General

JERG－0－049D（E）

SOFTWARE DEVELOPMENT STANDARD

Mar. 29 , 2024

Japan Aerospace Exploration Agency

The official version of this standard is written in Japanese. This English version is issued for convenience of English speakers. If there is any difference between Japanese version and English one, the former has precedence.

^

This is an English translation of JERG-0-049D.
If there is anything ambiguous in this document, the original document (the Japanese version) shall be used to clarification.

Published by
Japan Aerospace Exploration Agency
Safety and Mission Assurance Department
2-1-1 Sengen Tsukuba-shi, Ibaraki 305-8505, Japan

JERG-0-049D
SOFTWARE DEVELOPMENT STANDARD

1. JAXA JERG-0-049D is hereby established.

2. For any questions concerning this standard, contact the Safety and Mission Assurance Department of the Japan Aerospace Exploration Agency (JAXA).

Establishment: Safety and Mission Assurance Department Director
March 29, 2024

Table of Contents

1 Scope

This standard applies to the activities relating to the development, operation, and maintenance of software for satellites, probe, launch vehicle and ground systems, and the activities needed for system development and related support. When this standard is applied, embodying and tailoring may be performed in accordance with the characteristics of specific projects and other factors.

In principle, this document does not define the categories of personnel who implement processes. These personnel shall be defined by individual contracts or other agreements to which this standard is applied.

2 References

2.1 Informative references

(1) ISO/IEC/IEEE 12207:2017 Systems and software engineering -- Software life cycle processes
(2) JIS X0160: 2021 Software life cycle processes
(3) ISO/IEC 14764:2006 Software Life Cycle Processes - Maintenance
(4) JIS X0161:2008 Software Engineering - Software life cycle processes - Maintenance
(5) ISO/IEC 33004:2015 Information technology -- Process assessment – Requirements for process reference, process assessment and maturity models
(6) ISO/IEC 33020:2015 Information technology -- Process assessment – Process measurement framework for assessment of process capability
(7) JIS X 33020:2019 Information technology -- Process assessment -- Process measurement framework for assessment of process capability
(8) Software Life Cycle Processes - Japan Common Frame 2013 (Japanese) (Copyright IPA/SEC 2013)
(9) ISO 9000: 2015 Quality management systems—Fundamentals and vocabulary
(10) JIS Q 9000: 2015 Quality management systems—Fundamentals and vocabulary
(11) JMR-004 Reliability Program Standard
(12) JERG-0-063 Space Development Reliability Technical Handbook (Japanese Only)
(13) JMR-001 System Safety Standard

3 Terms, definitions and abbreviated terms

Term	Definition
Acceptance inspection and acceptance testing	Acceptance inspection and acceptance testing are actions to evaluate the compliance with requirements at the time of acquiring software products. Inspection is an action to confirm the compatibility of a product, in accordance with evaluation criteria based on either requirements specifications or a predetermined value, by visually checking quantities and test results. Test comprises analysis, evaluation and checking of the functionality and capabilities of the software in order to obtain evaluation results and data required for use in inspection.
Activity	Activity is a component of a process and a set of strong correlative tasks.

Term	Definition
Adaptive maintenance	The modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment NOTE—Adaptive maintenance provides enhancements necessary to accommodate changes in the environment in which a software product must operate. These changes are those that must be made to keep pace with the changing environment. For example, the operating system might be upgraded and some changes may be made to accommodate the new operating system. (ISO/IEC 14764: 2006)
Architecture	Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution (ISO/IEC/IEEE 12207:2017)
Assessment	Assessment is a process to evaluate the strengths and the weaknesses of a subject process and to identify opportunities to improve the process for some predetermined purpose.
Computer system	Computer system is the entire system consisting of sets of software, platforms, and hardware, including the platform and hardware that are able to execute the targeted software for development. Although the definition of what a computer system contains is arbitrary, the definition shall be unique for software products that are subjected for development. As a computer system may be defined in various ways, from a one-chip microcomputer to multiple general-purpose computers connected to a network.
Configuration management	Configuration management is an action to define the configuration items, i.e. computer systems or projects, to record changes of content and to manage such aspects as their storage, handling, and distribution. If the software consists of multiple modules, then not only must the software version be managed, but the version of each software module must also be managed. In addition, configuration management items require software consistent modules, requirements specifications, operation manuals, and so on.
Corrective maintenance	The reactive modification of a software product performed after delivery to correct discovered problems NOTE—The modification repairs the software product to satisfy requirements (ISO/IEC 14764: 2006)
COTS	COTS is the abbreviation of Commercial Off-The-Shelf. It has already been developed and is available in the commercial marketplace.
CIL	CIL is the abbreviation of Critical Item List. It is a list of critical items specified by the reliability analysis. (Refer to JMR-004 para. 4.3.16.2.2)
Cyclomatic complexity	It is an indicator of the complexity of the program logic and is calculated from the number of vertices (n), number of edges (e) and number of connected program elements (p) in a graph representation of the program structure, as follows: e-n+2p

Term	Definition
Enabling system	System that supports a system-of-interest during its life cycle stages but does not necessarily contribute directly to its function during operation (ISO/IEC/IEEE 12207:2017) Note 1 For example, when a system-of-interest enters the production stage, a production-enabling system is required. When the target system is a spacecraft installation software, if speaking throughout the life cycle, the enabling system means such as an emulator, or simulator. Note 2 Each enabling system has a life cycle of its own. This Standard is applicable to each enabling system when it is treated as a system-of-interest.
Identifier	Identifier is a short line of numbers, letters, and symbols, which is appended to each item of output and input to enable each item to be identified and classified by item type. Identifiers shall consist of not only a set of numbers, but also a set of letters and symbols. In addition, it is not always necessary identifiers be serial numbers. Appending a unique identifier to each item is convenient for requirements management and for traceability.
Incident	Anomalous or unexpected event, set of events, condition, or situation at any time during the life cycle of a project, product, service, or system (ISO/IEC/IEEE 12207:2017)
Independent Verification and Validation: IV&V	IV&V are the verification and the validation that are performed by the organization independent of the software development organization. With regard to independence, financial, technical and management viewpoints shall be considered.
Input	Input is information needed to implement an activity.
Integrity	Integrity is defined as the following properties in this standard: (1) Software component is complete with no deficiencies. (2) Software component is at an appropriate version.
Interoperability	A series of exchanges among multiple functions to accomplish a specified objective.
Knowledge asset	Generic name of assets including reusable software items, reusable code libraries, reference architectures, design elements (such as architectures or design patterns), processes, criteria, other technical information related to domain knowledge (such as training materials), lesson learned, and environments reflecting the knowledge of the organization (enabling systems or services) (Refer to Appendix VI.)
Mission	A specific plan or activity, including the final status or results to be obtained at the end of the project (e.g. at the end of the regular operation period) or through use, research, etc. after the end of the project.
Model based technology	A development method, an abstraction of system/software, using a model representing target system/software with one view point or one abstraction level
Module	The smallest unit in software development. Some modules work on their own, while others work in combination with other modules.
Non-functional requirements	Non-functional requirements are all requirements except functional requirements, such as performance, safety, and reliability.
Operation	Operation is an action to carry out missions for the achievement of a purpose by means of an appropriate computer system. An operation utilizes the computer system from beginning to end, and includes monitoring and maintenance functions, and so on.
Output	Output is information that is transformed from input by performing activities.

Term	Definition
Perfective maintenance	The modification of a software product after delivery to detect and correct latent faults in the software product before they are manifested as failures NOTE—Perfective maintenance provides enhancements for users, improvement of program documentation, and recoding to improve software performance, maintainability, or other software attributes (ISO/IEC 14764: 2006)
Preventive maintenance	The modification of a software product after delivery to detect and correct latent faults in the software product before they become operational faults (ISO/IEC 14764: 2006)
Problem	Difficulty, uncertainty, or otherwise realized and undesirable event, set of events, condition, or situation that requires investigation and corrective action (ISO/IEC/IEEE 12207:2017)
Process	Process is a set of interrelated or interacting activities to transform input to output.
Project	Project is a time-limited endeavor to be implemented by means of specified resources and a temporary organization, with the purpose of fulfilling the project's mission.
Requirement	Requirement is one of a set of functions and performance targets requested for computer system or software and they may be also included such as not embodied and not detailed enough, or ambiguity in expression and vague expectations.
Requirements specification	Requirements specification is defined as the description of functions and performance required for computer system or software, embodied and specifically defined, and which also consider feasibility. In principle, the specified requirements specifications shall be verifiable as both feasible and mutually consistent. However, on the characteristic of adopted development process and required functions and performance, if the requirements specifications representative format is not a feasible verification format, the verification of feasibility of the requirements specification shall be complemented by the following methods: (1) It shall include the planning of agreement procedure with computer system users that the requirements specifications are satisfactory, and software verification plan shall include the planning of agreement procedure. (2) It shall include the test specifications enough to verify the requirements in the verification plan. In principle, any restrictions, laws, rules, and a project policy shall be included in the requirements specifications.
Risk	Risk is defined as the degree of danger attaching to a system's safety and surrounding projects. It includes assessment of undesirable outcomes which may occur as a result.
Service	Service is defined as a provider of functions and operations to users and systems.

Term	Definition
Software	Software is a set of computer system configuration items comprising commands and data, which are executed or processed by a CPU to fulfill functions and capabilities defined in the software requirements specification. If it is a set of commands and data which are implemented or managed by a CPU, it is categorized as software, and the software development process standards apply to it. However, for the driver of firmware, OS, and middleware, appropriate development processes are applied, in accordance with the characteristics and therefore may be removed from the software development process standard. For example, hardware and its integrated driver development shall be considered as such a case.
Software life cycle	Software life cycle is the period from the beginning of the requirements analysis phase until the termination of use of the software.
Software products	Software products are the set of software, source code, and related documentation.
Software test specifications	Software test specifications are defined as the descriptions of test conditions and expected results, expressed unambiguously, to prove that software meets the requirements specifications. If the requirements specifications are represented in a verifiable format, they may be treated as appropriate software test specifications.
Software under test	Software under test is software that is being subjected to testing and inspection.
Software user's manual	Software user's manual is the set of information a user needs in order to use software. It includes operation unit manuals, computer system operation manuals, and work operation manuals.
Software test plan	Software test plan is a documentation of the following for the identified items whose functions and performances are to be tested and verified in the software verification plan. ・Testing objectives ・Software items for testing ・Test configuration ・Test facilities ・Schedules ・Test structure
Software verification plan	Software verification plan is a documentation of the scope, content, method, environment (such as test equipment) and schedule pertaining to the verification of software development. A validation plan may be included.
Source code	The source of the generation of software (so-called object code). It is typically described as programming language suitable for reading and writing by humans.
Strategy	Policies to consider in planning an execution plan aimed at effective utilization of specific resources and time to achieve organizational business objectives or project missions. (Refer to Appendix VIII.)
System	A set of organized functional elements (hardware and software), such as launch vehicles, satellites, and ground equipment. The systems are combined comprehensively to achieve the mission.
Tailoring	Tailoring is defined as the activity to change processes defined in this standard in order to meet the project's particular characteristics and to establish an appropriate framework for each system development project.
Tasks	Tasks are components of activities corresponding to each stage of work.

Term	Definition
Test coverage	One of the quantitative quality indicators for testing and code, the following test coverages are used in this document. C0: Instruction coverage (statement coverage), this is the test coverage where every statement in the code is executed at least once. C1: Branch coverage (decision coverage), this is the test coverage where every branch in the code is executed at least once. MC/DC: Modified Condition/Decision Coverage, this is the test coverage that executes tests that satisfy the following: (1) Each entry and exit point are invoked. (2) Each decision takes every possible outcome (3) Each condition in a decision takes every possible outcome (4) Each condition in a decision is shown to independently affect the outcome of the decision.
Test plannability	Test plannability is defined as the aspect of test specification descriptions that indicates the possibility of testing and planning using the appropriate development phases and test environment for target test items. (Refer to Appendix IX.)
The stability (maturity) of software requirements specification	The stability (maturity) of software requirements specification is defined as the index which shows the possibility of specification change is small because of the software requirements specification be extracted and analyzed sufficiently. The definition of the index, and how it is evaluated, are arbitrary. Generally, provision for essential or refined changes to software requirements specification affect process cost, delivery date and quality which let the software requirements specification inputs. It is hoped that the index which is used to evaluate this shall be selected based upon the stability and maturity of software requirements specification.
Traceability	Traceability is the capability to trace multiple inputs and outputs. These include the correspondences between higher and lower-level specifications as well as between design specifications and source code, requirements specifications for the computer system, and computer system integration test specifications.
Validation	Validation is a process. It uses objective evidence to confirm that the requirements which define an intended use or application have been met. Whenever all requirements have been met, a validated status is achieved. The process of validation can be carried out under realistic use conditions or within a simulated use environment. [ISO9000]
Verification	Verification is a process. It uses objective evidence to confirm that specified requirements have been met. Whenever specified requirements have been met, a verified status is achieved. [ISO9000]
Verifiability	It indicates the capability of establishing verification criteria for requirements specifications and implementing verifications including tests and analyses to confirm that these criteria are met.
Configuration audit	A third person, who differs from personnel implementing configuration management activities, confirms the applicability of the configuration management activities to the configuration management plan.
Waiver	Refers to the case of accepting a system or component item as is despite a nonconformance to a requirement of the configuration identification document which occurred after the start of its fabrication or accepting it after repaired by an approved method. (JMR-006)

4 Organization of this standard

This standard categorizes the software life cycle into three primary life cycle processes and eight supporting life cycle processes, and defines these processes. The definition of these processes is shown in Figure 4.1 and Table 4.1.

The primary life cycle processes are processes in a software life cycle directly related to the development of target software, and consist of processes implemented during development, operation, or maintenance. The supporting life cycle processes are processes that indirectly affect the software life cycle process, with reference specifically to the development of object software, and act to support a primary life cycle process and are called by other processes, as necessary.

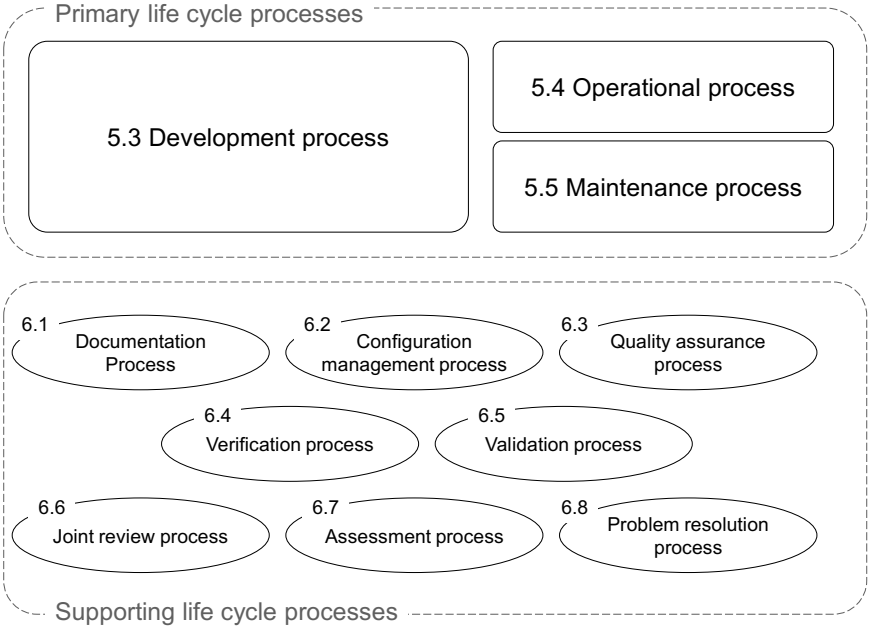


Figure 4.1 - Structure of the standard

The processes may be implemented in an order different from their clause number order in this document. Also, note that there will be cases where identical activities are described in multiple processes. For example, activities relating to software verification may be described as part of the development process, which is a primary life cycle process. These are also activities relating to the verification process, which is a supporting life cycle process.

The classification of processes is better understood as a classification according to the different viewpoints. This standard aims to define processes from various viewpoints, so that all of the required contents is covered completely (duplication is allowed). Therefore, this standard assumes that it will be applied after appropriate concretizing and tailoring have been performed regarding the relevant processes.

Table 4.1 Process list

Process		Description
Primary life cycle processes	5.1 Not used	
	5.2 Not used	
	5.3 Development	Process to be implemented from the viewpoint of development. Requirements analysis, design, coding and testing, installation on target platforms (embedding), supply, introduction, acceptance, and so on.
	5.4 Operation	Process to be implemented from the viewpoint of operation. Drafting plans and rules for operations, operational testing, operation, user support, and so on.
	5.5 Maintenance	Process to be implemented from the viewpoint of maintenance. Drafting plans and rules for maintenance, problem identification, modification, retirement, and so on.
Supporting life cycle processes	6.1 Documentation	Process regarding the record of the outcomes of individual processes.
	6.2 Configuration management	Process regarding the management of software and documents.
	6.3 Quality assurance	Process regarding the confirmation that the process meets the requirements of this standard and processes are managed according to the plan.
	6.4 Verification	Process regarding the confirmation that specified requirements have been fulfilled, based on the provision of objective evidence.
	6.5 Validation	Process regarding the confirmation that the requirements for a specific intended use or application have been fulfilled, based on the provision of objective evidence.
	6.6 Joint review	Process regarding a joint review means it is conducted by multiple personnel with different viewpoints.
	6.7 Assessment	Process regarding the confirmation of the executing status and identifying the items which need to be improved.
	6.8 Problem resolution	Process regarding resolving problems which occur during implementation.

4.1 Tailoring

This standard is intended to be applied after appropriate concretization and tailoring of the process. Tailoring shall be performed according to the following procedures:

(1) Set the criticality class (hereinafter referred to as SW CC of the software to be developed) in accordance with Appendix XI.

(2) Tailor the requirements of this standard based on the requirements mapping matrix shown in Table-Appendix XII, taking into account the project characteristics in addition to the SW CC established in (1).

5 Primary life cycle processes

This clause defines the following primary life cycle processes:

(1) Not used
(2) Not used
(3) Development process
(4) Operation process
(5) Maintenance process

5.1 Not used

5.2 Not used

5.3 Development process

The development process is a collective process of defined activities, inputs, and outputs comprising the following processes:

(1) computer system requirements analysis process;
(2) computer system architectural design process;
(3) software requirements analysis process;
(4) software design process;
(5) software coding and testing process;
(6) software integration process;
(7) software integration test process;
(8) software installation into target platforms (embedding) process;
(9) computer system integration and computer system integration test process;
(10) supply and introduction of software product process;
(11) software acceptance process.

