Model-Based Structured Requirements in SysML

Daniel R. Herber, Jayesh B. Narsinghani, Kamran Eftekhari-Shahroudi

Colorado State University – Department of Systems Engineering
daniel.herber@colostate.edu

SYSCON 2022
April 25-28, 2022

Outline

1. Introduction

2. Model-Based Structured Requirements

3. Examples

4. Summary and Future Work

1

Introduction

→ Introduction (1)

- Architecture-centric practices are gaining widespread acceptance in systems engineering (SE)
- Central to the rules governing a system are the requirements placed on it, often by various stakeholders
- These requirements help guide the system development process of a complex entity
- However, requirements do not specify how the system will meet these needs
  - It is up to the solution and project team to decide how the requirements shall be fulfilled
- Therefore, it is critical to have a well-defined, complete, and adaptable representation of the requirements in the system model

→ Introduction (2)

- There are a variety of approaches for developing and managing requirements[1]
- It has been established that a well-defined requirement shall possess the following general characteristics[2]:
  - *Necessary*, *Appropriate*, *Unambiguous*, *Complete*, *Singular*, *Feasible*, *Verifiable*, *Correct*, and *Conforming*
- Still a common problem of tremendous effort being spent understanding the requirements and multiple iterations because these guidelines and rules must be adhered to manually by engineers
- Furthermore, many approaches do not fully integrate with the system model
- These issues lead to increased costs and risks during development due to the poor quality of requirements defined during the early stages
- Here, an approach for supporting rigorous model-based requirements is proposed using the existing concepts of structured requirements and SysML

[1] Pohl 2010; *ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering* 2018; Ryan et al. 2019; R. Carson 2021; *IEEE Guide for Developing System Requirements Specifications* 1998 [2] *ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering* 2018; Ryan et al. 2019

②

Model-Based Structured Requirements

→ Structured Requirements

- A *structured requirement* defines an orderly requirement structure with specified attribute placeholders
  - Helps capture the precise meaning and communicate the required information to define a complete requirement
  - Existing concept from R. S. Carson 2015
- For example, a structured requirement statement may look like:

The [**Who**] *shall* [**What**] [**How Well**] under [**Condition**]. (1)

  - [**Who**]: Defines a subject term specified by an agent or user role that provides a capability or performs a function
  - [**What**]: Refers to an action verb term specified by a required functionality or characteristic
  - [**How Well**]: Indicates a comparison factor specified by constraints that can be applied to restrict the implementation of a required functionality or a design characteristic
  - [**Condition**]: Describes the measurable qualitative or quantitative terms specified by characteristics such as an operational scenario, environmental condition, or a cause that is stipulated

→ Structured Requirement Example

- For example, this a structured requirement written in natural language:

  The [**actuation system**]$_{\textbf{Who}}$ *shall* [**prevent inadvertent stowing**]$_{\textbf{What}}$ [**with at least three levels of safety**]$_{\textbf{How Well}}$ under [**normal deploy operation**]$_{\textbf{Condition}}$. (R1)

Remark

As currently presented, this is simply a refinement to the classical textual requirements approach and has only an imprecise relationship to the system model.

→ Classical SysML Requirements Modeling (1): Definition

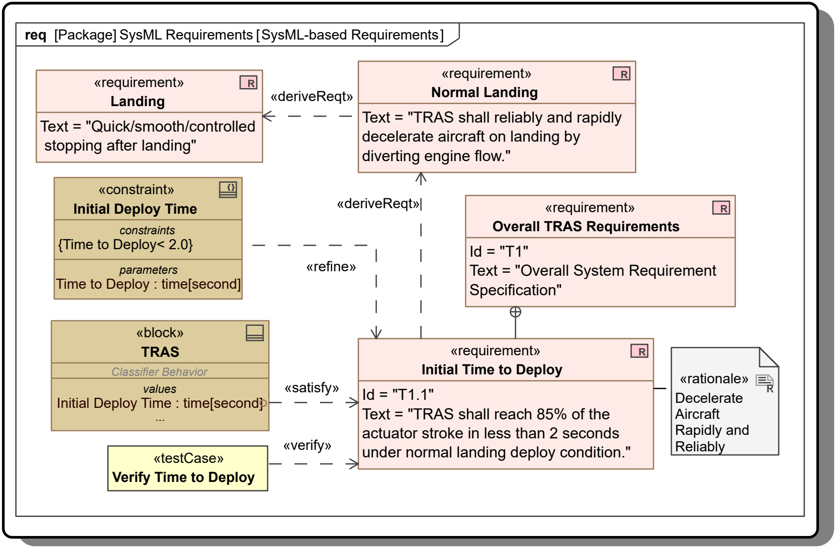
- As with many aspects of system modeling, SysML supports the inclusion of requirements[1]
- A classical SysML-based requirement is developed by defining:
  - Some predefined abstract attributes:
    - *Name*, *Id*, and *Text*
  - Some traceability relationships that include attributes such as:
    - *Owner*, *Derived*, *Derived From*, *Satisfied By*, *Refined By*, *Traced To*, and *Verified By*
  - Hierarchical relationships between requirements with a containment relationship

