ICS 35.080
CCS L 77

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35009—2025

复杂软件系统功能安全性技术要求

Technical requirements for functional safety of complex software systems

2025-09-12 发布　　2025-09-12 实施

中国指挥与控制学会　发布

目　　次

前　言

本文件按照 GB/T 1.1—2020《标准化工作导则 第 1 部分： 标准化文件的结构和起草规则》的规定 起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由中国指挥与控制学会提出并归口。

本文件起草单位：北京航空航天大学、杭州市北京航空航天大学国际创新研究院（北京航空航天大学国际创新学院）、可靠性与环境工程技术重点实验室、北京航空航天大学可靠性工程研究所、中国科学院声学研究所、中国电力科学研究院有限公司、中国船舶集团有限公司综合技术经济研究院、成都飞机设计研究所、中国航发控制系统研究所、中国航空工业集团公司西安飞行自动控制研究所、中国船舶集团有限公司系统工程研究院、中科工业人工智能研究院、中国科学院空间应用工程与技术中心、中国农业大学、华威大学。

本文件主要起草人：杨顺昆、曾福萍、佘志坤、张逸卓、郝程鹏、张立军、宋雁翔、陈伟静、朱烨、安冬、黄海驰、陶新、路丹、卫泽晨、徐鑫、王南洋、闫然、沈晓美、许兆伟、代国良、刘海亮、姚龙辉、方嫚、马欣瑞、张晓晨、王浩、马云云、赵星宇、吴梦丹、王英凡、周怡婧。

复杂软件系统功能安全性技术要求

1 范围

本文件规定了复杂软件系统在功能安全性方面的核心要素、定性要求、定量要求、支撑技术与方法以及生命周期阶段的过程与活动。

本文件适用于各类嵌入式软件、信息系统软件、工业控制软件等复杂软件系统安全相关功能的识别、开发和维护。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 11457—2006 信息技术 软件工程术语

GB/T 20438—2017 电气/电子/可编程电子安全相关系统的功能安全

GB/T 33750—2017 工业自动化系统与集成 安全相关系统的功能安全

GB/T 34590—2017 道路车辆 功能安全

GB/T 43698—2024 网络安全技术 软件供应链安全要求

GJB 900A—2012 装备安全性工作通用要求

GJB/Z 142—2004 军用软件安全性分析指南

GJB/Z 102A—2012 军用软件安全性设计指南

T/CSAC 004—2024 软件供应链安全要求测评方法

IEC 61508—2010 功能安全 工业过程中的电气/电子/可编程电子安全相关系统（functional safety of electrical/electronic/programmable electronic safety-related systems）

IEC 62061—2021 机械安全 安全相关电气、电子和可编程电子控制系统的功能安全（safety of machinery functional safety of safety-related control systems）

ISO 13849—1:2023 机械安全 控制系统的安全相关部件（safety of machinery safety-related parts of control systems）

ISO/TR 12489—2013 石油、石化及天然气工业 安全系统的可靠性建模与计算（petroleum, petrochemical and natural gas industries – reliability modelling and calculation of safety systems）

ISO/IEC/IEEE 29119 系统与软件工程 软件测试（systems and software engineering – software testing）

ISO/IEC TS 25058 系统和软件工程 人工智能系统质量评估指南（systems and software engineering guide for the evaluation of quality of AI systems）

NASA-HDBK—7009:2013 概率风险评估（PRA）手册（probabilistic risk assessment (PRA) handbook）

3 术语与定义

T/CICC 35008—2025和GB/T 11457—2006 确立的以及下列术语和定义适用于本文件。

3.1

复杂软件系统　complex software system

由大量相互依赖、相互作用的软件组件（计算机程序、模块、服务等）通过复杂的逻辑和物理关系连接而成，并遵循严格的规程（流程、协议、策略）进行协同运作，需动态应对内外部软件、硬件与环境的变化以实现复杂业务或关键领域目标的软件集成。其本质特征在于结构庞大、功能多样、环境多变、动态交互、任务复杂，通常具备多层次架构、模块协作、高度耦合以及较长的生命周期。

[来源: T/CICC 35008—2025, 3.1]

3.2

功能安全性　functional safety

安全相关的系统对输入的正确响应，以及在可预见的操作条件下，即使发生故障，也能在规定的时间内进入或保持安全状态。

3.3

软件模型　software model

用于描述、分析或验证软件功能和行为的抽象模型，支持功能安全设计与验证。

3.4

软件产品　software product

a） 指定交付给用户的计算机程序、规程和可能相关的文档和数据的完整集。

b） a）中的任一单独的项。

[来源: GB/T 11457—2006, 2.1520]

