CONTENTS

RECORD OF REVISIONS

Rev.	Date	Description	POC	RM
0	06/23/2016	Initial issue	Tobin Oruch, *ES-DO*	Mel Burnett, *CENG-OFF*
1	05/25/2017	Added design output clarifications and other minor editorial clarifications	Tobin Oruch, *ES-DO*	Lawrence Goen, *ES-DO*
2	02/11/2026	Streamlined while maintaining most essential content, and merged SSC and Non-SSC requirements	Tobin Oruch, *ES-FE*	Michael Richardson, *ES-DO*

As with all Los Alamos National Laboratory (LANL) Engineering Standards, please contact the chapter point of contact (POC) (LANL only) with comments, issues, etc. For LANL, suggestions and questions may be entered in the Engineering Standards Manual (ESM) tool here.

SOFT-DESIGN: SOFTWARE DESIGN

1.0 INTRODUCTION

1.1 Purpose

SOFT-DESIGN describes Chapter 21 requirements for designing (developing) software.

1.2 Applicability

See Sections 1.0–2.0 of SOFT-INTRO for a description of the applicability of general Chapter 21 requirements.

1.3 Resources

Document examples and templates relevant to software design activities are provided on the Chapter 21 website under SOFT-DESIGN. They are for informational purposes only, may be used as a guide when creating documentation.

The documents may include the following:

- Form 2323, Software Requirements Specification (SRS) (docx)
- Software requirements specification (SRS) example (Source Tracker)
- Design document template and example for designed software (Source Tracker)
- User manual (O&M instructions) for designed software (Source Tracker)

Additional deliverables will generally be required.

1.4 Summary Tables

The following tables provide an overview of the requirements and deliverables of this Section. **They also contain details on who performs related tasks, and when.**

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]	ESM Ch. 21 Ref. Section[4]
1	Develop software design requirements (design input)	R	Gr	▪ Develop software design input deliverables: ○ SSC drawings (e.g., PFDs, P&IDs, SOOs, hardware/instrument location drawing, location drawing, instrument/device list) ○ SSC performance specification ○ FDD or SDD ○ Other SSC design inputs/outputs as required	▪ Earliest practical time ▪ 60%, 90%, 100% SSC design ▪ Prior to software design as much as practical	▪ SSC DA ▪ (D, R, A) LANL SRLM (R, A) ▪ FDAR (R, A)	SOFT-DESIGN, 3
2	Design the software (design output)	R	Gr	▪ Design the software (output) deliverables: ○ Software specification (SWRS) ○ Software design (SWDD) ○ Software system hazard analysis and mitigation documentation (SWHA; optional) ○ Software design traceability documentation (SWTM) ○ Operations and maintenance (O&M) instructions ○ SOFT-GEN deliverables ○ 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 ▪ Before software coding as much as practical	▪ SD (D, R, A) ▪ SRLM (R, A) ▪ LANL SRLM (R,A)	SOFT-DESIGN, 4
3	Translate the software design into computer program language	R	Gr	▪ Develop computer program language (code): ○ Data structures/files ○ Source code (where feasible) ○ Executable code ▪ Provide computer program listings (e.g., PDF printout of source code)	▪ Earliest practical time ▪ Before final acceptance testing	▪ SD (D, R, A) ▪ SRLM (R, A)	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]	ESM Ch. 21 Ref. Section[4]

[1] ML = associated management level per Form 2033; R = required; Gr = 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 (before 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]	ESM Ch. 21 Ref.[5]
1	Develop software design requirements (input)	R	Gr	▪ Develop system requirements (input) specification (SWYRS)	▪ Earliest practical time ▪ Before software design as much as practical	▪ SO (D, R, A) ▪ SRLM (R, A) ▪ FDAR (R, A)	SOFT-DESIGN, 3
2	Design the software (output)	R	Gr	▪ Design the software (output) deliverables: ○ Software specification (SWRS) ○ Software design (SWDD) ○ Software system hazard analysis and mitigation documentation (SWHA; optional) ○ Software design traceability documentation (SWTM) ○ O&M instructions ○ SOFT-GEN deliverables ○ Other software design deliverables as required ○ Computer model mathematical terms[4]	▪ Earliest practical time ▪ Before software coding as much as practical	▪ SD (D, R, A) ▪ SRLM (R, A)	SOFT-DESIGN, 4