These processes and activities may be implemented in a different order from what is described in this document. However, the overall configuration of the processes shall be defined as well as the management method of the entire process, so that appropriate process management is performed. In performing the processes, the use of the model based technologies shall be considered, as necessary.

5.3.1 Process implementation

When software development is started, activities that meet the following requirements shall be performed:

(1) Define a development strategy including the following.
  (a) Development policies and rules
    (i) Suitable safety, security, privacy, and policy for environment activity
    (ii) Programming and coding standard
    (iii) Unit testing policy
  (b) In case of software reuse, conformation method of the applicability to the computer system and the safety of the acquisition route
  (c) Performing of the software construction, peer review, and walkthrough review
  (d) In case change management is conducted by not using any tool, how the configuration management is performed during software coding and testing
  (e) Priorities in transitions of software and related data accompanied by computer system disposal
  (f) Knowledge asset
    (i) Obtainment and maintenance plans in the useful life of knowledge assets
    (ii) Criteria for accepting, qualifying, and retiring knowledge assets
    (iii) Procedures for controlling change in knowledge assets
    (iv) Plans, mechanisms, and procedures for protecting, controlling, and accessing classified or sensitive data and information

(2) Based on the development strategy, the software development plan including the following information shall be established to cover:
  (a) Purpose and constraints of the software development
  (b) Scope of computer system
  (c) Identification of target software
  (d) Definition of the identification of software development processes and their relationship (operation process, maintenance process, and so on are considered)
  (e) Definition of software development processes and activities (*)
  (f) Activities including roles, authorities, and responsibilities in each individual development process implementation management plan
  (g) Review plan
  (h) Preparation of development related documents' structure, and relationships of input and output in each development process
  (i) Documentation plan, including development department and schedule
  (j) Appropriate work allocations and methods including work plans (schedules, milestones) and achievement criteria, and progress management for each work
  (k) Obtainment of an environment to be used for software development and verification (simulator, real hardware, test environment, and so on, enabling systems or services) or acquisition of access to the environment
  (l) Management plan for COTS and knowledge assets used. The following shall be included:
    (i) Identification of COTS items and knowledge assets
    (ii) Definition of the quality assurance process regarding COTS items and knowledge assets

(m) Evaluation plan of applicability with computer system

(3) Software development plan shall be documented and approved.

* Including activities regarding computer system with software installed.

5.3.1.1 Output

(1) Software development plan

5.3.2 Items to be applied to all processes

5.3.2.1 Activity

Activities that meet the following requirements shall be performed throughout the entire development process:

(1) The software development plan shall be updated and managed in accordance with the development status.

(2) The progress of software development shall be monitored. It shall be reported to administrators as necessary.

5.3.2.2 Input

(1) Software development plan

5.3.2.3 Output

(1) Software development plan (updated)

(2) Software development progress report

5.3.3 Computer system requirements analysis

5.3.3.1 Activity

Activities that meet the following requirements shall be performed with respect to the computer system requirements analysis:

(1) Requirements extraction

(a) The functional boundary of the developed computer system shall be clarified.

(b) The requirements for the computer system to be developed, the operational concept shall be analyzed, and operational scenarios shall be documented.

(c) The state transition (including operation mode) necessary for the developed computer system shall be identified.

(d) If another state transition is defined separately for a system related to the computer system to be developed, the relationship between the state transitions of the systems shall be clarified (refer to Appendix VII).

(2) Requirements specifications development

(a) Feasibility and consistency shall be confirmed, based on the operational scenarios, and the requirements specification for the computer system shall be defined.

(b) Specifications for data and databases to be handled by the computer system shall be included in the requirements specification.

(c) Risks, computer system severity, and specifications for important quality characteristics shall be included in the requirements specification.

(d) Interface requirements shall be analyzed, and requirements specifications shall then be developed. Agreement with the relevant parties on the interface requirements shall be made

based on a common understanding and interpretation of the contents.

(e) The rationale for the requirements specification for the computer system shall be clarified, and the traceability of higher level requirements for the computer system shall be evaluated and maintained.

(f) Rationale and consideration methods of the individual requirement of the requirements specifications shall be clarified, and their feasibility shall be evaluated.

(g) If COTS or reused software is used, its applicability with the requirements specifications shall be analyzed.

(h) In addition to the feedback of the requirement analysis contents to appropriate stakeholders, developed requirements specifications shall be reviewed and approved by the stakeholders.

(i) Problems, inadequacies, and inconsistencies included in the requirements specifications shall be identified and planned to resolve them.

(j) Verifiability of individual requirement of the requirements specifications shall be evaluated.

5.3.3.2 Input

(1) Requirements for the computer system

(2) Operational concept

5.3.3.3 Output

(1) Operational scenarios

(2) Requirements specification for the computer system

(3) Interface specifications

(4) Evaluation results of the traceability of the requirements specifications and requirements for the computer system

(5) Rationale of the requirements specification for the computer system and its feasibility evaluation results

(6) COTS or reused software applicability evaluation results

(7) Evaluation results of the verifiability of the requirements specification for the computer system

5.3.4 Computer system architectural design

5.3.4.1 Activity

Activities that meet the following requirements shall be performed for the computer system architectural design:

(1) Computer system architecture shall be designed based on the requirements specification for the computer system and the operational scenarios. Configuration items and their various categories (hardware, firmware, software, and operational) shall be clarified.

(2) Requirements pertaining to the requirements specification for the computer system shall be allocated among the individual configuration items of the system.

(3) Computer system architectural design specifications shall be developed by combining the results of the above-mentioned design.

(4) Feasibility of software items in fulfilling their allocated requirements shall be evaluated.

(5) Rationale for the design and preconditions (e.g. operational assumptions) for the computer system architectural design specifications shall be identified, and an appropriate evaluation shall be performed.

(6) Traceability of the computer system architectural design specifications relative to higher level requirements, such as the requirements specification for the computer system, shall be evaluated.
(7) Interface requirements for the software shall be extracted.
(8) Set the evaluation criteria for a computer system architectural design based on the requirements specification for the computer system and operational scenarios. The computer system architecture design shall be evaluated by that. Additionally, the computer system architectural design selection rational shall be recorded.

5.3.4.2 Input

(1) Operational scenarios
(2) Requirements specification for the computer system

5.3.4.3 Output

(1) Computer system architectural design specifications
(2) Requirements for the software, including operational scenarios after the analysis
(3) Interface requirements
(4) Evaluation results of traceability relative to architectural design specifications with the computer system and requirements specification for the computer system
(5) Evaluation results of the computer system architectural design and rationale for the selection of the computer system architectural design

5.3.5 Software requirements analysis

5.3.5.1 Activity

The following activities shall be performed for the software requirements analysis:

(1) Software requirements specification shall be developed, based upon the analysis of the computer system architectural design specifications, interface requirements, and requirements for software including non-functional requirements.
(2) The required software state transition (including operation mode) shall be identified.
(3) If another state transition is defined separately for a system related to the software to be developed, the relationship between the state transitions shall be clarified (refer to Appendix VII).
(4) Identifiers shall be included in the individual requirement of software requirements specification.
(5) Specifications for data and databases to be handled by the software shall be included in the software requirements specification.
(6) Specifications for failure detection and handling functions shall be included in the software requirements specification.
(7) Risks, software severeness, and specifications for important quality characteristics shall be included in the software requirements specification.
(8) User interface (in case having), information provided to users, and specifications for user trainings shall be included in the software requirements specification.
(9) In case of a software transition to an on going system, specifications for the satisfaction of the transition condition shall be included in the software requirements specification.
(10) Interface requirements shall be analyzed, and interface specifications shall then be developed. Agreement with the relevant parties regarding the interface specifications shall be made based on a common understanding and interpretation of the contents.

(11) Traceability and consistency of the software requirements specification relative to the computer system architectural design specifications and interface requirements shall be analyzed and documented.
(12) Rationale of individual requirement of the software requirements specification shall be clarified, and their feasibility shall be evaluated.
(13) If COTS or reused software is used, compliance with the software requirements specification and its applicability with the computer system architectural design specifications shall be analyzed.
(14) Operational assumptions and constraints regarding software requirements specification shall be extracted.
(15) In addition to the feedback of the requirement analysis contents to appropriate stakeholders, developed software requirements specification shall be reviewed and approved by the stakeholders.
(16) Problems, inadequacies, and inconsistencies included in the software requirements specification shall be identified and planned to resolve them.
(17) Verifiability of the individual requirements of the software requirements and interface specifications shall be evaluated, and a software verification plan, including the validation method, shall be established.
(18) Software verification coverage pertaining to software function, performance, and operational scenarios in the verification plan shall be evaluated, and test plannability regarding the software requirements specification and interface specifications shall be evaluated.
(19) With regard to the software verification plan, whether the test is affected by the behavioral difference between the test environment and the real hardware, or whether verification is performed by review, analysis and so on, without testing, the evaluation that shows the adequate identification and verification methods shall be included.

5.3.5.2 Measurement

In terms of evaluating stability and quality levels of software requirements specification, the following measurement shall be performed for software requirements analysis:

(1) The definition of the data to be collected for evaluating the stability (maturity) of the software requirements specification and their evaluation methods shall be defined. This measurement result is used as input for the project management, and is also referenced data for the post-project.
(2) Collection and evaluation of data defined in (1) above shall be planned.
(3) Collection and evaluation of data defined in (1) above shall be performed, and the results recorded.

5.3.5.3 Input

(1) Computer system architectural design specifications
(2) Requirements for software, including operational scenarios after the analysis
(3) Interface requirements

5.3.5.4 Output

(1) Software requirements specification
(2) Interface specifications
(3) Software requirements specification traceability and consistency evaluation results
(4) Software requirements specification rationale and their feasibility evaluation results

(5) COTS or reused software applicability evaluation results
(6) Operational assumptions and constraints
(7) Software verification plan, including validation plan
(8) Verification coverage and test plannability for the software verification plan evaluation results
(9) Software requirements stability (maturity) evaluation results
(10) Software requirement analysis measurement results

5.3.5.5 Review

For each output, a software requirements review shall be performed. Items to be reviewed shall be chosen based on 5.3.5.4 and shall be defined in a plan document such as the software development plan document. Appropriate follow-up of action items shall be performed in accordance with due dates, follow-up status, degree of influence, and so on. If a review is performed, it shall be documented in a technical review record after its completion. In addition, quantitative data such as the reviewers' positions, their review time, the number of questions, and their comments shall be recorded, and the quality of the review shall be evaluated.

5.3.6 Software design

In this standard, software architectural design and detailed design are not separated especially. However, in an actual development process, the software architectural design and the detailed design phases may be separated as necessary.

5.3.6.1 Activity

Activities that meet the following requirements shall be performed for the software design:

Software architectural design

(1) The design guidelines (software architecture, etc.) and the design characteristics to be considered shall be selected, and the design shall be performed by considering priorities.
(2) Functional decomposition and module partitioning shall be performed based on the software requirements specification and the relationships between the modules comprising the functions, and the structures between modules and themselves shall be clarified, so that an appropriate software architectural design is performed.
(3) Software architectural design shall include the design and distribution of non-functional requirements (processing time requirements, requirements of resources such as memory, and so on) and be defined as software requirements.
(4) Interface specifications shall be detailed in accordance with the considerations of the boundaries and interrelations, and the decomposition of the software functions and modules. Agreement with the relevant parties as to the interface specifications shall be arrived at based on a common understanding and common interpretation of the contents.

Software detailed design

(5) Each individual module shall be designed in accordance with the decomposition of functions and modules, and a software detailed design shall be performed.
(6) Interface specifications shall be detailed in accordance with the considerations of the boundaries and interrelations, and the design of the module. Agreement with the relevant parties on the interface specifications shall be made based on a common understanding and interpretation of the contents.

Common to software architectural and detailed designs

(7) Software (architectural and detailed) design specifications shall be developed based on the result of the software design.
(8) The needed design methods or organizationally maintained past knowledge shall be identified, prepared and acquired.
(9) Traceability and consistency of the software design with the software requirements specification, interface specifications, and with the necessary related documents shall be analyzed, documented and maintained.
(10) Operational assumptions and constraints regarding the software design shall be identified.
(11) As necessary, with regard to the individual software design, the design rationale shall be clarified and its feasibility and testability (including test case) evaluated. (Refer to Appendix IX.)
(12) If COTS or reused software is used, its applicability with the software design shall be analyzed.
(13) Software test plans and specifications shall be established in accordance with the software verification plan.
  (a) For software test specifications, the following shall be considered:
    (i) Operational scenarios
    (ii) Interface specifications
    (iii) Maximum load for assumed scenarios
    (iv) Coverage for software requirements specification and software design specifications
    (v) Anomalies, such as exceptions and failures
    (vi) Applicability of COTS or reused software items with computer system
(14) If new operational assumptions and constraints arise or are identified, they shall be updated.

5.3.6.2 Measurement

The following measurement shall be performed for software design in order to evaluate the progress risk of the software design:

(1) Definition of the data to be collected and of the evaluation methods for progress management and risk evaluation of software design shall be defined.
(2) Collection and evaluation of data defined in (1) above shall be planned.
(3) Collection and evaluation of data defined in (1) above shall be performed, and the results recorded.

5.3.6.3 Input

(1) Software requirements specification
(2) Interface specifications
(3) Operational assumptions and constraints
(4) COTS or reused software applicability evaluation results
(5) Software verification plan

5.3.6.4 Output

(1) Software (architectural and detailed) design specifications
(2) Interface specifications (updated)
(3) Software design traceability and consistency evaluation results
(4) Software design rationales and their feasibility evaluation results
(5) COTS or reused software applicability evaluation results (updated)
(6) Operational assumptions and constraints (updated)

(7) Software test plan
(8) Software test specifications
(9) Software design measurement results

5.3.6.5 Review

For each output, a software design review shall be performed. Items to be reviewed shall be chosen based on 5.3.6.4 and shall be defined in a document such as the software development plan document. Appropriate follow-up on action items shall be performed in accordance with due dates, follow-up status, degree of influence, and so on. If a review is performed, it shall be documented in a technical review record after its completion. In addition, quantitative data such as the reviewers' positions, their review time, the number of questions, and their comments shall be recorded, and the quality of the review shall be evaluated.

5.3.7 Not used

5.3.8 Software coding and testing

5.3.8.1 Activity

Activities that meet the following shall be performed for the software coding and testing:

(1) Source codes shall be developed based on constrains, the software design specifications and interface specifications, and reviewed.
(2) Definitive implementation guidelines for error handling shall be considered.
(3) Source code shall be developed based on the defined coding standard.
(4) Static analysis shall be performed with a source code checking tool or equivalent, and the source quality shall be evaluated. In addition, the Cyclomatic Complexity criteria in Table 5.3.8-1 shall be satisfied.

Table 5.3.8-1 Cyclomatic Complexity of Modules

SW CC	A	B	C	D
Cyclomatic Complexity	15 or lower	15 or lower	15 or lower	20 or lower

(5) Unit testing specifications shall be developed in accordance with a software verification plan and a software test plan.
(6) Unit testing shall be performed in accordance with the unit testing specifications, and the test results shall be recorded in a format that it allows determination of pass or failure.
(6) For unit testing, the criteria in Table 5.3.8-2 for the test coverage for source code shall be established and the test shall be performed so that these criteria are satisfied. If the test coverage criteria cannot be achieved, analysis, inspection, or design review shall be applied to the untested code.

Table 5.3.8-2 Test Coverage for Source Code

SW CC	A	B	C	D
C0 : Instruction coverage	100%	100%	※1	※1
C1 : Branch coverage	100%	100%	※1	※1
MC/DC : Modified Condition/ Decision Coverage	100%	※1	※1	※1

*1: Criteria should be established and agreed upon according to the characteristics of the project.

(8) The traceability of the source code and software design specifications shall be analyzed and the results shall be recorded. The correspondence between the naming convention (variable name, function name, etc.) in source codes and the design specifications shall be checked.
(9) If new operational assumptions and constraints arise or are identified, they shall be updated.

5.3.8.2 Measurement

The following measurement shall be performed for software coding and unit test in order to evaluate the quality:

(1) The definition of data to be collected for evaluating source code quality, and the evaluation method, shall be defined.
(2) Collection and evaluation of data defined in (1) above shall be planned.
(3) Collection and evaluation of data defined in (1) above shall be performed and the results recorded.
(4) Results of the evaluation in (3) above shall be reported periodically, or for each milestone.

5.3.8.3 Input

(1) Software design specifications
(2) Interface specifications
(3) Operation assumptions and constraints
(4) Software verification plan
(5) Software test plan

5.3.8.4 Output

(1) Source code
(2) Operation assumptions and constraints (updated)
(3) Unit testing specifications
(4) Unit testing record
(5) Traceability analysis record
(6) Software coding and testing measurement results

5.3.8.5 Review

For each output, a software coding and testing review shall be performed. Appropriate follow-up shall be performed with regard to action items in accordance with due dates, follow-up status, degree of influence, and so on. If a review is performed, it shall be documented in a technical review record after its completion. In addition, quantitative data such as the reviewers' positions, their review time, the number of questions, and their comments shall be recorded, and the quality of the review shall be evaluated.

5.3.9 Not used

5.3.10 Software integration

5.3.10.1 Activity

Activities that meet the following shall be performed for the software integration:

(1) Based on objectives, the integration criteria and verification points for software functions (operations of correct interfaces and completeness) shall be selected and defined.
(2) The integration constraints included in the system/software requirements, architecture, and

design shall be identified.

(3) Software shall be integrated based on the software design specifications, and the baseline shall be determined after software integration.

(4) Debug information collected during software integration shall be recorded and, as necessary, related processes such as the problem resolution (refer to 6.8) and configuration management (refer to 6.2) shall be implemented.

5.3.10.2 Measurement

The following measurement shall be performed for software integration for quality evaluation:

(1) For debug information collected during software integration, the definition of data to be collected for evaluating software products quality and its evaluation method shall be defined.

(2) Collection and evaluation of data defined in (1) above shall be planned.

(3) Collection and evaluation of data defined in (1) above shall be performed and the results recorded.

(4) Results of evaluation in (3) above shall be reported periodically, or at every milestone.

