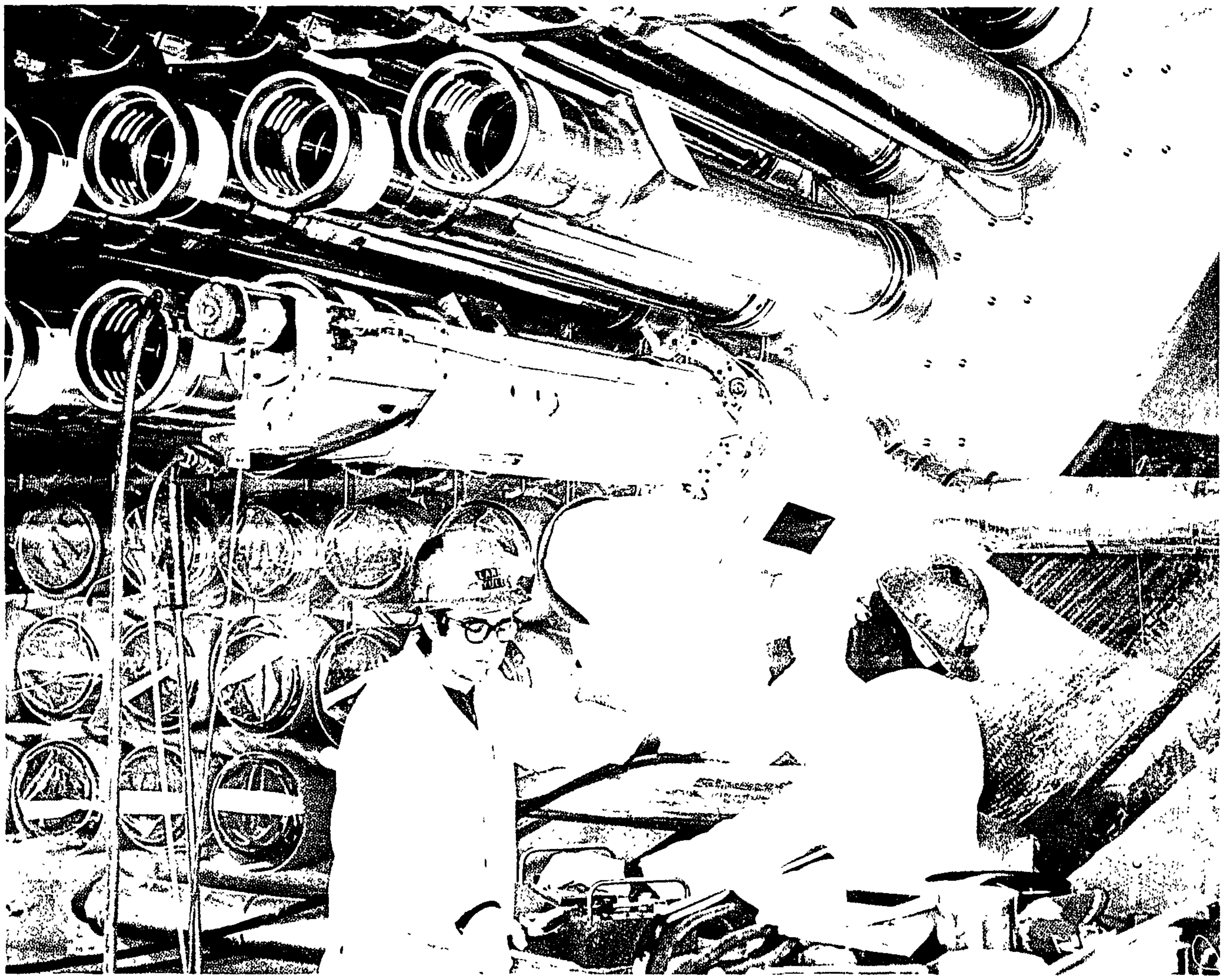
CA9600029

N286.7-94

Quality Assurance of Analytical, Scientific, and Design Computer Programs for Nuclear Power Plants

Nuclear



General Instruction No. 1
N286.7-94
June 1994

CSA Standard N286.7-94, *Quality Assurance of Analytical, Scientific, and Design Computer Programs for Nuclear Power Plants,* consists of **33 pages** (viii preliminary and 25 text), each dated **June 1994**.

This Standard, like all CSA Standards, is subject to periodic review, and amendments in the form of replacement pages may be issued from time to time; such pages will be mailed automatically to those purchasers who complete and return the attached card.* Some Standards require frequent revision between editions, whereas others require none at all. It is planned to issue new editions of the Standard, regardless of the amount of revision, at intervals not greater than 5 years. Except in unusual circumstances, replacement pages will not be issued during the last year of that edition.

**This card will appear with General Instruction No. 1 only.*

Although any replacement pages that have been issued will be sold with the Standard, it is for the purchaser to insert them where they apply. The responsibility for ensuring that his or her copy is complete rests with the holder of the Standard, who should, for the sake of reference, retain those pages which have been replaced.

Note: *A General Instruction sheet will accompany replacement pages each time they are issued and will list the latest date of each page of the Standard.*

Cut along dotted line

Name ______________________________

Organization ______________________________

____

Address ______________________________

City ______________________________

Province/State ______________________________

Country ______________ Postal/Zip Code ______________

N286.7-94

Place
Stamp
Here

Canadian Standards Association
Consolidated Mailing List
178 Rexdale Boulevard
Rexdale (Toronto), Ontario
Canada
M9W 1R3

N286.7-94

Quality Assurance of Analytical, Scientific, and Design Computer Programs for Nuclear Power Plants

Nuclear

ISSN 0317-7874
Published in June 1994
by
Canadian Standards Association
178 Rexdale Boulevard,
Rexdale (Toronto), Ontario,
Canada
M9W 1R3

Technical Editor: *Brian Weir*
Managing Editor: *Bernard Kelly*

Contents

Technical Committee on Overall Quality Assurance for Nuclear Power Plants

Preface

This is the first edition of CSA Preliminary Standard N286.7, *Quality Assurance of Analytical, Scientific, and Design Computer Programs for Nuclear Power Plants.* This Standard addresses computer programs for analytical, scientific, and design applications for safety-related systems and components.

This Standard was approved by the CSA Technical Committee on Overall Quality Assurance for Nuclear Power Plants under the jurisdiction of the Steering Committee on Nuclear Standards.

The increasing importance of analytical, scientific, and design computer programs has made it essential that such programs be developed in a formal, disciplined manner. Such computer programs are recognized as a distinct group of products and appropriate quality assurance is recognized as a valuable and complementary management technique for the development, use, and maintenance of such computer programs.

This document is being issued as a Preliminary Standard to create a document that can be referenced by the nuclear industry and to allow the industry to explore the application of the Standard in its sphere of activities. This Standard will offer to industry users the opportunity to gain experience before a formal Standard is published.

This Standard complements the Standards in the CAN/CSA-N286 series of Standards which define quality assurance requirements for the activities related to a nuclear power plant project.

In addition, the CAN/CSA-Q396 series of Standards should be used where the procurement of computer programs and associated documentation from external vendors is involved.

June 1994

Notes:
(1) *Use of the singular in this Standard does not exclude the plural (and vice versa) when the sense allows.*
(2) *Although the intended primary application of this Standard is stated in its Scope, it is important to note that it remains the responsibility of the users of the Standard to judge its suitability for their particular purpose.*
(3) *This publication was developed by consensus, which is defined by* CSA Regulations Governing Standardization *as "substantial agreement reached by concerned interests. Consensus includes an attempt to remove all objections and implies much more than the concept of a simple majority, but not necessarily unanimity." It is consistent with this definition that a member may be included in the Technical Committee list and yet not be in full agreement with all clauses of the publication.*
(4) *CSA Standards are subject to periodic review, and suggestions for their improvement will be referred to the appropriate committee.*
(5) *All enquiries regarding this Standard, including requests for interpretation, should be addressed to Canadian Standards Association, Standards Development, 178 Rexdale Boulevard, Rexdale, Ontario M9W 1R3.*
Requests for interpretation should
(a) define the problem, making reference to the specific clause, and, where appropriate, include an illustrative sketch;
(b) provide an explanation of circumstances surrounding the actual field condition; and
(c) be phrased where possible to permit a specific "yes" or "no" answer.
Interpretations are published in CSA's periodical Info Update. *For subscription details, write to CSA Sales Promotion,* Info Update, *at the address given above.*

N286.7-94 ***Quality Assurance of Analytical, Scientific, and Design Computer Programs for Nuclear Power Plants***

1. Scope

1.1

This Standard applies to the design and development, modification, documentation, execution, and configuration management of computer programs used to perform analytical, scientific, and design computations during the design and analysis of safety-related nuclear power plant equipment, systems, structures, and components as identified by the owner.

