中 国 指 挥 与 控 制 学 会

中国指挥与控制学会“复杂系统可靠性与安全性”系列九项团体标准发布公告

中国指挥与控制学会根据《中国指挥与控制学会团体标准管理办法》规定，对中国指挥与控制学会可靠性系统科学与工程专业委员会负责的《复杂软件系统信息安全性技术要求》《复杂软件系统保障性技术要求》《复杂软件系统故障预测与健康管理技术要求》《复杂智能系统信息安全性技术要求》《复杂智能系统保障性技术要求》《复杂智能系统故障预测与健康管理技术要求》《装备体系结构可靠性建模与预计》《装备体系任务可靠性建模与评估》《装备体系韧性建模与评估》九项团体标准进行了立项、征求意见、评审等工作。将在全国团体标准信息平台上进行发布。现予以公告，公告期为 10 个工作日，联系方式：cicc_tb@c2.org.cn。

具体内容见附件。

特此公告

中国指挥与控制学会

2026 年 2 月 28 日

ICS 35.080
CCS L 77

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35001—2026

复杂软件系统信息安全性技术要求

Technical requirements for security of complex software systems

2026-02-28发布 2026-02-28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。

本文件起草参与单位：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）、中国科学院声学研究所、中国船舶集团有限公司综合技术经济研究院、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、彭晟豪、郝程鹏、周怡婧、司昌龙、丁健洋、王晶、吴梦丹、刘佳、张炜华。

复杂软件系统信息安全性技术要求

1 范围

本文件规定了复杂软件系统在信息安全性方面的定性要求、定量要求、支撑技术以及生命周期阶段的过程与活动。

本文件适用于复杂软件系统的信息安全性设计、实现、测试与运维管理。

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，仅所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 8566—2022 系统与软件工程 软件生存周期过程

GB/T 22239—2019 信息安全技术网络安全等级保护基本要求

GB/T 25000.10—2016 系统与软件工程 系统与软件质量要求和评价系统与软件质量模型

GB/T 25000.51—2016 系统与软件工程 系统与软件质量要求和评价（SQuaRE）第51部分：就绪可用软件产品（RUSP）的质量要求和测试细则

GB/T 32857—2016 保护层分析（LOPA）应用指南

GJB/Z 157—2011 军用软件安全保证指南

GJB 5236—2004 军用软件质量度量

T/CICC 35008—2025 复杂软件系统可靠性技术要求

3 术语与定义

GB/T 8566－2022、GJB/Z 157－2011和GB/T 25000.10－2016界定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

风险分析 risk analysis

识别安全风险、判断其程度并且确定需要采取安全控制措施的领域的过程。

[来源：GJB/Z 157－2011，3.1.5]

3.3

安全控制 security controls

为保护某一信息系统及其信息的安全性，而对其规定的管理、运行和技术的控制措施（即防护措施或对抗措施）。

[来源：GJB/Z 157－2011，3.1.8]

3.4

脆弱性　vulnerability

能够被某种威胁利用的某个或某组资产的弱点。

[来源：GJB/Z 157－2011，3.1.9]

3.5

保密性　confidentiality

产品或系统确保数据只有在授权时才能被访问的程度。

[来源：GB/T 25000.10－2016，4.3.2.6.1]

3.6

完整性　integrity

系统、产品或组件防止未授权的访问、篡改计算机程序或数据的程度。

[来源：GB/T 25000.10－2016，4.3.2.6.2]

3.7

抗抵赖性　non-repudiation

活动或事件发生后可以被证实且不可被否认的程度。

[来源：GB/T 25000.10－2016，4.3.2.6.3]

3.8

可核查性　verifiability

实体的活动可以被唯一地追溯到该实体的程度，重点在追溯实体的程度。

[来源：GB/T 25000.10－2016，4.3.2.6.4，有修改]

3.9

真实性　authenticity

对象或资源的身份标识能够被证实符合其声明的程度。

[来源：GB/T 25000.10－2016，4.3.2.6.5]

3.10

可用性　availability

系统、产品或组件在需要使用时能够进行操作和访问的程度。

[来源：GB/T 25000.10－2016，4.3.2.5.2，有修改]

3.11

可追溯性　traceability

在两个或两个以上的逻辑实体之间建立关系的程度，特别是相互之间有前后相继或主从关系的实体。

[来源：GB/T 8566－2022，3.1.69]

4　缩略语

下列缩略语适用于本文件：

API	应用程序编程接口 （Application Programming Interface）
CD	持续交付/持续部署 （Continuous Delivery / Deployment）
CI	持续集成 （Continuous Integration）

DAST 动态应用程序安全测试 （Dynamic Application Security Testing）
HTTP 超文本传输协议 （HyperText Transfer Protocol）
IAST 互动式应用安全测试 （Interactive Application Security Testing）
RASP 运行时应用自我保护 （Runtime Application Self-Protection）
SAST 静态应用安全测试 （Static Application Security Testing）
SQL 结构化查询语言 （Structured Query Language）
SBOM 软件物料清单 （Software Bill of Materials）
SCA 软件成分分析 （Software Composition Analysis）
STRIDE 身份伪造，篡改，否认，信息泄露，拒绝服务，权限提升 （Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege）

5 软件信息安全性定性要求

5.1 软件信息安全性分析

5.1.1 保密性要求

软件的保密性要求如下：

a) 确保数据只有在授权时才能被访问；
b) 对通信过程中整个会话过程进行加密；
c) 保证敏感信息在存储过程中的保密性；
d) 应启用访问控制功能，依照安全策略和用户的角色设置访问控制矩阵；
e) 对涉及个人身份信息的敏感数据，需遵循“同意与选择”原则，确保数据收集与使用获得用户明确授权，且符合目的合法性要求。

5.1.2 完整性要求

软件的完整性要求如下：

a) 软件校验应增设校验位，并采用循环冗余校验方式实现；
b) 采用散列运算和数字签名等方式实现通信过程中的数据完整性；
c) 采用关系型数据库保存数据；
d) 增加数据完整性约束；
e) 实现事务的原子性，避免因操作中断或回滚导致的数据不一致及完整性被破坏。

5.1.3 抗抵赖性要求

软件的抗抵赖性要求如下：

a) 启用安全审计功能，对活动或事件进行追踪；
b) 对审计日志进行管理，日志不能被任何人修改或删除，形成完整的证据链；
c) 采用数字签名处理事务，在收到请求的情况下为数据原发者或接收者提供数据原发和接收证据。

