Inquirement

Transforming Requirements, Knowledge and Innovation into Traceable Solutions

Inquirement Whitepaper | July 2026 | *inquirement.com* [1] -[2] | Jan Baan | v1.0.0

Executive Summary

The success of engineering and innovation projects depends heavily on decisions made at an early stage when teams must deal with uncertainty, evolving requirements, competing ideas, hidden assumptions, and incomplete information.

Traditional engineering tools excel at managing complexity once it exists. However, comparatively little support exists for the discovery phase in which organizations must determine what the correct requirements should be and which solution directions are worth pursuing. This challenge has long been recognized within the systems engineering community, where requirement quality, traceability, validation, and stakeholder alignment are considered foundational to successful projects. [3] -[5] .

Inquirement was created to address this challenge.

Inquirement allows, engineers, designers, innovators, and decision-makers to explore competing solutions, evolving requirements, assumptions, risks, and supporting evidence in increasing levels of detail while maintaining complete traceability.

Traceability makes decisions easier to understand, defend, review, and reuse. This captures know-how, avoids unnecessary complexity, and improves quality. The process supports creativity and promotes rigor to help organizations generate better solutions.

Introduction

The systems engineering discipline has long emphasized the importance of clear requirements, traceability, and systematic decomposition. The INCOSE Systems Engineering Handbook describes systems engineering as a discipline concerned with realizing successful systems through a structured lifecycle that connects stakeholder needs to implementation and validation activities. Similarly, ISO/IEC/IEEE 29148 identifies requirements engineering as a lifecycle process that is essential to defining, communicating, and validating the intended characteristics of a system [3] -[5] .

However, before requirements can be managed, validated, or allocated, they must first be discovered. This early discovery phase is often characterized by ambiguity. Requirements exist only partially. Alternative solution concepts are not visible. Risks are identified insufficiently. The reasoning behind decisions may remain entirely undocumented.

Inquirement was developed as the culmination of more than twenty-five years of practical experience in design engineering, systems engineering, requirements management, risk management, and knowledge management. The objective was not to create yet another requirements repository, but rather to create a practical environment that helps people ask the right questions, structure reasoning, and discover the requirements that truly matter.

At a time when automation and Artificial Intelligence are rapidly reducing the cost of applying knowledge, the ability to structure, preserve, and transfer knowledge is becoming increasingly important. Knowledge management research has consistently demonstrated that the long-term value of organizations depends not only on what they know, but also on how effectively they capture, structure, and reuse that knowledge across projects. Inquirement supports this objective by combining creativity, traceability, and organization within a single framework.

The Need for Structured Reasoning

Requirements are often treated as independent statements.

In reality, requirements are highly interconnected. Requirements inherit assumptions from stakeholder expectations, influence architectural decisions, constrain implementation choices, and create obligations that propagate throughout a project.

International requirements engineering guidance emphasizes that requirements should be clear, unambiguous, consistent, verifiable, and traceable. Poorly written requirements are recognized as a significant source of project risk, rework, cost, and integration challenges [3] , [5] , [6] .

Even seemingly straightforward requirements may contain hidden design decisions.

A simple requirement such as: *"A door hinge must..."* already contains multiple assumptions.

The grammatical subject assumes the existence of a door. It assumes that a hinged solution has already been selected, excluding alternatives such as sliding doors. It also introduces a level of decomposition by focusing attention on a specific subsystem rather than the overall function.

Similarly, descriptions such as: *Mission* → *Rocket* → *Engine* → *Nozzle* implicitly create a hierarchy and a particular architectural breakdown before alternative approaches have been explored.

Poor wording therefore does more than create ambiguity. It introduces hidden relationships that can propagate throughout an entire project.

The contrast between unstructured and structured reasoning illustrates the challenge. In an unstructured environment, requirements, assumptions, and decisions become increasingly difficult to trace. In contrast relationships remain visible, understandable, and manageable in a structured environment. The requirement decomposition logic flows down. The traceability allows the validation and verification to flow up at the same time.

