TABLE OF CONTENTS

1.0 PURPOSE, SCOPE, AND APPLICABILITY 1
2.0 KEY DEFINITIONS AND ACRONYMS....5
3.0 SSC SOFTWARE DESIGN REQUIREMENTS (INPUT)....5
4.0 SSC SOFTWARE DESIGN (OUTPUT) 7
4.1 General Software Design Requirements....7
4.2 Software System Architectural Design....9
4.3 Software Requirements Specification (SWRS) 9
4.4 Software Architectural Design 9
4.5 Software Detail Design....10
4.6 Software System Hazard Analysis and Mitigation....10
4.7 Software Design Traceability and Testability 11
4.8 Operations and Maintenance (O&M) Instructions 11
4.9 SOFT-GEN Requirements and Deliverables....11
5.0 SSC SOFTWARE COMPUTER PROGRAM LANGUAGE (CODE) 12
6.0 NON-SSC SOFTWARE DESIGN REQUIREMENTS (INPUT)....12
7.0 NON-SSC SOFTWARE DESIGN (OUTPUT) 14
8.0 NON-SSC SOFTWARE PROGRAM LANGUAGE (CODE) 14

REFERENCE: SOFTWARE DESIGN DOCUMENTATION EXAMPLE

Rev	Date	Description	POC	RM
0	06/23/16	Initial issue	Tobin Oruch, ES-DO	Mel Burnett, CENG-OFF

1.0 PURPOSE, SCOPE, AND APPLICABILITY

This section describes the “how, when, and who” for designing (developing) software. See Tables 21.3-1 and 21.3-2 for a summary of this section for SSC software and Non-SSC software respectively.

See Chapter 21 applicability in SOFT-GEN. In addition:

- This section does not apply to “simple and easily understood” software used in the design of SSCs if individually verified as described in SOFT-V&V.
- See SOFT-V&V for required verification and validation activities and deliverables for designed software.
- See SOFT-GEN for additional requirements that apply during design.

Table 21.3-1 SOFT-DESIGN Section Summary for SSC Software
(This table is a summary only and does not include all requirement details. See text for details.)

Activity No.	SQM Activity	ML[1]				Implementation Detail			Reference	
		1	2	3	4	How	When	Who[2, 3]	P1040 R9 Ref. Section	ESM Ch. 21 Ref. Section[4]
1	Develop software design requirements (input)	R	R	G	G	▪ Develop software design input deliverables: o SSC drawings (e.g., PFDs, P&IDs, SOOs, hardware/instrument location drawing, location drawing, instrument/device list). o SSC performance specification o SSC FDD or SDD o Other SSC design inputs/outputs as required	▪ Earliest practical time ▪ 60%, 90%, 100% SSC detail design ▪ Prior to software design as much as practical	▪ SSC DA (D, R, A) ▪ LANL SRLM (R, A) ▪ FDAR (R, A)	3.3.4.b	SOFT-DESIGN, 3
2	Design the software (output)	R	R	G	G	▪ Design the software (output) deliverables: o software specification (SWRS) o software design (SWDD) o software system hazard analysis and mitigation documentation (SWHA) o software design traceability documentation (SWTM) o operations and maintenance (O&M) instructions o SOFT-GEN deliverables o Other software design deliverables as required	▪ Earliest practical time ▪ If in scope of SSC design, software requirements (draft) at 30% design submittals, firmed up at 60%. ▪ If not in scope of SSC design, then in detail design submittals ▪ Prior to software coding as much as practical	▪ SD (D, R, A) ▪ SRLM (R, A) ▪ LANL SRLM (R, A)	3.3.4.b	SOFT-DESIGN, 4
3	Translate the software design into computer program language	R	R	G	G	▪ Develop computer program language (code): o data structures/files, o source code (where feasible), o executable code ▪ Provide computer program listings	▪ Earliest practical time ▪ Prior to final acceptance testing	▪ SD (D, R, A) ▪ SRLM (R, A) ▪ LANL SRLM (R, A)	3.3.4	SOFT-DESIGN, 5

Table 21.3-1 SOFT-DESIGN Section Summary for SSC Software
(This table is a summary only and does not include all requirement details. See text for details.)

