Adopting MBSE at Kongsberg Defence & Aerospace – Joint Strike Missile project

June 9th 2015

Svein-Erik Soegaard
Kongsberg Defence & Aerospace
MBSE using SysML
Adopting MBSE in the Joint Strike Missile (JSM)

Svein-Erik Soegaard
Kongsberg Defence and Aerospace
About Presenter

- Svein Erik Søgård, MSc
  - Principal Engineer, Missile Systems

- Engaged since 1995 at the Missile Division in Kongsberg Defence & Aerospace (KDA)
- Background from SW development, system integration and test in NSM (Naval Strike Missile)
- Current work (since 2010) : System Architect in JSM and responsible for adopting MBSE using SysML
Joint Strike Missile (JSM)

- Next Generation Cruise Missile from Kongsberg Defence and Aerospace
  - Based on technology from current generation Naval Strike Missile (NSM)
- To be integrated on the F-35 Joint Strike Fighter (block 4)
- Contracts with the Royal Norwegian Air Force (development) and Lockheed Martin (aircraft integration)

2009 start of development
2015 first test F-16 platform
2018 verified
2020 deployed
JSM – some key characteristics

• Many different technologies to be integrated (multi disciplinary):
  – Passive Infrared Imaging Target Seeker
  – multi-sensor Navigation System
  – Jet Engine and bank-to-turn flight control
  – In-flight radio communication (Weapon Data Link)
  – on-board Flight Route Planning based on situation awareness
  – programmable Fuze/Warhead
  – Multicore Computing Platform
  – ……

• SW intensive
  – >60% of the system requirements affects SW

• >30 years product lifecycle, mid life updates

How do we manage this?
Continuing going document based?

«Engineers hate documents»

«It takes an awful amount of time to write them»

«Hard to keep up to date, has redundant information»

«Difficult to find information»

«They are always too late»

«Not easily navigable, hard to see the «big picture»»
MBSE in the JSM Project – Initial objectives

• Establish System Architecture Model (SAM) – “The Big Picture”
  – Consistent model ensuring successful functional/logical integration of discipline components
  – fulfil system requirements by tracing
  – Navigation ONLY through diagrams to READ information
    • Both structural drill-down and between views
  – Information available outside tool -> publishing to web and some type of docs (Requirement Specs)
  – Linking related/detailed information (docs, other models etc) scoped by the nodes in the system architecture

• Life cycle focus – System Architecture understandable for maintenance, mid-life update, new product variants
JSM Modeling scope

Define customer needs

Analyse requirements

Define concept & behaviour

Implement

Integrate

Verify system

Validate system

Customer requirements, operational concepts and JSM System Requirements in DOORS

Not modeling alternative concepts and analysis in SysML, done in separate tools

Establish SAM - Focus in first stage
System Architecture Framework - Views

Specification
- WHAT shall the system do?
- Context/interfaces
- Requirements and Behavior

Design
- Functional
  - Given concept, HOW shall the system work?
  - Activity Diagrams with dataflow
- Logical
  - Given functional design, HOW shall the system be constructed?
  - Interfaces
  - Block Diagrams
  - Sequence Diagrams
- Physical
  - Mechanical design, 3D DMU in Catia®
  - HOW is the product assembled (MBOM)?
KDA Missile Reference Model

• Developed reference model – «KDA Missile Reference Model»
  – Stripped down unclassified version of JSM product

• Purposes
  – Define how we shall organize the model, define the Architecture Framework
  – Define which subset of SysML that should be used for which view
  – Cover all modeling aspects/principles in the real product
  – Basis for training and presentations
  – Basis for developing customizations (validation rules, plugins etc)

• Evolved in parallel with product development
Navigable model: Top –level index
Functional System - index

Functional System: Manage complexity, context for further functional decomposition and functional IT&V
Functional Architecture

Example from Missile Reference Model
Requirements

Example from Missile Reference Model
Example from Missile Reference Model

Contextualization:
Document describing algorithm linked to function

Navigation plugin,
Link to diagram under type

Allocation annotation by plugin (allocation derived from owner of activity)
Logical design view - Functional Structure

Show logical components in their functional context

Navigation plugin, Link to diagram for Port Type

Functional System: Manage complexity, context for further functional decomposition and functional IT&V

Example from Missile Reference Model.
Defining Behavior

State and Activity Diagrams shows external visible behavior for System and Components
Logical Architecture

- Functional Systems
  - Functional breakdown
  - Requirement development and tracing
  - Logical view

- Component Specifications
  - Information interfaces
  - Allocated Requirements
  - Allocated Functions
  - Behavior

- Logical Architecture
  - Realization oriented
  - Modules integrating components to a product
  - Modules have block diagrams for different layers: information, protocol, electrical, cabling
  - Design constraints and Physical Requirements
Missile – Logical architecture structure

Example from Missile Reference Model.
Logical Architecture – layers

Logical modules may have different layers depending on technology

Information view

Electrical Signal view

Power distribution view

Cabling view

Tracing (allocation) of connectors between different layers
Logical Architecture (Realization)

