Engineering Systems with Standards and Digital Models: Specifying Stakeholder Needs and Capabilities - MGOS

Kevin MacG. Adams *, Irfan Ibrahim, Steven L. Krahn

Posted Date: 6 March 2026

doi: 10.20944/preprints202603.0509.v1

Keywords: systems engineering; model-based systems engineering; needs and capabilities; requirements; standards; MGOS; mission; goals; objectives

 doi:10.20944/preprints202603.0509.v1

Article

Engineering Systems with Standards and Digital Models: Specifying Stakeholder Needs and Capabilities - MGOS

Kevin MacG. Adams [1,*], Irfan Ibrahim [2] and Steven L. Krahn [3]

[1] Research engineer, Department of Civil & Environmental Engineering, Vanderbilt University, Nashville, TN, 37235, USA

[2] Graduate research assistant, Department of Civil & Environmental Engineering, Vanderbilt University, Nashville, TN, 37235, USA

[3] Professor of the Practice, Department of Civil & Environmental Engineering, Vanderbilt University, Nashville, TN, 37235, USA

* Correspondence: kevin.adams@vanderbilt.edu; +1-207-977-3454

Abstract

This paper proposes a formal method and associated techniques for completing the ISO/IEC/IEEE Standard 15288 technical process 6.4.2 – stakeholder needs and requirements definition within the 15288-SysML grid framework. The paper is a companion work to *Engineering systems with standards and digital models: Development of a 15288-SysML Grid*, which describes an engineering design method that supports the tenets of the Industry 4.0 paradigm. The formal method presented here is grounded using established constructs from systems science; specifically, the systems principles of hierarchy, emergence, requisite parsimony, minimum critical specification, and requisite saliency. The application of accepted principles ensures that stakeholders are able to objectively specify measurable criterion that can satisfy stakeholder needs and capabilities. The method uses: (1) international standards for systems (e.g., ISO/IEC/IEEE 15288); (2) adopts the four fundamental aspects of system design supported by model-based systems engineering (MBSE); (3) invokes the international standard for the systems modeling language (SysML); and (4) adopts a hierarchical requirements tree that specifies Mission, Goals, Objectives, and Sub-objectives (MGOS) to provide the stakeholder-analysis process for the articulation of system-level engineering requirements. Utilization of the MGOS is intended to have a positive impact on the system design process by ensuring reproducibility, replicability, transparency, and generalization.

Keywords: systems engineering; model-based systems engineering; needs and capabilities; requirements; standards; MGOS; mission; goals; objectives

1. Introduction

This paper proposes a formal method and associated techniques for completing the ISO/IEC/IEEE Standard 15288 technical process 6.4.2 – stakeholder needs and requirements definition through the utilization of the 15288-SysML grid. The paper is a companion work to *Engineering systems with standards and digital models: Development of a 15288-SysML Grid* [1] which describes an engineering design method that supports the tenets of the Industry 4.0 paradigm.

The formal method and associated techniques presented herein are grounded using established constructs from systems science; specifically, the systems principles of hierarchy, emergence, requisite parsimony, minimum critical specification, and requisite saliency [2–4]. The application of accepted principles ensures that stakeholders are able to objectively specify measurable criterion that can satisfy stakeholder needs and capabilities and serve as the starting point for the development of system-level requirements.

 doi:10.20944/preprints202603.0509.v1

The method utilizes: (1) international standards for systems (e.g., ISO/IEC/IEEE 15288); (2) adopts the four fundamental aspects of system design supported by model-based systems engineering (MBSE) [5]; (3) invokes the international standard for the systems modeling language (SysML) [6] and (4) adopts a hierarchical requirements tree that specifies Mission, Goals, Objectives, and Sub-objectives (MGOS) to provide the stakeholder analysis process for the articulation of system-level engineering requirements.

The use of the terms *method* and *technique* are purposeful, where the terms are defined as: (1) *method* is "A systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art" [7] (p. 781) and (2) *technique* is "A body of technical methods (as in a craft or in scientific research)" [7] (p. 1283).

Using these definitions, the method described includes several unique techniques. The techniques that will be emphasized are: (1) the use of model-based systems engineering (MBSE), with particular emphasis on the Systems Modeling Language, SysML [6]; and (2) the development of stakeholder needs and capabilities using a systems-based framework that expresses the needs in an MGOS hierarchy. The combination of these techniques takes advantage of SysML's unique design aspects and perspectives that enable system relationships to be graphically depicted and related through their behavior, requirements, structure, and parametric relationships.

Specifically, the method has been constructed such that a practitioner can use the *15288-SysML Grid* framework to identify the process outcome(s) of interest, verify the design aspect, and then select the SysML technique and appropriate diagram type (e.g., ***uc, pkg, req, par***, etc.)[1] indicated in the grid. Following this technique supports the MBSE approach to system life cycle development, while also adhering to the technical processes outlined in the 15288 standard. Use of the grid, in conjunction with a formal life cycle model and international standards, is intended to have a positive impact on the system design process. Specifically, the improvement of the engineering design process through reproducibility, replicability, transparency, and generalization.

Reproducibility. In this sense, the method and supporting techniques are addressing computational reproducibility where the design process is "obtaining consistent computational results using the same input data, computational steps, methods, and code, and conditions of analysis"[8](p. 1).
Replicability. "Obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data" [8](p. 1).
Transparency. "A transfer of information from an autonomous system or its designers to a stakeholder that is truthful; contains information relevant to the causes of some action, decision, or behavior; and is presented at a level of abstraction and in a form meaningful to the stakeholder. Transparency should be mindful of the stakeholders' likely perception and comprehension, and should avoid disclosing information in a manner that, while technically true, is framed in a way that leads to misapprehension" [9] (p. 15).
Generalizability. "Refers to the extent that results of a study apply in other contexts or populations that differ from the original one" [8](p. 1).

This paper is organized into four major sections. After the introduction in the first section, the second section provides background on where the method and associated techniques fit within the larger life cycle model utilized for engineering systems. The third section describes the formal method for stakeholder needs and requirements definition, presenting the method and the supporting techniques utilized in producing reliable, accurate, and measurable stakeholder needs and capabilities. The section includes examples that presents the development of stakeholder needs and capabilities using a systems-based framework that expresses the needs in a MGOS hierarchy using MBSE and the SysML language. The fourth and final section summarizes the method and supporting techniques and their value added to the systems engineering literature.

[1] The nine SysML diagram types and their abbreviations are described in Table .

 doi:10.20944/preprints202603.0509.v1

2. Background

The design, or more exactly, *the engineering of,* modern complex systems is accomplished through formal engineering design processes. The seven sub-sections that follow will provide an overview and high-level description of how complex systems are engineered and will: (1) expose the reader to a focused definition for *engineering;* (2) provide an overview of the basic high-level engineering design process and the discipline of systems engineering; (3) introduce the use of international systems engineering standards for implementation in the acquisition and management of systems; (4) present a modern *system life cycle model* that encapsulates and constrains the system endeavors that flow from concept to retirement and disposal in direct support of Industry 4.0; (5) describe a life cycle process model that includes a hierarchy of processes, activities, and tasks that need to be performed to successfully accomplish the ISO/IEC/IEEE 15288 technical processes defined in the life cycle model; (6) review the technical process (6.4.2) for stakeholder needs and requirements definition process; and concludes with (7) a high-level description of how the 15288-SysML Grid may be used to address invoke the 15288 technical processes utilizing the specific methods and techniques in MBSE and SysML.

2.1. Engineering Definition

Engineering is "the science by which the properties of matter and the sources of power in nature are made useful to humans in structures, machines, and products" [10] (p. ix). Theodore von Kármán [1881-1963], the aeronautical engineering pioneer and co-founder of Cal Tech's Jet Propulsion Laboratory, has been credited with the following statement:

The scientist seeks to understand what is. The engineer seeks to create what never was. [11] (p. 20).

Von Kármán was able to simply state what many committees, boards, and regulators have tried to do in defining engineering. The root for the word engineering is derived from the Latin *ingenium*, which means innate or natural quality. Engineering has many definitions. One of the more comprehensive and thoughtful has been assembled by the historians of engineering.

"The art of the practical application of scientific and empirical knowledge to the design and production or accomplishment of various sorts of constructive projects, machines, and materials of use or value to man" [12] (p. 2).

The application of empirical knowledge during the design of modern, interconnected, socio-technical systems has become a complex en[1881–1963deavor resulting in a new class of system labeled an *engineering system*, formally defined as:

"A class of systems characterized by a high degree of technical complexity, social intricacy, and elaborate processes, aimed at fulfilling important functions in society" [13] (p. 31).

These characteristics of engineering systems require the use of formal methods. The next section will address the engineering design process and the sub-discipline of systems engineering.

2.2. Engineering Design and Systems Engineering

Penny [14] discusses the principles of engineering design, using Dixon's [15] perspective, and as a starting point for a discussion about the process of engineering design. The process that we label *engineering design* has evolved throughout time, as human beings have constructed larger systems of increasing complexity. The simple schematic in Figure may be used to capture the essential process of engineering design.

 doi:10.20944/preprints202603.0509.v1

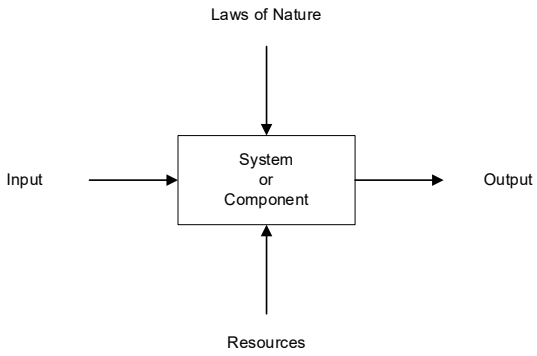


Figure 1. Schematic of engineering design process [adapted from Figure 1.3 in [15], p. (12)].

The elements in Figure are described in terms of one another and the processes used to approach their solution in Table .

Table 1. Description of elements in the engineering design process.

Element	Solve for	Process
Input, laws of nature, system or component	Output	Analysis (i.e., deduction)
Output, laws of nature, system or component	Input	Inverse analysis (i.e., reverse engineering)
Input, output, system or component	Laws of nature	Science (i.e., induction)
Input, output, laws of nature	System or component	Engineering design

In order to successfully accomplish the processes in Table , engineering disciplines (e.g., mechanical, electrical, software, etc.) must adopt and apply methods and techniques to successfully accomplish any of the myriad of activities associated with the design of engineering systems and their sub-systems and components. The sub-discipline within the broader engineering community that addresses the special class of system, engineering systems, is *systems engineering*. Systems engineering's definition is:

Systems engineering: "Interdisciplinary approach governing the total technical and managerial effort required to transform a set of stakeholder needs, expectations, and constraints into a solution, and to support that solution throughout its life" [16] (p, 457).

To achieve uniform methods and techniques for systems engineering, the international systems engineering community has created and adopted a number of standards to "derive rules for the development of these systems from the system elements and their relationships" [17] (p. 9). The next section will discuss the international standards for systems engineering.