Activity No.	SQM Activity	ML[1]				Implementation Detail			Reference	
		1	2	3	4	How	When	Who[2, 3]	P1040 R9 Ref. Section	ESM Ch. 21 Ref. Section[4]

[1] ML = Associated Management Level per Form 2033. R = Required. G = Required but graded.

[2] D = Develop or implement; R = Review; A = Approve.

[3] SSC DA = Design agency for SSC that is associated with the software); SRLM = Software Responsible Line Manager (prior to turnover to LANL). LANL SRLM = SRLM after turnover to LANL. FDAR = Facility Design Authority Representative.

[4] Ref. = ESM Chapter 21 section name and subsection number (e.g., section SOFT-GEN, subsection 3, Software Identification and Determination).

Table 21.3-2 SOFT-DESIGN Section Summary for Non-SSC Software
(This table is a summary only and does not include all requirement details. See text for details.)

Activity No.	SQM Activity	ML[1]				Implementation Detail			Reference	
		1	2	3	4	How	When	Who[2, 3]	P1040 R9 Ref. Section	ESM Ch. 21 Ref.[5]
1	Develop software design requirements (input)	R	R	G	G	▪ Develop system requirements (input) specification (SWYRS)	▪ Earliest practical time ▪ Prior to software design as much as practical	▪ SO (D, R, A) ▪ SRLM (R, A) ▪ FDAR (R, A)	3.3.4.b	SOFT-DESIGN, 6
2	Design the software (output)	R	R	G	G	▪ Design the software (output) deliverables: o software specification (SWRS) o software design (SWDD) o software system hazard analysis and mitigation documentation (SWHA) o software design traceability documentation (SWTM) o SOFT-GEN deliverables o Other software design deliverables as required o Computer model mathematical terms[4]	▪ Earliest practical time ▪ Prior to software coding as much as practical	▪ SD (D, R, A) ▪ SRLM (R, A)	3.3.4.b	SOFT-DESIGN, 7

Table 21.3-2 SOFT-DESIGN Section Summary for Non-SSC Software
(This table is a summary only and does not include all requirement details. See text for details.)

Activity No.	SQM Activity	ML[1]				Implementation Detail			Reference	
		1	2	3	4	How	When	Who[2, 3]	P1040 R9 Ref. Section	ESM Ch. 21 Ref.[5]
3	Translate the software design into computer program language (code)	R	R	G	G	▪ Develop computer program language (code): o data structures/files, o source code (where feasible), o executable code ▪ Provide computer program listings	▪ Earliest practical time ▪ Prior to final acceptance testing	▪ SD (D, R, A) ▪ SRLM (R, A)	3.3.4.b	SOFT-DESIGN, 8

[1] ML = Associated Management Level per Form 2033. R = Required. G = Required but graded.

[2] D = Develop or implement; R = Review; A = Approve.

[3] SO = Software Owner. SRLM = Software Responsible Line Manager. FDAR = Facility Design Authority Representative.

[4] Applicable to computer program models only.

[5] Ref. = ESM Chapter 21 section name and subsection number (e.g., SOFT-GEN, Subsection 3, Software Identification and Determination).

2.0 KEY DEFINITIONS AND ACRONYMS

See Chapter 21, SOFT-GEN for key definitions and acronyms.

3.0 SSC SOFTWARE DESIGN REQUIREMENTS (INPUT)

Define, control and verify the design. This section describes the minimum requirements, activities and deliverables for preparing SSC software design requirements (i.e., inputs). SSC software design requirements are the information provided to the SSC software designer to design the SSC software.

A. How.

1. SSC software design requirement (input) deliverables depend on the nature, complexity and associated ML of the software. Use Table 21.3-3, *SSC Software Design Requirement (Input) Deliverables* and direction from the LANL SRLM as a guide to define software design requirement deliverables.

2. Process and retain input documentation in accordance with the SRLM's governing document control and records management process. Ensure the correct SWID (obtained when completing Form 2033) is part of the documentation record number. See AP-341-402, *Engineering Document Management in Operating Facilities* for details.

