INCOSE

ASIA-OCEANIA SYSTEMS ENGINEERING CONFERENCE

Assessment Method for System Development to Ensure the Quality of Deliverables

An Approach using Quality Criteria and Limit Samples

INCOSE AOSEC · 2025

AUTHORS

Seiji Honda

PUBLICATION DATE

July, 2026

SUBMISSION #

77

INCOSE PUBLICATIONS

Copyright © 2025 by the authors. Permission granted to INCOSE to publish and use.

Assessment Method for System Development to Ensure the Quality of Deliverables: An Approach using Quality Criteria and Limit Samples

Seiji Honda
Toyota Motor Corporation
Toyota, Aichi, Japan
seiji_honda_aa@mail.toyota.co.jp

Copyright © 2025 by the author(s). Permission granted to INCOSE to publish and use.

Abstract

This paper proposes an assessment method to suppress inconsistencies in the quality of deliverables in the upstream phases of the technical processes in systems engineering (SE) and to ensure a certain level of quality of deliverables during this phase. The method presented in this paper standardizes evaluation criteria to improve the quality of design deliverables across seven process stages, from SYS.0 to SYS.6. Traditionally, there have been no clear process audits or compliance evaluations specifically targeting SE processes, and although ISO/IEC/IEEE 15288 and Automotive SPICE (A-SPICE) are effective for confirming adherence to development process procedures, it has been difficult to use for controlling inconsistency in deliverable quality (VDA/QMC,2023). Addressing these challenges, we propose a framework that pairs *quality judgment criteria*, described in sufficient detail for both assessors and developers, with *limit samples* as clear, minimally acceptable examples meeting those criteria. The scope covers SYS.1 to SYS.5 of A-SPICE, as well as a newly defined SYS.0 for organizing past asset information, and SYS.6 for validating stakeholder requirements. In this paper, we applied the method to specific deliverables of an example system created for this study and verified particularly how the *quality judgment criteria* contribute to visualizing and improving quality. As an effect of this method, it is expected not only to suppress subjective judgments during reviews and ensure deliverable quality across different personnel and departments, but also to improve the quality of the system requirements specification document created by gathering these deliverables, thus enabling the enhancement of the overall quality of specifications.

Keywords

Systems engineering, System development, Assessment, Quality criteria, Limit samples

Introduction

In complex system development fields such as the automotive industry, aerospace, and railways, development processes conforming to international lifecycle process standards such as ISO/IEC/IEEE 15288 and process assessment models widely used in the automotive industry, such as Automotive SPICE (A-SPICE), have begun to be adopted. While

these standards are effective as frameworks for evaluating the maturity of development processes and the implementation status of activities, they do not specify pass/fail criteria or quantitative quality assessment methods for the deliverables themselves. Therefore, even when strictly following the prescribed system development processes, the issue of quality inconsistency in deliverables repeatedly occurs in the field.

In practice, designers' tendencies in writing specifications, inconsistencies in expression, and subjective judgments by reviewers often lead to omissions, ambiguities, and unresolved mismatches in viewpoints as the work proceeds to the next phase. This often results in rework during design or verification stages and increased effort. For example, although A-SPICE is used for process evaluation, it does not specify pass/fail criteria or quantitative quality assessments for deliverables, so each organization decides these on its own. Therefore, quality inconsistency remains even when going through the same process. Additionally, on-site discussions often get stuck on debates about how much detail is sufficient, causing delays in progress.

Considering the above, this paper starts from the recognition that following the process exactly does not guarantee high-quality deliverables and proposes a method that complements process audits like A-SPICE by clearly defining *quality judgment criteria* for deliverables, enabling consistent judgments regardless of the reviewer, combined with concrete examples called *limit samples* that meet these criteria. The contribution of this method is expected not only to improve the quality of system development deliverables but also to provide a framework that enables anyone to comfortably carry out design reviews aligned with systems engineering.