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]	ESM Ch. 21 Ref.[5]
3	Translate the software design into computer program language (code)	R	Gr	▪ Develop computer program language (code): ○ Data structures/files ○ Source code (where feasible) ○ Executable code ▪ Provide computer program listings (e.g., PDF printout of source code)	▪ Earliest practical time ▪ Before final acceptance testing	▪ SD (D, R, A) ▪ SRLM (R, A)	SOFT-DESIGN, 5

[1] ML = associated management level per Form 2033; R = required; Gr = 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 DEFINITIONS AND ABBREVIATIONS

Appendix A of SOFT-INTRO addresses defined Chapter 21 terms.

3.0 SOFTWARE DESIGN INPUTS

This subsection describes the minimum requirements, activities, and deliverables for preparing software design requirements (i.e., inputs). Software design inputs are provided to the software designer. (Requirement 21-0401)

A. Software design requirement (input) deliverables depend on the nature, complexity, and associated ML of the software. Common software design input deliverables include the following:

- Software requirements (e.g., performance specification, software requirements specification [SRS], or requirement definition in the statement of work [SOW])
- System drawings (e.g., process flow diagrams [PFDs], process and instrumentation diagrams [P&IDs], sequences of operations [SOOs], hardware/instrument location drawing, network drawing[1], instrument/device list)
- Facility design description (FDD), system design description (SDD), or both
- Other design inputs as required to support software design[2]

B. Software design requirements (inputs) must

1. be identified and documented using a systems engineering process (Ref. P341, *Facility Engineering Processes Manual,* and ESM Chapter 20);
2. be based on upper-tier performance and functional requirements.
3. identify the operating system, function, interfaces, performance requirements, installation considerations, applicable SSC (structure, system, and component) design inputs, critical characteristics (as applicable), and design constraints of the computer program (Requirement 21-0402);
4. specify technical and software engineering requirements, including security features;
5. identify applicable reference drawings, specifications, codes, standards, regulations, procedures, or instructions that establish testing, inspection, and acceptance criteria;
6. be commensurate with the risk of unauthorized use and address security requirements (e.g., vulnerability/cyber-security protections; see SOFT-ACQUIRE §2.1); (Requirement 21-0403)
7. be traceable throughout the software life cycle (ML-1 through ML-3 software);
8. consider the computer program's operating environment (see definitions in SOFT-INTRO);
9. identify and address any user human-machine interface (HMI) requirements, factor in existing user operational protocols and the conventions and methods

[1] Level of detail as required by facility design authority representative (e.g., a notional or "block" diagram of the network at a minimum, which will later become a detailed technical drawing)
[2] As determined by the SO, SRLM, or both; may include hazard analyses, requirements criteria document (RCD), identification of commercial grade dedication (CGD) critical characteristics, etc.

employed by users/operators, and as appropriate, specify prototypes/HMI screenshots for review and comment as part of the design deliverables; and

10. include user-level input/review of screenshots, prototypes, etc., as applicable.

For guidance on software design inputs, 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.

Table 21.3-3 Software Design Input Deliverables

No.	Deliverable
01	Software requirements (e.g., performance specification, software requirements specification, or listing in SOW)
02	For SSC: system drawings (e.g., PFDs, P&IDs, SOOs, hardware/instrument location drawing, network drawing[1], instrument/device list)
03	For SSC: FDD, SDD, or both
04	Other design inputs/outputs as required to support software design[2]

Note: Required for all software ML-1 through -4 or other ESM chapters.
1. Level of detail as required by FDAR (e.g., a notional or "block" diagram of the network at a minimum, which will later become a detailed technical drawing)
2. As determined by SRLM; may include hazard analyses, RCD, identification of CGD critical characteristics, etc.