Port Allocation

Example from Missile Reference Model
From System Design to Software Design
## SW Component Specification – Flight Controller

### Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Id</th>
<th>Name</th>
<th>Text</th>
<th>Derived From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FC-1</td>
<td>Follow Route</td>
<td>The flight controller shall control the missile flight along the given route</td>
<td>Route Following</td>
</tr>
<tr>
<td>2</td>
<td>FC-1.1</td>
<td>Route Following Precision</td>
<td>The Precision of the Route following shall be better than 2 meters</td>
<td>Target Hit Precision</td>
</tr>
</tbody>
</table>

### Black – Box (Ports/Interfaces)

- `FlightController`
- `flightControl`
- `ILaunchEventReceiver`
- `launchEvents`
- `NavigationDataReceiver`
- `finControl`

### Behavior (states, activities)

- `Awaiting Route`
- `Awaiting Launch`
- `Controlling Flight`
- `missileLaunched()`

### Documents

- A3s, algorithms, models etc..

### Linked Non-SysML Information

Example from Missile Reference Model
Missile – Logical architecture structure

Example from Missile Reference Model
Software Logical Architecture

Example from Missile Reference Model
Example from Missile Reference Model

Inheriting logical black-boxes from the System Architecture Model
Software Physical Architecture

Example from Missile Reference Model
Software: Code generation from model

- UML Interfaces → Generated IDL → Generated C++ Interfaces
- UML Logical Architecture → Generated XML (CCM compliant) → Generated C++ Component Configuration
- UML Component Design → Generated C++ State Machine skeletons → Executable Code
Summary (System to SW)

- Smooth and consistent transition to SW design
  - Inheriting Component black boxes with ports/Interface
    - Chosen UML interfaces for information interfaces at sysML
  - Tracing to other model elements (requirements, state machines, activities)

- Challenges
  - Requires frequent iterations between system and SW
  - Where is the border between System and SW?
    - Ground rule: Define course grained SW components at system level, one component for a function that plays a role in its functional system
Lessons learned

Experiences and recommendations
Experiences and **Recommendations #1**

- Adopting a MBSE solution is a long journey
  - It’s about learning new methodology, new architecture framework and a new language/tool in parallel
    - In JSM it took several years to get the Architecture Framework mature
      - 10 Workshops and trainings (2-3 days), extensive mentoring
      - MBSE test bed – tested in student project
    - Many people are in general not motivated to spend much time on learning tools, MagicDraw is not a tool for everyone
      - Invest in training and mentoring, Establish core team(s) and mentor(s)

- Keeping the model update and consistent is mandatory
  - Do not put too much details into the model
  - Throw away duplicated/obsoleted information
  - (sub)Model ownership mandatory

- SysML very expressive and powerful, but complex
  - Define a language subset and a strict guideline to develop large models -> Establish a reference model expressing which subset of sysML to use for which purpose
Experiences and Recommendations #2

• Systems Engineering terminology is overloaded
  – Functional versus logical versus physical? What is a system?
  – Clear terminology is essential in communicating the model -> define!
  – Communicate terminology with examples

• Complex System Models require well managed abstractions
  – abstractions are not popular at the first glance for many
    • «abstractions hiding the details that is important»
    • «the information become fragmented by applying separate views»
  – Framework and abstractions need to be taught frequently

• It is a challenge to develop methodology and guidelines in parallel with product development
  – Start small: Establish methodology and Architecture Framework on pilot projects or small products/small parts of a product
  – Documenting existing products components – good way to learn and establish methodology & framework, «sandwich» process – meet in the middle
  – Roll out stuff that works!
The recipe for success 1)

think BIG

start SMALL

and EVOLVE

1) From presentation by Darius Silingas, No Magic
Is MBSE in JSM a success story?

We have made a good foundation

- Established SAM expressing R,F, L and P of the JSM
  - > 25 systems, > 80 components, > 4000 diagrams
  - 20-30 persons contributed to modeling the SAM (R,F,L)
- Precise specifications for component development, especially for SW
  - Smooth transition to SW component design
  - Generating code for interfaces defined in SAM
- Commitment from Management

Success so far! But we still need to improve and evolve……

- Integration, test and verification of the next JSM product increments must be successful
- «everyone» has to understand the model
- new employees should efficiently maintain the product
- model (elements) should be reused from JSM in other product variants
- the modeling culture must be sustainable
Next steps for KDA
Test and verification

- Add more requirements/capabilities for test and verification
  - «extracted» from functional design and behavior
  - Hooks for test and verification
- Define Test Specifications/Test Cases
Future state of KDA Integrated Development Platform:

- Multi-disciplinary process
- Cross-disciplinary platform
- Model-based knowledge management

Data originating from multiple sources are integrated in a joint database. Using Enovia/Catia, the data is contextualized.

Knowledge is related to the system element(s) to which it refers.

R = Requirements
F = Functional
L = Logical
P = Physical

RFLP (R F L P)
Questions?

• What is the best way to communicate the SysML models?
  • Easy navigable model?
    • Structural drill down, navigation between views
  • Generating viewpoints/documents?
    • Internal/external users
  • Other?