5.1.4 可核查性要求

软件的可核查性要求如下：

a) 用户活动的日志记录包含关键信息；
b) 采取审计跟踪设置，定义审计跟踪极限的阈值，当存储空间被耗尽时，应采取必要的保护措施。

5.1.5 真实性要求

软件的真实性要求如下：

a) 软件系统应提供专用的登录控制模块对登录用户进行身份标识和鉴别，验证其身份的真实性，同时需证实符合其声明的程度；
b) 软件系统中应有唯一标识符且与用户一一对应，采用用户名和口令的方式对用户进行身份鉴别（如采用强化管理的口令鉴别，基于令牌的动态口令鉴别等），提高用户的口令开启复杂度；
c) 软件系统应提供登录失败处理功能，采取如结束会话、限制非法登录次数和自动退出等措施；
d) 审计日志应包含个人身份信息处理记录（如访问、修改、传输），且日志本身需符合隐私保护要求（如避免过度记录无关个人信息），秉持“公开与透明”原则。

5.2 代码信息安全性要求

5.2.1 安全编码规范

源码编写环节需满足的安全开发要求如下：

a) 源码编写应全面遵循安全开发规范；
b) 必须禁止硬编码凭证、明文存储密码或密钥等高风险操作；
c) 所有输入必须经过合法性校验与长度限制，禁止信任用户输入；
d) 代码应对异常处理进行敏感信息隐藏设计，确保攻击者无法通过异常信息倒推出系统内部逻辑。

5.2.2 安全依赖与组件管理

软件依赖组件的安全管理要求如下：

a) 软件应维护SBOM；
b) 所有第三方库与开源组件必须定期进行安全审计与漏洞扫描；
c) 部署前应确认依赖组件无已知严重安全漏洞，并制定补丁更新策略。

5.2.3 代码自检与静态分析

开发流程中的代码安全检测要求如下：

a) 开发阶段应引入静态代码安全分析工具，以发现注入攻击、缓冲区溢出、越权访问等潜在问题；
b) 自动化构建流程中应集成安全扫描，禁止高风险漏洞代码进入主干分支。

5.3 接口信息安全性要求

5.3.1 接口规范化

接口设计与使用的安全规范化要求如下：

a) 所有接口必须有完整的接口文档，明确参数范围、返回值、错误码和安全约束；
b) 接口应遵循最小信息暴露原则，禁止返回敏感系统信息（如堆栈信息、数据库错误细节）。

5.3.2 接口认证与授权

接口访问的身份鉴别与权限管控要求如下：

a) 必须对所有接口调用方进行身份鉴别与访问授权；
b) 对于关键接口，应强制启用双因素认证或令牌认证机制；
c) 接口权限设计应遵循最小权限原则，仅授予当前任务所需功能。

5.3.3 数据保护与可观测性

接口通信安全与运行监控的防护要求如下：

a) 接口通信必须强制启用适合版本的传输安全协议，禁止明文传输敏感数据；
b) 接口调用日志必须完整记录访问主体、时间、参数与结果，支持安全溯源；

c) 应支持对异常输入、恶意请求的检测与告警。

5.3.4 接口鲁棒性与隔离性

接口的容错防护与外部依赖降级保障要求如下：

a) 接口应对非法输入、超长输入和攻击载荷输入进行容错处理，避免导致系统崩溃或被控制；
b) 对于尚不可用或高成本的外部依赖，应提供可替代的虚拟接口或模拟服务，以确保系统独立性。

5.4 模型信息安全性要求

5.4.1 模型管理与访问控制

模型资产作为软件系统设计的核心载体，其管理与访问控制要求如下：

a) 包含类图、组件图、部署图在内的所有模型文件或数据库，应纳入配置管理体系，实施严格的版本控制，确保设计变更的可追溯性；
b) 需对核心设计模型实施细粒度的访问控制，仅授权人员可对其中涉及关键架构或核心逻辑的部分进行查阅与修改；
c) 应确保模型数据的完整性，采用数字签名或校验机制，防止存储设计蓝图的元数据或实体在存储或传输过程中被非法篡改；
d) 应对包含敏感信息的模型视图进行加密存储，以防止设计方案泄露。

5.4.2 模型逻辑与形式化验证

在设计阶段对软件系统逻辑进行安全性验证的要求如下：

a) 应对关键安全机制（如访问控制、状态迁移）建立形式化的状态机图或活动图模型，并进行形式化验证，证明其满足无死锁、活性及一致性要求；
b) 应对业务流程模型或活动图中的关键决策点和分支进行审查，确保不存在可被利用的逻辑漏洞。

5.4.3 模型实现一致性

针对从模型到代码或硬件实现的转换过程，安全要求如下：

a) 若采用基于类图或实体关系图的自动代码生成技术，应验证代码生成器的可信度，或对生成的代码进行审查，确保未引入额外的安全漏洞；
b) 应建立从需求到设计再到代码的可验证追溯链接，确保每一项安全需求都能明确映射到具体的设计模型元素，并最终体现在代码实现中。

5.4.4 模型仿真与早期评估

利用模型的可执行特性进行早期安全评估的要求如下：

a) 应对数据流图或活动图进行数据流分析，验证高密级数据与低密级数据在逻辑路径上的隔离有效性；
b) 用于执行行为模型仿真的环境，应与外部网络或生产环境隔离，防止仿真数据泄露或被外部攻击影响；
c) 应基于交互模型，对用户与系统交互的路径进行仿真，识别可能导致权限提升或非预期操作的场景。

5.5 软件系统信息安全性要求

5.5.1 软件系统模式与可测性

软件系统安全运行保障与监控审计的能力要求如下：

a) 软件系统应支持安全模式与正常模式切换，确保安全测试不影响业务运行；

b) 关键软件系统需支持在线安全检测与离线安全审计两种能力；
c) 软件系统须具备全维度、实时化的系统状态可观测能力，实现运行及安全相关状态的可感知、可度量、可追溯；
d) 软件系统必须提供统一的安全监控接口，实时采集关键指标（如登录失败率、流量模式异常、自编码重构误差）；
e) 需集中化日志与告警管理，保障可追踪与审计。

5.5.2 软件系统行为可控性

软件系统安全防护有效性的验证与演练能力要求如下：
a) 软件系统应具备攻击仿真与防御演练能力；
b) 在安全运营中，应能通过配置或脚本复现攻击路径，以验证安全防护措施有效性。

5.5.3 软件系统自检与持续安全运营

软件系统安全自检、运维安全管控及信息安全核心防护的要求如下：
a) 软件系统应内置安全自检机制：
   1) 启动自检：验证配置、安全策略是否有效；
   2) 运行自检：周期性检测安全策略与访问控制状态；
   3) 维护自检：运维阶段执行全面的安全审计。
b) 运维过程应包含自动化漏洞扫描、补丁管理和安全基线检查机制。

