LAB LECTURE 8:
REQUIREMENTS, REQUIREMENTS DIAGRAMS AND REQUIREMENTS TABLES

TAs: Pranay Kanagat, Arturo Davila, Siddharth Bansal, Kersasp Cawasji.
Instructor: John Baras
ENES 489P
Spring, 2018

PURPOSE

- The purpose of this lecture is to provide:
  - An overview of the INCOSE Requirements Definition process.
  - The different levels of requirements.
  - The different types of requirements.
  - The INCOSE Guide for Writing Requirements.
  - An overview of Requirements Decomposition Analysis.
  - Explain the purpose and general use of a Requirements Diagram.
  - Show how to construct Requirements Diagrams that may be used to generate a Requirements Document.
  - Show how to generate a Requirements Table that may be used as a Requirements Document.

REFERENCES

- **Chapter 4.3 of the INCOSE SE Handbook**
- K&S Chapter 7.7
- **Guide for Writing Requirements Summary Sheet, V. 2.0, INCOSE, 2015**
- Guide for Writing Requirements, V. 2.0, INCOSE, 2015
- IEEE Standards <http://www.ieee.org>:
  - **IEEE Std 1233-1998, IEEE Guide for Developing System Requirements Specifications***
  - **ANSI/IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications**
- Active DOD Standards <http://www.everyspec.com >
  - Defense and Program-Unique Specifications Format and Content [MIL-STD-961E, August, 2003]
  - **System/Subsystem Specification Data Item Description (SSS DID) DI - IPSC-81431**
  - Interface Requirements Specifications (IRSs) (DI-IPSC-81434)

- IEEE Standards <http://www.ieee.org>:
  - **IEEE Std 1233-1998, IEEE Guide for Developing System Requirements Specifications***
  - **ANSI/IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications**
- Active DOD Standards <http://www.everyspec.com >
  - Defense and Program-Unique Specifications Format and Content [MIL-STD-961E, August, 2003]
  - **System/Subsystem Specification Data Item Description (SSS DID) DI - IPSC-81431**
  - Interface Requirements Specifications (IRSs) (DI-IPSC-81434)

REQUIREMENTS

- A **requirement** is a property of the system that must be provided in order for the system to be acceptable to the user (that the customer is willing to pay to have) .
- **Stakeholders**/Users/Operational **Requirements**:
  - Generally focused on the **required high-level operational capabilities** of the system (MOEs and KPPs).
- **System Requirements:**
  - Specifies the **system-level technical requirements** for the system.
- According to INCOSE the purpose of the **System Requirements Definition process** is “to **transform** the stakeholder, user-oriented view of desired capabilities **into a technical view** of a solution the meets the operational needs of the user.”

REQUIREMENTS DEFINITION PROCESS

- **System (and Stakeholder) Requirements Definition is perhaps the most important job of a systems engineer.**
  - Poor requirements for complex systems can almost guarantee failure.
  - The Stakeholders Needs and Requirements Definition Process may be looked at as a special case of the System Requirements Definition Process.
- The **system requirements** serve as the technical **basis for the entire development effort.**
- **Mistakes** in this process will result in:
  - The **wrong thing being built** (and likely customer rejection of the product)
  - Cost and schedule **overruns** (and possible project cancellation)
  - Potential **loss of reputation** and/or **legal action**
- Defining system (and element) requirements one of the **most important jobs** of a systems engineer

LEVELS OF REQUIREMENTS

- **Stakeholders**/Users/Operational **Requirements**:
  - Generally focused on the **required high-level operational capabilities** of the system (MOEs and KPPs).
  - Written in terms that are meaningful to the stakeholders (users).
  - May include some cost and schedule requirements.
- **System Requirements:**
  - Specifies the **system-level technical requirements** for the system.
  - Derived from the Stakeholder Requirements.
  - Written in legally binding engineering terms.
- **Element Requirements:**
  - Specifies the technical requirements for a given element.
  - Derived from the System Requirements.
  - Written in engineering terms.
- **Sub-element Requirements:**
  - Specifies the technical requirements for a given sub-element.
  - Derived from the Element Requirements.

TYPES OF REQUIREMENTS

- **Programmatic Requirements:**
  - Cost Objectives
  - Schedule Objectives
  - Other Programmatic Requirements
- Programmatic Requirements are generally documented in program baseline documents and possibly the stakeholders' requirements documents, but not system or element specifications
- **Capability Requirements:**
  - Functional Requirements
  - Performance Requirements
- **Technical Requirements:**
  - States and Modes
  - Interface Requirements
  - Design Constraints
  - Other ("Non-Functional" Technical) Requirements
    - Standards
    - Specialty engineering –related requirements
    - Others
- Technical Requirements are generally documented in system, element, and sub-element specifications.

TYPES OF PERFORMANCE REQUIREMENTS

- **Measures of Effectiveness (MOEs)** are "'operational' measures of success that are closely related to the achievement of the mission or **operational objective** being evaluated, in the intended operational environment under a specified set of conditions; i.e., how well the solution achieves its intended purpose."
  - Generally an important high-level **operational requirement.**
- Measures of Suitability (MOSs) are similar to MOEs and not addressed by the SEHB
- **Measures of Performance (MOPs)** are "measures that characterize **physical or functional attributes** relating to the system operation, measured or estimated under specified testing and/or operational environmental conditions."
  - Generally an important high-level **system requirement** or constraint.
  - **System or Element Level (Not addressed in incose)**
- **Key Performance Parameters (KPPs)** are "a **critical subset** of performance parameters representing those capabilities and characteristics so significant that **failure to meet** the threshold value of performance can be **cause for** concept, or system selection to be reevaluated or the project to be reassessed, or **terminated**."
  - Generally an MOE or MOP.
- **Technical Performance Measures (TPMs)** "measure attributes of a system element to determine how well a system or system element is satisfying a technical requirement or goal."
  - Generally an important system or element performance requirement or constraint.
  - Generally related to a MOP and provides insight into the progress toward, or likelihood of, achieving the requirement associated with the MOP
- Critical Technical Parameters (CTPs) similar to TPMs and not addressed by the SEHB.

WRITING REQUIREMENTS

- **Writing good requirements is one of the most important jobs of a SE.**
- There are a number of guidelines for writing good requirements.
  - The INCOSE “Guide to Writing Requirements Summary Sheet” is probably the “gold standard”
  - Many of these are captured in the:
    - Chapter 4.3 of the INCOSE Systems Engineering Handbook
    - K&S
  - Every SE text book has a list that is similar to these.

SEHB CHARACTERISTICS OF GOOD REQUIREMENTS

- **Individual Requirements:**
  - Every requirement should be a **"Shall Statement,"** i.e., use the syntax "The system shall …"
  - **Necessary:** every requirement should be necessary. There are three forms of unnecessary requirements - desirements, redundant requirements, over-specification.
  - **Implementation independent:** specify what the system needs to do, not the equipment/ design that will be used (i.e., not how it is to be done)
  - **Unambiguous (aka Clear):** it should not be subject to interpretation. All terms should be clearly defined.
  - **Verifiable (aka Testable):** it should be clear how one can verify that the requirement has been achieved using one of the four standard methods of verification.
  - **Complete:** the requirement should not need further description.
  - **Singular:** the requirement should only state one requirement, not a collection of requirements (e.g., connected by "and")
  - **Achievable:** the requirements should be technically achievable given the cost and schedule constraints.
  - **Conforming (to applicable standards)**
  - **Traceable:** the requirement may be traced to one or more a higher-level requirements.
- **Sets of Requirements:**
  - **Complete:** the set of requirements for an entity includes all the requirements for that entity.
  - **Consistent:** there are no contradictory or inconsistent requirements.
  - **Feasible/Affordable:** the set of requirements are achievable given the cost and schedule constraints.
  - **Bounded:** the set of requirements for an entity should define the entire required scope of the solution to the stakeholder needs.