3.5

软件供应链　software supply chain

软件研发和运行过程中涉及的第三方库、编译器、操作系统、数据库和服务接口等外部依赖集合。

3.6

安全设计原则　safety design principle

在系统设计过程中应遵循的基本准则，用于预防、缓解或控制潜在风险。

3.7

故障预防与危险控制　fault prevention and hazard control

通过设计和过程手段避免故障的产生，并在故障发生时控制其危险发生扩散。

3.8

冗余架构　redundant architecture

通过增加多余的硬件或软件资源与各部分组件，提高系统的可靠性和安全性的设计方法。

3.9

安全降级　safe degradation

在故障发生时，将系统功能降级至较低但安全的运行模式。

3.10

可控性保障　controllability assurance

确保系统在异常或故障情况下，使用者或系统自身能够保持对运行状态的控制。

3.11

接口防护　interface protection

在系统接口层面采取的防护措施，用于防止不安全数据或异常交互影响系统安全。

4　缩略语

下列缩略语适用于本文件。

PRA	概率风险评价（Probabilistic Risk Assessment）
FC	故障覆盖率（Fault Coverage）
DC	诊断覆盖率（Diagnostic Coverage）
SAR	安全事故率（Software Accident Rate）
MTBA	平均事故间隔时间（Mean Time Between Accident）
SSR	软件安全可靠度（Software Safety Reliability）
SLR	软件出事率（Software Loss Rate）
MTBCF	严重故障平均间隔时间（Mean Time Between Critical Failures）
SIL	安全完整性等级（Safety Integrity Level）
SRF	安全冗余系数（Safety Redundancy Factor）
DMR	双模冗余（Dual Modular Redundancy）
TMR	三模冗余（Triple Modular Redundancy）
RPN	风险优先级数（Risk Priority Number）
STAMP	系统理论事故模型与过程（System-Theoretic Accident Model and Processes）
FMEA	失效模式及影响分析（Failure Modes and Effects Analysis）
FTA	故障树分析（Fault Tree Analysis）

5　复杂软件系统功能安全性的核心要素

5.1　数据

对软件系统中的安全关键输入、输出数据及关键的内部接口数据和状态数据进行功能安全性设计、实现、测试、运行与维护。

5.2　代码

依照GJB 900A—2012、GJB/Z 142—2004和GJB/Z 102A—2012，对软件系统源代码进行功能安全风险建模分析、设计防护与测试验证。

5.3　可执行文件

依照GB/T 20438—2017、GB/T 34590—2017、GJB 2786A—2009和GJB/Z 142—2004，对软件系统的配置文件、应用软件日志等关键重要可执行文件开展功能安全性风险分析。

5.4 软件模型

依照 ISO/IEC TS 25058和ISO/IEC/IEEE 29119，对于具有模型的软件系统，在软件模型层面开展形式化验证、故障注入、仿真测试等安全分析与测试验证。

5.5 软件供应链

依照GB/T 43698—2024和T/CSAC 004—2024，对于软件系统中调用的第三方库、编译器、操作系统、数据库、服务接口、软件产品等软件供应链开展必要的功能安全风险分析、测试验证与安全防护。

6 复杂软件系统功能安全性定性要求

6.1 软件系统安全设计原则

6.1.1 故障预防与危险控制

系统通过结构化设计、冗余和容差机制，以及运行防护策略，主动降低故障发生概率和危险，保障全生命周期安全。具体要求如下：

a） 系统应通过冗余设计、关键部件质量保障及容差设计，降低故障发生概率；
b） 系统应采取防护屏障、隔离装置及逻辑安全策略，防止危险发生；
c） 系统应建立全生命周期故障预防计划，包括设计、实现、测试、运行与维护。

6.1.2 防错、容错与纠错设计

通过防错、容错与纠错设计，减少错误发生和传播，并在错误出现后恢复系统功能，提高鲁棒性和可用性。具体要求如下：

a） 系统应采取防错措施，避免错误输入、操作或数据；
b） 系统应具备容错能力，故障发生时维持可接受功能；
c） 系统宜在检测到错误后具备纠错能力，能够恢复正确状态或数据。

6.1.3 冗余架构与数据备份

通过硬件、软件冗余和数据备份策略，降低单点故障风险，确保系统连续性和信息安全，并可在意外事件后快速恢复。具体要求如下：

a） 系统应根据安全等级采用硬件冗余策略（主动/热/冷备份）；
b） 系统应采用软件冗余或信息冗余策略，提高安全功能和数据可靠性；
c） 系统应建立数据备份机制，包括定期备份、增量备份及实时同步备份等，确保数据的可用性和完整性；
d） 系统应对备份数据完整性进行校验，确保数据恢复的可行性。

6.1.4 安全降级与可控性保障

在高安全风险环境下，系统可进入可控的安全降级模式，保障核心功能持续运行并防止二次事故。具体要求如下：

a） 系统应在严重故障时进入预设安全降级模式；
b） 降级过程应可控，避免二次事故；
c） 系统宜在设计阶段验证降级模式可靠性。

6.2 软件系统安全相关功能划分

6.2.1 安全相关功能识别与分级

系统必须系统性地识别所有可能影响安全的功能，包括硬件、软件及其交互接口。采用识别安全功能的方法，可以结构化识别潜在危险功能，并根据安全完整性等级进行分类。具体要求如下：