6 软件信息安全性定量指标

6.1 访问的可审核性

访问的可审核性是访问记录可审核能力的度量，指按照规格说明得到正确记录的访问类型数量与规格说明中需要记录的访问类型数量之比，计算方法见公式（1）。

$$A = \frac{R}{T} \cdots\cdots (1)$$

式中：

A：访问的可审核性；

R：按规格说明被记录的访问类型数；

T：规格说明中要求记录的访问类型数。

6.2 访问的可控制性

访问的可控制性是衡量对系统访问的可控程度的度量，指按照规格说明已被正确实现的可控制访问需求数与规格说明中的可控制访问需求数之比，计算方法见公式（2）。

$$C = \frac{I}{S} \cdots\cdots (2)$$

式中：

C：访问的可控制性；

I：正确实现的可控制访问需求数；

S：规格说明中的可控制访问需求数。

6.3 数据的抗抵赖性

数据抗抵赖性的度量，用于评估防止参与方抵赖其数据操作或访问行为的完备程度。该度量值为实际实现防抵赖能力的数据操作（访问）实例数与需求中规定需实施防抵赖的同类数据实例数的比值，具体计算方法见公式（3）。

$$R_{nr} = \frac{V}{\mathrm{D}} \quad \cdots\cdots (3)$$

式中：

R_{nr}：数据抗抵赖性度量值；

V：实际实现防抵赖能力的数据操作（访问）实例数；

D：需求中规定需实施防抵赖的同类数据实例数。

6.4 数据加密覆盖率

数据加密覆盖率是衡量敏感数据保密性的度量，指在存储或传输过程中已按规定实现加密保护的敏感数据字段或条目数与规格说明中要求加密的所有敏感数据字段或条目总数之比，计算方法见公式（4）。

$$C_{enc} = \frac{I_m}{Res} \quad \cdots\cdots (4)$$

式中：

C_{enc}：数据加密覆盖率；

I_m：已按规定实现加密保护的敏感数据字段或条目数；

R_{es}：规格说明中要求加密的敏感数据字段或条目总数。

6.5 抗注入攻击能力覆盖率

抗注入攻击能力覆盖率是衡量软件抵御注入类攻击（如SQL注入、命令注入）能力的度量，指已采用参数化查询或严格输入验证等安全措施进行防护的外部数据输入点数量，与系统中存在的全部外部数据输入点总数之比，计算方法见公式（5）。

$$C_{ia} = \frac{P}{N} \quad \cdots\cdots (5)$$

式中：

C_{ia}：抗注入攻击能力覆盖率；

P：已采用安全措施进行防护的外部数据输入点数量；

N：系统中存在的全部外部数据输入点总数。

6.6 权限变更合规率

权限变更合规率是衡量系统权限变更操作符合安全规范程度的度量，指符合预设安全流程的权限变更次数与总权限变更次数之比，计算方法见公式（6）。

$$P_c = \frac{C}{M} \quad \cdots\cdots (6)$$

式中：

P_c：权限变更合规率；

C：符合规范的权限变更次数；

M：总权限变更次数。

6.7 敏感操作响应时效达标率

敏感操作响应时效达标率是衡量系统对高风险操作的实时监控与响应能力的度量，指在规定时间内完成处置的敏感操作次数与总敏感操作次数之比，计算方法见公式（7）。

$$R_t = \frac{H}{S} \cdots\cdots (7)$$

式中：

R_t：敏感操作响应时效达标率；

H：时效内完成处置的敏感操作次数；

S：总敏感操作次数。

6.8 安全补丁安装覆盖率

安全补丁安装覆盖率是衡量系统对已知安全漏洞修复及时性的度量，指已安装指定安全补丁的终端设备与功能模块的总数量，与需安装该补丁的终端设备与功能模块的总数量之比，计算方法见公式（8）。

$$C_p = \frac{D}{L} \cdots\cdots (8)$$

式中：

C_p：安全补丁安装覆盖率；

D：已安装补丁的终端设备和功能模块的总数量；

L：需安装补丁的终端设备和功能模块的总数量。

6.9 数据传输加密率

数据传输加密率是衡量系统数据在网络传输过程中加密保护程度的度量，指采用指定加密算法进行传输的会话数与总数据传输会话数之比，计算方法见公式（9）。

$$R_{et} = \frac{E}{\mathrm{T}_t} \cdots\cdots (9)$$

式中：

R_{et}：数据传输加密率；

E：加密传输的会话数；

T_t：总传输会话数。

6.10 身份认证成功率

身份认证成功率是衡量系统身份鉴别机制准确性的度量，指合法用户通过身份认证的次数与合法用户发起认证尝试总次数之比，计算方法见公式（10）。

$$R_{auth} = \frac{K}{Q} \cdots\cdots (10)$$

式中：

R_{auth}：身份认证成功率；

K：合法用户成功认证次数；

Q：合法用户认证尝试总次数。

6.11 安全日志留存达标率

安全日志留存达标率是衡量系统日志管理合规性的度量，指满足预设留存时长要求的安全日志条目数与总安全日志条目数之比，计算方法见公式（11）。

$$R_{\log}=\frac{F}{W} \cdots\cdots（11）$$

式中：

$R_{\log}$：安全日志留存达标率；

F：达标留存的日志条目数；

W：总日志条目数。

6.12 漏洞扫描覆盖率

漏洞扫描覆盖率是衡量系统安全检测全面性的度量，指已完成指定周期漏洞扫描的系统组件数与应进行扫描的系统组件总数之比，计算方法见公式（12）。

$$C_v=\frac{U}{O} \cdots\cdots（12）$$

式中：

C_v：漏洞扫描覆盖率；

U：已扫描组件数；

O：应进行扫描的系统组件总数。

6.13 会话超时合规率

会话超时合规率是衡量系统会话管理安全性的度量，指在规定闲置时间内自动超时的会话数与总非活跃会话数之比，计算方法见公式（13）。

$$R_{to}=\frac{J}{X} \cdots\cdots（13）$$

式中：

R_{to}：会话超时合规率；

J：合规超时的会话数；

X：总非活跃会话数。

6.14 错误信息泄露率

错误信息泄露率是衡量系统异常处理安全性的度量，指错误提示中泄露敏感信息的错误发生次数与系统总错误发生次数之比，计算方法见公式（14）。

$$L_e=\frac{G}{B} \cdots\cdots（14）$$

式中：

L_e：错误信息泄露率；

G：泄露敏感信息的错误次数；

B：总错误次数。

6.15 日志自动化分析覆盖率

日志自动化分析覆盖率是衡量系统产生的日志数据被安全分析工具有效处理的程度，指已接入自动化分析工具的日志类型数与系统生成的关键日志总类型数之比，计算方法见公式（15）。

$$R_{la}=\frac{Y}{Z}\cdots\cdots(15)$$

式中：

R_{la}：日志自动化分析覆盖率；

Y：已接入自动化分析工具（如自编码器）进行实时监测的日志类型数；

Z：规格说明中要求生成的关键日志总类型数。

7 软件信息安全性支撑技术

7.1 软件信息安全性需求阶段支撑技术

7.1.1 需求阶段代码级信息安全支撑技术

7.1.1.1 安全编码规范定义与基线化方法

7.1.1.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过建立标准化的编码规则集（安全基线），系统性地解决软件开发初期因编码习惯和安全意识不足而引入的安全问题；

b) 核心原理：将业界实践、内部经验和技术要求进行归纳、提炼并固化为可执行的规则，以实现安全知识的规模化复用与传递；

c) 主要优势：

1) 安全前置：将安全要求前置到编码活动的最前端；

2) 统一基准：统一团队的安全编码标准，提升代码内建安全性；

3) 降本增效：通过早期预防漏洞来降低后期修复成本。

7.1.1.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 项目所使用的特定编程语言、框架和开发平台；