Making Relationships Explicit

The central philosophy behind Inquirement is that important relationships should be explicit rather than hidden within text.

Requirements traceability is widely recognized as a fundamental systems engineering activity. The Systems Engineering Body of Knowledge describes requirements management as including traceability between stakeholder needs, system requirements, architecture, verification activities, validation activities, and implementation artifacts. Traceability supports impact analysis, change management, validation planning, and stakeholder confidence [4] .

Inquirement extends this concept into the discovery phase by maintaining traceability not only between requirements and solutions, but also between assumptions, arguments, risks, design problems, option evaluations, and evidence.

Instead of relying solely on requirement statements, Inquirement explicitly captures:

- Requirement decomposition
- Alternative solution paths
- Logical dependencies
- Design assumptions
- Functional decomposition
- Integration points
- Validation evidence

This approach creates a transparent chain of reasoning from stakeholder needs to detailed implementation decisions. Each requirement becomes connected to the rationale that created it. Each solution is linked to the challenge it addresses. Every assumption can be traced to supporting evidence or transformed into a requirement that must be satisfied. As a result, decisions become easier to understand, defend, review, and reuse.

The Inquirement Fractal

At the heart of Inquirement is a repeating structure that can be applied at any level of a project. Whether a team is exploring an entire system architecture, analyzing a subsystem, or refining a specific component, the same reasoning pattern can be applied repeatedly. This creates what can be described as the Inquirement fractal.

Two fundamental types of branching occur within this structure.

The first is alternative solution branching.

For a given challenge, multiple competing options may exist: Option A OR Option B.

Only one option may ultimately be selected, but maintaining visibility of alternatives ensures that potentially valuable ideas are not prematurely dismissed.

The second is functional decomposition.

A solution may require multiple elements working together: Option X AND Option Y.

In this case, all contributing elements must be valid for the overall solution to remain valid.

The combination of alternative branching and functional decomposition enables organizations to move seamlessly from high-level concept exploration to detailed engineering decisions with the same repeating structure.

Traceability

Definition of a solution evolves through increasing levels of detail. The requirement decomposition logic flows down. The validation and verification can flow-up if the flow-down is traceable.

Decomposition Flow-Down

The decomposition of stakeholder needs into increasingly detailed requirements is a well-established systems engineering practice. Requirements are typically allocated from system level to subsystem level and subsequently decomposed into implementable elements [6] -[7] .

A high-level requirement often needs to be transformed into more detailed requirements before implementation becomes possible. For example, a requirement involving joint force may eventually lead to detailed requirements concerning bolt diameter, material selection, and installation procedures. Detail increases as requirements flow downward.

In the Inquirement process, derived requirements emerge through structured arguments rather than appearing as isolated statements (**Figure *1***). The structure of parent-child relations preserve the reasoning that led to each decomposition step.

Verification, Validation and Flow-Up

NASA, INCOSE, and other systems engineering authorities emphasize the importance of connecting requirements to verification and validation activities throughout the system lifecycle. Evidence is required to demonstrate that solutions satisfy stakeholder needs and system requirements [3] -[4] .

Inquirement formalizes this relationship through evidence-based validation, where each valid branch of reasoning ultimately terminates in supporting evidence at required Readiness Levels and Evidence-Based Validation.

Evidence that validates detailed requirements contributes to linked higher-level requirement validation. Supporting calculations, analyses, demonstrations, simulations, and tests all become part of a traceable validation chain.

This creates a continuous connection between stakeholder intent and technical proof.

The Inquirement Objects

Inquirement organizes knowledge using a set of interconnected objects (**Figure *1***). Together, they form the framework used to discover, refine, validate, and document solutions.

