IEEE Std 1228-1994

IEEE Standard for Software Safety Plans

Sponsor

Software Engineering Standards Committee of the IEEE Computer Society

Approved March 17, 1994

IEEE Standards Board

Abstract: The minimum acceptable requirements for the content of a software safety plan are established. This standard applies to the software safety plan used for the development, procurement, maintenance, and retirement of safety-critical software. This standard requires that the plan be prepared within the context of the system safety program. Only the safety aspects of the software are included. This standard does not contain special provisions required for software used in distributed systems or in parallel processors.

Keywords: safety-critical software, software safety plan, software safety program, safety requirements

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Print: ISBN 1-55937-496-9, SH94255
PDF: ISBN 0-7381-0419-1, SS94255

Introduction

(This introduction is not part of IEEE Std 1228-1994, IEEE Standard for Software Safety Plans.)

This standard describes the minimum acceptable requirements for the content of a software safety plan. This standard contains four clauses. Clause 1 discusses the application of the standard. Clause 2 lists references to other standards. Clause 3 provides a set of definitions and acronyms used in the standard. Clause 4 contains the required content of a software safety plan. In order to be in compliance with this standard, users of the standard shall adhere to clause 4. The informative annex discusses software safety analyses.

This standard was written for those who are responsible for defining, planning, implementing, or supporting software safety plans.

Participants in the working group were individually supported by their employers with travel expenses and working days. This support does not constitute or imply approval or endorsement of this standard.

This standard was developed by the Software Safety Plans Working Group consisting of the following members who attended two or more meetings, provided text, or submitted comments on two or more drafts of the standard.

Cynthia L. Wright, *Chair* **Anthony J. Zawilski,** *Vice chair*
Patricia Trellue, *Configuration manager*

Dick Bair
Leo Beltracchi
Mordechai Ben-Menachem
P. V. Bhansali
David Burrows
Betty Chao
John Cherviavsky
Tony Clark
Taz Daughtrey
David Dini
L. G. Egan
Alwyn Goodloe
Herb Hecht
John Horch
Grady Lee
Nancy Leveson
Stanley Levinson
Ben Livson
D. P. Mannering
Scott Mathews
John McHugh
Archibald McKinlay
Bonnie Melhart
Bret Michael
Tammy Pelnik
John Perry
Norman Schneidewind
David Schultz
Anita Shagnea
Peter Shilling
Edgar Sibley
Mel Smyre
Jack Spraul
Leslie Stepanek
V. K. Srivastava
Leonard Tripp
Ron Vaickavski
Delores Wallace
Chuck Weinstock

The following individuals also contributed to the development of the standard by attending one meeting or providing comments on one draft:

Paul Ammann
Jordan Anderson
Ron Berlack
Richard Blauw
Thomas Braudt
J. Brazendale
Fletcher Buckley
David Card
James Cardow
Geoff Cozens
Ron Daly
Paul Davis
C. G. Diderich
Audrey Dorofee
Caroline Evans
Peter Farrell-Vinay
Fred Freiburger
Al Friend
David Gelperin
Donna Grush
John Harauz
Albert Hoheb
Charles Horn
Frank Houston
Diane Jachinowski
Jon Jacky
George Kambic
Alasdair Kemp
Phil Keys
Thomas Kurihara
Victor Maggioli
John Matras
A. D. McGettrick
Rovert Metro
Judy Moore
Bahman Mostafazadeh
Melissa Murphy
Dennis Nickle
Mark Oliver
Tuncer Oren
Dave Peercy
Dev Raheja
Steven Rakitin
Juri Reinfelds
Heinz Rogen
Art Rubino
Leonard Russo
Damian Saccochio
Hans Schaefer
Robert Shillato
Richard Thayer
Jerry Thrasher
Dinos Tsagoes
David Vickers
Jim Widmyer
Eugene Wilhelm
Bill Wood
N. Yogaramanan
Janusz Zalewski
Peter Zoll

Contents

IEEE Standard for Software Safety Plans

1. Overview

1.1 Purpose

This standard establishes the minimum acceptable requirements for the content of a Software Safety Plan (also referred to as the Plan) to address the processes and activities intended to improve the safety of safety-critical software.

1.2 Scope

This standard applies to the Plan used for the development, procurement, maintenance, and retirement of safety-critical software; for example, software products whose failure could cause loss of life, serious harm, or have widespread negative social impact. This standard requires that the Plan be prepared within the context of the system safety program. The scope of this standard includes only the safety aspects of the software. This standard does not contain special provisions required for software used in distributed systems or in parallel processors.

1.3 Application

The Plan is prepared under the direction of project or system safety program management to address the identified potential software safety risks.

Compliance with this standard requires the creation of a written plan that addresses each topic, subtopic, and stipulation described in clause 4. The level of detail in, and the resources required by an software safety plan will be determined by factors including the type and level of risks associated with the software product, the complexity of the application, and external forces such as contractual requirements.