Table 21.3-3 SSC Software Design Requirement (Input) Deliverables

No	Deliverable	Apply by ML[1]			
		1	2	3	4
01	SSC System Drawings (e.g., process flow diagrams (PFDs) process and instrumentation diagrams (P&IDs), Sequence of Operations (SOOs), hardware/instrument location drawing, network drawing[2], instrument/device list.)	R	R	G	G
02	SSC performance specification	R	R	G	G
03	SSC Facility Design Description (FDD) and/or System Design Description (SDD)	R	R	G	-
04	Other SSC Design inputs/outputs as required to support software design[3]	R	R	G	-

[1] R = As required by ESM Chapter 1, General, Section Z10 or other ESM chapters; G = Required but graded per ESM Chapter 1: General, Section Z10 or other ESM chapters. "-" = Not required.

[2] Level of detail as required by FDAR; e.g., a notional or "block" diagram of the network as a minimum, detailed technical drawing later.

[3] As determined by LANL SRLM; may include SSC hazard analyses, Requirements Criteria Document (RCD), identification of commercial grade dedication (CGD) critical characteristics etc.

3. SSC software design requirements (inputs) must:

   a. Be identified and documented using a systems engineering process (Ref. P341, *Facility Engineering Processes Manual* and ESM Chapter 20 (pending publication)).

   b. Be documented as part of the SSC design documents (e.g., Process and Instrumentation drawings (P&IDs) with Sequence of Operations (SOOs), system specifications, system design descriptions (SDDs)).

   c. Be based on upper tier performance and functional requirements.

d. Identify the operating system, function, interfaces, performance requirements, installation considerations, applicable SSC design inputs, critical characteristics (as applicable), and design constraints of the computer program.

e. Specify technical and software engineering requirements, including security features.

f. Identify applicable reference drawings, specifications, codes, standards, regulations, procedures, or instructions that establish test, inspection, and acceptance criteria.

g. Be commensurate with the risk of unauthorized use; address security requirements (e.g., vulnerability/cyber-security protections).

h. For ML-1 through ML-3 software, be traceable throughout the software life cycle.

i. For ML-1 through ML-3 software, be based on system/component hazard analyses that identify system/component risks and the means for controlling them.

j. The software design shall consider the computer program's Operating Environment (see Definitions in SOFT-GEN).[1]

k. Identify and address the user human machine interface (HMI) requirements. Factor in existing user operational protocols, conventions and methods by users/operators. As appropriate, specify prototypes/HMI screenshots for review and comment as part of the design deliverables.

l. Ensure software design requirements, as applicable, are consistent with SSC technical baseline documents. For information on SSC software technical baseline documents, see STD-342-100, *Engineering Standards Manual, Chapter 1, Section Z10, General*; AP-341-616, *Technical Baseline Change During Design;* and AP-341-405, *Identification and Control of Technical Baseline, Variances, Alternate Methods, and Clarifications in Operating Facilities*.

m. Ensure the design inputs and sources are identified and documented, specified on a timely basis, translated into design documents, and their selection review and approved.

n. Include user-level input/review of screen shots, prototypes, etc.

4. For guidance, see:

- IEEE Std 1233, *IEEE Guide for Developing System Requirements Specifications*, and
- ISO/IEC/IEEE 29148, *Systems and Software Engineering-Life Cycle Processes-Requirements Engineering*.

B. When. Develop software design requirements at the earliest practical time and as much as practical, prior to software design. As applicable, submit software requirements (draft) at 30% design submittals, firmed up at 60%.

[1] An integral part of software design is the design of a computer program that is part of an overall system. ASME NQA-1a-2009, Part II, Subpart 2.7-402

C. Who. For SSC software, the SSC Design Agent (or Design Agency [DA]) that develops the SSC detailed design also develops, reviews, and approves the SSC software design requirements. The LANL SRLM and FDAR review and approve the software requirements.