2) 组织全体软件开发人员。

b) 实施步骤：

1) 选取基础规范：基于业界公认的安全编码实践，结合组织技术栈进行选择；

2) 本地化与定制：根据组织特有的业务场景、技术架构和历史漏洞经验，对基础规范进行补充、细化和定制；

3) 提供正反案例：针对每条规范，提供清晰的“正确”与“错误”的代码示例，增强规范的可操作性和易理解性；

4) 评审与培训：组织安全专家和资深开发人员对规范进行评审，并对全体开发人员进行系统性培训。

c) 实施要点：

1) 核心目标：为开发人员提供一套清晰、可执行的编码准则，用于在编写代码时规避常见的安全漏洞；

2) 关键方法：将安全知识和最佳实践转化为具体的编码规则；

3) 主要产出：一份针对特定技术栈的安全编码规范文档，以及配套的培训材料；

4) 最终价值：在编码阶段直接预防安全漏洞的产生，统一开发团队的安全编码习惯，从根本上提升代码的内建安全性。

7.1.1.2 密码算法选型与敏捷性规约技术

7.1.1.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在需求阶段确立系统所允许使用的加密算法集合及敏捷性要求，防止使用弱加密算法，并确保系统具备在算法被攻破时快速切换的能力；

b) 核心原理：建立允许，禁止或弃用的算法清单，并强制要求代码通过抽象层调用密码服务，而非直接依赖底层硬编码实现；

c) 主要优势：

1) 合规性与强度：确保所有加密操作符合行业标准，消除弱密码学风险；

2) 未来适应性：当某算法被发现存在漏洞时，通过配置即可完成替换，无需重构核心业务代码。

7.1.1.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 所有涉及数据加密、哈希、签名操作的代码模块；

2) 负责架构设计的系统架构师与安全工程师。

b) 实施步骤：

1) 建立算法清单：定义“推荐”“允许”“弃用”及“禁止”的算法列表、密钥长度及工作模式；

2) 设计抽象接口：定义统一的接口，封装具体的加解密实现细节；

3) 配置化管理：将具体的算法选择与密钥管理策略移至外部配置文件中；

4) 合规性审查：在需求评审阶段，核对所有加密需求是否落入允许清单范围内。

c) 实施要点：

1) 核心目标：在代码编写前规约密码学使用标准，构建具备算法敏捷性的加密架构；

2) 关键方法：制定严格的算法白名单，并强制推行面向接口的密码学编程模式；

3) 主要产出：一份密码算法选型规范文档，以及配套的密码学抽象层接口定义；

4) 最终价值：根除因开发人员随意选择算法导致的安全隐患，大幅降低未来应对量子计算等新型威胁时的改造成本。

7.1.2 需求阶段接口级信息安全支撑技术

7.1.2.1 接口安全契约规约技术

7.1.2.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在将接口的安全要求作为其定义的核心组成部分，以确保在接口设计之初，安全属性与功能属性获得同等的明确与共识；

b) 核心原理：在接口描述语言或设计文档中，通过增加专门的字段或章节，对认证授权、数据校验、速率限制等安全要素进行形式化、结构化的规约，形成一份包含安全条款的“契约”；

c) 主要优势：

1) 前瞻性与契约性：强制在服务提供方和消费方之间就安全细节达成一致，避免接口误用；

2) 驱动自动化：为后续的自动化安全测试、API网关策略配置和安全审计提供精确、可执行的依据。

7.1.2.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 系统对外或对内提供的所有应用程序编程接口；
   2) 微服务架构中的服务间接口。
b) 实施步骤：
   1) 明确认证授权机制：在接口定义中，明确规定调用方需要使用的认证方式（如令牌、证书）和必需的权限；
   2) 定义数据格式与校验规则：精确定义接口请求和响应的数据结构、字段类型、长度限制和内容约束；
   3) 规定敏感数据处理：标识接口中传输的敏感数据，并规定其处理方式，如必须加密、脱敏或禁止传输；
   4) 约定错误处理与速率限制：定义统一的错误码格式和安全相关的错误处理机制，并明确接口的调用频率限制策略。
c) 实施要点：
   1) 核心目标：将安全要求作为接口定义的一部分，与功能需求一同明确；
   2) 关键方法：在接口设计文档或标准化的接口描述语言中，增加专门的安全规约章节或字段；
   3) 主要产出：一份包含明确安全约定的接口设计文档或契约文件；
   4) 最终价值：使接口的安全要求在服务提供方和消费方之间达成共识，为后续的自动化测试和安全防护提供精确依据，避免因接口误用导致安全问题。

7.1.2.2 接口防自动化与速率限制规约技术

7.1.2.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在接口定义阶段明确防自动化工具滥用（如撞库、爬虫）的具体技术指标与策略；
b) 核心原理：基于令牌桶或漏桶算法定义限流阈值，并结合工作量证明或指纹识别机制区分人类用户与自动化脚本；
c) 主要优势：
   1) 可用性保障：防止因恶意高频调用导致的资源耗尽，保障服务；
   2) 资产保护：增加攻击者批量获取数据或暴力破解凭证的成本。

7.1.2.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 公网暴露的网关入口及登录、查询类高风险接口；
   2) 内部微服务间的关键调用链路。
b) 实施步骤：
   1) 定义限流维度：明确基于互联网协议地址、用户标识或地理位置的速率限制粒度；
   2) 设定阈值策略：针对不同接口的业务属性，设定每秒查询数及并发数上限；
   3) 规约挑战机制：定义触发限流后的响应动作（如拒绝服务、弹出验证码、暂缓响应）；
   4) 异常豁免设计：为合法的自动化测试或内部调用设计白名单机制。

c) 实施要点：
   1) 核心目标：在接口设计之初植入防滥用基因，确保接口在非预期流量下的生存能力；
   2) 关键方法：将速率限制与人机识别作为接口的标准非功能性需求；
   3) 主要产出：包含限流阈值与熔断策略的接口防滥用规范文档；
   4) 最终价值：有效抵御应用层拒绝服务攻击及自动化业务欺诈风险。

7.1.3 需求阶段模型级信息安全支撑技术

7.1.3.1 STRIDE威胁分析技术

7.1.3.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在软件生命周期的最早阶段（设计阶段），系统性地识别和评估潜在的安全威胁，从而指导安全设计的方向和重点；
b) 核心原理：基于一个成熟的威胁分类框架（身份仿冒、篡改、抵赖、信息泄露、拒绝服务、权限提升），对系统分解后的每个元素进行审视，以确保威胁识别的广度；
c) 主要优势：
   1) 结构化与全面性：提供标准化的、可重复的流程，确保威胁识别的广度；
   2) 设计驱动：在设计初期进行威胁建模，确保安全措施的针对性，将安全考虑融入架构；
   3) 降低成本：降低后期修复因设计缺陷导致的安全问题的成本。