The roles of *Quality judgment criteria* and *Limit samples*

The core of this method consists of two components: the *quality judgment criteria* and the *limit samples*. The former clearly defines a set of evaluation items for objectively assessing the quality of deliverables, minimizing interpretation differences among users. The latter presents concrete examples corresponding to each item in the quality judgment criteria, using tabular or graphical formats. This allows even first-time users of the method to evaluate the quality of actual deliverables by comparing them against both the quality judgment criteria and the limit samples.

Each component alone has limited effectiveness. If only the quality judgment criteria are used, it becomes difficult to grasp concrete images of the deliverables. Conversely, if only the limit samples are provided, it is challenging to make appropriate evaluations because the criteria are unclear. Therefore, it is important to operate both the quality judgment criteria and the limit samples in combination. By maintaining and revising both components as an integrated set, consistency in judgment and continuous improvement can be achieved simultaneously. These components also serve as effective tools for educating new members and reviewers for systems engineering.

While these provide a standardized definition of deliverable quality in the system development process, it is acceptable to adjust the criteria based on each project's specific conditions or constraints. In addition, the content should be regularly updated to gather best practices across organizations, which will improve the organization's overall evaluation ability over time.

Scope of Process Stages of This Method

In addition to SYS.1 to SYS.5 in A-SPICE, this paper newly defines SYS.0, which organizes past asset information, and SYS.6, which validates stakeholder requirements. In SYS.0, development information such as past similar projects, related standards, and verification records is systematically organized to identify reusable requirements and design rationale at an early stage, thereby suppressing scratch development efforts in system design. SYS.6 performs a final quality assurance by comparing the design intent of the developed system with its actual operation in the usage

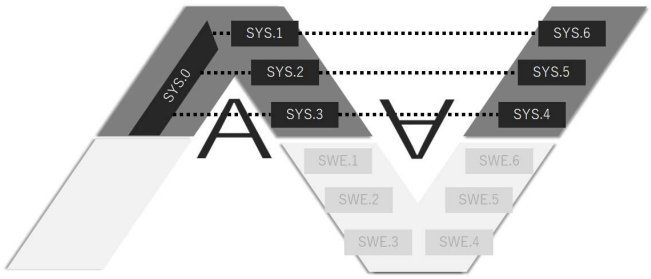

Figure 1. Overview of the Target Process Stages of This Method

environment, ensuring that stakeholder requirements and expectations are met. For reference, SYS.1 to SYS.5 conform to A-SPICE; for example, SYS.1 involves comprehensive collection of stakeholder requirements and their preparation into verifiable forms, SYS.2 covers analysis, structuring of system requirements, and establishment of traceability, and SYS.3 refers to architecture design from both static structure and dynamic behavior perspectives.

This method comprehensively covers the seven process stages from SYS.0 to SYS.6, defining evaluation viewpoints that promote quality improvement of deliverables within the system hierarchy, and is intended for continuous operation as a development standard.

Figure 1 shows the overall process stages, with the black-shaded areas indicating the stages targeted by this method. The figure extends the left side of a typical Vee model development process by adding SYS.0, and since the deliverables of SYS.0 are assumed to be linked with those of SYS.1 to SYS.3, which in turn are linked to SYS.4 to SYS.6, these connections are indicated by dotted lines. Due to its appearance, this is referred to as the A∀ - read as A Turn-A shaped development process.

Examples of *Quality Judgment Criteria* for Deliverables

As concrete examples of the *quality judgment criteria*, which are characterized by explicitly defining a set of evaluation items to objectively assess the quality of deliverables and ensure transparency, this paper explains those for SYS.0 through SYS.3 as representative examples.

For example, the SYS.0 deliverable Legacy design information Asset Catalog requires clearly defining the scope of past assets related to the target system and systematically listing documents, records, standards, and other sources of information for each process. It manages versions, locations, and acquisition methods of past assets, collects related glossaries to ensure consistency, and identifies lists of past stakeholders and requirement items to determine whether they can be reused or adapted for the current development system. This clarifies design direction and technical prerequisites from the initial stage, significantly reducing reconsiderations and rework in later phases.