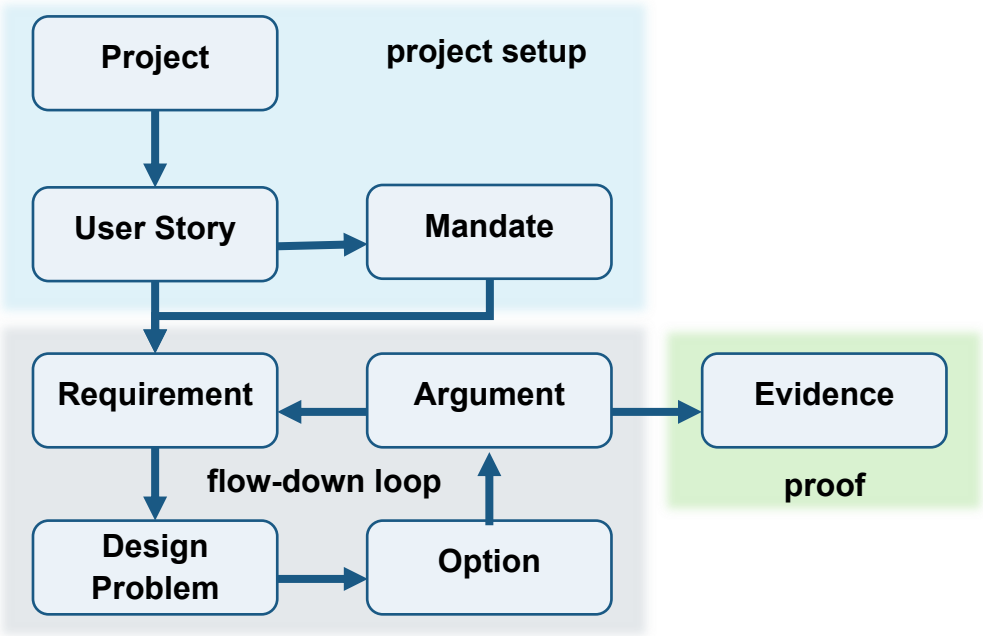

Figure 1. Interconnected objects in Inquirement.

Project
The Project object defines the scope, objective, and context of the discovery effort.

User Story
User Stories describe what stakeholders expect the project to accomplish. They provide an informal description of desired outcomes before formal requirements are established.

Mandate
The Mandate records project authorization and defines the Readiness Level that must be achieved before a solution can be considered acceptable.

Requirement
Requirements formalize stakeholder needs and project constraints. Requirements may be functional, goal-oriented, range-based, or fixed-value requirements.

Design Problem
Design Problems combine requirements into clearly defined challenges that must be solved. A Design Problem effectively asks: *How can these requirements be satisfied simultaneously?* The availability of this question opens the possibility to use an Artificial Intelligence to answer the question.

Option
Options represent candidate solutions. Multiple options may coexist until evidence and reasoning allow the preferred solution to emerge.

Argument
Arguments explain why an Option is believed to satisfy a Requirement. Arguments capture assumptions, constraints, risks, experience-based knowledge, and supporting logic.

Evidence
Evidence objects provide proof. Evidence may consist of analyses, calculations, simulations, demonstrations, reports, laboratory tests, field tests, or operational experience. Every branch in Inquirement ultimately terminates in Evidence, ensuring that reasoning is supported rather than assumed.

Semantic Intelligence Through Subject, Verb and Capability

A distinctive aspect of Inquirement is its semantic foundation (**Figure *2***) that maintains consistency and reduces user input needed.

The methodology uses three semantic key elements:

- Subject
- Verb
- Capability

Subject, Verb and Capability are shared across related objects.

For example, a requirement stating:

The SYSTEM must KEEP OUT PEOPLE

automatically supports generation of a corresponding Design Problem:

How can a SYSTEM KEEP OUT PEOPLE?

A possible solution option might be:

LOCK

Which leads naturally to an Argument:

A LOCK can KEEP OUT PEOPLE if...