C. Additionally, SSC software design requirements (inputs) must

1. be documented as part of the SSC design documents (e.g., P&IDs, SOOs, system specifications, SDDs);
2. be based on system/component hazard analyses that identify system/component risks and the means for controlling them (for ML-1 through ML-3 software); and
3. be 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*.

D. Ensure the design inputs and sources are identified and documented, specified on a timely basis, and translated into design documents — and that their selection is reviewed and approved.

E. Finally, these requirements must call for the processing and retention of software design input documentation in accordance with the SRLM's governing document control and records management process. It is important to ensure that the software identification number is part of the documentation record number.

4.0 SOFTWARE DESIGN OUTPUTS

Design software at the earliest practical time and complete as much of the design as practical before 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 the software design deliverables as part of the required submittal process.

Deliverables and documentation must be reviewed during the software design process and are subject to software design verification requirements in accordance with Section 4.0 of SOFT-V&V.

4.1 General Software Design Requirements

This Subsection provides a high-level description of the software design output requirements. (Requirement 21-0404)

A. For ML-1 and ML-2 software, software designer and computer programmers (coders) must be provided with qualifications commensurate with the risk associated with the software.

B. The software design and the computational sequence necessary to meet the software requirements must be documented. This documentation should include (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.

C. Verification and validation (V&V) documentation, including test plans, interim test reports, and design review documentation, is typically produced concurrently with the design. See SOFT-V&V.

D. Deliverables consist of drawings, other descriptive documents, or both.

Typical design deliverables include the following:

- Software system architectural design[3]
- Software requirements specification (SWRS)[3]
- Software architectural design[3]
- Software detail design[3]
- Software system hazard analysis and mitigation documents (SWHA)[3]
- Software design traceability documentation (e.g., SWTM)[3]
- Computer program listings, including the program (design product)[3]
- O&M instructions
- SOFT-GEN deliverables[4]
- Other software design deliverables[5,6]

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

E. It is important to ensure that the design is defined, controlled, and verified; the design interfaces are identified and controlled; and design adequacy is verified by individuals other than those who designed the software. The eventual system engineer, users, or both should work with the design/development team to ensure documentation is completed to an appropriate level commensurate with the ML of the software.

[3] May be combined in one or more documents or provided separately. SWHA is not mandatory, but good practice; see SOFT-DESIGN Appendix A.
[4] Required and graded based on the nature, complexity, and associated ML per direction from the LANL SRLM.
[5] Includes software list, Form 2033, software data sheet, software baseline.
[6] As determined by LANL SRLM or as required in other ESM chapters (e.g., HMI graphical layouts/slides). See SOFT-V&V for associated V&V deliverables.

4.2 Software Requirements Specification

Refine software system requirements that are allocated to software into detailed software-level requirements. When complexity warrants, document these software-level requirements in a standalone SRS. Form 2323 is available for this purpose, and an SWRS document number for an SRS may be obtained from CoE (SharePoint). See Institute of Electrical and Electronics Engineers (IEEE) Std 830, *IEEE Recommended Practice for Software Specifications*, for guidance. An SRS is a valuable input to the V&V testing planning.

4.3 Software System Architectural Design

4.3.1 Applicability

This section deals primarily with the hardware and network architecture that the software integrates with and runs upon. As such, it is mainly applicable to SSC software, although non-SSC software that relies upon specialized hardware may be subject to these requirements and guidance. This section does not apply to SSC software that is designed to run upon existing hardware and that requires no hardware configuration changes.

4.3.2 Requirements

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

B. Sufficiently detail the system architectural design such that a person technically qualified in the subject can verify and validate that the design satisfies requirements without recourse to the designer.

