General

SOFTWARE DEVELOPMENT STANDARD FOR SPACECRAFT

Mar. 30, 2021

Japan Aerospace Exploration Agency

This is an English translation of JERG-2-610B. Whenever there is anything ambiguous in this document, the original document (the Japanese version) shall be used to clarify the intent of the requirement.

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

Table of Contents

1 Scope

This standard applies to the activities relating to a computer system (each subsystem, component, mission sensor system, and so on) boarded on spacecraft and the development, operation and maintenance of software, and also necessary system development activities and related support activities. When this standard is applied, embodying and tailoring may be performed in accordance with the characteristics of specific projects and other factors.

2 References

2.1 Document for compliance

(1) JERG-0-049B Software Development Standard

2.2 Informative references

(1) ISO/IEC/IEEE 12207:2017 Information technology - 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)

3 Terms, definitions and abbreviated terms

The definitions of terms and abbreviated terms used in this standard shall basically correspond to JERG-0-049B. Terms redefined into the details and newly defined are shown below.

3.1 Terms redefined in this standard

Term	Definition
Incident	An anomalous or unexpected event, set of events, condition, or situation. It can occur at any time during the life cycle of a project, product, service, or system. [ISO/IEC/IEEE 12207:2017]
Knowledge asset	A set of existing knowledge managed by an organization for reuse. It indicates a generic name of assets including software products, processes, criteria, and other technical information related to domain knowledge (such as training material), lesson learned, and environment reflecting the knowledge of the organization (enabling systems or services). (Refer to Appendix 1)
Model based technology	A development method using models.

Term	Definition
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 file maintenance functions, and so on.
Traceability	The correspondence between higher and lower level specifications, or between other similar ones.
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. [JERG-0-049B, ISO9000]. All activities to verify that the software satisfies the requirements and design specifications. Beside tests as the verification procedure, these include simulation, walk-through, document review such as source code review, inspection, demonstration, and so on.

3.2 Terms newly defined in this standard

Term	Definition
Automatic code generation	The automatic generation of code equivalent to a high-level programming language from source data (e.g., models) for automatic code generation for the flight software. However, it can only be applied to the scope where model unit testing can be performed.
Automatic code generation handbook	A document defining description methods and other matters of source data (e.g., models) for automatic generated code. For example, documents provided officially or those defined in the developer's organization in accordance with these documents.
Configuration audit	A third person, who differs from personnel implementing configuration management activities, confirms the configuration management activities and evaluates a set of configuration management items and these items individually.
Model	A theoretical description of systems/software, representing target systems/software by using a particular description method with one viewpoint and one abstraction level.
Model based development	The development and verification, with model based technology, of the whole or a part of development (or primary life cycle) processes to which the software development standard applies. Developed models form a part of products.
Model based simulation	It evaluates the validity of requirements and design by using simulation with movable models and other similar ones in the computer architectural design, software requirements analysis, and software design processes when model based development is implemented.

Term	Definition
Modeling handbook	A document defining description methods and other matters of models. For example, documents provided officially or those defined in the developer's organization in accordance with these documents.
Model testing coverage	The degree of coverage for a model when automatic code generation is implemented and model unit testing is performed. Criteria required for quality evaluation are required to be set in advance. A part of the evaluation criteria may be fulfilled by implementing model based simulation.
Model unit testing	When automatic code generation is implemented, a test equivalent to unit testing is performed by using source data (e.g., models) for automatic code generation in the software design process or other processes. The test specifications (test cases) used in this testing are applied to the equivalence evaluation of unit testing in the software coding and testing process with the same specifications.
Operator	An individual or organization operating system.
Release	The distribution of the official version, delivery to system, and delivery to testing.
Service	It is provided as a function accompanying a product.
SOOH	Spacecraft Orbital Operations Handbook: SOOH SOOH provides the necessary information for the operation of the spacecraft. It is the source to develop SOP, to set the database for tracking and control, and during actual operations, it may be referred to confirm operational procedures, planning countermeasures for anomalies, and planning and setting new operational plans and procedures.
SOP	Spacecraft Operations Procedure: SOP Usually it is prepared for each independent functional unit, to properly utilize the functions of the tracking and control system enough, and in the operation phase, it is used by a single unit or by combined multiple units.
Source code	The source of the generation of software (so-called object code) in the software integration process. It is typically described as code suitable for reading and writing by humans.
Testability	The capability of establishing test criteria for requirements specifications and implementing tests to determine whether these criteria are met.
User	An individual or organization obtaining benefit from system.
Verifiability	The capability of establishing verification criteria for requirements specifications and implementing some items of verification (such as tests and analyses) to determine whether these criteria are met.

3.3 Terms defined in JERG-0-049B

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 compliance 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.
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 weaknesses of a subject process in accordance with objective metrics 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. The definition of what is contained in a computer system may be determined by the format which is defined at the boundary where it is not included in a computer system, or by the format which is defined as items included inside the boundary. In addition, even if the format is defined within the boundary, it is assumed that the definition of items inside the boundary shall be included as part of the computer system design. As a computer system may be defined in various ways, from a one-chip microcomputer to multiple general-purpose computers connected to a network.

Term	Definition
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.
COTS	COTS is the abbreviation of Commercial Off-The-Shelf. It has already been developed and is available in the commercial marketplace.
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 When a target system is onboard software, for example, an enabling system means software development environment, an emulator, a simulator, test equipment, or a tool generating a telemetry and command DB or other similar ones . 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.
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.
Interoperability	A series of exchanges among multiple functions to accomplish a specified objective.
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.
Non-functional requirements	Non-functional requirements are all requirements except functional requirements, such as performance, safety, and reliability.
Output	Output is information that is transformed from input by performing activities.

Term	Definition
Process	Process is a set of interrelated or interacting activities to transform input to output.
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]
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.

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 specifications. 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 integration	Integrate the software components step by step and construct the software that operates in the target platform
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 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.
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.
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
The stability (maturity) of software requirements specifications	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 specifications affect process cost, delivery date and quality which let the software requirements specifications 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 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]
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.

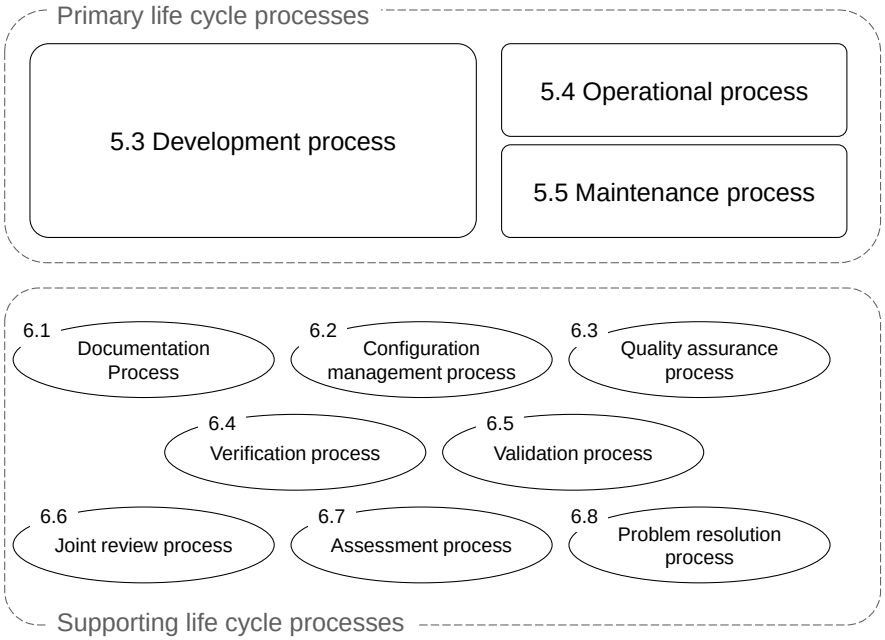

Figure 4.1 - Structure of the standard

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	Operational	Process to be implemented from the viewpoint of operation. Operation in this standard is used for spacecraft operation support by spacecraft operator and user,
	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.

5 Primary life cycle processes

This clause defines the following primary life cycle processes:

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

5.1 Not used

5.2 Not used

5.3 Development process

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 (*1);
(11) software acceptance process (*2).