5.3.10.3 Input

(1) Source code (unit)

(2) Operation assumptions and constraints

(3) Software design specifications

5.3.10.4 Output

(1) Source code (integrated)

(2) Software (integrated)

(3) Identified results of constraints for the integration

(4) Software integration measurement results

5.3.11 Software integration test

5.3.11.1 Activity

Activities that meet the following shall be performed for the software integration test:

(1) Test preparation

(a) As the result of software coding, testing, and software integration, the software test specifications shall be updated as necessary.

(b) For software test, the following shall be considered:

(i) Operational scenarios

(ii) Interface specifications

(iii) Maximum load for assumed scenarios

(iv) Coverage for software requirements specification and software design specifications

(v) Anomalies, such as exceptions and failures

(vi) Applicability of COTS or reused software items with computer system

(c) Software integration test procedures shall be documented in accordance with the software verification plan, the software test plan, and the software test specifications.

(d) If new operational assumptions and constraints arise or are identified, they shall be updated.

(e) Test specifications and procedures shall be checked.

(2) Implementation of tests

(a) The tests shall be performed in accordance with the software integration test procedure.

(b) During the software integration test, quick reviews of the test results shall be performed as necessary, and judgment shall be made about whether the test shall be continued or not.
(c) With regard to the software integration test, information about the test environment, test data, configuration and version of the software under test shall be recorded to ensure the reproducibility of test conditions.
(d) Test results shall be recorded and stored appropriately, and they shall be presented as necessary.
(e) If software or software integration test specifications need to be revised during a software integration test, the effectiveness of the activities such as the joint reviews, verifications, and validations performed formerly shall be evaluated and tests shall be performed again as necessary.

5.3.11.2 Measurement

For software integration tests, the following measurement shall be performed in order to evaluate the quality:

(1) Collection of quality metrics data
  (a) Failures found during software integration tests shall be recorded, together with related information such as test cases.
(2) Quality metrics data setting
  If quality metrics other than (1) above are set, collected, and evaluated, the following shall be implemented:
  (a) Metrics for quality evaluation during tests shall be set
  (b) Identified data shall be collected
  (c) Analysis evaluation method for the identified data shall be defined
  (d) Analysis and evaluation of identified data shall be performed
(3) Result of data analysis and evaluation shall be reported periodically or for each milestone

5.3.11.3 Input

(1) Interface specifications
(2) Software requirements specification
(3) Software design specifications
(4) Software verification plan
(5) Software test plan
(6) Software test specifications
(7) Operation assumptions and constraints
(8) Source code (integrated)
(9) Software (integrated)
(10) Operational scenarios

5.3.11.4 Output

(1) Software integration test procedure
(2) Software integration test record, including pass or failure judgment results
(3) Operational assumptions and constraints (updated)
(4) Source code (tested)
(5) Software (tested)

(6) Software test specifications (updated)
(7) Software integration test measurement results

5.3.11.5 Review

For each output, a software test review shall be performed. Items to be reviewed shall be chosen based on 5.3.11.4 and shall be defined in the relevant plan document, such as the software development plan. Appropriate follow-up on action items shall be performed in accordance with due dates, follow-up status, degree of influence, and so on. If a review is performed, it shall be documented in a technical review record after its completion. In addition, quantitative data, such as the reviewers' positions, their review time, the number of questions, and their comments shall be recorded, and the quality of the review shall be evaluated.

5.3.12 Software installation into target platforms (embedding)

5.3.12.1 Activity

Activities that meet the following shall be performed for the software installation into target platforms:

(1) Software shall be prepared in a form that allows installation into the target platform, and the configuration management information (filename, version information, and so on) of the software shall be acquired.
(2) Configuration management information of software to be released shall be prepared, to include the installation plan, schedule and procedure into the target platform, the installation result check procedure, and the operational assumptions and constraints.

In principle, in planning an installation, the following shall be considered.

(a) Identification of the enabling system/service needed for the installation and the acquisition of the access right of them
(b) Set an advanced verification for the installation
(c) Provision and support for the software or the installation service (provide supports for the acquirer)

(3) Before installation, a check of the installation preparation status (whether operational conditions/constraints have been resolved, the advanced verification has been done, and so on) shall be performed. After that, software shall be installed into the target platform in accordance with the installation procedure. However, this can be omitted if the software has already been installed into the target platform.
(4) A check that the software has been properly installed shall be performed, in accordance with the results check procedure and confirmed results.

5.3.12.2 Input

(1) Software
(2) Operational assumption and constraints

5.3.12.3 Output

(1) Software prepared in a form that allows installation into the target platform
(2) Configuration management information
(3) Installation plan, schedule, and procedure
(4) Software installed computer system

(5) Installation result check procedure, confirmed result

5.3.13 Computer system integration and computer system integration test

5.3.13.1 Activity

Activities that meet the following shall be performed for the computer system integration and computer system integration test:

(1) Test preparation
   (a) Computer system integration shall be performed with the identification of the integration constraints included in the system/software requirements, architecture and designs.
   (b) Based on objectives, the computer system integration criteria and verification points for software functions (operations of correct interfaces and completeness) shall be selected and defined.
   (c) Computer system integration test specifications and procedures shall be documented in accordance with the software verification plan.
   (d) With regard to the computer system integration test specifications, the following viewpoints shall be considered:
      (i) Operational scenarios
      (ii) Software requirements specification
      (iii) Maximum load
      (iv) Coverage regarding requirements specification for the computer system and computer system architectural design specifications
      (v) Anomalies, such as exceptions and failures
      (vi) Applicability of COTS or reused software items with the computer system
   (e) Incidents and problems found during test preparation (test procedure checks, and so on) shall be recorded and managed.

(2) Implementation of tests
   (a) The tests shall be performed in accordance with the computer system integration test procedure.
   (b) As necessary, based on the operational scenarios, tests shall be performed by using tools such as simulators, and the verification coverage regarding operational scenarios shall be checked.
   (c) With regard to the computer system integration test, information about the test environment, test data, configuration, and version of the software under test shall be recorded to ensure the reproducibility of test conditions.
   (d) Test results shall be recorded and stored appropriately.
   (e) When software or computer system integration test specifications need to be revised during a computer system integration test, the performance effectiveness of the activities such as the joint reviews, verification, validation, tests, and others shall be evaluated and performed again, if necessary.

5.3.13.2 Measurement

The following measurement shall be performed for computer system integration and computer system integration tests in order to evaluate the quality:

(1) Collection of quality metrics data

(a) Failures found during the computer system integration tests shall be recorded, together with related information such as information about test cases.
(b) Incidents and problems found during the test preparation (test procedure checks, and so on) shall be recorded, together with related information such as information about test cases.

(2) Quality metrics data definition

If a quality metrics other than (1) above is defined, collected, and evaluated, the following shall be implemented:

(a) The metrics for quality evaluation during computer system integration tests shall be defined.
(b) Identified data shall be collected.
(c) The analysis evaluation method for the identified data shall be defined.
(d) Analysis and evaluation of identified data shall be performed.

(3) The result of data analysis and evaluation shall be reported periodically, or for each milestone.

5.3.13.3 Input

(1) Requirements specification for the computer system
(2) Computer system architectural design specifications
(3) Software requirements specification
(4) Software verification plan
(5) Operational assumptions and constraints
(6) Operational scenarios
(7) Software installed computer system

5.3.13.4 Output

(1) Computer system integration test specifications
(2) Computer system integration test procedure
(3) Computer system integration test record, including pass or failure judgment results
(4) Operational assumptions and constraints (updated)
(5) Identified results of constraints for the integration (after updated)
(6) Computer system integration and integration test measurement results

5.3.14 Supply and introduction of software product

5.3.14.1 Activity

Activities that meet the following requirements shall be performed for the supply and introduction of software product:

(1) Documentation of manuals

Software user's manuals shall be documented.

(2) Preparation for supply

The software that is ready to be supplied shall be confirmed, and the confirmation results shall be recorded.

(3) Establishment of an introduction plan

The introduction of software shall be planned, including replacement of the existing system and temporary parallel operation, and a procedure shall be documented. For the introduction plan and procedure, not only installation of the software into the target platform, but also work for introducing computer system into the actual operational environment, shall be considered.

(4) Implementation of introduction and recording of results

Software shall be installed into the target platform in accordance with the introduction plan and procedure, and the computer system shall be introduced to the actual operational environment. The results of the introduction shall be recorded.

5.3.14.2 Input

(1) Software requirements specification
(2) Software design specifications
(3) Source code or software
(4) Installation procedure

5.3.14.3 Output

(1) Software user's manual
(2) Source code or software
(3) Confirmation results that software is ready to be supplied
(4) Installation procedure
(5) Introduction plan
(6) Introduction procedure
(7) Record of introduction results

5.3.15 Software acceptance

5.3.15.1 Activity

Activities that meet the following shall be performed for the software acceptance:

(1) With regard to acceptance inspection and acceptance testing, plans shall be established, subsequently, specifications and procedures shall be documented. If this is to be substituted by tests performed by the suppliers, the acquirers' approval shall be needed regarding the content of those tests.
(2) Acceptance inspection and acceptance testing shall be performed in accordance with the plans, the specifications and the procedures described in (1). Records of the acceptance inspection and acceptance testing shall be maintained.
(3) A review shall be performed regarding confirmation of the results that the software to be acquired is ready to be supplied.

5.3.15.2 Input

(1) Software user's manual
(2) Source code or software
(3) Configuration management information
(4) Record of confirmation that software is ready to be supplied
(5) Installation procedure

5.3.15.3 Output

(1) Acceptance inspection and testing plan
(2) Acceptance inspection and testing specifications
(3) Acceptance inspection and testing procedure
(4) Acceptance inspection and testing record

5.4 Operation process

The purpose of the Operation process is to operate a computer system including software in an intended environment and to provide supports to customers and users.

This process consists of the following activities

(1) Process implementation
(2) Operational testing
(3) Operation of the computer system including software
(4) Management of operational results
(5) Customer and user support

5.4.1 Process implementation

5.4.1.1 Definition of an operation strategy

(1) An operation strategy shall be defined. In principle, the following shall be considered in the strategy.
   (a) Expected each criteria (capacity, occupancy rate, response, safety, etc.) between service installation, periodic operation and disposal
   (b) Software or computer system release criteria and schedule considering the correction to maintain the current service
   (c) Method to implement each operation mode (regular operation/preparation stage/operation test/assumed occurrence of disasters and troubles)
   (d) Operation metrics to evaluate performance level

5.4.1.2 Establishment of an operational plan

A plan to perform the operation process based on the operation strategy shall be planned and executed. Additionally, a standard related to the operation shall be set. In principle, in planning, the following shall be considered.

(1) Constraints which exist within the software or computer system requirement analysis, architecture, design, implementation, and transition in the operation process shall be identified.
(2) Enabling systems or services needed to support the operation shall be identified.
(3) The enabling systems or services to be used shall be obtained or access to them shall be acquired.

5.4.1.3 Establishment of problem management for the operation

In accordance with the problem resolution process (refer to 6.8), a problem report handling procedure shall be developed e.g. for receiving problem reports, recording, resolving, tracking problems, and notice of the status.

5.4.1.4 Establishment of operational procedures for the computer system including software operation and user support

The following procedures shall be established.

(1) The procedure for testing under the operational environment for the computer system including software
(2) The procedure for change request to the maintenance process (refer to 5.5)
(3) The procedure for releasing the software or computer system for operational use.

5.4.2 Operational testing

An environment to perform the operational testing shall be established. For each release of the software or the computer system, operational testing shall be performed. The fact that the operational testing has been performed as planned and finished shall be confirmed, and the review for them shall be performed. After the software or the computer system satisfies the specified criteria, it shall be released for operational use.

5.4.3 Operation of computer system including software

The tasks that fill the following requirements shall be performed.

(1) Operate in the designated operational environment followed the software user's manual
(2) Apply facilities and resources as needed to maintain operation and service
(3) Consider the followings and monitor the operation
  (a) Conformity to the operation strategy (such as procedures related the operation)
  (b) Records and reports of the possible invasions to the software and the serious matters such as data confidentiality and completeness
  (c) Record that the software, computer system or service capacity is out of the tolerable parameter
(4) According to the operation strategy, develop operation procedures to minimize operation trouble risks and automate them as much as you can
(5) According to the operation strategy, analyze the measurement results to confirm the followings
  (a) The service capacity is in the tolerable parameter or in the agreed service level for agreed work volume
  (b) Availability and response for the software or computer system, and service are in the tolerable range
  (c) Possible improvement items are identified and prioritized
(6) Perform the disaster and trouble operations as necessary

5.4.4 Operation results management

This activity consists of the following tasks

(1) The results of operation and anomalies encountered shall be recorded.
(2) Operational incidents and problems shall be recorded and their resolution shall be tracked
(3) The traceability of the operational services and configuration items shall be maintained.
(4) Key artifacts and information items that have been selected for baselines shall be provided.

5.4.5 Customer and user support

The following tasks shall be performed for the customer and user support:

(1) Support service shall be provided to the customers and users to resolve incidents, problems and service requests. These requests and subsequent actions taken for support shall be recorded and managed to provide support adequately.
(2) If a problem is identified in the user support, it shall be resolved in accordance with the procedure established in 5.4.1.3.
(3) If there is a temporary work-around for identified problem, it shall be provided to customers and users.
(4) Determine the degree of the satisfaction to which the delivered software/computer system or services satisfy the needs of the customers and users.

5.5 Maintenance process

The purpose of the Maintenance process is to sustain the capability of the software or the computer system to provide service and the maintenance process is defined when the modification and maintenance of the software products, including design documents, requires a change of development environments. This process consists of the following activities:

(1) Process implementation
(2) Problem identification and modification analysis
(3) Modification implementation
(4) Software reprogramming
(5) Logistics support implementation
(6) Management of the maintenance and logistics result
(7) Transition
(8) Software disposal

5.5.1 Process implementation

5.5.1.1 Definition of a maintenance strategy

(1) A maintenance strategy shall be defined. The followings shall be considered in the strategy.
  (a) Establishing priorities, typical schedules, and procedures for performing, verifying, distributing, and installing software maintenance changes in conformance with operational availability requirements
  (b) Establishing techniques and methods for becoming aware of the need for correction
  (c) Establishing priorities and resources to obtain access to the correct versions of the product and product information needed for performing maintenance (e.g., scheduled or phased installation, maintenance patches or software upgrades)
  (d) Agreed rights to data and the impact on data in the software or computer system during problem resolution and maintenance activity
  (e) Approach to assure that counterfeit or unauthorized software elements are not introduced into the software or computer system
  (f) Impact of the maintenance change on other software systems elements versus the risk of leaving a reported software anomaly in place
(2) For non‐software elements, a logistics strategy (number of the store target elements, variation, location, condition, expected replace ratio, duration, and update frequency) shall be defined. Logistics helps to ensure that the necessary material and resources, in the right quantity and quality, are available at the right place and time.
(3) Constraints from maintenance to be incorporated in the software or computer system requirements, architecture, or design shall be defined.
(4) Trades such that the maintenance and logistics actions result in a solution that is affordable, operable, and sustainable shall be coordinated.
(5) The necessary enabling systems or services needed to support maintenance shall be identified and planned.
(6) The enabling systems or services to be used shall be obtained or access to them shall be acquired.

5.5.1.2 Establishment of a maintenance plan

The maintenance plan for implementation of the maintenance process shall be documented in accordance with the maintenance strategy and implemented. The following shall be included in the plan:

(1) Maintenance management method for developed software products
(2) The structure and related documentation
(3) Data, including the management information necessary for maintenance
(4) Record
(5) Maintenance environment (environment of development phase, and so on)
(6) Actions for maintaining other related processes appropriately (problem resolution process (refer to 6.8), configuration management process (refer to 6.2), and so on)

5.5.2 Problem identification and modification analysis

(1) In accordance with the plans, the analysis of the requested contents regarding maintenance or modification of the software product, and correspondence shall be evaluated.
(2) If the software modification and reprogramming are implemented, agreement for the modification plan shall be obtained in accordance with the maintenance plan.

5.5.3 Modification implementation

If a modification is needed, it shall be implemented in accordance with the modification implementation plan and procedures. The corresponding process of this standard shall be implemented again, as necessary.

5.5.4 Software reprogramming

This standard shall also be applied to the work such as the partial modification of the software which is called patches, and to the addition and extension of software functions during operational periods. In principle, upon encountering unexpected faults that cause a system failure while reprogramming, the software or the computer system shall be restored to the operational status.

*This does not imply that their development processes shall be exactly the same as the development-to-retirement processes of the software that is being used for operations. For projects assuming reprogramming, the development-to-retirement processes regarding reprogramming shall be concretized and tailored in advance.

In principle, this standard shall be applied and the compatibility shall be evaluated.

5.5.5 Perform logistics support

This activity consists of the following tasks.

(1) Obtain resources to support the software product through project lifecycle
(2) Monitor the quality and availability of the reprogramed software elements and enabling systems, their distributing mechanisms and their continued integrity during storage.
(3) Implement mechanisms for software product distribution
(4) Confirm that the activities needed for logistics support for software products are planned and implemented.

5.5.6 Manage results of maintenance and logistics

This activity consists of the following tasks.

(1) Record incidents and problems, including their resolutions, and significant maintenance and

logistics results.

(2) Identify and record trends of incidents, problems, and maintenance and logistics actions.
(3) Maintain traceability of the software or computer system elements being maintained.
(4) Provide key artifacts and information items that have been selected for baselines.
(5) Monitor and measure customer and user satisfaction with software product or computer system and maintenance support.

5.5.7 Transition

This activity is a process for transitioning software to a new environment.

5.5.7.1 Define a transition strategy

A transition strategy as a preparation for software product transition shall be defined. In principle, the following shall be considered.

(1) Establishing the type of transition and transition success criteria
(2) Determining the frequency of recurring transitions, such as updates and upgrades to development, test, and operational software or computer systems
(3) Minimizing security risks, disruption, and downtime during transition
(4) Archiving, destroying, or converting and validating data from previous systems to the new system; including data received through external interfaces
(5) Contingency planning (problem resolution, backup and return to the last working system version)
(6) Scheduling transitions consistent with ongoing business processing, with phased or synchronized transition of systems
(7) Change management for stakeholders, including interface partners, human operators, system administrators, and software system or service users
(8) Associated strategies for validation of the transitioning system or element
(9) Initiating user support and maintenance activities with the transfer and update of system design documentation, user documentation, and test specifications
(10) Concurrent execution of the Transition, Operational, and Disposal processes, when a new system is commissioned and an old system is decommissioned

5.5.7.2 Establishment and execution of transition plan

Based on the transition strategy, a transition plan shall be developed and executed.

