A Report from the Trenches

FIGHTER AIRCRAFT DEVELOPMENT WITH MODEL-BASED SYSTEM ENGINEERING

Kerstin Persson, Erik Herzog
AGENDA

The future is model based!

A fighter aircraft developed using models
SAAB – THE DOMAIN
IS IT DIFFICULT TO DEVELOP AN FIGHTER AIRCRAFT?
CHALLENGE: HANDLE MULTIPLE SYSTEM PROPERTIES

RCS
Fuel consumption
Center of Gravity
Safety
Availability
Operational cost
Payload
Service life
Flight envelope
Survivability
Fuel capacity
Weight
Environmental impact
Range
Supportability
Development Cost
Maintenance interval

COMPANY UNCLASSIFIED | NOT EXPORT CONTROLLED | NOT CLASSIFIED
Kerstin Persson, Erik Herzog | Document number | Issue 1
USE OF MODELLING & MODELS
DIFFERENT KINDS OF MODELS

- **Requirements** containing model fragments expressing required system properties.
- **Prototype model** potentially executable, meeting high level functional requirements. May be used for code generation.
- **Design model**: Architectural and functional design at a suitable level of abstraction.
- **Realisation model** potentially executable, meeting requirements, structure corresponds to the realisation.
- **Realisation** executable object code (for verification in target system)
- **Test model** realises requirements based testing. May be used to verify the prototype and realisation models as well as the realisation.
WHAT BENEFITS DID WE EXPECT?

• Early validation – ability to simulate design concepts to increase
  – Understanding feasibility
  – Acceptance for solution

• Improved communication – ability to discuss design alternatives in an objective way

• Improved accuracy – ability to determine and tune performance early in development
  – Fewer flight-test

• Improved quality – right the (almost) first time

• Improved efficiency – quicker turn-around
USE OF MODELLING TOOLS

Objective is to optimise this loop

Model Integration and System Simulation

Control

Information

Physical systems

Electronics

Structure

Catia & Co

DOORS

SysML

MMI Virtual

VAPS

Models of information objects, needs, services

Models of physical objects

Scalable and adaptive simulation and verification framework

Catia & Co

Dymola (Modelica)

Simulink

xtUML

xtUML

Class
Drop
Goto
Number

Class
Number

f(t) -> L (s)

G(s)

F(.)

Scaled and adaptive simulation and verification framework

Objective is to optimise this loop.
DOES MBSE STAND UP TO THE HYPE?

+ Ability to model complex systems
  + As long as we remain within a single modelling domain
+ Simulation allow for early feedback
  + Higher quality
+ Code generation
  + Decrease in the number of implementation errors
+ Availability of block libraries seems to facilitate success

- Limited CM/PDM capabilities
  - Except for Catia & Co
- No integration with change management
  - Truncated workflows
- UML tools for code generation do work
  - Very general language, organisations need to build their own domain support
- Modeling domain interoperability and model interoperability is a challenge

Many modelling domains still need to mature to reach its full potential
CASE STUDY

Integrating models from multiple disciplines
COMPLEX SYSTEM COVERING MANY DISCIPLINES

- Different disciplines and design methods
- Highly integrated software functions across multiple units on one platform

Fuel
Radar
Hydraulic
Navigation
Landing gear
Communication
Engine
Planning
Display
Flight control
Weapons
INTEGRATION IN TRADITIONAL DEVELOPMENT

Integration
- Case-by-case solution
- Work intensive
- High risk for low decoupling

Software platform
INTEGRATING MODELS FROM DIFFERENT DOMAINS

Who is in charge, and who must adapt?
INTRODUCING THE INTERFACE DOMAIN

- **API**: Continuous data flow
- **Events**: Interface domain
- **Rules**: Request (Req.)

Diagram:
- Interface domain
- API
- Continuous data flow
- Events
- Rules (Req.)
CASE STUDY – CONCLUSIONS

• Technical solution for an interface domain
  – Collect requirements from all applicable stakeholder domains.
  – Separate data and transformation rules.
  – Automate transformation to reduce implementation errors.

• Organisational Challenges
  – Go from flexible case-by-case implementation to a structured approach.
  – Define a solution that can be applied to all domains, don’t create a solution that fits one domain only.

• Conclusion summary
  – Technical challenges are just hard work!
  – Organisational challenges are harder to deal with.
SUMMARY
LESSONS LEARNED – IT’S ABOUT THE PEOPLE NOT THE TOOLS

Systems maturity/system lifecycle

- Adding deltas to a highly mature system
- Known architecture and constraints
- Experienced organisation – in terms of continuous development of an existing system

- No baseline system available, only some proven parts
- New architecture, constraints are not known
- Inexperienced organisation - in terms of development of a new system
QUESTIONS?