4.0 SSC SOFTWARE DESIGN (OUTPUT)

A. How.

4.1 General Software Design Requirements

A. Software design is an expanding and complex area. Accordingly, this section is limited to providing only a high-level description of the software design output requirements. For additional detail, see:

- DOE-STD-1195, *Design of Safety Significant Safety Instrumented Systems Used at a DOE Nonreactor Nuclear Facilities*
- ANSI/ISA 84.00-01, *Functional Safety: Safety Instrumented Systems for the Process Industry Sector*;
- ANSI/IEEE STD 7-4.3.2, *IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations,*
- IEEE Std 1016, *IEEE Standard for Information Technology – System Design – Software Design Descriptions*[2]
- IEEE Std 1016.1, *IEEE Guide to Software Design Descriptions*
- ESM Chapter 8: *Instrumentation and Controls*

B. Design software to meet the software requirements and deliverables of this section, SOFT-DESIGN, and other ESM chapters as applicable. Design deliverables may be combined as needed in one or more documents.

C. Use an accepted design methodology (see IEEE Std 1016.1).

D. As specified by the LANL SRLM for ML-1 and ML-2 software, require and provide software designer and computer programmers (coders) qualifications commensurate with the risk associated with the software.

E. Document the software design and the computational sequence necessary to meet the software requirements including as applicable: numerical methods, mathematical models, physical models, control flow, control logic, data flow, process flow, data structures, supporting calculations (Ref. AP-341-605, *Calculations*), software baseline, process structures, and the applicable relationships between data structures and process structures. Combine with the documentation of the software design requirements, or the computer program listings (see definitions) resulting from implementation of the design.

F. Ensure software design outputs, as applicable, are consistent with SSC technical baseline documents.

G. Software design (output) deliverables depend on the nature, complexity and associated ML of the software. See SOFT-DESIGN References for examples. Deliverables consist of drawings and/or other descriptive documents. They may be combined in one or more documents or provided separately. Software design deliverables may be divided into the following categories: (a) software design

[2] Flow diagrams, charts and/or pseudo-code should be provided.

documentation (system architecture, software architecture, software detail design collectively referred to as SWDD), (b) Software Requirements Specification (SWRS), (c) hazard analysis and mitigation documentation (SWHA), (d) software design traceability documentation (SWTM), (e) O&M documentation, (f) SOFT-GEN deliverables, and (g) other software deliverables.

Note: Verification and Validation (V&V) documentation, including test plans, interim test reports, and design review documentation, are typically produced concurrent with the design. See SOFT-V&V.

H. Provide associated software deliverables as required by other ESM chapters (e.g., Chapter 8 for instrumentation and control systems).

I. Provide design deliverables using Table 21.3-4, *Software Design (Output) Deliverables* as a guide, and direction from the LANL SRLM.

J. Ensure the design is defined, controlled and verified; the design interfaces are identified and controlled; design adequacy is verified by individuals other than those who designed the software. *Guidance: The eventual system engineer and/or users should work with the design/development team to ensure documentation etc. is done to an appropriate level commensurate with the ML of the software/SSC.*

K. Ensure design documentation and records include not only final design documents, such as drawings and specifications, and revision to those documents, but also documentation that identifies the important steps in the design process, including sources of design inputs.

Table 21.3-4 Software Design (Output) Deliverables

No	Design Deliverable Element	Apply by ML[1]			
		1	2	3	4
01	Software system architecture design	R	R	G	G
02	Software requirements specification (SWRS)	R	R	G	-
03	Software architecture design	R	R	G	G
04	Software detail design	R	R	G	-
05	Software system hazard analysis and mitigation docs. (SWHA)	R	R	G	-
06	Software design traceability documentation (e.g., SWTM)	R	R	G	-
07	Computer Program Listings, including the program (design product)	R	R	G	-
08	Operations and maintenance (O&M) instructions	R	R	R	R
09	SOFT-GEN deliverables[2]	See SOFT-GEN			
10	Other software design deliverables[3, 4]	See LANL SRLM and other ESM chapters.			

[1] R = Required; G = Required but graded; "-" = Not required.

[2] Includes software list, Form 2033, software data sheet, software baseline.

[3] As determined by LANL SRLM and/or as required in other ESM chapters (e.g., Human Machine Interface (HMI) graphical layouts/slides). See SOFT-V&V for associated V&V deliverables.

[4] See ESM Chapter 8 for software related deliverables (e.g., logic diagrams, loop diagrams, point list tables, etc.) specific to instrument and control (I&C) system designs.