7.1.3.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 系统的设计模型，特别是数据流图；
   2) 系统的组件、数据流动路径、处理过程以及信任边界。
b) 实施步骤：
   1) 分解系统：使用数据流图等形式，将系统分解为外部实体、进程、数据存储和数据流等基本元素，并清晰地标识出信任边界；
   2) 识别威胁：针对数据流图中的每一个元素，系统性地应用STRIDE的六个威胁类别进行分析，识别潜在威胁；
   3) 确定对策：为每一个已识别的有效威胁设计缓解措施或安全控制方案；
   4) 验证模型：在设计和实现完成后，验证所设计的对策是否被正确实施并有效缓解了威胁。
c) 实施要点：
   1) 核心目标：在软件生命周期的最早阶段，系统性地识别和分类潜在安全威胁，为安全设计提供明确输入；
   2) 关键方法：基于标准化的STRIDE框架对系统设计模型进行结构化分析；
   3) 主要产出：一份详细的威胁列表、风险评估结果以及与之对应的安全需求或缓解措施文档；
   4) 最终价值：确保安全措施从设计之初就具有高度针对性，避免后期昂贵的安全返工，是实现设计阶段安全需求分析的实践方法。

7.1.3.2 攻击树分析技术

7.1.3.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在从攻击者的视角出发，系统地分析并可视化达成某一特定攻击目标所有可能的路径，用以识别系统的薄弱环节；
b) 核心原理：采用一种自顶向下的分解方法：将高层次的攻击目标作为树的根节点，然后逐层向下分解为实现父节点目标的子步骤，直至最底层的叶节点是具体的攻击动作；
c) 主要优势：
   1) 直观性与聚焦性：通过可视化的树状结构，清晰地展示所有潜在的攻击向量；
   2) 量化决策：通过对叶节点进行评估，可量化分析每条攻击路径的威胁程度，为安全加固的优先级排序提供依据。

7.1.3.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 一个高层次的、全局性的攻击目标；
   2) 系统的业务逻辑与交互流程。
b) 实施步骤：
   1) 定义根节点：将一个全局性的、高价值的攻击目标设定为攻击树的根节点；
   2) 逐层分解：采用自顶向下的方式，将根目标逐层分解为达成该目标所需的一系列更具体的子任务或前提条件；
   3) 细化至叶节点：持续分解，直至树的叶节点代表具体的、不可再分的攻击动作；
   4) 分析攻击路径：分析从叶节点到根节点的完整路径，评估每条路径的实现难度、成本与成功率。
c) 实施要点：
   1) 核心目标：以攻击者视角，可视化所有可能的攻击路径，从而识别出系统最关键、最薄弱的防护点；
   2) 关键方法：采用自顶向下的分解方法，将抽象的攻击目标具体化为一系列可执行的步骤；
   3) 主要产出：一个可视化的攻击树图，以及基于该树的攻击路径分析与优先级评估报告；
   4) 最终价值：为量化评估攻击成本和成功率提供基础，帮助团队科学地确定安全需求的优先级。

7.1.4 需求阶段软件系统级信息安全支撑技术

7.1.4.1 安全需求基线生成与裁剪方法

7.1.4.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在建立一套标准化的、可复用的基础安全需求集合（安全需求基线），以确保组织内所有软件项目都能满足统一的基础安全标准；
b) 核心原理：结合通用风险分析、行业标准和法律法规，将这些要求条目化、可度量化，形成覆盖身份认证、访问控制、日志审计、数据保护等多个维度的通用需求列表；
c) 主要优势：
   1) 效率：避免在每个新项目中从零开始、重复地定义基础安全需求；
   2) 一致性与灵活性：保证所有项目安全需求的覆盖度与一致性，同时允许根据项目风险进行合理的裁剪或增强。

7.1.4.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 特定类型或等级的软件系统；
   2) 整个软件产品线或组织。
b) 实施步骤：
   1) 风险与合规分析：识别系统面临的通用安全风险、行业标准及法律法规要求；
   2) 需求条目化：将分析得到的安全要求，转化为明确、可验证、可度量的需求条目，覆盖身份认证、访问控制、日志审计、数据保护等方面；
   3) 分级与裁剪：根据不同系统的数据敏感度和业务重要性，对安全需求基线进行分级，并允许针对特定项目进行合理的裁剪和增强；
   4) 评审与发布：组织相关方对基线内容进行评审，通过后正式发布，作为后续开发活动的强制性输入。
c) 实施要点：
   1) 核心目标：建立一套标准化的、可复用的基础安全需求集合，确保所有项目满足最低安全标准；
   2) 关键方法：结合风险评估、合规性分析和安全最佳实践，形成制度化的需求规范；
   3) 主要产出：一份正式的安全需求基线文档，作为项目启动时的检查清单和验收标准；
   4) 最终价值：提升安全需求的覆盖度和一致性，避免在每个项目中重复定义基础安全功能，提高开发效率并保障基础安全水平。

7.1.4.2 第三方组件选型与准入控制方法

7.1.4.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在从源头上管控因引入第三方组件而带来的安全风险，建立起对软件供应链的治理能力；
b) 核心原理：在需求和设计阶段建立制度化的“准入”机制，要求在决定使用任何第三方组件前，必须先对其进行全面的安全评估（如已知漏洞、许可证合规性、社区活跃度等）；
c) 主要优势：
   1) 主动性与前瞻性：将安全审查从事后的漏洞扫描，转变为主动的、事前的选型决策，从源头避免引入风险；
   2) 风险管控：降低软件供应链攻击的风险，并为组件出现新漏洞时的应急响应奠定基础。

7.1.4.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 计划在软件中引入的所有开源或商业第三方组件；
   2) 组织的软件供应链管理流程。
b) 实施步骤：
   1) 建立准入标准：定义引入第三方组件前必须满足的安全标准（如许可证合规性、漏洞状况、社区活跃度、可维护性等）；
   2) 制定评估流程：明确组件引入的申请、评估、批准流程，要求提供充分的选型理由和安全评估证据；
   3) 定义使用规范：规定在开发过程中如何安全地使用和配置已批准的组件，包括禁用不安全的功能、遵循安全配置指引等；
   4) 明确应急响应：定义当已引入组件爆出严重漏洞时的应急响应计划。

c) 实施要点：
   1) 核心目标：从源头控制第三方组件带来的安全风险，建立全生命周期的组件安全管理机制；
   2) 关键方法：通过建立制度化的策略和流程，规范化组件的引入、使用和维护；
   3) 主要产出：一份第三方组件安全管理策略文档，包括准入标准、评估流程和应急响应计划；
   4) 最终价值：降低软件供应链攻击的风险，确保第三方组件的安全性与软件系统整体安全目标保持一致。