EXAMPLES OF POOR AND BETTER REQUIREMENTS

- What is wrong with the following examples?
  - "The system should process data."
    - Many problems
  - "The system shall use a 11kW Generac Guardian 6439 Home Standby Generator."
    - One problem
- collect, purify, and store all the water that enters the system."
    - Many problems

- Better Examples:
  - The system shall accept "processed sensor data" from the sensor (see "Data Dictionary Appendix for definition of processed sensor data).
  - The system shall provide 11 kW of power to the user.
  - The system shall collect up to 100 gal of water over a 24 hour period.
  - The system shall store up to 80 gal of water.

REQUIREMENTS ANALYSIS

- The objective of **requirements analysis** is “to provide an understanding of the interactions between various functions **[in order] to obtain a balanced set of requirements** based on user objectives.”
- Some Types of Requirements Analyses:
  - Performance Analyses
  - Constraints Analyses:
    - Standards
    - Utilization environments
    - Essential design considerations (see specialty engineering)
    - Design constraints
  - Specialty Engineering Analyses
  - Completeness and Consistency Analyses
  - Traceability Analyses
  - Feasibility Analyses
  - Cost Analyses
  - Cost/Affordability Analyses
  - Schedule Analyses
  - **Trade-off Analyses**
  - Others
- Models and Simulations are used to support these analyses
- We will consider this topic in more detail later in the course

SOME KEY EARLY REQUIREMENTS ANALYSES

- Stakeholder Needs Analysis
- Requirements Completeness & Clarity Analysis
- Functional Requirements Analysis
- **Requirements Decomposition Analysis**
- Interface Requirements Analysis
- **Requirements Verification Analysis**
- **Requirements Traceability Analysis**
- **Requirements Allocation Analysis**
- **Requirements Quality Analysis**

REQUIREMENTS DECOMPOSITION ANALYSIS

- **Requirements Decomposition Analysis** consists of defining higher level requirements in terms of lower-level requirements (often called “derived requirements”).
- The **purpose of requirements decomposition** is to decompose higher-level requirements into lower-level requirements to the point where:
  - The meaning of the higher-level requirement is complete and unambiguous (i.e., clear)
  - Each “leaf node” (lowest-level) requirement can be allocated to a single logical element (sub-element).
- There are **three major forms** of requirements decomposition:
  - **Functional decomposition** (e.g., understand what it means to “monitor” the forest)
  - **Performance decomposition** (e.g., deriving element reliability and maintainability requirements from system-level availability requirements)
  - **Constraints decomposition** (e.g., deriving element weight constraints from system weight constraints)
- **<u>This is one of the most important forms of analysis</u>.**
  - The results of this analysis are reflected in the lower-level structure of **requirements documents** and is therefore the principal basis for **system/element verification testing**.
- Principal Tools and Techniques:
  - Requirements Diagrams & ADs

REQUIREMENTS DECOMPOSITION ANALYSIS

- Functional Decomposition Heuristics:
  - Decompose higher-level requirement into 3-8 lower-level requirements.
    - If there are less that 3, you should consider reorganizing the higher-level requirements
    - If there are more that 8 you should consider creating a lower level requirement that is composed of three even lower-level requirements.
  - Make sure that your **set of lower-level requirements is complete**, i.e.,
    - By convention/definition, if all lower-level requirements are satisfied, the parent-level requirement is considered satisfied.
    - If this is not the case, your set of lower-level requirements is NOT complete.
  - Continue decomposition until the meaning of the upper-level requirement is unambiguous.

REQUIREMENTS DECOMPOSITION ANALYSIS

- **Example:** Provide Mobile Monitoring System Capabilities
  - Provide Mobility
    - Maximum cruise speed of 100 mph
    - Range of 200 mi.
    - Turning Radius of < 100 m.
  - Monitor Environment
    - Provide sensor coverage of an area of 100 sq mi.
    - Provide Zoom Capability
      - Provide a field of view (FOV) that varies from 1 sq mi to 10 sq mi
      - Provide image data with a resolution of 1 sq meter for the 1 sq mi. FOV
      - Provide image data with a resolution of 10 sq meter for the 10 sq mi. FOV
    - Detect Stationary Objects
      - Detect SOs under standard conditions
        - Detect objects that are larger that X with a Pfd = 0.1 and Pmd = 0.2
      - Detect SOs under degraded conditions
      - Detect SOs under harsh conditions
    - Detect Moving Objects
    - Classify Objects

- **Example continued:**
  - Control System
  - Provide Communications
    - Provide Human Interface
      - Accept input data
      - Display output data
      - Provide alarms
    - Provide External Communications
    - Provide Internal Communications
      - Uplink Data Rate
      - Downlink Data Rate
      - …

VERIFICATION, TRACEABILITY, & ALLOCATION ANALYSES

- Requirements Verification Analysis
  - RVM
- Requirements Traceability Analysis
  - RTM
- Requirements Allocation Analysis
  - RAM
- These will be addressed in upcoming lectures

THE REQUIREMENTS DIAGRAM

PURPOSE/USE OF A REQUIREMENTS DIAGRAM

- A **Requirements Diagram (req)**:
  - "represents text-based requirements and their relationship with other requirements, design elements, and test cases to support requirements traceability ..." [A Practical Guide]
  - "captures text-based requirements." [INCOSE SEHB]
- Major Uses:
  - Used to capture and document formal system, element, and sub-element requirements.
  - **Supports automated requirements document generation.**
  - Can support requirements management.
  - Provides a graphical representation of these requirements.
  - Provides linkages to other elements of the system architecture.
  - Supports requirements decomposition/definition.
  - Supports **Requirements Analysis** (i.e., completeness, consistency, & traceability)
  - Supports requirements **allocation**.
  - Supports **design verification**.
  - Supports **system**/element/sub-element **verification and validation**.

SOME TYPES OF REQUIREMENTS DIAGRAMS

- **Requirements Decomposition Diagrams => Requirements Documents**
  - **We will focus on this type.**
- Requirements Verification Diagrams => RVM
- Requirements Trace Diagrams => RTM
- Requirements Allocation Diagrams => RAM

BASIC REQUIREMENTS DIAGRAM ELEMENTS

- **Requirements Blocks:**
  - Specifies a requirement.
  - Represented by box with various components.
  - **Requirement Title Component:**
    - Unique for each block
    - Can be a category of requirements (e.g., performance requirements)
    - Or a predicate phrase (an action) (e.g., detect intruders)
  - **Requirement Specification Component:**
    - **Requirement ID:**
      - Unique for each block
      - Generally expressed in a “Dot” notation (series of characters separated by dots)
      - By our convention, the first two or three characters are used to denote a stakeholder requirement (SH), system requirement (S), or Element X requirement (EX).
      - The next digit is used to denote which level 1 requirement is being addressed
      - The next digit is used to denote which level 2 requirement is being addressed
      - …
    - **Requirement Text:**
      - A complete, grammatically correct sentence that expresses the requirement.
      - It should be a “shall statement” that adheres to the criteria associated with “writing good requirements.”
  - Other Components (used to display relationships)

- **Relationships:**
  - **Containment Relationship:**
    - Indicates that a higher-level requirement is **completely defined** by the associated set of connected, lower-level requirements.
    - Represented by a line with a “circled X” at the end.
    - Note: this implies that **if all lower-level (leaf) requirements are satisfied, the upper-level requirement is satisfied**.
    - If this is not the case, additional definition is required.
    - **In general, a given higher-level Requirement Block should have 3-9 leaf nodes.**
  - Satisfy Relationship:
    - Indicates the system/element/sub-element that is expected to satisfy the requirements (or to which the requirement is allocated).
    - Represented by a dashed arrow.
  - Verify Relationship:
    - Indicate the tests and/or test case(s) that will be used to verify that the system/ element/ sub-element satisfies a given requirement.
    - Represented as a dashed arrow.
  - Other Relationships:
    - DeriveReq Relationship
    - Refine Relationship
    - Trace Relationship
    - Copy Relationship