And ultimately to a derived Requirement:

The LOCK must RESIST BREAK-IN.

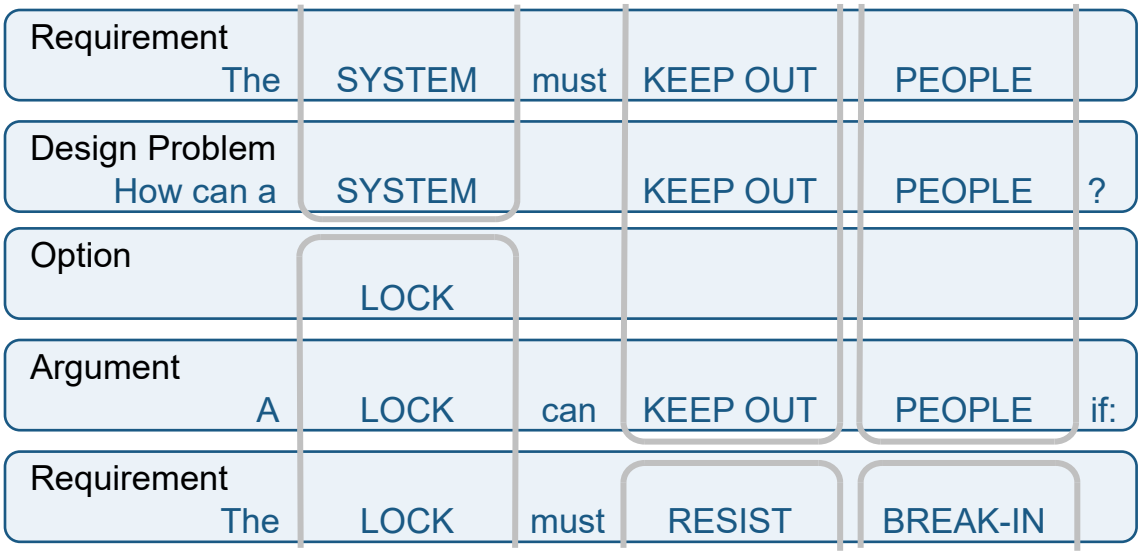

Figure 2. The semantic foundation of Inquirement allows consistency and reduces user effort. The Subject, Verb, and Capability elements are shared across related objects.

Inquirement's semantic foundation delivers several important benefits:

- Reduced manual input
- Improved consistency
- Automatic problem formulation
- Better requirement quality
- Automatic AI prompt generation

By embedding semantic relationships directly into the model, Inquirement helps users formulate precise reasoning structures while minimizing administrative effort.

The Inquirement Process

The workflow within Inquirement follows a structured progression from project definition to validated solution, which is depicted in the tab pages of the Inquirement application.

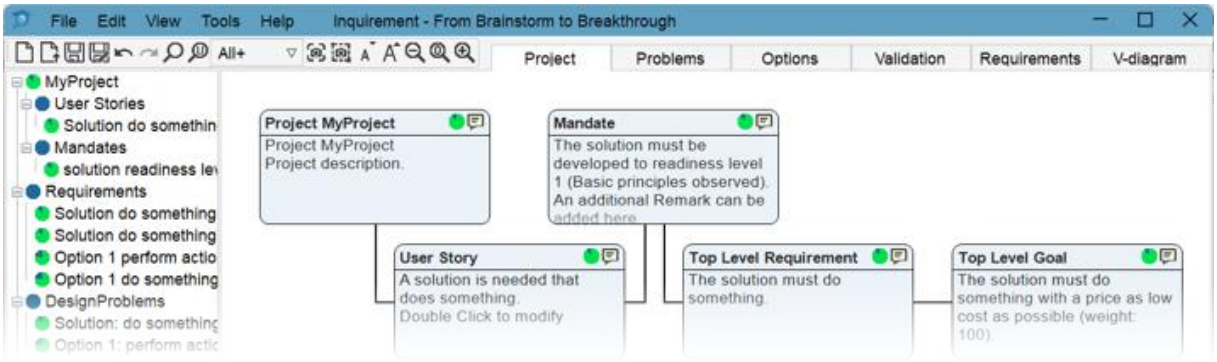

Figure 3. The tab pages of the Inquirement application depict the structured progression from project definition to validated solution.