In the SYS.1 deliverable Stakeholder Requirements List, it is required to comprehensively extract stakeholder needs and constraints, assign IDs and derivation classifications (e.g., interview, regulations, usage methods) to each requirement, and write requirement statements free from ambiguous terms. Verification criteria such as thresholds, units, and environmental conditions must be clearly described, enabling unambiguous interpretation as a mandatory condition. Furthermore, the list and hierarchical (parent-child) relationships of requirements are organized, with careful attention to cross-references, duplication, and circular references. Priority levels must be clearly indicated along with reasons. Version, approval date, approver, and change reasons are maintained as history to clearly indicate the status. These ensure that requirements are consistent and complete, enabling efficient evaluation and consensus from the early stages. Table 1 lists these items as the quality judgment criteria for the Stakeholder Requirements List deliverable.

In the SYS.2 System Requirements List, bidirectional traceability with stakeholder requirements and verification items is ensured. Requirements are defined with unique and quantitative expressions. Priority is explicitly indicated, and TBD items have deadlines and responsible persons assigned. Quantification of performance margins and safety, as well as clarification of conflicts and non-functional requirements, are mandated. Table 2 lists these items in the same manner as above.

In the SYS.3 System Structure Diagram, based on the principle that one function corresponds to one

Table 1. SYS.1 Deliverable - Stakeholder Requirements List (partial)

#	items
1	Stakeholder needs and constraints are comprehensively identified, with each requirement assigned a unique ID and derivation classification (e.g., request, regulation, usage).
2	Each requirement statement is atomic and unambiguously interpretable, allowing for verification.
3	The list of requirements and their hierarchical (parent-child) relationships are well organized, with no conflicts, duplications, or circular references.
4	Priorities (Must/Should/Could) are defined, and the reasons for these priorities are documented.
5	Baseline management maintains version, approval, and change histories.

Table 2. SYS.2 Deliverable - System Requirements List (partial)

#	Items
1	Bidirectional traceability with stakeholder requirements is ensured (no missing traces).
2	Each requirement statement is atomic and unambiguously interpretable, allowing for verification.
3	Priorities (Must/Should/Could) are defined, and the reasons for these priorities are documented.
4	For performance requirements, performance margin (e.g., bandwidth, delay margin) is specified.
5	If there are any TBD items, deadlines and responsible persons are always specified (no blanks).
6	Baseline management maintains version, approval, and change histories.

responsibility, elements are hierarchically decomposed into 3 to 7 units. Functional elements are traceable bidirectionally with related system requirements and dynamic behaviors. Interfaces (hereinafter I/F) are explicitly indicated using port notation. Reusable assets and externally procured items are identified among functional elements. Safety and regulatory requirements are allocated to functional elements, and naming is standardized across deliverables.

As these representative examples show, a set of quality judgment criteria covering a total of 20 deliverables from SYS.0 to SYS.6 has been formulated. Each criterion is not merely a formal checklist but is structured as specific judgment conditions that satisfy objectivity and transparency. These criteria are composed by extracting main items from A-SPICE, the author's practical experience, the INCOSE Systems Engineering Handbook 4th Edition, and the IPA Embedded Software Development Process Guide (IPA, 2007). This approach enables quality assurance of deliverables that does not depend on subjective judgments but achieves cross-organizational and reproducible quality assurance.

Examples of *Limit Samples* for Deliverables

The *limit samples*, characterized by presenting concrete examples corresponding to each item of the quality judgment criteria in tabular or graphical form, are explained here using SYS.1 through SYS.3 as representative examples. The subject system for the limit samples was based on the familiar example of a smartphone to help readers easily understand a concrete image of the deliverables.

For example, the limit sample for the SYS.1 deliverable Stakeholder Requirements List (Table 3) is structured as minimal examples that fulfill all mandatory items of the quality judgment criteria. Each column includes a unique requirement ID, requirement descriptions written in verifiable quantitative or specific terms, hierarchical structuring by level, conflicting relationships with other requirements, stakeholder names, derivation classifications (e.g., request/Constraint), applicable lifecycle phases, and priority without omission or redundancy. This clarifies relationships and priority rationale among requirements, enabling users to quickly verify completeness, consistency, and validity. Furthermore, this format contributes to early clarification of design

policies during requirement definition and suppression of rework in later phases.