2.3. International Standards for Systems Engineering

Formal systems engineering practices have been developed to define a set of processes that can be used to help establish a technical template and standardized guidance for the life cycle endeavors associated with systems. These practices are contained in a series of international standards. The standards are developed, issued, and maintained by a tripartite of standards organizations: (1) the International Organization for Standardization - ISO; (2) the International Electrotechnical Commission - IEC; and (3) the Institute of Electrical and Electronics Engineers - IEEE. Table 1 is a partial listing of the international standards relevant to engineering systems.

Table 1. Partial listing of relevant international standards for the life cycle of systems.

Standard Number	Year	Description and reference
ISO/IEC/IEEE 15026-1	2019	Part 1: Concepts and vocabulary [18]
ISO/IEC/IEEE 15026-4	2021	Part 4: Assurance in the life cycle [19]
ISO/IEC/IEEE 15288	2023	System life cycle processes [20]

 doi:10.20944/preprints202603.0509.v1

ISO/IEC/IEEE 15289	2019	Content of life cycle information product (documentation) [21]
ISO/IEC/IEEE 15939	2017	Measurement process [22]
ISO/IEC/IEEE 16085	2021	Life cycle processes — Risk management [23]
ISO/IEC/IEEE 16326	2019	Life cycle processes — Project management [24]
ISO/IEC/IEEE 24641	2023	Methods and tools for model-based systems and software engineering [25]
ISO/IEC/IEEE 24748-1	2018	Part 1: Guidelines for life cycle management [26]
ISO/IEC/IEEE 24748-2	2018	Part 2: Guide to the application of ISO/IEC 15288 [27]
ISO/IEC/IEEE 24748-4	2016	Part 4: Systems engineering planning [28]
ISO/IEC/IEEE 24748-6	2023	Part 6: Life cycle management [29]
ISO/IEC/IEEE 24748-7000	2022	Part 7: Process for addressing ethical concerns during system design [30]
ISO/IEC/IEEE 24765	2017	Vocabulary [31]
ISO/IEC/IEEE 24774	2021	Life cycle management — Guidelines for process description [32]
ISO/IEC/IEEE 29148	2018	Life cycle processes — Requirements engineering [33]
ISO/IEC/IEEE 42010	2022	Architecture description [34]
ISO/IEC/IEEE 42020	2019	Architecture processes [35]
ISO/IEC/IEEE 42030	2019	Architecture evaluation framework [36]

In this context, a standard is defined as "A document that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context" [37] (p. 562). The standards in Table 1, while not prescriptive, provide high-level guidance to organizations and practitioners in order to provide improved communication and understanding among stakeholders involved with any element of a system endeavor, from a system's concept development, its operation and maintenace, and on to retirement and disposal. Thus, it is important to study, understand, and use standards because they provide a common framework to "design flexible, adaptable, robust systems that can be easily modified and reconfigured to satisfy changing requirements and new technological opportunities" [13] (p. 31).

The next section will discuss how a system life cycle model is utilized as the framework for a system that start with stakeholder concepts and complete with retirement and disposal of the system.

2.4. System Life Cycle Model

A system's life cycle begins with the establishment of a unique need identified by a group of stakeholders. In this context, *stakeholders* are defined as:

Stakeholder: "1. individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations; 2. an individual, group, or organization who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project; 3. individual, team, organization, or classes thereof, having an interest in a system; 4. individual, group or organization that can affect, be affected by, or perceive itself to be affected by, a risk. EXAMPLE: end users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, supplier organizations and regulatory bodies, interested parties, decision-makers Note 1 to entry: Some stakeholders can have interests that oppose each other or oppose the system" [16] (p. 435).

Once there is an identified need for a system, as defined by stakeholder(s), a life cycle model is selected and used to guide the overall life cycle of the proposed system. A *system life cycle model* is defined as the "framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding" [16] (p. 251). Life cycle models typically include a number of unique *stages*, which are defined as a "period within the life cycle of an entity that relates to the state of its description or realization" [16] (p. 435). The type of life cycle model may vary depending upon the domain for the system-of-interest (SoI), but every life cycle model is a depiction of the flow of stages from needs identification (i.e., conception) through the system's ultimate retirement and disposal. Life cycle models are differentiated by the number and types of stages and how their flows are represented in the

 doi:10.20944/preprints202603.0509.v1

movement from stage-to-stage. Typical life cycle models include waterfall models, spiral models, Vee models, iterative models, etc. [38].

The Boeing Corporation developed an advanced life cycle model, termed the *Boeing Diamond*© [39], to take advantage of the tenets in the modern digital enterprise and Industry 4.0. Figure is a depiction of the *Articulated diamond life cycle model* [1] which presents the life cycle relationships described in the traditional Vee life cycle process model with and the concepts introduced in the digital enterprise.

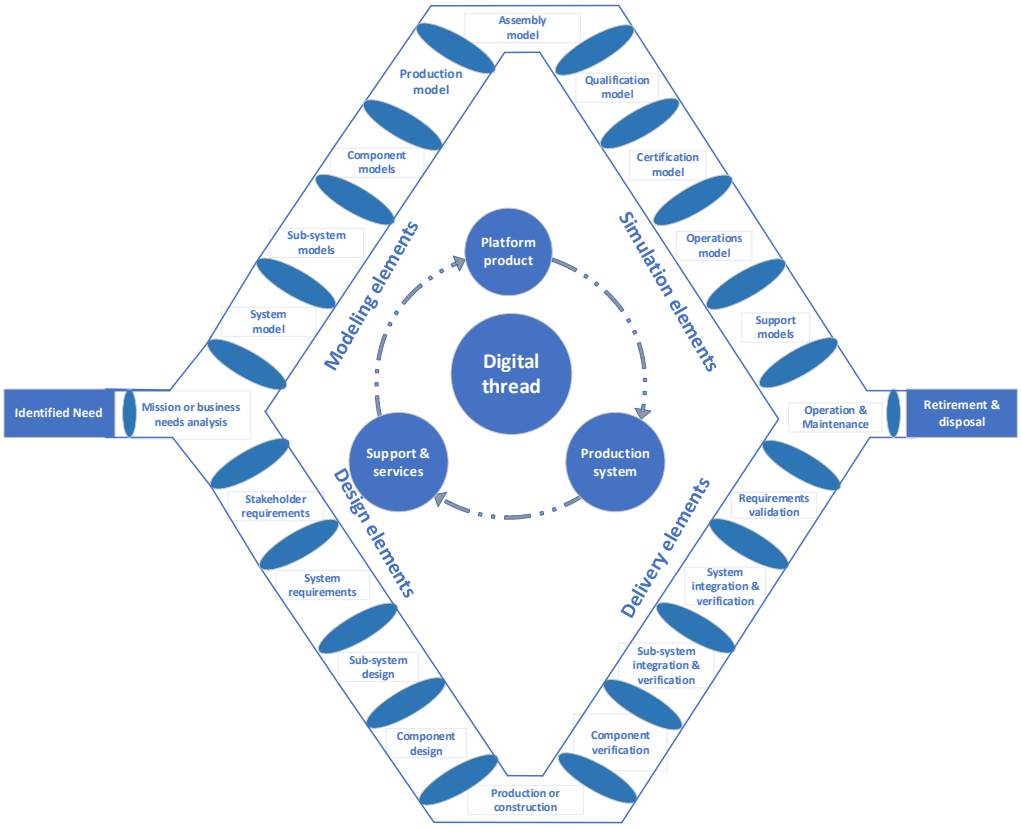


Figure 2. Articulated diamond life cycle [1].

The primary function of a life cycle model is to serve as the overarching structure or framework that will be used to describe the movement of a system, stage-by-stage, from the stakeholder identified needs (i.e., during conception) on to follow-on stages of the life cycle. Each stage in the life cycle model contains formal processes, frequently described in the standards, which specify the inputs, outputs, and outcomes necessary for successful accomplishment of the stage.

The next section will discuss how a system life cycle model for systems is created by adding the processes from the standards in Table 1 to a life cycle model.

2.5. Life Cycle Process Model

The addition of formal processes to a life cycle model enables the transition to a *life cycle process model*. The *life cycle process model* describes the specific processes that must be performed to successfully accomplish the stages defined in the life cycle of the system.

The stages within life cycle model are accomplished through the planning and execution of dedicated processes. The international community has specified the processes for the stages of the systems life cycle in the international standards presented earlier in Table 1. Each of the international standards include the processes and an associated hierarchy of activities and tasks that provide the inputs, outputs and outcomes required to successfully accomplish the life cycle processes [40]. The elements of the hierarchy specified in ISO/IEC/IEEE Standard 15288, Systems and software engineering — System life cycle processes [20] are defined as follows:

 doi:10.20944/preprints202603.0509.v1

- Process: “Set of interrelated or interacting activities that transforms inputs into outputs” [16] (p. 337).
- Activity: “set of cohesive tasks of a process” [16] (p. 9).
- Task: “Required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process” [16] (p. 460).

ISO/IEC/IEEE Standard 15288 [41] outlines the life cycle processes and the associated activities, and tasks required to successfully complete each of the life cycle stages. Figure depicts the primary 15288 technical processes and some of the supporting international standards.