These activities may be implemented in a different order from what is described in this document. However, the overall configuration of the development process including order of implementation of each activity and relationship between 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.

*1: In this standard, supply and introduction of the software product process is not applied.
*2 In this standard, the software acceptance process is not applied.

5.3.1 Process implementation

5.3.1.1 Activity

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

(1) A development strategy shall be defined. The following shall be considered in the strategy:
   (a) Management of knowledge assets
      (i) Obtainment and maintenance plans in the valid period of knowledge assets
      (ii) Characterization of types such as knowledge and ability collected and maintained as knowledge assets
      (iii) Criteria for accepting, qualifying, and retiring knowledge assets
      (iv) Procedures for controlling change in knowledge assets
      (v) Plans, mechanisms, and procedures for protecting, controlling, and accessing classified or sensitive data and information
      (vi) Mechanisms for storage and retrieval

(2) Based on the development strategy, a 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 (*1)
   (c) Identification of target software
   (d) Definition of the identification of software development processes and their relationship
   (e) Definition of software development processes and activities (*2)
   (f) Definition of activities in accordance with automatic code generation tools, scope (*3), verification policy (*4), management policy, the tools to be used and selection criteria and rationale applied to the selection of the tools, and the automatic code generation handbook when automatic code generation is implemented
   (g) Following development policies and rules relating to the software coding and testing
      (i) Suitable safety, security, privacy, and policy for environment activity
      (ii) Programming and coding standard (including definitive implementation guidelines for error handling)
      (iii) Unit testing policy
      (iv) Performing of software integration, peer review, and walkthrough review
      (v) In case change management is conducted by not using any tool, use of configuration management during software coding and testing
   (h) Activities including roles, authorities, and responsibilities in each individual development process and their implementation management plan
   (i) Review plan
   (j) Preparation of development related documents' structure, and relationships of input and output in each development process
   (k) Documentation plan, including development department and schedule
   (l) Appropriate work allocations, and methods for progress management including work plans and criteria for achievement for each work. They shall be applied from initial development schedule

(m) Environment to be used for the software development and verification (enabling systems or services such as simulators, real hardware, test environment, source code analysis tools, and automatic code generation tools) shall be identified and made available

(n) Management plan for software products other than those newly developed in the developer's organization (refer to Appendix 1). The following shall be included:

(i) Identification of software products other than those newly developed in the developer's organization

(ii) Definition of the quality assurance process regarding identified software products other than those newly developed in the developer's organization newly developed in the developer's organization

(iii) (iii) Conformation method of applicability to the computer system and safety of the acquisition route

(o) Evaluation plan of applicability to the computer system

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

*1: Each subsystem, component, mission sensor system, and so on of spacecraft may be considered.

*2: It is desirable to take account of the software life cycle. Activities related to computer system integrated software may be included, as necessary.

*3: Functions and modules to which automatic code generation is applied.

*4: Verification policy defines source code review, verification methods (model based simulation and model unit testing) of source data (e.g., models) for automatic code generation, unit testing, integration tests, equivalence evaluation of models and code, and comprehensive verification strategy combining them.

5.3.1.2 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.

(3) For model based development, static (functions and structure) and dynamic (behavior and performance) traceability for the description of specifications, including models, and code shall be evaluated as traceability evaluation in model based development for each process.

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 in accordance with the software development plan:

(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) States (including operation mode) required for the developed computer system and the transition of these states shall be identified.
  (d) If another state transition is defined separately such as the spacecraft system mode, the relationships between states required for the developed computer system, the transition of these states, and the other state transition defined separately shall be clarified.
(2) Requirements specification development
  (a) Feasibility and consistency shall be confirmed, based on the operational scenarios, and the requirements specifications for the computer system shall be defined.
  (b) Specifications for data and databases to be handled by the computer system shall be identified in the requirements specifications.
  (c) Risks, computer system severity, and requirements for important quality characteristics shall be identified.
  (d) Interface requirements shall be analyzed, and interface 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 specifications for the computer system shall be clarified, and the traceability of higher level requirements for the computer system shall be evaluated. In addition, the consistency of the requirements specifications shall be analyzed, documented, and maintained.
  (f) Rationale and its consideration processes of the individual requirement of the requirements specifications shall be clarified, and their feasibility shall be evaluated.
  (g) When software products other than those newly developed in the developer's organization are 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 the 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 specifications for the computer system
(3) Interface specifications
(4) Evaluation results of the traceability of the requirements specifications and requirements for the computer system
(5) Requirements specification rationale and their feasibility evaluation results for the computer system
(6) Applicability evaluation results of software products other than those newly developed in the developer's organization
(7) Evaluation results of the verifiability of the requirements specifications 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 in accordance with the software development plan:

(1) Computer system architecture shall be designed based on the requirements specifications 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 specifications for the computer system shall be allocated among the individual configuration items of the computer system.
(3) When model based development is implemented, models shall be developed in accordance with the modeling handbook.
(4) Computer system architectural design specifications shall be documented by combining the results of the above-mentioned design.
(5) Interface requirements for the software shall be extracted.
(6) Feasibility of software items in fulfilling their allocated requirements shall be evaluated.
(7) Rationale for the design and preconditions (e.g., operational assumptions) for the computer system architectural design specifications shall be identified.
(8) Traceability of the computer system architectural design specifications relative to higher level requirements, such as the requirements specifications for the computer system, shall be evaluated.
(9) Evaluation criteria for the computer system architectural design shall be defined, and the results of the computer system architectural design shall be evaluated in accordance with these criteria. Additionally, rationale for the selection of the computer system architectural design shall be recorded. When model based development is implemented, the validity of the design shall be evaluated by applying model based simulation with the level of the computer system architectural design specifications.

5.3.4.2 Input

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

5.3.4.3 Output

(1) Computer system architectural design specifications
(2) Software Requirements
(3) Interface requirements, including requirements for telemetry and command
(4) Evaluation results of the traceability relative to the computer system architectural design specifications and the requirements specifications 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 in accordance with the software development plan:

(1) The computer system architectural design specifications, interface requirements, and software requirements including non-functional requirements shall be analyzed.
(2) The required software states (including operation mode) and the transition of these states shall be identified.
(3) The relationships between the software states, the transition of these states, and the state transition of higher level systems shall be clarified.
(4) Identifiers shall be included in the individual software requirements specifications.
(5) Specifications for data and databases to be handled by the software shall be included in the software requirements specifications.
(6) Specifications for failure detection and handling functions shall be included in the software requirements specifications.
(7) Risks, software severity, and requirements for important quality characteristics shall be identified.
(8) When model based development is implemented, models shall be developed in accordance with the modeling handbook.
(9) Software requirements specifications shall be documented by combining the results of the above-mentioned design.
(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 specifications relative to the computer system architectural design specifications and interface requirements shall be analyzed, documented, and maintained.
(12) Rationale of the individual requirements of the software requirements specifications shall be clarified, and their feasibility shall be evaluated.
(13) If software products other than those newly developed in the developer's organization

are used, compliance with the software requirements specifications and its applicability with the computer system architectural design specifications shall be analyzed.

(14) When automatic code generation is implemented, the interface between manually-developed source code and automatically-generated code shall be analyzed in accordance with the scope decided in the establishment of the plan.

(15) Operational assumptions and constraints regarding software requirements specifications shall be extracted.

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

(17) Problems, inadequacies, and inconsistencies included in the software requirement specifications shall be identified and planned to resolve them.

(18) 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.

(19) When model based development is implemented, the validity of the requirements shall be evaluated by applying model based simulation with the level of the software requirements specifications.

(20) Software verification coverage pertaining to software function, performance, and operational scenarios in the verification plan shall be evaluated, and testability regarding the software requirements specifications and interface specifications shall be evaluated.

(21) 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 the software requirements specifications, the following measurement shall be performed for the software requirements analysis:

(1) The definition of the data to be collected for evaluating the stability (maturity) of the software requirements specifications and their evaluation methods 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.