Project tab page

The process begins by defining the project, user stories, mandate, and top-level requirements. Only a limited number of initial requirements are necessary. Additional requirements emerge naturally as understanding increases.

Problems tab page

Requirements are organized into Design Problems. Rather than attempting to solve every requirement simultaneously, teams can focus on the most critical or contradictory requirements first.

Options tab page

Potential solution concepts are generated for each Design Problem. This stage supports both human brainstorming and AI-assisted idea generation.

Validation tab page

Options are evaluated against requirements. Arguments can point to supporting evidence or assume new requirements, which must be satisfied before the options is valid. New requirements are defined based on logic, experience or because certain identified risks must not materialize. AI-assistance is also available in this stage.

Requirements tab page

The Requirements view provides visibility into requirement origins and dependencies, enabling users to understand why each requirement exists.

V-Diagram tab page

The V-Diagram identifies and visualizes the preferred solution path, connecting requirements, options, and evidence into a coherent structure. The repeating structure of the Inquirement object fractal is especially suitable to be represented as a V-diagram.

AI-Assisted Discovery

Recent work in requirements engineering demonstrates growing interest in applying Artificial Intelligence to detect ambiguity, assist with requirement analysis, and support exploration of the solution space. However, studies also emphasize the importance of context and human judgment because requirements often depend on domain-specific knowledge that cannot be derived from text alone [5] , [8] .

Inquirement therefore positions AI as an assistant rather than a decision-maker. AI contributes ideas and suggestions, while responsibility for interpretation, validation, and selection remains with the user in a controlled and deliberate manner.

AI can assist with:

- Developing solution options
- Exploring alternatives
- Identifying validation approaches
- Suggesting additional considerations

Inquirement generates prompts that can be submitted to any Large Language Model. Responses are provided in a predefined XML format that can be imported back into the project.

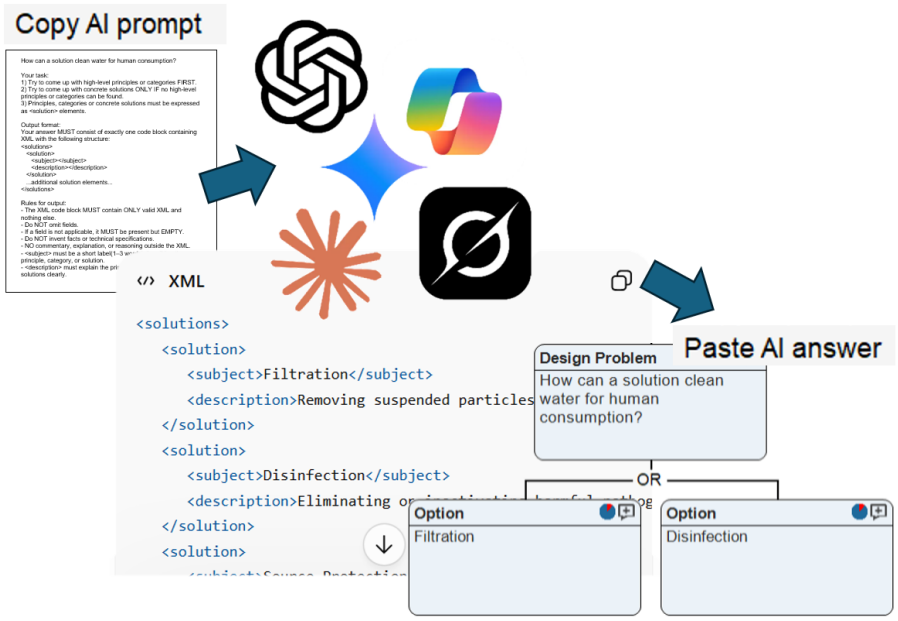

Figure 4. Workflow for AI assistance.