(1) Requirement analysis and definition for transitions including software requirements that are integrated as well as constraints of architecture/design that may be found according to the transition
(2) Frequency of transitions and their schedules
(3) Facilities, sites, communication network and hardware environment that are needed to be changed because of the transition (or new release)
(4) Documents for related people such as operators, users and system support members, and trainings
(5) Tools for the transition (including enabling systems/services)
(6) Conversion of software product and data
(7) Transition rehearsal
(8) Transition execution
(9) Transition confirmation
(10) Future support for old environments

5.5.7.3 Notification to users

The migration plan and contents of implementation shall be notified to users. Notifications shall include the following viewpoints:

(1) Statement of why the old environment is no longer to be supported
(2) Description of the new environment, with its date of availability
(3) Description of other support options available, if any, once support for the old environment has been removed

5.5.7.4 Manage transition results

The following records accompanied by the software transitions shall be managed.

(1) Incidents and problems happened during transitions
(2) Maintenance of the traceability of the transitioned software elements
(3) Key artifacts and information items that have been selected for baselines

5.5.7.5 Storage of old environments

The records, such as documentation, logs, and other items that relate to old environments shall ideally be retained.

5.5.8 Software disposal

The following viewpoints shall be considered for the software disposal:

(1) A disposal strategy to remove active support by an institution engaged in operation and maintenance shall be defined. In principle, the following shall be considered in the strategy.
  (a) Identification of permanent termination of the system's functions and delivery of services
  (b) Identification of ownership and responsibility for retention or destruction of data and intellectual property in the software system
  (c) Transformation of the product into, or retention in a socially and physically acceptable state, thereby avoiding subsequent adverse effects on stakeholders, society and the environment
  (d) Disposal actions that reflect health, safety, security, and privacy concerns based on the long-term condition of hardware and data
  (e) Notification to relevant stakeholders of significant disposal activities.
  (f) Identification of schedules, actions, responsibilities, and resources for disposal activities.
  (g) Identification of the constraints on disposal for system/software requirements, architecture and design characteristics, or implementation techniques.
  (h) Identification and planning for the necessary enabling systems or services needed to support disposal.
  (i) Obtainment of the necessary enabling systems and services or acquisition of access to them.
  (j) Containment facilities, storage locations, inspection criteria and storage periods, if the software system or data is to be stored.
  (k) Preventive methods to preclude disposed software.
(2) A disposal plan shall be developed based on the disposal strategy. Users shall be included in the development of the plan.
(3) Users shall be given notification of the disposal plans and activities. Notifications shall include the following:
  (a) Description of the replacement or upgrade of to the software or computer system with its

date of availability

(b) Statement of why the software or computer system is no longer to be supported

(c) Description of other support options available, once support has been removed

(4) According to the disposal plan, the following activities shall be performed.

(a) Parallel operation of the retiring and the new software or computer system shall be performed, for smooth transition to the new computer system. It is recommended that user training shall be provided during this period.

(b) Remove the target software, computer system and data.

(c) Set the new software or computer system to the last status of the old system or the status already decided.

(d) Reuse, recycle, recondition, overhaul, archive, or destroy the removed software/computer system/data.

(5) When the scheduled disposal arrives, notification shall be sent to all concerned parties. It is recommended that all associated development documentation, logs, and code shall be archived.

(6) For the disposed software or computer system, data shall be collected to permit audits and reviews in the event of long-term hazards to health, safety, security, privacy, and the environment.

6 Supporting life cycle process

This clause defines the following supporting life cycle processes:

(1) Documentation process
(2) Configuration management process
(3) Quality assurance process
(4) Verification process
(5) Validation process
(6) Joint review process
(7) Assessment process
(8) Problem resolution process

The activities and tasks in a supporting process are the responsibility of the organization performing that process. This organization ensures that the process is in existence and functional.

6.1 Documentation process

This process consists of the following activities:

(1) Process implementation
(2) Development
(3) Production
(4) Maintenance/Revision/Disposal

6.1.1 Process implementation

A plan including the following shall be established as a documentation plan:

(1) Documents developed through the software and computer system life cycle, and the documents schedule for development shall be clarified.
(2) Procedures for the development of documents (development, inspection, and approval), issue (issue, distribution, and storing), and revision (revision and disposal) shall be decided.
(3) The form (content, format, and so on) appropriate for such documents shall be decided.

6.1.2 Development

Documents shall be developed in accordance with the decided procedure. The following shall be considered:

(1) The appropriateness of sources of information for the documents shall be confirmed.
(2) The documents shall be checked and approved regarding format, technical content, against their documentation standards.

6.1.3 Production

Documents production shall be implemented in accordance with the defined procedures. The following shall be considered:

(1) Production and distribution of documents shall be at the appropriate version.
(2) The documents shall be managed in accordance with requirements (security management, backup, and so on) including parties for distribution.

6.1.4 Maintenance/Revision/Disposal

Documents shall be maintained, revised or disposed in accordance with the decided procedure. In

addition, the following shall be considered.

(1) In documentation, the integrity requirement of the information included in that shall be considered.
(2) The revised documents shall be checked and approved regarding the form and technical content.
(3) Needless, ineffective or valid-less documents shall be disposed.

6.2 Configuration management process

Configuration management process is a process of applying the following administrative and technical procedures through the software life cycle:

(1) Process implementation
(2) Configuration identification
(3) Configuration change control
(4) Record of configuration change status
(5) Evaluation of configuration change status
(6) Release management and delivery

6.2.1 Process implementation

6.2.1.1 Definition of a configuration management strategy

A configuration management strategy shall be defined. In principle, the following shall be considered in the strategy.

(1) Control of software licenses, data authorities, and other intellectual property rights
(2) Frequency of releases, priorities, and content of software versions and software
(3) The audit strategy and the responsibilities for validating continuous integrity and security of the configuration definition information
(4) Change management, including users in operational software/services

6.2.1.2 Establishment of a configuration management plan

A configuration management plan shall be developed according to the configuration management strategy. In principle, the following shall be included in the plan.

(1) Configuration management activities
  (a) Consideration of the level of risk and degree of influence in approval of configuration baselines, and regular and emergency change requests
(2) Procedures and schedule for performing the activities
  (a) The necessary baseline to be established, including criteria for accessing and changing of configuration items, controlling of actions, commencing configuration control and maintaining baselines of evolving configurations.
(3) The organization responsible for performing the activities.
  (a) Roles and responsibilities of the configuration management activities including the operation of the configuration control boards
(4) The relationship with other organizations
  (a) Coordination of the configuration management across the set of acquirer, supplier, and supply chain organizations for the lifecycle of the software, or the extent of the agreement or project, as appropriate.
  (b) Configuration management and change management plan, including users in operated software/services

(5) Store, archive and access procedure for configuration items, configuration management work products and records

6.2.2 Configuration identification

(1) Configuration items shall be identified.
(2) For configuration items, the following shall be identified:
  (a) Version references
  (b) Other identification details
(3) For managed items, baselines shall be established according to their objectives.
(4) Obtain stakeholders agreement to the established configuration baselines.

6.2.3 Configuration change control

The following shall be identified and defined for the configuration changes:

(1) Identification and recording of change requests (including waiver)
(2) Analysis and evaluation of changes
(3) Approval or disapproval of change requests
(4) Implementation, verification, and release of modified configuration items
(5) Traceable review records against the reason for the modification, and authorization of the modification
(6) Not used

6.2.4 Record of configuration change status

Management records and status reports that show the status and history of controlled configuration items including baselines shall be prepared and the status shall be reported.

6.2.5 Evaluation of configuration change status

According to the planned schedule in advance, the following confirmations shall be performed.

(1) The functional completeness of the configuration items against their requirements and the physical completeness of the configuration items
(2) The validities of the approved configuration changes

Confirmed results and required actions shall be recorded.

6.2.6 Release management and delivery

The release and delivery of software product shall be formally controlled in accordance with the procedure. The source code and documentation that contain safety or security critical functions shall be handled, stored, packaged, and delivered in accordance with the policies of organizations involved.

The distribution of the assigned software product releases shall be managed.

6.2.7 Configuration audit implementation

Configuration audit implementation in accordance with the configuration management plan shall be confirmed.

6.3 Quality assurance process

The quality assurance process is a process for providing adequate assurance that the process activities and outputs, of the software or computer system life cycle and the project life cycle, adhere to their established plans which are based on this standard or by tailoring its results.

This process consists of the following:

(1) Process implementation
(2) Products and services quality assurance
(3) Process assurance
(4) Assurance of quality system
(5) Management of the quality assurance record

6.3.1 Process implementation

This activity consists of the following tasks:

6.3.1.1 Confirmation of the independence of the Organization and establishment of the adequate structure

The management organization of the quality assurance process shall be clarified, and the implementation status and validity of quality assurance process shall be reported.

The management organization of the quality assurance process shall be authorized to maintain organizational freedom and authority to assess the problem and offer a resolution.

In addition, the person who is invested with all the responsibilities and authority for quality assurance process shall be independent of the development organization.

6.3.1.2 Definition of a quality assurance strategy

(1) A quality assurance strategy shall be defined. In principle, the followings shall be considered in the strategy.
    (a) With priorities, the Quality Assurance resources to processes and tasks that have the most significant impact on the quality of the delivered products and services shall be applied.
    (b) Defined responsibilities and authorities
    (c) Evaluation criteria and methods for processes, products, and services, including criteria for product or service acceptance
    (d) Quality Assurance activities to each supplier (including subcontractors)
    (e) Verification, validation, monitoring, measurement, review, inspection, audit or assessment, and test activities specific and required to the products or services
    (f) Problem resolution and process and product improvement activities

6.3.1.3 Establishment of a quality assurance activity plan

Establish a quality assurance activity plan based on the quality assurance strategy.

The quality assurance activity plan shall include the following:

(1) Identification of the system to be applied
(2) Resources, quality standards, methodologies, procedures, and tools needed for performing the quality assurance process (including identification of all documents)
(3) Selected activities and tasks from the processes, such as the verification process (refer to 6.4), the validation process (refer to 6.5), the joint review process (refer to 6.6), the assessment process (refer to 6.7), and the problem resolution process (refer to 6.8)
(4) Schedules
(5) Procedures for review
(6) Procedures for the works, such as identification, collection, filing, maintenance, disposition and disposal of quality assurance activity records
(7) Requirements and process regarding quality assurance for purchase management and supplier

(8) Management for existing software items (COTS items or knowledge assets)
(9) Procedures for the delivery
(10) Procedures for process and product improvement activities in problem identifications and their resolutions.

6.3.2 Product and service quality assurance

(1) Products and services shall be assured to fulfil their plans.
(2) Through the verification and validation of the software product or computer system development process for each product, the product shall be assured to fulfil its approved requirement.
(3) Products quality shall be assured by IV&V as necessary.

6.3.3 Process assurance

(1) It shall be assured that the software development plans, operational plans, and processes defined by maintenance plans, comply with this standard.
(2) It shall be assured that the software developments perform in accordance with the processes defined in a development plan, an operational plan, or a maintenance plan.
(3) It shall be evaluated that tools and environments that support or automate the processes are usable in the processes of this standard.

6.3.4 Assurance of quality system

It shall be assured that the quality system contains the quality management tasks listed below:

6.3.4.1 Education and training

All techniques, abilities, and qualifications needed for personnel engaged in development, maintenance and operation work with the software or computer system shall be identified, and education and training shall be conducted.

6.3.4.2 Purchase management and supplier management

(1) Purchase management

The reliability and quality of purchased items (COTS included) shall meet the software quality assurance requirements of the organization for the developing software product.

(2) Suppliers and purchase vendor selection

Based on the capability evaluation and selection record regarding suppliers and purchase vendor maintained in an organization, suppliers and purchase partners shall be selected.

6.3.4.3 Management of items supplied by acquirer

Procedures for inspection at the time of accepting items supplied or lent from the acquirer, and procedures for their storage and maintenance management, shall be established and followed.

6.3.4.4 Management of existing software items (COTS or knowledge assets)

With regard to existing software items (COTS or reused software items), the following shall be included in the management items. The items below are to be managed for applied projects and future reuse:

(1) Any objectives, reasons, and benefits in using existing software items
(2) Evaluation items and levels that judge the availability of existing software items
  (a) Applicability of existing software with regard to the developing software

(b) Traceability relative to requirements applied to development software items
(c) Risk obtained from the information such as the past performance of product, and so on, of software items to be used
(d) Acceptance and assurance conditions

(3) Consideration points and items in managing the use of the existing software items
(a) Associated documents, which are obtainable and usable
(b) Introduction, preparation, training, and constraints
(c) Identification of versions and other details, and the configuration management method
(d) Maintenance and future support
(e) The rights such as intellectual property rights
(f) Identification of newly available knowledge assets

6.3.4.5 Handling, storing, and labeling

To ensure appropriate handling, storing, and labeling of software or computer system, requirements which include the following items, shall be documented, and the products shall be released in accordance with that document:

(1) Media shall have labels (names, identifiers, and so on) so that stored software can be identified.
(2) Identification of software shall be checked when reading software from media.

6.3.5 Manage quality assurance records

(1) Create records related to quality assurance activities
(2) Maintain and store the records.

6.4 Verification process

The purpose of the verification process is to provide objective evidence that a computer system fulfils its specified requirements and characteristics. The verification process consists of the following activities. This process may include the checking works such as test, review, analysis, and so on:

(1) Process Implementation
(2) Verification
(3) Management of the verification results

6.4.1 Process implementation

6.4.1.1 Definition of a verification strategy

(1) A verification strategy shall be defined. In principle, the followings shall be considered in the verification strategy.
(a) Identify the verification scope (including the target software, items and products), the characteristics to be verified, and the expected verification results. In principle, the characteristics include system/software requirements, architecture, design characteristics such as security and important quality characteristics, integration, and the correctness of the documents.
(b) Identify the constraints that potentially limit the feasibility of verifications.
(c) Identify the priorities of the verification scenarios.

(2) Identify the integration constraints included in the system/software requirements, the architecture, and designs.

6.4.1.2 Establishment of a verification plan

(1) Based on the verification strategy, the verification plan including the followings shall be established.
   (a) The target for verification shall be determined, and appropriate tasks defined in 6.4.2 below shall be selected according to degree of importance.
   (b) Appropriate methods or technologies of the verification (inspection/analysis/demonstration/test), and the standard about the verification shall be selected.
   (c) The verification procedures and a series of activities shall be defined. The procedures include how to record, analyze, store, and report the verification results.
   (d) Identify and acquire access to the enabling systems or services needed to support verification.
   (e) The verification (test, review, analysis and so on) shall be performed according to the verification plan. Incidents and problems detected through verification shall be resolved in accordance with the problem resolution process (refer to 6.8).

6.4.2 Verification

This activity consists of the following tasks:

6.4.2.1 Process verification

The following shall be considered:

(1) Adequacy of the processes selected for the project
(2) Planning of the processes is adequate
(3) Applicable standards and environment for the project's processes in place
(4) Adequate levels of competent staff allocated to the processes
(5) Proper execution of the processes

6.4.2.2 Requirements verification

The following shall be considered:

(1) The requirements are consistent, feasible, and verifiable.
(2) Requirements for software items are appropriately allocated (not including requirements for hardware items and operation).
(3) The higher level requirements and the standards applied to items shall be satisfied.
(4) Concerning to the requirement to be especially taken care such as safety and security and so on, it is able to show the proper method that it satisfies the higher level requirements and the standards applied to items.

6.4.2.3 Design verification

The following shall be considered:

(1) The design shall meet requirements, and shall be traceable to requirements.
(2) Designed properly with respect to data interface, timing, computer resource (memory capacity, processing speed, and so on), logic design, processing sequence and processing contents (especially initialization, termination, exception handling and so on).
(3) The characteristics such as portability, modifiability and ease of problem resolution have been covered.

6.4.2.4 Source code verification

The following viewpoint shall be considered:

(1) Source code shall meet design, and shall be traceable to the design.
(2) Implemented properly with respect to data interface, timing, computer resources (memory capacity, processing speed, and so on.), logic design, processing sequence and processing contents (especially initialization, termination, exception handling and so on).
(3) The characteristics such as portability, modifiability and ease of problem resolution have been covered.
(4) Concerning to the source code to be especially taken care such as safety and security and so on, it is able to show the proper method that it satisfies the requirements and the standards applied to items.
(5) Source code shall conform to e.g. coding standards.

6.4.2.5 Integration verification

The following viewpoint shall be considered:

(1) The configuration of the software modules and data is in proper and correct versions. The software components and units of each software item have been completely and correctly integrated into the software item.
(2) The modules and data configuring software have been completely and correctly integrated into the system and the traceability of the integration has been maintained.
(3) The integration tasks have been performed in accordance with the plan.

6.4.2.6 Documentation verification

The following shall be considered:

(1) The documentation is adequate and consistent.
(2) The documentation has been organized in accordance with the plan.
(3) Configuration management of the documentation is performed in accordance with the proper procedures.

6.4.3 Manage the verification results

Performed verification results shall be managed. This activity consists of the following tasks.

(1) Maintain the traceability results of the verified software and items.
(2) Provide key artifacts and information items (for example, verification strategy, verification procedure) that have been selected for baselines.

6.5 Validation process

The purpose of the validation process is to provide objective evidence that the computer system, when in use in its intended operational environment and ways, fulfils its expected mission objectives and stakeholder requirements.

This process consists of the following activities:

(1) Process implementation
(2) Validation
(3) Management of the validation result

6.5.1 Process implementation

6.5.1.1 Definition of a validation strategy

(1) A validation strategy shall be defined. In principle, the followings shall be considered in the validation strategy.
   (a) Identification of the validation scope (including the target software, items and products) and the expected validation results.
   (b) Identification of the constraints that potentially limit the feasibility of validations.
   (c) Identification of the priorities of the validation scenarios.
(2) Identify the integration constraints included in the system/software requirements, the architecture, and designs.

6.5.1.2 Establishment of a validation plan

(1) Based on the validation strategy, the validation plan shall be established and include the following:
   (a) Not used.
   (b) A validation plan shall be developed. The plan shall include the following:
      (i) Items subject to validation
      (ii) Validation tasks to be performed
      (iii) Resources, responsibilities, and schedule for validation
      (iv) Procedures for distributing validation reports
   (c) Appropriate methods or technologies of the validation and the validation criteria shall be selected (*).
   (d) The validation procedures and a series of activities shall be defined. The procedures include how to record, analyze, store, and report the validation results.
   (e) Identify and acquire access to the enabling systems or services needed to support validation.
   (f) Validation shall be performed according to the validation plan. Incidents and problems detected by validation shall be resolved in accordance with the problem resolution process (refer to 6.8).