Table 3. Limit Sample for the Deliverable - Stakeholder Requirements List

Requirement ID	Requirement Description	Rationale	Level	Conflict	Stakeholder	Derivation	Lifecycle	Priority	Priority Reason
User-Req-001	The user requires that the device can be used continuously for at least 12 hours on a single charge.	To enable long-term use and ensure convenience.	Lv1	–	User	Request	Use /Operation	Must	Ensure usability for long usage
User-Req-002	The telecom company wants to provide a stable communication connection to the smartphone.	To maintain communication service quality and improve customer satisfaction.	Lv1	–	Telecom Company	Constraint	Use /Operation	Must	To maintain communication quality
User-Req-002-01	The telecom company supports data communication speeds up to 1 Gbps on the smartphone.	To meet the demand for high-speed communication and maintain competitiveness.	Lv2	–	Telecom Company	Constraint	Use /Operation	Must	To maintain communication quality
User-Req-003	The app developer wants the smartphone OS to support the app development environment.	To support the latest development environments and enhance app functionality and stability.	Lv1	–	App Developer	Request	Use /Operation	Could	To improve app development efficiency
User-Req-004	Other users want wireless communication between smartphones to be simple and fast.	To enable smooth data exchange and collaboration with friends and partners.	Lv1	User-Req-002-02	Other Users	Request	Use /Operation	Should	To improve comfort

Table 4. Limit Sample for the Deliverable - Environment / External System I/F List

Interface ID	Interface Name	Exchange Content	Type	Output Destination (To)	Input Source (From)	Data Name	Unit	Reason Details	Related Req. ID	Related System/Service
IF-REQ-001	Power Supply I/F	Electric Power	Energy	Smartphone	Charger	Power Voltage	V	To maintain power supply	SysReq-001	Power Outlet
IF-REQ-002	Communication Auth I/F	Communication Authentication Info	Information	Smartphone	Telecom Company Server	Authentication Token	(String)	To control network access	SysReq-002	Communication Network
IF-REQ-003	Location Info Tx/Rx I/F	Location Information	Information	Smartphone	GPS Satellite	Latitude/Longitude	deg	To improve service usability via shared location info	SysReq-003	Location Information Service
IF-REQ-004	User Operation I/F	Touch Operation	Physical	Smartphone	User	Contact Pressure	N	To recognize user operations	SysReq-004	-
IF-REQ-005	App Update Notification I/F	App Update Information	Information	Smartphone	App Developer Server	Update Info	(String)	To support latest app environment	SysReq-005	App Distribution Service

