NDIA 2017
20th Annual Systems Engineering Conference

MBSE to Address Logical Text-Based Requirements Issues

Saulius Pavalkis, PhD, No Magic, Inc.
Saulius Pavalkis

- Chief MBSE Solutions Architect, training and consulting companies as: Ford, SGT, Abbott, Raytheon, UTAS, NYTA, Orbital ATK, SMEE, GMTO, BAH, DRAPER, LSST.
- PLM Product Integrations Manager, working with all major PLM vendors.
- Former Analyst on the MagicDraw R&D team for over 10 years.
- Major expertise area is MBSE, Requirements engineering, PLM, Traceability.
- Ph.D. from Kaunas University of Technology (KTU) in model traceability area. Former researcher at Kaunas University of Technology on multimillion projects. Master and Bachelor in telecommunication and Electronics
- Research and technical articles in model-based solutions presented at INCOSE IS, NDIA. Check modeling community blog (blog.nomagic.com) for more.
- Representative at INCOSE CAB.
Agenda

• Requirements Quality Problem
• Systems Engineering (MBSE) Solution
• Application Case with Samples from Actual Projects
• Full Picture - Framework and Method
• What next?
  o Validation and Verification
  o Documentation Generation
  o Traceability
• Conclusions
Requirements Quality Problem
Source of program failures is the result of poor requirements definition. This increase cost significantly which result in program delay or fail if program is with fix cost.

Source: Defense Acquisition University, 1993. Cost of defects increases disproportionately after the design phase.
Recent Case

- A new modern Spanish submarine S-80 Cost of 2.2 billion euros.
- Discovered to contain a serious design flaw - it is **75-100 tons overweight**.
- **Calculations mistake** in beginning by putting comma in wrong place noticed before production.
- Now have to be lengthened. **Would take two years - $9.7 million per meter**.
- **Priority project** which failure created problems for government and company.
- Engineering mistakes can happen for any system. In **document based system engineering** big problem is disconnection between design and analytical models and in general between different design artifacts.

Source: http://www.dailymail.co.uk; http://www.huffingtonpost.com
Requirements Management Theory

Requirements quality can be improved through these approaches:

- **Visualization.** Using tools that promote better understanding of the desired end-product such as visualization and simulation.
- **Consistent language.** Using simple, consistent definitions for requirements described in natural language and use the business terminology that is prevalent in the enterprise.
- **Guidelines.** Following organizational guidelines that describe the collection techniques and the types of requirements to be collected. These guidelines are then used consistently across projects.
- **Consistent use of templates.** Producing a consistent set of models and templates to document the requirements.
- **Documenting dependencies.** Documenting dependencies and interrelationships among requirements.
- **Analysis of changes.** Performing root cause analysis of changes to requirements and making corrective actions.

Source: "PMI Requirements CoP Webinar on Requirements Quality"
Systems Engineering Theory

Traditional Requirements Visualization Method

Text based requirement e.g. Excel + Drawing

• Producing text based artefacts
• Multiple unconnected sources of information: non traceable, non consistent, hard to change.
• Generally not machine readable
• Verification done by manual inspection
“Model-Based Engineering (MBE): An approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle.”

Final Report, Model-Based Engineering Subcommittee, NDIA, Feb. 2011

“Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.”

INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02, Sep 2007)

“Mechanisms to remove requirements errors up front should mitigate program risk - MBSE helps achieve this. Increasing the level and quality of systems engineering has positive effect on cost compliance, schedule compliance, and subjective quality of the programs.”