*As a validation method, methods other than testing (analysis, modeling, simulation, and so on) may be appropriate.

6.5.2 Validation

(1) Test requirements and test cases shall be selected for validation, and test specifications shall be prepared.
(2) Test cases shall be checked to ensure they reflect the method and use of the software or computer system.
(3) Tests in (1) and (2) above shall be performed to include the following view points, as necessary:
   (a) Not used
   (b) Fault testing
   (c) Testing that representative users can successfully achieve their intended tasks
(4) Validation shall be performed so that the software or computer system satisfies its intended use.
(5) Tests for the software or computer system shall be performed as appropriate in the target environment. When a simulated instead of a real environment is used, the difference between them shall be evaluated.

6.5.3 Manage the validation results

Performed validation results shall be managed. This activity consists of the following tasks.

(1) Review validation results and anomalies encountered and identify follow-up actions.
(2) Record incidents and problems during validation and track their resolution.
(3) Obtain stakeholder agreement that the software system or element meets the stakeholder needs.
(4) Maintain traceability results of the validated software and items.
(5) Provide key artifacts and information items that have been selected for baselines.

6.6 Joint review process

The joint review process consists of the following activities:

(1) Process implementation
(2) Project management reviews
(3) Technical reviews

6.6.1 Process implementation

(1) Periodical reviews shall be held at predetermined milestones, as specified in the project plans. It is recommended that ad hoc reviews shall be called when deemed necessary by either reviewing party or reviewed party.
(2) All resources required to perform the reviews shall be agreed on between all the parties. These resources include personnel, location, facilities, hardware, software, and tools.
(3) It is recommended that all the parties concerned shall agree on the following at each review:
    (a) Matters to be reviewed
    (b) Review scope and viewpoint
    (c) Review method
    (d) Entry and exit criteria for the review
(4) Incidents and problems detected during the reviews shall be recorded, and appropriate action taken through the problem resolution process (refer to 6.8), as necessary.
(5) The review results shall be documented and distributed.
(6) All the parties concerned shall agree on the outcome of the review and any action item responsibilities and closure criteria.

6.6.2 Project management reviews

The software project status shall be evaluated against the applicable project plans, and risks to the project's complete development in accordance with the plan shall be managed. If a project delay is detected and it is difficult to complete the development according to the plan, a change of plan including a revision to the schedule and software requirements specification shall be considered.

6.6.3 Technical reviews

Technical reviews shall be undertaken for software items from a technical viewpoint, and these reviews shall clarify the risks involved with the implementation of software and computer system that meet requirements specifications and standards.

This activity consists of the following tasks:

6.6.3.1 Review

(1) The following shall be implemented for review preparation:
    (a) Reviewers shall be selected for review implementation. Reviewers shall include appropriate personnel, consisting not only of parties to the work but also personnel who have enough

sufficient knowledge of the area under review, as well as project parties which include interface designers and the originator of the requirements.

(b) The review subjects and purposes shall be clarified, and materials for a review shall be completed in advance.

(c) The review purposes, subjects and reviewers shall be clarified, and documented.

(2) Software items shall be evaluated in a step-by-step approach as developments progress.

(3) Technical actions regarding software shall be evaluated.

(4) Review reports shall be developed after the review is completed. Also, quantitative data, such as the evaluation time, and the number of questions and comments at the review, shall be recorded, and a quality evaluation for the review shall be implemented.

6.6.3.2 Walk-through

A walk-through, including a peer review and so on, is an action to improve quality by detecting and removing errors early in the design and development. A walk-through shall be implemented, and performed mainly by the persons in charge, as necessary:

(1) As preparation for the walk-through, the reviewed party shall provide the relevant documents and codes, including those still in development.

(2) Parties necessary for walk-through (such as persons in charge of higher level process, persons in charge of lower process, and so on) shall check the documents and codes.

(3) Questions and problems found through walk-through shall be recorded and followed up until the resolution is completed, and mutually agreed.

6.7 Assessment process

The assessment process is a process for checking the process implementation status and identifying the items to be improved.

This process consists of the following activities:

(1) Process implementation

(2) Assessment implementation

6.7.1 Process implementation

The following directions for project designated personnel who are responsible for performing an assessment (hereafter, referred to as the "sponsor"), shall meet the following tasks and shall be planned and agreed to with sponsors:

(1) It shall be evaluated whether the software development process to be implemented meets the requirements of this standard. The strengths and the weaknesses of the process shall be identified.

(2) Personnel who are well informed of assessment methods and relevant standards shall be selected as assessors.

(3) Assessment results, including improvement offers on items needing improvement, are reported in documents to sponsors.

6.7.2 Assessment implementation

Based on the assessment plan, the assessment shall be implemented.

6.8 Problem resolution process

The problem resolution process consists of the following activities:

(1) Process implementation

(2) Problem resolution
(3) Prevention
(4) Problem trend analysis

6.8.1 Process implementation

The following shall be determined in advance for preparation of incidents and problems (notifications) occurrence regarding software products and services:

(1) Software products for management
(2) Management period
(3) Set of procedures for resolution (action)

If an incident or a problem has occurred, it shall be recorded and managed in accordance with the list above.

6.8.2 Problem resolution

This activity consists of the following tasks.

6.8.2.1 Incident and problem identification

If an incident has occurred, it shall be identified in order to be managed and linked to the known incidents or problems. When a corrective action is required, it shall be managed as a problem. It is recommended that it shall be prioritized for resolution.

Relevant parties shall be informed about the occurrence and status for both incidents and problems.

6.8.2.2 Incident and problem investigation and analysis

By doing the following activities, the incident phenomenon and occurrence condition shall be investigated. The root cause of a problem shall be cleared by investigation and analysis. Problem resolution shall be requested to the other organizations, as necessary.

(1) Record, analyze and classify incidents. Any incident that needs to take corrective action shall be identified as a problem.
(2) Record, analyze and classify the identified problems. Root causes shall be cleared by investigations.
(3) Analyze trends in incidents and problems.
(4) Inform of the status of incidents and problems.

6.8.2.3 Consideration for problem resolution

The following methods of problem resolution shall be considered:

(1) Ways of avoiding modification to, or of decreasing the impact on, the software product shall be considered.
(2) Determination of whether the software product requires a change shall be considered.

In addition, it is desirable that multiple options for resolution be prepared, including solutions which do not modify the software. If software modifications are required, the options must be evaluated by taking into consideration the points of view, such as the cost, the schedule, the overall risk, and the extent of the impact.

6.8.2.4 Determination of problem resolution plan

Plans shall be documented and agreed with the parties:

(1) As a temporary measure, if there is a way of mitigating the problem, plans shall be documented

and agreed with parties.

(2) With respect to a permanent solution, the problem resolution methods shall be documented and agreed with parties.

6.8.2.5 Problem resolution implementation

In accordance with the agreed plans, problem resolution shall be implemented, and users shall be notified. Track problems to closure.

6.8.2.6 Recording and monitoring

A set of incident or problem resolution actions and the status shall be recorded, monitored, and managed. In principle, after the step has been implemented, monitoring shall continue until it is determined that the problems are resolved, and no new problem has occurred.

6.8.3 Prevention

Improvements for processes and software products shall be identified to prevent the occurrence of known incidents and problems (relapse prevention).

Preventive measures shall be implemented if the risks that need to be prevented can be assessed.

6.8.4 Problem trend analysis

Incident and problem trend analysis shall ideally be performed.

Appendix

Appendix I Example of the overall configuration of the software development process (Water fall type)

Below is an example of the overall configuration of the software development process.

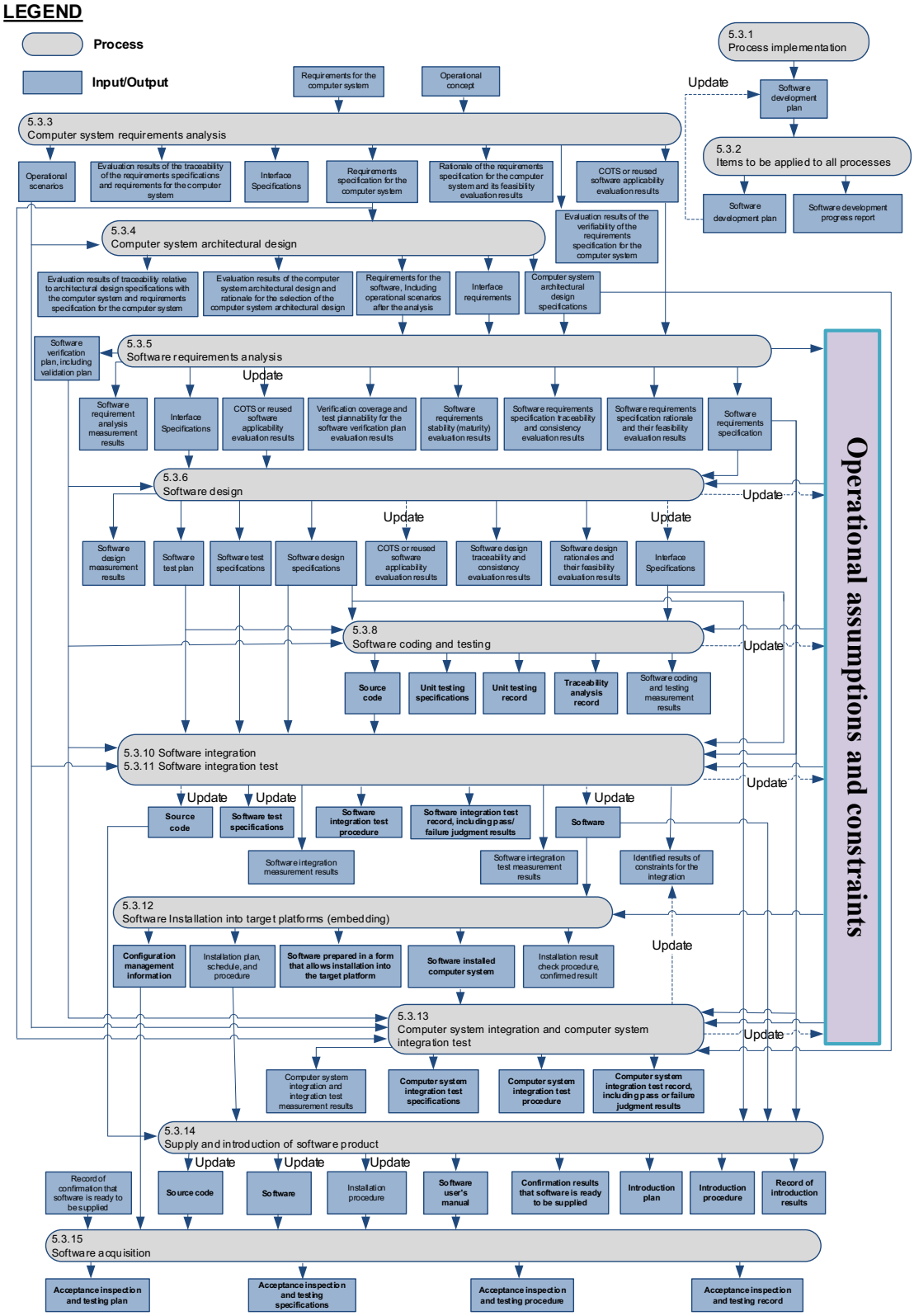


Figure I-1 Example of the overall configuration of the software development process

Appendix II　Matrix of input, output and processes

Below is the correspondence between inputs and outputs for each process in the software development process.

Table II-1 Matrix of input, output and processes

Input / Output	Process name	Process implementation	Items to be applied to all	Computer system requirements analysis	Computer system architectural design	Software requirements analysis	Software design	Not used	Software coding and testing	Not used	Software integration	Software integration test	Software installation into target platforms (embedding)	Computer system integration and computer system integration test	Supply and introduction of software product	Software acquisition
		5.3.1	5.3.2	5.3.3	5.3.4	5.3.5	5.3.6	5.3.7	5.3.8	5.3.9	5.3.10	5.3.11	5.3.12	5.3.13	5.3.14	5.3.15
Software development plan		O	M													
Software development progress report			O													
Requirements for the computer system				I												
Operational concept				I												
Operational scenarios				O	I							I		I		
Requirements specification for the computer system				O	I									I		
Interface specifications				O		M	M		I			I				
Evaluation results of the traceability of the requirements specifications and requirements for the computer system				O												
Rationale of the requirements specification for the computer system and its feasibility evaluation results				O												
COTS or reused software applicability evaluation results				O		M	M									
Evaluation results of the verifiability of the requirements specification for the computer system				O												
Computer system architectural design specifications					O	I								I		
Requirements for the software, including operational scenarios after the analysis					O	I										
Interface requirements					O	I										
Evaluation results of traceability relative to architectural design specifications with the computer system and requirements specifications for the computer system					O											
Evaluation results of the computer system architectural design and rationale for the selection of the computer system architectural design					O											
Software requirements specification						O	I					I		I	I	
Operational assumptions and constraints						O	M		M		I	M	I	M		
Software verification plan, including validation plan						O	I		I			I		I		
Verification coverage and test plannability for the software verification plan evaluation results						O										
Software requirements stability (maturity) evaluation results						O										
Software requirements specification traceability and consistency evaluation results						O										
Software requirements specification rationale and their feasibility evaluation results						O										
Software requirement analysis measurement results						O										
Software (architectural or detailed) design specifications							O		I		I	I			I	
Software design traceability and consistency evaluation results							O									
Software design rationales and their feasibility evaluation results							O									
Software design measurement results							O									
Software test plan							O		I			I				
Software test specifications							O					M				
Unit testing specifications									O							
Unit testing record									O							
Traceability analysis record									O							
Software coding and testing measurement results									O							
Source code									O		M	M			M	I
Software											O	M	M		M	I
Identified results of constraints for the integration											O			M		
Software integration measurement results											O					
Software integration test procedure												O				
Software integration test record, including pass or failure judgment results												O				
Software integration test measurement results												O				
Configuration management information													O			I
Installation plan, schedule, and procedure													O		M	I
Installation result check procedure, confirmed result													O			
Software installed computer system													O	I		
Software prepared in a form that allows installation into the target platform													O			
Computer system integration test specifications														O		
Computer system integration test procedure														O		
Computer system integration test record, including pass or failure judgment results														O		
Computer system integration and integration test measurement results														O		
Software user's manual															O	I
Confirmation results that software is ready to be supplied															O	
Record of confirmation that software is ready to be supplied																I
Introduction plan															O	
Introduction procedure															O	
Record of introduction results															O	
Acceptance inspection and testing plan																O
Acceptance inspection and testing specifications																O
Acceptance inspection and testing procedure																O
Acceptance inspection and testing record																O

Legend: I->Input,　O->Output,　M->Input and Output(updated)　　<Typical>

Appendix III "Verification" and "Validation"

As a supplementary, the difference in concept between "Verification" and "Validation" is shown in the below example.

The purpose of "Verification" is the confirmation that the specified requirements, determined in the previous process, have been fulfilled.

The purpose of "Validation" is the confirmation that the requirements for a specific intended use or application have been fulfilled, and that it makes the accumulated difference from the requirements, occurred with the progress of development, smaller.

The explanation of verification and validation:

"Verification" refers to the provision of objective evidence, that specified requirements have been fulfilled.

"Validation" refers to the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.

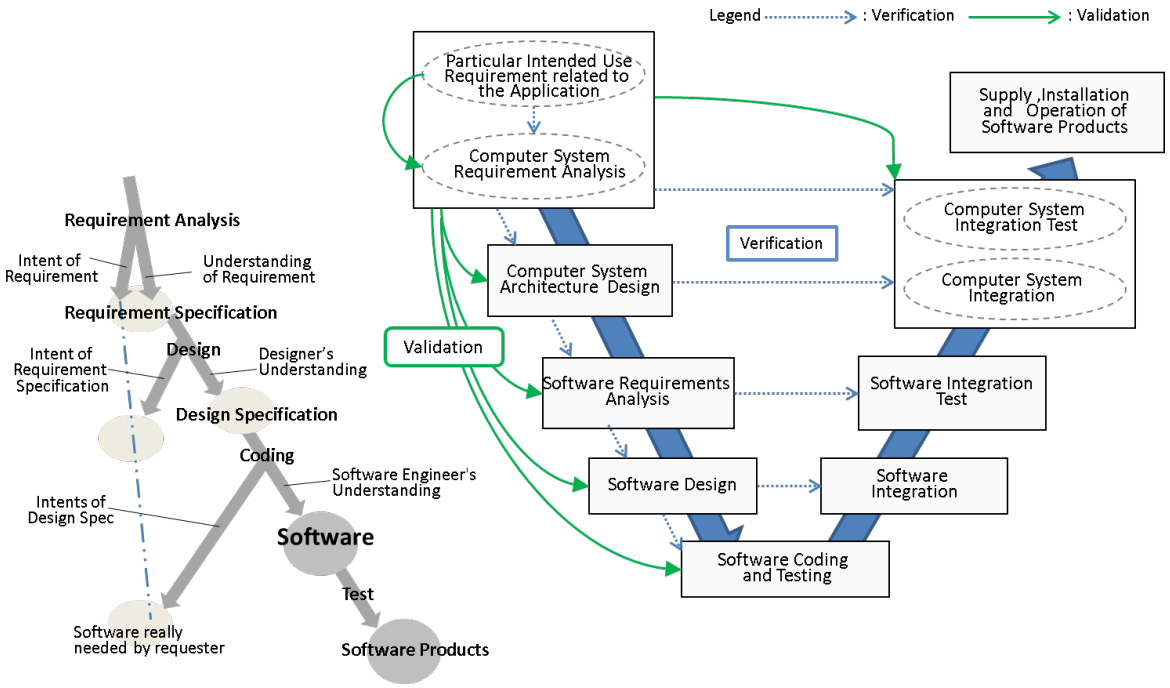


Figure III-1 Concept of verification and validation

Appendix IV Relationship between the problem resolution process and another process

As a supplementary between the relation of the problem resolution process and other processes, in case of incident occurrences during an operation, the relationship between the problem resolution process and the operation process are shown in below.