a） 系统应采用识别方法进行软件安全功能的识别；

b） 系统应对安全功能明确其安全完整性等级。

6.2.2 安全功能与非安全功能的隔离与防护机制

为防止非安全功能对安全功能产生干扰，必须在硬件和软件层面实施隔离措施，包括虚拟化及访问控制策略，以降低故障传播风险。具体要求如下：

a） 系统应通过物理隔离及通信链路实现硬件隔离；

b） 系统应通过虚拟化、操作系统安全域及访问控制实现软件隔离。

6.2.3 安全功能独立性与接口防护要求

安全功能必须在逻辑和物理上独立，接口必须具备防护能力，包括防止数据篡改、非法访问及潜在网络攻击，以保障系统整体安全。具体要求如下：

a） 安全功能应逻辑和物理独立，避免单点故障影响功能；

b） 接口应具备防注入、防越权访问及防数据篡改能力。

6.3 软件系统故障模式与影响分析要求

6.3.1 故障模式建模与分类方法

系统应采用标准化方法对故障模式建模和分类，覆盖硬件、软件及接口，确保分析结果可追溯、完整可靠。具体要求如下：

a） 系统应采用统一的术语、编码规则、分类框架进行故障编码和描述；

b） 故障建模应覆盖多维度，包括硬件、软件及通信链路。

6.3.2 故障影响分析范围、深度与精度要求

故障影响分析应覆盖组件级、子系统级及系统级，满足安全完整性等级要求。高等级功能需进行详细的故障路径分析和影响范围评估。具体要求如下：

a） 分析范围应覆盖组件级、子系统级及系统级；

b） 分析深度应符合安全完整性等级要求，等级高的安全功能实施更为详尽的分析。

6.3.3 故障传播路径抑制与隔离策略

通过防火墙、隔离网关、逻辑隔离和断路器等措施限制故障传播，提高系统鲁棒性和安全性。具体要求如下：

a） 系统应采用防火墙、隔离网关、断路器及逻辑隔离机制抑制故障传播；

b） 跨子系统接口应设计防扩散保护策略。

6.3.4 潜在危险源识别与风险映射技术

通过危险分析和失效模式分析识别潜在危险源，并建立系统化的风险映射，将危险源与系统功能及安全等级对应，为安全验证、测试及运行监控提供依据。具体要求如下：

a） 系统应识别潜在危险源并建立风险映射表；

b） 风险映射应作为安全验证和测试的依据。

6.4 软件系统安全风险分析

6.4.1 系统安全目标与基本属性

系统应在设计初期明确安全目标，并在全生命周期中保持一致性。具体要求如下：

a） 系统应定义安全属性及其适用边界；

b） 系统应明确安全目标与业务目标的协调关系。

6.4.2 定性风险评估与分级

通过基于专家经验、历史案例和场景推演的方法识别潜在风险，依照NASA-HDBK—7009:2013建立定性分级标准。具体要求如下：

a） 系统应通过场景化推演识别关键风险点；

b） 系统应采用定性分级方法对风险等级进行排序；

c） 系统宜结合历史数据与专家评审进行多方验证。

6.5 软件系统安全验证要求

6.5.1 安全性验证的定性方法与适用性分析

通过静态分析、专家审查和混沌测试等方法，验证软件在复杂环境下的安全性与稳定性。具体要求如下：

a） 软件应采用静态检查、逻辑推演等方法验证安全目标实现情况；

b） 软件应引入混沌测试手段，验证在扰动与异常场景下的鲁棒性；

c） 软件应根据不同安全等级，可选符号执行、混沌测试和模糊测试等验证方法进行验证。

6.6 软件系统运行与维护要求

6.6.1 软件运行阶段的安全性保障原则

运行过程中通过监测、隔离与防护手段，确保软件在复杂环境下仍可维持核心安全功能。具体要求如下：

a） 软件应建立定性运行安全指标并持续监测；

b） 软件应在运行异常时提供预警与人工干预通道。

6.6.2 运维风险控制策略

运维阶段应重点控制操作风险、升级风险和外部扰动风险，避免非计划停机和潜在安全隐患。具体要求如下：

a） 软件应在变更和升级前进行定性风险分析；

b） 软件应通过混沌实验验证运维策略的可行性；

c） 软件宜采用定性评估方法确保运维措施的稳健性。

6.6.3 生命周期内定性安全评估与改进机制

通过全生命周期的安全评估与迭代改进，实现持续提升软件安全性。具体要求如下：

a） 软件应定期开展定性风险评估与安全性验证；

b） 软件应结合历史故障与运行数据改进安全策略；

c） 软件应形成安全改进的闭环机制，确保长期有效性。

7 复杂软件系统功能安全性定量要求

7.1 软件系统安全相关失效概率要求

7.1.1 安全事故率

安全事故率（Software Accident Rate，SAR），在规定的条件下和规定的时间内，软件的事故总次数与寿命单位总数之比。事故率的概率度量亦称事故概率，计算见公式（1）。

$$P_A = \frac{N_a}{N_t} \quad \cdots\cdots (1)$$

其中：

P_A —— 安全事故率SAR;

N_a —— 事故总次数；

N_t —— 寿命单位总数。

7.1.2 平均事故间隔时间

平均事故间隔时间（Mean Time Between Accident，MTBA），平均事故间隔时间是安全事故率的倒数，表示两次软件事故之间的平均时间间隔，计算见公式（2）。

$$M_{TBA} = \frac{1}{P_A} \quad \cdots\cdots（2）$$

其中：

M_{TBA} —— 平均事故间隔时间 MTBA。

7.1.3 软件安全可靠度

软件安全可靠度（Software Safety Reliability，SSR）,在规定的条件下和规定的时间内软件执行任务过程中不出现安全性事故的概率。若从时间t=0 开始，软件执行任务未出现安全事故的持续时间为随机变量T，则软件安全可靠度为，计算见公式（3）。

$$S_{SR}(t) = P_A(T \geq t) \quad \cdots\cdots（3）$$

其中：

S_{SR} —— 软件安全可靠度 SSR。

7.1.4 软件出事率

软件出事率（Software Loss Rate，SLR）,软件出事率是软件安全可靠度的补余，表示在时间t 内出现至少一次安全事故的概率，计算见公式（4）。

$$S_{LR}(t) = 1 - S_{SR}(t) \quad \cdots\cdots（4）$$

其中：

S_{LR} —— 软件出事率 SLR。

7.1.5 严重故障平均间隔时间

严重故障平均间隔时间（Mean Time Between Critical Failures，MTBCF）是衡量软件或组件在出现严重故障（导致软件功能丧失或安全故障）之前平均工作时间的重要指标。它用于评估关键部件的安全性，指导维护计划及安全风险管理。