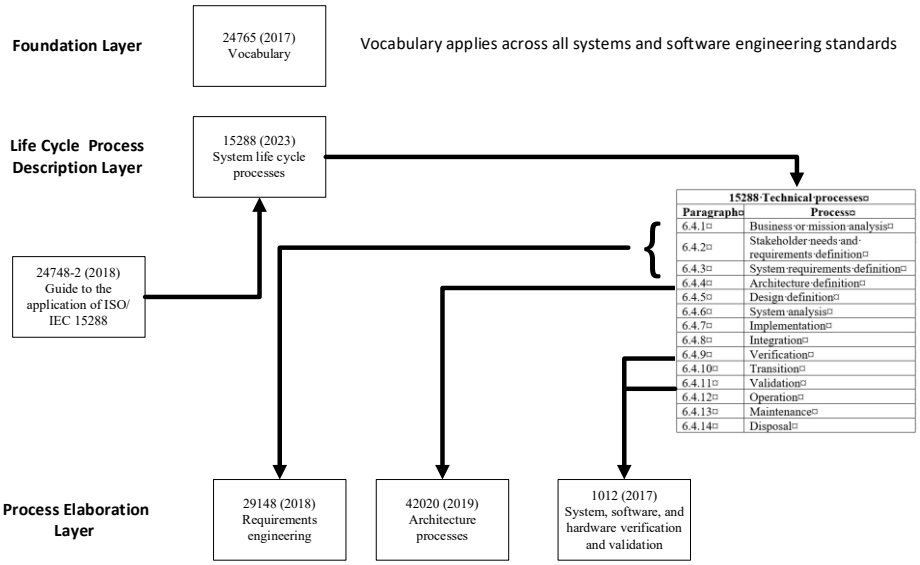
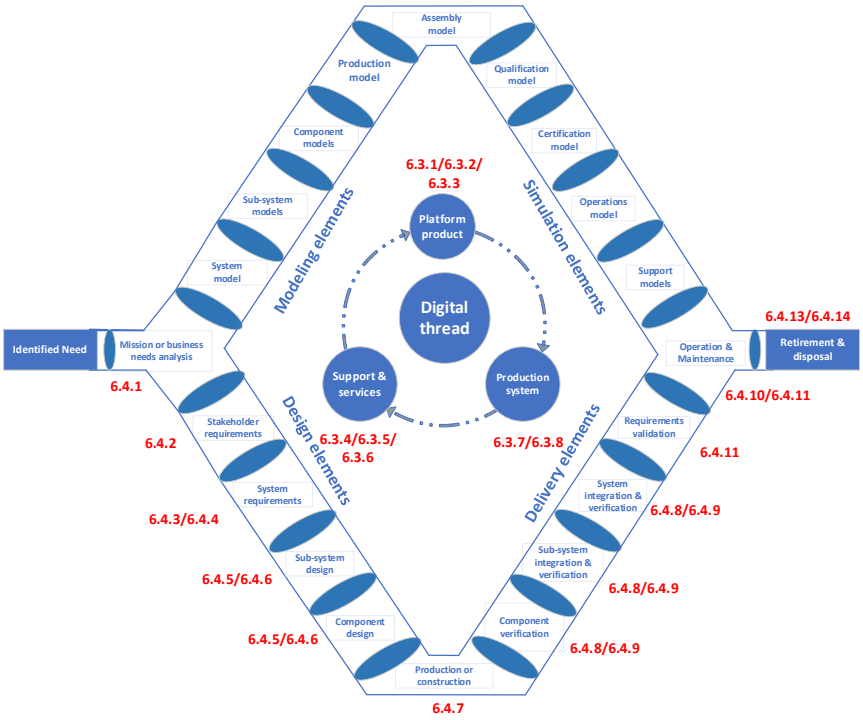


Figure 3. Interrelationships between ISO/IEC/IEEE standards and 15288 technical processes [1] (p. 11).

Each of the 15288 technical processes presented in the block in Figure and annotated in red in), are added to the *Articulated diamond life cycle model*, that the authors have previously introduced [1]. Figure includes the technical processes as part of the system life cycle model. where the annotations in red refer to the unique technical processes.



 doi:10.20944/preprints202603.0509.v1

Figure 4. Articulated diamond life cycle process model [1] (p. 13).

Once the Life Cycle Process Model has been selected, and the 15288 technical processes are defined, additional organization- or discipline-specific methods and techniques are invoked to execute the processes, activities and tasks specified in the 15288 technical processes.

The challenge is to ensure that there is enough detail for an organization's personnel to understand and successfully execute the goals of the systems endeavor. Figure is a depiction of how to approach the construction of a tailored system life cycle process model.

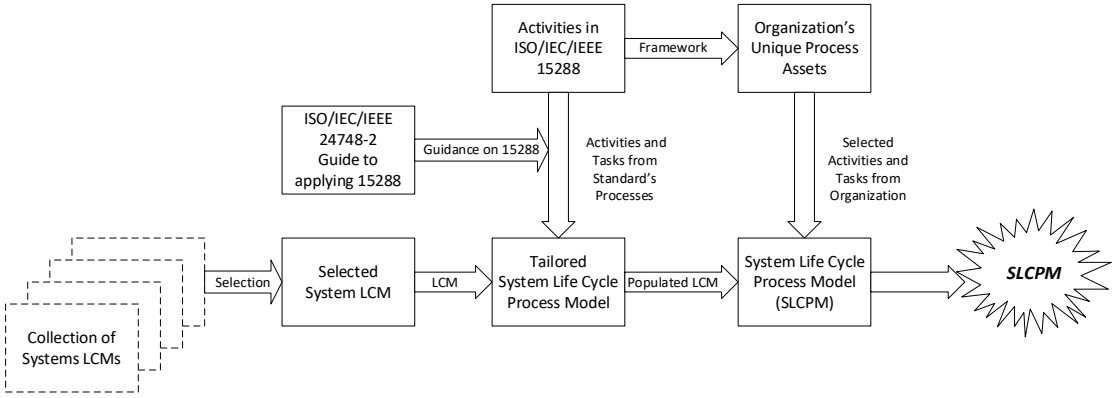


Figure 5. Building a Tailored Systems Life Cycle Process Model (SLCPM).

The next section will address the how the technical processes of 15288 are utilized in the early stage of the development process for an engineering system.

2.6. Stakeholder Identification of Needs and Capabilitities

Once there is an identified need for a system, the conceptual design stage of the system's life cycle commences. The first design element, business or mission analysis (technical process 6.4.1), defines "the overall strategic problem or opportunity, characterize the solution space, and determine potential solution class(es) that can address a problem or take advantage of an opportunity" [41] (p. 59). Once the business need is clarified (i.e., the second design element in Figure), stakeholder needs and requirements definition commences.

The purpose of the stakeholder needs and requirements definition process (technical process 6.4.2) is:

> "to define the stakeholder requirements for a system that can provide the capabilities needed by users and other stakeholders in a defined environment. It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs. It analyzes and transforms these needs into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational capability is validated. The stakeholder requirements are defined considering the context of the SoI, which includes the interoperating systems and enabling systems. This also includes consideration of laws and regulations, environmental restrictions, and and ehtical values" [41] (p. 62).

It is important to note that Section 6.4.2 of 15288 is supplemented by ISO/IEC/IEEE Std 29148 – Requirements engineering [42], which includes additional processes, activities, and tasks which support the development and specification of stakeholder needs and requirements. Figure is a depiction of the requirements engineering process. Note that the focus of the discussion in ISO/IEC/IEEE-29148 is limited to *elicitation and derivation* of stakeholder needs and capabilities *within the problem domain*, by which we mean "area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area" [16] (p. 145).

 doi:10.20944/preprints202603.0509.v1

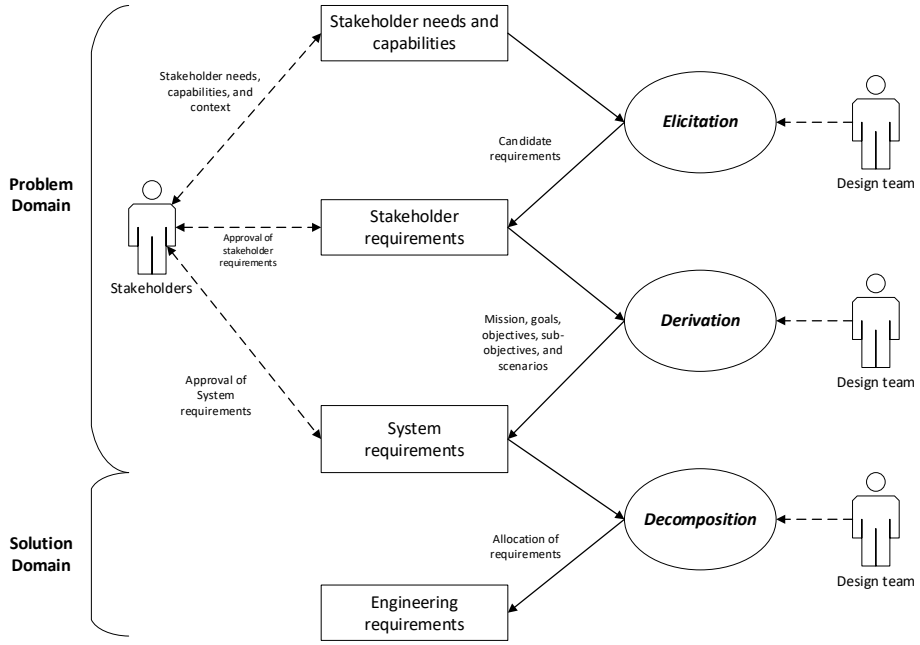


Figure 6. Stakeholder needs and capabilities and system-level engineering requirements process.

The next section will show how the stakeholder needs and requirements definition process (technical process 6.4.2) can be executed through the application of model based systems engineering (MBSE) and the International Standard for the Systems Modeling Language (SysML) [6].

2.7. Technical Process 6.4.2 Execution using MBSE and SysML

This sub-section will include both a brief introduction to the *15288-SysML grid* and a high-level description of the nine diagram types used in SysML.

2.7.1. The 15288-SysML Grid

The *15288-SysML grid* described in [1] depicts 15288 technical process(es) on the vertical axis and the four fundamental aspects of system design, as invoked in SysML (i.e., behavior, requirements, structure, and parametric relationships) on the horizontal axis. The combination of the two axes result in a Zachman [43,44] type grid where the resultant inner cells reveal the relevant SysML techniques and diagram types that may be utilized to support the realization of the process outcome.

The intersection point on the grid is where the specific methods and techniques utilized in MBSE and SysML are specified. Table is an example of how the *15288-SysML Grid* may be used to address two of the stakeholder needs and requirements definition process (15288 process 6.4.2) tasks.

Table 3. 15288 grid example that addresses two 6.4.2 technical process tasks.

Technical Process 6.4.2 Stakeholder Needs and Requirements Definition	**Four design aspects**			
	Behavior	**Requirements**	**Structure**	**Parametric**
Task a.1: Identify the stakeholders who have an interest in the solution throughout its life cycle.	Stakeholders for the system. ***uc.***		Structure of system. p***kg.***	
Task e.2: Define critical performance measures and quality characteristics that enable the assessment of technical achievement.		Stakeholder mission, goals, objectives, and sub-objectives. ***req***		Critical operational issues (COI) are linked to goals. ***par.***

While Table does not include all of the 15288 process 6.4.2 activities and tasks, it serves to illustrate how 15288 and the methods and techniques contained within SysML design aspects and nine associated SysML diagram types are utilized to satisfy the process. Appendix A provides an explication of how the *15288-SysML grid* addresses each of the six (6) activities and twenty-three (23) tasks in technical process 6.4.2.

 doi:10.20944/preprints202603.0509.v1

2.7.2. SysML Diagram Types and Utilization

In this section, each of the SysML diagram types will be explained, at a high-level, to permit the reader to understand how SysML provides the diagrams required to effectively model all facets of a system, its internal interactions with sub-systems, and its external interactions with the environment. The diagram explanation will be using visual aids from the International Standard for the Systems Modeling Language (SysML) [6] which is the internationally accepted standard for the OMG SysML.

The four fundamental aspects of system design and the SysML diagram types are:

1. Structure. Block definition diagrams (#2), internal block diagrams (#3), and package diagrams (#4) represent the system structure.
2. Behavior. Use case diagrams (#9), activity diagrams (#1), sequence diagrams (#7), and state machine diagrams (#8).
3. Requirements. Requirements diagrams (#6) provide a bridge between the typical requirements management tools and the system models.
4. Parametrics. Parametric diagrams (#5) represent constraints on system property values as a mean to integrate the specification and design models with engineering analysis models.

The nine SysML diagrams types that invoke the four fundamental aspects of system design are presented in Table .