7.2 软件信息安全性设计阶段支撑技术

7.2.1 设计阶段代码级信息安全支撑技术

7.2.1.1 不可信数据处理与输出编码设计方法

7.2.1.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在设计一套系统性的机制，以防御因不当处理外部输入而引致的各类注入型攻击；
b) 核心原理：要求对所有进入信任边界的数据进行严格验证并要求在输出数据前，使用相应的编码函数中和其代码执行能力；
c) 主要优势：
   1) 系统性与根本性：从架构层面解决一整类数据处理相关的安全问题，而非对特定漏洞的个别修复；
   2) 可靠性：确保所有数据流都经过一致、可靠的安全处理，降低开发人员因疏忽引入漏洞的可能性。

7.2.1.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 所有接收外部输入的代码模块；
   2) 所有将数据输出到外部解释器的代码模块。
b) 实施步骤：
   1) 定义信任边界：明确划分系统的信任与非信任边界，所有跨越边界进入系统的数据都必须被视为不可信；
   2) 设计统一的输入验证框架：设计一个集中的验证组件或框架，对所有不可信输入，进行严格的类型、长度、格式和范围检查；
   3) 设计上下文相关的输出编码：针对不同的输出上下文，设计使用相应的编码函数，以中和数据中的特殊字符，防止其被解释为可执行代码；
   4) 规范化处理流程：规定所有数据处理遵循“输入验证、规范化、业务处理、输出编码”的标准流程。
c) 实施要点：
   1) 核心目标：设计一套完整的机制，以防止诸如注入攻击、跨站脚本等由于不当处理外部数据而引发的漏洞；
   2) 关键方法：遵循对所有不可信输入进行验证以及对输出至外部解释器的数据进行上下文相关编码的原则；

3) 主要产出：输入验证和输出编码的框架设计文档、公共库的接口规范以及开发人员使用指南；
4) 最终价值：系统性地解决与数据处理相关的最主要的安全漏洞类别。

7.2.1.2 异常处理与日志脱敏设计方法

7.2.1.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在设计一套安全的错误处理架构，确保系统在异常状态下不会泄露敏感信息（如堆栈跟踪、配置细节）或进入不安全状态；
b) 核心原理：采用“屏蔽－记录－告警”机制，向客户端返回统一定义的模糊错误码，同时在服务端日志中记录详细堆栈，并利用正则替换或哈希算法对日志中的个人敏感信息进行实时脱敏；
c) 主要优势：
   1) 信息防泄露：阻断攻击者通过错误回显刺探系统内部架构或逻辑漏洞的路径；
   2) 合规审计：确保日志记录符合个人信息保护相关的法律法规要求。

7.2.1.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 全局异常捕获模块；
   2) 系统日志记录组件及审计模块。
b) 实施步骤：
   1) 定义错误码规范：设计对用户无害但便于排查的统一错误响应格式；
   2) 设计全局拦截器：在框架层设计拦截器以捕获所有未处理异常，强制转换为标准错误响应；
   3) 配置脱敏规则：识别日志中的敏感字段（如手机号、卡号、密码），配置自动化脱敏策略；
   4) 故障安全设计：确保安全机制（如鉴权）在异常发生时默认为拒绝访问状态。
c) 实施要点：
   1) 核心目标：消除因异常处理不当导致的信息泄露，确保日志数据的隐私安全性；
   2) 关键方法：统一全局异常处理逻辑，并集成基于规则的日志自动化脱敏组件；
   3) 主要产出：异常处理设计规范文档、统一错误码表及日志脱敏规则集；
   4) 最终价值：即使在系统故障时也能维持信息保密性，并为安全审计提供合规、清洁的日志源。

7.2.2 设计阶段接口级信息安全支撑技术

7.2.2.1 访问凭证与会话管理设计方法

7.2.2.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在设计一个安全、健壮的身份认证与会话管理体系，以确保只有合法的用户才能访问系统，并维持会话的完整性和安全性；
b) 核心原理：综合运用密码学技术（如数字签名、哈希）和安全协议设计，来创建、分发、验证和管理访问凭证（如无状态令牌），并设计防重放、防伪造、防劫持的机制；
c) 主要优势：

1) 可靠性与安全性：通过严谨的密码学设计，有效防止未经授权的访问、会话劫持和重放攻击；
2) 伸缩性：基于签名的无状态令牌方案，具有更好的伸缩性和跨域能力，适用于现代分布式和微服务架构。

7.2.2.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 需要进行用户身份认证和状态保持的软件系统；
   2) 涉及接口调用的客户端与服务器端交互流程。
b) 实施步骤：
   1) 凭证方案选择：根据安全级别和场景需求，选择合适的凭证类型及其生成、分发和验证机制；
   2) 会话生命周期设计：设计会话的创建、维持、刷新和销毁机制；
   3) 凭证安全存储设计：为客户端和服务器端设计安全的凭证存储方案，防止凭证在存储和传输过程中被窃取；
   4) 凭证防滥用设计：设计防止凭证重放、伪造和劫持的机制。
c) 实施要点：
   1) 核心目标：设计一套健壮、安全的凭证和会话管理体系，确保用户身份的正确识别和会话状态的完整性；
   2) 关键要素：遵循无状态、防窃取、防滥用的原则，综合运用密码学和安全协议进行设计；
   3) 主要产出：访问凭证与会话管理设计方案，包括流程图、状态机和关键算法说明；
   4) 最终价值：构建系统访问控制的第一道防线，有效防止未经授权的访问和会话劫持攻击。

7.2.2.2 端到端数据流加密与密钥交换技术

7.2.2.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在设计一个全面的数据加密方案，确保敏感信息在跨越网络传输或持久化存储时，始终保持机密性和完整性；
b) 核心原理：采用分层设计思想，在传输层和应用层（针对特定字段）应用加密措施，并设计安全的密钥交换与管理机制，确保通信双方能够安全地协商和保护会话密钥；
c) 主要优势：
   1) 纵深防御：即使传输层加密被攻破，应用层对核心敏感字段的加密依然能提供保护；
   2) 系统性解决：系统性地解决数据在流动和静止状态下的安全问题，有效防止数据被窃听和篡改，满足合规性要求。

7.2.2.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 跨越不信任网络边界的数据流；
   2) 接口中传输的敏感数据字段。
b) 实施步骤：
   1) 确定加密范围：识别需要加密的通信链路（传输层加密）和具体数据字段（应用层加密）；