MTBCF 可用于：预测关键组件的维护周期、制定备件库存策略和作为安全完整性等级（SIL）及风险评估的参考依据。在安全关键应用中，应结合诊断覆盖率和故障分类标准进行修正，计算见公式（5）。

$$Y_{mf} = \frac{X_T}{X_t} \quad \cdots\cdots（5）$$

式中：

Y_{mf} —— 严重故障平均间隔时间；

X_T —— 软件总运行时间；

X_t —— 发生严重故障的次数。

7.2 软件系统安全冗余与容错定量要求

7.2.1 冗余安全计算

冗余是提高软件功能安全性的常用手段，通过设置多余的功能单元，确保部分单元失效时软件仍能运行。常见冗余模式包括双模热备份（Dual Modular Redundancy，DMR）和三模多数投票（Triple Modular Redundancy，TMR）。DMR 和 TMR 系统可靠度的计算分别见公式（6）和公式（7）。

$$R_{system} = 1 - (\lambda t)^2 \quad \cdots\cdots（6）$$

$$R_{system} = 3R^2 - 2R^3 \quad \cdots\cdots (7)$$

式中：

R_{system} —— 系统可靠度

λ —— 组件失效率；

t —— 系统考察的工作时间；

R —— 表示单个功能单元的可靠度。

7.2.2 故障覆盖率与诊断覆盖率

故障覆盖率（Fault Coverage，FC）与诊断覆盖率（Diagnostic Coverage，DC）用于评估冗余和容错措施的有效性。FC 是测试或诊断手段能够检测到的故障总数占所有可能故障总数的比例，面向的是所有故障。DC 是诊断机制能够检测到的危险故障占所有危险故障总数的比例，面向的是危险故障。FC 计算见公式（8），DC 计算见公式（9）。

$$F_C = \frac{N_D}{N_t} \quad \cdots\cdots (8)$$

式中：

F_C —— 故障覆盖率 FC；

N_D —— 被检测出的故障数；

N_t —— 所有可能的故障总数。

$$D_C = \frac{\lambda_{DD}}{\lambda_D} \quad \cdots\cdots (9)$$

式中：

D_C —— 诊断覆盖率 DC；

λ_{DD} —— 诊断功能可检测到的危险故障数；

λ_D —— 总的可能出现的危险故障数。

7.2.3 安全冗余系数

安全冗余系数（Safety Redundancy Factor，SRF）表示冗余软件可靠性相对单一软件的提升比例，计算见公式（10）。

$$S_{RF} = \frac{1}{1-D_C} \quad \cdots\cdots (10)$$

式中：

S_{RF} —— 安全冗余系数 SRF。

7.3 软件系统失效后果与风险指标

7.3.1 风险优先级数

风险优先级数（Risk Priority Number，RPN）结合事故严重度、发生概率和检测难度，对风险进行量化排序，计算见公式（11）。

$$P_{\mathrm{PN}} = S_{RPN} \times O_{RPN} \times D_{RPN} \quad \cdots\cdots (11)$$

式中：

P_{PN} —— 风险优先级数；

S_{RPN} —— 事故后果的严重度（通常 1–10，数值越大表示后果越严重）；

O_{RPN} —— 事故发生的发生概率（通常 1–10，数值越大表示越容易发生）；

D_{RPN} —— 事故被发现的检测难度（通常 1–10，数值越大表示越不容易被发现）。

7.3.2 软件系统安全裕度

软件系统安全裕度（Software Systems Safety Margin）是设计最大承载能力与实际使用载荷之比。该指标用于确保系统在极端条件下仍具有足够的安全余量，计算见公式（12）。

$$S_{SSM}=\frac{R_S}{S_S} \quad (12)$$

式中：

S_{SSM} —— 软件系统的裕度；

R_S —— 软件系统设计的最大承载能力；

S_S —— 软件系统的实际使用载荷。

7.3.3 概率风险评价

概率风险评价（Probabilistic Risk Assessment, PRA）是一种定量化的安全风险分析方法，用于评估危险发生概率及后果严重性。它通过事件树与故障树等模型，对事故演变过程和系统故障机理进行建模与量化。PRA最终提供风险评价和防范建议，为系统全寿命周期的安全决策提供支撑。实施流程如下：

a） 目标定义与系统分析：分析系统功能、结构及运行特性，建立系统模型；辨识可能故障类型与事故发展序列，形成系统事故清单；

b） 确定初因事件：利用主逻辑图确定或补充初因事件清单，并剔除非目标范围内的事件；

c） 初因事件分组：根据保护和缓解措施的相似性，将初因事件归类，以减少后续建模工作量；

d） 事故链建模：采用事件树描述系统对初因事件的响应与事故链发展过程；

e） 场景事件建模：在事件树分支点确定场景事件概率，必要时借助故障树展开并计算；

f） 数据采集与分析：收集系统基本信息、可靠性数据及事故报告，对数据精度和可信度进行评估；

g） 事故序列定量计算：通过事件树和故障树计算事故序列发生概率；

h） 故障树定量计算：利用最小割集方法计算顶事件概率；

i） 不确定性分析：采用蒙特卡罗模拟、模糊集或区间估计等方法，描述基本事件和顶事件概率的变动范围；

j） 风险影响因素分析：排序风险因素，识别关键改进环节；

k） 敏感性分析：研究模型参数变化对结果的影响，确定关键输入参数；

l） 结果输出：列出主要事故序列及发生频率，建立事故序列分布函数，提出防范建议。

7.4 软件系统安全完整性等级

7.4.1 安全完整性等级的定义

安全完整性等级（SIL）功能安全标准的核心评估指标。通常在软件开发初期概念阶段就要根据相关功能进行风险分析，从而确定SIL等级，以量化的方法平衡开发时的安全与成本。核心评估指标见表 1 所示。