Remark

Even though some relationships between the requirement and system model are now possible, the primary text statement is static, which is more likely to be incomplete, contain errors, and be inconsistent with the rest of the SysML-based system model.

[1] *OMG System Modeling Language* 2019; Friedenthal, Moore, and Steiner 2015

→ Classical SysML Requirements Modeling (2): Example

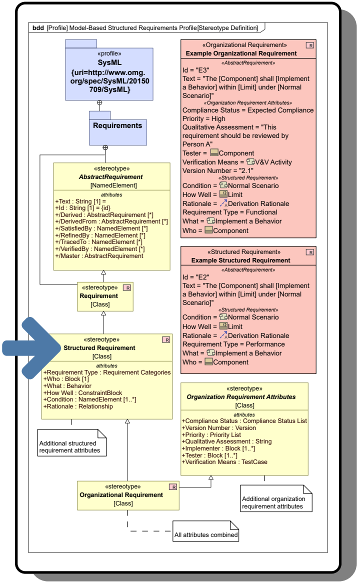


→ Model-Based Structured Requirement in SysML (1): Motivation

- To address the concerns regarding the use of textual structured requirements and classical SysML requirements modeling, we define here an approach that combines the two together to leverage the advantages of both[1]
- We want a more rigorous approach for structuring requirements that also directly links to the SysML system model elements, similar to the existing traceability relationships
- With direct links to the system model, this approach to requirements is more heavily model-based and structured; hence, we term these *model-based structured requirements* (MBSRs)

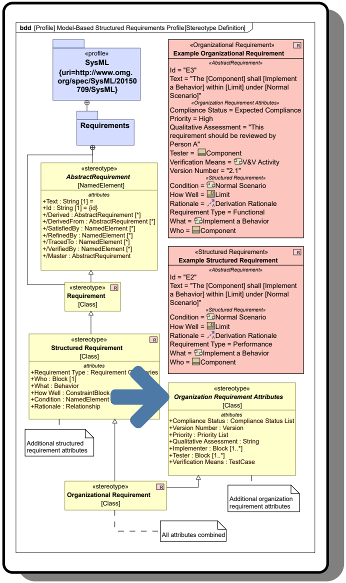
[1] Narsinghani 2021

→ Model-Based Structured Requirement in SysML (2): Stereotype



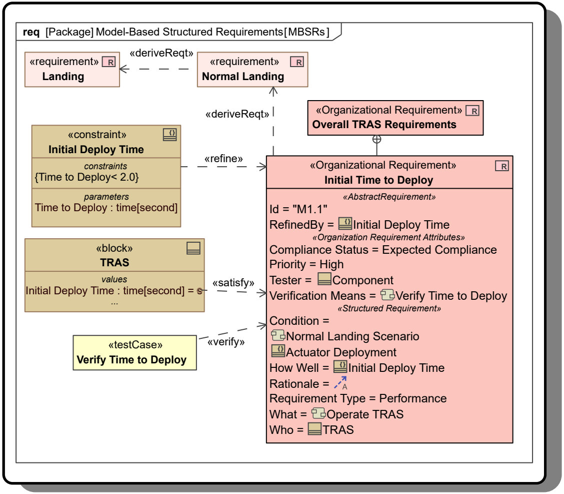
- ≪*Structured Requirement*≫ stereotype is defined in the figure on the left
- Now, an MBSR element can be created by:
  - Applying the stereotype to a requirement
  - Specifying the SysML attributes necessitated by this stereotype

→ Enhancing with Organizational Attributes



- Additionally, we seek to integrate organizational attributes into the fundamental definition of requirements that align with the specific rules and policies of the organization
- Attributes in ≪*Organization Requirement Attributes*≫ include **Compliance Status**, **Version Number**, **Priority**, **Qualitative Assessment**, **Implementer**, **Tester**, and **Verification Means**
- Then, ≪*Organizational Requirement*≫ combines ≪*Organization Requirement Attributes*≫ and ≪*Structured Requirement*≫