1.2

The principal activities associated with such computer programs are shown in Figure 1.1. This functional model illustrates the main interactions between these elements. The double arrows indicate how the principal activities influence each other. For example, a modification may affect the execution of a computer program. Also, experience with a computer program during execution may lead to a modification.

Solid lines in Figure 1.1 indicate interactions between activities within the scope of the Standard. Interpretation of computer program results for design and analysis is outside the scope of this Standard and is normally covered by the requirements of CSA Standard CAN3-N286.2.

1.3

This Standard does not apply to commercially available and widely used database, spreadsheet, and graphics computer programs, or compilers, interpreters, and operating systems.

1.4

This Standard applies to owners and designated participants who have implemented quality assurance programs in accordance with the applicable CSA Standard in the CAN/CSA-N286 series of Standards. The requirements identified in this Standard complement those requirements in the CAN/CSA-N286 series of Standards to ensure that a complete set of quality assurance program requirements is applied to computer programs in analytical, scientific, and design applications. In particular, this applies to audits and quality assurance program reviews.

1.5

Where computer programs and associated documentation are procured from vendors who are not participants, the appropriate requirements of the CAN/CSA-Q396 series of Standards, as defined by the procuring organization, apply.

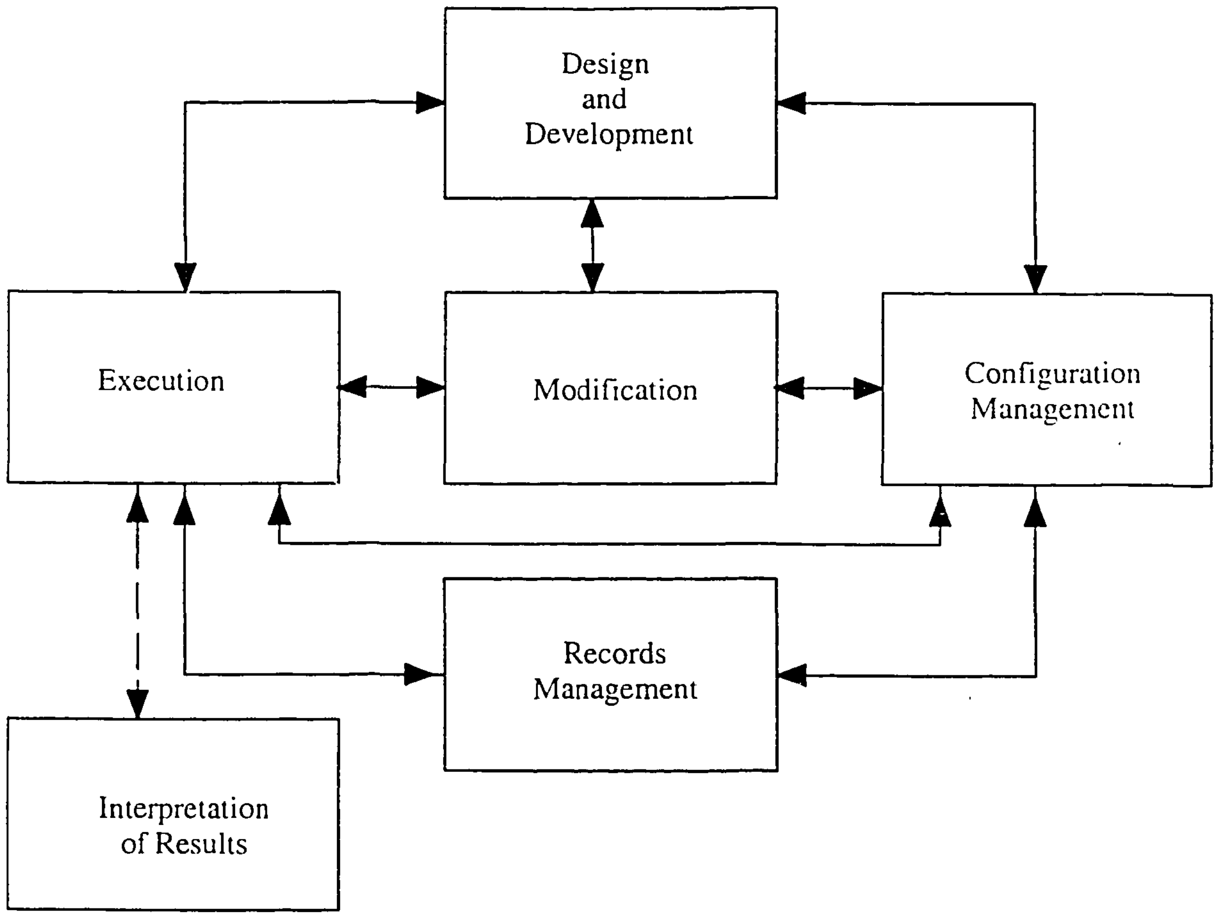


Figure 1.1
Computer Programs for Design and Analytical Work —
A Functional Model

2. Definitions

The following definitions apply in this Standard:

Acceptance testing — formal testing conducted to determine whether or not a system satisfies the acceptance criteria as set out by the owner or participant and to enable the customer to determine whether or not to accept the system.

Algorithm — a finite set of welldefined rules
(a) for the solution of a problem in a finite number of steps;
(b) that gives a sequence of operations for performing a specific task.

Assemble — to translate a program expressed in an assembly language into a machine language.

Assembly language — a machine-specific language containing instructions that are usually in one-to-one correspondence with computer instructions.

Change control — the process by which a change is proposed, evaluated, approved or rejected, scheduled, and tracked.

Code —
(a) to represent data or a computer program in a symbolic form that can be accepted by a central processor unit;
(b) data or computer program in a form that can be accepted by a computer program.

Compile — to translate a high level language program into its machine code equivalent.

Computer program — a sequence of instructions suitable for processing by a computer and designed to achieve a certain result.

Computer program abstract — a brief description of a computer program, providing sufficient information for potential users to determine the appropriateness of the computer program to their needs and resources.

Computer program routine — a computer program segment that performs a specific task.

Computer system — a system composed of computer(s), peripheral equipment such as disks, printers, and terminals, and the computer programs necessary to make them operate together.

Central processor unit — the computing system that contains the circuits that control the interpretation and execution of instructions, including the necessary arithmetic, logic, and control circuits to execute the instructions.

Comment statement — a statement that is embedded in a computer program in order to provide clarification to human readers and that does not affect machine interpretation.

Configuration component — a collection of computer program elements treated as a unit for the purpose of configuration management.

Configuration management — the process of identifying configuration components, controlling changes, and maintaining the integrity and traceability of the arrangement of the computer program components.

Construct — a complex image or idea formed from a number of simpler images or ideas.

Control structure — a construct that determines the flow of control through a computer program.

Correctness — the extent to which a computer program is free from design and coding defects.

Data structure — a representation of logical relationships among individual data elements.

Debugging — the process of locating, analyzing, and correcting suspected faults.

Design phase — the period of time in the computer program life cycle during which the designs for architecture, computer program components, interfaces, and data are created, documented, and verified to satisfy requirements.

Developer — the party that creates a computer program or a component thereof and its associated documents for its own use or that of others.

Development phase — the period of time that begins with the decision to develop a computer program and ends when the computer program is delivered.

Document — a data medium and the data recorded on it, that generally has permanence and that can be read by a person or machine.

Documentation — a collection of documents on a given subject.

Embedded computer program — computer program for an embedded computer system.

Embedded computer system — a computer system that is integral to a large system whose primary purpose is not computational.

Empirical correlation — a mathematical representation of an assumed interdependence between two or more variables based on actual measurement, observation, or experience, rather than theory.