Systems Engineering (MBSE) Solution
Current Practice to Future Practice

**Today:** Standalone models related through documents

**Future:** Shared system model with multiple views, and connected to discipline models

Source: MBSE 101 by Elyse Fosse
Document Centric v Model-Based

**Document Centric**
- Producing text based artefacts
- Multiple unconnected sources of information: non traceable, non consistent, hard to change.
- Generally not machine readable
- Verification done by manual inspection

**Model-Based Centric**
- Producing system model and generating artefacts
- Aims to minimise the sources of information and manage their relationships (acknowledges we are unlikely to ever have just one repository)
- Machine readable and thus able to query
- Verification enhanced by automatic rule checking
OMG Systems Modeling Language (SysML) is a ISO Standard graphical modeling language for specification, analysis, design, verification and validation of systems.

Dedicated for modeling complex systems that may include hardware, software, information, personnel, procedures, facilities, etc.

- UML reused by SysML (UML4SysML)
- SysML’s extensions to UML
- Not required by SysML

[Diagram showing the relationship between UML and SysML]
The Four Pillars of SysML
SysML Diagram Kinds

- **Activity Diagram**
- **Sequence Diagram**
- **State Machine Diagram**
- **Use Case Diagram**
- **Behavior Diagram**
- **Requirement Diagram**
- **Structure Diagram**
  - **Block Definition Diagram**
  - **Internal Block Diagram**
  - **Package Diagram**
  - **Parametric Diagram**

- The same as in UML 2
- Modified from UML 2
- New diagram type
Requirement Relations in SysML
Application Case with Samples from Actual Projects
## Stakeholder Needs Import

### Stakeholder Needs in

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>STN1</td>
<td>STN1 Power source</td>
<td>Headphones shall charge using USB 2.0 as charging source</td>
</tr>
<tr>
<td>STN2</td>
<td>STN2 Pairing</td>
<td>Headphones pairing shall be intuitive and straightforward</td>
</tr>
<tr>
<td>STN3</td>
<td>STN3 Battery capacity</td>
<td>Battery shall last long enough to see full movie.</td>
</tr>
<tr>
<td>STN4</td>
<td>STN4 Charging time</td>
<td>Charging time shall be short</td>
</tr>
<tr>
<td>STN5</td>
<td>STN5 Dynamic range</td>
<td>High quality dynamic range</td>
</tr>
<tr>
<td>STN6</td>
<td>STN6 Solid material</td>
<td>Headphones shall look and feel as very solid.</td>
</tr>
<tr>
<td>STN7</td>
<td>STN7 Comfortable</td>
<td>User shall not feel weight or pressure</td>
</tr>
<tr>
<td>STN9</td>
<td>STN9 Audio source</td>
<td>Headphones shall play sound from Bluetooth and Line-in sources</td>
</tr>
</tbody>
</table>

... and in Requirements Table in
Interface Requirements Formalization

- User
- Audio, Status
- Controls
- Line In Audio
- USB Power
- Headphones
- USB Charging Source
- Bluetooth Audio
- Line In Audio Device
- Bluetooth Audio Device

Extended Requirement:
- Audio source
  - Id = "STN9"
  - Text = "Headphones shall play sound from Bluetooth and Line-in sources"

Extended Requirement:
- Power source
  - Id = "STN1"
  - Text = "Headphones shall charge using USB 2.0 as charging source"
# Functional Requirements Formalization

**User**

- Presses ON button for 2 seconds

**Headphones**

- Announce "Pairing"
- Enter pairing mode.
- [else]
  - [New Device Detected]
  - enter passcode "0000" on Bluetooth source.
- [else]
  - [Registered Device Detected]
  - Pair with Registered Device.
  - Pair with new device.

**Extended Requirement**

*Pairing*

- Id = "STN2"
- Text = "Headphones pairing shall be intuitive and straightforward"

**2 Bluetooth Audio Source**

- [refine]

**Extended Requirement**

*Audio source*

- Id = "STN9"
- Text = "headphones shall play sound from Bluetooth and Line-in sources"

**3 Line In Audio Source**
System Characteristics Formalization

Measurements of Efficiency (MoE) define validation criteria for stakeholder requirements.
Leveraging Model Traceability

SysML enables to link elements with relations to facilitate traceability.
MBSE at Bombardier Transportation
Change Analysis on Different System Hierarchy Levels
Interface System Requirements

- **Audio source**
  - Id = "STN9"
  - source = "marketing"
  - Text = "headphones shall play sound from bluetooth and Line-in sources"

- **BT protocol**
  - Id = "BT_17"
  - Text = "The BT radio shall support 802.15.1 protocol."