DEVELOPING REQUIREMENTS DIAGRAMS

- Identify the requirement level being addressed (i.e., Stakeholder, System, Element, etc.)
- Identify the source documents for this level.
- Identify the principal, overarching capability that is to be provided by the system (e.g., “Provide X capability”)
- Identify the important categories of requirements for this level:
  - Example programmatic categories: cost, schedule, technical.
  - Example technical categories: States and Modes, Performance, Functional, Interface, Data, Specialty Engineering, Design Constraints, Other.
- Identify leaf requirements from higher-level requirements diagrams (sources) that were allocated to this system/ element/ sub-element.
  - Refine each of these so they are well-defined within the context of the system/element under consideration.
  - Refine each of these to the level they may allocated to the next lower-level element/ sub-element.
- Use the “writing good requirements” criteria and techniques in writing the text portion of each requirements block.
- Extend the system architecture to address the current level.
  - BDDs & IBDs
  - UCNs, Activity Diagrams, & Sequence Diagrams
  - State Machine Diagrams
- Use the Architecture to verify completeness and consistency of the requirements captured by the Requirements Diagram.
- Validate requirements with customer
- **Some Important Heuristics:**
  - The **name** of a given **system/ element/ sub-element** should be the **same** in every SysML diagram.
  - Every BDD/IBD **operation** and AD **activity** allocated to the system/element should correspond to a **functional requirement in the RD**.
    - Names of bdd/ibd operations and AD activities should correspond to the functional requirements predicate phrase found in the RD.
  - **RD performance requirements** should appear as **values** in BDD/IBD Blocks.
  - The same name should be used for the same **message, data structure, or data element** in all SysML artifacts (including the RD).

PURPOSE/USE OF A REQUIREMENTS TABLE

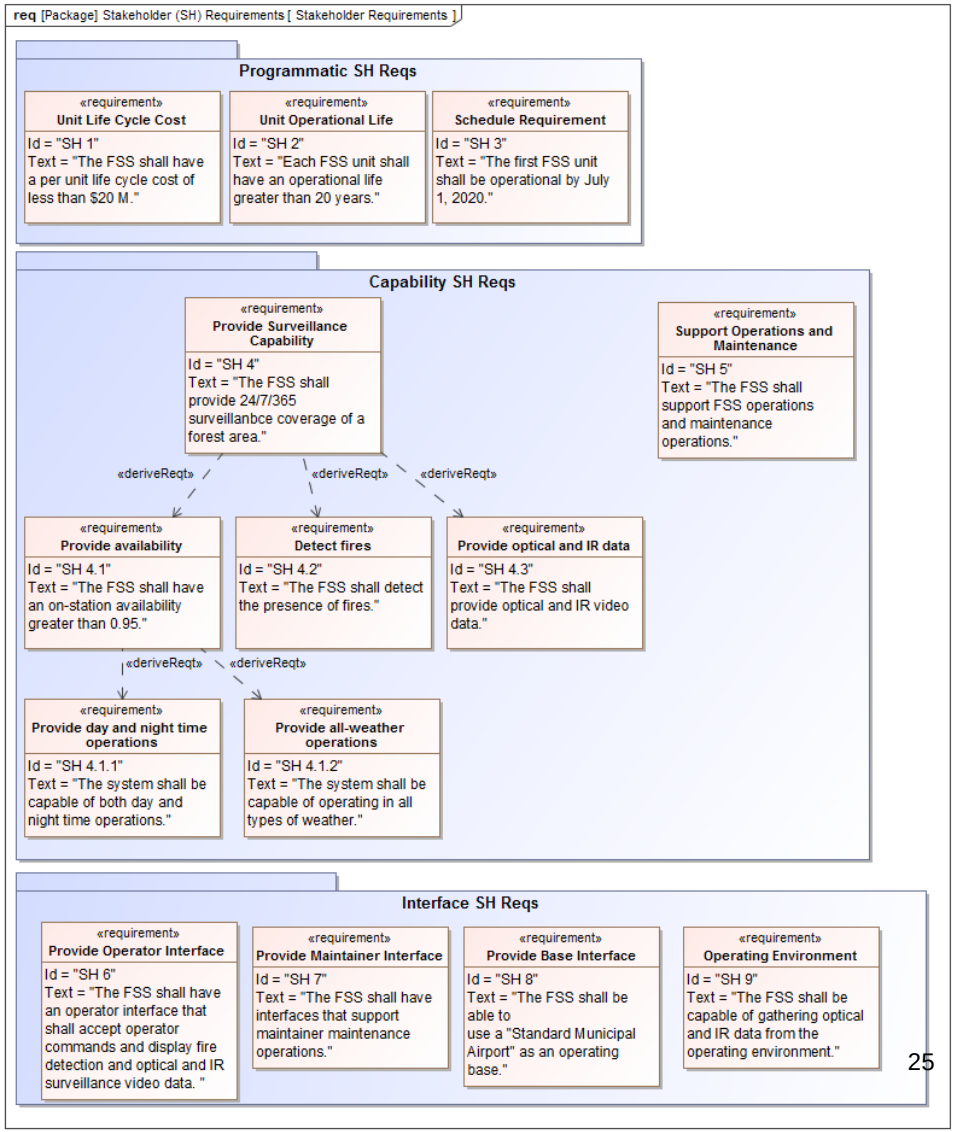
- A **SysML Requirements Table** provides a hierarchical tabular representation of the requirements captured in the Requirements Diagram.
  - Automatically generated from a Requirements Diagram
- Major Uses:
  - Compact summary of the requirements captured in a Requirements Diagram.
  - **May be used to construct a Requirements Document.**
  - **May be used to construct a Requirements Trace Matrix.**
  - **May be used to construct a Requirements Verification Matrix.**
  - **May be used to construct a Requirements Allocation Matrix.**

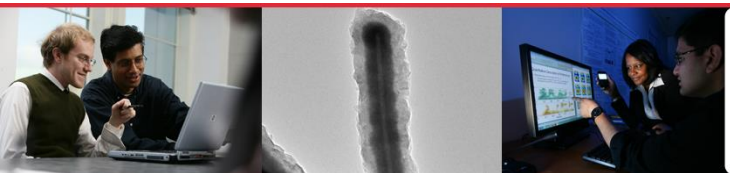
STAKEHOLDER REQUIREMENTS ORGANIZATION

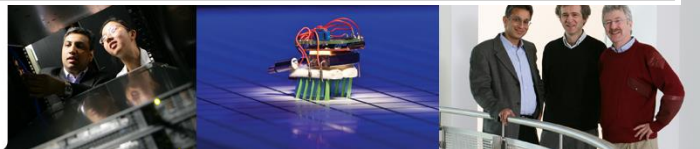
- Use a requirements organization scheme that will facilitate automated generation of a requirements diagram
  - Use separate RDs for different chapters
  - Use an Requirements ID and Title scheme that is consistent with chapter organization.
- As an example, consider developing separate RDs that address
  - Programmatic Requirements:
    - Cost goals – cost of the product
    - Schedule goals – when product needs to be available
    - Other programmatic requirements
  - System Capabilities:
    - What the stakeholders want the system to do and how well
  - External Interface Requirements
  - Specialty Engineering Requirements
  - Other Requirements

STAKEHOLDERS' REQUIREMENTS DIAGRAM

- The Stakeholders' Requirements diagram show the stakeholders' requirements.
- It should be structured to separate programmatic requirements from operational requirements and to support system requirements traceability.
- The ID/Numbering system should be such that one may reference stakeholder requirements unambiguously when doing an upward trace from system requirements.