L. Retain documentation in accordance with the governing records management process. For ES-Div Non-SSC software and SSC software, ensure the correct SWID

(obtained when completing Form 2033) is part of the documentation record number. See AP-341-402 for details.

4.2 Software System Architectural Design

A. Develop the software system architectural design. Allocate system-level requirements to hardware, software, and interfaces of the system.

B. For ML-1, ML-2, and some ML-3 systems where software operational complexity and lifecycle maintenance can be disadvantages, evaluate and as appropriate, employ designs that use no or minimal software.

C. Sufficiently detail the system architectural design such that a person technically qualified in the subject can verify and validate (V&V) that the design satisfies requirements without recourse to the Software Designer (SD).

D. Consider the following concepts, as applicable, in the software system architectural design.

1. **Isolation**: Critical elements should be separated from each other to preclude undefined and/or unintended interactions. For ML-1 through ML-3 software, safety systems should be separated from non-safety systems where possible. Barriers (e.g., separation of safety from non-safety modules) should be used to prevent non-safety functions from interfering with safety functions. Consider use of different platforms, power supplies, inputs/outputs to physically separate the safety from non-safety elements.

2. **Independence/Diversity**: For ML-1 and ML-2 software, independent, diverse systems should be employed. These are systems where the stimuli originate from and are handled by separate elements with different designs, independent hardware inputs and/or independent software modules.

   Note: Because different manufacturers may use the same processor or software, common mode failures may not be averted simply by acquiring from several manufacturers.

3. **Fail-Safe Design**: Design such that the system remains in a safe mode upon failure without compromising isolation features or other safety functions.

4. **Incompatibility/Longevity**: Design to ensure integrated hardware-software compatibility as much as possible throughout the life-cycle of the system.

4.3 Software Requirements Specification (SWRS)

A. For software system requirements that are allocated to software, further refine the requirements and develop software-level requirements. Document the software requirements. Software requirements are documented in a software requirements specification (SWRS). See IEEE Std 830, *IEEE Recommended Practice for Software Specifications* for guidance.

4.4 Software Architectural Design

A. Transform the software requirements into the software architectural design. The software architecture design specifies the structures of the software and the various software components, interactions and interfaces between the software components.

B. Include the high-level design for the external interfaces between the software components and other components within the system (hardware and human interface) and between the software and entities outside the system.

C. Sufficiently detail the software architectural design such that a person technically qualified in the subject can V&V the design without recourse to the SD.

D. Where appropriate and/or as required by the software requirements document, design software control functions that are performed incrementally rather than in a single step to reduce the potential that a single failure of a software element can cause an unsafe system state.

E. Where appropriate and/or as required by the software requirements document, design built-in fault detection and self-diagnostics that detect and report software faults and failures in a timely manner and allow actions to be taken to avoid an impact on the system's safe operation.

4.5 Software Detail Design

A. For ML-1 through ML-3 software, define the internals of each software component down to individual software modules that can be coded.

B. Specify/document the software coding standards and/or conventions.

C. As applicable, include data structures/databases and external interfaces.

D. Sufficiently detail the detailed software design to: (a) allow a person technically qualified in the subject to V&V the design without recourse to the SD; and (b) allow computer programmers to code the design without undue difficulty.

4.6 Software System Hazard Analysis and Mitigation

A. For ML-1 through ML-3 software, conduct and document a **software** hazard analysis at the system and component level to identify software risks and develop mitigating approaches for controlling them. Potential failures should be identified and evaluated for their consequences of failure and probability of occurrence. Some potential problems may include (1) complex or faulty algorithm, (2) lack of proper handling of incorrect data or error conditions, (3) buffer overflow, and (4) incorrect sequence of operations due to either logic or timing faults.

B. Ensure the software hazard analysis is consistent with the system safety documentation for the associated facility. See SBP111-1, *Facility Hazard Categorization and Documentation* for associated facility safety documentation.

C. For ML-1 and ML-2 software, perform and document the hazard analysis based on recognized consensus standards. See ESM Chapter 8, *Instrument & Controls* for application of hazard analysis/mitigation with respect to the following standards for SSC software:

- ANSI/Instrumentation, Systems, and Automation Society (ISA) S84.00.01, *Functional Safety: Safety Instrumented Systems for the Process industry Sector*; and,
- ANSI/IEEE Std 7-4.3.2, IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations).