5.3.5.3 Input

(1) Computer system architectural design specifications

(2) Software requirements

(3) Interface requirements

(4) Operational scenarios

(5) Applicability evaluation results of software products other than those newly developed in the developer's organization

5.3.5.4 Output

(1) Software requirements specifications
(2) Interface specifications
(3) Software requirements specification traceability and consistency evaluation results
(4) Software requirements specification rationale and their feasibility evaluation results
(5) Applicability evaluation results of software products other than those newly developed in the developer's organization (updated)
(6) Operational assumptions and constraints
(7) Software verification plan, including validation plan
(8) Verification completeness and testability for the software verification plan evaluation results
(9) Software requirements stability (maturity) evaluation 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 an actual development, the software design may be divided into architectural design and detailed design to perform them.

5.3.6.1 Activity

The following activities shall be performed for the software design in accordance with the software development plan:

Software architectural design

(1) Functional decomposition and module partitioning shall be performed based on the software requirements specifications and module structures and the structures between the modules comprising the functions shall be clarified, so that an appropriate software architectural design is performed. When automatic code generation is implemented, the scope of automatic code generation decided in the establishment of the plan shall be established, and interface specifications between manually-developed source code and automatically-generated code shall be established.
(2) The design guidelines (software architecture requirement, etc.) and the design characteristics to be considered shall be selected, and the design shall be performed by considering priorities.
(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 shall be defined as the software requirements specifications.
(4) Interface specifications shall be detailed in accordance with consideration for the boundaries of external interfaces and interoperability, 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 considerations of the module 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 documented based on the result of the software design.

(8) When model based development is implemented, the validity of the design shall be evaluated by applying model based simulation.

(9) The needed design methods or organizationally maintained past knowledge shall be identified, prepared and acquired.

(10) When model based development is implemented, models shall be developed in accordance with the modeling handbook.

(11) When automatic code generation is implemented, source data (e.g., models) for automatic code generation shall be developed in accordance with the automatic code generation handbook.

(12) Traceability and consistency of the software design with the software requirements specifications, interface specifications, and with the necessary related documents shall be analyzed, documented and maintained.

(13) As necessary, with regard to the individual software design, the design rationale shall be clarified and its feasibility and testability (including test case) evaluated.

(14) If software products other than those newly developed in the developer's organization are used, its applicability with the software design shall be analyzed.

(15) Operational assumptions and constraints regarding the software design shall be identified.

(16) When automatic code generation is implemented, model unit testing shall be performed so that the predetermined evaluation criteria for model testing coverage are met.

(17) Software test plans and specifications shall be established in accordance with the software verification plan.

For software test specifications, the following shall be considered:

(a) Operational scenarios
(b) Interface specifications
(c) Maximum loads for assumed scenarios
(d) Coverage for software requirements specifications and software design specifications
(e) Anomalies, such as exceptions and failures
(f) Applicability of software products other than those newly developed in the developer's organization to computer system

(18) 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 implement the progress management and risk evaluation 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 specifications
(2) Interface specifications
(3) Operational assumptions and constraints
(4) Applicability evaluation results of software products other than those newly developed in the developer's organization
(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) Applicability evaluation results of software products other than those newly developed in the developer's organization (updated)
(6) Operational assumptions and constraints (updated)
(7) Software test plan
(8) Software test specifications
(9) Evaluation results of collected data regarding progress management and risk evaluation

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 plan 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 in accordance with the software development plan:

(1) Preparation
  (a) For unit testing, criteria shall be established to determine pass or failure per software design specifications.
  (b) For unit testing, criteria for the test coverage for source code shall be established. The branch of source code shall be well covered at least.

(2) Coding and unit testing
  (a) Source code shall be developed based on constrains, the software design specifications and interface specifications.
  (b) When automatic code generation is implemented, the automatic code generation shall be performed in accordance with the automatic code generation handbook.
  (c) Source code shall be developed based on the defined coding standard. When automatic code generation is implemented, it shall be developed in accordance with the automatic code generation handbook.
  (d) The review of source code shall be performed according to the verification policy of the development plan.
  (e) Unit testing specifications shall be developed in accordance with the software verification plan, the software test plan and the acceptance criteria defined in (1) (a) above. When automatic code generation is implemented, unit testing specifications including the test specifications (test cases) used for the model unit testing shall be developed.
  (f) 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.
  (g) For unit testing, the test shall be performed so that the criteria of the test coverage for source code are satisfied. When automatic code generation is implemented, in addition, model unit testing is performed in the software design process (refer to 5.3.6.1 (16)). Then, unit testing for automatically-generated code is performed in accordance with the test specifications (test cases) used for the model unit testing, and the equivalence between source data (e.g., models) for automatic code generation and the automatically-generated code shall be evaluated.
  (h) Static analysis shall be performed with a code checking tool or equivalent and the source code quality shall be evaluated. (Automatically-generated code is included.)
  (i) 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 code and the design specifications shall be checked.
  (j) The operational assumptions and constraints identified during software coding and testing shall be identified.

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) Operational assumptions and constraints
(4) Software verification plan

5.3.8.4 Output

(1) Source code
(2) Operational assumptions and constraints (updated)
(3) Unit testing specifications
(4) Unit testing record (When automatic code generation is implemented, the result of equivalence evaluation is included.)
(5) Traceability analysis record
(6) Source code quality evaluation 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

For software integration, activities that meet the following items shall be conducted in accordance with the software development plan:

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

architecture, and design shall be identified.

(2) Implementation of Integration

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

(b) When automatic code generation is implemented, software generated from manually-developed source code and one generated from automatically-generated code shall be integrated. Then, the applicability of the integrated one with the interface specifications clarified in the design shall be checked.

5.3.10.2 Measurement

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

(1) For problems found 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) Operational assumptions and constraints

(3) Software design specifications

5.3.10.4 Output

(1) Source code (integrated)

(2) Software (integrated)

(3) Evaluation results of collected data regarding software quality defined and evaluated in the software integration

5.3.11 Software integration test

5.3.11.1 Activity

Activities that meet the following shall be performed for the software integration test in accordance with the software development plan:

(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 loads for assumed scenarios

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

(v) Anomalies, such as exceptions and failures

(vi) Applicability of software products other than those newly developed in the

developer's organization to the computer system

(vii) Equivalence between source data (e.g., models) for automatic code generation and automatically-generated code when automatic code generation is implemented

(viii) When automatic code generation is implemented, software generated from manually-developed source code and that generated from automatically-generated code shall be integrated. Then, the validity of correct software behavior in an environment equivalent to the real target shall be checked.

(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) Criteria to determine pass or failure shall be established.

(e) In cases where any operational assumptions and constraints are found to be necessary during the preparation of the software integration test procedure, the operational assumptions and constraints shall be updated.

(f) The test specifications and procedures shall be checked in the viewpoint of whether the test has been prepared.

(2) Implementation of tests

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

(b) During the software integration test, a joint review of the test results shall be performed as necessary, and a decision shall be made on 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 various tests performed after software coding and testing shall be evaluated, and tests performed after software coding and testing 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) Problems found during software integration tests shall be recorded, together with related information such as test cases.

(2) Quality index 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 specifications
(3) Software design specifications
(4) Software verification plan
(5) Software test plan
(6) Software test specifications
(7) Operational assumptions and constraints
(8) Source code (integrated)
(9) Software (integrated)
(10) Operational scenarios
(11) Applicability evaluation results of software products other than those newly developed in the developer's organization

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) Evaluation results of quality metrics data that were set, collected, and evaluated in the software integration test

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 a 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 (embedding) in accordance with the software development plan:

(1) Installation Preparation

(a) Software shall be prepared from the repository of the software development environment and 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.
Installation into a target platform means embedding software into the targeted computer, including writing into ROM.

(b) Configuration management information of software to be released shall be prepared, to include the installation procedure into the target platform, the installation result check procedure, and the operational assumptions and constraints.

(c) Status of installation preparation (advanced verification, procedures, and other matters of installation) shall be checked.

(2) Implementation of installation

(a) 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.

(b) A check (*) that the software has been properly installed shall be performed, in accordance with the installation results check procedure.

*: For example, the original data after installation and the data on the target platform shall be compared by a file dump, and shall be confirmed by a comparison of their consistency.

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 procedure
(4) Software installed computer system
(5) Installation results check procedure
(6) Installation confirmation result
(7) Operational assumption and constraints (updated)

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 in accordance with the software development plan:

(1) Test preparation

(a) Criteria for the computer system integration and verification points for interfaces and software system functions after the integration shall be identified and defined.