Software is a portion of a system. Other portions of that system include computer hardware, other devices (possibly including mechanical, electrical, chemical, or nuclear devices), and people. Software alone is not a safety issue; it is only an issue in the context of this larger system. Hence, software safety must begin with the larger system. Software safety must be considered in the context of its associated hardware, environment, and operators. The Plan needs to address interfaces with these elements.

The existence of this standard should not be construed to discourage or prohibit the imposition of additional or more stringent requirements where the need exists. An assessment should be made for the specific software project to ensure adequacy of coverage and safety assurance. Where this standard is invoked for a project engaged in producing several software products, the applicability of the standard should be specified for each of the software products encompassed by the project. This standard contains a minimum set of requirements for the content of software safety plans. The addition of more stringent requirements shall be the only acceptable tailoring process for this standard.

1.4 Disclaimer

Preparation of software safety plans according to this standard does not automatically ensure software safety. Compliance with this standard does not absolve the software designer, producer, or vendor from any statutory obligations.

2. References

This standard shall be used in conjunction with the following publications.The latest revisions shall apply.

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI).[1]

IEEE Std 730-1989, IEEE Standard for Software Quality Assurance Plans (ANSI).

IEEE Std 828-1990, IEEE Standard for Software Configuration Management Plans (ANSI).

IEEE Std 829-1983 (Reaff 1991), IEEE Standard for Software Test Documentation (ANSI).

IEEE Std 830-1993, IEEE Recommended Practice for Software Requirements Specifications (ANSI).

IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI).

IEEE Std 983-1986, IEEE Guide for Software Quality Assurance Planning (ANSI).[2]

IEEE Std 1008-1987, IEEE Standard for Software Unit Testing (ANSI).

IEEE Std 1012-1986, IEEE Standard for Software Verification and Validation Plans (ANSI).

IEEE Std 1016-1987, Recommended Practice for Software Design Descriptions (ANSI).

IEEE Std 1028-1988, IEEE Standard for Software Reviews and Audits (ANSI).

IEEE Std 1042-1987, IEEE Guide to Software Configuration Management (ANSI).

IEEE Std 1058.1-1987, IEEE Standard for Software Project Management Plans (ANSI).

IEEE Std 1063-1987, IEEE Standard for Software User Documentation (ANSI).

IEEE Std 1074-1991, IEEE Standard for Developing Software Life Cycle Processes.

[1]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

[2]IEEE Std 983-1986 has been withdrawn; however, copies can be obtained from the IEEE Standards Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

3. Definitions and abbreviations

3.1 Definitions

All definitions within this standard are consistent with IEEE Std 610.12-1990.[3] Terminology introduced within this standard includes the following:

3.1.1 accident: An unplanned event or series of events that results in death, injury, illness, environmental damage, or damage to or loss of equipment or property.

3.1.2 previously developed software: Software that has been produced prior to or independent of the project for which the Plan is prepared, including software that is obtained or purchased from outside sources.

3.1.3 risk: A measure that combines both the likelihood that a system hazard will cause an accident and the severity of that accident.

3.1.4 safety-critical software: Software that falls into one or more of the following categories:

a) Software whose inadvertent response to stimuli, failure to respond when required, response out-of-sequence, or response in combination with other responses can result in an accident.
b) Software that is intended to mitigate the result of an accident.
c) Software that is intended to recover from the result of an accident.

3.1.5 software hazard: A software condition that is a prerequisite to an accident.

3.1.6 software safety: Freedom from software hazards.

3.1.7 software safety program: A systematic approach to reducing software risks.

3.1.8 system hazard: A system condition that is a prerequisite to an accident.

3.1.9 system safety: Freedom from system hazards.

3.2 Abbreviations

The following appear within the text of this standard:

PHA	Preliminary Hazards Analysis
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
SVVP	Software Verification and Validation Plan
SVVR	Software Verification and Validation Report
V&V	Verification and Validation

[3]Information on references can be found in clause 2.

4. Contents of a software safety plan

In order to be in compliance with this standard, the contents of a software safety plan (also referred to as the Plan) shall include the sections shown in the following outline. An explanation of what each section should contain is in the subclauses following the outline. In a particular plan, if there is no information pertinent to a section or a required paragraph within a section, the following shall appear below the section or paragraph heading together with the appropriate reason for the exclusion:

This section/paragraph is not applicable to this Plan.

Additional sections may be added at the end of the Plan as required. If some of the material appears in other documents, reference to those documents shall be made in the body of the Plan.