Table 4. Nine SysML diagram types and typical utilization.

Diagram Type	Typical Utilization
1. Activity ***(act)***	"Emphasizes the inputs, outputs, sequences, and conditions for coordinating other behaviors. It provides a flexible link to blocks owning those behaviors" [6]. In SysML activities can (1) enable actions to start; (2) may disable actions; and (3) may restrict the rate at which entities flow.
2. Block definition ***(bdd)***	"Define features of blocks and relationships between blocks such as associations, generalizations, and dependencies. It captures the definition of blocks in terms of properties and operations, and relationships such as a system hierarchy or a system classification tree" [5]
3. Internal block ***(ibd)***	"Captures the internal structure of a block in terms of properties and connectors between properties. A block can include properties to specify its values, parts, and references to other blocks. Ports are a special class of property used to specify allowable types of interactions between blocks" [5].
4. Package ***(pkg)***	"Represents the organization of a model in terms of packages that contain model elements" [43].
5. Parametric ***(par)***	"Describes the constraints among the properties associated with blocks. This diagram is used to integrate behavior and structure models with engineering analysis models such as performance, reliability, and mass property models" [5].
6. Requirement ***(req)***	"Specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve. SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements. The requirements diagram described in this clause can depict the requirements in graphical, tabular, or tree structure format. A requirement can also appear on other diagrams to show its relationship to other modeling elements. The requirements modeling constructs are intended to provide a bridge between traditional requirements management tools and the other SysML models" [6].
7. Sequence ***(sd)***	"The Sequence diagram describes the flow of control between actors and systems (blocks) or between parts of a system. This diagram represents the

 doi:10.20944/preprints202603.0509.v1

	sending and receiving of messages between the interacting entities called lifelines, where time is represented along the vertical axis. The sequence diagrams can represent highly complex interactions with special constructs to represent various types of control logic, reference interactions on other sequence diagrams, and decomposition of lifelines into their constituent parts" [6].
8. State machine (***stm***)	"Models discrete behavior through finite state transition systems. The state machine represents behavior as the state history of an object in terms of its transitions and states" [5].
9. Use-case (***uc***)	"The use case diagram describes the usage of a system (subject) by its actors (environment) to achieve a goal, that is realized by the subject providing a set of services to selected actors" [5].

A few of the nine SysML diagram types will appear in the next major section, where the details of the formal, SysML-based method and associated techniques for the definition of stakeholder needs and requirements in an engineering system are explained.

3. Formal Method for Stakeholder Needs and Requirements Definition

The conceptual design stage of the system life cycle is the time period during which the stakeholders' needs are identified and system concepts and capabilities are explicitly described and analyzed.

The specification of stakeholder needs and capabilities *establishes the foundation for all follow-on processes in the system's life cycle*. The stakeholder's needs and capabilities for a system should be clearly expressed in terms that communicate the desired condition and outputs that the system must produce in order to satisfy the stakeholders. These ARE NOT requirements, but capabilities that the proposed systems will possess in the fulfillment of the identified needs. Needs and capabilities are expressed in the language of the stakeholder, and ARE NOT necessarily written in the language or syntax of typical engineering requirements.

The stakeholders define the needs of the system and the desired capabilities. It is important to note that these specify what capabilities the system should provide in meeting the stakeholders' needs within the unique problem domain. As such, the needs and capabilities are written using the non-technical language of the stakeholders and should avoid any allusion to how the system would, could, or should provide the capabilities in fulfilling the desired needs. Section 6.4.2 of ISO/IEC/IEEE 15288 [41] describes the process for capturing the needs and capabilities that the system must perform.

"Defining requirements begins with stakeholder needs (or goals, or objectives) that are refined and evolve before arriving as valid stakeholder requirements. Initial stakeholder concerns do not serve as stakeholder requirements, since they often lack definition, analysis and possibly consistency and feasibility" [42] (p. 10).

The output of this process is the stakeholder requirements specification (StRS), which is expressed using the language and terminology most familiar to the stakeholders. The StRS is formally defined as a "structured collection of the requirements [characteristics, context, concepts, constraints and priorities] of the stakeholder and the relationship to the external environment" [42] (p. 6). It is unfortunate that the international standards refer to the stakeholder requirements specification (StRS) using the term *requirements*. Stakeholders have needs and capabilities and a "a requirement is an expression of one or more particular needs in a very specific, precise and unambiguous manner" [42] (p. 5).

Six 15288 technical activities must be completed to successfully define stakeholder needs and capabilities process in 15288 (6.4.2): (1) prepare for stakeholder needs and requirements definition; (2) develop the operational concept and other life cycle concepts; (3) define stakeholder needs; (4) transform stakeholder needs into stakeholder requirements; (5) analyze stakeholder needs and

 doi:10.20944/preprints202603.0509.v1

requirements; and (6) manage stakeholder needs and requirements. The sections that follow will describe how these activities are accomplished through techniques contained invoked by MBSE through SysML.

3.1. Prepare for Stakeholder Needs and Requirements Definition

This activity must identify the stakeholders who have an interest in the system, from conception through retirement and disposal, define a plan where all stakeholders are included in the definition of capabilities and needs, and identify and gain access to enabling systems required to support the envisioned system.

Typically, organizations utilize a Concept of Operations (ConOps) document that states the high-level needs the system will provide, its relationship to other systems and interfaces, and the concept for how it will be used during operation. This type of document has existed in many forms during systems and software development endeavors and can be traced to documents such as the United States Department of Defense's Operations Concept Description (OCD) contained in DI-IPSC-81438 [45] and the United States Department of Energy's Mission Needs Statement (MNS) [46].

In all cases, the ConOps document serves to bound the operating space of the proposed system, how it relates to existing or proposed systems, the environment within which it will operate, and the activities (e.g., maintenance, technical refresh, etc.) required to continue the system's operation throughout its proposed life cycle. This ConOps provides the system's stakeholders with information that permits them to construct a series of supporting goals and objectives to ensure compliance with the functional and non-functional needs and capabilities in the ConOps. For example, system availability is a non-functional need that mandates specific capabilities within the system to ensure that the technical measure may be met. Failure to address this during the conceptual design for the system may lead to unrealistic estimates in both time and cost.

The technique within SysML used to capture this information is to construct a use-case diagram (***uc***), which typically covers many scenarios and may be used in conjunction with a block definition diagram ***(bdd)*** where the system and its external boundaries are represented. It is important to emphasize that no paper products are produced in this activity, as all of the data and information are contained within the SysML model diagrams.

3.2. Develop the Operational Concept and Other Life Cycle Concepts

This activity must define the context of use, within the ConOps, for the system. The context of use is typically achieved through the use of scenarios, where the operational environment and intended users are characterized, and interfaces to and from external systems are identified.

SysML includes a use-case diagram that can be used to depict stakeholders (e.g. actors), scenarios, and their relationships. A use-case is formally defined as "a sequence of tasks that a system can perform, interacting with users of the system and providing a measurable result of value for the user" [16] (p. 494). Development of a use case diagram "is conducted with the Operational Stakeholders when working with the stakeholders, to determine their overall Uses, Goals and relationships with Users/Actor (and Organization) and External Systems (system functions)" [47] (p. 15)

The use-case diagram ***(uc)*** captures the envisioned operations and users for the system, as well as its interfaces to and from external systems. The ***uc*** may be used in conjunction with a ***bdd*** to represent the system scenarios and its relationship to external boundaries.

Once again, it is important to remember that no paper products are produced in this activity, as all of the data and information are contained within the SysML model diagrams. An example of a SysML *uc* diagram is shown below in Figure , where several stakeholders for a hybrid sports utility vehicle (SUV) are defined, along with the high-level functions that each of the stakeholders perform with respect to the SUV system. The stakeholders in this example are depicted in the ***uc*** as *actors*, which are represented in SysML with schematic human figures. The stakeholders and their functions with respect to the hybrid SUV example are as follows:

 doi:10.20944/preprints202603.0509.v1

- Driver: Operate the vehicle
- Registered Owner: Insure the vehicle and register the vehicle
- Maintainer: Maintain the vehicle
- Insurance Company: Insure the vehicle
- Department of Motor Vehicles: Register the vehicle

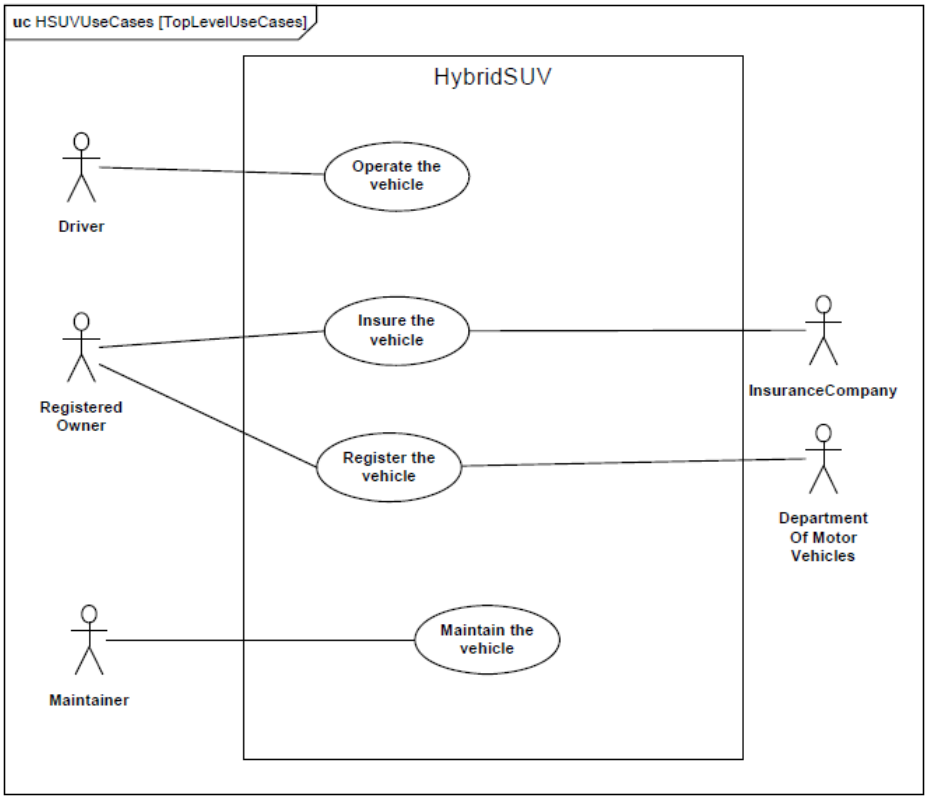


Figure 7. Use-case diagram (*uc*) for a sport utility vehicle [6] (p. 218).

3.3. Define Stakeholder Needs

The stakeholders use the scenarios in the ***uc*** diagram and the constraints in the ***bdd*** diagram as a principal sources in articulating the needs and capabilities for the proposed system. The needs and capabilities are specified in a cascading, interrelated, hierarchic structure of: (1) mission; (2) goals; (3) objectives; and (4) sub-objectives (i.e., MGOS), that permit the stakeholders to objectively specify measurable criterion that can satisfy the system's needs and capabilities.