表 1　SIL 核心评估指标

参数	等级划分示例	说明
严重性	S0（轻微伤害）- S3（致命伤害）	根据危险事件可能造成的人员伤害等级划分
暴露概率	E0（极低）- E3（高）	考虑软件使用频率、操作环境及用户暴露情况

核心评估指标的参数包括：

a） 危险事件严重性（Severity）如表 2 所示，危险事件严重性衡量危险事件对人员安全造成的伤害程度。严重性越高，对应的SIL等级要求通常越高。

表 2　严重度等级说明

	等级			
	S0	S1	S2	S3
描述	无伤害	轻度和中度伤害	严重的和危及生命的伤害	致命伤害

b） 危险事件暴露概率（Exposure）如表3所示，危险事件发生的暴露概率衡量软件或人员在危险条件下发生事故的频率或可能性。高暴露概率意味着软件更容易遭遇危险事件，需要更高SIL等级。

表 3　暴露概率等级说明

	等级			
	E0	E1	E2	E3
描述	非常低的概率	低概率	中等概率	高概率

7.4.2　SIL级别详细要求和评估方法

SIL是用于量化功能安全的一个重要指标。SIL从1到4，SIL等级越高，表示软件在规定的条件下运行安全相关的功能时的可靠性越高。

SIL 1是最低的SIL等级，要求较少的工程投入。SIL 4是最高的SIL等级，要求严格的设计和验证过程。

如表 4 所示，展示了每个SIL等级的详细要求和评估方法，具体风险降低要求可根据需求按照GB/T 33750-2017、IEC 61508和ISO 13849-1:2023等文件的相关内容进行调整：

表 4　SIL 等级说明

SIL等级	安全要求	描述	验证方法
SIL 1	低	检测到的危险事件概率低于10^{-2}至10^{-3}	预防性维护、周期性测试
SIL 2	中	检测到的危险事件概率低于10^{-3}至10^{-4}	软件测试、安全监控
SIL 3	高	检测到的危险事件概率低于10^{-4}至10^{-5}	独立的安全层、故障诊断
SIL 4	非常高	检测到的危险事件概率低于10^{-5}至10^{-6}	全面的冗余设计、严格的测试

7.4.3　SIL矩阵

应根据严重度等级、暴露概率等级创建一个如表 5 所示的矩阵。严重度等级和暴露概率等级分别构成矩阵二维坐标(行、列)中的一个，同时每一个矩阵元素为一个SIL等级。

表 5　SIL 矩阵

严重度等级	暴露概率等级			
	非常低的概率	低概率	中等概率	高概率
无伤害	SIL 1	SIL 1	SIL 2	SIL 2
轻度伤害	SIL 1	SIL 2	SIL 2	SIL 3
严重伤害	SIL 2	SIL 2	SIL 3	SIL 3
致命伤害	SIL 2	SIL 3	SIL 3	SIL 4

8　复杂软件系统功能安全性支撑技术与方法

8.1　风险模型构建

风险模型构建是基于软件功能需求和运行环境的量化分析技术，采用形式化建模方法对软件系统潜在功能失效模式、安全威胁及风险因子进行结构化表示。实现可计算、可追踪的风险结构，为功能安全设计和验证提供定量化输入。

适用对象：

a） 安全关键类型的复杂软件系统早期软件需求阶段；

b） 需为后续安全分析提供输入的软件系统。

实施步骤：

a） 收集软件系统运行场景与边界条件；

b） 准备软件系统需求规格说明书、运行环境分析报告、历史故障数据等；

c） 使用建模工具建立风险模型，生成模型文档（结构图、参数表、假设条件）等；

d） 输出初步风险评估报告。

实施要点：

a） 模型覆盖全部关键功能与运行场景；

b） 所有假设条件必须有可追溯来源；

c） 模型结果需经安全评审确认有效。

8.2　风险模型再评估

风险模型再评估采用迭代修正与结构优化技术，通过集成运行数据、历史事件和最新安全信息，对已有风险模型进行参数更新与耦合关系调整。保证模型在软件系统全生命周期中持续反映实际功能安全风险。

适用对象：

安全风险显著动态变化的复杂软件系统。

实施步骤：

a） 维护风险模型与历史数据库；

b） 收集运行反馈数据与最新风险事件记录；

c） 修正风险模型参数，更新模型结构；

d） 形成更新后的风险模型与更新风险评估。

实施要点：

a） 模型版本与数据版本保持匹配；

b） 更新原因与影响分析记录完整。

8.3 风险知识图谱

风险知识图谱采用图数据库和语义建模技术，将复杂软件中的风险元素、失效模式、威胁事件与控制措施建立多维语义关联的方法。该方法适合大型复杂工程中多团队协作下的功能安全管理，可通过图查询、路径分析和影响评估实现软件系统级风险识别与决策支持。

适用对象：

a） 需进行长期安全数据管理和多方协作的软件系统；

b） 需符合行业风险分类标准与数据接口规范的软件系统。

实施步骤：

a） 建立统一的风险术语与数据格式；

b） 准备风险分类体系、历史风险事件数据库、控制措施清单；

c） 构建风险知识图谱文件；

d） 开展风险关联分析并形成报告。

实施要点：

a） 知识图谱节点与边定义完整、准确；

b） 支持风险推理和可视化展示；

c） 符合数据更新与版本管理要求。

8.4 安全需求可追踪性矩阵

安全需求可追踪性矩阵利用矩阵化建模与自动化工具，将功能安全需求与设计元素、实现代码及测试用例建立映射关系，实现软件系统全生命周期的追踪和验证。确保功能安全要求落实到每个设计与测试环节，并支持定期审查与报告生成，为功能安全验证提供量化依据。