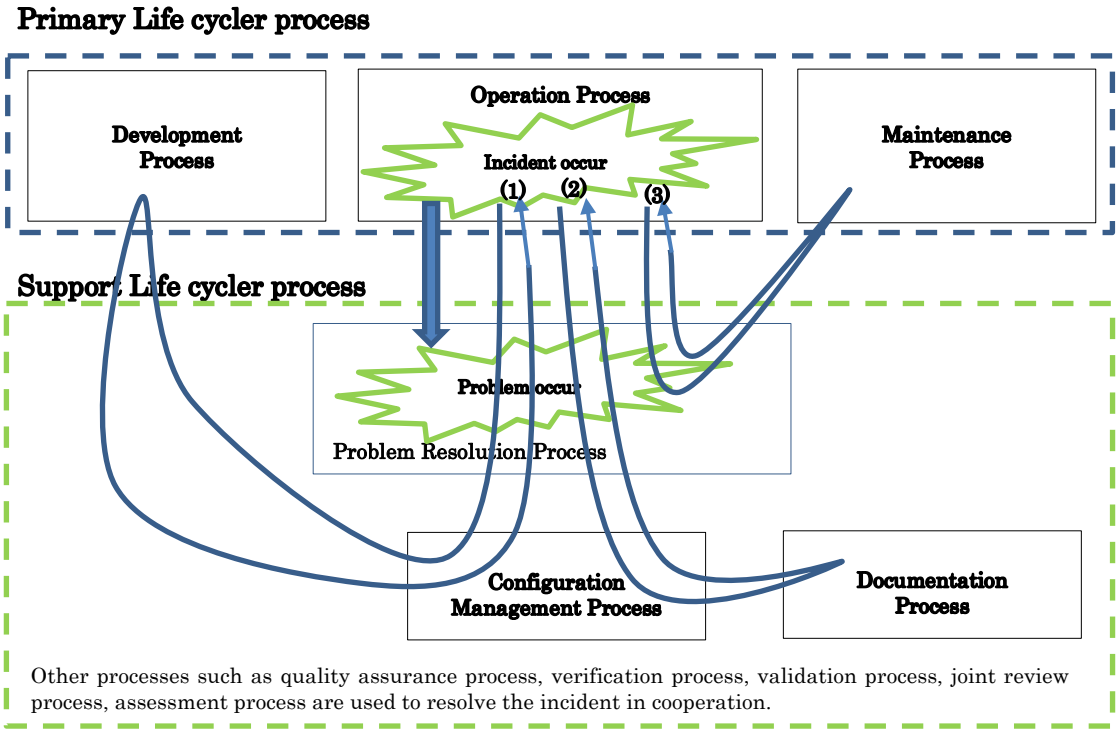


Figure IV-1 Relationship between the problem resolution process and another process

[(1) In case of reworked software]

Incidents occur during operation. → Sent to the problem resolution process. → As a result of the examination, a problem is identified, and the root cause and the countermeasure (modification of the software) is determined. → Take out software from the configuration management process. → Rework software in the development process. → Return software to the configuration management process. → Verify the results in the problem resolution process. → Return to the operation process.

[(2) In case of a document revision]

Incidents occur during operation. → Sent to the problem resolution process. → As a result of the examination, a problem is identified →As a result of the examination, the root cause and the countermeasure is determined. → Manage in the configuration process (judge whether under configuration) → Fix in the documentation process. → manage (update) in the configuration process (in case under configuration) → Verify the results in the problem resolution process. → Return to the operation process.

[(3) In case of countermeasure by operation]

Incidents occur during operation → Sent to the problem resolution process. → As a result of the examination, a problem is determined, and the root cause and a countermeasure is determined without modifying the software in order to support the operation. → Implement countermeasures operation in the maintenance process. → Check the results in the problem resolution process. → Return to the operation process.

Additionally, below is a conceptual example between the difference of incident and problem in the software development standard.

While an incident indicates an anomalous or unexpected event, condition, or situation that has occurred, a problem indicates an event, condition, or situation identified as one requiring investigation and corrective action as a result of the analysis of an incident that has been unknown.

In addition, some incidents include multiple problems.

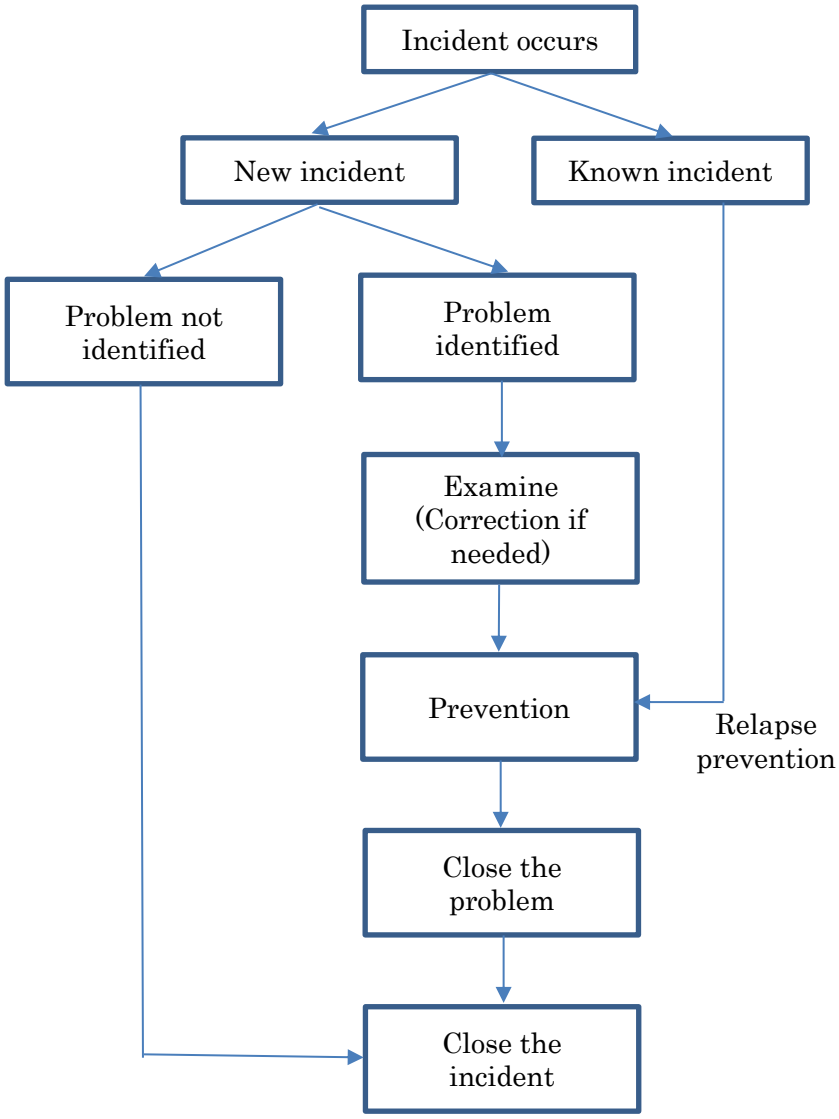


Figure IV-2 Procedure overview of incident and problem in the software development standard

Appendix V　Four maintenance types

As a supplementary of four maintenance types, the following is a four maintenance types relation example using a ground system.

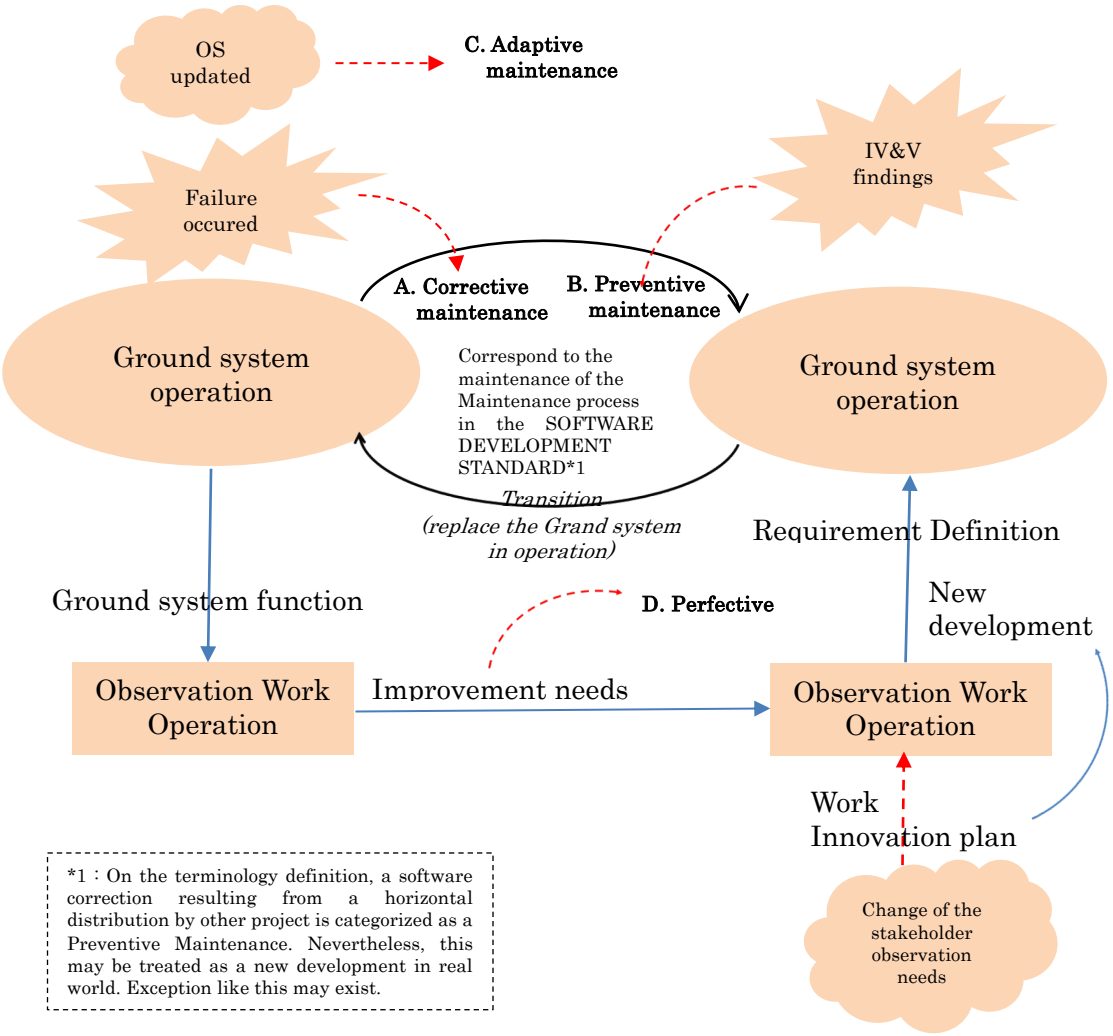


Figure V-1 A relationship of four maintenance types, Ground system and Observation work (Example)

Correction

[A. Corrective maintenance]

During operation, unintended event (failure) happens in the ground system. -> The ground system developer corrects the software and installs it to the operation environment.

[B. Preventive maintenance]

After the transition to the operation, potential bugs are found by performing IV&Vs for the ground system software. (Any failure caused by those bugs has not been happened.) -> The ground system developer corrects the software for the correction and installs them to the operation environment.

Improvement

[C. Adaptive maintenance]

The ground system developer regularly monitors the upgrade status of the operating system. -> The operating system is upgraded. -> The ground system developer corrects the software for the correction

and installs them to the operation environment because skipping of the corresponding activities to any operating system update leads out of date and increase of security risks.

[D. Perfective maintenance]

The design of the Observation work is modified based on the improvement needs of its operation shown by the tracking Control Team. To implement the before mentioned design result of the Observation work, the ground system developer corrects the software for the processing time improvement, additional functionalities and so on, and installs them to the operation environment.

Appendix VI Addition for software products, knowledge assets, and enabling systems

Knowledge assets and software products, and knowledge assets and enabling systems cannot be classified consistently and are overlapped. Their relationships are thus clarified in Figure VI-1.

Classification	Software products					Environment (Enabling systems or services)	
	Software		Source code	References			
Example	Software of catalog items (COTS)	Software developed by the developer's organization or external parties (including telemetry and command DBs (Note))	Source code developed by the developer's organization or external parties (including models in model based development, automatically-generated code, and FOSS (Free and Open Source Software))	All documents (covering design elements in terms of reuse and knowledge assets)	Processes, criteria, other technical information related to domain knowledge (such as training material), and lesson learned	Tools (e.g., those generating telemetry and command DBs) and data (e.g., simulated data such as dynamics) that reflect organizational knowledge gained by requiring organizational self-development or suppliers to incorporate their own organizational knowledge	Tools and data that do not reflect the knowledge of organizations (e.g., when COTS tools are used as they are)
Classification		Knowledge asset					

(Note) Some telemetry and command DBs are managed as knowledge assets of the system and hardware sides. This standard is not applied to telemetry and command DBs based on system and hardware design, but is applied to those based on software design.

Figure VI-1 Relationships between software products, knowledge assets, and enabling systems

Appendix VII Supplementary for the systems related to the computer system/software to be developed

This is a supplementary explanation of “If another state transition is defined separately for a system related to the computer system/software to be developed, the relationship between the state transitions shall be clarified” in 5.3.3.1 (1) (d) and 5.3.5.1 (3).

(1) Target state transitions of related systems

(a) Spacecraft system and launch vehicle system

The following state transitions shall be considered for the software to be developed.

- State transition defined by higher level systems
- State transition of lower level hardware
- State transition of a system/component that has data interfaces

(b) Ground system

Most related systems have data interfaces with a route necessary for operations. Therefore, the state transitions of all systems from data source to destination including the data relay system shall be the target (*). When the state transitions of related systems differ between software and hardware, they shall be considered as individual systems.

(*) The following systems shall be excluded.

- Relay systems that only relay regardless of the internal state transitions
- Systems that have interfaces designed not to be affected by the state transitions of the systems from data source to destination

(2) Relationships to be clarified

A spacecraft system is used as an example. When the software to be developed is in Component 2 in the system configuration, component configuration, and state transitions of the spacecraft system in Figure VII-1, the following relationship shall be clarified.

(a) Relationship with the higher level systems

- It shall be confirmed that the power-saving/nominal mode defined in the system level and the power-saving/nominal mode of the CPU of the software to be developed are consistent. If they are inconsistent, how that affects the software and what constraints are imposed shall be identified. The relationship with the state transitions of the software to be developed (safe/sun pointing/regular operation mode) shall be clarified. For example, no inconsistency shall be confirmed in the sustainable functions and performances of the software to be developed when the CPU is in the power-saving mode. The software to be developed shall be revised if necessary.
- The relationship between the attitude control mode defined by the system level and the state transitions of the software to be developed (safe/sun pointing/regular operation mode) shall be clarified. For example, which mode the state transition of the software to be developed enters shall be confirmed when the regular operation mode in the system level includes the sun pointing mode.

(b) Relationship with hardware

- When the software to be developed interfaces to Component 1 in the figure, the relationship between the power-saving/nominal mode defined in the system level and the software, CPU-1, and the mission hardware in Component 1 shall be identified (for example, the mission hardware is turned off in the power-saving mode) to clear the relationship with the state transition of the software to be developed.
- When the power-saving/nominal mode defined in the system level, state transitions of CPU-2 in which the software to be developed operates, and state transitions of interfaced CPU-1 work together, it shall be confirmed that the functions and performances of the software to be developed are not affected depending on the operations in state transitions of each CPU.

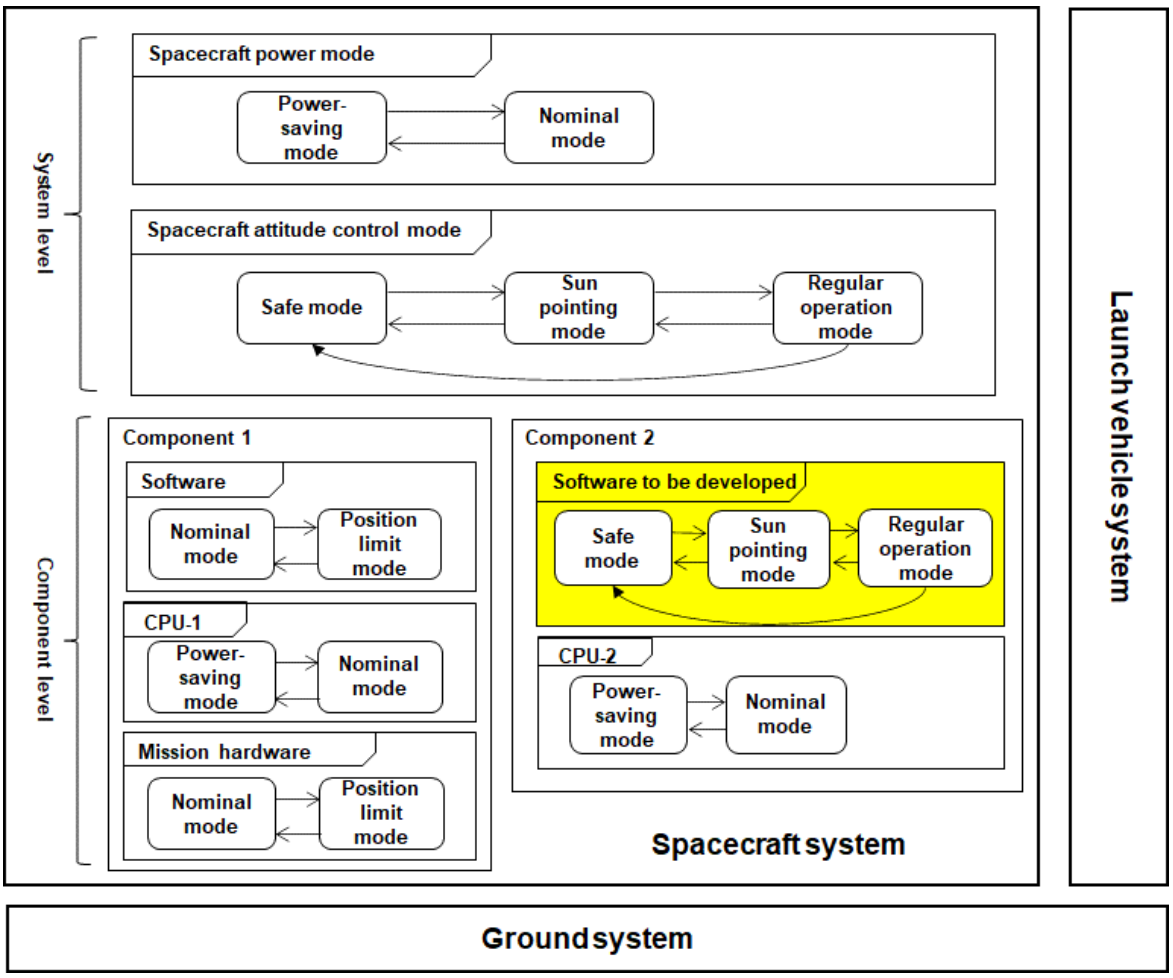
(c) Relationship with a component that has data interfaces

• When the software to be developed interfaces to Component 1 in the figure, the relationship between the attitude control mode defined in the system level and the software, CPU-1, and mission hardware in Component 1 shall be identified (for example, the mission hardware operates in the sun pointing mode) to clear the relationship with the state transition of the software to be developed.