The requirements diagram (***req***) is used by SysML to capture stakeholder needs and capabilities. The ***req*** "Specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve. SysML provides modeling constructs to represent text-based requirements and relate them to other modeling elements. The requirements diagram can depict the requirements in graphical, tabular, or tree structure format. The requirements modeling constructs are intended to provide a bridge between traditional requirements management tools and the other SysML models" [6] (p. 157).

The foundation for the development of the ***req*** are as follows:

3.3.1. Theoretical Construct for the MGOS Hierarchy

The construct for the MGOS hierarchy is directly supported by five design-related principles from systems theory [2–4], which are briefly described below.

Hierarchy: This is a central principle of systems where "Entities meaningfully treated as wholes are built up of smaller entities which are themselves wholes … and so on. In a hierarchy, emergent

 doi:10.20944/preprints202603.0509.v1

properties denote the levels"([48] (p. 314), see below for further discussion of emergence). Nobel Laureate Herbert Simon [1916–2001] opined that "complex systems are nearly incomprehensible unless we simplify them using alternative levels of description" [49] (p. 133). In an analogy for systems he continued by stating "By a 'hierarchic system,' or hierarchy, I mean a system that is composed of interrelated subsystems, each of the latter being, in turn, hierarchic in structure until we reach some lowest level of elementary subsystem" [50] (p. 68). An overview of hierarchy theory, focusing on its key concepts, tenets, and history is contained in Wu [51].

The hierarchy concept underpins the construction of the MGOS through the use of interrelated sets, specifically sub-objectives, objectives, and goals to make up and support the mission, which is the whole.

Emergence: Aristotle understood that *wholes* display properties that do not necessarily exist within the individual components. For instance, "whole entities exhibit properties which are meaningful only when attributed to the whole, not its parts – e.g. the smell of ammonia. Every model of systems exhibits properties as a whole entity which derive from its component activities and their structure, but cannot be reduced to them" [48] (p. 314). Grobstein describes the phenomenon as follows: emergence occurs in "set-superset transitions" where "new collective sets with relationally transformed information not resident in the individual components" [52] (p. 46).

The concept of emergence is applied to the construction of the MGOS by understanding that the whole (i.e., mission) can and will create more than the sum of the parts, through unique interactions between the purposefully arranged parts (i.e., sub-objectives, objectives, and goals).

Requisite Parsimony: Psychologist George Miller [1917–2012] wrote a seminal paper *The Magical Number, Seven plus or Minus Two* [53] positing human short-term memory is incapable of recalling more than seven plus or minus two items. Miller's work provides a starting place for defining human span of control. In particular, information must be organized and presented in a way that prevents human information overload. Building on Miller's concept, systems pioneer John Warfield [1925–2009] invoked what he called *requisite parsimony* by applying it to create an algorithmic approach to the construction of a hierarchic objectives tree which can be used to illustrate exceedingly complex analyses [54].

Requisite parsimony is applied to the construction of the MGOS by limiting the number of lower-level elements within a given hierarchy to between seven plus or minus two hierarchic elements.

Minimum Critical Specification: Systems practitioner Albert Cherns [1921–1987] stated that the development of specifications has two aspects, one negative and one positive. The negative simply states that no more should be specified than is absolutely essential; the positive requires that we identify what is essential [55,56].

The principle of minimum critical specification is applied to the MGOS by ensuring that each level in the hierarchy requires only what is essential to support the definition specified in the higher level.

Requisite Saliency: Systems pioneer Kenneth Boulding [1910–1993] wrote that the factors that will be considered in a system design are seldom of equal importance. Instead, there is an underlying logic awaiting discovery in each system design that will reveal the saliency of these factors [57].

Requisite saliency is applied to the evaluation of the constructed MGOS when needs are prioritized and down-selected as part of the stakeholder needs and capabilities specification process. Requisite saliency requires the design team to develop and apply metrics to design factors as key factors in needs and capabilities evaluation.

It is important to note that the prioritization of stakeholder needs and capabilities is useful when designing a system because it helps focus engineers on the design team on delivering the most important needs and capabilities first. It also supports the mandatory trade-off decisions encountered during development of a hierarchic MGOS.

Application of the theoretical elements discussed in this section refers to what Gregor and Jones [58] label as the fifth class of theory: *theory for design and action*, where the theory specifies *how to do*

 doi:10.20944/preprints202603.0509.v1

something. Specifically, "the theory gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact" [59] (p. 620).

The next section will define the MGOS elements.

3.3.2. Definition of MGOS Elements

Hierarchy in design is typically implemented through decomposition, where the overall mission for a system is broken down into supporting goals, objectives, and subobjectives (MGOS). This structure permits the system's concept to be formulated at a high level (e.g., ConOps and mission statement) and then decomposed into lower levels that expand upon and enhance the specificity of the stakeholder's identified needs and the capabilities. This approach mirrors how systems are organized into subsystems and lower-level components, all the way down to unique parts. Application of, and adherence to, the five systems principles from systems theory discussed in the previous section ensures that the MGOS results in a comprehensive expression of the stakeholder's needs and capabilities.

"The process of developing hierarchical structure may be viewed top-down as successively imposing increasingly specific constraints on the form of an ultimate solution. It may also be viewed bottom-up as successively exploiting the constraints satisfied by lower levels" [60] (p. 20). The levels and definitions for each level include:

Mission: The top level in the MGOS is the system's mission. The mission is a clear, concise statement that conveys the need for the system and the desired end-point that the to-be system must achieve. A mission is typically evaluated by its ability to meet the supporting goals.

Goals: Goals are defined as "statements of the intentions and desired outcomes of a system that have different levels of abstraction" [61] (p. 6). The United States' National Aeronautics and Space Administration (NASA) defines a goal as "An elaboration of the need, which constitutes a specific set of expectations for the system. Goals address the critical issues identified during the problem assessment. Goals need not be in a quantitative or measurable form, but they should allow us to assess whether the system has achieved them" [62] (p. 49).

Objectives: Objectives are formally defined as "1. something toward which work is to be directed, a strategic position to be attained, or a purpose to be achieved, a result to be obtained, a product to be produced, or a service to be performed; 2. practical advantage or intended effect, expressed as preferences about future states" [16] (p. 296). Objectives apply the five systems principles from the previous section in order to satisfy the goal to which they are related.

"Specific target levels of outputs the system must achieve. Each objective should relate to a particular goal. Generally, objectives should meet four criteria. (1) They should be specific enough to provide clear direction, so designers, customers, and testers will understand them. They should aim at results and reflect what the system needs to do but not outline how to implement the solution. (2) They should be measurable, quantifiable, and verifiable. The project needs to monitor the system's success in achieving each objective. (3) They should be aggressive but attainable, challenging but reachable, and targets need to be realistic. 'Objectives To Be Determined (TBD)' may be included until trade studies occur, operations concepts solidify, or technology matures. Objectives need to be feasible before requirements are written and systems designed. (4) They should be results-oriented focusing on desired outputs and outcomes, not on the methods used to achieve the target (what, not how)" [62] (p. 49).

Sub-objectives: The term sub-objective utilizes the same definition as objective, but is at a level below the objective, providing another level of refinement in defining the stakeholder needs and capabilities.

A notional MGOS structure is depicted in Figure .

 doi:10.20944/preprints202603.0509.v1

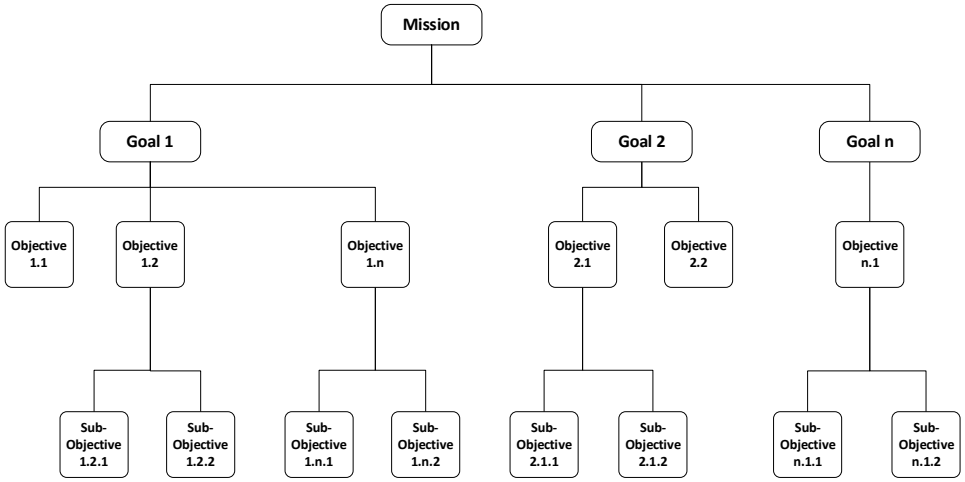


Figure 8. Notional MGOS structure.

The ***req*** diagram contains all of the system-levels needs and capabilities and is capable of interacting with all other diagrams in SysML. As such, requirements may be linked to all of the system's components and provide traceability to and from the lowest-level components all the way back to the original stakeholder needs and capabilities developed in earlier activities.

Figure is a SysML requirements diagram for a feedwater systems in a nuclear power plant using the MGOS hierarchy [63].

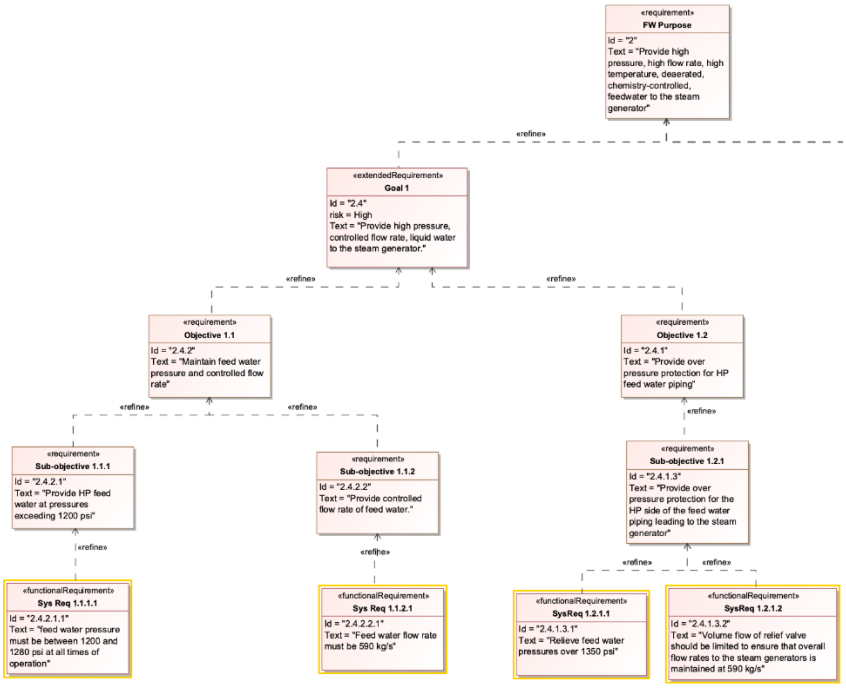


Figure 9. MGOS structure in a SysML Requirements (*req*) diagram.