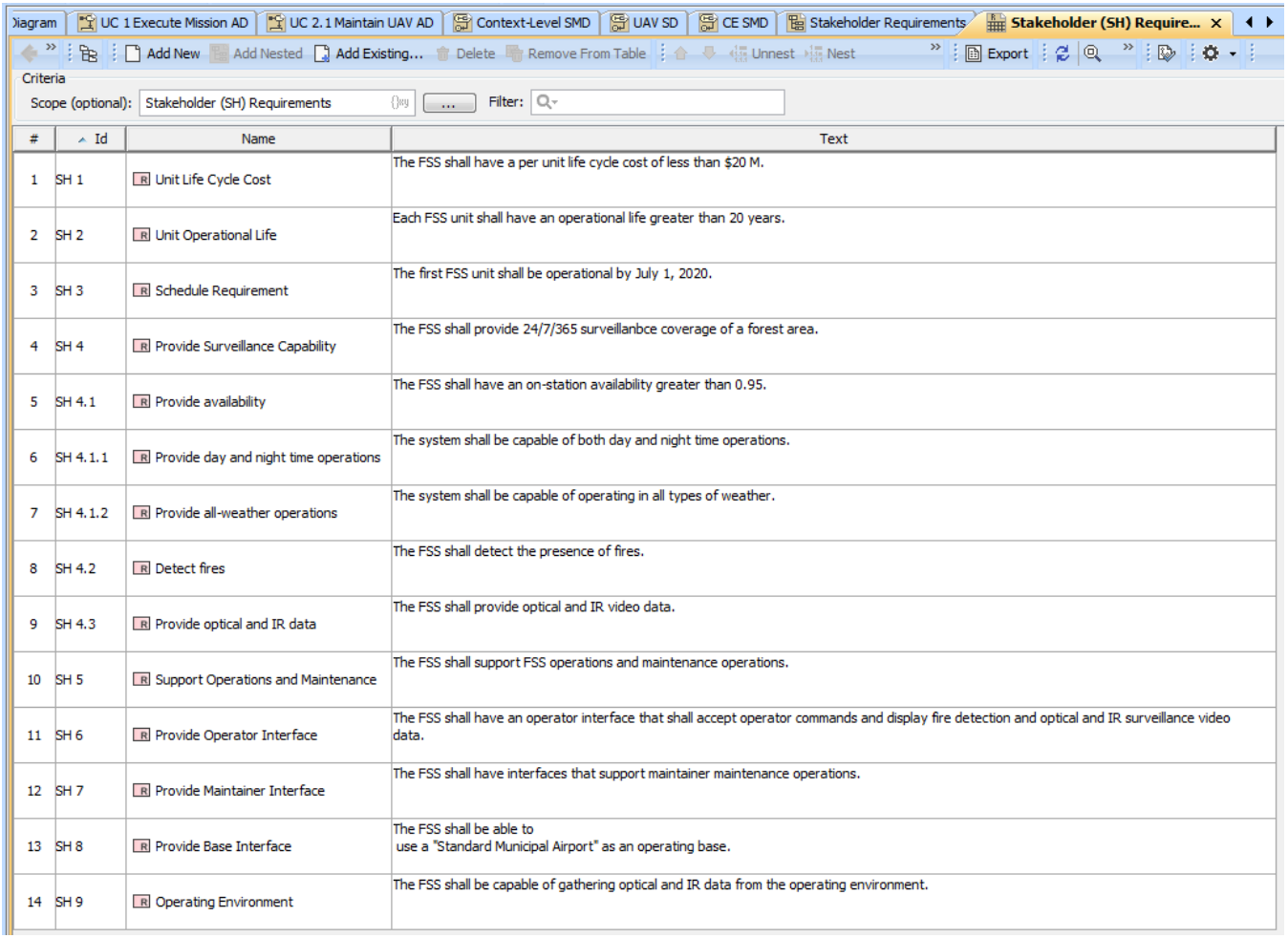






STAKEHOLDERS' REQUIREMENTS TABLE

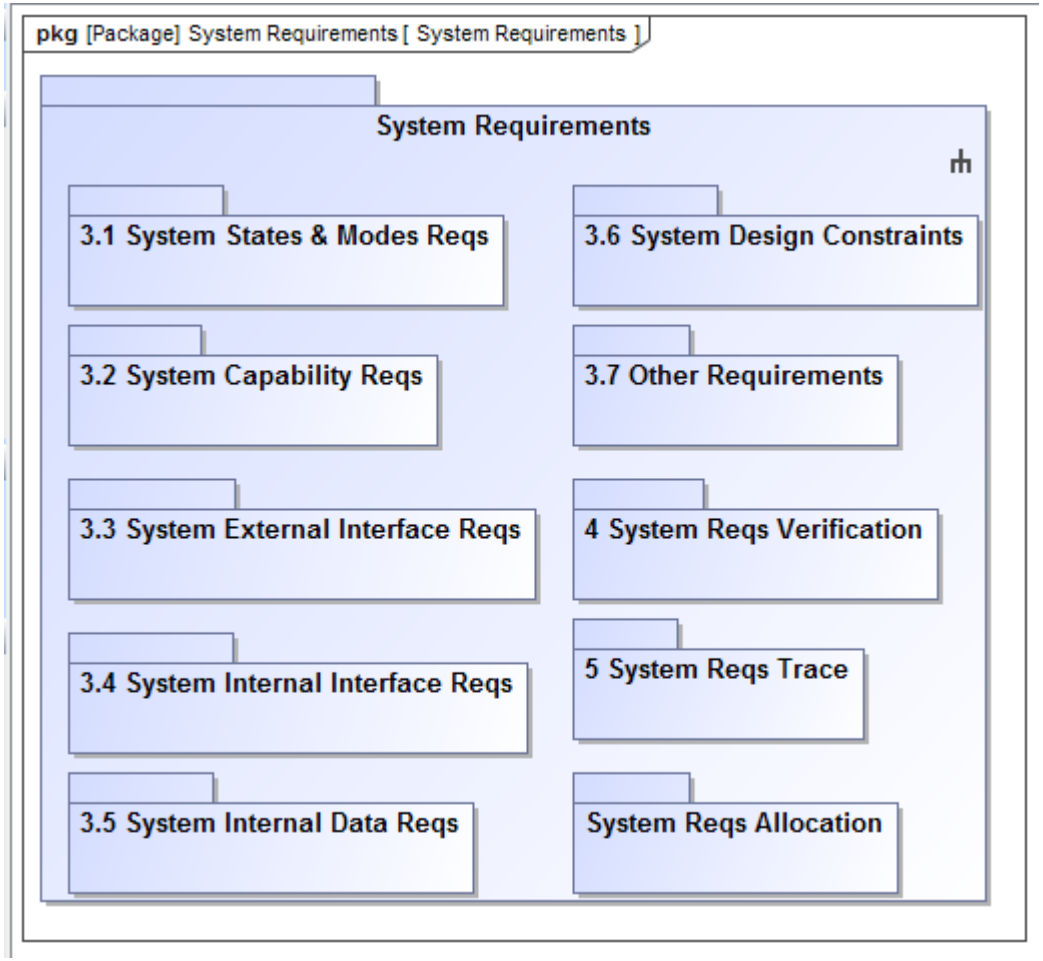
- The Stakeholders' Requirements Table represents an instantiation of the requirements diagram that may be exported and used to construct a Stakeholders' Requirements Document.



#	Id	Name	Text
1	SH 1	Unit Life Cycle Cost	The FSS shall have a per unit life cycle cost of less than $20 M.
2	SH 2	Unit Operational Life	Each FSS unit shall have an operational life greater than 20 years.
3	SH 3	Schedule Requirement	The first FSS unit shall be operational by July 1, 2020.
4	SH 4	Provide Surveillance Capability	The FSS shall provide 24/7/365 surveillanbce coverage of a forest area.
5	SH 4.1	Provide availability	The FSS shall have an on-station availability greater than 0.95.
6	SH 4.1.1	Provide day and night time operations	The system shall be capable of both day and night time operations.
7	SH 4.1.2	Provide all-weather operations	The system shall be capable of operating in all types of weather.
8	SH 4.2	Detect fires	The FSS shall detect the presence of fires.
9	SH 4.3	Provide optical and IR data	The FSS shall provide optical and IR video data.
10	SH 5	Support Operations and Maintenance	The FSS shall support FSS operations and maintenance operations.
11	SH 6	Provide Operator Interface	The FSS shall have an operator interface that shall accept operator commands and display fire detection and optical and IR surveillance video data.
12	SH 7	Provide Maintainer Interface	The FSS shall have interfaces that support maintainer maintenance operations.
13	SH 8	Provide Base Interface	The FSS shall be able to use a "Standard Municipal Airport" as an operating base.
14	SH 9	Operating Environment	The FSS shall be capable of gathering optical and IR data from the operating environment.