1. Purpose
2. Definitions, acronyms and abbreviations, and references
3. Software safety management
   - 3.1 Organization and responsibilities
   - 3.2 Resources
   - 3.3 Staff qualification and training
   - 3.4 Software life cycle
   - 3.5 Documentation requirements
   - 3.6 Software safety program records
   - 3.7 Software configuration management activities
   - 3.8 Software quality assurance activities
   - 3.9 Software verification and validation activities
   - 3.10 Tool support and approval
   - 3.11 Previously developed or purchased software
   - 3.12 Subcontract management
   - 3.13 Process certification
4. Software safety analyses
   - 4.1 Software safety analyses preparation
   - 4.2 Software safety requirements analysis
   - 4.3 Software safety design analysis
   - 4.4 Software safety code analysis
   - 4.5 Software safety test analysis
   - 4.6 Software safety change analysis
5. Post development
   - 5.1 Training
   - 5.2 Deployment
     - 5.2.1 Installation
     - 5.2.1 Startup and transition
     - 5.2.3 Operations support
   - 5.3 Monitoring
   - 5.4 Maintenance
   - 5.5 Retirement and notification
6. Plan approval

Software safety plan outline

4.1 Purpose (Section 1 of the Plan)

This section of the Plan shall define the purpose and scope of the Plan, including those safety goals that are expected to be achieved by adherence to the Plan. The acceptable risks and safety objectives specific to the software project should be stated, including the factors used to determine acceptable risks and factors used to define the safety objectives.

4.2 Definitions, acronyms and abbreviations, and references (Section 2 of the Plan)

This section of the Plan shall list any acronyms, abbreviations, documents, standards, etc., used or referenced by the Plan. Definitions used in the Plan shall be consistent with IEEE Std 610.12-1990, except where redefinition results in a significant improvement in clarity for all users of the Plan. When definitions are not consistent with IEEE Std 610.12-1990, such definitions shall be provided in this section of the Plan.

4.3 Software safety management (Section 3 of the Plan)

This section of the Plan shall describe the organization, schedule, resources, responsibilities, tools, techniques, and methodologies used in the development of safety-critical software.

4.3.1 Organization and responsibilities (3.1 of the Plan)

This section of the Plan shall depict the software safety activities within the overall organization and shall describe organizational and functional relationships pertaining to software safety issues, lines of communication, and authority. The relationship of software safety program tasks to system safety program tasks, if present, shall be described. The relationship between other organizations having responsibility for tasks impacting software safety and the organization managing the software safety program shall be presented. Oversight, review, and approval authority of software safety program tasks shall be described in this section of the Plan. The authority of the software safety program management to enforce compliance with safety requirements and practices shall be described.

This section of the Plan shall identify one person to be responsible for the overall conduct of the software safety program. This individual and other personnel responsible for software safety activities shall have sufficient organizational autonomy and authority to ensure proper conduct of the software safety program. The accomplishment of software safety program activities may be performed by dedicated safety personnel, or may be integrated with and performed by personnel performing other activities in the normal course of development. Management of the software safety program includes responsibility to do the following:

a) Prepare the Plan
b) Obtain and allocate resources to ensure effective implementation of the Plan
c) Coordinate safety task planning with other organizational components or functions, such as development, system safety, software quality assurance, software reliability, software configuration management, V&V, and software testing
d) Coordinate software safety tasks within the overall context of the system safety program
e) Coordinate technical issues related to software safety with other components of the development and support organization, with the project sponsor, or with the customer
f) Ensure that adequate records are kept to document the conduct of software safety activities
g) Participate in audits of software safety plan implementation
h) Ensure training of safety and other appropriate personnel in methods, tools, and techniques used in software safety tasks

This section of the Plan shall define the mechanism for communication of safety concerns between project staff and personnel responsible for software safety.

4.3.2 Resources (3.2 of the Plan)

This section of the Plan shall specify resource requirements and the allocation of those resources to safety tasks. This section of the Plan shall describe how resource use shall be monitored during the software safety program implementation. Resources shall include, but not be limited to, financial, schedule, safety personnel, other personnel, computer and other equipment support, and tools.

4.3.3 Staff qualifications and training (3.3 of the Plan)

This section of the Plan shall specify the qualifications required for personnel who will perform, at a minimum, the following tasks:

a) Define safety requirements
b) Design and implement safety-critical portions of the system
c) Perform software safety analysis tasks
d) Test safety-critical features
e) Audit software safety plan implementation
f) Perform process certification

This section of the Plan shall define the on-going training requirements and the methods of accomplishing training objectives for personnel with software safety-related responsibilities.

4.3.4 Software life cycle (3.4 of the Plan)

This section of the Plan shall identify the software life cycle that will be used and specify the relationship among specific software safety tasks and the activities embodied in the organization’s chosen software life cycle. Further information relating to this activity may be found in IEEE Std 1074-1991.

4.3.5 Documentation requirements (3.5 of the Plan)

This section of the Plan shall specify the documents to be prepared and their contents. This section of the Plan shall specify the change and approval process for software safety-related portions of all project documents, including the Plan itself.