D. *For ML-3 software, standard-based methods or less formal methods should be used using a graded approach (e.g., failure modes and effects analysis, fault-tree modeling, event-tree modeling, cause-consequence diagrams, hazard and operability analysis, and interface analysis).*

E. For ML-4 software, a hazard analysis is not required but may be performed.

F. *Multiple/common-cause failures should be evaluated in the hazard analysis.*

Note: Failure mode and effect analysis (FMEA) approaches, when used alone, do not address multiple failures/common-cause failures. (ANSI/ISA 84.00-01-2004-Part 1 and IEEE STD-7-4.3.2-2003 provide guidance.)

G. The hazard analysis and design must include analysis and possible problems with the computer program's operating environment (including security environment) and external and internal abnormal conditions and events that can affect the computer program.

H. In the software design documentation (SWDD), as part of the SWHA, or in a separate deliverable, provide documentation that shows how the consequences of hazards/problems are mitigated. *Mitigation strategies should be included for:*

1. *Software standard hazards (i.e., the basic hazards associated with the software or process);*
2. *Software system failures (i.e., the functionality written into the software itself to protect from failure); and*
3. *Software system overrides (i.e., the functionality written into the software to keep a user or other system from bypassing the safety features within the software item.)*

4.7 Software Design Traceability and Testability

A. For ML-1 and ML-2 software, document traceability from the software design requirements to the software design deliverables. For ML-3 software, a software traceability matrix (SWTM) is recommended to document traceability. Other methods however, such as tracing the requirement within the design deliverable are also acceptable. Requirements tools and/or databases may be used.

B. For large, complex ML-1 and ML-2 software, consider bi-directional traceability (tracing from the requirements to the design as well as from the design to the requirements) to identify functionality that may have been inadvertently added without requirement drivers.

C. Design the software and the associated SSC such that it can readily be tested against the requirements.

4.8 Operations and Maintenance (O&M) Instructions

A. Develop and provide operations and maintenance (O&M) instructions to use and maintain the computer program(s).

B. Sufficiently detail the O&M instructions to allow an individual trained in the use of such software to follow the instructions and use the program without undue difficulty and/or probability of misuse.

C. *For ML-1 and ML-2 software, O&M instructions should be developed using standard conventions for instructions (e.g., ANSI/IEEE Std 26514, Systems and Software Engineering – Requirements for Designers and Developers of User Documentation).*

4.9 SOFT-GEN Requirements and Deliverables

Satisfy software requirements and provide deliverables as described in SOFT-GEN.

A. When. Design software at the earliest practical time and as much as practical, prior to software coding. Unless the software design is deferred to subcontractors, submit software design deliverables with the 90% and 100% detail designs. If software design is deferred, then specify and submit software design deliverables as part of the required submittal process.

B. Who. The SD develops, reviews and approves the deliverables of this section including the architectural design, software specification, detail design, hazard analysis and mitigation documentation, and design traceability documentation; the SD, SRLM (prior to turnover to LANL), and LANL SRLM review and approve.

5.0 SSC SOFTWARE COMPUTER PROGRAM LANGUAGE (CODE)

A. How.

1. Translate the software design into computer program language (code).
2. For ML-1 through ML-3 software, use standards and/or conventions as approved in the software design (e.g., IEEE Std 1666, *IEEE Standard for Standard System C Language Reference*).
3. Develop source code suitable for compilation or translation.
4. Provide both source code and/or executable code as deliverables. If source code cannot be provided (e.g., for proprietary reasons), then provide objective evidence that the source code was V&V'd as per IEEE Std. 1012 and/or acceptable methods.
5. For ML-1 and ML-2 software, provide computer program listings (see definitions).
6. Code should be developed to sufficient detail and clarity to allow someone technically qualified in the computer programming language to review, understand, and verify it meets the software design requirements without recourse to the SD and/or coder.
7. For ML-1 and ML-2 code, ensure most code lines have explanatory notes/comments (pseudo code) to support future troubleshooting, bug fixes, and/or system modifications.
8. Employ configuration management (including computer program code labeling) and problem reporting/corrective management in accordance with SOFT-GEN.