- **802.15.1 IF**
  - Source In : ~802.15.1 IF
  - BT Audio Source : BT Audio Source
  - Source Out : 802.15.1 IF
  - Line In : ~Analog Audio IF
  - Line Out : Analog Audio IF
Requirements

Example from Missile Reference Model
Full Picture - Framework and Method
You always end-up using an architecture framework whether you want one or not, or whether you intend to or not
An Approach: MBSE for RM

Stakeholder Needs

- OSLC
- import/export
- refer

? via ReqIF

import/export

derive

System Requirements

Component Requirements
<table>
<thead>
<tr>
<th>Layer of Abstraction</th>
<th>Requirement</th>
<th>Behavior</th>
<th>Structure</th>
<th>Parametrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>Stakeholder Needs</td>
<td>Use Cases</td>
<td>System Context</td>
<td>Measurements of Effectiveness (MoEs)</td>
</tr>
<tr>
<td>Problem</td>
<td>Goals &amp; Objectives</td>
<td>Functional Analysis</td>
<td>Logical Subsystems</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>Component Requirements</td>
<td>Component Behavior</td>
<td>Component Assembly</td>
<td>Component Parameters</td>
</tr>
</tbody>
</table>
Requirements Traceability

Layer of Abstraction

<table>
<thead>
<tr>
<th>Black Box</th>
<th>White Box</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>Stakeholder Needs</td>
<td>Component Requirements</td>
</tr>
<tr>
<td>Requirements</td>
<td>Behavior</td>
<td>Structure</td>
</tr>
<tr>
<td>C1</td>
<td>Refine</td>
<td>Refine</td>
</tr>
<tr>
<td>P1</td>
<td>Same Pattern</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>Use Cases</td>
<td>System Context</td>
</tr>
<tr>
<td>P3</td>
<td>Functional Analysis</td>
<td>Logical Subsystems Communication</td>
</tr>
<tr>
<td>S1</td>
<td>Component Behavior</td>
<td>Component Assembly</td>
</tr>
<tr>
<td>S2</td>
<td>Component Assembly</td>
<td>Component Parameters</td>
</tr>
</tbody>
</table>

C4-P4 Measurements of Effectiveness

No Magic
Validation and Verification
GUI Mockups for System Validation
Requirements Verification: Vocabulary

Vocabulary maps natural language phrases to mathematical expressions for automated requirements formalization.

Stopping distance requirements visualization

Requirement texts are automatically parsed to mathematical expressions.
Step 1: Formalize Requirements
Verification reports

**Vehicle**

- **stoppingDistance**: ft

**Brake**

- **pad**: Pad
- **heat**: kW

**Brake Pad Life**

- **life**: mi

**StopDistance Analysis**

- **id**: "1"
- **Test**: "Four braking wheels shall be capable of stopping the vehicle from 60 miles per hour in less than 180 feet. upperBound = 180.0"

**Brake Heating**

- **id**: "2"
- **Test**: "Braking at 60 miles per hour shall not generate more than 53 kW of heat at each wheel. upperBound = 53.0"

**Vehicle UT@255e3e3e3**

- **numberOfWheels**: integer [1]
- **weight**: integer [3209.0800]

**Run Test**

- **Instance Table**

<table>
<thead>
<tr>
<th>Test</th>
<th>Testcase1 verdict</th>
<th>Testcase2 verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>2</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>3</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td>4</td>
<td>Test at 2014.05.03</td>
<td>fail</td>
</tr>
<tr>
<td>5</td>
<td>Test at 2014.05.03</td>
<td>fail</td>
</tr>
<tr>
<td>6</td>
<td>Test at 2014.05.03</td>
<td>pass</td>
</tr>
</tbody>
</table>
External Analytical Model Integration for Requirements Verification