This section of the Plan shall specify whether the organization elects to prepare independent safety documents or to integrate the safety documentation with other project documents. Several IEEE Standards, such as IEEE Std 730-1989, IEEE Std 829-1983, and IEEE Std 1012-1986, identify documentation for both critical and noncritical software. Whichever form is chosen, this section of the Plan shall define the content of all such documentation. At a minimum, the following additional documentation requirements shall be satisfied for all safety-critical software:

a) *Software project management*. Documentation of how the Software Safety Program will be implemented, integrated, and managed with other development activities shall be prepared. Further guidance for this documentation may be found in IEEE Std 1058.1-1987.
b) *Software configuration management*. Information regarding the configuration management of software safety-related modules and documents shall be prepared (see 4.3.7).
c) *Software quality assurance*. Information regarding the quality assurance of software safety-related modules and documents shall be prepared (see 4.3.8).
d) *Software safety requirements*. Specification of safety requirements to be met by the software to avoid or control system hazards shall be prepared. Further guidance for this documentation may be found in IEEE Std 830-1993.
e) *Software safety design*. Descriptions of the software design elements that satisfy the software safety requirements shall be prepared. Further guidance for this documentation may be found in IEEE Std 1016-1987.

f) *Software development methodology, standards, practices, metrics, and conventions*. Approved and controlled practices that are essential to satisfy system and software safety objectives and requirements shall be specified. Further guidance for this documentation may be found in IEEE Std 730-1989 and IEEE Std 982.1-1988.

g) *Test documentation*. Software safety-related test planning, test design, test cases, test procedures, and test reports shall be prepared. Further guidance for this documentation may be found in IEEE Std 829-1983 and IEEE Std 1008-1987.

h) *Software verification and validation*. Information regarding how software safety will be verified and validated shall be prepared. The software safety-related analyses to be performed shall be specified. The method(s) to ensure the traceability of safety requirements to the specifications, implementation, and software safety-related test cases shall be specified. Further guidance for this documentation may be found in IEEE Std 1012-1986.

i) *Reporting safety verification and validation*. Information documenting the results of software safety-related verification and validation activities shall be recorded and reported. Further guidance for this documentation may be found in IEEE Std 1012-1986.

j) *Software user documentation*. Information that may be significant to the safe installation, use, maintenance, and/or retirement of the system shall be prepared. Further guidance for this documentation may be found in IEEE Std 1063-1987.

k) *Results of software safety requirements analysis*. See 4.4.2.

l) *Results of software safety design analysis*. See 4.4.3.

m) *Results of software safety code analysis*. See 4.4.4.

n) *Results of software safety test analysis*. See 4.4.5.

o) *Results of software safety change analysis*. See 4.4.6.

4.3.6 Software safety program records (3.6 of the Plan)

This section of the Plan shall identify the software safety program records to be generated, maintained, and retained by the project. This section of the Plan shall identify the persons responsible for the records generation, retention, and maintenance tasks. Software safety program records shall include the following:

a) Results of analyses, including V&V, performed on requirements, design, code, test, and other technical documentation
b) Information on suspected or confirmed safety problems in the prerelease or installed system
c) Results of audits performed on software safety program tasks
d) Results of safety tests conducted on all or any part of the entire system
e) A record of training provided to software safety program personnel
f) Results of any certifications performed

This section of the Plan shall specify how software safety program records are to be maintained. Evidence that the software safety program has been properly carried out shall be recorded during every phase of the software life cycle. This section of the Plan shall define initiation and completion criteria for software safety program tasks. The records shall be sufficient to certify that the processes and tasks specified in the Plan have been carried out satisfactorily.

This section of the Plan shall specify the tracking system used to ensure that hazards, their responsibility assignment, and their status can be tracked throughout the software life cycle through retirement. This section of the Plan shall specify:

a) The traceability requirements to be met by the tracking system
b) The tracking system to be used
c) The criteria to determine its applicability

4.3.7 Software configuration management activities (3.7 of the Plan)

This section of the Plan shall describe how the safety-critical software will be managed in accordance with an approved software configuration management plan. Software configuration management shall be in force during all phases of the software life cycle, from initiation of development through software retirement, and shall include control of project documentation, source code, object code, data, development tools, environments (both hardware and software), and test cases. For further information concerning this activity, see IEEE Std 828-1990.

This section of the Plan shall document the approved methods and/or tools that will be used for configuration control, access control, change control, and status reporting. Guidance on planning software configuration management practices can be found in IEEE Std 1042-1987. This section of the Plan shall specify the process by which changes to safety-critical software items are authorized and access granted to specific safety-critical configuration items for incorporation of approved changes (see 4.4.6).

This section of the Plan shall contain a description of the roles and responsibilities of safety personnel in the change evaluation, change approval, and change verification processes. The relationship between the configuration control board and other boards that may have software safety-related responsibilities shall be identified.