Once again, it is important to remember that no paper products or external sources are required for this activity as all of the data and information are contained within the SysML model diagrams. Furthermore, retention and management of stakeholder needs in the ***req*** diagram is an inherent function contained within the SysML model.

3.4. Transform Stakeholder Needs into Stakeholder Requirements

During this process the stakeholders review the sub-objectives in the MGOS to ensure that all of the functional and non-functional needs and capabilities are included within the MGOS. This is an

 doi:10.20944/preprints202603.0509.v1

important activity as stakeholders typically address only the functional needs and capabilities that a system must possess when creating their scenarios in the ***uc*** diagram. Failing to identify the non-functional requirements [64] during the conceptual design stage has proven problematic and has led to cost overruns during the systems implementation stage [65–67]. By formally addressing this activity the stakeholders include additional goals, objectives, and sub-objectives that address any missing functional requirements.

The data and information with respect to non-functional requirements are specified as scenarios in new ***uc*** diagrams in the SysML model.

3.5. Analyze the Complete Set of Stakeholder Needs and Capabilities

In this final activity of the process, the entire MGOS is reviewed to ensure that both purpose and behavior are described in the context of the operational environment and conditions for the proposed system. The MGOS provides traceability between mission and supporting goals and the lower-level objectives and sub-objectives, permitting stakeholders to validate the sources and rationale for their expressed needs and capabilities.

A major task within this activity is the definition of critical performance measures, in order to assess the system's ability to technically achieve its mission, through its' goals, objectives and sub-objectives. In some processes this task occurs iteratively and concurrently with the development of the system's mission, goals, objectives, and sub-objectives (MGOS) tree; however, we describe the steps linearly here for simplicity and clarity.

The framework for the development of performance measures for systems is based upon the definition used by the United States Department of Defense for operational test and evaluation (OT&E) which is defined as an effort "to determine the performance of a system under the most current operational conditions" [68] (p. 4). The sub-sections that follow will address technical performance measures for each level in the MGOS using the notions from OT&E.

3.5.1. Mission-Level Performance Measures

At the top or first level of the technical measurement tree, alongside of the system's goals, are critical operational issues (COIs). COIs are defined as "incontrovertible stakeholder-recognized needs derived from problems that must be satisfactorily addressed" [69] (p. 16). COIs are emergent properties that the system must have to perform its designated mission.

The COIs serve as the top-level in a series of system-wide performance measures that include three additional levels: (1) measures of effectiveness (MOEs); measures of performance (MOPs); and (3) technical performance measures (TPMs), Each of these measures should be independent of specific solutions and do not specify methods to achieve measurement or performance criteria. The sections that follow will address each of these measures.

3.5.2. Goal-Level Performance Measures

At the second level of the technical measurement tree, alongside of the system's objectives, are the MOEs. A measure of effectiveness (MOE) is defined as "operational measure of success that is closely related to the achievement of the operational objective being evaluated in the intended operational environment under a specified set of conditions" [16] (p. 268). MOEs are stated from the stakeholder's point of view and represent criteria that must be met in order to satisfy an associated goal, where the "the intended results refer to how well the solution meets the stakeholders needs" [70] (p. 53). Two key characteristics must be the satisfied by MOEs: (1) the ability to be tested; and (2) that they can be quantified in some manner. The NASA systems engineering handbook includes a discrete system design competency (SE 1.1) during technical requirements definition which includes "establishing a set of Measures of Effectiveness (MOEs)" [62] (p. 14).

"MOEs may be either quantitative or qualitative (subjective) in nature. MOEs are typically not directly used as a technical requirement for the system but will be the basis for the concept of

 doi:10.20944/preprints202603.0509.v1

operations and requirements definition. MOEs are intended to focus on how well mission or operational objectives are achieved, not on how they are achieved, i.e., MOEs should be independent of any particular solution. As such, MOEs are the standards against which the "goodness" of each proposed solution may be assessed in trade studies and decision analyses. Measuring or evaluating MOEs not only makes it possible to compare alternative solutions quantitatively, but sensitivities to key assumptions regarding operational environments and to any underlying MOPs can also be investigated" [71] (p. 317).

For example, if a design team is tasked with designing an electric vehicle a valid MOE may be stated as: The electric vehicle must be able to drive fully loaded from Norfolk, VA to Washington, DC without recharging. This MOE is clearly measurable and quantifiable. MOEs are typically supported by a hierarchy of MOPs [70].

3.5.3. Objective-Level Performance Measure

At the third level of the technical measurement tree, alongside of the system's sub-objectives, are the MOPs. A measure of performance (MOP) is defined as "engineering parameter that provides critical performance requirements to satisfy a measure of effectiveness. Note 1 to entry: An MOP typically characterizes physical or functional attributes relating to the system operation" [16] (p. 268). Another definition states "MOPs are measures that characterize physical or functional attributes relating to system operation, measured or estimated under specified testing and/or operational environment conditions" [72] (p. 732). Both definitions support the notion that MOPs are derived from the MOEs but are stated in more technical terms from the engineering team's perspective. It is important to note that there may be two or more MOPs for each MOE. The NASA systems engineering handbook includes a discrete system design competency (SE 1.2) during technical requirements definition which includes "defining the Measures of Performance (MOPs) for each MOE, and defining appropriate Technical Performance Measures (TPMs) by which technical progress will be assessed" [62] (p. 14).

"The MOPs, being quantitative, are used to validate an MOE. A MOE is validated when all the associated MOPs are satisfactorily achieved" [73] (p. 318).

In the example the MOE stated that the electric vehicle must be able to drive fully loaded from Norfolk, VA to Washington, DC without recharging. In support of these more than one MOP will be developed. An example of a supporting MOP is: The vehicle range must be equal to or greater than 250 miles. This establishes a more precise requirement than the distance from Norfolk, VA to Washington, DC. MOPs are supported by any number of specific requirements-based Technical Performance Measures (TPMs).

3.5.4. Sub-Objective Level Performance Measure

The fourth level of the technical measurement tree is made up of technical performance measures (TPM). TPMs are defined as a "measure used to assess design progress, compliance to performance requirements, and technical risks for critical performance parameters" [16] (p. 463). TPMs are directly associated with and established from the MOPs. However, TPMs are not developed during the technical processes of sections 6.4.1 and 6.4.2, depicted in Figure 6, but during the requirements development process in section 6.4.3, depicted in Figure 6, where the "stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user" [20] (p. 54).

In the previous example for an electric vehicle the MOE stated that the electric vehicle must be able to drive fully loaded from Norfolk, VA to Washington, DC without recharging. In support of this MOE an MOP was developed and stated the vehicle range must be equal to or greater than 250 miles. This established a more precise requirement than the distance from Norfolk, VA to Washington, DC—since many different routes might be taken to drive this route. In support of this a series of specific Technical Performance Measures (TPMs), at varying levels of detail, may be invoked.

 doi:10.20944/preprints202603.0509.v1

The TPMs for this example could include battery capacity, vehicle weight, drag, power train friction, etc. ….

3.5.5. Integrated MGOS and Performance Measure Levels

The notional MGOS from Figure is has the technical performance levels added in Figure , below.

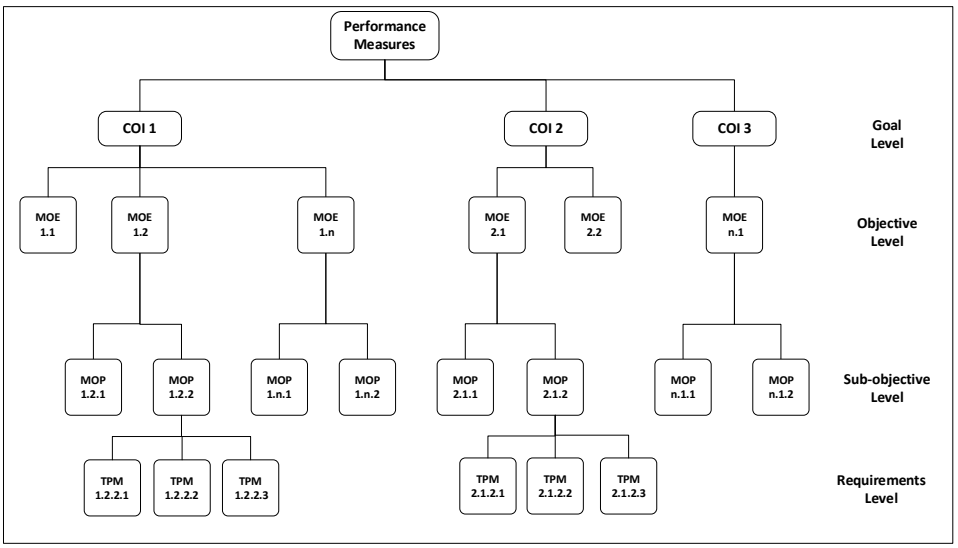


Figure 10. Stakeholder needs and capabilities and system-level engineering requirements process.

The hierarchy in Figure is developed in SysML using a *pkg* diagram as depicted in Figure .

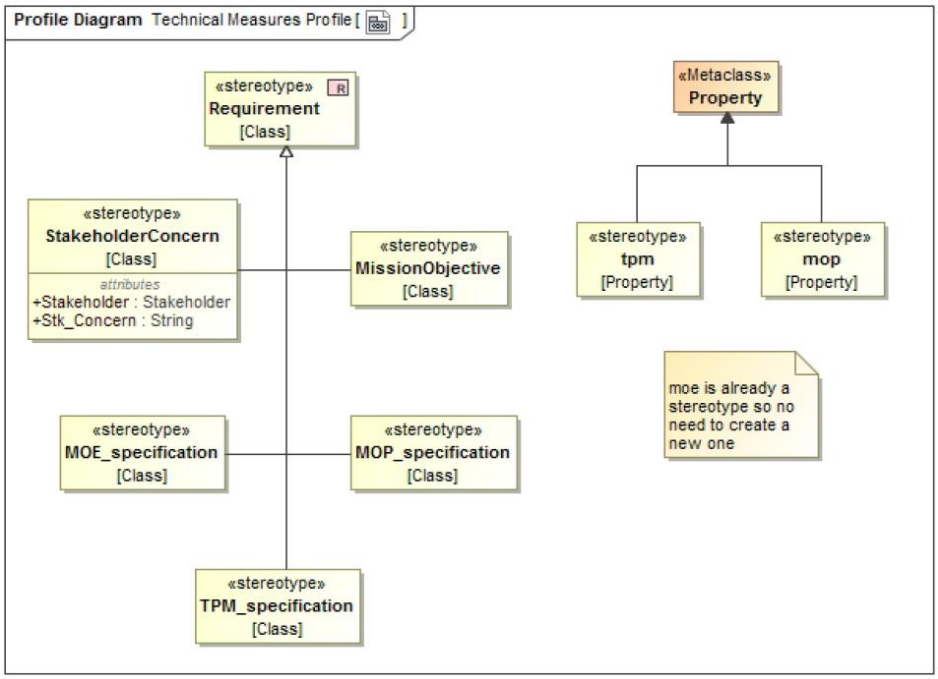


Figure 11. Technical measures profile [72] (p. 734).