(b) Computer system integration shall be performed with the identification of the integration constraints included in the system/software requirements, architecture and designs.

(c) Computer system integration test specifications shall be documented in accordance with a software verification plan. With regard to the computer system

integration test specifications, the following viewpoints shall be considered:

(i) Operational scenarios
(ii) Maximum load for assumed scenarios
(iii) Coverage regarding requirements specifications for the computer system, computer system architectural design, and software requirements specifications
(iv) Anomalies, such as exceptions and failures
(v) Applicability of software products other than those newly developed in the developer's organization to the computer system

(d) Computer system integration test procedure shall be developed in accordance with a software verification plan and computer system integration test specifications.
(e) Problems found during test preparation (test procedure checks, and so on) shall be recorded and managed.
(f) In cases where operational assumptions and constraints are found during the preparation of computer system integration test specifications and a computer system integration test procedure, the operational assumptions and constraints shall be updated.

(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 computer system integration tests, the effectiveness of activities such as the performed tests, verification, validation, joint reviews shall be evaluated, and performed again as necessary.
(f) During computer system integration test, a joint review of the test results shall be performed as necessary, and decision shall be made as to whether the test should be continued or not.

(3) Regarding software to be released, the configuration management information operational assumptions and constraints shall be prepared.

5.3.13.2 Measurement

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

(1) Collection of quality metrics data
(a) Problems 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 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) Operational scenarios
(2) Requirements specifications for computer system
(3) Computer system architectural design specifications
(4) Software requirements specifications
(5) Software verification plan
(6) Operational assumptions and constraints
(7) Software installed computer system
(8) Configuration management information
(9) Applicability evaluation results of software products other than those newly developed in the developer's organization

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) Software products
(6) Configuration management information (updated)
(7) Evaluation results of quality metrics data that were defined, collected, and evaluated in the computer system integration and computer system integration test

5.3.14 Supply and introduction of software product

Not applicable.

5.3.15 Software acceptance

Not applicable.

5.4 Operational process

In this process, the activity to perform operational support for the operator and user, before

and after the launch of spacecraft, is defined regarding software integrated into the spacecraft.

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

5.4.1 Process implementation

5.4.1.1 Definition of an operational strategy

(1) An operational strategy shall be defined. In principle, the following shall be considered in the strategy.
    (a) Criteria expected from the start to end of operation (capacity, occupancy rate, response, safety, etc.)
    (b) Software or computer system release criteria and schedule considering the correction to maintain the current service
    (c) Approaches to the achievement of operation modes (including preparation and checks for procedures for regular operation and emergency operation)

5.4.1.2 Establishment of an operational plan

A plan (operational plan) to support operators and users shall be established based on the operational strategy in order to perform the operational process. Planning shall be considered on the premise of the following.

(1) Enabling systems or services needed to support the operation shall be identified and made available.

5.4.1.3 Establishment of a procedure regarding problem report and change requests

A procedure regarding problem report and change requests of software under spacecraft operation shall be established in accordance with the problem resolution process (refer to 6.8).

5.4.1.4 Establishment of procedure regarding software reprogramming

The procedure regarding software reprogramming for spacecraft in operation shall be established.

5.4.2 Operational testing

Not applicable.

5.4.3 Operation of computer system including software

Not applicable.

5.4.4 Management of operational results

Not applicable.

5.4.5 Operator and user support

5.4.5.1 Support before the start of a spacecraft operation

(1) The following contents shall be provided to the persons in charge of a computer system or subsystem, and shall be handled as the source data for SOOH and SOP:
   (a) Command information that software receives, such as a command list, a command format, the data pattern of each command (fixed and variable value), and command transmission procedures that are necessary for operation
   (b) Telemetry information that software outputs, such as a telemetry list, telemetry format, and frequency of occurrence of telemetry
   (c) Interrelationships between transmission commands and related telemetry
   (d) Interface between the on-orbit reprogramming files and the ground software, necessary for reprogramming and comparison of spacecraft software
   (e) Memory map and procedure necessary for reprogramming and comparison of software
   (f) Operational constraints identified form the software design to the testing
   (g) Operational constraints for changes to requirements, architecture, design, implementation and installation of software or computer system

5.4.5.2 Support after start of spacecraft operation

(1) Support for the requirements (including incidents and problems) by operators and users shall be performed. These requirements and the response status shall be recorded and followed.
(2) Problems shall be resolved in accordance with the procedures established in 5.4.1.3, and the requirements by operators and users shall be delivered to the maintenance process (refer to 5.5) as necessary. These requirements shall be clarified and planned, and the action to be performed shall be reported to the requester. Every solution shall be monitored until it reaches its conclusion.
(3) Before the final solution of the reported problem, if any workaround may exist, the option of whether the workaround is selected or not shall be provided to the originator of the problem report.
(4) The permanent modification and functional addition for software operating on-orbit shall be performed by software reprogramming according to the maintenance process (refer to 5.5).

5.5 Maintenance process

This process is performed in cases where the software and related documents must be modified for the purpose of correction and improvement, or to apply partial modifications called patches, of onboard software or additional modification of software for the reason of functional addition, extension, and so on. This process is performed for onboard software as necessary, after completion of software tests or spacecraft system tests, and for the delivery of the modified software to the spacecraft system to end spacecraft operations. Software reprogramming

during spacecraft operation is also included in this process.

This process terminates at the end of the spacecraft operation, that is, through retirement of the computer system onboard the spacecraft.

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 maintenance results
(7) Migration
(8) Software retirement

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 change in accordance with system availability, safety, and other matters on the basis of mission requirements
  (b) The access to the correct versions and information of software elements required to perform maintenance (e.g., software reprogramming for scheduled phases/problem modification) shall be acquired. In addition, priorities and resources for this shall be established
  (c) Agreed rights to data and the impact on data in the software or computer system during problem resolution and maintenance activity
  (d) Approach to assure that counterfeit or unauthorized software elements are not introduced into the software or computer system
(2) Constraints from maintenance to be incorporated in the software or computer system requirements, architecture, or design shall be defined.
(3) Activities shall be compared and considered so that maintenance is affordable, operable, and sustainable.
(4) Enabling systems or services required to support maintenance shall be identified and made available.

5.5.1.2 Establishment of a maintenance plan

(1) The maintenance plan for implementation of the maintenance process shall be documented in accordance with the maintenance strategy. The following shall be included in the plan:
  (a) Maintenance management method for developed software products
  (b) The structure and the related documentation
  (c) Data, including the management information necessary for maintenance
  (d) Record
  (e) Maintenance environment (environment of development phase, and so on)

(f) Actions for maintaining other related processes appropriately (problem resolution process (refer to 6.8), configuration management process (refer to 6.2), and so on)

(2) Quality assurance shall be planned according to the quality assurance process (refer to 6.3).

5.5.2 Problem identification and modification analysis

(1) Problem identification and analysis of the details of necessary modification shall be performed according to the plan.

(2) If software modification and reprogramming are performed, prior agreement for the modification plan and software reprogramming procedure shall be obtained.

5.5.3 Modification implementation

If a modification is needed, it shall be implemented in accordance with the agreed modification implementation plan. In addition, this shall include procedures for the development processes performed again from the highest level process influenced by the modification. (Refer to Appendix 2.)

If an automatically-generated code part is modified manually, the same activities as for manually-developed source code shall be performed.

5.5.4 Software reprogramming

5.5.4.1 Software reprogramming and verification procedure

Reprogramming from the currently operating software on spacecraft to the modified software or previously installed software shall be performed according to the following. Note that the software shall include parameters.

(1) The procedure for software reprogramming shall be modified, as necessary.

(2) The software version after software reprogramming shall be defined.

(3) The version of software and its parameter setting values before reprogramming shall be defined and recorded.

(4) The procedure used for software reprogramming and verification shall be documented and software reprogramming and verification shall be executed. The recovery procedure to be applied in case the software reprogramming was not performed successfully shall be included in the procedure.

(5) Completion of software reprogramming shall be notified to all concerned parties.

5.5.4.2 Implementation of quality assurance

The quality assurance activity planned in 5.5.1.2(2) shall be performed.

5.5.5 Logistics support implementation

Not applicable.