This section of the Plan shall describe the provisions for ensuring that configuration management meets the additional requirements necessary for safety-critical software for the following:

a) Software development tools
b) Previously developed software
c) Vendor-provided software
d) Subcontractor-developed software

4.3.8 Software quality assurance activities (3.8 of the Plan)

This section of the Plan shall describe the role of software quality assurance in ensuring proper performance of key software safety program activities. Guidance on planning software quality assurance and preparing software quality assurance plans can be found in IEEE Std 730-1989 and IEEE Std 983-1986. This section of the Plan shall include, at a minimum, descriptions of how

a) The Plan is prepared, approved, implemented, changed, and made consistent with predecessor documents
b) The technical recommendations resulting from software safety tasks are reviewed, considered by change control authority, and, where appropriate, implemented
c) The reviews and audits will address software safety concerns, requirements, guidelines, and process certification
d) The conduct of the software safety program will be monitored

4.3.9 Software verification and validation activities (3.9 of the Plan)

This section of the Plan shall specify the means by which the results of each life cycle activity will be matched against the system safety requirements and system hazard analysis to ensure that

a) All system safety requirements have been satisfied by the life cycle phases
b) No additional hazards have been introduced by the work done during the life cycle activity

4.3.10 Tool support and approval (3.10 of the Plan)

This section of the Plan shall specify the process to be used and the criteria to be applied in selecting, approving, and controlling tools, such as Computer-Aided Software Engineering (CASE) products, compilers, editors, fault tree generators, and test environments for hardware and software. In order to lessen the possibility of inadvertent introduction of software hazards by project tools, the following areas shall be addressed:

a) Tool approval for use on the project
b) Installation of upgrades to previously approved tools
c) Withdrawal of a previously approved tool
d) Identification of limitations that may be imposed on tool use

This section of the Plan shall describe how the possibility of inadvertent introduction of software hazards by project tools will be controlled.

4.3.11 Previously developed or purchased software (3.11 of the Plan)

This section of the Plan shall state the provisions for ensuring that previously developed or purchased software meets the requirements of the safety-critical application, shall define the role of software safety program personnel in the approval process, and shall define the approval authority required before previously developed or purchased software can be used. This section of the Plan shall describe the approval process for previously developed or purchased software that will be used in a system with safety-critical operations. At a minimum, the approval process shall

a) Determine the interfaces to and functionality of the previously developed or purchased software that will be used in safety-critical systems.
b) Identify relevant documents (e.g., product specifications, design documents, usage documents) that are available to the obtaining organization and determine their status.
c) Determine the conformance of the previously developed or purchased software to published specifications.
d) Identify the capabilities and limitations of the previously developed or purchased software with respect to the project's requirements.
e) Following an approved test plan, test the safety-critical features of the previously developed or purchased software independent of the project's software.
f) Following an approved test plan, test the safety-critical features of the previously developed or purchased software with the project's software.
g) Perform a risk assessment to determine if the use of the previously developed or purchased software will result in undertaking an unacceptable level of risk.

Only previously developed or purchased software that 1) can be adequately tested, 2) presents acceptable risk, or 3) remains safe in the context of its planned use shall be used in safety-critical software products. The inability to determine the level of risk present or the consequence of failure shall be sufficient justification for rejecting the use of the previously developed or purchased software.

This section of the Plan shall describe the equivalent analyses, tests, and demonstrations by the vendor to show the adequacy of the vendor-supplied software for use in a safety-critical application if such are used to satisfy the intent of steps a) through g).

This section of the Plan shall describe how the software safety change analysis shall be applied to all previously developed or purchased software (see 4.4.6).

4.3.12 Subcontract management (3.12 of the Plan)

This section of the Plan shall specify the provisions for ensuring that subcontractor software meets established software safety program requirements. Whenever the developer of safety-critical software employs the services of a subcontractor to modify or develop software that will be used in safety-critical situations, this section shall

a) Describe the methods for control of subcontractors insofar as it impacts the execution of the software safety programs
b) Describe the methods to be used
    1) Determine the capabilities of the subcontractor to support the software safety program requirements
    2) Evaluate and approve the subcontractor's software safety plan
    3) Monitor adherence to the requirements of the Plan
c) Describe the process used to assign responsibility for, and track the status of, hazards identified by or impacting the subcontractor

4.3.13 Process certification (3.13 of the Plan)

This section of the Plan shall specify the method to be used for certifying that the software product was produced in accordance with the processes specified in the Plan. This section of the Plan shall identify the person or persons responsible for performing process certification.

4.4 Software safety analyses (Section 4 of the Plan)

This section of the Plan shall require that certain software safety-related analyses be performed as part of the software development process. See the annex for a discussion of software safety analyses.

