simulated Environment

Model Re-Use - Simulation Platforms

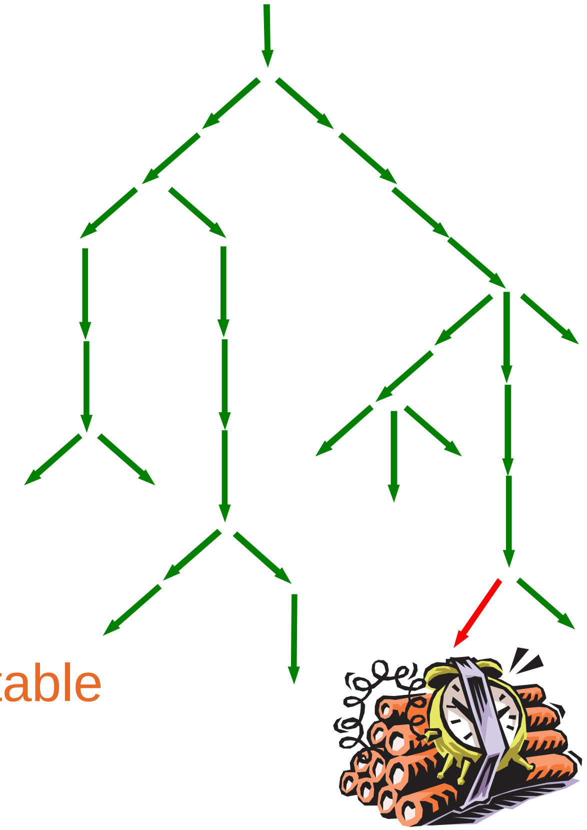
- Full Flight Simulator
  - Fully Simulated Systems and Environment
  - Single model for all platforms
    - Interfaces pre-configured for each platform

Multi Team Model Design Process



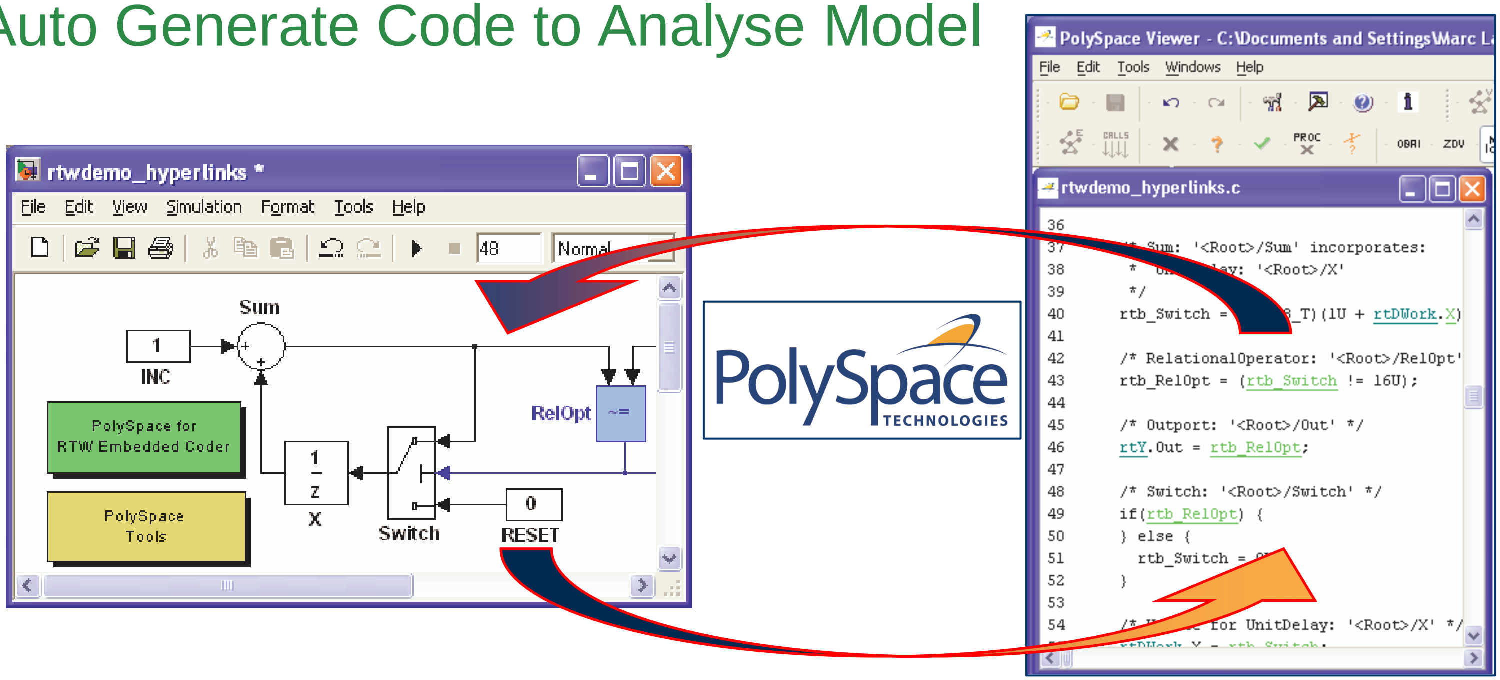
New Developments - Formal Methods

- MBD is not “Formal” in the mathematical sense
  - Once created possible to apply formal methods
- Proof Technology – Design Verifier
  - Mathematical analysis of the Model
    - without traversing all possible scenarios
    - complete in a mathematical sense
      - correct and desired behaviour
      - wrong and not desired behaviour
- Some restrictions may hinder progress
  - E.g. Non-Virtual Buses, Stateflow Structures
  - Model may need changing to make it Validatable
    - May alter the intent of the requirement