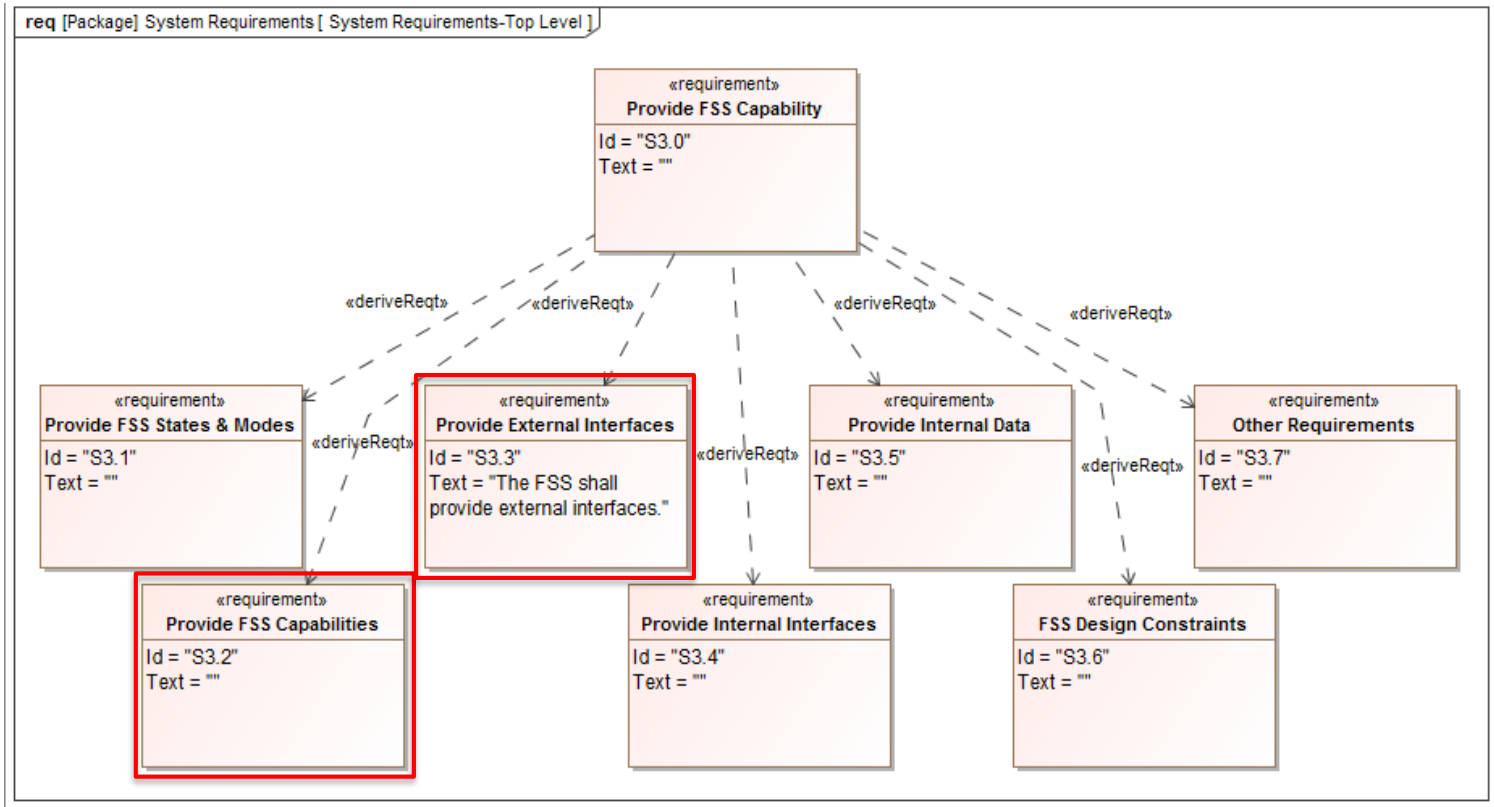
SYSTEM REQUIREMENTS DIAGRAM(S)

- The System Requirements diagram shows the system's requirements.
  - It should be structured so as to support the generation of a requirements document.
    - See Package Diagram to the right
  - The ID/Numbering system should be consistent with the requirements document to ease construction of the document.
  - Consider decomposing system requirements should be decomposed to the level that the "leaf nodes" may be allocated to elements.
  - When decomposing requirements, make sure that the upper level requirement is met iff all lower-level requirements are met.
- **It is generally a bad idea to try to put all system requirements into one diagram**
  - Rather one should divide the diagrams up into "chunks" that fit into the structure of the requirements document.
  - For slide presentations, "chunks" should have less than 20-30 requirements.
  - You may nest the diagrams