4.4.1 Software safety analyses preparation (4.1 of the Plan)

This section of the Plan shall define the software-safety related activities for obtaining the following:

a) A Preliminary Hazard Analysis (PHA) and any additional hazard analyses performed on the entire system or any portion of the system that identifies
    1) Hazardous system states
    2) Sequences of actions that can cause the system to enter a hazardous state
    3) Sequences of actions intended to return the system from a hazardous state to a nonhazardous state
    4) Actions intended to mitigate the consequences of accidents
b) A high-level system design identifying those functions that will be performed by software and specifying the software safety-related actions that will be required of the software to prevent the system from entering a hazardous state, or to move the system from a hazardous state to a nonhazardous state, or to mitigate the consequences of an accident
c) The interfaces between the software and the rest of the system

4.4.2 Software safety requirements analysis (4.2 of the Plan)

This section of the Plan shall define the software safety-related activities that will be carried out as part of the software requirement's phase of development. This section of the Plan shall specify the following:

a) The types of analyses that will be performed as part of the software safety requirements analysis, and when they will occur

b) How the results of these analyses will provide
    1) An identification of hazards and relevant associated software requirements
    2) The software safety design constraints and guidelines
    3) The software safety-related test requirements and inputs to the test planning process
    4) A list of required, encouraged, discouraged, and forbidden design, coding, and test techniques

c) The formal review and inspection requirements for the software safety requirements analysis (Further guidance on software reviews may be found in IEEE Std 1028-1988.)

4.4.3 Software safety design analysis (4.3 of the Plan)

This section of the Plan shall define the safety-critical activities that will be carried out as part of the software design phase of software development. This section of the Plan shall specify the following:

a) The methods by which safety-critical software design elements will be identified and classified

b) The types of analyses that will be performed on each safety-critical software design element as part of the software safety design analysis, and when they will occur

c) How the results of these analyses for each level of design will be documented. At a minimum, the documentation shall specify
    1) Design techniques and practices covering the partitioning of the software into design elements and the effect of these techniques and practices on analyses and tests for safety
    2) The relationship between system hazards and the software design elements that may affect or control these system hazards
    3) The classification of each software design element according to the methods specified in section 4.3 of the Plan
    4) An evaluation of the software architecture for supporting the system goals for redundancy and separation of safety-critical software functions
    5) An evaluation of compliance of the design with the system safety requirements
    6) Tests required for the integration of subsystems and systems that will demonstrate that the software supports the system safety goals
    7) Modifications to the list of required, encouraged, discouraged, and forbidden coding and test techniques

d) The formal review and inspection requirements for the software safety design analysis (Further guidance on software reviews may be found in IEEE Std 1028-1988.)

4.4.4 Software safety code analysis (4.4 of the Plan)

This section of the Plan shall define the software safety-related activities that will be carried out as part of the coding phase of software development. This section of the Plan shall specify the following:

a) The analyses that will be carried out on the software code (program), and when they will occur

b) How the results of these analyses will be documented. At a minimum, the documentation shall provide the following:
    1) Identification of the specific analyses performed on each software element and the rationale for these analyses
    2) Identification of the specific code analysis tools used and the rationale for the selection of those tools
    3) Recommendations for design and coding changes suggested by the analyses
    4) Detailed software safety-related test requirements suggested by the analyses
    5) An evaluation of the compliance of the code with the safety requirement
    6) Modifications to the list of required, encouraged, discouraged, and forbidden coding and test techniques

c) The formal review and inspection requirements for the software safety code analysis (Further guidance on software reviews may be found in IEEE Std 1028-1988.)

4.4.5 Software safety test analysis (4.5 of the Plan)

This section of the Plan shall define the software safety-related activities that will be carried out as part of the software testing phase of software development. This section of the Plan shall describe how the results of software testing will be used to show testing coverage for all software safety requirements. This section of the Plan shall specify the following:

a) The analyses that will be performed on the results of testing of software safety-critical design elements, and when they will occur
b) At a minimum, the analyses shall include
    1) The relationship between each software safety-related test and the software safety requirement that the test supports
    2) Evidence that can be used to determine whether or not each software safety requirement has been satisfactorily addressed by one or more software test
    3) An assessment of the risk associated with the implementation as indicated by the analyses of the software tests
    4) A recommendation as to whether or not adequate safety testing has been performed
c) The formal review and inspection requirements for the software safety test analysis (Further guidance on software reviews may be found in IEEE Std 1028-1988.)

4.4.6 Software safety change analysis (4.6 of the Plan)

This section of the Plan shall define the software safety-related activities that will be carried out in response to changes made in assumptions, specifications, requirements, design, code, equipment, test plans, environment, user documentation, and training materials. This section of the Plan shall specify the following:

a) The means for determining the impact of each change on safety
b) The techniques used to determine which safety-critical software design elements (if any) are affected by changes
c) The documentation to be revised to accurately reflect all software safety changes
d) The analyses that must be repeated whenever the system or its environment is modified
e) The extent to which regression testing is to be performed as a consequence of modifications to the system

4.5 Post development (Section 5 of the Plan)

This section of the Plan shall define the requirements for training, deployment, monitoring, maintenance, and retirement of safety-critical software that are necessary to ensure the continued safety of the system after its deployment and until its orderly retirement.