Importantly, project information remains under user control rather than being automatically exposed to external services.

Readiness Levels and Evidence-Based Validation

A unique aspect of Inquirement is its integration of Readiness Levels into the decision-making process [10] . Each project defines a required maturity level through the Mandate. Evidence demonstrates achievement of that level.

The framework uses the following progression:

Level 0 — Expectation
Level 1 — Observed Principle
Level 2 — Concept Analysis
Level 3 — Concept Simulation
Level 4 — Laboratory Validation
Level 5 — Relevant Environment Testing
Level 6 — Relevant Environment Demonstration
Level 7 — Operational Demonstration
Level 8 — Fully Qualified
Level 9 — Fully Operational

As evidence accumulates, confidence automatically propagates upward through the reasoning structure. Higher-level decisions therefore inherit validation from lower-level proof.

Managing Risk Through Requirements

In practice, risk management rarely goes beyond passive observation. Inquirement takes a different approach. Risks are identified as requirements stating that undesirable outcomes must not occur, which brings teams into solution mode. For example, instead of merely recording a risk that a component may fail, a requirement can be introduced requiring system continuity. This can then be achieved by either failure prevention, redundancy or recovery. The resulting option is considered invalid until evidence demonstrates compliance. This simple shift encourages proactive mitigation rather than passive observation.

Preventing Over-Engineering

One of the most common causes of unnecessary complexity is solving problems that should not exist in the first place. Unstructured reasoning tends to generate additional features, constraints, and requirements that provide little value. In Inquirement, only solutions exist that solve actual requirements. Every solution element can be traced back to the stakeholder needs. The fully traceable reasoning structure helps reduce waste, simplify designs, and improve decision quality.

Capturing and Reusing Know-How

Engineering decisions frequently depend on experience and tacit knowledge. If reasoning is not documented, organizations repeatedly solve the same problems and lose valuable insights when individuals leave projects. Many organizations repeatedly solve similar problems because previous reasoning was never structured or preserved.

Knowledge management literature has highlighted the importance of converting tacit knowledge into explicit, reusable knowledge structures [9] . Inquirement supports this by preserving both selected solutions and rejected alternatives.

Inquirement addresses this through explicit knowledge capture. The reasoning behind decisions is embedded in the requirement decomposition logic flow-down and validation and verification flow-up. Branches in the Inquirement fractal represent:

- Selected solutions
- Rejected alternatives
- Supporting evidence
- Derived requirements

This allows future teams to build upon existing knowledge rather than reinventing it. Knowledge therefore becomes transferable, accessible, and less dependent on individual experts.

Current reuse is limited to using existing branches in the Inquirement fractal to recreate a new branch and adapt it to the specifics of the new branch. Special functionality will be developed to support this process better. For certain cases full instantiation from a library or another branch might be possible.

Where Inquirement Fits Within the Engineering Ecosystem

Inquirement complements existing engineering environments. It is particularly valuable during the high-ambiguity stage of development when problems are still being defined.

It is not intended to replace:

- Requirements repositories
- Model-Based Systems Engineering tools
- Product Lifecycle Management systems
- Digital thread solutions
- Document management platforms

Instead, Inquirement provides the missing link between first ideation and formal engineering execution. It supports exploration before organizations commit significant effort to downstream engineering activities.