Error — a discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition, including that resulting from human actions.

Executable code — a computer program in a language that can be directly executed by a computer.

Fault — a manifestation of an error in a computer program.

File — a set of related computer records treated as a unit.

Flowchart — a graphical representation of the definition, analysis, or solution of a problem in which symbols are used to represent operations, data, flow, and equipment.

Hardware — physical equipment used in data processing, as opposed to computer programs, procedures, rules, and associated documents.

Hierarchy — a structure whose components are ranked into levels of subordination according to a specific set of rules.

High level language — a programming language that usually includes features such as nested expressions, user-defined data types, and parameter-passing not normally found in lower level languages, that does not reflect the structure of any one given computer or class of computers, and that can be used to write machine-independent source programs.

Instruction — in a programming language, a meaningful expression that specifies and identifies its operands.

Integration — the process of combining computer program elements, hardware elements, or both into an overall system.

Integration testing — an orderly progression of testing in which computer program elements, hardware elements, or both are combined and exercised until the entire system has been integrated.

Interface — a shared boundary between computer system components.

Interpret — to translate and to execute each source language statement of a computer program before translating and executing the next statement.

Job control language — a language that expresses the statements of a job and that is used to identify the job or describe its requirements, usually to an operating system.

Latent defect — a fault that exists but is as yet concealed.

Lifecycle — the period of time that starts when a computer program is conceived and ends when the program is no longer available for use and analyses performed with it no longer support any nuclear power plant operating licences.

Milestone — a scheduled event that is used to measure progress.

Model — a representation of a device, concept, or physical process.

Module — a computer program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading into the computer.

Module header — a set of comments forming part of the source code of a module that identifies it and defines its purpose and use.

Nest — to incorporate a structure or structures of a certain kind into a structure of the same kind.

Nonconformance — a deficiency in characteristic, documentation, or procedure that renders the quality of a computer program unacceptable or indeterminate, or not according to specified requirements.

Object module — a module containing a fully compiled or assembled program that is ready to be loaded into a computer.

Operand — an entity on which an operation is performed.

Operating system — computer programs that control the execution of computer programs, provide hardware resource allocations, input and output control, and related services.

Organization — the owner of or participant in a nuclear power plant project.

Owner — the party who has or will have title to a nuclear power plant.

Parameter — a variable that is used to pass values between program routines.

Participant — an organization required by the owner to meet one or more of the secondtier Standards in the N286 series.

Portability — the ease with which a computer program can be transferred from one computer system or environment to another.

Process — in a computer system, a unique, finite course of events defined by its purpose or by its effect, achieved under given conditions.

Program component — a basic part of a computer program.

Programming language — an artificial language designed to generate or express computer programs.

Program loop — a set of instructions in a computer program that may be executed repeatedly while a certain condition prevails.

Program statement — see Source statement.

Program structure — a representation of the logical relationship among individual program components.

Programming segment — the sequence of computer program statements between two consecutive branch points.

Quality assurance — a planned and systematic pattern of actions designed to provide adequate confidence that a computer program will be of the required quality.

Record — information stored in a document or other media that furnishes evidence of the quality and/or activities.

Safety-related system — those systems, and the components and structures thereof, which, by virtue of failure to perform in accordance with the design intent, have the potential to impact on the radiological safety of the public or plant personnel from the operation of the nuclear power plant. Those systems, and the components and structures thereof, are associated with
(a) the regulation (including controlled startup and shutdown) and cooling of the reactor core under normal conditions (including all normal operating and shutdown conditions);
(b) the regulation, shutdown, and cooling of the reactor core under anticipated transient conditions, accident conditions, and the maintenance of the reactor core in a safe shutdown state for an extended period following such conditions; and
(c) limiting the release of radioactive material and the exposure of plant personnel and/or the public to meet the criteria established by the licensing authority with respect to radiation exposure during and following normal, anticipated transient conditions and accident conditions.

Notes:
(1) *The term "safety-related system" covers a broad range of systems, from those having very important safety functions to those with a less direct effect on safety. The larger the potential radiological safety effect due to system failure, the stronger the "safety-related" connotation.*
(2) *The term "safety-related" also applies to certain activities associated with the design, manufacture, construction, commissioning, and operation of safety-related systems, and to other activities that could similarly affect the radiological safety of the public or plant personnel, such as environmental and effluent monitoring, radiation protection and dosimetry, and radioactive material handling (including waste management). The larger the potential radiologic safety effect associated with the performance of the activity, the stronger the "safety-related" connotation.*
(3) *Certain failures of other systems could adversely affect a safety-related system (eg, through flooding or by mechanical damage). The potential for this and the necessary means to control it must be addressed in appropriate phases.*

Source module — a module in a computer program that must be compiled, assembled, or interpreted before being executed by a computer.

Source program — a computer program that must be compiled, assembled, or interpreted before being executed by a computer.

Source language — a language used to write programs that must be compiled, assembled, or interpreted before being executed by a computer.

Source statement — an instruction in a computer program that must be compiled, assembled, or interpreted before being executed by a computer.

Subprogram — a computer program unit that may be invoked by one or more other computer program units.

System — a collection of hardware, computer programs, people, facilities, and procedures organized to accomplish some common objectives.

System library — a controlled collection of system-resident computer programs that can be accessed for use or incorporated into other programs by reference.

Testing — the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements.

Unit — see Module.

User — a party that uses a computer program to perform analytical, scientific, and/or design computations related to nuclear power plant systems, equipment, and components.

Validation — the process of evaluating a computer program by a
(a) developer at the end of its development process to ensure compliance with the (original) requirements (developer's perspective);
(b) user in order to confirm its applicability to the user's specific needs (user's perspective).

Variable —
(a) a quantity that can assume any of a given set of values;
(b) a character or a group of characters that refers to a value and, in execution of a computer program, corresponds to a specific memory location.

Verification — the process of determining whether or not the products of a given phase of the computer program development cycle fulfil the requirements established during the previous phase. Reviews, testing, and walkthroughs are some of the means of accomplishing verification.
Note: *This definition of the term "Verification" applies to activities related to the computer program development cycle and differs from the definition given in CSA Standard CAN3-N286.2 which applies to design activities.*

Walkthrough — a review process in which a designer or programmer leads one or more other members of the development team through a segment of design or computer program that she/he has written, while the other members ask questions and make comments about technique, style, possible errors, violation of development standards, and other problems.

3. References

This Standard refers to the following publications, and where such reference is made, it shall be to the editions listed below, including all amendments published thereto:

CSA Standards
CAN/CSA-N286.0-92,
Quality Assurance Program Requirements for Nuclear Power Plants;

CAN3-N286.1-84,
Procurement Quality Assurance for Nuclear Power Plants;

CAN3-N286.2-86,
Design Quality Assurance for Nuclear Power Plants;

CAN3-N286.3-83,
Construction Quality Assurance for Nuclear Power Plants;

CAN/CSA-N286.4-M86,
Commissioning Quality Assurance for Nuclear Power Plants;

CAN/CSA-N286.5-M87,
Operations Quality Assurance for Nuclear Power Plants;

CAN/CSA-Q396.1.1-89,
Quality Assurance Program for the Development of Software Used in Critical Applications;

CAN/CSA-Q396.1.2-89,
Quality Assurance Program for Previously Developed Software Used in Critical Applications;

CAN/CSA-Q396.2.1-89,
Quality Assurance Program for the Development of Software Used in Noncritical Applications;

CAN/CSA-Q396.2.2-89,
Quality Assurance Program for Previously Developed Software used in Noncritical Applications.

ANSI* Standards
ANSI/ANS†-10.2-1988,
Recommended Programming Practices to Facilitate the Portability of Scientific Computer Programs;

ANSI/ANS-10.3-1986,
Guidelines for the Documentation of Digital Computer Programs;

ANSI/ANS-10.5-1986,
Guideline for Considering User Needs in Computer Program Development;

ANSI/IEEE‡ 1063-1989,
IEEE Standard for Software User Documentation.