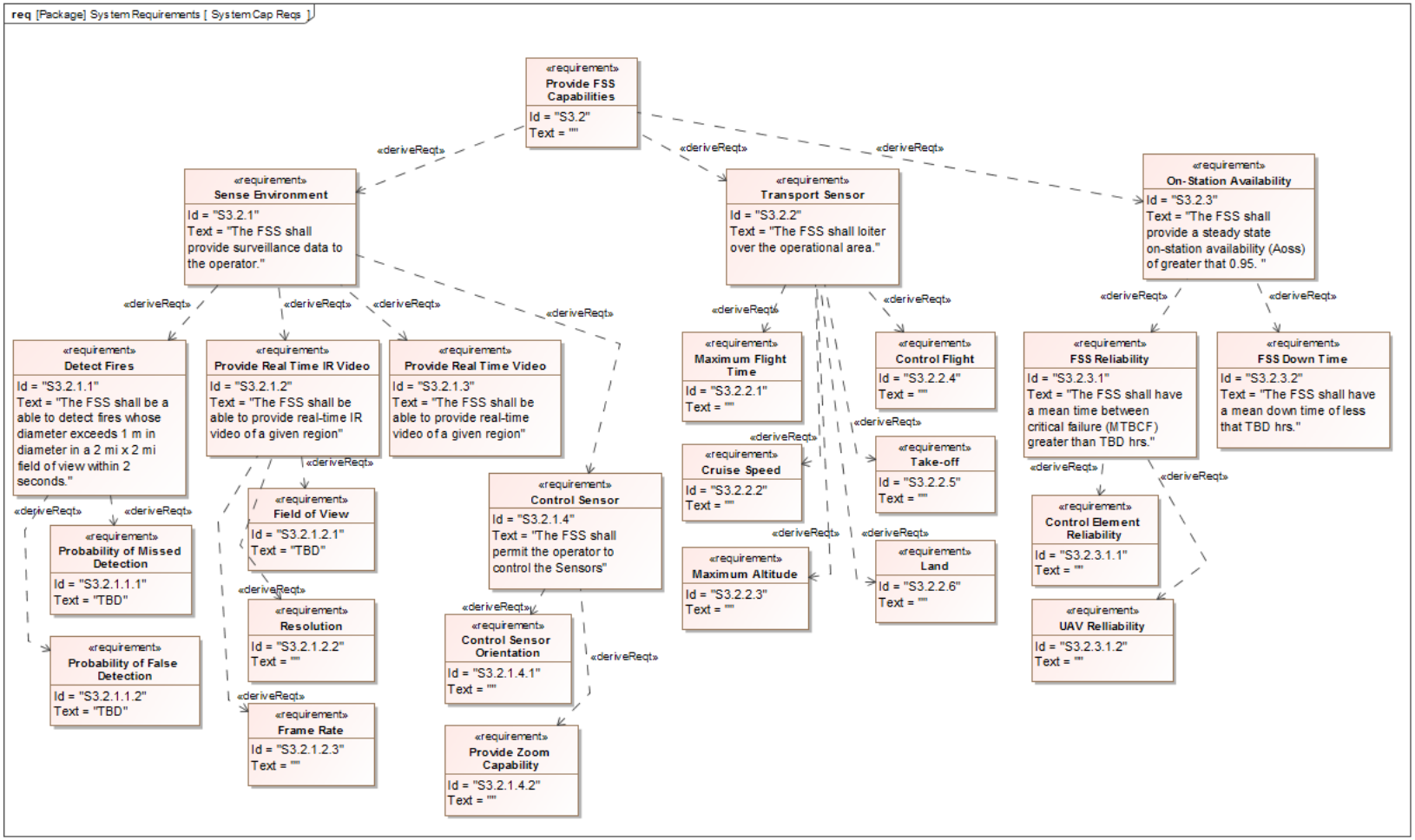


SYSTEM REQUIREMENTS DIAGRAM(S): NESTING EXAMPLE

- The following shows how the system requirements decompose to the next lowest level.
  - Each of the leaf node requirements here should have one or more diagrams devoted to their decomposition (nested as needed).
  - In the following two slide we will examine diagrams devoted to the decomposition of S3.2 and S3.3.
    - Note “S” indicates these are system requirements. Element requirements would be denoted by an identifier that indicated the specific element.

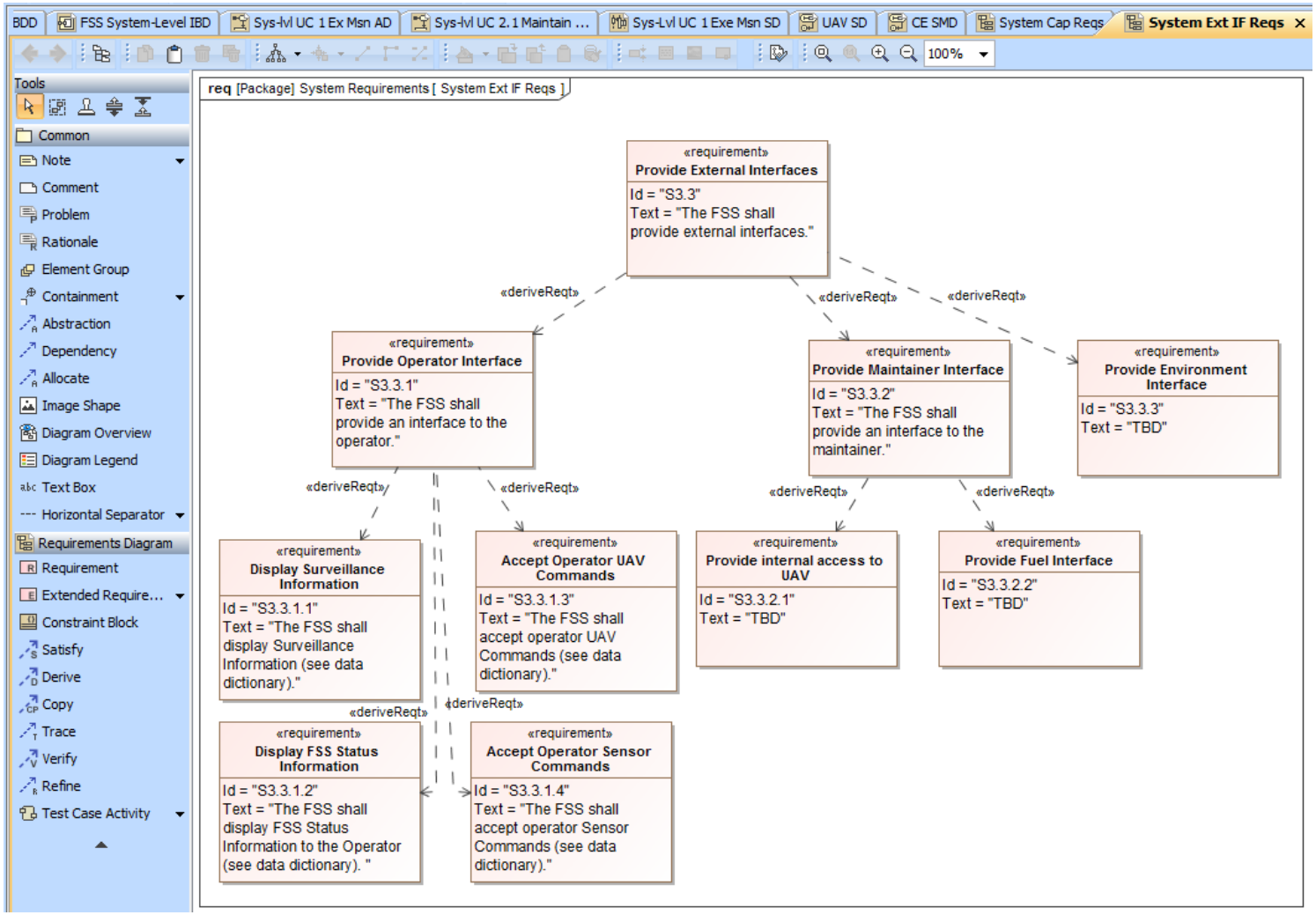


SYSTEM REQUIREMENTS DIAGRAM: SYSTEM CAPABILITIES



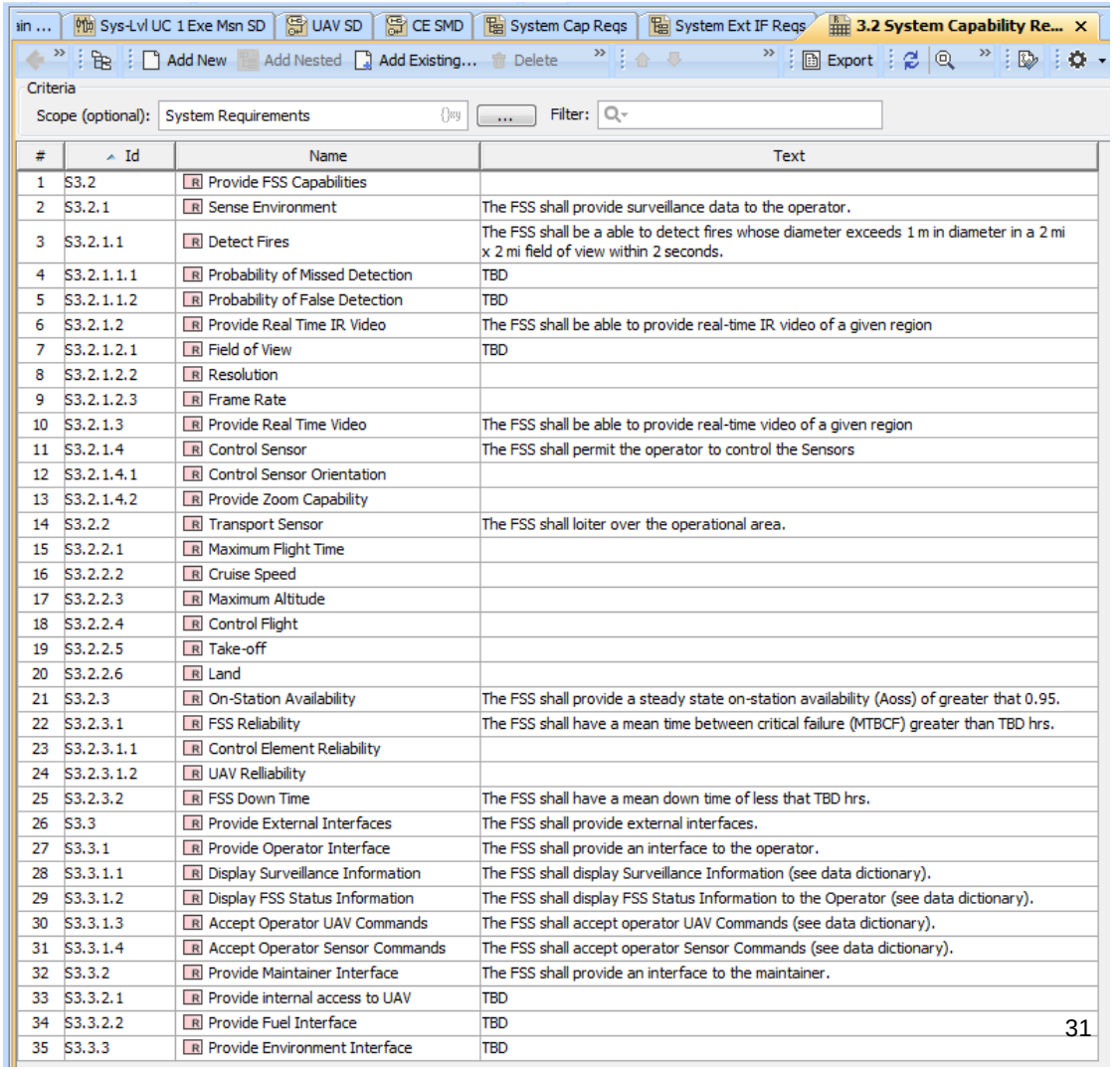
As you can see, this will likely have to be broken into nested lower-level diagrams as the decomposition continues.

