Simplifying Requirements Based Verification with Model-Based Design

Fraser Macmillen
Requirements & Model-Based Design
Verification & Validation Products

86 slides of new features in 2019...
DOORS 9 Baseline Navigation

Simulink Requirements is aware of baseline in DOORS 9

Navigate

Specified Baseline

1.0

1.1

1.2
Unlock and Edit Referenced Requirements

Augment referenced requirements with additional details

- Add additional custom attributes without modifying imported content
  - Update restores data from external source and preserves additional content

- Export to ReqIF for roundtrip workflow of local edits with third party tool

See: Roundtrip Workflows with ReqIF Files
Test Specification Report

• Generate report in PDF, ZIP or DOCX format consisting of test specifications (models, inputs, baseline, assessments etc)

• Customization through templates for report formatting

• Custom sections to add extra content that are user specific

<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Example TestFile</td>
</tr>
<tr>
<td>1.1. Example TestSuite</td>
</tr>
<tr>
<td>1.1.1. Example BaselineTest</td>
</tr>
<tr>
<td>1.1.2. Example EquivalenceTest</td>
</tr>
<tr>
<td>1.1.3. Example SimulationTest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Baseline Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Name</td>
</tr>
<tr>
<td>baseCap.mat</td>
</tr>
<tr>
<td>• Input Conversion Subsystem:2</td>
</tr>
<tr>
<td>• vehiclespeed</td>
</tr>
</tbody>
</table>
But what are my requirements for this talk?

1. I shall expand awareness and capability in the use of our tools that support verification of requirements

2. I shall not repeat content from previous EXPOs

3. I shall deliver content appropriate to a masterclass

4. I shall make it interesting!
Minimising
the less we have to deal with the simpler it is

Insight
insight leads to understanding and makes our work simpler

Automation
to speed up the process and avoid errors makes our work simpler
Simplifying Requirements Based Verification with Model-Based Design

Minimising
the less we have to deal with the simpler it is

Insight
insight leads to understanding and makes our work simpler

Automation
to speed up the process and avoid errors makes our work simpler
Minimising – being at the right level

Requirements at the right level

Verification at the right level

Tracing to the right level
Minimising – handling multiple levels

Example

Control system requirements specify the need for filtering of certain signals

The design uses a reusable custom filter to implement the requirement

There are lower level requirements for the filter behaviour itself

Let’s start by creating some links to an instance of the custom filter in the design
Minimising – handling multiple levels

When linking a requirement to a Simulink block…

- Can link from either end
- The Simulink block is always the source
- The requirement is always the destination
- The link is saved in the file associated with the source: i.e. [modelFileName].slmx
Minimising – handling multiple levels

Linking Between Requirements At Different Levels

The prime purpose of traceability is to infer what is the origin/parent/source of an object.

i.e. a link is from child to parent, from source to destination

i.e. the source is the lower level requirement
the destination is the upper level requirement

The link is saved in the file associated with the source

So: click on source (lower-level requirement) first and create link from the parent requirement…
Minimising
Minimising - Links
Minimising

Using re-usable components can help
- minimise requirements
- minimise requirement links
Simplifying Requirements Based Verification with Model-Based Design

Minimising

the less we have to deal with the simpler it is

Insight

insight leads to understanding and makes our work simpler

Automation

to speed up the process and avoid errors makes our work simpler
Insight - Observers
Observers: Separate verification logic from design

- Access nested signals
- Without modifying interface
Insight – Logical & Temporal Assessments
Translate textual requirements into unambiguous Assessments

- Compose assessments using form based editor
- View assessments as English-like sentence
- Review and debug temporal assessment results
- Link to requirements
Temporal Assessments

Revisions:

- 2.2.5

Custom ID: SKD-CONTROL-84

Summary: The turbine shall enter the Generating mode \([\text{GeneratorTrip} \text{FALSE}]\) if the generator speed is greater than 1200 rpm and less than 2200 rpm.