→ Model-Based Structured Requirement in SysML Example



- Connecting requirement attributes with model elements allows requirement information to remain current and available
- Therefore, this can further reduce the errors and iterations while developing requirements, saving time and other resources

③

Examples

→ Example Overview

- We now present several more examples of requirements implemented as MBSRs (Slide 16), as well as pure textual structured (Slide 14) and classical SysML requirements (Slide 15) for comparison
- All of the examples are based on a notional thrust reverser actuation system (TRAS)
  - A necessary subsystem in most commercial aircraft to achieve and maintain safe ground stopping distance after a touchdown in adverse conditions such as wet/slippery runways by reversing fan bypass air flow[1]
- The model for these examples and MBSR stereotype definition is publicly available on GitHub[2]

[1] Maré 2018; Yetter 1995 [2] https://github.com/danielrherber/model-based-structured-requirements

→ Textual Requirements for TRAS

ID	Name	Requirement Type	Text	Rationale
1.1	Initial Time to Deploy	Performance	TRAS shall reach 85% of the actuator stroke in less than 2 seconds under normal landing deploy condition.	Decelerate Aircraft Rapidly and Reliably
1.2	TRAS MTBF	Non-Functional	TRAS MTBF shall be greater than 15000 mission flight hours under normal landing condition.	Decelerate Aircraft Rapidly and Reliably
1.3	TRAS Average Power Consumption During Deploy Operation	Interface	During Thrust reverser deploy operation, from ECU/DCR opening to fully extended actuator position, TRAS average power consumption shall be lower than 35 kW.	Decelerate Aircraft Rapidly and Reliably
1.4	TRAS Fluid Interface	Interface	TRAS shall be able to withstand the hydraulic fluid temperature within a range of -70F to 280F on ground, under storage conditions.	Improve Efficiency, Safety and Sustainability
1.5	TRAS Weight Limit	Physical	System total mass shall be less than 320 pounds.	Differentiate with More Electric Aircraft
1.6	Jam During Reverser Deployment	Design Constraint	TRAS shall withstand the actuator lock jam without deformation when subjected to a compressive load of -5075 lbf.	Improve Efficiency, Safety and Sustainability
1.7	Interruption	Functional	TRAS shall be capable of changing the direction of reverser motion on command at any point in the actuation cycle under normal loading conditions.	Decelerate Aircraft Rapidly and Reliably

- Much of the necessary information is captured in the pure spreadsheet version
- However, the pieces of information are generally static, unlinked from any other representation of the system of interest

→ Classical SysML Requirements for TRAS

#	Id	Name	Text	Rationale	Derived From	Satisfied By	Refined By	Verified By
1	T1	T1 Overall TRAS Requirements	Overall System Requirement Specification					
2	T1.1	T1.1 Initial Time to Deploy	TRAS shall reach 85% of the actuator stroke in less than 2 seconds under normal landing deploy condition.	Decelerate Aircraft Rapidly and Reliably	3.1 Normal Landing	Initial Deploy Time : time[second]	Initial Deploy Time	Verify Time to Deploy
3	T1.2	T1.2 TRAS MTBF	TRAS MTBF shall be greater than 15000 mission flight hours under normal landing condition.	Decelerate Aircraft Rapidly and Reliably	3.6 TR Probability of Failure	TRAS MTBF : Mission Flight Time	TRAS MTBF	Verify TRAS MTBF
4	T1.3	T1.3 TRAS Average Power Consumption During Deploy Operation	During Thrust reverser deploy operation, from ECU/DCR opening to fully extended actuator position, TRAS average power consumption shall be lower than 35 kW.	Decelerate Aircraft Rapidly and Reliably	3.5 Energy Efficiency Gain	Average Power Consumption : power[kilowatt]	TRAS Average Power Consumption During Deploy Operation	Verify Average Power Consumption
5	T1.4	T1.4 TRAS Fluid Interface	TRAS shall be able to withstand the hydraulic fluid temperature within a range of -70F to 280F on ground, under storage conditions.	Improve Efficiency, Safety and Sustainability	3.1 Normal Landing	Fluid Temperature : thermodynamic temperature[kelvin] = 366.0 K	Hydraulic Fluid Temperature Range	Measure Hydraulic Fluid Temperature
6	T1.5	T1.5 TRAS Weight Limit	System total mass shall be less than 320 pounds.	Differentiate with More Electric Aircraft	3.4 TR Level Weight Constraint	Total Mass : mass[pound]	Total Mass	Verify Total Mass
7	T1.6	T1.6 Jam During Reverser Deployment	TRAS shall withstand the actuator lock jam without deformation when subjected to a compressive load of -5075 lbf.	Improve Efficiency, Safety and Sustainability	3.1 Normal Landing	TRAS	Overcome Jamming Loads	Verify Jamming Loads
8	T1.7	T1.7 Interruption	TRAS shall be capable of changing the direction of reverser motion on command at any point in the actuation cycle under normal loading conditions.	Decelerate Aircraft Rapidly and Reliably	3.1 Normal Landing	Control Motion	Deployment Direction	Verify Deployment Direction