3.6. Manage the Stakeholder Needs and Requirements Definition

In this activity the stakeholders must explicitly indicate their agreement with the stakeholder needs and capabilities, as reflected in the MGOS and the associated technical measures. Formal concurrence is ideal; however, in practice design teams may need to be satisfied with other

 doi:10.20944/preprints202603.0509.v1

indications of agreement. It is important that the nature and form of the agreement, along with any remaining disagreements, be documented in the model[2].

All stakeholder decisions are recorded within the system. The SysML model contains all information and provides traceability from every requirement in the model to and from the systems sub-systems and lower level components. At this point, the SysML model may be considered a baseline model in the life cycle development process.

3.7. System Requirements Definition

Upon completion of MGOS interactions with the stakeholders, the sub-objective level of the MGOS contains the stakeholder requirements. These requirements are written in the language of the stakeholders, represent the needs and capabilities required by the proposed system, and are written from the stakeholders' perspective or point of view. This perspective or view represents the specification of a capability or condition that the system must satisfy.

The next step in the system life cycle process model is to derive technical system requirements from the sub-objectives, which is accomplished in 15288 Technical Processes 6.4.3 and 6.4.4. It is important to note that system-level requirements are derived and specified in a formal requirements language. The formal specification of engineering requirements is a vast subject, addressed in engineering courses, texts [75–78], and academic journals [79], and as such, will not be addressed further here.

4. Conclusion

This paper has proposed a formal method and associated techniques for completing the ISO/IEC/IEEE Standard 15288 technical process 6.4.2 – stakeholder needs and requirements definition as a companion work to *Engineering systems with standards and digital models: Development of a 15288-SysML Grid* [1], which describes an engineering design method that supports the tenets of the Industry 4.0 paradigm.

The formal method presented uses established constructs derived from systems science; specifically, the systems principles of hierarchy, emergence, requisite parsimony, minimum critical specification, and requisite saliency, to ground the approach in accepted principles of systems science. The application of the accepted principles ensures that stakeholders are able to objectively specify measurable criterion that can satisfy stakeholder needs and capabilities.

The method is strengthened through:

- The use of international standards for systems engineering (e.g., ISO/IEC/IEEE 15288);
- Adoption of the four fundamental aspects of system design supported by model-based systems engineering (MBSE) [5]; and
- Invocation of the international standard for the systems modeling language (SysML) [6]

Practitioners are encouraged to adopt this technique for the development of a hierarchical requirements tree–one that specifies Mission, Goals, Objectives, and Sub-objectives (MGOS) as a method and SysML as a technique to provide an improved stakeholder process for the clear articulation of system-level engineering requirements. Use of the MGOS is intended to have a positive impact on the system design process by ensuring reproducibility, replicability, transparency, and generalization during the specification of stakeholder needs and capabilities.

---

[2] It is important to note that the engineering team needs to address the need to not only analyze and document stakeholder requirements, but actively manage interactions with identified stakeholders that lead to identifications of such requirements, characterize their inputs and outline strategies to deal with potential disagreements. 74. Hester, P.T.; Adams, K.M. *Systemic decision making: Fundamentals for addressing problems and messes*, 2nd ed.; Springer International Publishing: Cham, Switzerland, 2017.

 doi:10.20944/preprints202603.0509.v1

Author Contributions: Conceptualization and methodology, Kevin Adams; writing—original draft preparation, Kevin Adams and Irfan Ibrahim; conceptualization, writing—review and editing, Steven Krahn. All authors have read and agreed to the published version of the manuscript.

Funding: Irfan Ibrahim's work was partially supported under Cooperative Agreement DE-NE0009086 awarded by the US Department of Energy Office of Nuclear Energy (DOE-NE) Nuclear Energy University Program (NEUP) Fellowship and Scholarship support program. Although not directly funded by the Electric Power Research Institute (EPRI), the authors' understanding of MBSE was enhanced by project-specific work performed for EPRI.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. 15288-SysML Grid for Technical Process 6.4.2 Stakeholder Needs and Requirements Definition.

Technical Process 6.4.2 Stakeholder Needs and Requirements Definition	**Four design aspects**			
	Behavior	**Requirements**	**Structure**	**Parametric**
Activity a: **Prepare for stakeholder needs and requirements definition**				
Task a.1: Identify the stakeholders who have an interest in the solution throughout its life cycle.	Stakeholders for the system. ***uc.***		Structure of system. p***kg.***	
Task a.2: Define the stakeholder needs and requirements strategy.	Depicted with use case diagram. ***uc.***			
Task a.3: Identify and plan for the necessary enabling systems or services needed to support the stakeholder needs and requirements definition.	Achieved using use case diagram. ***uc.***			
Task a.4: Obtain or acquire access to the enabling systems or services to be used.	Achieved using use case diagram, via SysML. ***uc.***			
Activity b: **Develop the operational concept and other life cycle concepts.**				
Task b.1: Define context of use within the concept of operations, the preliminary life cycle concepts, and the preferred solution class(es).	Stakeholders are assigned system-related use cases and preliminary operational behavior defined. ***uc, act, or seq.***		Preliminary operational concept may also be defined using blocks. ***bdd.***	
Task b.2: Define the context of use and a set of scenarios (or use cases) to identify all required capabilities that correspond to anticipated operational concepts and other life cycle concepts.	Stakeholders are assigned system-related use cases and preliminary operational behavior defined. ***uc, act, or seq.***			
Task b.3: Characterize the operational environment and the intended users.	Stakeholders are assigned system-related use cases and preliminary operational behavior defined. ***uc, act, or seq.***		Operational environment. ***bdd.***	
Task b.4: Identify interactions between users and the system and the factors affecting the interactions.	Stakeholders are assigned system-related use cases and preliminary operational behavior defined. ***uc, act, or seq.***			
Task b.5: Identify all interface boundaries across which the SoI interacts with external systems.			Operational environment and boundaries. ***bdd.***	
Task b.6: Identify the constraints on a system solution.			Parameters defined. ***bdd.***	Constraints assigned. ***par.***
Activity c: **Define stakeholder needs.**				

 doi:10.20944/preprints202603.0509.v1

Task c.1: Identify stakeholder needs within the constraints imposed by the life cycle concepts.		Stakeholder mission, goals, objectives, and sub-objectives (MGOS). ***req***		
Task c.2: Prioritize and down-select needs.		Facilitated with requirements diagram. ***req.***		
Task c.3: Record the stakeholder needs and rationale.		Facilitated with requirements diagram. ***req.***		
Activity d: Transform stakeholder needs into stakeholder requirements.				
Task d.1: Identify the stakeholder requirements and functions that relate to critical quality characteristics, such as assurance, safety, security, environment, or health.		Facilitated and recorded in requirements diagram. ***req.***		
Task d.2: Define stakeholder requirements, consistent with life cycle concepts, scenarios, interactions, constraints, critical quality characteristics, and SoS considerations.		Stakeholder requirements defined. ***req.***		
Activity e: Analyze stakeholder needs and requirements.				
Task e.1: Analyze the complete set of stakeholder requirements.		Facilitated with requirements diagram. ***req.***		
Task e.2: Define critical performance measures and quality characteristics that enable the assessment of technical achievement.		Defined alongside stakeholder requirements. ***req.***		
Task e.3: Feedback the analyzed requirements to applicable stakeholders to validate that their needs and expectations have been adequately captured and expressed.		Facilitated with requirements diagram. ***req.***		
Task e.4: Resolve stakeholder requirements issues.		Facilitated with requirements diagram. ***req.***		
Activity f: Manage the stakeholder needs and requirements definition.				
Task f.1: Obtain explicit agreement on the stakeholder requirements.		Documented in requirements diagram. ***req.***		
Task f.2: Record key stakeholder requirements decision and rationale.		Documented in requirements diagram. ***req.***		
Task f.3: Maintain traceability of stakeholder needs and requirements.		Facilitated with requirements diagram. ***req.***		
Task f.4: Provide key artifacts that have been selected for baselines.		Documented in requirements diagram. ***req.***		

References

1. Adams, K.M.; Ibrahim, I.; Krahn, S. Engineering systems with standards and digital models: Development of a 15288-SysML grid. *Systems* **2024**, *12*, Article 276, doi:10.3390/systems12080276.
2. INCOSE. *Systems engineering principles*; International Council on Systems Engineering (INCOSE): San Diego, CA, 2022.
3. Adams, K.M.; Hester, P.T.; Bradley, J.M.; Meyers, T.J.; Keating, C.B. Systems theory: The foundation for understanding systems. *Systems Engineering* **2014**, *17*, 112–123, doi:10.1002/sys.21255.
4. Whitney, K.; Bradley, J.M.; Baugh, D.E.; Chesterman, C.W. Systems theory as a foundation for governance of complex systems. *International Journal of System of Systems Engineering* **2015**, *6*, 15–32, doi:10.1504/IJSSE.2015.068805.
5. Delligatti, L. *SysML distilled: A brief guide to the systems modeling language*; Addison-Wesley Professional: Upper Saddle River, NJ, 2013.
6. ISO/IEC. International Standard ISO/IEC 19514 Information technology — Object management group systems modeling language (OMG SysML). **2017**.
7. Mish, F.C., (Ed.) *Merriam-Webster's collegiate dictionary*. 11th ed.; Merriam-Webster, Incorporated: Springfield, MA, 2009.