(d) Relationship with a system that has data interfaces

• When the software to be developed has data interfaces to the launch vehicle system, the relationship with the state transitions of the launch vehicle system shall be clarified.

• When the software to be developed has data interfaces to the ground system, the relationship with the state transitions of the ground system shall be clarified.

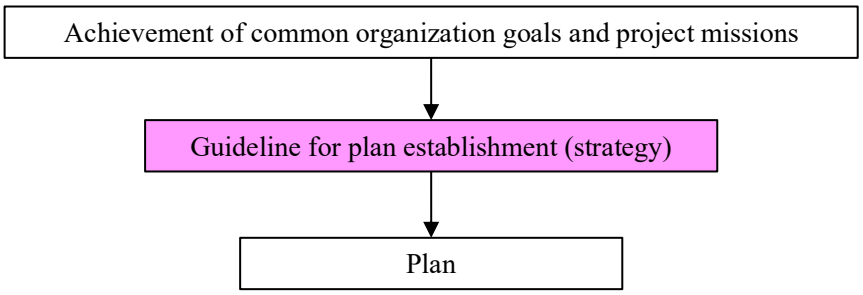


(Note: The system configuration, component configuration, and state transitions shown in the figure are shown as an example and differ from the actual system and software.)

Figure VII-1 Example of state transitions for software to be developed and related systems (spacecraft)

Appendix VIII Clarification of strategy

A strategy is a guideline defined for establishing plans according to the common organization goals (such as quality, productivity, and environment).



A strategy is assumed to have additional definitions of the necessary items required for software development in the organization, not only those described in this standard.

All contents to be implemented in the project shall be described in a plan, including those derived from sources other than the strategy. Coverage of the plan for the strategy shall not always be required.

Appendix IX Supplementary for “test plannability” and “testability”

In this standard, “test plannability” is evaluated in the software requirement analysis process, and “testability” is evaluated in the software design process. Differences in these two terms shall be added as follows from the viewpoint of purpose.

◆ “Test plannability”

The purpose of test plannability is to refine the descriptions of software requirements specification and interface specifications while confirming whether the descriptions can be tested.

The intent of the refinement confirms that each specification description is testable. For example, requirements stated with negative or vague expressions shall be revised to have positive or explicit expressions, otherwise, the software cannot be implemented and tested.

◆ “Testability”

The purpose of “testability” is to confirm the feasibility of the test itself, then, refine the design specifications, concretize the environment and test cases, and improve the test accuracy.

Appendix X N/A

Appendix XI Determination of Software Critical Classes (SWCC)

Software Criticality Class (SW CC) is a class assigned according to the characteristics of software functions (safety and reliability), etc. There are four levels: A, B, C, and D. Depending on the assigned SW CC, the requirements of the applicable software development standards change.

SW CCs are assigned according to the characteristics of software functions (Step 1) and the presence or absence of impact prevention functions (Step 2).

There are two steps, Step 1 and Step 2, but either step can be completed.

In addition, the SW CC assignment can be changed as the design progresses after it is assigned during software development. However, any changes should be agreed upon by all parties involved.

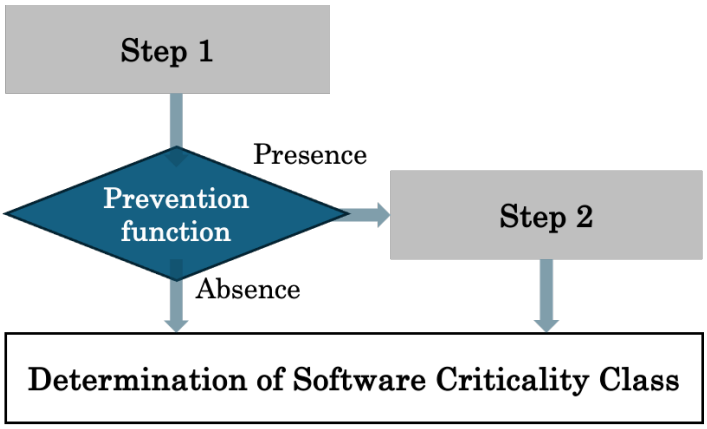


Figure-Appendix XI: Flow of determining SW CC

1 Step 1: Determining SW CC by software function characteristics

If SW CCs are uniformly defined for a target project, for example, software that does not significantly affect safety and reliability in terms of executable file units may need to be developed with excessive SW CCs. Therefore, it is possible to set different SW CCs for individual software within the same project, depending on the characteristics of the individual software functions.

Step 1 is evaluated using Table-Appendix XI-1. Individual software is evaluated in the order of SW CCs from A to D. The SW CC is determined when any one of the safety or reliability aspects is applicable. Software that may have a significant impact on reliability is identified as a CIL (Critical Item List) in the FMEA analysis based on Reference Document (11) JMR-004 and Reference Document (12) JERG-0-063. In addition, from a safety perspective, the concept of Reference Document (13) JMR-001 is used as a reference.

The software criticality class is determined by analyzing the impact on safety and reliability perspectives using FMEA and other methods to determine what results may result if individual software is not executed, if it is not executed correctly, or if it behaves abnormally.

Table - Appendix XI-1 SW CC by software function characteristics

SW CC	Safety perspective	Reliability perspective
A	・Death or severe personal damage*1 ・Irreversible significant environmental impact ・Impacts that are considered to severely damage public	Loss of all functions or missions of the system of the target project

SW CC	Safety perspective	Reliability perspective
	property or services, or the livelihoods of their owners, due to the inability to restore or financially compensate for public property or services or third-party property ・Loss of facilities or equipment that are difficult to replace, such as systems or launch site facilities outside of the target project	
B	・Major personal damage*2 ・Reversible significant environmental impact ・Impacts that are likely to temporarily damage public property or services, or the livelihood of the owners, etc., although restoration or financially compensate for public property or services or third-party property is possible. ・Severe damage to facilities and equipment that are difficult to replace, such as system or launch site facilities outside of the target project	Loss of some of the system's functions or mission-critical functions (i.e., functions that affect success criteria or customer satisfaction, excluding extras) of the target project
C	・Minor personal damage*3 ・Reversible moderate environmental impact by investing funds ・Minor damage to public or third party property to the extent that only monetary compensation is available ・Minor damage to systems outside of the subject project	Partial loss of function or mission or unrecoverable performance degradation of the system of the terget project
D	Any conditions that cause less damages above classes	Little or no impact

*1 : Severe personal damage: Damage that leaves a disability or is severe and critical (potentially life-threatening; see below) according to the criteria used by the fire department to ascertain the status of the injured or sick person.

*2 : Major personal damage : A non-disabling personal damage with or without lost workdays (Refer to JMR-001)

Personal damage with lost work days: Work-related injuries or illnesses, which requires the worker to be away from work from the next day of receiving the damage.

Personal damage without lost work days: Work-related injuries or illnesses, which do not require the worker to be away from work, after receiving treatment from medical facilities (including the clinic in the office). This also applies to the case where the worker was away from work for a less than a day.

*3 : Minor personal damage : Minor injuries resulting in for the worker to return to work immediately after receiving treatment. (same as JMR-001)

2 Step 2 : Changing SW CC based on impact prevention function

In Step 2, if the impact prevention function is taken, the SW CC can be changed to a SW CC that is one class lower than the SW CC assigned in Step 1.

The reason why only one class is used is that if the SW CC is repeatedly lowered to a lower class, it is considered difficult to ensure safety and reliability with the impact prevention function using a lower SW CC class.

Table - Appendix XI-2 Conditions under which SW CC can be changed

SW CC at step 1	Conditions under which SW CC can be changed
A	SW CC can be set to B if at least one of the following impact prevention features (compensating

	provisions) is taken. • A hardware implementation (e.g., implementation of a function to stop output of anomaly data, interlock function in case of anomaly, etc.) • SW CC A software (e.g., function to monitor software operation and stop the software if anomaly is detected, function to monitor the interface and prevent deviant data from being sent or received, etc.) • An operational procedure (e.g., periodic restart before buffer overflow)
B	SW CC can be set to C if at least one of the following impact prevention features (compensating provisions) is taken. • A hardware • SW CC B software • An operational procedure
C	SW CC can be set to D if at least one of the following impact prevention features (compensating provisions) is taken. • A hardware • SW CC C software • An operational procedure
D	-

It is necessary to demonstrate that there are no common-cause problems (failures) in the primary function and the impact prevention function, that there is no interference (failure of one does not affect the other), and that the impact prevention function is sufficient in time to effect for a failure.

Appendix XII　Criteria for application of each requirement of this standard

Projects may tailor the application of this Standard (tailoring) in accordance with 4.1, depending on the characteristics of the project.

Table-Appendix XII shows the mapping matrix for tailoring the requirements of this Standard to the established SW CCs. In the SW CC columns in the table, ◎ indicates that all the requirements including the subordinate ones are mandatory, ○ indicates that all the requirements including the subordinate ones can be adjusted, and a blank column indicates that the subordinate requirement is marked with ◎ or ○. The explanatory notes of ◎ and ○ in the table below are as follows.

◎Requirements that require justification if not applied

○Requirements that are recommended to be applied when possible (non-application is also acceptable)

Table-Appendix XII Requirements Mapping Matrix

JERG-0-049 requirements	SW CC			
	A	B	C	D
4.1 Tailoring	◎	◎	◎	◎
5 Primary life cycle processes				
5.3 Development process				
5.3.1 Process impleme When software development is started, activities that m(1eet the following requirements shall be performed:	◎	◎	◎	◎
(1) Define a development strategy including the following.	◎	◎	◎	◎
(a) Development policies and rules	◎	◎	◎	◎
(i) Suitable safety, security, privacy, and policy for environment activity	◎	○ Security, Privacy、◎ safety, policy for environment activity	○ Security, Privacy、◎ safety, policy for environment activity	○
(ii) Programming and coding standard	◎	◎	◎	◎
(iii) Unit testing policy	◎	◎	◎	◎
(b) In case of software reuse, conformation method of the applicability to the computer system and the safety of the acquisition route	◎	◎	◎	◎
(c) Performing of the software construction, peer review, and walkthrough review	◎	◎	◎	◎
(d) In case change management is conducted by not using any tool, how the configuration management is performed during software coding and testing	◎	◎	◎	◎
(e) Priorities in transitions of software and related data accompanied by computer system disposal	◎	◎	◎	◎
(f) Knowledge asset	◎	◎	◎	◎
(2) Based on the development strategy, the software development plan including the following information shall be established to cover:	◎	◎	◎	◎
(3) Software development plan shall be documented and approved.	◎	◎	◎	◎
* Including activities regarding computer system with software installed.	◎	◎	◎	◎
5.3.1.1 Output	◎	◎	◎	◎
5.3.2 Items to be applied to all processes	◎	◎	◎	◎
5.3.3 Computer system requirements analysis				
5.3.3.1 Activity Activities that meet the following requirements shall be performed with respect to the computer system requirements analysis:	◎	◎	◎	◎
(1) Requirements extraction	◎	◎	◎	◎
(2) Requirements specifications development	◎	◎	◎	◎
(a) Feasibility and consistency shall be confirmed, based on the operational scenarios, and the requirements specification for the computer system shall be defined.	◎	◎	◎	◎
(b) Specifications for data and databases to be handled by the computer system shall be included in the requirements specification.	◎	◎	◎	◎
(c) Risks, computer system severity, and specifications for important quality characteristics shall be included in the requirements specification.	◎	◎	◎	◎
(d) Interface requirements shall be analyzed, and requirements specifications shall then be developed. Agreement with the relevant parties on the interface	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
requirements shall be made based on a common understanding and interpretation of the contents.				
(e) The rationale for the requirements specification for the computer system shall be clarified, and the traceability of higher level requirements for the computer system shall be evaluated and maintained.	◎	◎	◎	○
(f) Rationale and consideration methods of the individual requirement of the requirements specifications shall be clarified, and their feasibility shall be evaluated.		◎	◎	◎
(g) If COTS or reused software is used, its applicability with the requirements specifications shall be analyzed.	◎	◎	◎	◎
(h) In addition to the feedback of the requirement analysis contents to appropriate stakeholders, developed requirements specifications shall be reviewed and approved by the stakeholders.	◎	◎	◎	◎
(i) Problems, inadequacies, and inconsistencies included in the requirements specifications shall be identified and planned to resolve them.	◎	◎	◎	◎
(j) Verifiability of individual requirement of the requirements specifications shall be evaluated.	◎	◎	◎	◎
5.3.3.2 Input				
5.3.3.3 Output				
(1) Operational scenarios	◎	◎	◎	◎
(2) Requirements specification for the computer system	◎	◎	◎	◎
(3) Interface specifications	◎	◎	◎	◎
(4) Evaluation results of the traceability of the requirements specifications and requirements for the computer system	◎	◎	◎	○
(5) Rationale of the requirements specification for the computer system and its feasibility evaluation results	◎	◎	◎	◎
(6) COTS or reused software applicability evaluation results	◎	◎	◎	◎
(7) Evaluation results of the verifiability of the requirements specification for the computer system	◎	◎	◎	◎
5.3.4 Computer system architectural design				
5.3.4.1 Activity				
Activities that meet the following requirements shall be performed for the computer system architectural design:	◎	◎	◎	◎
(1) Computer system architecture shall be designed based on the requirements specification for the computer system and the operational scenarios. Configuration items and their various categories (hardware, firmware, software, and operational) shall be clarified.	◎	◎	◎	◎
(2) Requirements pertaining to the requirements specification for the computer system shall be allocated among the individual configuration items of the system.	◎	◎	◎	◎
(3) Computer system architectural design specifications shall be developed by combining the results of the above-mentioned design.	◎	◎	◎	◎
(4) Feasibility of software items in fulfilling their allocated requirements shall be evaluated.	◎	◎	◎	◎
(5) Rationale for the design and preconditions (e.g. operational assumptions) for the computer system architectural design specifications shall be identified, and an appropriate evaluation shall be performed.	◎	◎	◎	◎
(6) Traceability of the computer system architectural design specifications relative to higher level requirements, such as the requirements specification for the computer system, shall be evaluated.	◎	◎	◎	○
(7) Interface requirements for the software shall be extracted.	◎	◎	◎	◎
(8) Set the evaluation criteria for a computer system architectural design based on the requirements specification for the computer system and operational scenarios. The computer system architecture design shall be evaluated by that. Additionally, the computer system architectural design selection rational shall be recorded.	◎	◎	◎	◎
5.3.4.2 Input				
5.3.4.3 Output				
(1) Computer system architectural design specifications	◎	◎	◎	◎
(2) Requirements for the software, including operational scenarios after the analysis	◎	◎	◎	◎
(3) Interface requirements	◎	◎	◎	◎
(4) Evaluation results of traceability relative to architectural design specifications with the computer system and requirements specification for the computer system	◎	◎	◎	○
(5) Evaluation results of the computer system architectural design and rationale for the selection of the computer system architectural design	◎	◎	◎	◎
5.3.5 Software requirements analysis				
5.3.5.1 Activity				
The following activities shall be performed for the software requirements analysis:	◎	◎	◎	◎
(1) Software requirements specification shall be developed, based upon the analysis of the computer system architectural design specifications, interface	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
requirements, and requirements for software including non-functional requirements.				
(2) The required software state transition (including operation mode) shall be identified.	◎	◎	◎	◎
(3) If another state transition is defined separately for a system related to the software to be developed, the relationship between the state transitions shall be clarified (refer to Appendix VII).	◎	◎	◎	◎
(4) Identifiers shall be included in the individual requirement of software requirements specification.	◎	◎	◎	◎
(5) Specifications for data and databases to be handled by the software shall be included in the software requirements specification.	◎	◎	◎	◎
(6) Specifications for failure detection and handling functions shall be included in the software requirements specification.	◎	◎	◎	◎
(7) Risks, software severeness, and specifications for important quality characteristics shall be included in the software requirements specification.	◎	◎	◎	◎
(8) User interface (in case having), information provided to users, and specifications for user trainings shall be included in the software requirements specification.	◎	◎	◎	◎
(9) In case of a software transition to an on-going system, specifications for the satisfaction of the transition condition shall be included in the software requirements specification.	◎	◎	◎	◎
(10) Interface requirements shall be analyzed, and interface specifications shall then be developed. Agreement with the relevant parties regarding the interface specifications shall be made based on a common understanding and interpretation of the contents.	◎	◎	◎	◎
(11) Traceability and consistency of the software requirements specification relative to the computer system architectural design specifications and interface requirements shall be analyzed and documented.	◎	◎	◎	○
(12) Rationale of individual requirement of the software requirements specification shall be clarified, and their feasibility shall be evaluated.	◎	◎	◎	◎
(13) If COTS or reused software is used, compliance with the software requirements specification and its applicability with the computer system architectural design specifications shall be analyzed.	◎	◎	◎	◎
(14) Operational assumptions and constraints regarding software requirements specification shall be extracted.	◎	◎	◎	◎
(15) In addition to the feedback of the requirement analysis contents to appropriate stakeholders, developed software requirements specification shall be reviewed and approved by the stakeholders.	◎	◎	◎	◎
(16) Problems, inadequacies, and inconsistencies included in the software requirements specification shall be identified and planned to resolve them.	◎	◎	◎	◎
(17) Verifiability of the individual requirements of the software requirements and interface specifications shall be evaluated, and a software verification plan, including the validation method, shall be established.	◎	◎	◎	◎
(18) Software verification coverage pertaining to software function, performance, and operational scenarios in the verification plan shall be evaluated, and test plannability regarding the software requirements specification and interface specifications shall be evaluated.	◎	◎	◎	◎
(19) With regard to the software verification plan, whether the test is affected by the behavioral difference between the test environment and the real hardware, or whether verification is performed by review, analysis and so on, without testing, the evaluation that shows the adequate identification and verification methods shall be included.	◎	◎	◎	◎
5.3.5.2 Measurement	◎	◎	○	○
5.3.5.3 Input				
5.3.5.4 Output				
(1) Software requirements specification	◎	◎	◎	◎
(2) Interface specifications	◎	◎	◎	◎
(3) Software requirements specification traceability and consistency evaluation results	◎	◎	◎	○
(4) Software requirements specification rationale and their feasibility evaluation results	◎	◎	◎	◎
(5) COTS or reused software applicability evaluation results	◎	◎	◎	◎
(6) Operational assumptions and constraints	◎	◎	◎	◎
(7) Software verification plan, including validation plan	◎	◎	◎	◎
(8) Verification coverage and test plannability for the software verification plan evaluation results	◎	◎	◎	◎
(9) Software requirements stability (maturity) evaluation results	◎	◎	○	○
(10) Software requirement analysis measurement results	◎	◎	○	○
5.3.5.5 Review	◎	◎	◎	○
5.3.6 Software design				
5.3.6.1 Activity Activities that meet the following requirements shall be performed for the software design:	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
Software architectural design	◎	◎	◎	◎
(1) The design guidelines (software architecture, etc.) and the design characteristics to be considered shall be selected, and the design shall be performed by considering priorities.	◎	◎	◎	◎
(2) Functional decomposition and module partitioning shall be performed based on the software requirements specification and the relationships between the modules comprising the functions, and the structures between modules and themselves shall be clarified, so that an appropriate software architectural design is performed.	◎	◎	◎	◎
(3) Software architectural design shall include the design and distribution of non-functional requirements (processing time requirements, requirements of resources such as memory, and so on) and be defined as software requirements.	◎	◎	◎	◎
(4) nterface specifications shall be detailed in accordance with the considerations of the boundaries and interrelations, and the decomposition of the software functions and modules. Agreement with the relevant parties as to the interface specifications shall be arrived at based on a common understanding and common interpretation of the contents.	◎	◎	◎	◎
Software detailed design	◎	◎	◎	◎
(5) Each individual module shall be designed in accordance with the decomposition of functions and modules, and a software detailed design shall be performed.	◎	◎	◎	◎
(6) Interface specifications shall be detailed in accordance with the considerations of the boundaries and interrelations, and the design of the module. Agreement with the relevant parties on the interface specifications shall be made based on a common understanding and interpretation of the contents.	◎	◎	◎	◎
Common to software architectural and detailed designs	◎	◎	◎	◎
(7) Software (architectural and detailed) design specifications shall be developed based on the result of the software design.	◎	◎	◎	◎
(8) The needed design methods or organizationally maintained past knowledge shall be identified, prepared and acquired.	◎	◎	◎	◎
(9) Traceability and consistency of the software design with the software requirements specification, interface specifications, and with the necessary related documents shall be analyzed, documented and maintained.	◎	◎	◎	○
(10) Operational assumptions and constraints regarding the software design shall be identified.	◎	◎	◎	◎
(11) As necessary, with regard to the individual software design, the design rationale shall be clarified, and its feasibility and testability (including test case) evaluated. (Refer to Appendix IX.)	◎	◎	◎	◎
(12) If COTS or reused software is used, its applicability with the software design shall be analyzed.	◎	◎	◎	◎
(13) Software test plans and specifications shall be established in accordance with the software verification plan.	◎	◎	◎	◎
(14) If new operational assumptions and constraints arise or are identified, they shall be updated.	◎	◎	◎	◎
5.3.6.2 Measurement	◎	◎	○	○
5.3.6.3 Input				
5.3.6.4 Output				
(1) Software (architectural and detailed) design specifications	◎	◎	◎	◎
(2) Interface specifications (updated)	◎	◎	◎	◎
(3) Software design traceability and consistency evaluation result	◎	◎	◎	○
(4) Software design rationales and their feasibility evaluation results	◎	◎	◎	◎
(5) COTS or reused software applicability evaluation results (updated)	◎	◎	◎	◎
(6) Operational assumptions and constraints (updated)	◎	◎	◎	◎
(7) Software test plan	◎	◎	◎	◎
(8) Software test specification	◎	◎	◎	◎
(9) Software design measurement results	◎	◎	○	○
5.3.6.5 Review	◎	◎	◎	○
5.3.8 Software coding and testing				
5.3.8.1 Activity				
Activities that meet the following shall be performed for the software coding and testing:	◎	◎	◎	◎
(1) Source codes shall be developed based on constrains, the software design specifications and interface specifications, and reviewed.	◎	◎	◎	◎
(2) Definitive implementation guidelines for error handling shall be considered.	◎	◎	◎	◎
(3) Source code shall be developed based on the defined coding standard.	◎	◎	◎	◎
(4) Static analysis shall be performed with a source code checking tool or equivalent, and the source quality shall be evaluated.	◎	◎	◎	◎
In addition, the Cyclomatic Complexity criteria in Table 5.3.8-1 shall be satisfied.	Refer to Table 5.3.8-1			
(5) Unit testing specifications shall be developed in accordance with a software verification plan and a software test plan.	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
(6) Unit testing shall be performed in accordance with the unit testing specifications, and the test results shall be recorded in a format that it allows determination of pass or failure.	◎	◎	◎	◎
(7) For unit testing, the criteria in Table 5.3.8-2 for the test coverage for source code shall be established and the test shall be performed so that these criteria are satisfied. If the test coverage criteria cannot be achieved, analysis, inspection, or design review shall be applied to the untested code.	Refer to Table 5.3.8-2			
(8) The traceability of the source code and software design specifications shall be analyzed, and the results shall be recorded. The correspondence between the naming convention (variable name, function name, etc.) in source codes and the design specifications shall be checked.	◎	◎	◎	○
(9) If new operational assumptions and constraints arise or are identified, they shall be updated.	◎	◎	◎	◎
5.3.8.2 Measurement	◎	◎	○	○
5.3.8.3 Input				
5.3.8.4 Output				
(1) Source code	◎	◎	◎	◎
(2) Operation assumptions and constraints (updated)	◎	◎	◎	◎
(3) Unit testing specifications	◎	◎	◎	◎
(4) Unit testing record	◎	◎	◎	◎
(5) Traceability analysis record	◎	◎	◎	○
(6) Software coding and testing measurement results	◎	◎	○	○
5.3.8.5 Review	◎	◎	○	○
5.3.10 Software Integration				
5.3.10.1 Activity	◎	◎	◎	◎
5.3.10.2 Measurement	◎	◎	○	○
5.3.10.3 Input				
5.3.10.4 Output				
(1) Source code (integrated)	◎	◎	◎	◎
(2) Software (integrated)	◎	◎	◎	◎
(3) Identified results of constraints for the integration	◎	◎	◎	◎
(4) Software integration measurement results	◎	◎	○	○
5.3.11 Software integration test				
5.3.11.1 Activity	◎	◎	◎	◎
5.3.11.2 Measurement	◎	◎	○	○
5.3.11.3 Input				
5.3.11.4 Output				
(1) Software integration test procedure	◎	◎	◎	◎
(2) Software integration test record, including pass or failure judgment results	◎	◎	◎	◎
(3) Operational assumptions and constraints (updated)	◎	◎	◎	◎
(4) Source code (tested)	◎	◎	◎	◎
(5) Software (tested)	◎	◎	◎	◎
(6) Software test specifications (updated)	◎	◎	◎	◎
(7) Software integration test measurement results	◎	◎	○	○
5.3.11.5 Review	◎	◎	◎	○
5.3.12 Software installation into target platforms (embedding)				
5.3.12.1 Activity	◎	◎	◎	◎
5.3.12.2 Input				
5.3.12.3 Output	◎	◎	◎	◎
5.3.13 Computer system integration and computer system integration test				
5.3.13.1 Activity	◎	◎	◎	◎
5.3.13.2 Measure	◎	◎	○	○
5.3.13.3 Input				
5.3.13.4 Output				
(1) Computer system integration test specifications	◎	◎	◎	◎
(2) Computer system integration test procedure	◎	◎	◎	◎
(3) Computer system integration test record, including pass or failure judgment results	◎	◎	◎	◎
(4) Operational assumptions and constraints (updated)	◎	◎	◎	◎
(5) Identified results of constraints for the integration (after updated)	◎	◎	◎	◎
(6) Computer system integration and integration test measurement results	◎	◎	○	○
5.3.14 Supply and introduction of software product				
5.3.14.1 Activity	◎	◎	◎	◎
5.3.14.2 Input				
5.3.14.3 Output	◎	◎	◎	◎
5.3.15 Software acceptance				