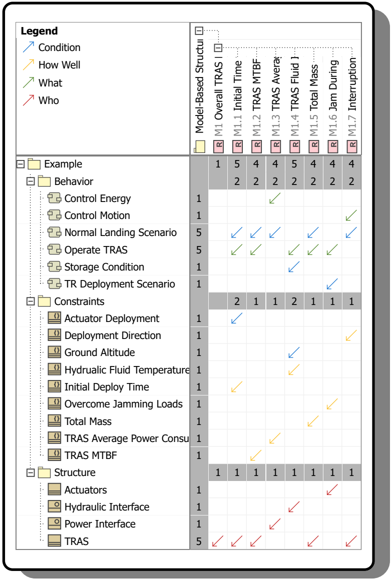
- Now, we have a lot more information connected to the system model
- However, we are still left to understand much of a specific requirement through its static text-based statement

→ Model-Based Structured Requirements (MBSRs) for TRAS

#	Id	Name	Requirement Type	Text	Who	What	How Well	Condition	Rationale	Satisfied By	Verification Means	Priority
1	M1	M1 Overall TRAS Requirements		Overall System Requirement Specification	TRAS							
2	M1.1	M1.1 Initial Time to Deploy	Performance	TRAS shall reach 85% of the actuator stroke in less than 2 seconds under normal landing deploy condition.	TRAS	Operate TRAS	Initial Deploy Time	Normal Landing Scenario Actuator Deployment	DeriveReqt[Initial Time to Deploy -> Normal Landing]	Initial Deploy Time : time[second]	Verify Time to Deploy	High
3	M1.2	M1.2 TRAS MTBF	Non-Functional	TRAS MTBF shall be greater than 15000 mission flight hours under normal landing condition.	TRAS	Operate TRAS	TRAS MTBF	Normal Landing Scenario	DeriveReqt[TRAS MTBF -> TR Probability of Failure]	TRAS MTBF : Mission Flight Time	Verify TRAS MTBF	Medium
4	M1.3	M1.3 TRAS Average Power Consumption During Deploy Operation	Interface	During Thrust reverser deploy operation, from ECU/DCR opening to fully extended actuator position, TRAS average power consumption shall be lower than 35 kW.	Power Interface	Control Energy	TRAS Average Power Consumption During Deploy Operation	Normal Landing Scenario	DeriveReqt[TRAS Average Power Consumption During Deploy Operation -> Energy Efficiency Gain]	Average Power Consumption : power[kilowatt]	Verify Average Power Consumption	Medium
5	M1.4	M1.4 TRAS Fluid Interface	Interface	TRAS shall be able to withstand the hydraulic fluid temperature within a range of -70F to 280F on ground, under storage conditions.	Hydraulic Interface	Operate TRAS	Hydraulic Fluid Temperature Range	Ground Altitude Storage Condition	DeriveReqt[Normal Landing -> Landing]	Fluid Temperature : thermodynamic temperature[kelvin] = 366.0 K	Measure Hydraulic Fluid Temperature	Low
6	M1.5	M1.5 Total Mass	Physical	System total mass shall be less than 320 pounds.	TRAS	Operate TRAS	Total Mass	Normal Landing Scenario	DeriveReqt[Total Mass -> TR Level Weight Constraint]	Total Mass : mass[pound]	Verify Total Mass	Low
7	M1.6	M1.6 Jam During Reverser Deployment	Design Constraint	TRAS shall withstand the actuator lock jam without deformation when subjected to a compressive load of -5075 lbf.	Actuators	Operate TRAS	Overcome Jamming Loads	TR Deployment Scenario	DeriveReqt[Jam During Reverser Deployment -> Safety Against IAS]	TRAS	Verify Jamming Loads	Medium
8	M1.7	M1.7 Interruption	Functional	TRAS shall be capable of changing the direction of reverser motion on command at any point in the actuation cycle under normal loading conditions.	TRAS	Control Motion	Deployment Direction	Normal Landing Scenario	DeriveReqt[Interruption -> Normal Landing]	Control Motion	Verify Deployment Direction	Low
9	M2	M2 [Incomplete Requirement]										