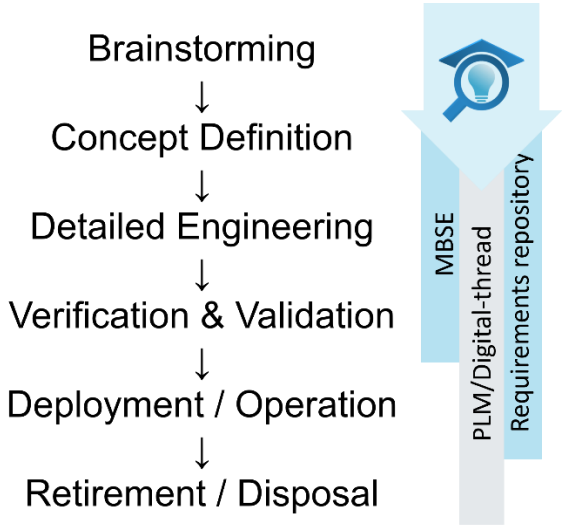

Figure 5. Inquirement provides the missing link between brainstorming and formal engineering.

Once a solution has been explored and refined, Inquirement enables a smooth transition into existing engineering toolchains through its PDF and ReqIF export capabilities. These export options allow users to transfer validated results into their requirements management, MBSE, PLM, or other engineering tools of choice, ensuring that insights generated during the exploration phase can be seamlessly integrated into formal engineering processes.

Inquirement aims to support this transition as much as possible. Please contact us for specific requests that can also help us include more and more export capabilities.

Conclusion

The greatest engineering challenges are rarely caused by insufficient technical expertise. More often, they originate from unclear requirements, hidden assumptions, disconnected knowledge, and poorly understood relationships.

Inquirement addresses these challenges by providing a structured framework for transforming uncertainty into validated solutions. By combining semantic consistency, explicit reasoning, requirement evolution, evidence-based validation, risk-driven thinking, and knowledge capture, Inquirement helps organizations move from brainstorm to breakthrough.

The results are better questions, requirements, decisions, and ultimately better solutions.

Most engineering tools help manage complexity after it exists.
Inquirement helps prevent unnecessary complexity before it spreads.

References

[1] Inquirement, “Inquirement Website.” [Online]. Available: https://inquirement.com. [Accessed: Jul. 14, 2026].

[2] Inquirement, **Inquirement: From Brainstorm to Breakthrough – User Manual**, Version 1.0.3, Jul. 8, 2026. [Online]. Available: https://inquirement.com. [Accessed: Jul. 14, 2026].

[3] International Council on Systems Engineering (INCOSE), **Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities**, Version 5.0. Hoboken, NJ, USA: Wiley, 2023.

[4] Systems Engineering Body of Knowledge (SEBoK), “Requirements Management.” [Online]. Available: https://www.sebokwiki.org/wiki/Requirements_Management. [Accessed: Jul. 14, 2026].

[5] ISO/IEC/IEEE Standard 29148:2018, **Systems and Software Engineering—Life Cycle Processes—Requirements Engineering**, 2018.

[6] F. Valera, “INCOSE Guide to Writing Requirements,” *Visure Solutions*, Sep. 18, 2025. [Online]. Available: https://visuresolutions.com/alm-guide/incose-guide-to-writing-requirements/. [Accessed: Jul. 14, 2026].

[7] O. de Weck, **Fundamentals of Systems Engineering**. Massachusetts Institute of Technology OpenCourseWare. [Online]. Available: https://ocw.mit.edu/. [Accessed: 14-Jul-2026].

[8] V. V. Poulsen, M. Guertler, B. Eisenbart, L. Tomidei, and N. Sick, “Context-aware large language models for ambiguity detection in requirements,” *Proceedings of the Design Society*, vol. 6, 2026, Art. no. 10607. doi: 10.1017/pds.2026.10607.

[9] I. Nonaka, “A Dynamic Theory of Organizational Knowledge Creation,” *Organization Science*, vol. 5, no. 1, pp. 14–37, Feb. 1994, doi: 10.1287/orsc.5.1.14.

[10] NASA Earth Science Technology Office. (n.d.). *Technology Readiness Levels (TRLs).* National Aeronautics and Space Administration. https://esto.nasa.gov/trl/ .