**American National Standards Institute.*
†*American Nuclear Society.*
‡*Institute of Electrical and Electronic Engineers.*

4. Responsibilities

4.1 Owner's Responsibilities
The owner shall be responsible for compliance with this Standard.

4.2 Organizations' Responsibilities
(a) Each organization shall establish appropriate procedures to control the work associated with computer programs used for analytical, scientific, and design applications in a manner consistent with this Standard and the applicable Standard in the CAN3-N286 series of Standards.
(b) Each organization shall ensure that any change to a computer program or its associated documents is verified before the program is released for use and that records of verification are maintained.
(c) Each organization responsible for computer program development is also accountable for its quality. Therefore, the responsible organization shall undertake a commitment to adhere to the computer program user's stated requirements and to the declared development program.
(d) Each organization responsible for computer program development shall notify other affected organizations of changes in computer programs or associated documents.

(e) Where an organization limits its activities to the use of previously developed and validated computer programs it shall comply, as a minimum, with the applicable requirements stated in this Standard in Clauses 5.5.1, Requirements Specification and Verification; 5.6, Validation; 6, Configuration Management; 7, Change Control; 8, Computer Program/Hardware Integration; 9, Execution of Computer Programs; and 10, Documentation.
(f) When selecting a specific product from a developer's normal line of products, an organization shall obtain evidence that the product has been developed in compliance with this Standard. Otherwise, the organization shall validate the product prior to use.
(g) Each organization shall identify those individuals responsible for providing expert guidance in the proper use of a particular computer program.
(h) Each organization shall establish the required qualifications for personnel involved in development and use of computer programs. Computer program use includes the production of user models, subprograms to the computer program, and the execution of the computer program. The organization shall ensure that only qualified personnel perform these activities and that the computer program and associated documentation is properly used.
(i) Each organization acquiring a computer program externally shall ensure that the computer program has been validated for the specific application.
(j) Each organization shall establish measures for
 (i) evaluating information from previous analyses performed with the computer program; and
 (ii) incorporating resultant improvements in
  (1) the execution and maintenance of computer programs; and
  (2) computer program development practices.

5. Computer Program Design and Development

5.1 General

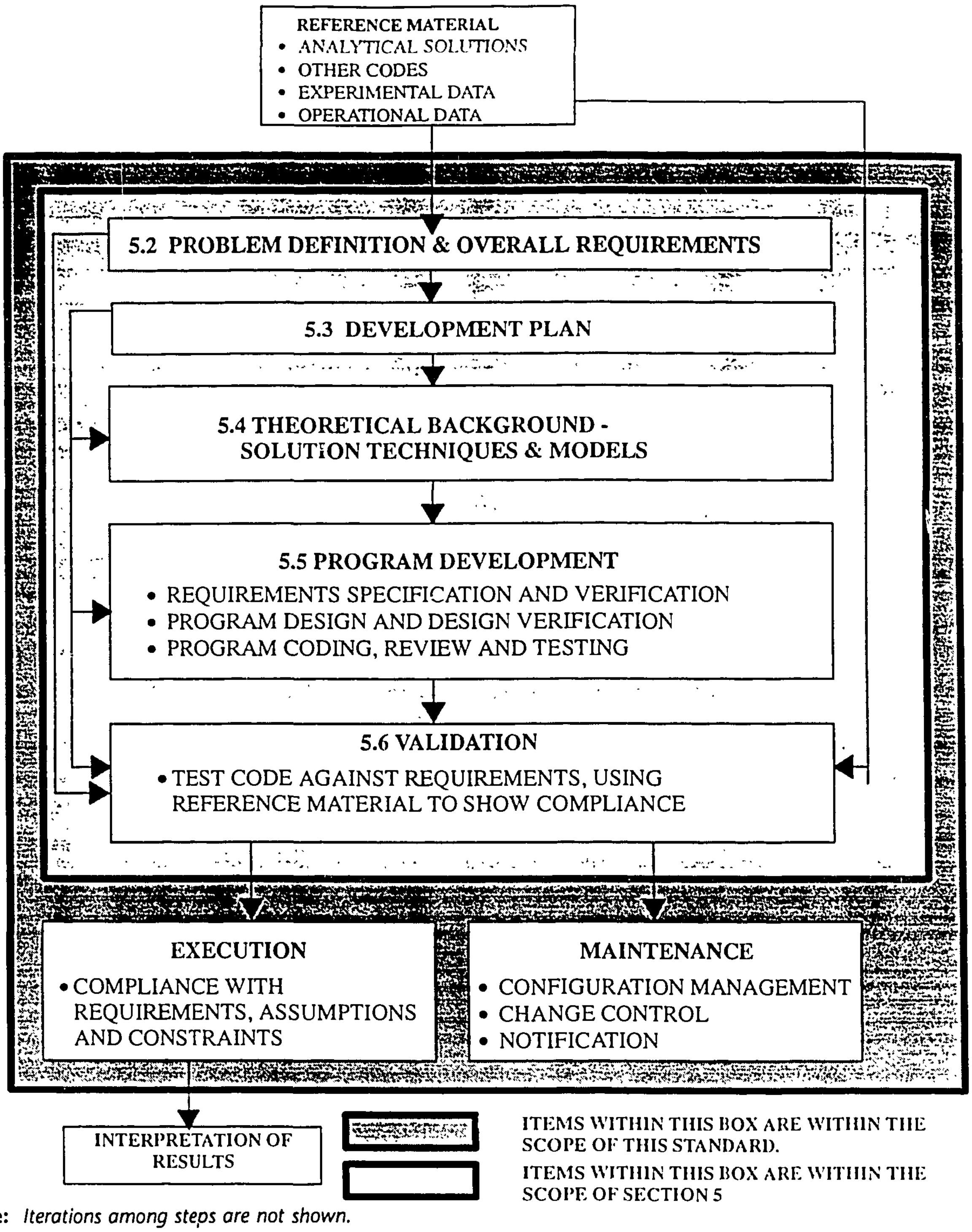
Figure 5.1 shows the overall process that is typically followed in the development and usage of a program. The large shaded box shows the development, execution, and maintenance processes for which requirements are described in this Standard. The smaller shaded box shows the development processes for which requirements are described in Clause 5, and the relationship between these processes and other processes covered by this Standard.

The subsequent clauses in this section describe requirements for the components of the process of developing a computer program. Figure 5.1 shows the clause corresponding to each major step in the development process. Requirements on documentation to demonstrate that the requirements of this section are met are described in Clause 10.

This Clause is based on the premise that the computer program's correctness shall be ensured through testing. In addition, each major development step shall be followed by a review of the resulting output to confirm compliance with requirements, performed by suitably qualified persons who did not perform the activity.

5.2 Problem Definition and Overall Requirements

The physical problem to be solved by the computer program, along with a clear statement of the overall requirements (for the computer program) shall be documented. The operating and boundary conditions under which the problem is to be solved shall be described. An explanation shall be provided for the nomenclature or any conventions used to describe the physical problem.


REFERENCE MATERIAL
• ANALYTICAL SOLUTIONS
• OTHER CODES
• EXPERIMENTAL DATA
• OPERATIONAL DATA
5.2 PROBLEM DEFINITION & OVERALL REQUIREMENTS
5.3 DEVELOPMENT PLAN
5.4 THEORETICAL BACKGROUND - SOLUTION TECHNIQUES & MODELS
5.5 PROGRAM DEVELOPMENT
• REQUIREMENTS SPECIFICATION AND VERIFICATION
• PROGRAM DESIGN AND DESIGN VERIFICATION
• PROGRAM CODING, REVIEW AND TESTING
5.6 VALIDATION
• TEST CODE AGAINST REQUIREMENTS, USING REFERENCE MATERIAL TO SHOW COMPLIANCE
EXECUTION
• COMPLIANCE WITH REQUIREMENTS, ASSUMPTIONS AND CONSTRAINTS
MAINTENANCE
• CONFIGURATION MANAGEMENT
• CHANGE CONTROL
• NOTIFICATION
INTERPRETATION OF RESULTS
ITEMS WITHIN THIS BOX ARE WITHIN THE SCOPE OF THIS STANDARD.
ITEMS WITHIN THIS BOX ARE WITHIN THE SCOPE OF SECTION 5

Note: *Iterations among steps are not shown.*

Figure 5.1
Computer Program Development and Usage Process

5.3 Development Plan

At the start of computer program development, the developer organization shall prepare a plan for the development of the computer program. This development plan shall specify the theory development, specification, design, coding, testing, verification, and validation tasks to be performed. The development plan shall be followed throughout the development of the computer program, although the plan may be updated from time to time during the course of computer program development to reflect improvements deemed to be necessary.