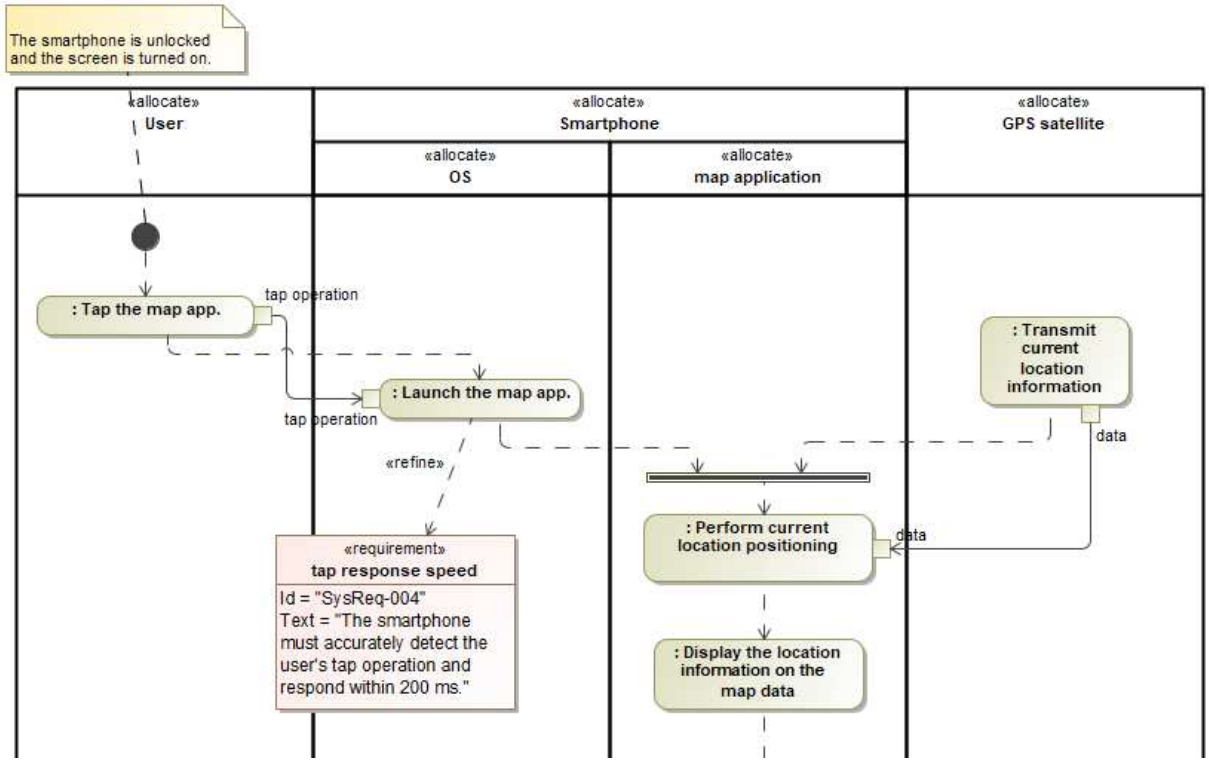

Figure 2. Limit Sample for Deliverable - System Architecture Dynamic Behavior Diagram (Partial)

In the SYS.2 deliverable Environment / External System I/F List (Table 4), all external I/F are clearly specified with I/F ID, name, content and type of exchange, input/output direction, data name, unit, and update cycle. Additionally, detailed reasons are documented, and related system requirement IDs and associated systems/services are identified. This structure allows a comprehensive overview to verify consistency, safety, and operational validity with external elements.

SYS.3 deliverable System Architecture Dynamic Behavior Diagram (Figure 2) visualizes interactions with related actors and external elements, presents the nominal flow in sequential order, and includes related system requirements and elements from static structure diagrams to ensure traceability and specification consistency.

Thus, limit samples possess a sufficient level of detail to enable even first-time users to grasp a concrete image of the deliverables. However, as mentioned above, it is essential to always verify the quality of deliverables in conjunction with the quality judgment criteria.

Verification of the Method using a specific example

This chapter verifies the effectiveness of the proposed method based on a specific example. Specifically, the *quality judgment criteria* are applied to fictional deliverables of an example system created for this study, and the effects are discussed.

The subject system is a vehicle operated by a humanoid robot through conventional steering wheel and pedal controls, enabling users to reach destinations via autonomous driving.

A generative AI was instructed to proceed with design considerations while being aware of the systems engineering process and to generate deliverables. Although fictional, these deliverables are assumed to be at a level close to those created by typical engineers and were used as the target for effectiveness verification of the method.

Examples of deliverables include Table 5 showing the system functional requirements list, Table 6 with the architecture trade-off analysis results, Figure 3 with a representative use case scenario, and Figure 4 showing the context diagram. The results of applying the quality judgment criteria to these diagrams and tables are summarized in Figure 5.

As shown in Figure 5, the quality of the deliverables was visualized, making clear which aspects should be improved to approach high-quality deliverables. Even though the deliverables were generated by a generative AI with the role set as systems engineering expert, the evaluation results included marks such as × and △, highlighting quality issues. Therefore, it is expected that general engineers who have not yet specialized in systems engineering would face similar issues when producing similar types of deliverables.

The important point here is that by revising the deliverables to change × and △ marks to ○ based on the quality judgment criteria, and thereby improving the process, anyone can smoothly practice design considerations aligned with systems engineering. Hence, this method is considered to function effectively as a system development framework.