5.5.6 Management of maintenance results

This activity consists of the following tasks:

(1) Record incidents and problems, including their resolutions, and significant maintenance results. Furthermore, incidents and problems that were addressed in support after the start of spacecraft operation are covered by 5.4.5.2 (1).

5.5.7 Migration

Not applicable.

5.5.8 Software retirement

Not applicable.

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

The documentation process is to define the management plan of the documents necessary for all persons concerned who develop and utilize software or computer system, and to perform document production and distribution and so on according to the management plan.

This process consists of the following activities:

(1) Process implementation
(2) Development
(3) Document production
(4) Maintenance/revision/disposal of documents

6.1.1 Process implementation

A management plan for developed and received documents in all processes required by this standard shall be established. In addition, a plan shall be established so that relevant necessary documents can be maintained, revised, and disposed of after the completion of the development of software products.

The following information shall be included in the document management plan:

(1) Type of document
(2) Purpose of the document
(3) Plan for issue date and acquisition date of the document
(4) Relevant departments
(5) Procedures and responsibilities regarding the input information including received documents, documentation (development, inspection, and approval), issue (issue, distribution, and storage), and revision (revision and disposal)

(6) The form (content, format, and so on) appropriate for such documents shall be decided.
(7) Actions for anonymity requirements for development, inspection, issue, revision, and disposal shall be defined.

6.1.2 Development

(1) Each document shall be developed according to the separately defined form, style, and technical contents.
(2) The appropriateness of information sources for the developed documents shall be confirmed.
(3) The integrity of the information included in the developed document shall be checked.
(4) The document shall be confirmed and corrected in accordance with the standard format.
(5) The developed document shall be checked for its compliance with the actions for anonymity requirements defined in the document management plan.
(6) If the developed document has a disposal deadline, the appropriateness of this deadline shall be checked.
(7) The document shall be checked and approved for its appropriateness by authorized personnel prior to production.

6.1.3 Document production

(1) The document shall be issued to the suitable parties for distribution according to the management plan.
(2) The produced document shall be stored according to the related standard.
(3) At the production, it shall be considered so that the latest edition is surely utilized as necessary.

6.1.4 Maintenance/revision/disposal of documents

Tasks for documents such as maintenance, revision, and disposal shall be performed in accordance with the document management plan. In addition, the following shall be considered.

(1) The integrity of the information included in the revised document shall be checked.
(2) The revised document shall be checked for its compliance with the actions for anonymity requirements defined in the document management plan.
(3) When the document reaches its disposal deadline, some courses of action such as the extension of the disposal deadline shall be performed in accordance with the document management plan.
(4) The revised documents shall be checked and approved regarding the form and technical content.
(5) Needless, ineffective or valid-less documents shall be disposed.

6.2 Configuration management process

Configuration management process is a process that applies to the following administrative and technical procedures through the software life cycle.

This process consists of the following activities:

(1) Process implementation
(2) Configuration identification
(3) Configuration change control

(4) Configuration management accounting and status report
(5) Configuration evaluation
(6) Release management

6.2.1 Process implementation

6.2.1.1 Definition of a configuration management strategy

A configuration management strategy shall be defined. The following shall be considered in the strategy:

(1) Control of software licensing, 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 necessary stakeholders
(5) Consideration of risks and influence levels in the approval of the baseline and change requests

6.2.1.2 Establishment of a configuration management plan

A configuration management plan shall be developed according to the configuration management strategy. The following information shall be included in the plan:

(1) Configuration management activities
(2) Procedures for performing activities (e.g., applicable criteria and tools) and schedules (milestones for configuration management, such as time to start and period of configuration management activities which are applied to management system)
    (a) Control of access, change, and action to configuration items
    (b) Criteria required to identify configuration management items and criteria or implementation times for establishing and maintaining baseline
(3) Organizations responsible for performing activities, and roles/mutual relationships/cooperative relationships of departments in them
    (a) Roles and responsibilities of the configuration management activities shall be included.
(4) Other concerned organizations, and their roles/mutual relationships/cooperative relationships
    (a) Coordination of the configuration management across the set of acquirer, supplier, and supply chain organizations for the life cycle of the software, or the extent of the agreement or project, as appropriate
(5) Store, archive and access procedure for configuration items, configuration management work products and records

6.2.2 Configuration identification

(1) Identification of configuration management items
    Configuration management items shall be identified from the products generated through the software development during the project.
(2) For configuration items, the following shall be identified:
    (a) Version references (*1)
    (b) Other identification details (*2)

(3) Establishment of baseline
The configuration baseline shall be established in accordance with its purpose (*3). The configuration items and versions included in the baseline shall be clarified.

(4) Version control system
Units for version control, a version control system, and a procedure shall be established. By using the system and procedure, the history of configuration management activity shall be recorded and shall be traceable.

(5) Obtain stakeholders agreement to the established configuration baselines.

*1: When model based development and automatic code generation are implemented, the version of each of the elements such as models to be used, source data (e.g., models) for automatic code generation, tools, and simulators shall be included.

*2: The configuration management information of the following shall be able to be referred to: models, tools, simulators, parameters, and other matters used in model based simulation and model unit testing, and source data (e.g., models) for automatic code generation and tools for automatically-generated code.

*3: The following may be items for this purpose:
i. Identification of the configuration for design review baseline
ii. Identification of the configuration after software integration tests
iii. Identification of the configuration at the end of checkout at launch

6.2.3 Configuration control

6.2.3.1 Management for configuration management items

For the configuration management items, the history of the revision shall be recorded and controlled, based on the control system established in 6.2.2(4).

6.2.3.2 Management for a set of configuration items

A set of configuration items (*) shall be confirmed so that configuration items are included neither too much nor too little and are the appropriate version.

The contents shall be recorded and the history shall be managed.

*: Combination of items selected according to their purpose from configuration management items. This sub-clause shall be applied to the baseline control.

6.2.3.3 Change request control

(1) Identification and recording of change requests (including waiver)
Change requests for configuration management items shall be identified. And the relationship to other change requests shall be evaluated if other change requests exist. Change requests, rationale of change, analysis results, modification status, and the status of change request shall be recorded.

(2) Analysis and evaluation of the changes
For the change requests, technical influence, cost, and merit and demerit of the change shall be analyzed and evaluated.
The verification and validation activities performed at the change shall be analyzed.

(3) Approval or disapproval of change requests
Approval or disapproval of change requests shall be determined by review.
The result of the review shall be recorded so that the reasons for the change and its

approval or disapproval are traceable.

(4) Confirmation of modification and verification of software items

The completion of modification and verification of software items shall be confirmed based on the approved change request.

6.2.4 Configuration management accounting and status report

Configuration management accounting and status reports that show the status and history of configuration management items shall be prepared and the status shall be reported.

6.2.5 Configuration evaluation

The baseline of a set of configuration items shall be evaluated whether configuration items are functionally and physically neither too much nor too little and have appropriate versions. Confirmed results and required actions shall be recorded.

6.2.6 Release management

6.2.6.1 Implementation of release management

The release of software products and computer system shall be managed in accordance with the decided procedures.

And, the management for a set of configuration items, applied with the sub-clause 6.2.3.2, shall be performed for each release.

The original and the master copy of the configuration management items shall be maintained through the life cycle of software products or computer system.

And, the source code and documents shall be handled and stored according to the policy of the organization (*) and the policy defined in accordance with the type of information included in the source code and documents.

*: In “the policy of the organization” mentioned here, company regulations and project rules are included.

6.3 Quality assurance process

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 spacecraft life cycle, adhere to their established plans which are based on this standard or by tailoring its results.

This process consists of the following activities:

(1) Process implementation
(2) Implementation of quality assurance activities
(3) Products quality assurance
(4) Process assurance
(5) Assurance of quality system
(6) Management of the quality assurance record

6.3.1 Process implementation

6.3.1.1 Appointment of the responsible person for quality assurance

The responsible person who is invested with the responsibilities and authorities independent of the development organization shall be appointed. The responsible person shall be responsible for the management of quality assurance activities and the activities for the check

of status and improvement.