Keywords:
- No

Revision information:

Show in document Unlock

Custom Attributes

Links

- Implemented by:
  - [GeneratorSpeed -> GeneratorSpeedCutIn]
  - [WindSpeed -> WindSpeedCutOut] [WindSpeedCutInLower] [WindSpeed -> WindSpeedCutOut] [GeneratorSpeed] [GeneratorTrip]

- Verified by:
  - [GeneratingMode]

Comments

Add Comment
No comments
Insight - What if verification is by analysis, not simulation?
What if verification is by analysis, not simulation?
Minimising
the less we have to deal with the simpler it is

Insight
insight leads to understanding and makes our work simpler

Automation
to speed up the process and avoid errors makes our work simpler
Simplification – Automation

Examples:

- Checking parameter values against requirements

- Continuous Integration
## Simulink Requirements — Functions

### Requirements Definition

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sreq.ReqSet</td>
<td>Work with Requirements sets</td>
</tr>
<tr>
<td>sreq.Reference</td>
<td>Work with external requirement proxy objects</td>
</tr>
<tr>
<td>sreq.Requirement</td>
<td>Work with Requirement objects</td>
</tr>
<tr>
<td>sreq.clear</td>
<td>Clear requirements and links from memory</td>
</tr>
<tr>
<td>sreq.convertAnnot</td>
<td>Convert annotations to requirement objects</td>
</tr>
<tr>
<td>sreq.editor</td>
<td>Open Requirements Editor</td>
</tr>
<tr>
<td>sreq.find</td>
<td>Find requirement, reference, and link set artifacts</td>
</tr>
<tr>
<td>sreq.import</td>
<td>Import requirements from external documents</td>
</tr>
<tr>
<td>sreq.load</td>
<td>Load requirements/link set</td>
</tr>
<tr>
<td>sreq.new</td>
<td>Create requirements set</td>
</tr>
<tr>
<td>sreq.open</td>
<td>Open requirements set</td>
</tr>
<tr>
<td>sreq.resetViewSettings</td>
<td>Reset saved view settings</td>
</tr>
<tr>
<td>sreq.importViewSettings</td>
<td>Import view settings</td>
</tr>
<tr>
<td>sreq.exportViewSettings</td>
<td>Export view settings</td>
</tr>
</tbody>
</table>

### Requirements Traceability

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sreq.LinkSet</td>
<td>Work with link sets</td>
</tr>
<tr>
<td>sreq.Link</td>
<td>Work with link objects</td>
</tr>
<tr>
<td>sreq.clear</td>
<td>Clear requirements and links from memory</td>
</tr>
<tr>
<td>sreq.createLink</td>
<td>Create traceable links</td>
</tr>
<tr>
<td>sreq.find</td>
<td>Find requirement, reference, and link set artifacts</td>
</tr>
<tr>
<td>sreq.load</td>
<td>Load requirements/link set</td>
</tr>
<tr>
<td>sreq.cmConfigureVersion</td>
<td>Set version of linked requirements documents</td>
</tr>
<tr>
<td>sreq.cmGetVersion</td>
<td>Get configured version of linked requirements documents</td>
</tr>
</tbody>
</table>
Simplification – Automation

Programmatic Interface:
- Find and interrogate requirements and links
- Use to create custom artefacts or utilities

Example…
The rotor cut-in wind speed shall be [windSpeedCutIn]
Continuous Integration (CI) originated as a software development process in which developers integrate their code into a shared repository on a regular basis.

Each commit into a repository is verified by an automated build and test.

These tests may be a pre-cursor to pushing the changes to a main branch.

Continuous Integration can be applied to Mode-Based Design workflows.
Simplification – Automation – Continuous Integration (CI)

How quickly can one set up a continuous integration project to run Simulink Tests against requirements?

How many lines of MATLAB code are required?
Simplification – Automation - Continuous Integration (CI)

How quickly can one set up a continuous integration project to run Simulink Tests against requirements?

< 5 minutes!

How many lines of MATLAB code are required?

- None!
Simplification – Automation - Continuous Integration (CI)

Best practices can minimise the work required

Model-Based verification tools continue to develop to provide insight
- more tool integration
- more ways of accessing information you need intuitively & unobtrusively
- more control over granularity

Automation can be quick to set up, and offers significant benefits
Learn More

Key products covered in this presentation:

- Simulink Requirements
- Simulink Test
- Simulink Coverage
- System Composer

Learn more at Verification, Validation and Test Solution Page:
mathworks.com/solutions/verification-validation.html

fmacmill@mathworks.com
MATLAB EXPO 2019