4.5.1 Training (5.1 of the Plan)

This section of the Plan shall specify the training requirements needed to ensure safe operation and use of the software within the overall system. These training requirements shall include safety training for the users, operators, maintenance personnel, and management personnel, as appropriate. If there is an existing system, training requirements for system startup shall be defined. If transitioning from an old to a new system, training requirements for this transition shall be defined. This section of the Plan shall describe how traceability shall be achieved between safety training requirements and the related safety training documents.

4.5.2 Deployment (5.2 of the Plan)

This section of the Plan shall define the installation, operations support, startup, and transition of the safety-critical software.

4.5.2.1 Installation (5.2.1 of the Plan)

This section of the Plan shall identify the requirements for the installation of the software developed under this Plan, including, if appropriate, any steps that should be taken to configure the software for a particular application. The section shall also identify all documents that must be completed in order to provide an accurate record of the installation process.

4.5.2.2 Startup and transition (5.2.2 of the Plan)

This section of the Plan shall address the requirements for safely starting the new system, and, if an old system is to be replaced, for making a safe transition from the old system to the new system. At a minimum, the following shall be addressed:

a) Fallback modes for the new system
b) Startup of backup components and subsystems
c) Startup of the new system
d) Parallel operation with backups
e) Parallel operation of the old system and the new system
f) Subsystem vs. full system operation
g) Switchover to full system operation
h) Validation of results from the new system
i) Cross validation of results between the old system and the new system
j) Fallback in the case of failure of the new system, including fallback to an old system if one exists

4.5.2.3 Operations support (5.2.3 of the Plan)

This section of the Plan shall identify all documentation or manuals that provide operations support and/or instructions necessary to ensure the safe operation of the system containing software developed under the Plan.

4.5.3 Monitoring (5.3 of the Plan)

This section of the Plan shall set forth the procedures for monitoring the operation of the safety-critical software within the system and shall identify procedures for documenting and reporting all safety concerns that are detected during its operation. This section of the Plan shall identify the procedures for verifying the integrity of the safety-critical software after its deployment. These procedures may include built-in and/or external measures for evaluating the software and its data or inspections to detect unauthorized modification of the software or its data. The criteria for the recall, mechanism for notification of the recall, and authorization procedures for the recall shall be defined in this section of the Plan.

4.5.4 Maintenance (5.4 of the Plan)

This section of the Plan shall describe any differences between the way software safety-related changes are made after development as compared to the change control activities to be used during development (see sections 4.5 and 4.6 of the Plan).

4.5.5 Retirement and notification (5.5 of the Plan)

This section of the Plan shall describe the activities to be performed for retiring any software developed under the Plan. The activities shall ensure that all users are notified of the change of status of the software.

4.6 Plan approval (Section 6 of the Plan)

This section of the Plan shall specify the formal review and inspection requirements for the Plan itself and the list of individuals who must approve the Plan.

Annex Discussion of software safety analyses

(informative)

Section 3.4 of the Plan requires the specification of the types of analyses that will be performed during the software life cycle. The information in this annex may be helpful in preparing section 3.4 of the Plan.

A.1 Software safety requirements analyses

The software safety requirements analysis evaluates software and interface requirements and identifies errors and deficiencies that could contribute to a hazard. It is the basis for subsequent software safety analyses. Analyses may include, but are not limited to, those listed below.

a) *Criticality analysis* identifies all software requirements that have safety implications. Each requirement in the Software Requirements Specification (SRS) is evaluated against the various system hazard analyses (including the PHA) to assess its potential for unacceptable risk. Each requirement in the SRS is evaluated against the system design to ensure that each safety-critical software requirement imposed by the system design is satisfied in the SRS. Requirements that satisfy either portion of this analysis are termed safety-critical requirements. A criticality level is assigned to each safety-critical requirement based on the estimated risk.
b) *Specification analysis* evaluates each safety-critical software requirement. This evaluation is with respect to a list of qualities, such as, completeness, correctness, consistency, testability, robustness, integrity, reliability, usability, flexibility, maintainability, portability, interoperability, accuracy, auditability, performance, internal instrumentation, security, and training. (Items are listed in no particular order.)
c) *Timing and sizing analysis* evaluates safety implications of safety-critical requirements that relate to execution time, clock time, and memory allocation. During requirements analysis, timing analysis identifies conditions, events, and time intervals that satisfy one or more of the following criteria:
    1) If Condition C becomes true, then Event A must occur within T seconds.
    2) If Condition C becomes true, then Event A must not occur until T seconds have elapsed.
    3) Event B must not occur until T seconds after Event A has occurred.