 doi:10.20944/preprints202603.0509.v1

8. NAESM. *Reproducibility and replicability in science*; The National Academies Press: Washington, DC, 2019.
9. IEEE. IEEE Standard for Transparency of Autonomous Systems. **2001**, doi:10.1109/IEEESTD.2022.9726144.
10. Parker, S., (Ed.) *McGraw-Hill dictionary of engineering*. McGraw-Hill: New York, 1994.
11. Petroski, H. *The essential engineer: Why science alone will not solve our global problems*; Alfred A. Knopf: New York, 2010.
12. Kirby, R.S.; Withington, S.; Darling, A.B.; Kilgour, F.G. Engineering in history. **1990**.
13. de Weck, O.L.; Roos, D.; Magee, C.L. *Engineering systems: Meeting human needs in a complex technological world*; MIT Press: Cambridge, MA, 2011.
14. Penny, R.K. Principles of engineering design. *Postgraduate Medical Journal* **1970**, *46*, 344–349, doi:10.1136/pgmj.46.536.344.
15. Dixon, J.R. *Design engineering: Inventiveness, analysis, and decision making*; McGraw Hill: New York, 1966.
16. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24765: Systems and software engineering — Vocabulary. **2017**, doi:10.1109/IEEESTD.2017.8016712.
17. Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K.-H. *Engineering design: a systematic approach*, 3rd ed.; Springer: Darmstadt, 2007.
18. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 15026-1: Systems and software engineering — Systems and software assurance — Part 1: Concepts and vocabulary. **2019**, doi:10.1109/IEEESTD.2019.8657410.
19. ISO/IEC/IEEE. *International Standard ISO/IEC/IEEE 15026-4: Systems and software engineering — Systems and software assurance* — Part 4: Assurance in the life cycle. **2021**, doi:10.1109/IEEESTD.2021.9444258.
20. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 15288: Systems and software engineering — System life cycle processes. **2015**, doi:10.1109/IEEESTD.2015.7106435.
21. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 15289: Systems and software engineering — Content of life-cycle information products (documentation). **2019**, doi:10.1109/IEEESTD.2019.8767110.
22. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 15939: Systems and software engineering — Measurement process. **2017**, doi:10.1109/IEEESTD.2017.7907158.
23. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 16085: Systems and software engineering — Life cycle processes — Risk management. **2021**, doi:10.1109/IEEESTD.2021.9325968.
24. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 16326: Systems and software engineering — Life cycle processes — Project management. **2019**, doi:10.1109/IEEESTD.2019.8932690.
25. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24641: Systems and Software engineering — Methods and tools for model-based systems and software engineering. **2023**, doi:10.1109/IEEESTD.2023.10123376.
26. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24748-1: Systems and software engineering — Life cycle management — Part 1: Guidelines for life cycle management. **2018**, doi:10.1109/IEEESTD.2018.8526560.
27. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24748-2: Systems and software engineering — Life cycle management — Part 2: Guide to the application of ISO/IEC 15288 (System Life Cycle Processes). **2018**, doi:10.1109/IEEESTD.2018.8764712.
28. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24748-4: Systems and software engineering — Life cycle management — Part 4: Systems engineering planning. **2016**, doi:10.1109/IEEESTD.2016.7470727.
29. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24748-6: Systems and software engineering — Life Cycle Management. **2023**, doi:10.1109/IEEESTD.2023.10194524.
30. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24748-7000: Systems and software engineering — Life cycle management — Part 7000: Standard model process for addressing ethical concerns during system design. **2022**, doi:10.1109/IEEESTD.2022.9967807.
31. IEEE. IEEE Standard 828: Configuration Management in Systems and Software Engineering. **2012**, doi:10.1109/IEEESTD.2012.6170935.
32. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 24774: Systems and Software Engineering — Life Cycle Management — Guidelines for Process Description. **2021**, doi:10.1109/IEEESTD.2021.9442426.
33. IEEE. IEEE Guide to the Adoption of ISO/IEC Standard 24774: Systems and Software Engineering — Life Cycle Management — Guidelines for Process Description. **2012**, doi:10.1109/IEEESTD.2012.6190704.

 doi:10.20944/preprints202603.0509.v1

34. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 42010: Systems and software engineering — Architecture description. **2022**, doi:10.1109/IEEESTD.2022.9938446.
35. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 42020: Software, systems and enterprise — Architecture processes. **2019**, doi:10.1109/IEEESTD.2019.8767004.
36. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 42030: Software, systems and enterprise — Architecture evaluation framework. **2019**, doi:10.1109/IEEESTD.2019.8767001.
37. PMI. *A guide to the project management body of knowledge (PMBOK® Guide)*, 5th ed.; Project Management Institute: Newtown Square, PA, 2013.
38. Wynn, D.C.; Clarkson, P.J. Process models in design and development. *Research in Engineering Design* **2018**, *29*, 161–202, doi:10.1007/s00163-017-0262-7.
39. Hatakeyama, S.J.; Seal, D.; Farr, D.; Haase, S.C. Systems engineering "V" in a model-based engineering environment: Is it still relevant? *Proceedings of the 2018 AIAA SPACE and Astronautics Forum and Exposition (AIAA 2018-5326)* **2018**, 1–7, doi:10.2514/6.2018-5326.
40. Sales, D.C.; Becker, L.B. Systematic literature review of system engineering design methods. *Proceedings of the 2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)* **2018**, 213–218, doi:10.1109/SBESC.2018.00040.
41. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 15288: Systems and software engineering — System life cycle processes. **2023**, doi:10.1109/IEEESTD.2023.10123367.
42. ISO/IEC/IEEE. International Standard ISO/IEC/IEEE 29148: Systems and software engineering — Life cycle processes — Requirements engineering. **2018**, doi:10.1109/IEEESTD.2018.8559686.
43. Sowa, J.F.; Zachman, J.A. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal* **1992**, *31*, 590–616, doi:10.1147/sj.313.0590.
44. Zachman, J.A. A framework for information system architecture. *IBM Systems Journal* **1987**, *26*, 276–292, doi:10.1147/sj.263.0276.
45. DoD. *Operational Concept Description (DI-IPSC-81438)*; Department of Defense: Washington, DC, 2000.
46. DoE. *Mission need statement guide*; Department of Energy: Washington, DC, 2008.
47. Walker, L.M. Model based systems engineering-Focus on the initial stages; "Get it right in the first stage". *Proceedings of the 25th Annual INCOSE International Symposium* **2015**, *25*, 600–618, doi:10.1002/j.2334-5837.2015.00084.x.
48. Checkland, P.B. *Systems thinking, systems practice*; John Wiley & Sons: New York 1999.
49. Pattee, H.H. *Hierarchy theory: The challenge of complex systems*; George Braziller: New York, 1973.
50. Simon, H.A. The architecture of complexity. *Proceedings of the American Philosophical Society* **1962**, *106*, 467–482.
51. Wu, J. Hierarchy theory: An overview. In *Linking Ecology and Ethics for a Changing World: Values, Philosophy, and Action*, Rozzi, R., Pickett, S.T.A., Palmer, C., Armesto, J.J., Callicott, J.B., Eds.; Springer Netherlands: Dordrecht, 2013; pp. 281–301.
52. Grobstein, C. The hierarchical order and neogenesis. In *Hierarchy theory: The challenge of complex systems*, Pattee, H.H., Ed.; George Braziller: New York, 1973; pp. 29–48.
53. Miller, G.A. The magical number seven, plus or minus two: Some limits on our capability for processing information. *Psychological Review* **1956**, *63*, 81–97, doi:10.1037/h0043158.
54. Warfield, J.N. On arranging elements of a hierarchy in graphic form. *IEEE Transactions on Systems, Man, and Cybernetics* **1973**, *SMC-3*, 121–132, doi:10.1109/TSMC.1973.5408493.
55. Cherns, A. The principles of sociotechnical design. *Human Relations* **1976**, *29*, 783–792, doi:10.1177/001872677602900806
56. Cherns, A. The principles of sociotechnical design revisited. *Human Relations* **1987**, *40*, 153–161, doi:10.1177/001872678704000303.
57. Boulding, K.E. *The impact of social sciences*; Rutgers University Press: New Brunswick, NJ, 1966.
58. Gregor, S.; Jones, D. The anatomy of a design theory. *Journal of the Association for Information Systems* **2007**, *8*, 312–335, doi:10.17705/1jais.00129.
59. Gregor, S. The nature of theory in information systems. *MIS Quarterly* **2006**, *3*, 611–642, doi:10.2307/25148742.

 doi:10.20944/preprints202603.0509.v1

60. Ross, D.T.; Goodenough, J.B.; Irvine, C.A. Software engineering: Process, principles, and goals. *Computer* **1975**, *8*, 17–27, doi:10.1109/C-M.1975.218952.
61. Gregoriades, A.; Sutcliffe, A. Using task support requirements during socio-technical systems design. *Systems* **2024**, *12*, Article 348, doi:10.3390/systems12090348.
62. NASA. *NASA systems engineering handbook (NASA SP-2016-6105 Rev 2)*; National Aeronautics and Space Administration: Washington, DC, 2016.
63. Ibrahim, I.; Krahn, S.; Adams, K.M. Development of a PWR feedwater system model using MBSE and SysML. *Proceedings of the ANS Winter Meeting 2022* **2022**, 755–758, doi:10.13182/T127-39756.
64. Adams, K.M. *Non-functional requirements in systems analysis and design*; Springer: New York, 2015.
65. Dongmo, C. Analyzing non-functional requirements (NFRs) beyond requirements engineering. *Engineering, Technology & Applied Science Research* **2025**, *15*, 23790–23798, doi:10.48084/etasr.9800.
66. Ameller, D.; Xavier Franch ; Gomez, C.; Martınez-Fernandez, S.; Araujo, J.; Biffl, S.; Cabot, J.; Cortellessa, V.; Fernandez, D.M.; Moreira, A.; et al. Dealing with non-functional requirements in model driven development: A survey. *IEEE Transactions on Software Engineering* **2021**, *47*, 818–835, doi:10.1109/TSE.2019.2904476.
67. Maier, J.R.A.; Ezhilan, T.; Fadel, G.M.; Summers, J.D.; Mocko, G.M. A hierarchical requirements modeling scheme to support engineering innovation. *DS 42: Proceedings of the 16th International Conference on Engineering Design (ICED 2007)* **2007**, 582–593.
68. Stevens, R.T. *Operational test and evaluation: A systems engineering process*; John Wiley & Sons: New York, 1979.
69. Hester, P.T.; Meyers, T.J. Multi-criteria performance measurement: An introduction. *Perspectives on Performance* **2011**, *9*, 16–18.
70. Sproles, N. Coming to grips with measures of effectiveness. *Systems Engineering* **2000**, *3*, 50–58, doi:10.1002/(SICI)1520-6858(2000)3:1<50::AID-SYS4>3.0.CO;2-U.
71. Hirshorn, S.R. *Expanded Guidance for NASA Systems Engineering*; National Aeronautics and Space Administration: Washington, DC, 2016.
72. Kaslow, D.; Ayres, B.; Cahill, P.T.; Hart, L. A model-based systems engineering approach for technical measurement with application to a CubeSat. *Proceedings of the 2018 IEEE Aerospace Conference* **2018**, 731–740, doi:10.1109/AERO.2018.8396443.
73. Hirshorn, S.R. *Expanded Guidance for NASA Systems Engineering*; National Aeronautics and Space Administration Washington, DC, 2016.
74. Hester, P.T.; Adams, K.M. *Systemic decision making: Fundamentals for addressing problems and messes*, 2nd ed.; Springer International Publishing: Cham, Switzerland, 2017.
75. Hull, E.; Jackson, K.; Dick, J. *Requirements engineering*, 3rd ed.; Springer: New York, 2011.
76. Robertson, S.; Robertson, J. *Mastering the requirements process*, 2nd ed.; Addison-Wesley: Upper Saddle River, NJ, 2006.
77. Grady, J.O. *System requirements analysis*; Academic Press: Burlington, MA, 2006.
78. van Lamsweerde, A. *Requirements engineering: From system goals to UML models to software specifications*; John Wiley & Sons: Hoboken, NJ, 2009.
79. Springer. Requirements Engineering [ISSN 0947-3602]. **1995**.