New Developments - Static Analysis

- Static Code Analysis
  - Ability to “prove” correctness of code
    - - Out of bound values
    - - Divide by zero
    - - Infinite Loops
    - - Overflow/Underflow
    - - Unreachable code/modules
    - - Square Root negative numbers
- Polyspace Model-Link
  - Auto Generate Code to Analyse Model



Lessons Learnt - Model Based Design

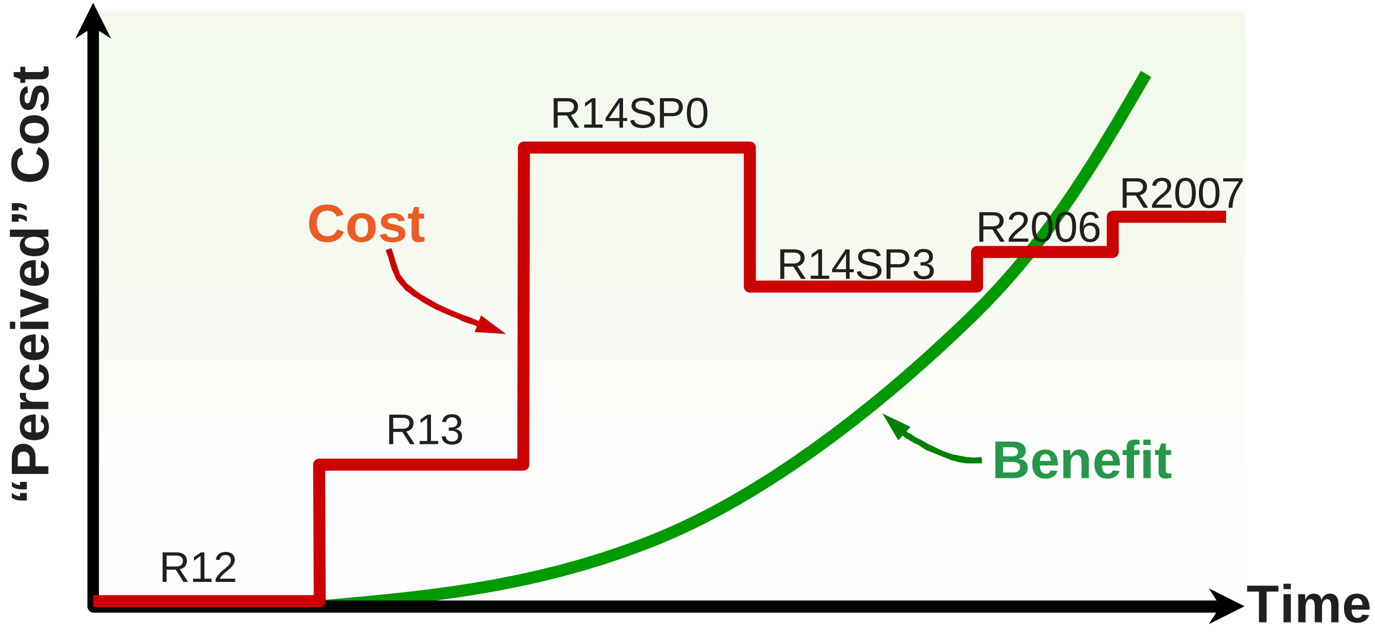
- Model build process can reveal anomalies/ambiguities
  - Validation for free
    - Identify Assumptions separately from requirements
    - Identify Executable Implementation from Requirements
- Validation Testing
  - A test that is more complex than that being tested is probably wrong
  - Easy to be caught in the trap of “Test for Success”
    - Testing for intentional, but not unintentional behaviour
    - Project managers demand simple progress metrics
- Model Architecture
  - Separate Requirements Model from Environment Model
  - Separate real interfaces from simulator

Lessons Learnt – System Design

- System Designers focus on Designing the System
  - The System Model is the System Requirements
    - Extra functionality required to exercise the model are not requirements
- Discontinuity between Design and Implementation
  - Detailed Models required for Integration Simulators
    - Required before availability of equipment
    - Need to create models of *potential* implementation
- Easy for Designers can be Difficult for Simulators
  - Matlab Function Blocks
  - M-File S-Functions
  - Test Harnesses
    - Can break the automatic code generators

Lessons Learnt - Migration

- Aircraft Life Cycle w.r.t. COTS
  - A/C measured in Decades – COTs measured in Months.
    - Tool versions *will* become obsolete – so must plan it in from start
- Cost of upgrading
  - Installation, Training, Hardware
  - Rework obsolete features, Model regression testing & re-validation
- Benefit (Cost of not upgrading)
  - Bugs
  - Utilization of new features



Any Questions?

AIRBUS

AIRBUS

AN EADS COMPANY