2) 选择加密算法与协议：根据安全要求和性能考量，选择标准化的、强度足够的加密算法（对称、非对称）和安全传输协议；
3) 设计密钥交换与管理：设计安全、高效的密钥协商和分发机制，确保通信双方能够安全地获取会话密钥；
4) 定义数据封装格式：设计加密数据的封装格式，包含必要的元数据（如算法标识、初始化向量），以便接收方正确解密。

c) 实施要点：
1) 核心目标：设计端到端的数据加密方案，保障数据在传输过程中的机密性和完整性；
2) 关键方法：分层设计，将传输层加密与应用层加密相结合；
3) 主要产出：一份数据流加密设计方案，详细说明所选协议、算法、密钥管理流程和数据格式；
4) 最终价值：防止数据在网络传输中被窃听和篡改，满足数据保护的合规性要求。

7.2.3 设计阶段模型级信息安全支撑技术

7.2.3.1 基于形式化方法的安全属性证明技术

7.2.3.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过严谨的数学方法，证明系统的关键安全机制设计（如认证协议、访问控制模型）满足特定的安全属性；
b) 核心原理：将安全属性和系统设计用精确的数学或逻辑语言进行形式化描述，然后利用模型检测器等自动化工具进行穷尽性状态空间搜索或逻辑推演，以检查安全属性是否恒成立；
c) 主要优势：
1) 严谨性与深度：提供数学确定性的保证，能够发现常规测试手段极难触及的深层次设计缺陷；
2) 精准定位：验证失败时，工具能提供导致失败的具体路径（反例），帮助设计者定位并修复问题。

7.2.3.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
1) 系统的关键安全机制设计，如访问控制模型、认证协议、加密流程；
2) 形式化或半形式化的系统模型。

b) 实施步骤：
1) 形式化定义属性：将期望的安全属性用精确的逻辑语言或数学语言进行描述；
2) 建立系统模型：将待验证的安全机制设计抽象为一个形式化模型；
3) 应用验证工具：使用模型检测器、定理证明器等形式化方法工具，自动或半自动地检查模型是否在所有可能的情况下都满足已定义的安全属性；
4) 分析反例：如果验证失败，工具会提供一个不满足属性的具体执行路径，设计人员据此修正设计缺陷。

c) 实施要点：
1) 核心目标：通过数学上严谨的推理，证明关键安全设计不存在逻辑层面的缺陷；
2) 关键方法：运用形式化方法对抽象模型进行穷尽性分析，而非基于样本的测试；

3) 主要产出：一份形式化验证报告，证明（或证伪）设计满足特定的安全属性，并提供反例用于调试；
4) 最终价值：在设计阶段发现协议或逻辑中深层次、难以通过常规测试发现的漏洞，极大地提升核心安全机制的可靠性。

7.2.3.2 攻击面分析与最小化设计方法

7.2.3.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过系统性地识别和评估系统的所有“入口点”，来主动地、有策略地减少系统暴露给潜在攻击者的可攻击范围；
b) 核心原理：通过全面枚举所有外部实体可与系统交互的通道（API、端口、输入框等），对它们进行分类、评估风险，并审视其存在的必要性，最终设计出移除或加固入口的方案；
c) 主要优势：
   1) 主动性与根本性：在设计阶段从宏观架构层面精简系统，是一种高效的主动防御策略，而非被动修补；
   2) 降低风险与成本：从根本上降低系统被攻击的概率，简化后续的安全防护和监控工作，使防御资源更加集中。

7.2.3.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 软件的体系架构模型；
   2) 系统的所有入口点和交互点。
b) 实施步骤：
   1) 识别入口点：系统性地识别并列出所有外部实体可以与系统交互的通道和接口；
   2) 分类和评估：对识别出的入口点进行分类，并评估其暴露风险；
   3) 审查设计决策：全面审查设计，判断每个入口点的必要性，并寻找减少或加固暴露点的方法；
   4) 输出最小化方案：形成一份攻击面最小化的设计改进建议。
c) 实施要点：
   1) 核心目标：在设计阶段通过体系化的方式，主动移除或加固不必要的暴露点，从而最小化系统的整体攻击面；
   2) 关键方法：系统性地枚举、分类和评估所有潜在的攻击入口，并审查其存在的必要性；
   3) 主要产出：一份攻击面清单及对应的最小化设计方案；
   4) 最终价值：从根本设计上降低系统被攻击的风险，是一种高效的主动防御策略。

7.2.4 设计阶段软件系统级信息安全支撑技术

7.2.4.1 基于信任域的架构隔离与访问控制技术

7.2.4.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过在系统架构层面建立多层安全屏障（“纵深防御”），来限制安全事件的影响范围，提升系统的弹性和韧性；

b) 核心原理：采用“分而治之”思想，根据组件功能和数据重要性将系统划分为不同的“信任域”，在域之间设计强制性的技术隔离边界，并遵循“最小权限”原则精确定义跨域通信策略；

c) 主要优势：

1) 遏制横向移动：在单个组件可能被攻破的情况下，通过架构隔离可防止攻击向其他信任域扩散；

2) 保护核心资产：将潜在的系统性故障限制为局部功能失效，有效保护系统核心资产；

3) 提升系统韧性：是构建大型复杂系统、确保其被攻击后仍能维持核心功能的关键架构模式。

7.2.4.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 软件系统的部署架构和网络拓扑；

2) 不同信任等级的应用程序模块、进程或数据存储区。

b) 实施步骤：

1) 划分信任域：根据组件的功能和数据敏感度，在系统架构中划分出不同的信任区域；

2) 设计隔离边界：为不同信任域之间设计强制性的隔离边界，选择具体实现技术；

3) 定义通信策略：明确规定跨越隔离边界的合法通信规则，遵循最小权限原则，仅允许必要的交互；

4) 设计监控与阻断：设计在隔离边界上部署监控和策略执行点的方案，用于审计和阻断违规访问。

c) 实施要点：

1) 核心目标：通过划分和隔离，将系统分割成更小的、独立的单元，限制安全事件的影响范围；

2) 关键方法：应用“纵深防御”和“最小权限”原则，在系统架构层面建立多层安全屏障；

3) 主要产出：一份包含信任域划分、隔离技术选型和跨域访问控制策略的系统安全架构设计文档；

4) 最终价值：有效遏制攻击的横向移动，即使单个组件被攻破，也能保护系统的其余部分不受影响，显著提升系统的弹性。

7.2.4.2 密钥管理基础设施集成与自动化技术

7.2.4.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在为软件系统中使用的所有加密密钥和数字证书，设计一个安全、可靠的全生命周期管理方案；

b) 核心原理： 建立一个集中化、自动化的密钥管理体系，将密钥本身的管理（生成、存储、轮换、销毁等）与使用密钥的业务应用逻辑分离开来，并利用专业技术保障其安全性；

c) 主要优势：

1) 专业性与健壮性：将复杂的密钥管理任务从普通开发人员手中剥离，交由专业系统处理，避免因管理不善导致整个加密体系失效，从而为系统的数据机密性与完整性保护提供基础支撑；