JERG-0-049 requirements	SW CC			
	A	B	C	D
5.3.15.1 Activity	◎	◎	◎	◎
5.3.15.2 Input				
5.3.15.3 Output	◎	◎	◎	◎
5.4 Operation process				
5.4.1 Process Implementation				
5.4.1.1 Definition of an operation strategy (1) An operation strategy shall be defined. In principle, the following shall be considered in the strategy.	◎	◎	◎	◎
(a) Expected each criteria (capacity, occupancy rate, response, safety, etc.) between service installation, periodic operation and disposal	◎	◎	◎	◎ Other than Safety, ○ Safety
(b) Software or computer system release criteria and schedule considering the correction to maintain the current service	◎	◎	◎	◎
(c) Method to implement each operation mode (regular operation/preparation stage/operation test/assumed occurrence of disasters and troubles)	◎	◎	◎	◎
(d) Operation metrics to evaluate performance level	◎	◎	◎	◎
5.4.1.2 Establishment of an operational plan	◎	◎	◎	◎
5.4.1.3 Establishment of problem management for the operation	◎	◎	◎	◎
5.4.1.4 Establishment of operational procedures for the computer system including software operation and user support	◎	◎	◎	◎
5.4.2 Operational testing	◎	◎	◎	◎
5.4.3 Operation of computer system including software	◎	◎	◎	◎
5.4.4 Operation results management	◎	◎	◎	◎
5.4.5 Customer and user support	◎	◎	◎	◎
5.5 Maintenance process				
5.5.1 Process implementation				
5.5.1.1 Definition of a maintenance strategy	◎	◎	◎	◎
5.5.1.2 Establishment of a maintenance plan	◎	◎	◎	◎
5.5.2 Problem identification and modification analysis	◎	◎	◎	◎
5.5.3 Modification implementation	◎	◎	◎	◎
5.5.4 Software reprogramming	◎	◎	◎	◎
5.5.5 Perform logistics support	◎	◎	◎	◎
5.5.6 Manage results of maintenance and logistics	◎	◎	◎	◎
5.5.7 Transition				
5.5.7.1 Define a transition strategy	◎	◎	◎	◎
5.5.7.2 Establishment and execution of transition plan	◎	◎	◎	◎
5.5.7.3 Notification to users	◎	◎	◎	◎
5.5.7.4 Manage transition results	◎	◎	◎	◎
5.5.7.5 Storage of old environments	○	○	○	○
5.5.8 Software disposal The following viewpoints shall be considered for the software disposal:	◎	◎	◎	◎
(1) A disposal strategy to remove active support by an institution engaged in operation and maintenance shall be defined. In principle, the following shall be considered in the strategy.	◎	◎	◎	◎
(a) Identification of permanent termination of the system's functions and delivery of services	◎	◎	◎	◎
(b) Identification of ownership and responsibility for retention or destruction of data and intellectual property in the software system	◎	◎	◎	◎
(c) Transformation of the product into, or retention in a socially and physically acceptable state, thereby avoiding subsequent adverse effects on stakeholders, society and the environment.	◎	◎	◎	◎
(d) Disposal actions that reflect health, safety, security, and privacy concerns based on the long-term condition of hardware and data	◎	○ Security, Privacy、◎ safety	○ Security, Privacy、◎ safety	○
(e) Notification to relevant stakeholders of significant disposal activities	◎	◎	◎	◎
(f) Identification of schedules, actions, responsibilities, and resources for disposal activities	◎	◎	◎	◎
(g) Identification of the constraints on disposal for system/software requirements, architecture and design characteristics, or implementation techniques	◎	◎	◎	◎
(h) Identification and planning for the necessary enabling systems or services needed to support disposal.	◎	◎	◎	◎
(i) Obtainment of the necessary enabling systems and services or acquisition of access to them.	◎	◎	◎	◎
(j) Containment facilities, storage locations, inspection criteria and storage periods, if the software system or data is to be stored	◎	◎	◎	◎
(k) Preventive methods to preclude disposed software	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
(2) A disposal plan shall be developed based on the disposal strategy. Users shall be included in the development of the plan.	◎	◎	◎	◎
(3) Users shall be given notification of the disposal plans and activities. Notifications shall include the following:	◎	◎	◎	◎
(4) According to the disposal plan, the following activities shall be performed.	◎	◎	◎	◎
(5) When the scheduled disposal arrives, notification shall be sent to all concerned parties. It is recommended that all associated development documentation, logs, and code shall be archived.	◎	◎	◎	◎
(6) For the disposed software or computer system, data shall be collected to permit audits and reviews in the event of long-term hazards to health, safety, security, privacy, and the environment.	◎	○ Security, Privacy、◎ safety	○ Security, Privacy、◎ safety	○
6 Supporting life cycle process				
6.1 Documentation process	◎	◎	○	○
6.2 Configuration management process				
6.2.1 Process implementation				
6.2.1.1 Definition of a configuration management strategy	◎	◎	◎	◎
6.2.1.2 Establishment of a configuration management plan	◎	◎	◎	◎
6.2.2 Configuration identification	◎	◎	◎	◎
6.2.3 Configuration change control	◎	◎	◎	◎
6.2.4 Record of configuration change status	◎	◎	◎	◎
6.2.5 Evaluation of configuration change status	◎	◎	◎	◎
6.2.6 Release management and delivery The release and delivery of software product shall be formally controlled in accordance with the procedure. The source code and documentation that contain safety or security critical functions shall be handled, stored, packaged, and delivered in accordance with the policies of organizations involved.	◎	○ Security, ◎ Other than security	○ Security, ◎ Other than security	○ Security, ◎ Other than security
The distribution of the assigned software product releases shall be managed.	◎	◎	◎	◎
6.2.7 Configuration audit implementation	◎	◎	◎	◎
6.3 Quality assurance process				
6.3.1 Process implementation				
6.3.1.1 Confirmation of the independence of the Organization and establishment of the adequate structure	◎	○ Security, ◎ Other than security	○ Security, ◎ Other than security	○ Security, ◎ Other than security
6.3.1.2 Definition of a quality assurance strategy	◎	◎	◎	◎
6.3.1.3 Establishment of a quality assurance activity plan	◎	◎	◎	◎
6.3.2 Product and service quality assurance (1) Products and services shall be assured to fulfil their plans.	◎	◎	◎	◎
(2) Through the verification and validation of the software product or computer system development process for each product, the product shall be assured to fulfil its approved requirement.	◎	◎	◎	◎
(3) Products quality shall be assured by IV&V as necessary.	◎	◎	○	○
6.3.3 Process assurance	◎	◎	◎	◎
6.3.4 Assurance of quality system				
6.3.4.1 Education and training				
All techniques, abilities, and qualifications needed for personnel engaged in development, maintenance and operation work with the software or computer system shall be identified, and education and training shall be conducted.	◎	◎	◎	○
6.3.4.2 Purchase management and supplier management	◎	◎	◎	◎
6.3.4.3 Management of items supplied by acquirer	◎	◎	◎	◎
6.3.4.4 Management of existing software items (COTS or knowledge assets)	◎	◎	◎	◎
6.3.4.5 Handling, storing, and labeling	◎	◎	◎	◎
6.3.5 Manage quality assurance records	◎	◎	◎	◎
6.4 Verification process				
6.4.1 Process Implementation	◎	◎	◎	◎
6.4.2 Verification This activity consists of the following tasks.				
6.4.2.1 Process verification	◎	◎	◎	◎
6.4.2.2 Requirements verification The following shall be considered:	◎	◎	◎	◎
(1) The requirements are consistent, feasible, and verifiable.	◎	◎	◎	◎
(2) Requirements for software items are appropriately allocated (not including requirements for hardware items and operation).	◎	◎	◎	◎
(3) The higher level requirements and the standards applied to items shall be satisfied.	◎	◎	◎	◎

JERG-0-049 requirements	SW CC			
	A	B	C	D
(4) Concerning to the requirement to be especially taken care such as safety and security and so on, it is able to show the proper method that it satisfies the higher level requirements and the standards applied to items.	◎	○ Security, ◎ Other than security	○ Security, ◎ Other than security	○ Security, ◎ Other than security
6.4.2.3 Design verification The following shall be considered:	◎	◎	◎	◎
(1) The design shall meet requirements and shall be traceable to requirements.	◎	◎	◎	○
(2) Designed properly with respect to data interface, timing, computer resource (memory capacity, processing speed, and so on), logic design, processing sequence and processing contents (especially initialization, termination, exception handling and so on).	◎	◎	◎	◎
(3) The characteristics such as portability, modifiability and ease of problem resolution have been covered.	◎	◎	◎	◎
6.4.2.4 Source code verification The following viewpoint shall be considered:	◎	◎	◎	◎
(1) Source code shall meet design and shall be traceable to the design.	◎	◎	◎	○
(2) Implemented properly with respect to data interface, timing, computer resources (memory capacity, processing speed, and so on.), logic design, processing sequence and processing contents (especially initialization, termination, exception handling and so on).	◎	◎	◎	◎
(3) The characteristics such as portability, modifiability and ease of problem resolution have been covered.	◎	◎	◎	◎
(4) Concerning to the source code to be especially taken care such as safety and security and so on, it is able to show the proper method that it satisfies the requirements and the standards applied to items.	◎	○ Security, ◎ Other than security	○ Security, ◎ Other than security	○ Security, ◎ Other than security
(5) Source code shall conform to e.g. coding standards.	◎	◎	◎	◎
6.4.2.5 Integration verification	◎	◎	◎	◎
6.4.2.6 Documentation verification	◎	◎	○	○
6.4.3 Manage the verification results	◎	◎	◎	◎
6.5 Validation process	◎	◎	◎	◎
6.6 Joint review process	◎	◎	◎	○
6.7 Assessment process	◎	◎	○	○
6.8 Problem resolution process				
6.8.1 Process implementation	◎	◎	◎	◎
6.8.2 Problem resolution	◎	◎	◎	◎
6.8.3 Prevention	◎	◎	◎	○
6.8.4 Problem trend analysis	◎	◎	◎	○