    A preliminary performance analysis may be performed.
d) *Different software system analyses* may be required if more than one software system is being integrated. Such integration will significantly expand the amount of analysis required in the software safety requirements analysis. Specific analysis of the allocation of software requirements to the separate systems can reduce subsequent integration and interface errors related to safety. This is particularly true when a system hazard results in a requirement that is partially implemented in two or more software systems.

A.2 Software safety design analysis

The software safety design analysis verifies that the safety-critical portion of the software design correctly implements the safety-critical requirements and introduces no new hazards. Analyses may include, but are not limited to, those listed below.

a) *Logic analysis* evaluates the safety-critical equations, algorithms, and control logic of the software design.
b) *Data analysis* evaluates the description and intended use of each data item in the software design. This analysis ensures that the structure and intended use of data will not result in a hazard. Data

structures should be assessed for data dependencies that circumvent isolation, partitioning, data aliasing, and fault containment issues affecting safety, and the control or mitigation of hazards.

c) *Interface analysis* verifies the proper design of a software component's safety-critical interfaces with other components of the system, both internal and external. The major areas of concern with interfaces are properly defined protocols and control and data linkages. External interfaces should be analyzed to demonstrate that communication protocols in the design are compatible with interfacing requirements. Hazards associated with an interface are also related to the system context and the environmental context as defined by their state at any point in time. The interface analysis must document this system and environment contexts. Interface analysis is also a tool that indicates the source of a system-level hazard and areas where further analyses are required.
d) *Constraint analysis* evaluates the safety of restrictions imposed on the selected design by the requirements and by real-world restrictions. The impacts of the environment on this analysis can include such items as the location and relation of clocks to circuit cards, the timing of a bus latch when using the longest safety-related timing to fetch data from the most remote circuit card, interrupts going unsatisfied due to a data flood at an input, and human reaction time.
e) *Functional analysis* ensures that each safety-critical software requirement is covered and that an appropriate criticality level is assigned to each software element.
f) *Software element analysis* examines software elements that are not designated safety-critical and ensures that these elements do not cause a hazard.
g) Based on the results of the *timing and sizing analysis* conducted for software safety requirements analysis, timing and sizing estimates can be established to allow evaluation of the operating environment.
h) Reliability predictions may be made for safety-critical software elements. Acceptable risk levels as defined in the software safety requirements may set reliability goals (see IEEE Std 982.1-1988 for additional information).

A.3 Software safety code analysis

The software safety code analysis verifies that the safety-critical portions of the design are correctly implemented in the code. Analyses may include, but are not limited to, those listed below.

a) *Logic analysis* evaluates the sequence of operations represented by the coded program and detects programming errors that might create hazards.
b) *Data analysis* evaluates the data structure and usage in the code to ensure each is defined and used properly by the program. Analysis of the data items used by the program is usually performed in conjunction with logic analysis.
c) *Interface analysis* ensures compatibility of program modules with each other and with external hardware and software.
d) *Constraint analysis* ensures that the program operates within the constraints imposed upon it by requirements, the design, and the target computer. Constraint analysis is designed to identify these limitations, to ensure that the program operates within them, and to ensure that all interfaces have been considered for out-of-sequence and erroneous inputs.
e) *Programming style analysis* ensures that all portions of the program follow approved programming guidelines.
f) *Noncritical code analysis* examines portions of the code that are not considered safety-critical code to ensure that they do not cause hazards. As a general rule, safety-critical code should be isolated from non-safety-critical code. The intent of this analysis is to prove that this isolation is complete and that interfaces between safety-critical code and non-safety-critical code do not create hazards. If isolation is not provable, the Plan should discuss how to handle the risk.
g) *Timing and sizing analysis* is further refined to ensure that no hazards due to timing or sizing factors have been added by the process of writing the code.

A.4 Software safety test analysis

Software safety test analysis demonstrates that safety requirements have been correctly implemented and the software functions safely within its specified environment. Tests may include, but are not limited to, the following:

a) Computer software unit level testing that demonstrates correct execution of critical software elements.
b) Interface testing that demonstrates that critical computer software units execute together as specified.
c) Computer software configuration item testing that demonstrates the execution of one or more system components.
d) System-level testing that demonstrates the software's performance within the overall system.
e) Stress testing that demonstrates the software will not cause hazards under abnormal circumstances, such as unexpected input values or overload conditions.
f) Regression testing that demonstrates changes made to the software did not introduce conditions for new hazards.

A.5 Software safety change analysis

The starting point of the change analysis is the safety-critical design elements that are affected directly or indirectly by the change. The purpose of software safety change analysis is to show that the change does not create a hazard, does not impact on a previously resolved hazard, does not make a currently existing hazard more severe, and does not adversely affect any safety-critical software design element.