适用对象：

a） 安全需求完整、明确且可量化的软件系统；

b） 需对安全需求落实情况进行持续追踪与验证的软件系统。

实施步骤：

a） 收集安全需求规格说明书、软件系统设计文档、测试用例列表；

b） 构建需求可追踪性矩阵文档；

c） 输出需求覆盖差异分析报告。

实施要点：

a） 矩阵覆盖率达到 100%，每条需求对应至少一个设计元素和测试用例；

b） 所有差异和缺失均有记录与处置方案。

8.5 容错设计与冗余建模

容错设计与冗余建模采用软件系统建模与分析方法，将冗余机制（如双机热备、N+1 架构、多版本冗余）集成到功能模块与接口设计中。通过模拟和分析确保安全关键功能在组件或子系统失效情况下的持续可用性和功能安全性。

适用对象：

a） 安全关键和高可靠性软件系统；

b） 支持容错与冗余机制的软件系统架构。

实施步骤：

a） 收集软件系统安全关键功能需求、模块划分与接口设计；

b） 利用冗余策略库进行建模；

c） 输出冗余设计方案与容错分析报告。

实施要点：

a） 冗余方案覆盖所有关键功能模块；

b） 容错分析结果达到预定安全完整性等级。

8.6 功能共振方法

功能共振方法基于功能交互建模技术，对软件系统功能偏差及其叠加效应进行软件级分析。通过提出缓解措施如功能隔离、时间同步、异常检测和冗余设计，以保障复杂功能交互下的软件安全性。

适用对象：

a） 功能交互复杂难以覆盖的软件；

b） 软件功能存在明显的时序依赖或资源共享关系；

c） 需评估功能偏差叠加效应的安全关键场景。

实施步骤：

a） 识别软件功能，建立功能清单并描述输入、输出、前提条件、资源、控制和时间约束；

b） 分析各功能可能的偏差与变异来源；

c） 构建功能间交互和依赖关系模型，形成可分析的功能交互结构；

d） 识别可能导致危险状态的功能共振路径；

e） 提出缓解措施，如功能隔离、时间同步、异常检测与冗余设计；

f） 通过仿真或故障注入验证分析结果，并迭代修正模型。

实施要点：

a） 功能建模应覆盖关键软件功能及其交互关系；

b） 分析应突出偏差叠加效应，而非单点失效；

c） 风险场景均需形成可视化共振路径图；

d） 缓解措施与风险路径需双向验证有效性。

8.7 STAMP 分析方法

STAMP 基于软件理论，将安全问题视为控制环失效，适合分析软件密集型复杂软件中的交互风险，并为功能安全设计优化提供改进建议。

适用对象：

a） 高复杂度的控制软件；

b） 可结合安全需求建模工具的场景。

实施步骤：

a） 准备控制结构图、功能需求说明书、安全约束列表；

b） 建立控制结构模型并开展分析；

c） 输出 STAMP 分析报告与改进建议。

实施要点：

a） 覆盖所有关键安全路径；

b） 识别并处置不满足安全约束的设计。

8.8 FMEA 方法

FMEA 通过系统性地识别宏观设计元素的潜在功能故障模式，分析其后果对系统、用户、环境的影响，并评估其严重度、发生度和难检度，从而主动发现架构性缺陷与单点故障。

适用对象：

a） 需系统性识别潜在失效模式的软件或硬件系统；
b） 适合单个功能或部件的局部风险分析；
c） 需对风险进行优先级排序的场景。
实施步骤：
a） 定义系统或功能的分析边界；
b） 逐项识别潜在失效模式、后果及原因；
c） 评估严重度、发生概率和检测难度，计算 RPN；
d） 提出风险降低措施并修订 RPN；
e） 形成 FMEA 表格与改进建议。
实施要点：
a） 确保失效模式覆盖所有功能路径和关键部件；
b） 严重度评估应与系统安全目标一致；
c） RPN 计算应统一分级标准，避免主观偏差；
d） 改进措施需落实到责任人和验证环节。

8.9 FTA 方法

FTA 是一种自顶向下的逻辑推演方法，通过构建故障树，用逻辑门描述顶事件与底事件之间的关系，系统性分析事故或危险状态的成因。
适用对象：
a） 适合分析重大事故、顶层危险事件的原因；
b） 适用于功能安全目标验证与系统级安全论证；
c） 适合多因耦合和复杂逻辑条件下的风险建模。
实施步骤：
a） 明确顶事件（系统级失效或危险状态）；
b） 逐层分解，建立故障树逻辑结构（与门、或门、非门等）；
c） 识别底事件并收集相关概率数据；
d） 进行定性分析（最小割集、重要度排序）；
e） 进行定量分析（顶事件概率评估）；
f） 提出改进措施并重新评估。
实施要点：
a） 顶事件定义需明确且与安全目标一致；
b） 逻辑分解应完整，不得遗漏关键路径；
c） 输入数据需准确，概率假设应可追溯；
d） 结果应包括定性结论（关键路径）与定量结果（失效概率）。