The development plan shall contain, or reference documents that contain, the information relevant to the development. This shall include

(a) the computer program's required functions, quality attributes (eg, reviewability, maintainability, etc), and how they are to be achieved, an any performance characteristics;
(b) the computer program development life cycle to be followed;
(c) the breakdown of computer program development into manageable tasks and the assignment of related responsibilities, including review and approval authority;
(d) the sequence and timing of activities to be performed including their output products;
(e) the development tools, techniques, and methodologies to be used;
(f) the review, testing, verification, and validation activities to be performed, and methods to be used, and the rationale for their selection;
(g) means to achieve independence between the activities in Item (f) above, and those related to the specification, design, and coding of computer programs;
(h) the key milestones in the development process;
(i) any computer program components to be developed by others and the applicable quality assurance programs;
(j) the methods to control interfaces between all contributors to the computer program development including the intended user organization;
(k) identification of the specific documents to be produced as part of the development process, a description of their purpose and content, and identification of responsibility for producing, reviewing, and approving documents; and
(l) identification of the specific configuration management methods for the project.

5.4 Theoretical Background—Solution Techniques and Models

The theory is the basis for development of the computer program. The theory shall be documented according to the requirements in Clause 10, and, as a minimum, shall

(a) specify assumptions and constraints used in the model that are developed to solve the problem;
(b) specify the solution parameters (the type of results) and any transient characteristics;
(c) describe the engineering principles and mathematical equations used to model the system and its components;
(d) describe the solution techniques to be employed, along with the rationale for selecting these techniques; a key aspect of the rationale is the accuracy requirements, and these must be specified, as well as any other limitations associated with the solution techniques;
(e) describe how the equations are transformed into forms suitable for application of the solution techniques;
(f) specify any empirical correlations and their uncertainties; and
(g) reference published material (such as texts or journals), or previously released reports, to support the model.

5.5 Design and Development

5.5.1 Requirements Specification and Verification

5.5.1.1 Requirements Specification

The requirements for the computer program shall be prepared prior to the start of computer program design. The requirements shall specify
(a) the problem to be solved with reference to the theory as a basis for validation;
(b) the expected classes of problems to be analyzed;
(c) assumptions and dependencies;
(d) user interface requirements;
(e) performance requirements such as computational speed;
(f) interfaces to existing or planned computer systems external to the host computer;
(g) the computer systems on which the computer program is to run, including system requirements and system limitations;
(h) computer program and file size limitations;
(i) requirements for portability of the computer program among several computer systems;
(j) operating system requirements;
(k) library functions to be used;
(l) definitions of the input and output data;
(m) the programming language to be used;
(n) file type requirements, including requirements on output format;
(o) the models and algorithms to be used in the computer program;
(p) the mathematical relationship between the output data and the input data;
(q) the required response to exceptions, including detection of invalid data or results, detection of conditions outside the range covered by the theory, and detection of calculation time limits being exceeded;
(r) accuracy requirements; and
(s) requirements for degraded operation.

5.5.1.2 Requirements Verification

Verification of the requirements shall demonstrate that a program that meets the requirements will solve the problem, while meeting all specified constraints. Any discrepancies shall be explained, justified, and approved according to the authority described in the development plan. Persons performing the verification shall be independent of those preparing the requirements.

5.5.2 Computer Program Design and Verification

5.5.2.1 Design

The design activities shall ensure that all of the requirements are met. The design activities shall include, as a minimum, the
(a) identification of the algorithms to be used to implement the functional requirements, including both algorithms already specified in the requirements and algorithms that have been chosen by the designer;
(b) determination of computational error sources, their effect on the overall accuracy of the results, and any steps taken to minimize errors such that the accuracy requirements are met;
(c) development of the program structure, including program flow, control structures, and processes;

(d) specification of modules and module interfaces to a level of detail sufficient to perform coding without performing further design activities, and to perform verification of the design without needing to refer to the computer program itself;
(e) setting of performance requirements for each module;
(f) identification of the source and type of data inputs to each module;
(g) identification of each module's input and output;
(h) setting of processing order dependencies; and
(i) use of available system library functions.

5.5.2.2 Design Verification

The computer program design shall be verified to ensure that it meets the requirements described in Clause 5.5.1. Design verification shall be performed by persons who were not involved in the design.

5.5.3 Computer Program Coding, Review, and Testing

5.5.3.1 Coding

The coding phase shall consist of translating the detailed design of each module into computer language, debugging the resultant computer program, and integrating the program modules.

Module coding and debugging shall be performed so that each module performs the functions identified in the computer program design.

Integration of the computer program modules shall be performed to ensure that the integrated computer program performs as specified in the computer program design.

The developer shall define and document a system of uniform programming practices. Appropriate guidelines to measure conformance to defined programming practices are contained in Appendix A.

5.5.3.2 Computer Program Review

Computer program reviews shall be performed as an activity separate from, but complementary to, testing. Computer program reviews shall be performed by suitably qualified persons who did not perform the coding.

Computer program reviews shall be performed to demonstrate that the program performs according to the design. Examples of verification methods are walk-throughs, independent review of the computer program text, and mathematical analysis of the computer program functions.

5.5.3.3 Computer Program Testing

Unit testing and integration testing shall be performed to demonstrate that the computer program meets its requirements.

(a) Unit (module) testing is performed before the modules are integrated, and is carried out to ensure that each module performs as described by the design of that module.
(b) Integration testing is performed after some or all of the computer program's modules are integrated, and is carried out to demonstrate that the computer program meets its specifications.

Test documents shall be prepared in accordance with the requirements of Clauses 10.2.7 to 10.2.9.

Persons performing integration tests shall be independent of those designing and developing computer programs.

5.6 Validation

For internally developed computer programs, validation testing shall be performed to demonstrate that the integrated computer program fulfils its intent. Validation shall demonstrate that the computer program implements the theory, meeting all accuracy requirements and other constraints.

For externally acquired computer programs, the user organization shall perform a validation to demonstrate that the computer program meets all requirements of the intended application. This validation is in addition to any previous validation performed by the external computer program developer.

Examples of acceptable validation methods include comparison

(a) of the computer program output with results of hand calculations;
(b) with actual experience in operating the physical systems and processes modelled;
(c) with another validated computer program;
(d) with data from the technical literature or experimental data; and
(e) with mathematical solutions.

Discrepancies in the validation results shall be investigated and resolved.

Validation tests shall be reproducible. Conclusions regarding validation shall take into account the accuracy of any test data and any mathematical models, or numerical or logic algorithms, that are used during validation.

A computer program shall be considered proven and acceptable to validate another computer program by comparison, if its range of applicability, accuracy, and level of verification and validation are known and documented and are compatible with the computer program to be validated. The validating organization shall define methods to establish and document these characteristics.

Computer program validations shall be performed by suitably qualified persons who did not design or develop the computer program.

6. Configuration Management

6.1 General

The owner shall ensure that each organization implements a configuration management system applicable to that portion of the computer program life cycle for which it is responsible.

The owner shall ensure that the end of a computer program's life cycle be determined, justified, and communicated in writing to affected participants.

6.1.1

Procedures shall be established for controlling the input and withdrawal of computer program components from a given configuration and changes to a configuration. These procedures shall include the definition of the approval process and the test requirements to verify any new configuration. They shall also include the methods used for document control.

Data files shall be included as components of a configuration when the data affect the validity of the results of executing the computer program.

6.1.2

Methods shall be established to maintain records that trace the evolution of a computer program and its associated documentation. Methods shall be defined to preserve configuration components required to recover any previous configuration.

6.1.3
Changes to some configuration components that are beyond the control of the owner may be such that it becomes impractical to comply with Clause 6.1.3. An example is a change in the computer system or operating system. In such cases, a new and complete configuration has to be created for computer programs that are still in use. Consistency between this new configuration and the last version of it in the old system shall be demonstrated and documented.