This study verified the effectiveness of the proposed method using a fictional example. In the future, we plan to apply the method to actual projects and quantitatively measure its effects on quality and efficiency improvements. Specifically, quality indicators such as the number of review comments and missing traceability links between deliverables, as well as efficiency indicators including review time and the number of rework occurrences, will be collected and compared with conventional processes. Furthermore, by comparing these indicators before and after the application, improvement rates in quality and efficiency will be calculated. We will also analyze trends in effectiveness according to the target system and project scale, thereby statistically validating the practical effectiveness of the proposed method.

Table 5. System Functional Requirements List of the example (partial)

ID	Requirement Description	Trace Source (Use Case Scenario ID)	Trace Source (Stakeholder Requirement)	Trace Reason (Rationale)
FR-01	The system shall allow the passenger to input the destination.	UC-01-MS01	REQ-01 (Passenger-1)	To support the operation of entering the destination.
FR-02	The system shall determine the route and decision-making for the vehicle.	UC-01-MS02	REQ-01, REQ-06 (Passenger)	To enable route calculation and decision making by the vehicle.
FR-03	The system shall communicate route information to the robot.	UC-01-MS03	REQ-02 (Passenger)	To link route information to the robot.
FR-04	The system shall respond accurately to physical operations by the robot.	UC-01-MS04	REQ-03 (Passenger)	To ensure the robot physically operates the vehicle reliably.
FR-05	The system shall allow destination changes during driving.	UC-01-AS01	REQ-01 (Passenger)	To accommodate changes in the destination during driving.

Table 6. Architecture Trade-off Analysis results of the example

Evaluation Criteria	Plan A (Separation)	Plan B (Integration)	Plan C (Safety)	Plan D (External Concentration)	Plan E (Supplier Division)
Functional Independence		△	◎	◎	◎
Scalability & Maintainability	◎	△	◎	△	
Performance (Response Delay)	◎	◎	△		
Safety in Case of Failure	△	△	△	△	◎
Development Efficiency	△	◎	△	◎	◎
Supplier Coordination	◎	△	◎	△	△
Number of ECUs	(3)	(1)	(2–3)	(2)	(2)
Risk Dispersion					

UC-01: Autonomous Driving by Robot

Actors

Passenger (giving instructions), Robot (driving operation)

Preconditions

* Passenger and robot are onboard the vehicle
* Robot is seated in the driver's seat and seatbelt is fastened
* Communication link between vehicle system (navigation, CAN, etc.) and robot is established

Trigger (Start Event)

* Passenger sends "destination information" via navigation or voice input
* Passenger gives a "start driving" command to the robot (voice, operation panel, etc.)

Main Scenario

1. Passenger inputs the destination into the vehicle navigation system, and navigation performs route planning
2. Navigation information is transmitted to the robot as data
3. Robot receives the destination and route, and prepares to start driving
4. Robot operates the steering wheel and pedals to move the vehicle forward
5. Robot recognizes traffic conditions using sensors and continues driving operation
6. Robot operates the vehicle to the destination while receiving updated route instructions as appropriate