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) Process and product improvement activities

6.3.1.3 Establishment of a quality assurance activities plan

A quality assurance activity plan shall be established in accordance with 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) Activities for quality assurance implementation
(4) Schedules
(5) Procedures for review
(6) Procedures for the work such as identification, collection, filing, maintenance, disposition and disposal of quality assurance activity records
(7) Education and training regarding quality assurance
(8) Requirements and activities regarding quality assurance for purchase management and suppliers, and quality assurance activities for suppliers (including subcontractors)
(9) Management in the reuse of software products
(10) Management in automatic code generation
(11) Management in model based development
(12) Procedures for release
(13) Procedures for process and product improvement activities in problem identifications and their resolutions
(14) Relationships to 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), and selected activities and tasks.
(15) Quality evaluation criteria and methods for processes and products including criteria for product acceptance
(16) Activities of verification, validation, monitoring, measurement, review, inspection, assessment, and tests specific to and required for products

6.3.1.4 Agreement on quality assurance activities plan

The quality assurance activities plan shall be agreed with the acquirer.

6.3.2 Implementation of quality assurance activities

Quality assurance activity shall be performed in accordance with the quality assurance activity plan.

6.3.3 Products quality assurance

(1) Software, or computer system, and related documentation shall be developed without insufficiency with the plans.

(2) Products shall fulfill their approved requirements specifications through the verification and validation of the software or computer system development processes for each of the products.
(3) Products quality shall be assured by implementing independent verification and validation (IV&V) as necessary.

6.3.4 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.

6.3.5 Assurance of quality system

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

6.3.5.1 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.

6.3.5.2 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.

6.3.5.3 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.

6.3.5.4 Management in the reuse of software products

With regard to software products such as reusable COTS items, software, source code, design elements included in relevant documents, and test procedures (refer to Appendix 1), the following shall be included in the management items.

(1) Any benefits gained by reusing software products
(2) Evaluation items and levels determining whether software products can be reused or not
  (a) Applicability of reused software products with regard to software to be developed
  (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

(3) Consideration points and items in managing the use of reused software products
   (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

6.3.5.5 Handling, storing, and labeling

Not applicable.

6.3.5.6 Continuous quality improvement activities

To perform continuous quality improvement for organizations and projects, information required for this is collected and fed back to succeeding development.

6.3.6 Management of the quality assurance record

(1) Create records related to quality assurance activities
(2) Maintain and store the records
(3) Records related to quality assurance activities shall be available for perusal by related parties based on any agreements.

6.3.7 Management in automatic code generation

When automatic code generation is implemented, situations of the implementation of the following management items shall be evaluated:

(1) Selection of tools for automatic code generation

Criteria are defined in terms of the evaluation viewpoints below, and the results of the evaluation of these criteria are documented when the tools required for automatic code generation, such as those for developing source data (e.g., models) for automatic code generation; generating code automatically from source data (e.g., models) for automatic code generation; collecting the metrics of source data (e.g., models) for automatic code generation, and managing the configuration of source data (e.g., models) for automatic code generation and parameters used for these source data (e.g., models); and other functions. The results can be replaced with the following perspectives based on the evaluation of achievements of other products having equal quality requirements.

   (a) Compliance with the identified automatic code generation handbook
   (b) Measurement environment of model testing coverage in model unit testing
   (c) Compatibility in cooperation with tools used in model based simulation and model unit testing
   (d) Compatibility with other tools (e.g., compilers and code management systems) using automatically-generated code as one of inputs
   (e) Whether tools required for the project can be customized or not
   (f) Configuration change control of tools including parameters
   (g) Whether the quality assurance information of tools (e.g., versions and upgrading

information) is available or not
(h) Performance of automatically-generated code (size and speed)

(2) Modification of automatically-generated code
(a) Automatically-generated code shall not be modified in the code itself but modified through the source data (e.g., models) for automatic code generation.
(b) If automatically-generated code is modified manually from necessity, activities of the development and configuration management processes, such as compliance with coding standards, the review of code, and unit testing, equivalent to those for manually-developed source code shall be performed.

6.4 Verification process

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 way such as review, test, and so on as the methods for verification.

(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) Identification of the verification scope (including the target software, elements 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 critical quality characteristics, integration, and the correctness of the documents.
(b) Identification of the constraints that potentially limit the feasibility of verifications.
(c) Identification of 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 included according to their degree of importance.
(b) Classifications of verification (e.g., inspection, analysis, demonstration, and tests), applied methods and techniques, and standards for 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) Enabling systems and services required to support the verification shall be identified and made available.

(e) Actions following the problem resolution process (refer to 6.8) for incidents and problems identified by verification (e.g., tests/review/analysis) in accordance with the verification plan.

6.4.2 Verification

This activity consists of the following tasks. In case of execution, the verification shall be performed considering the viewpoints (6.4.2.1 through 6.4.2.6) described for each task in accordance with the verification plan:

(1) Process verification
(2) Requirements verification
(3) Design verification
(4) Source code verification
(5) Integration verification
(6) Documentation verification

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 criteria 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 viewpoints shall be considered:

(1) Requirements for the computer system are properly reflected in the computer system requirements specifications. The requirements specifications for the computer system are consistent, feasible, and verifiable.
(2) Requirements specifications for the computer system are properly allocated to hardware, software, and operation.
(3) Software requirements specifications are consistent, feasible, and verifiable. And the requirements specifications for the computer system are properly reflected in the software requirements specifications.
(4) Software requirements specifications satisfy the standards applied to software items.
(5) Concerning to matters 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 viewpoints shall be considered:

(1) The design meets requirements specifications, and has the traceability for requirements specifications.
(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.
(4) Software design satisfies the requirements of the standards applied to software items.
(5) When model based development is implemented, model based simulation is performed.
(6) When automatic code generation is implemented, model unit testing is performed.

6.4.2.4 Source code verification

The following viewpoint shall be considered:

(1) Source code meets design, and has the traceability for 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) Source code shall conform to e.g. coding standards.
(5) When automatic code generation is implemented, equivalence between source data (e.g., models) for automatic code generation and automatically-generated code is evaluated.

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.
(2) The software modules and data are completely and correctly integrated.
(3) The components of the computer system are completely and correctly integrated.
(4) The integration of software and the integration of the computer system have been performed in accordance with plan.

6.4.2.6 Documentation verification

The following shall be considered:

(1) Documentation activity is performed properly in accordance with the documentation process (refer to 6.1).
(2) Documentation activity is performed with adequate timing
(3) Configuration management of the documentation is performed in accordance with the defined procedures.

6.4.3 Management of the verification results

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

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

6.5 Validation process

Validation is a process 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 results

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) System constraints that should be included in requirements for the computer system to be developed shall be identified.

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) 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) Enabling systems or services required to support the validation shall be identified and made available.
   (f) Actions following the problem resolution process (refer to 6.8) for identified incidents and problems

6.5.2 Validation

In accordance with the validation plan, the validation (*) shall be performed considering the following viewpoints:

(1) Software or computer system shall satisfy the assumed method and the use of the software. Software or computer system shall satisfy the requirements specifications.
(2) Regarding the validation for the software or computer system performed as appropriate in the target environment, the difference between the target environment and the simulated environment shall be evaluated, if simulated environment is used.

*: As the validation, the method except test (analysis, modeling and simulation and so on.) may be adopted.

6.5.3 Management of 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 is a process to optionally evaluate the action status and products of software development.

At a joint review, both management (progress status and so on.) and technical levels of software development are reviewed.

This process may be applied to all reviews.

Joint review process consists of the following activities:

(1) Process implementation
(2) Software development management reviews
(3) Technical reviews

6.6.1 Process implementation

When the software development is started, reviews corresponding to the project milestones including necessary resources (budget, personnel, location, facilities and tools, and so on.) shall be planned.

(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 the parties. These resources include personnel, location, facilities, hardware, software, and tools.
(3) Both reviewing party and reviewed party concerned shall agree on the following at each review:
    (a) Matters to be reviewed
    (b) Review scope and subjects
    (c) Review procedure
    (d) Entry and exit criteria for the review
(4) Incidents and problems identified during the reviews shall be recorded, and appropriate action taken through the problem resolution process (refer to 6.8), as necessary.
(5) Both reviewing party and reviewed party shall agree with the related persons regarding following and shall distribute the results:
    (a) Review results
    (b) Assignment of responsibilities for the handling of identified problems
    (c) Review close-out criteria

6.6.2 Software development management reviews

6.6.2.1 Evaluation of software development status

The development status shall be evaluated by comparing the current status to the software development plan and shall be evaluated by following viewpoints:

(1) Development is progressing according to the plan.
(2) Resources necessary for implementing the software development are allocated properly and managed.
(3) The gap with the plan is monitored, and the necessity of software development direction change or implementing an alternate plan is determined.
(4) Any risk that may jeopardize the success of the project is identified, evaluated, and managed.