8.10 形式化安全分析方法

形式化安全分析方法采用数学逻辑、模型检验与定理证明技术，将软件系统需求和设计转化为形式化规格说明。通过符号推理、模型验证、逻辑一致性检查等手段，实现对功能安全属性的严格验证，确保软件系统高安全完整性等级（如 SIL3/4）满足规范要求，消除自然语言歧义和潜在设计缺陷。
适用对象：
a） SIL3/4 等高安全完整性等级软件系统；
b） 需求、设计与代码可追踪的软件系统，便于形式化验证结果映射到实现和测试。
实施步骤：

a） 将需求与设计文档转化为形式化规格；

b） 准备软件系统设计模型；

c） 使用工具进行形式化验证，输出报告和缺陷清单。

实施要点：

a） 验证覆盖率达到目标；

b） 解决所有逻辑不一致性。

8.11 初步危险分析

通过系统化的方法，初步识别系统中潜在的危险、危险情境及其可能导致的严重后果，从而在设计源头规避重大安全风险，并为后续详细的安全性分析提供输入和焦点。

适用对象：

适用于所有安全关键或任务关键软件系统在概念设计或初步设计阶段的早期安全性分析。

实施步骤：

a） 该技术应按照如下步骤实施：

b） 确定分析范围，明确系统的设计意图、功能、边界条件、运行模式及预期的使用环境；

c） 基于经验、历史数据、标准规范或类似系统信息，采用头脑风暴、检查表等方法，全面识别系统中可能存在的能量源、有毒物质、辐射源等固有危险源及其可能导致的危险状态；

d） 分析每种危险状态触发或发生的可能原因，包括硬件故障、软件错误、人为操作失误、环境因素及它们的组合；

e） 评估每种危险状态一旦发生可能导致的后果严重性等级；

f） 结合原因发生的可能性，对已识别的危险进行初步风险评估与分级，并形成初步危险列表；

g） 针对中、高风险提出初步的风险控制措施建议，并将措施反馈至设计过程。

实施要点：

a） 在实施过程中，应注意如下要点：

b） 初步危险分析是一项迭代性工作，应随设计方案的细化而不断更新和深化；

c） 应着重于“什么是危险的”以及“后果有多严重”，而非深入分析具体的故障模式或概率，其核心目标是识别而非深挖；

d） 分析结果是制定系统安全性要求、规划后续更详细安全性分析活动的主要依据。

8.12 功能危险分析

通过系统性地识别系统功能在执行过程中潜在的失效状态，并评估其可能导致的功能丧失、性能退化或灾难性危险后果，从而确定各功能失效的严重等级。

适用对象：

适用于航空、航天、核电、医疗等安全攸关领域，尤其是具有多功能耦合、复杂运行工况或高安全完整性等级（SIL）的复杂软件系统。

实施步骤：

a） 开展系统功能分析；

b） 分析每一个功能可能出现的故障状态及可能导致的危险，并形成功能故障清单；

c） 确定功能故障状态的危险影响及分类。

实施要点

a） 应在需求分析阶段启动，并随着设计的细化和变更持续迭代更新；

b） 分析应覆盖所有已识别的软件功能，但可根据优先级采取不同的分析粒度。对于核心功能，需进行逐项深入分析；对于一般功能，可进行分组分析。

8.13 符号执行

符号执行将程序输入作为符号变量，探索不同执行路径，能覆盖边界条件并发现潜在缺陷，为安全关键算法模块和复杂控制程序提供深度功能安全性验证。

适用对象：

a） 安全关键算法模块；

b） 逻辑复杂度高的控制程序。

实施步骤：

a） 准备源代码、编译配置、安全需求规格和高风险代码路径清单；

b） 执行符号化分析，探索路径；

c） 输出覆盖率报告、缺陷清单和测试用例集。

实施要点：

a） 分支覆盖率达到目标（如 ≥90%）；

b） 所有高风险缺陷均分析并制定方案；

c） 报告完整，路径与输入可复现。

8.14 混沌测试

混沌测试利用故障注入、延迟模拟、资源扰动等技术，在受控环境中验证软件系统的鲁棒性与关键功能可用性。方法包括实验设计、异常注入脚本、恢复能力验证及数据采集分析，适用于分布式、云化或关键基础设施中的软件功能安全验证与持续改进。

适用对象：

a） 高可用性与高安全性要求的复杂软件系统；

b） 隔离或具备回退机制的环境。

实施步骤：

a） 准备运行架构图、关键服务清单、故障注入脚本；

b） 在受控环境中实施混沌测试；

c） 输出混沌测试报告与风险整改建议。

实施要点：

a） 软件系统需在规定时间内恢复关键功能；

b） 关键安全功能在测试期间保持可用；

c） 测试报告完整、可追溯。

8.15 模糊测试

模糊测试通过随机或畸形输入生成技术，对软件系统接口和数据处理逻辑进行高强度异常测试。技术方法包括输入生成策略、自动化执行、异常捕获、漏洞定位和修复建议生成，可发现功能安全相关模块的潜在漏洞和健壮性缺陷。