B. When. Develop code at the earliest practical time and prior to final program acceptance testing.

C. Who. The SD develops, reviews and approves the software code; the SRLM reviews and approves the code.

6.0 NON-SSC SOFTWARE DESIGN REQUIREMENTS (INPUT)

Guidance: Unlike SSC software, Non-SSC software typically does not use upper level SSC design documents (e.g., P&IDs, SOOs, SDDs) for design input. Therefore, the design input is communicated through other documentation. A ***system*** *requirements specification (SWYRS) is used to convey upper level design inputs analogous to the upper-level SSC design documents. A software requirements specification (SWRS) is developed for software system requirements that are allocated to software. The SWRS further refines the requirements and develops software-level requirements.* For designs where software is

relatively low in complexity and when allowed by the LANL SRLM, the SWYRS and SWRS may be combined.

A. How.

1. Using a graded approach for ML1 through ML-4 associated software, develop a system requirements specification (SWYRS). The SWYRS is a document that communicates the requirements of the customer to the technical community who will further specify and build the software. See the following for SWYRS guidance:

   - ISO/IEC/IEEE 29148, *Systems and Software Engineering-Life Cycle Processes-Requirements Engineering*.

2. The SWYRS must:

   a. Be identified and documented using a systems engineering process (Ref. P341, *Facility Engineering Processes Manual* and ESM Chapter 20 [pending publication]).

   b. Identify the operating system and other aspects of Operating Environment, function, interfaces, performance requirements, installation considerations, applicable design inputs, and design constraints of the computer program.

   c. Specify technical and software engineering requirements, including security features.

   d. Identify applicable reference drawings, specifications, codes, standards, regulations, procedures, or instructions that establish test, inspection, and acceptance criteria.

   e. Be commensurate with the risk of unauthorized use; address security requirements (e.g., vulnerability/cyber-security protections).

   f. For ML-1 through ML-3 software, be traceable throughout the software life cycle.

   g. For ML-1 through ML-3 software, be based on system/component hazard analysis that identifies system/component risks and the means for controlling them.

B. When. Develop the SWYRS at the earliest practical time and where practical, prior to software design.

C. Who. The SO develops the SWYRS and the SO (e.g., with assistance of SD); SRLM and FDAR review and approve the SWYRS.

7.0 NON-SSC SOFTWARE DESIGN (OUTPUT)

A. How.

1. See SSC Software design Subsection 4 and note that SSC-related text and/or references do not apply. For example, the following are not applicable:

   a. Integration of design outputs with SSC technical baseline documents (*4.1.F)*

   b. Provide associated software deliverables or use associated software standards as required for SSCs as stated by other ESM chapters (*4.1.B and 4.6.C*).

2. *Non-SSC software can include computer models. Models are simplifications of the real world constructed to gain insights into select attributes of a particular physical, biological, economic, engineered, or social system.*[3] For computer model design, also include the following:

   a. A description of the conceptual model for the problem to be solved. Include assumptions, algorithms, relationships and data.

   b. The mathematical terms that describe the conceptual model. Include mathematical equations, boundary values, initial conditions, and modeling data as required. Include numerical solution techniques such as finite element and finite difference when equations describing the mathematical model cannot be solved analytically.

   c. *Guidance: Use ANSI/ANS-10.7, Non-Real-Time, High-Integrity Software for the Nuclear Industry – Development Requirements as a guide*.

B. When. Design software at the earliest practical time and as much as practical, prior to software coding.

C. Who. The SD develops, reviews, and approves the software design; the SO and SRLM review and approve the design.

8.0 NON-SSC SOFTWARE PROGRAM LANGUAGE (CODE)

See Subsection 5.

REFERENCE: SOFTWARE DESIGN DOCUMENTATION EXAMPLES

Examples initially provided are for informational purposes only and have not been reviewed for compliance to the requirements of this chapter. Better examples will be posted as they become available.

[3] EPA/100/K-09/003, *Guidance on the Development, Evaluation, and Application of Environmental Models.*