- Math engines
  - Matlab/Simulink
  - Mathematica
  - Maple
  - Open Modelica

- Scripting
  - Javascript
  - Python
  - Groovy
  - Ruby
Validation and Verification (more)

- OMG standard UML Testing Profile (UTP) based test case definition
- Test cases recording based on system design
- Test cases execution
Template-based Generator

You can generate an HTML, XML or any other simple text report from the model data.

1 Concept

11 Stakeholder Needs

STN1 Power source
Headphones shall charge using USB 2.0 as charging source

Source:
Marketing

Derived:
BT_7 USB 2.0 power source
BT_15 One USB 2.0 IF

Refined By:
Charge
NA

Traced From:
BT_15 One USB 2.0 IF
BT_7 USB 2.0 power source

STN2 Pairing
Headphones pairing shall be intuitive and straightforward

Source:
Usability team

Derived:
BT_18 Reconnect

Refined By:
Pair With Device
Online Documentation And Review

Remove connector

Add interface
Traceability
## Requirements Analysis: Coverage

<table>
<thead>
<tr>
<th>#</th>
<th>Date</th>
<th>Requirements Count</th>
<th>Blocks Count</th>
<th>Requirements Covered By Blocks Count</th>
<th>Requirements Covered By Blocks Percentage</th>
<th>Requirements Covered By Blocks Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2014.03.01 16.00</td>
<td>18</td>
<td>4</td>
<td>2</td>
<td>11.11</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2014.03.01 16.00</td>
<td>18</td>
<td>10</td>
<td>4</td>
<td>22.22</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>2014.05.01 16.00</td>
<td>18</td>
<td>12</td>
<td>6</td>
<td>33.33</td>
<td>1.50</td>
</tr>
<tr>
<td>4</td>
<td>2014.05.01 16.00</td>
<td>18</td>
<td>8</td>
<td>8</td>
<td>44.44</td>
<td>1.33</td>
</tr>
</tbody>
</table>

**Requirements Coverage Analysis**

![Bar chart showing requirements coverage analysis for different dates and categories]
Requirements Analysis: Change Impact
Tracking Changes

Integrated Systems Engineering

Quick Diff of Requirement Braking Distance

Review differences

Review differences between the two versions of the element. Modified property values are marked in blue color, newly created property values are marked in green color, and deleted property values are marked in red color.

Properties

<table>
<thead>
<tr>
<th>Properties</th>
<th>#4</th>
<th>#5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement</td>
<td>Dry Pavement Braking Distance at 60 MPH: 110 ft.</td>
<td>Dry Pavement Braking Distance at 60 MPH: 90 ft.</td>
</tr>
<tr>
<td>Text</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Text Diff

Dry Pavement Braking Distance at 60 MPH: **110-90 ft.**
Conclusions

• Requirements quality is big problem which causes projects to delay and be out of budget. This means a lot of money in case of complex SE projects.
• Traditional methods for requirements quality does not address semantic of system under design.
• In document based system engineering big problem is disconnection between design and analytical models and in general between different design artifacts.
• MBSE provides: methods, tools, and languages for requirements formalization and significant quality increase.
• We presented MBSE with SysML based approach (MagicGrid) for requirements formalization. It:
  o Using SysML as a language provides rich means to address requirements quality issue by formalizing them with system model. Descriptive system model represents system from behavioral, structural, and analytical viewpoints giving full understanding of requirements at any level.
  o Ensures requirements traceability
  o Supports automated requirements analysis
Innovation drives success!

Håkan Forss @hakanforss http://hakanforss.wordpress.com
This illustration is inspired by and in part derived from the work by Scott Simmerman, “The Square Wheels Guy” http://www.performancemanagementcompany.com/