Figure 3. Use Case Scenario of the example

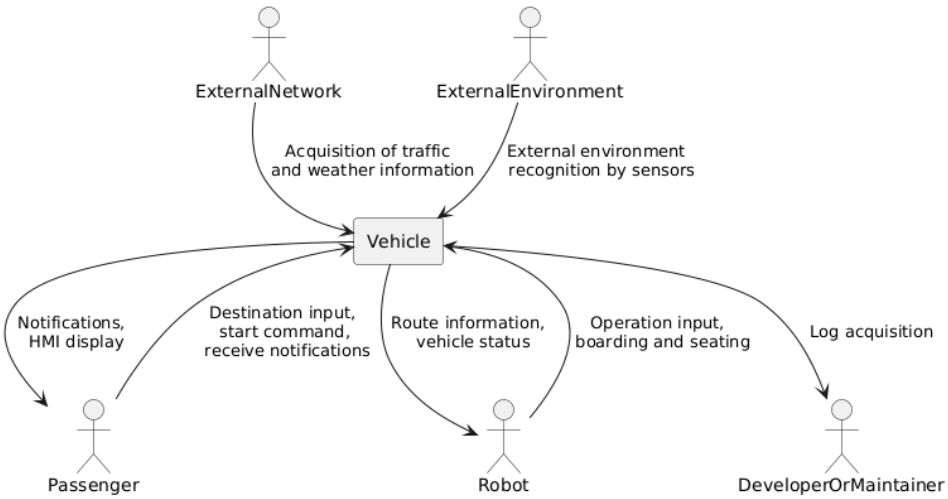

Figure 4. Context Diagram of the example

SYS.2 System Requirements Analysis

Deliverable: System Requirements List

#	Quality Judgment Criteria	Result	Reason
1	Bidirectional traceability with stakeholder requirements is ensured (no missing traces).	○	—
2	The writing style of requirement statements roughly follows EARS*. *Easy Approach to Requirements Syntax	△	—
3	The textual quality of requirement statements satisfies UnSINVAC*. *Unambiguous,Singular,Independent,Necessary,Verificable,Achivable,Complete		Partially, not all req. are written as verifiable.
4	Requirements are organized and classified by lifecycle phases (e.g., operation, maintenance, disposal).	×	Not classified by lifecycle phases.
5	The list and hierarchy (parent-child) of requirements are organized with zero conflicts, duplications, or circular references.	×	Hierarchy including parent-child relationships is not organized.
6	The conflicting relationships among the requirements are clarified.	×	The conflicting relationships among the requirements are not organized.
7	The requirements are classified (functional, non-functional, constraints) with zero unclassified items.	○	—
8	The results of the technical feasibility analysis are reflected, and risks (High/Med/Low) along with countermeasures are documented.	×	Risks and countermeasures have not been considered.

Deliverable: Environment / External System I/F List

#	Quality Judgment Criteria	Result	Reason
1	Each interface requirement clearly indicates the input/output direction (From/To) and the exchanged items (information/energy/materials, etc.).	○	—
2	It is desirable that the exchanged items include verifiable data names, units, and update cycles (e.g., wind speed [m/s], cycle = 100 ms).	×	Not described.
3	The relationship between each interface requirement and related system requirements is visualized (no missing links).	×	Not visualized, making traceability difficult.
4	The mutual dependency relationships between the system and external enabling systems/services through interfaces are clearly indicated.	○	—
5	When using external interfaces, compatibility with the counterpart's standards and protocols (e.g., CAN FD, Ethernet TSN) is explicitly specified.	×	Not clearly indicated.
6	Baseline management is implemented, maintaining histories of version, approval, and change reasons.	N/A	Not handled in this trial.

SYS.3 System Architecture Design

Deliverable: System Architecture Dynamic Behavior Diagram

#	Quality Judgment Criteria	Result	Reason
1	The purpose and scope of the diagram are explicitly stated.	×	Not described.
2	The diagram's description clearly specifies the targeted use cases and assumptions (preconditions and environmental conditions).	○	—
3	The names of diagram elements are consistent with the names in static/dynamic/requirement documents (no discrepancies).	○	—
4	Timing and performance constraints related to behavior (such as cycles and allowable tolerances) are explicitly indicated.	○	Not described.
5	The behavior diagram considers the three flows: normal, alternative, and exception (e.g., using alt/opt).	○	—
6	The behavior of parallel processing and mutual exclusion control is explicitly specified.	×	—
7	Bidirectional traceability is possible between diagram elements and related system requirements, use cases, or functional elements.	○	—

Deliverable: System Architecture trade-off analysis Results

#	Quality Judgment Criteria	Result	Reason
1	At least two or more alternative candidates have been considered.	○	5 candidates for evaluation exist.
2	5 or more evaluation criteria items have been defined based on high-priority system requirements.	○	8 items are defined and used for evaluation.
3	Both quantitative (e.g., 0–100 points) and qualitative (e.g., A to C) evaluations are conducted, with the calculation formulas clearly indicated.	×	Quantitative evaluation dominates; qualitative only for one item.
4	The basis data and rationale for the evaluation results are recorded and can be reproduced transparently.	×	The basis for evaluation results is not traceable.
5	Evaluations are weighted according to the importance of each criterion.	×	No weighting applied.
6	Baseline management is implemented, maintaining histories of versions, approvals, and reasons for changes.	N/A	Not handled in this trial.