6.1.4
Documentation requirements are defined in Clause 10.3.

6.2 Configuration Identification

6.2.1
The configuration management system shall define the methods used to identify the various computer program components defining a given configuration at a given time in the computer program life cycle. Included in these components are the operating system, compilers, the programming language, any specific subprograms or library functions used by the considered program, the control commands needed to execute the program, the source modules, and the object modules (if existing). The configuration management system shall apply to executable code only, when this form of the computer program is the only one available to the user organization. Wherever applicable, components shall be grouped functionally into libraries.

6.2.2
Documents related to each computer program component shall also be specified. The configuration management system shall relate specific versions of all computer program components to each other and to the corresponding revisions of associated documents.

6.3 Configuration Components Identification
The configuration management system shall define the methods used to specifically identify each configuration component. The methods shall ensure that the creation date and the configuration version of any component can be determined.

6.4 Change Identification
The configuration management system shall define the methods used to identify changes to a configuration component. The methods shall ensure that all documents associated with a change can be retrieved.

The change process itself shall comply, depending on the nature of the change, with the requirements of Clauses 7 to 9 inclusive.

7. Change Control

7.1 General

7.1.1
An organization may make changes to a computer program if the organization meets one of the following conditions:

(a) the organization is the developer of the computer program; or

(b) the organization has access to the applicable design and development documents and to the source code, and has staff who are qualified to make changes to the computer program and are experienced at developing similar computer programs.

7.1.2

Proposed changes shall be described, including identification of the computer program version and its associated documents, information connected with the corresponding deficiency or new requirements, and identification of the originating organization, rationale for the change, and affected computer program components.

7.2 Change Process

7.2.1

Each organization shall implement a system to control changes. This system shall ensure that proposed changes are documented, evaluated for impact on the computer program and the associated documents, approved and implemented according to defined procedures. The system shall ensure that adequate verifications and validations are carried out and that the resultant new version is identified.

7.2.2

The change control system shall require that
(a) the changes be cross-referenced to the corresponding deficiency, new requirement, or new method of computer program execution;
(b) the changes to the reference computer program be documented;
(c) the computer program changes be approved before implementation;
(d) the modified computer program be verified and validated to the requirements of Clause 7.4 before execution;
(e) where an established computer program version is modified to satisfy a special application related to an unusual analytical or design problem, the reference version of the computer program not be corrupted by the special modifications; and
(f) the documents describing the modification should form a part of the computer program documentation package and be subject to appropriate document control.

7.2.3

Changes shall be processed according to the change control plan, Clause 7.3.

7.2.4

The documentation of any change to a computer program shall describe the
(a) modified computer program and associated documents;
(b) identification of the revised computer program version;
(c) official date of issue;
(d) identification of affected computer program components;
(e) impact of the change on the execution of the computer program; and
(f) list of revised computer program documentation.

7.2.5

Each organization shall establish methods to ensure that all documents are revised to reflect the actual computer program changes.

7.3 Change Control Plan

7.3.1

The change control plan shall, as a minimum, meet the following requirements:
(a) Define the goal of the change including reference to the corresponding deficiency, new requirement, or new method of computer program execution.
(b) Identify the applicable computer program design and development processes such as preparing requirements specifications, design, coding, verification, testing, and validation.
(c) Establish the methods to implement the change.
(d) Identify the assessment methods to evaluate the impact of the proposed change on the execution and performance of the computer program.
(e) Define the documents to be issued. These documents shall, as a minimum, summarize any studies that were required to perform the change, give a brief description of the change, and present the results of the validation tests performed.
(f) Define the approval procedure the modified computer program and associated documents shall be subject to, before their issue.
(g) Specify the schedule of activities, and the resources necessary to execute the change.
(h) Identify the methods to ensure compliance with the requirements specified during the design and development of the original computer program and associated documents (see Clauses 5.4 and 5.5).

7.3.2

If changes are expected to be done frequently or regularly, a single, standard change control plan may be established by the organization.

7.4 Verification and Validation of Changes

7.4.1

Each organization shall implement a system to ensure that the development of approved changes follows a defined process that is consistent with that used to develop any previous version of the computer program, including verifications, testing, and validation.

7.4.2

Sufficient tests shall be done to
(a) ensure that the resulting modified computer program and associated documents satisfy the intent of the proposed change, and solve the corresponding deficiency or meet the new requirements or methods; and
(b) assess the impact of the change on the execution and performance of the modified computer program, especially with regard to undesired effects.

8. Computer Program/Hardware Integration

Computer programs may be executed on
(a) the same computer hardware on which they were developed; or
(b) different hardware.

If (a) applies, the requirements of Clauses 8.1 to 8.3 do not apply.

If (b) applies, or if there are significant changes to the computer hardware on which the computer program was developed under (a), then the organization performing the integration shall satisfy the requirements of Clauses 8.1 to 8.3.

8.1
The integration of computer programs and hardware shall be documented and verified.

8.2
The integration of computer programs and hardware includes, as a minimum, the specification of the
(a) computer program/hardware integration plan including verification method;
(b) integration test procedures and associated acceptance criteria that demonstrate the adequacy of the hardware and the computer program interfaces;
(c) test configuration for the computer system; and
(d) control of subsequent changes.

Test documents shall be prepared in accordance with the requirements of Clauses 10.2.7 to 10.2.9.

8.3
A test report shall record the results of the computer program/hardware integration testing. The test report shall identify the
(a) computer program and hardware used;
(b) test equipment and calibrations;
(c) test models used;
(d) acceptance criteria;
(e) test results,
(f) discrepancies; and
(g) corrective action taken.

9. Execution of Computer Programs

9.1 Requirements for Execution
Each organization shall implement methods for the proper execution of computer programs. Such methods shall ensure that the use of computer programs complies with the original requirements, assumptions, and constraints developed in accordance with the requirements of Clause 5.3, and that the solutions are valid. These methods shall ensure, as a minimum, that
(a) computer programs are validated before use;
(b) unless otherwise justified, only those physical states are analyzed with a computer program that are within the documented range of applicability;
(c) computer programs are used with due consideration for the assumptions inherent in the numerical techniques they employ;
(d) any user-supplied subprograms are documented and their interactions with the main computer program are validated;
(e) the input data are verified to ensure that they accurately reflect the physical systems or process to be analyzed;
(f) the derivations and sources of input data are fully documented in a form that facilitates review by an independent organization;
(g) the configuration of the computer program and the input data are completely identified such that results could be reproduced; and
(h) the output of computer programs is compared with the results of hand calculations, reference material, or other independently obtained results to reaffirm that the solutions are valid.

9.2 Feedback

9.2.1

Each organization shall implement methods to document and to communicate user experience to other users and to the developers. User feedback shall include as a minimum deficiencies and their remedies, further validations for new applications, and additional user-developed subprograms.

9.2.2

The organization responsible for maintaining the computer program shall implement methods for ensuring that user feedback is assessed and, where appropriate, incorporated by the change control system.

10. Documentation

10.1 General

10.1.1

Each organization shall ensure that the design and development of any computer program for analytical, scientific, and design applications has been properly documented in accordance with the requirements specified in this Clause.

10.1.2

Each organization shall maintain a list of documents that are clearly associated with a defined version of the related computer program.

10.1.3

All computer program design documents shall be subject to a program of design verification meeting the applicable requirements of CSA Standard CAN3-N286.2.

10.2 Design and Development Documents

The information that shall be documented and maintained during the design and development process is identified in this section. The information that must be documented is split into the following suggested list of documents that should be generated. However, if required, the information required to be documented can be consolidated into fewer documents as long as the contents of all the documents given below are covered:

(a) Computer Program Requirements Specification;
(b) Design and Development Plan;
(c) Computer Program Design Description;
(d) Verification and Validation Plan;
(e) Verification Report;
(f) Validation Report;
(g) Test Plans;
(h) Test Procedures;
(i) Test Reports;
(j) Configuration Management Plan; and
(k) Change Control Plan.

10.2.1 Computer Program Requirements Specification

The requirements specification shall include two separate sections, one describing the functional requirements and the other containing the specific design requirements of the computer program.

The functional requirements, in addition to Items (a), (b), (c), (e), (g), (h), (i) and (j) of Clause 5.5.1.1, shall include

(a) the name of the computer program to be developed;
(b) a description of objectives and goals;
(c) identification of the principal external interfaces of the computer program;
(d) a summary of the functions that the computer program will perform; and
(e) the general constraints, such as regulatory policies, that may limit the developer's options.