C. For ML-1, ML-2, and some ML-3 systems for which software operational complexity and life cycle maintenance can be disadvantageous, evaluate and, as appropriate, employ designs that use no or minimal software.

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

1. Isolation: Separate critical elements from each other to preclude undefined 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, and inputs/outputs to physically separate the safety elements from non-safety elements.

2. Independence/diversity: For ML-1 and ML-2 applications, employ independent, diverse systems in which the stimuli originate from and are handled by separate elements with different designs, independent hardware inputs, independent software modules, or a combination of these.

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 the software system such that it remains in a safe mode upon failure without compromising isolation features or other safety functions. Measures to mitigate the consequences of problems, as identified through analysis, shall be an integral part of the design. These potential problems include external and internal abnormal conditions and events that can affect the computer program. SOFT-DESIGN Appendix A provides guidance on analyzing

software to help ensure the necessary safety functions are met. (Requirement 21-0405)

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

Table 21.3-4 Software Design Output Deliverables	
No.	Deliverable
01	Software system architectural design[2]
02	Software requirements specification[2]
03	Software architectural design[2]
04	Software detail design[2]
05	Software system hazard analysis and mitigation documents (SWHA; optional)[2]
06	Software design traceability documentation (e.g., SWTM)[2]
07	Computer program listings, including the program (design product)[2]
08	O&M instructions
09	SOFT-GEN deliverables[3]
10	Other software design deliverables[4, 5]

Note: Required for all software ML-1 through -4 or other ESM chapters.
[2] May be combined in one or more documents or provided separately.
[3] Required and graded based on the nature, complexity, and associated ML per direction from the LANL SRLM.
[4] Includes software list, Form 2033, software data sheet, software baseline.
[5] As determined by LANL SRLM or as required in other ESM chapters (e.g., HMI graphical layouts/slides). See SOFT-V&V for associated V&V deliverables.
[6] See ESM Chapter 8 for software-related deliverables (e.g., logic diagrams, loop diagrams, point list tables) specific to I&C system designs.

4.4 Software Architectural Design

A. Transform the software requirements into the software architectural design. The software architectural 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.

D. Where appropriate 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 would cause an unsafe system state.

E. Where appropriate 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 impact on the system's safe operation.

4.5 Software Detailed 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 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 and (b) allow computer programmers to code the design.

4.6 Software Design Traceability and Testability

Document traceability from the software design requirements to the software design deliverables. An SWTM may be used to document traceability. Other methods, however, such as tracing the requirement within the design deliverable, are also acceptable. Requirements tools and databases may be used.

4.7 Operations and Maintenance Instructions

Develop and provide O&M instructions for using and maintaining the computer program(s). See SOFT-MAINT for use and maintenance requirements and address these requirements in the O&M instructions, the software data sheet, or both.

5.0 SOFTWARE COMPUTER PROGRAM LANGUAGE (CODE)

A. Translate the software design into computer program language (code). (Requirement 21-0406)

B. For software, use standards or conventions as specified and described in the SOW or requirements.

C. Provide executable code as deliverables.

D. Provide a printout of the program source code listings.

E. Ensure code is developed with sufficient detail and clarity to allow someone technically qualified in the programming language to review it, understand it, and verify that it meets the software design requirements.

F. Ensure most code lines have explanatory notes/comments (pseudo code) to support future troubleshooting, bug fixes, and system modifications.

G. Employ configuration management and problem reporting/corrective management in accordance with SOFT-GEN.

6.0 APPENDIX

Appendix A, Software System Hazard Analysis and Mitigation (Guidance)

Appendix A. Software System Hazard Analysis and Mitigation (Guidance)

Appendix A provides guidance on analyzing software to help ensure the necessary safety functions are met.

A. For ML-1 through ML-3 software, consider conducting and documenting 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.)

If a standalone SWHA is desired, the document number may be assigned here.