SYSTEM REQUIREMENTS DIAGRAM: EXTERNAL INTERFACE REQUIREMENTS



SYSTEM REQUIREMENTS TABLE (INTEGRATED)

- The System Requirements Table represents an instantiation of the requirements diagram that may be exported and used to construct a System Requirements Document
- Once can see that this is incomplete



ain ... | Sys-Lvl UC 1 Exe Msn SD | UAV SD | CE SMD | System Cap Reqs | System Ext IF Reqs | 3.2 System Capability Re...

Add New | Add Nested | Add Existing... | Delete | Export

Criteria

Scope (optional): System Requirements | ... | Filter:

#	Id	Name	Text
1	S3.2	Provide FSS Capabilities	
2	S3.2.1	Sense Environment	The FSS shall provide surveillance data to the operator.
3	S3.2.1.1	Detect Fires	The FSS shall be a able to detect fires whose diameter exceeds 1 m in diameter in a 2 mi x 2 mi field of view within 2 seconds.
4	S3.2.1.1.1	Probability of Missed Detection	TBD
5	S3.2.1.1.2	Probability of False Detection	TBD
6	S3.2.1.2	Provide Real Time IR Video	The FSS shall be able to provide real-time IR video of a given region
7	S3.2.1.2.1	Field of View	TBD
8	S3.2.1.2.2	Resolution	
9	S3.2.1.2.3	Frame Rate	
10	S3.2.1.3	Provide Real Time Video	The FSS shall be able to provide real-time video of a given region
11	S3.2.1.4	Control Sensor	The FSS shall permit the operator to control the Sensors
12	S3.2.1.4.1	Control Sensor Orientation	
13	S3.2.1.4.2	Provide Zoom Capability	
14	S3.2.2	Transport Sensor	The FSS shall loiter over the operational area.
15	S3.2.2.1	Maximum Flight Time	
16	S3.2.2.2	Cruise Speed	
17	S3.2.2.3	Maximum Altitude	
18	S3.2.2.4	Control Flight	
19	S3.2.2.5	Take-off	
20	S3.2.2.6	Land	
21	S3.2.3	On-Station Availability	The FSS shall provide a steady state on-station availability (Aoss) of greater that 0.95.
22	S3.2.3.1	FSS Reliability	The FSS shall have a mean time between critical failure (MTBCF) greater than TBD hrs.
23	S3.2.3.1.1	Control Element Reliability	
24	S3.2.3.1.2	UAV Relliability	
25	S3.2.3.2	FSS Down Time	The FSS shall have a mean down time of less that TBD hrs.
26	S3.3	Provide External Interfaces	The FSS shall provide external interfaces.
27	S3.3.1	Provide Operator Interface	The FSS shall provide an interface to the operator.
28	S3.3.1.1	Display Surveillance Information	The FSS shall display Surveillance Information (see data dictionary).
29	S3.3.1.2	Display FSS Status Information	The FSS shall display FSS Status Information to the Operator (see data dictionary).
30	S3.3.1.3	Accept Operator UAV Commands	The FSS shall accept operator UAV Commands (see data dictionary).
31	S3.3.1.4	Accept Operator Sensor Commands	The FSS shall accept operator Sensor Commands (see data dictionary).
32	S3.3.2	Provide Maintainer Interface	The FSS shall provide an interface to the maintainer.
33	S3.3.2.1	Provide internal access to UAV	TBD
34	S3.3.2.2	Provide Fuel Interface	TBD
35	S3.3.3	Provide Environment Interface	TBD

EXERCISE

- Construct the following Stakeholder Requirements Diagrams for your project: Programmatic Requirements, Capabilities and Interface Requirements.
  - The Programmatic and Interface RDs should each have at least three leaf-node requirements.
  - The Capabilities RD should have at least 7 “leaf-node” requirements. These should include Measures of System Effectiveness.
  - Use a numbering system that will facilitate generation of a well-organized Stakeholder Requirements Table.
- Generate a Stakeholder Requirements Table that includes all your Stakeholder Requirements.

- Construct the following System Requirements Diagrams for your Project: System Capabilities (Performance and Functional), Interface Requirements, Other Requirements.
  - The Capabilities RD should decompose the requirements to the point that each leaf-node requirement can be allocated to a single logical element.
  - The Interfaces RD should have at least 3 “leaf-node” requirements.
  - The Other Requirements RD should have at least 3 leaf-node requirements.
- Generate a Stakeholder Requirements Table that includes all your Stakeholder Requirements.

BACKUP SLIDES
FROM THE INCOSE GUIDE FOR WRITING REQUIREMENTS SUMMARY SHEET

DEFINITIONS & DESIRED REQUIREMENTS CHARACTERISTICS [1]

Definitions

*An **entity** is a single thing to which a need or requirement refers: an enterprise, business unit, system, or system element (which could be a product, process, human, or organization).*

*A **need** is the result of a formal transformation of one or more concepts for an entity into an agreed-to expectation for that entity to perform some function or possess some quality (within specified constraints).*

*A **requirement statement** is the result of a **formal transformation** of one or more needs into an **agreed-to obligation** for an entity to perform some function or possess some quality (within specified constraints).*

Formal Transformation. Given the requirement is a result of a formal transformation, the following characteristics of a well-formed requirement have been derived:

- C1 – *Necessary*: The requirement defines an essential capability, characteristic, constraint, or quality factor
- C2 – *Appropriate*: The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction).
- C5 – *Singular*: The requirement should state a single capability, characteristic, constraint, or quality factor.
- C8 – *Correct*: The requirement must be an accurate representation of the entity need from which it was transformed.
- C9 – *Conforming*: The individual requirements should conform to an approved standard template and style for writing requirements.

Agreed-to Obligation. Since the requirement is to be a part of a fair agreement to meet an obligation, the following characteristics of a requirement have been derived.

- C3 – *Unambiguous*: The requirement is stated in such a way that it can be interpreted in only one way.
- C4 – *Complete*: The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement.
- C6 – *Feasible*: The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
- C7 – *Verifiable*: The requirement is structured and worded such that its realization can be proven (verified) to the customer"s satisfaction at the level the requirement exists.