The specific computer program design requirements shall include the Items (d), (f), (k), (l), (m), (n), (o), (p), (q), (r), and (s) of Clause 5.5.1.1.

10.2.2 Development Plan

The design and development plan shall be prepared at the start of the design phase. This plan shall identify the design, development, and verification tasks to be performed. The plan shall be updated as required during the development process. The contents of the final plan should include Items (a) through (l) of Clause 5.3.

10.2.3 Computer Program Design Description

The computer program design description shall consist of the outputs of the activities identified in Clause 5.5.2.1.

10.2.4 Verification and Validation Plan

The verification and validation plan shall include verification and validation tasks to verify that the products of each life cycle phase summarized in Figure 5.1 comply with previous life cycle phase requirements and validate that the completed product complies with the system original requirements as defined in the requirements specification. The contents of the plan shall include

(a) a description of scope and purpose;
(b) identification of other documents required to supplement or implement this plan;
(c) a description of the organization, schedule, resources, responsibilities, and necessary tools;
(d) identification of relevant verification and validation tasks for each phase of the life cycle; and
(e) identification of inputs and outputs for each verification and validation task.

10.2.5 Verification Report

The type and format of the verification report shall be consistent with the owner's requirements. As a minimum, the verification report shall contain the following:

(a) the requirements to be verified;
(b) the method(s) and acceptance criteria used;
(c) a description of significant verification activities;
(d) the verification test results; and
(e) an assessment of verification results, including the extent to which these results comply with the requirements and disposition of any anomalies found.

10.2.6 Validation Report

The validation report shall be prepared by the developer to document the results of the evaluation of the computer program for the purpose of ensuring compliance with the original requirements; and by the user to document the results of his/her evaluation of the computer program with respect to its applicability to the user's specific needs.

The validation report shall contain
(a) identification of the validation requirements;
(b) a description of the methods used;
(c) representative runs and complete information on how these representative runs were executed for validation purposes;
(d) validation results including disposition of any discrepancies found; and
(e) discussion of the nature and extent of the user validation effort (applicable to user-prepared validation reports only).

10.2.7 Test Plans

A test plan, where required, shall contain the following items:
(a) identification of the computer program and its features to be tested;
(b) identification of the test items including their revision or version;
(c) identification of major test tasks;
(d) specification of pass/fail criteria;
(e) a description of the test environment;
(f) identification of responsibilities for various aspects of testing;
(g) a schedule and list of deliverables; and
(h) identification of applicable test procedures.

10.2.8 Test Procedures

A test procedure, where required, shall contain the following information:
(a) a description of the purpose;
(b) identification of any special requirements; and
(c) a description of the actions necessary for the test execution.

10.2.9 Test Reports

A test report is required to summarize the results of the designated testing activities. A test report, in addition to the items covered in Clause 8.3, shall contain the following:
(a) a unique report identifier;
(b) a summary of major testing activities and events;
(c) a summary of dispositions for the closed-out anomalies;
(d) a listing of items that are still open; and
(e) references to applicable test plans and test procedures.

10.2.10 Configuration Management Plan

Each organization shall develop a configuration management plan applicable to that portion of the computer program life cycle for which it is responsible. The contents of the configuration management plan shall include the following:
(a) definition of the procedures for controlling the input and withdrawal of computer program components from a given configuration and changes to a configuration;
(b) definition of the methods used for change control;
(c) definition of the methods necessary to maintain records that trace the evolution of a computer program and its associated documents;
(d) definition of the methods to preserve configuration components required to recover any previous configuration;

(e) definition of the methods used to unambiguously identify the various computer program components necessary to describe a given configuration at a given time in the computer program life cycle. These components may include the operating system, the programming language, any specific subprograms, the control commands needed to execute the program, the source modules, and object modules (if existing);
(f) identification of the documents related to each computer program component; and
(g) definition of the methods used to identify changes to a configuration component.

10.2.11 Change Control Plan

The changes to a computer program may be made either by a developer or by a user. The changes shall be processed in accordance with the change control plan. The change control plan shall address all the items covered in Clause 7.3.1.

10.3 Application Documents

Given below is a list of documents that shall be generated to guide users in installing, operating, and managing computer programs and conducting those aspects of maintenance which do not involve modification of the computer program source code. Depending on the complexity of the computer program, the contents of the individual documents may be modified and/or integrated into one or more documents.
(a) Computer Program Abstract;
(b) User's Manual;
(c) Theory Manual;
(d) Programmer's Manual; and
(e) Maintenance Manual.

Note: *Additional guidance and recommendations regarding the contents of these documents may be found by referring to the current editions of ANSI/IEEE Standard 1063, and ANSI/ANS Standard 10.3.*

10.3.1 Computer Program Abstract

The abstract provides a summary of the purpose, capabilities, operating environment, and limitations of the computer program. The abstract should include
(a) the computer program code name;
(b) the version identifier;
(c) a brief description of the problem solved;
(d) the method of solution;
(e) hardware requirements;
(f) system requirements; and
(g) capabilities, limitations, and restrictions.

10.3.2 User's Manual

The user's manual contains all the information necessary, with sufficient detail, to permit the effective execution of the computer program. This manual, in addition to the computer program's name and version identifier, shall include the following:
(a) instructions on installing and running the computer program;
(b) descriptions of features and capabilities;
(c) enumeration and specification of input and output data;
(d) errors and warning messages, their interpretation, and recommended corrective action;
(e) discussion of computational error sources;
(f) identification of values assigned to constants;
(g) description of capabilities, limitations, and restrictions;
(h) sample cases that illustrate the use of all components and modules of the computer program indicating the expected accuracy of outputs;

(i) standard forms for problem reports and enhancement requests; and
(j) detailed discussion of uncertainties due to error bands of empirical correlations.

10.3.3 Theory Manual

The theory manual describes in detail the theoretical and mathematical foundations of the computer program. The contents of the theory manual shall include all items covered in Clause 5.4.

10.3.4 Programmer's Manual

The programmer's manual shall provide information on
(a) programming considerations in translating the theory stated in the theory manual into workable computer programs;
(b) overall logic;
(c) programming philosophy;
(d) program structure including flowcharts;
(e) for computer programs developed in-house, information on how to modify and maintain the computer program;
(f) tools that must be used to aid programming and debugging;
(g) conventions to be adopted for variable naming, program headers, and code commentary;
(h) conventions to be adopted for identifying each component of a computer program in order to meet the requirements of the configuration management plan; and
(i) programming conventions that are to be used.

10.3.5 Maintenance Manual

The maintenance manual contains instructions for computer program support and maintenance and it shall address, as a minimum, the following items:
(a) hardware requirements;
(b) a description of self-checking capabilities;
(c) capabilities for accommodating expansions, alterations, or enhancements;
(d) capabilities to adapt to various hardware systems;
(e) documentation identifying modules that would be affected by potential changes for a specific application; and
(f) reference to error and warning messages described in the user's manual (see Clause 10.3.2).

Appendix A
Recommended Checklist to Confirm Good Programming Practice

Note: *This Appendix is not a mandatory part of this Standard.*

A1.

Documents defining standard and uniform programming practices, such as programmers' handbooks, should address the following issues:

(a) Program Organization
   (i) overall program structure, including
      (1) input data acquisition and processing;
      (2) checks and edits;
      (3) computation and data processing; and
      (4) final edits and saving of specific data.
   (ii) orderly progression of calculational flow;
   (iii) overall program control in a single module;
   (iv) a single, well defined function for each module;
   (v) a uniform module layout, including
      (1) module header;
      (2) data definition; and
      (3) programming segments.

(b) Programming Language
   (i) specification of standard programming language;
   (ii) adherence to an applicable programming Standard;
   (iii) limited use and thorough documentation of extensions to the standard programming language;
   (iv) use of assembly language only where absolutely necessary and thorough documentation of the programming logic.

(c) Data Management
   (i) documentation of data transfer techniques;
   (ii) use of one data transfer technique throughout the computer program unless a data structure requires special handling;
   (iii) use of one module to read or write a data file;
   (iv) documentation of data file content and structure.