适用对象：

a） 涉及外部输入的安全相关模块；

b） 需确保输入数据不破坏测试环境的软件系统。

实施步骤：

a） 准备接口规范、协议说明书、模糊测试工具配置文件和种子样本集；

b） 执行模糊测试，捕捉异常信息；

c） 输出缺陷列表与修复建议。

实施要点：

a） 测试过程中软件系统无崩溃、死锁等缺陷触发；
b） 对发现的缺陷制定修复或缓解措施；
c） 结果记录详细且可追溯。

8.16 安全回归测试

安全回归测试采用自动化测试与历史用例库，验证复杂软件在迭代更新或修复后功能安全性保持一致，确保功能安全特性在软件系统演进过程中的连续性和完整性。

适用对象：
a） 有完整历史安全测试用例库的软件系统；
b） 每次更新或修复后需回归验证的软件系统。

实施步骤：
a） 准备历史安全用例、当前版本信息、更新说明；
b） 在自动化环境中重复执行测试；
c） 输出回归测试报告与验证记录。

实施要点：
a） 历史测试用例通过率 100%；
b） 新增问题有完整记录与处置。

8.17 压力测试与负载测试

压力与负载测试利用模拟高并发、资源受限或极端操作条件，对复杂软件系统的稳定性、安全功能可用性和性能瓶颈进行技术验证，为关键功能安全性提供量化评估和优化依据。

适用对象：
a） 关键性能与安全相关软件系统；
b） 环境接近实际运行条件的软件系统。

实施步骤：
a） 准备性能需求、模拟负载场景与监控配置；
b） 实施压力与负载测试；
c） 输出测试报告与风险分析报告。

实施要点：
a） 软件系统在预设负载下，关键功能不失效；
b） 性能瓶颈明确并提出改进措施。

8.18 离线功能安全评估

离线评估通过历史运行数据批量处理与趋势分析技术，对软件系统潜在风险、失效模式和模型性能进行离线验证，为功能安全管理提供周期性分析和决策依据。

适用对象：
a） 需周期性安全评估的软件系统；
b） 运行数据覆盖完整周期的场景。

实施步骤：
a） 收集历史运行数据与模型参数；
b） 使用分析工具进行趋势分析和验证；
c） 输出离线评估报告与风险预测图表。

实施要点：
a） 数据分析覆盖率达标；

b） 重大风险趋势均生成预警。

8.19 在线功能安全评估

在线评估通过实时采集运行数据，利用阈值检测、异常分析和动态评估技术，对软件系统关键安全参数进行连续监控，保障关键基础设施中的复杂软件运行期间的功能安全性

适用对象：

a） 对实时安全性要求高的软件系统（电网、交通、工业控制等）；

b） 具备稳定在线监控与采集通道的软件系统。

实施步骤：

a） 采集实时运行数据；

b） 基于安全阈值进行分析；

c） 输出在线评估报告与告警记录。

实施要点：

a） 覆盖所有关键安全参数；

b） 所有超阈值情况均有响应记录。

9 复杂软件系统功能安全性全生命周期过程与活动

9.1 软件需求阶段

该阶段主要开展软件安全性要求的识别和验证活动。应识别功能安全相关需求、模块和操作场景，并对潜在危险源进行风险分析，明确功能安全性的定量和定性要求。可采用风险模型构建、风险知识图谱、威胁建模、初步危险分析、功能危险分析及安全需求可追踪性矩阵等方法，对功能安全相关需求的完整性、一致性和可追溯性进行验证，并形成初步风险控制措施。

9.2 软件设计阶段

该阶段主要开展软件安全性分析、设计和验证活动。应对安全相关模块进行安全性分析、架构设计与接口防护，确保安全功能正确性、独立性与防护隔离。可通过容错设计与冗余建模、STAMP 分析方法、FMEA分析、FTA分析、形式化方法等手段，确保软件系统在异常情况下的可控性和安全性，并建立设计文档的可追溯性和完整性验证机制。

9.3 软件实现阶段

该阶段主要开展软件安全性要求的实现和代码安全性分析验证活动。应按照安全编码规范实现功能安全相关模块，对外部依赖库和组件进行安全验证与版本管理。可通过符号执行、静态分析、代码审查及自动化工具，对潜在漏洞进行预防和控制，同时确保实现与设计的一致性。

9.4 软件测试阶段

该阶段主要开展软件安全性测试活动。应对功能安全相关要求、模块开展全面测试和验证工作。可采用混沌测试、模糊测试、安全回归测试、压力与负载测试等方法，保证软件系统在各种操作条件下的安全性，并对潜在风险进行隔离与防护。

9.5 部署与发布阶段

该阶段主要开展软件安全性分析评估活动。应对可执行文件、运行环境和依赖组件进行安全验证和配置管理。可结合在线评估、离线评估、版本控制与补丁管理措施，确保软件在上线后的运行安全，并为后续运行阶段提供安全基础和可追溯记录。

9.6 运行和使用阶段

该阶段主要开展软件安全性分析评估活动。对于功能安全相关的模块、任务和软件运行状态。可采用在线评估、离线评估、基于风险模型再评估的方法，结合规则、统计或深度学习模型，开展运行健康管理与风险预测，并对安全风险进行有效的溯源、隔离与防护。

参 考 文 献

[1] ISO/IEC 25010 系统与软件工程 系统与软件质量要求和评价
[2] IEC 61513—2011 核电厂对安全重要的测试设备和控制系统的一般要求
[3] IEC 60812—2018 失效模式和影响分析(FMEA和FMECA)
[4] IEC 60987—2021 核电站对安全重要的仪表和控制硬件要求
[5] GB/T 35778—2017 企业安全生产网络化监测系统技术规范
[6] GB/T 36344—2018 信息技术 数据质量评价指标
[7] GB/T 35273—2020 个人信息保护相关要求参考，可衍生隐私测试
[8] GB/T 25000—2021 系统与软件工程 系统与软件质量要求和评价
[9] GB/T 44442—2024 智能制造 远程运维系统 评价指标体系
[10] GB/T 44662—2024 健康管理 终端设备数据采集与传输协议
[11] T/CSAE 158—2020 智能网联汽车车载计算平台功能安全要求
[12] T/ITS 0156—2022 车路协同系统混沌测试方法
[13] DO-178C 航空系统和设备认证中的软件考虑因素
[14] NIST SP 800-53 Rev5 信息系统和组织的安全和隐私控制
[15] NIST NISTIR 8012 制造领域预测与健康管理（PHM）相关标准综述
[16] IEEE 1856—2017 EEE 电子系统预测与健康管理标准框架
[17] IEEE 3110—2025 IEEE 计算机视觉（CV）深度学习框架算法应用编程接口（API）技术要求标准
[18] US-SAE AIR7999 航空航天推进系统健康管理诊断与预测度量标准