INCOSE Guide to Writing Requirements Summary Sheet

DEFINITIONS & DESIRED REQUIREMENTS CHARACTERISTICS [2]

*A **set of requirements** is a **structured set of agreed-to requirement expressions** for the entity and its external interfaces documented in an Entity (Enterprise/Business Unit/ System/System Element/Process) Requirements Specification (Document). A set of requirements results from the **formal transformation** of the set of needs that represents an **agreed-to obligation** for the entity.*

Formal Transformation. Given the set of requirement is the result of a formal transformation, the following characteristics of the requirement set have been derived:

- C10 – *Complete*: The requirement set stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, and/or quality factors to meet the entity needs without needing other information.
- C11 – *Consistent*: The set of requirements contains individual requirements that are unique, do not conflict with or overlap other requirements in the set, and the units and measurement systems they use is homogeneous. The language used within the set of requirements is consistent (i.e., the same word is used throughout the set to mean the same thing).

Agreed-to Obligation. Since the set of requirements is to be a result of a fair agreement to meet an obligation, the following characteristics of the set have been derived:

- C12 – *Feasible*: The requirement set can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
- C13 – *Comprehensible*: The set of requirements must be written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part.
- C14 – *Able to be validated It must be able to be proven the requirement set will lead to the achievement of the entity needs within the constraints (such as cost, schedule, technical, legal and regulatory compliance).*

INCOSE Guide to Writing Requirements Summary Sheet

REQUIREMENTS WRITING HEURISTICS [1]

Rules for Requirement Statements and Sets of Requirements

Precision

R1 – Use definite article "the" rather than the indefinite article "a."
R2 – Use the active voice with the actor clearly identified.
R3 – Make the subject of the requirement appropriate to the layer in which the requirement exists.
R4 – Only use terms defined in the glossary.
R5 – Quantify requirements precisely – Avoid imprecise quantifiers that provide vague quantification, such as "some," "any," "several," "many," "a lot of," "a few," "approximately," "almost always," "nearly," "about," "close to," "almost," "approximate," "significant," "flexible," "expandable," "typical," "sufficient," "adequate," "appropriate," "efficient," "effective," "proficient," "reasonable."
R6 – Use appropriate units, with tolerances or limits, when stating quantities – Explicitly state units for all numbers.
R7 – Avoid the use of adverbs – words that end in -ly" Such as "usually," "approximately," "sufficiently," "typically"
R8 – Avoid the use of vague adjectives such as "ancillary," "relevant," "routine," "common," "generic" and "customary."
R9 – Avoid escape clauses such as "so far as is possible," "as little as possible," "as much as possible," "if it should prove necessary," "where possible" and "if practicable."
R10 – Avoid open-ended clauses such as "including but not limited to," "etc." and "and so on."

Concision

R11 – Avoid superfluous infinitives such as "…be designed to…"
R23 – Avoid parentheses and brackets containing subordinate text.
R24 – Enumerate sets of entities as explicit requirements instead of using generalizations.
R25 – When a requirement is related to complex behavior, refer to the supporting diagram or model rather than a complex textual description

Completeness

R26 – Avoid the use of pronouns.
R27 – Avoid using headings to support explanation of subordinate requirements.

Realism

R28 – Avoid using unachievable absolutes such as 100% reliability or 100% availability.

Conditions

R29 – State applicability conditions explicitly instead of leaving applicability to be inferred from the context.
R30 – Express the propositional nature of a condition explicitly instead of giving lists of conditions.

Uniqueness

R31 – Classify the requirement according to the aspects of the problem or system it addresses.
R32 – Express each requirement once and only once.

INCOSE Guide to Writing Requirements Summary Sheet

REQUIREMENTS WRITING HEURISTICS [2]

Concision

R11 – Avoid superfluous infinitives such as ".. be designed to...," "...be able to...," "...be capable of...."

R12 – Use a separate clause for each condition or qualification.

Non-ambiguity

R13 – Use correct grammar.

R14 – Use correct spelling.

R15 – Use correct punctuation.

R16 – Use the logical construct "X AND Y" instead of "both X AND Y."

R17 – Avoid the use of "X and/or Y."

R18 – Avoid the use of the oblique ("/") symbol except in units, i.e., Km/hr

Singularity

R19 – Write a single, simple, single-thought, affirmative, declarative sentence, conditioned and qualified by relevant sub-clauses.

R20 – Avoid combinators such as "and," "or," "then," "unless," "but," "/," "as well as," "but also," "however," "whether," "meanwhile," "whereas," "on the other hand" and "otherwise."

R21 – Use an agreed typographical device to indicate the use of propositional combinators for expressing a logical condition within a requirement.

R22 – Avoid phrases that indicate the purpose of the requirement.

problem or system it addresses.

R32 – Express each requirement once and only once.

Abstraction

R33 – Avoid stating a solution – focus on the problem "what" rather than the solution "how."

Quantifiers

R34 – Use "each" instead of "all," "any" or "both" when universal quantification is intended

R35 – Define the range of acceptable values associated with quantities.

R36 – Provide specific measurable performance targets.

R37 – Define temporal dependencies explicitly instead of using indefinite temporal keywords.

Uniformity of Language

R38 – Define a consistent set of terms to be used in requirement statements in a glossary – avoid the use of synonyms.

R39 – If acronyms are used in requirement statements, use a consistent set of acronyms.

R40 – Avoid the use of abbreviations in requirement statements.

R41 – Use a project-wide style guide.

Modularity

R42 – Group mutually dependent requirements together.

R43 – Group related requirements together.

R44 – Conform to a defined structure or template.

INCOSE Guide to Writing Requirements Summary Sheet

V&V OF REQUIREMENTS, DESIGN, & THE SYSTEM

Verification and Validation

"Verification" and "validation" are very ambiguous unless preceded by a modifier that clearly indicates what concept the term is referring to. When using these terms, it should be clear as to which concept is being referred to: requirement verification or requirement validation; design verification or design validation; system verification or system validation. The following definitions of these terms are included in terms of a product life cycle:

Requirement Verification: the process of ensuring the requirement meets the rules and characteristics defined for writing a good requirement. The focus is on the wording and structure of the requirement. "Is the requirement worded or structured correctly?" in accordance with the organization"s standards, processes, and checklists.

Requirement Validation: confirmation the requirement is an agreed-to transformation that clearly communicates the stakeholder needs and expectations in a language understood by the developers. The focus is on the message the requirement is communicating. "Does the requirement clearly and correctly communicate the stakeholder expectations and needs?" "Are we doing the right things?" or "Are we building the right thing?.

Design Verification: the process of ensuring the design meets the rules and characteristics defined for the organization"s best practices associated with design. The focus is on the design process "Did we follow our organizations guidelines for doing design correctly?" The design process also includes ensuring the design reflects the design-to requirements. Thus, design verification is also a confirmation the design is an agreed-to transformation of the design-to requirements into a design that clearly implements those requirements correctly. "Does the design clearly and correctly represent the requirements?" "Did we design the thing right?"

Design Validation: confirmation the design will result in a system that meets its intended purpose in its operational environment. Will the design result in a system that will meet the stakeholder expectations (needs) that were defined during the scope definition phase that should have occurred at the beginning of the project? The focus is on the message the design is communicating. "How well does the design meet the intent of the requirements.?" "Do we have the right design?" or "Are we doing the right things?" "Will this design result in the stakeholder expectations and needs being met?"

System Verification: System Verification is done after design and build, making sure the designed and built system meets its requirements. The focus is on the built system and how well it meets the agreed-to requirement set that drove the design and fabrication. Methods used for system verification include: test, demonstration, inspection, or analysis. Did we build the thing right?"

System Validation: System validation occurs after system verification and makes sure the designed, built, and verified system meets its intended purpose in its operational environment. The focus is on the completed system and how well it meets stakeholder expectations (needs) that were defined during the scope definition phase that should have occurred at the beginning of the project. "Did we build the right thing?"

INCOSE Guide to Writing Requirements Summary Sheet