(d) Computer Program Features
   (i) input data identification, preparation, and organization;
   (ii) initialization of arrays, variables, and default parameters;
   (iii) reproduction of input data for visual inspection;
   (iv) input and output error-checking, including array bounds;
   (v) testing of intermediate results;
   (vi) reports about calculational path, intermediate results, and calculation progress;
   (vii) adequacy of error-checking and associated warning and error messages;
   (viii) normal and abnormal computer program terminations;

(ix) output information formats and users' options.

(e) Source Statements and Variables

(i) specification statements for variables;
(ii) meaningful variable names;
(iii) use of comment statements;
(iv) reflection of program structure and techniques in comment statements;
(v) unique identification of comment and source statements;
(vi) single-purpose use of variables;
(vii) checking of parameter types;
(viii) arrays of fixed length.

(f) Hardware and Computer Program Dependencies

(i) independence from unique hardware or computer program features;
(ii) documentation of computer memory management during computer program execution;
(iii) impact of computer memory management on computer program portability.

(g) Good Programming Techniques

(i) selection of variable names to reflect scientific notation conventions and to take advantage of default variable types;
(ii) clear identification of range, beginning and end of program loops;
(iii) meaningful description of printed output;
(iv) programming to achieve definite branching decisions in the computer program;
(v) standards on the use of subscripted variables as subscripts;
(vi) avoidance of mixed mode operations;
(vii) standards for structuring arithmetic and logical expressions for clarity;
(viii) standards for tests to detect invalid or indefinite intermediate results;
(ix) standards on the use of nested constructs;
(x) independence of module interfaces from changes to individual functions;
(xi) recommendations on maximum module size;
(xii) limitation of module parameters to the minimum number required;
(xiii) standards for module entry and exit points;
(xiv) standards on computer program readability;
(xv) standards on efficient and reliable programming techniques.

Proposal for Change

To help our volunteer members to assess proposals to change requirements we recommend that each proposal for change be submitted in writing and identify the
(a) Standard number;
(b) Clause number;
(c) proposed wording of the Clause (requirement, test, or pass/fail criterion) using mandatory language and underlining those words changed from the existing Clause (if applicable); and
(d) rationale for the change, including all supporting data necessary to be considered.

The proposal should be submitted to the Standards Administrator at least one month prior to the next meeting of the Committee. It is CSA Committee practice that only those proposals sent out to members prior to a meeting can be the subject of discussion and action. This is to allow the members time to consider the proposal and to do any research they may feel necessary.

Date: ____ - ____ - ____
YY MM DD

To: The Standards Administrator of CSA Standard ______________________

From: ______________________

Affiliation: ______________________

Address: ______________________

____

Phone: ______________________ **Fax:** ______________________

Re: Request for an Amendment, Deletion, or Addition to Clause(s) ______________________

Proposed change:

(Use reverse and additional pages as required.)

Product Description Produit	Quantity Nombre		Price Prix		Sub-Total Total		Shipping (see chart) Frais de port (voir grille)		Sub-Total Total		GST* TPS*		PST** TVP**		Sub-Total Total		QST*** TVQ***		Sub-Total Total
		x		=		+		=		+		+		=		+		=	
		x		=		+		=		+		+		=		+		=	
		x		=		+		=		+		+		=		+		=	
		x		=		+		=		+		+		=		+		=	
		x		=		+		=		+		+		=		+		=	

Grand Total/Total global :

* Add GST to all Canadian Orders. (NOTE: The CE Code Handbook and all PLUS products are GST exempt.) Ajouter la TPS à toute commande passée au Canada. (NOTE: Le Guide explicatif du CCE et les répertoires sont exempts de TPS.)
** Add PST to all Electronic products placed in Alberta, British Columbia, Manitoba, New Brunswick or Ontario. Ajouter la TVP à toute commande de produits électroniques passée en Alberta, en Colombie-Britannique, au Manitoba, au Nouveau-Brunswick et en Ontario.
*** Add QST to all Quebec orders. (NOTE: The CE Code Handbook and all PLUS products are GST exempt.) Ajouter la TVQ à toute commande passée au Québec. (NOTE: Le Guide explicatif du CCE et les répertoires sont exempts de TVQ.)

SHIP TO • EXPÉDIER À

If billing address is different, please specify.
Si la facture doit être envoyée à une autre adresse, veuillez l'indiquer.

Name / Nom ______________________ Title / Titre ______________________

Organization / Entreprise ______________________

Address / Adresse ______________________

City / Ville ______________________ Prov / State / Prov / État ______________________

Country / Pays ______________________ Postal / Zip code / Code postal ______________________

Telephone / Téléphone ______________________ SM or CM NO. / N. MS ou MC* ______________________

* Note. Discount applicable only if sustaining or certification No. provided.
La remise est accordée uniquement si le n. de membre de soutien ou de certification est donné.

Shipping and Handling • Frais de port et d'emballage

Product Produit	Canada	United States États-Unis	Other Countries Autres pays
Printed Publications Publications imprimées	$4/copy 4 $/exemplaire	$6/copy 6 $/exemplaire	$10/copy 10 $/exemplaire
CD-ROM 2200 CD-ROM 2200	$60/year 60 $/année	$60/year 60 $/année	$200/year 200 $/année
Other Electronic products Autres produits électroniques	$15/copy 15 $/exemplaire	$15/copy 15 $/exemplaire	$60/copy 60 $/exemplaire

NOTES
(1) All printed publications are shipped by first class mail unless otherwise specified. All electronic products are shipped by courier.
(2) Any items shipped to the United States, air or ground, may be charged customs charges for clearance. These charges are passed on to the consignee and are based on the declared value of the shipment.

REMARQUES
(1) Sauf indication contraire, les publications imprimées sont envoyées par courrier de 1re classe, et les produits électroniques, par service de messagerie.
(2) Tout article expédié aux États-Unis, par transport aérien ou terrestre, peut faire l'objet de frais de douanes. Ces frais, le cas échéant, sont payables par le destinataire et sont basés sur la valeur déclarée de l'envoi.

METHOD OF PAYMENT • MODALITÉS DE PAIEMENT

All first-time orders are to be paid by cash, cheque or credit card.
Toute commande initiale doit être payée comptant, par chèque ou par carte de crédit.

1 ______________________
Purchase Order No. • Bon d'achat n.
(North America only / Amérique du Nord seulement)

2 ______________________
CSA Customer No. • N. de client CSA

3 ______________________
Payment enclosed $ • Montant inclus $
Make cheque payable to Canadian Standards Association. Le chèque doit être fait à l'ordre de l'Association canadienne de normalisation.

4 Charge to credit card indicated • Porter à mon compte
[ ] American Express [ ] Visa [ ] Master Card Expiration date / Date d'expiration ______________________
| | | | | | | | | | | | | | |
Carte n.

____
Card holder's name • Nom du titulaire ______________________ Signature

Order Form Bon de commande

Std

Association Activities

The Canadian Standards Association is a not-for-profit, independent, private sector organization that serves the public, governments, and business as a forum for national consensus in the development of standards, and offers them certification, testing, and related services. It is a membership Association open to any individual, company, or organization interested in standards activities.

The more than 1000 standards published by CSA are written, reviewed, and revised by over 7000 committee members, who represent users, producers, and regulatory authorities in all regions of Canada. In addition to these volunteers, some 2000 representatives from industry, labour, governments, and the public participate in the work of the Association through sustaining memberships. Approximately one-third of CSA's standards have been referenced into law by provincial and federal authorities.

Activities in the standards field cover a number of program areas: lifestyles and the environment, electrical/electronics, construction, energy, transportation/distribution, materials technology, business/production management systems, communications/information technology, and welding. These are all listed in our catalogue, which is available on request.

We welcome your comments and inquiries. Further information on standards programs may be obtained by writing to

The Director, Standards Programs
Standards Development
Canadian Standards Association
178 Rexdale Boulevard
Rexdale (Toronto), Ontario
M9W 1R3

A not-for-profit private sector organization
178 Rexdale Boulevard, Rexdale (Toronto), Ontario, Canada M9W 1R3
(416) 747-4000
Regional offices in Vancouver, Edmonton, Winnipeg, Montreal, Moncton

Printed in Canada