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 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.

6.6.3.3 Evaluation of development status

Software products or computer system shall be evaluated according to the following viewpoints:

(1) Software products or computer system are complete.
(2) Software products or computer system conform to standards and specifications.
(3) Changes to software products or computer system are implemented properly.
(4) Preparation to switch to the next action (preparation for the next process transition) is ready.
(5) Development, operation or maintenance are performed in accordance with the planning, schedule, standards, and guidelines of the project.

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 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) A third person who is well informed of assessment methods and relevant standards and differs from the developer itself and director of the software or computer systems shall be selected as assessors.
(3) Assessment results, including improvement offers on items needing improvement, are reported to the assessed organization. In addition, the final product including the result of the report to the assessed organization is 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 is a process for analyzing and resolving any problems that are detected during the development process (refer to 5.3), the operational process (refer to 5.4), the maintenance process (refer to 5.5), or other processes.

In addition, types of modification in problem resolution include correction and improvement.

This process consists of the following activities:

(1) Process implementation
(2) Problem resolution
(3) Problem trend analysis

(4) Corrective and preventive action

6.8.1 Process implementation

6.8.1.1 Problem resolution plan

Based on the quality assurance activity plan, the problem resolution process shall be established to handle all incidents and problems encountered in the software or the computer system.

According to the criteria of the problem, the following recording and countermeasure process shall be decided.

(1) Incident and problem management method (method of acceptance and recording, reporting, and method of maintenance (including storing period))
(2) Analysis of the cause of the problem and the resolution policy
(3) Follow-up policy for problem resolution status
(4) Policy for corrective and preventive action

6.8.2 Problem resolution

An incident shall be analyzed in accordance with the problem resolution plan. Then, if it is identified as a problem, this problem shall be resolved by performing necessary corrective action. Particularly in cases where a problem requires the modification of design specifications or source code, it shall be confirmed that the necessary items among the following activities based on the management level are included in the problem resolution plan, and appropriate problem resolution activities shall be performed.

6.8.2.1 Acceptance and record of incidents and problems

(1) All incidents and problems detected in the primary life cycle process and the support life cycle process shall be reported quickly after recording them for each process they affect, and the problems shall be accepted into the problem resolution process as quickly as possible.
(2) Recording of the incidents and problems shall be started.
(3) If an incident has occurred, it shall be identified and managed in order to be linked to the known incidents or problems.
(4) When a corrective action is required, it shall be managed as a problem. At that time, it is recommended that it shall be prioritized for resolution.
(5) Relevant parties shall be informed about the occurrence and status for incidents and problems.

6.8.2.2 Incident and problem investigation and analysis

The phenomenon and occurrence conditions of an incident shall be analyzed. Then, if the incident is identified as a problem, the root cause of it shall be investigated/analyzed to be clarified by performing the following actions. Problem resolution shall be requested to the other organizations, as necessary. In addition, the reproducibility of the reported problem shall be checked in test/maintenance environment, or verified by analysis or other impractical means if necessary. Verification methods to be applied shall be selected in accordance with the constraints and limits of the verification.

(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) 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) The determination as to the necessity of the modification (judgment whether the modification should be performed considering the effect of the modification, scale of the work, degree of fatality and emergency, and so on.) shall be performed.
(2) The method of modification shall be investigated based on the analysis. It is desirable that multiple candidates for the modification plan are investigated and the modification method is selectable from among the candidates.
(3) The modification method shall be recorded, and its correspondence to the problem report, change request, and analysis shall be managed so that they are easy to understand.
(4) Notes for the operation shall be documented.

6.8.2.4 Determination of problem resolution plan

Plans shall be documented and agreed with the parties:

(1) The problem resolution plan shall be determined and agreed with the related parties. The plan for the problem resolution shall be drafted, as necessary.
(2) In case the problem occurred after software release, as a temporary measure, if there is a way of mitigating the problem, the problem resolution plan shall be determined and agreed with related parties. The plan for the temporary problem resolution shall be drafted as necessary.

6.8.2.5 Problem resolution implementation

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

6.8.2.6 Problem report

A record of the problem shall be properly reported.

6.8.2.7 Recording and monitoring

A set of incident and 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.

6.8.2.8 Maintenance of records

Record of the incident and problem shall be stored for the period based on a plan.

6.8.3 Problem trend analysis

Incident and problem trend analysis shall be performed.

6.8.3 Corrective and preventive action

For the serious problems (an occurrence of similar problems and so on), the action to prevent from the reoccurrence, shall be taken, as necessary. And roll out of the proper information shall be implemented to the related or similar work process and products, and related department. 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.

Appendix

Appendix 1 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-appendix 1.

Classification	Software products					Environment (Enabling system or service)	
	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, or 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 are obtained by requiring the development of the developer's organization and suppliers to integrate the knowledge of the developer's organization and reflect the knowledge of organizations	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-appendix 1
Relationships between software products, knowledge assets, and enabling systems

Appendix 2　5.5.3 Addition for modification implementation

When software modification (including parameters) is implemented, the development processes shall be performed again from the highest level process influenced by the modification. When modifications are made retroactively to the computer system requirements analysis, for example, the development processes after “5.3.3 Computer system requirements analysis” shall be performed again and checked. Afterward, the reprogramming of software is performed.