- We have more specific information about each requirement, and this information comes in the form of links to various model elements
- For example, if the Normal Landing Scenario definition changed, the requirement has a direct link to those changes

→ Model-driven Dependency Matrix



- More so than the classical SysML approach, we have shared model elements across the MBSRs
- This is automatically visualized and counted in a dependency matrix that includes the key ≪*Structured Requirement*≫ attributes
- TRAS is shared among multiple requirements as well as Operate TRAS and Normal Landing Scenario

→ Metrics Measuring MBSR Completeness

#	Name	Metric Suite	Scope	Complete Requirement Percentage	Total Requirements	Includes Who	Includes Condition	Includes How Well	Includes What
1	Version 1	MBSR Completeness Metrics	Example	37.5	8	5	5	6	6
2	Version 2	MBSR Completeness Metrics	Example	77.7778	9	8	7	7	7

- We can automatically assess requirement completeness with respect to the MBSR specification
- For example, an MBSR is considered complete when it has nonempty [**Who**], [**What**], [**How Well**], and [**Condition**] attributes
- These metrics are automatically computed in the model using customized metric suites and scripts[1]

[1] https://github.com/danielrherber/model-based-structured-requirements

4

Summary and Future Work

→ Summary

- When requirements are written in the classical text format, significant resources (including many brains) are required to develop and manage them, which can lead to identifying problems late in the development cycle
- Overall, the proposed MBSR approach is more aligned with the model-centric philosophy of system development through its more broad use of system elements
- It adds systems thinking rigor when developing the model (and requirements) that support the wide range of SE activities
- The MBSR *restricts* us to create and define the right elements and relationships (or readily see that they are missing)
- It also directly connects pieces that help define a requirement to the functional/physical architecture elements and system verification/validation artifacts with more specific and relevant relationships[1]

[1] Wheatcraft et al. 2022

→ Future Work

- Customized MBSR validation rules will help eliminate errors and improve the quality in the early stages of requirements elicitation in a more automated manner
- More refinements to the attributes (both in naming, typing, and completeness) should be investigated to ensure that they holistically capture the concerns in requirements development
- Further investigations into requirement and model metrics (see Slide 18) that help define requirement completeness utilizing the structured attributes
- Development of specific profiles for different types of structured requirements

→ References

R. Carson (2021). *Developing Complete and Validated Requirements*. INCOSE Seattle-Metropolitan Chapter Monthly Meeting. DOI: 10.13140/RG.2.2.28526.74561

R. S. Carson (2015). "Implementing Structured Requirements to Improve Requirements Quality". *INCOSE International Symposium* 25.1. DOI: 10.1002/j.2334-5837.2015.00048.x

S. Friedenthal, A. Moore, and R. Steiner (2015). *A Practical Guide to SysML*. 3rd. Elsevier. DOI: 10.1016/c2013-0-14457-1

IEEE Guide for Developing System Requirements Specifications (1998). IEEE. DOI: 10.1109/ieeestd.1998.88826

ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering (2018). IEEE. DOI: 10.1109/ieeestd.2018.8559686

J.-C. Maré (2018). *Aerospace Actuators 3: European Commercial Aircraft and Tiltrotor Aircraft*. John Wiley & Sons, Inc. DOI: 10.1002/9781119505433

Model-based structured requirements (n.d.). URL: https://github.com/danielrherber/model-based-structured-requirements

J. B. Narsinghani (2021). "Towards a model-based implementation in technology/platform life cycle development processes applied to a thrust reverser actuation system (TRAS) concept". MA thesis. Colorado State University

→ References (continued)

- *OMG System Modeling Language* (2019). Object Management Group
- K. Pohl (2010). *Requirements Engineering*. Springer
- M. Ryan et al. (2019). *Guide for Writing Requirements*. Tech. Prod. INCOSE-TP-2010-006-03. INCOSE Requirements Working Group
- L. Wheatcraft et al. (2022). *Needs, Requirements, Verification, Validation Lifecycle Manual*. Tech. Prod. INCOSE-TP-2021.002-01. INCOSE Requirements Working Group
- J. A. Yetter (1995). *Why Do Airlines Want and Use Thrust Reversers? A Compilation of Airline Industry Responses to a Survey Regarding the Use of Thrust Reversers on Commercial Transport Airplanes*. Tech. rep. NASA-TM-109158. NASA Langley Research Center

Thank you.

Model-Based Structured Requirements in SysML

Daniel R. Herber*
Jayesh B. Narsinghani
Kamran Eftekhari-Shahroudi

https:// github.com/ danielrherber/ model-based-structured-requirements