Figure 5. Assessment Results of the Quality Judgment Criteria for the example (partial)

Expected Contributions

The expected effects of introducing this method can be summarized into three main points:

1. By clarifying evaluation criteria during reviews, subjective judgments among reviewers are minimized, ensuring transparency and accountability in system development.
2. Even when personnel or departments differ, the necessary description items and viewpoints for requirements and design are aligned. This alignment makes it easier to detect omissions such as duplication, deficiencies, or contradictions in specifications, enabling prompt correction through consensus among related departments.
3. By gathering deliverables whose quality is assured through this method, a high-quality system requirements specification document can be created. For example, by integrating the past asset information organized in SYS.0, stakeholder requirements and their rationale extracted in SYS.1, high-quality system requirements defined in SYS.2, architecture and dynamic behavior designed in SYS.3, and verification method specifications formulated in SYS.4 to SYS.6 comprehensively, it is believed that system requirements specification document following a typical chapter structure can be created with enhanced quality.

On the other hand, for this method to work effectively, it is essential to ensure users have the necessary skills from the introduction phase. In addition, it is necessary to address organizational issues such as disregarding the quality judgment criteria, even when such criteria are provided as described in this paper. If the criteria and limit samples are not regularly reviewed and updated, they may become outdated and could hinder quality assurance.

To overcome these challenges, it is essential to foster internal assessor personnel and to promote understanding of the significance and necessity of the quality judgment criteria among on-site staff and development management. Therefore, it is necessary to introduce this method in conjunction with certification program and activities to promote understanding of assessment, while systematically operating the continuous improvement of the quality criteria and limit samples within the organization.

It should be noted that while this method is effective for complex system development processes such as those in the automotive industry, significant modifications to the quality criteria items and limit samples are necessary for software-centric systems with substantially different deliverable formats and description styles, as well as for prototyping in research and development phases. In other words, this method is not a one-size-fits-all framework applicable uniformly across all industries; rather, adjustments tailored to the characteristics of each domain are indispensable.

Conclusion

This paper proposed an evaluation method combining *quality judgment criteria* that enable objective assessment of deliverable quality across the SYS.0 to SYS.6 process stages, and *limit samples* that complement these criteria. The quality judgment criteria are described with sufficient detail to facilitate easy judgment by even first-time users, ensuring objectivity and transparency in evaluation. The limit samples present concrete examples that meet these criteria in tabular or graphical form, enabling users to clearly visualize what constitutes high-quality deliverables.

Through verification using fictional deliverables of an example created for this study, it was confirmed that this method can clarify the quality issues in deliverables.

This method minimizes subjective judgments during reviews and enables sharing of deliverable quality among different personnel and departments, allowing early detection and correction of omissions and inconsistencies. Furthermore, by gathering deliverables whose quality is assured, it is expected to contribute to improving the overall quality of the system requirements specification documents.

In the future, we intend to apply this method to actual vehicle development projects, conducting quantitative evaluations such as reduction of deliverable quality variability and improvement of review efficiency, while iteratively improving through cycles of refining quality criteria items and limit samples as well as through user training.

This aims to enhance the overall design capability within the organization's system development and promote deliverable assessment activities.

Ultimately, while maintaining alignment with international standards and industry norms, we aim to further establish the effectiveness and versatility of this method.

References

INCOSE. (2023). Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities. Fifth Edition.

VDA/QMC. (2023). Automotive SPICE® process assessment model

IPA, Japan. (2007). Embedded system development process reference guide.

ISO. (2015). ISO/IEC/IEEE 15288:2015, Systems and Software Engineering - System Life Cycle Processes. International Organization for Standardization

Biography

Seiji Honda

Co-chair of JCOSE Automotive working group from 2024 to present, and MBSE Deployment Group Manager in Toyota Motor Corp. in Japan. Certified: INCOSE CSEP