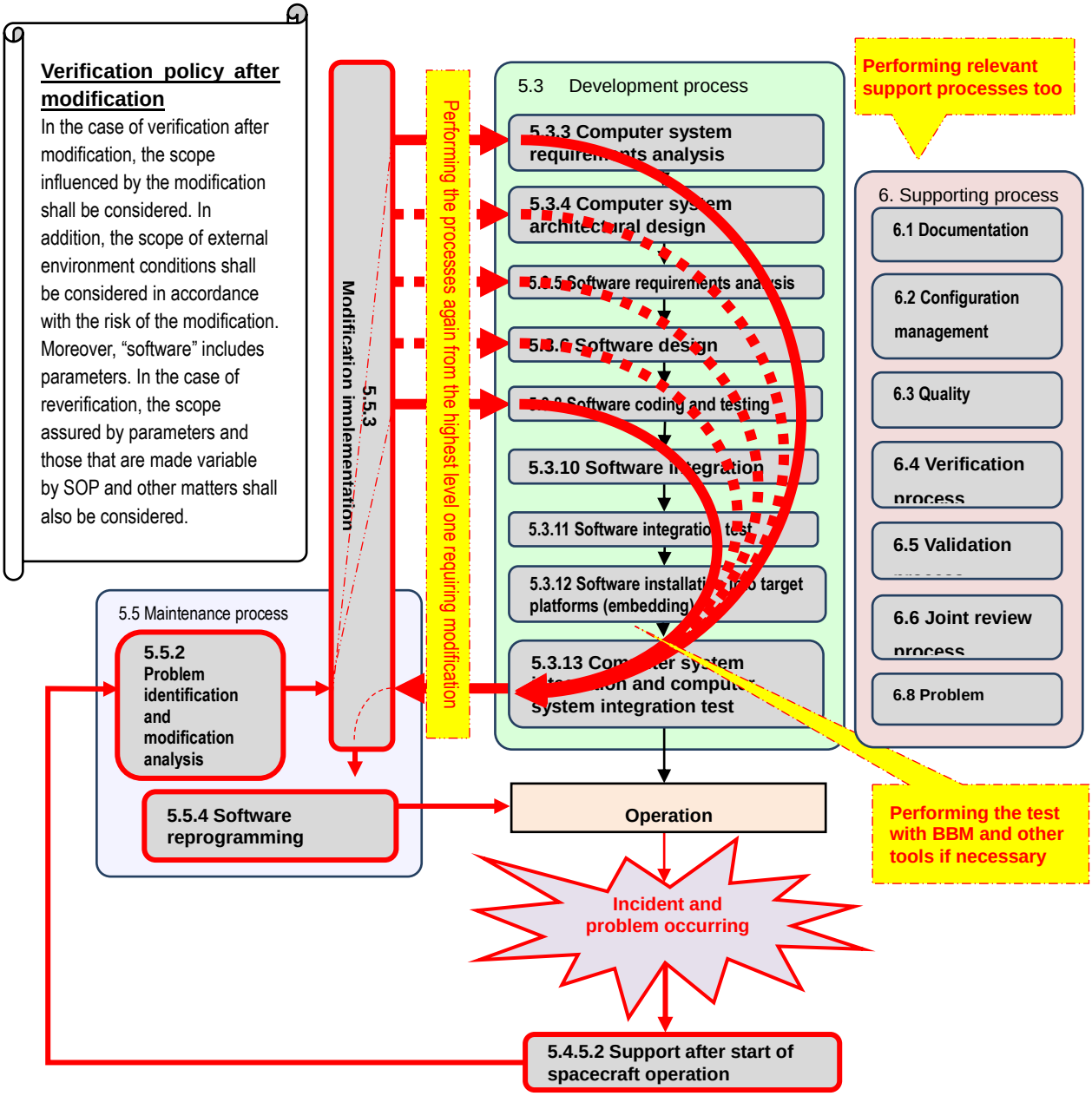

Figure-appendix 2. Addition for “5.5.3 Modification implementation”

Appendix 3　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 operational process are shown in below.

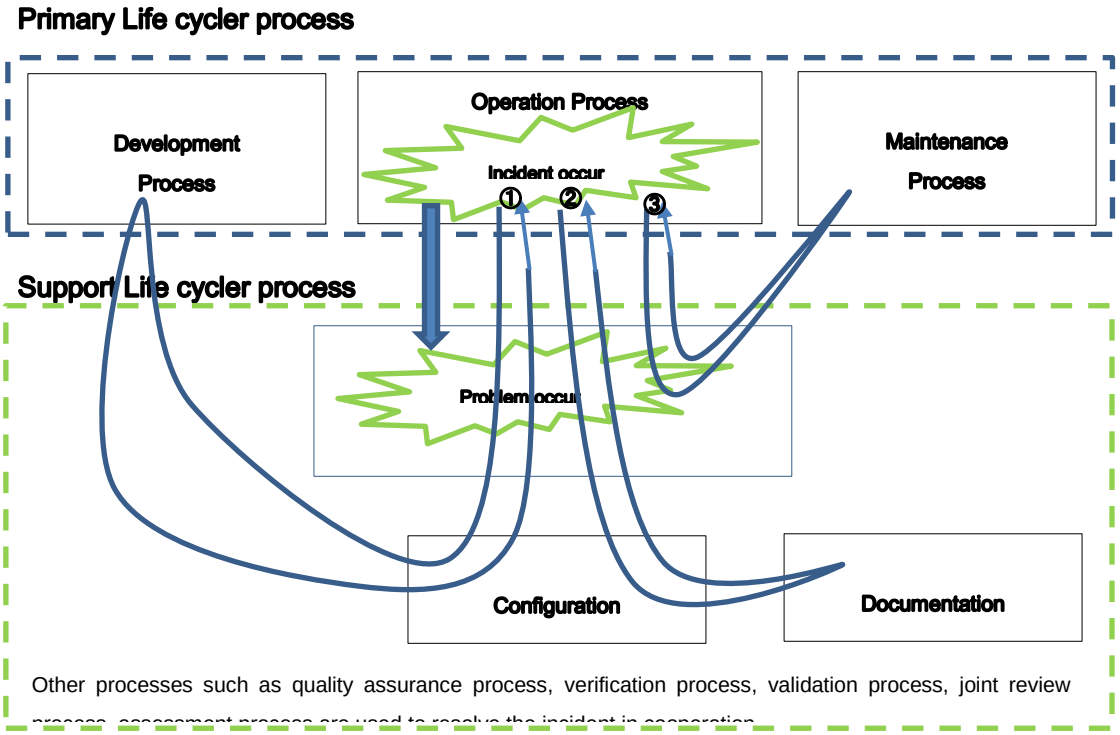

Figure-appendix 3-1.
Relationship between the problem resolution process and another process

[(1) In case of software modification]
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. → Modify software in the development process. → Return software to the configuration management process. → Verify the results in the problem resolution process. → Return to the operational 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 operational 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 operational 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/situation that has occurred, a “problem” indicates an event/condition/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.”

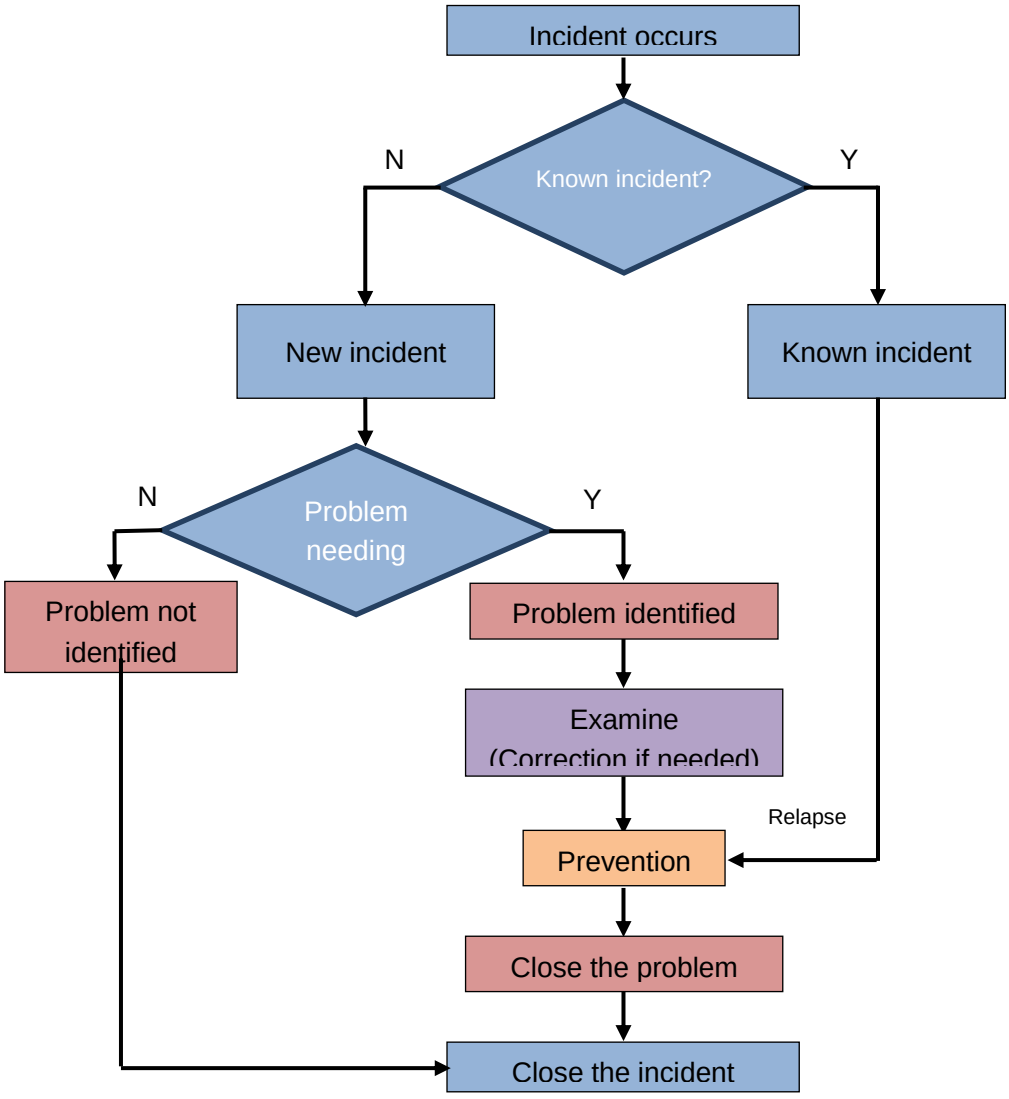

Figure-appendix 2-2.
Procedure overview of incident and problem in the software development standard