2) 核心支撑：为系统的数据机密性和完整性保护提供了最核心、最坚实的基础支撑。

7.2.4.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 软件系统中使用的所有加密密钥和数字证书；
   2) 密钥的生成、分发、存储、使用、轮换、归档和销毁等所有环节。
b) 实施步骤：
   1) 密钥分类与策略定义：根据密钥的用途和重要性进行分类，并为每类密钥定义不同的管理策略；
   2) 安全生成与分发设计：设计使用经过认证的密码学模块生成密钥的流程，以及将密钥安全地分发给合法使用方的机制；
   3) 安全存储与使用设计：设计密钥的存储方案，确保其机密性和完整性（如使用硬件安全模块或可信执行环境），并设计安全的密钥调用接口；
   4) 轮换与销毁设计：设计密钥的定期轮换或按需轮换流程，以及在密钥过期或泄露后进行安全销毁和替换的机制。
c) 实施要点：
   1) 核心目标：确保密钥在整个生命周期内都得到妥善保护，防止因密钥管理不善导致整个加密体系失效；
   2) 关键方法：建立集中化、自动化的密钥管理体系，将密钥本身与使用密钥的应用逻辑分离；
   3) 主要产出：一份密钥全生命周期管理设计方案，详细描述密钥管理的策略、流程和技术实现；
   4) 最终价值：为系统的加密功能提供坚实的基础，是保障数据机密性和完整性的核心支撑。

7.3 软件信息安全性实现阶段支撑技术

7.3.1 实现阶段代码级信息安全支撑技术

7.3.1.1 安全代码人工审查方法

7.3.1.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过人工审查的方式，发现自动化工具难以识别的深层次安全缺陷（如业务逻辑漏洞），同时促进团队安全知识的共享；
b) 核心原理：组织具备安全背景的人员，对照安全规范和威胁模型，有针对性地对高风险源代码进行深度分析，侧重于检查业务逻辑严密性、权限控制正确性及加密算法的恰当使用；
c) 主要优势：
   1) 深度与精准度：能更好地理解复杂业务逻辑，有效发现设计实现偏差、逻辑绕过等深层次漏洞；
   2) 知识传递：审查过程中的讨论和修复，有助于统一团队对安全规范的理解，是自动化扫描的必要补充。

7.3.1.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 新开发或有重大修改的源代码，特别是涉及安全功能的模块；
   2) 开发人员提交的代码变更请求。

b) 实施步骤：
   1) 确定审查范围：根据代码变更的风险等级和重要性，确定审查的重点区域；
   2) 分配审查任务：将代码分配给具备安全知识的同行或安全专家进行审查；
   3) 执行审查：审查人员对照安全编码规范，重点检查是否存在常见漏洞、逻辑缺陷和不当的加密使用；
   4) 跟踪与修复：记录发现的问题，并将其反馈给开发人员进行修复，验证修复方案的正确性后方可合入代码库。
c) 实施要点：
   1) 核心目标：通过人工审查，发现自动化工具难以检测到的业务逻辑漏洞和复杂的设计缺陷；
   2) 关键方法：结合开发人员的业务理解和安全专家的攻击者视角，对代码进行深度分析；
   3) 主要产出：一份包含具体漏洞位置、风险描述和修复建议的代码审查报告；
   4) 最终价值：作为自动化测试的有效补充，显著提升代码质量，并促进团队内部安全知识的传递和共享。

7.3.1.2 软件成分分析技术

7.3.1.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在自动化地识别、管理和监控软件项目中使用的第三方组件（开源或商业）所带来的安全风险；
b) 核心原理：通过扫描项目依赖生成一份SBOM，然后将其与持续更新的公共漏洞数据库和许可证信息库进行自动比对，从而发现已知安全漏洞和许可证合规风险；
c) 主要优势：
   1) 可见性：能够清晰地呈现软件的构成，提升手动管理第三方组件的效率与准确性；
   2) 快速响应能力：当第三方组件出现新漏洞时能及时获得告警，实现对供应链风险的快速评估与修复。

7.3.1.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 项目的构建产物或源代码依赖清单；
   2) 项目中使用的所有开源及商业第三方组件。
b) 实施步骤：
   1) 生成物料清单：通过扫描构建过程或依赖管理文件，自动生成一份详细的SBOM，列出所有直接和间接依赖的第三方组件及其版本；
   2) 漏洞与许可证匹配：将清单中的每个组件与公开的漏洞数据库和许可证数据库进行比对，识别出已知的安全漏洞和许可证合规风险；
   3) 风险评估与上报：根据漏洞的严重性和组件在项目中的使用方式，评估实际风险，并向开发团队发出警报；
   4) 指导修复：提供修复建议，如升级到无漏洞版本、替换组件或应用安全补丁。
c) 实施要点：
   1) 核心目标：自动化地识别和管理软件供应链中由第三方组件引入的安全风险；
   2) 关键方法：该技术通过构建准确的软件物料清单，并将其与持续更新的风险情报进行关联实现；

3) 主要产出：一份软件成分分析报告，包含风险组件列表、相关漏洞详情和许可证合规性问题；
4) 最终价值：提供对软件供应链的可见性，使团队能够快速响应第三方组件新爆出的漏洞，降低供应链攻击的风险。

7.3.2 实现阶段接口级信息安全支撑技术

7.3.2.1 接口实现的安全单元测试方法

7.3.2.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在将安全验证工作前移至开发人员的编码环节，倡导开发人员在完成功能编码后，依据安全契约和规范，主动编写安全测试用例进行自测；
b) 核心原理：在编写业务功能代码的同时，构造包含恶意输入的测试用例，专门用于验证代码单元是否正确处理了认证、授权、输入验证、输出编码等安全逻辑；
c) 主要优势：
   1) 高效率与低成本：在漏洞引入的初期阶段即被发现与修复，避免了缺陷流入后续环节，成本最低；
   2) 即时反馈：集成到持续集成流水线中，可实现对每次代码提交的自动化、即时安全反馈，形成快速有效的代码级安全防线。

7.3.2.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 开发人员编写的单个接口实现函数或方法；
   2) 安全功能相关的代码单元。
b) 实施步骤：
   1) 编写安全测试用例：针对接口的安全契约和安全编码规范，编写单元测试用例，覆盖认证、授权、输入验证和输出编码等逻辑；
   2) 模拟攻击输入：构造包含恶意载荷的测试数据（如SQL注入字符串、跨站脚本向量），验证输入验证逻辑是否有效；
   3) 验证边界条件：测试处理异常输入时的行为，确保程序不会崩溃或产生非预期结果；
   4) 集成到构建流程：将安全单元测试集成到持续集成流程中，确保每次代码提交都自动执行安全检查。
c) 实施要点：
   1) 核心目标：由开发人员在开发过程中，第一时间验证其编写的接口代码是否满足基本的安全要求；
   2) 关键方法：将安全考虑融入单元测试，实施“测试驱动安全”；
   3) 主要产出：一套可重复执行的接口安全单元测试用例集；
   4) 最终价值：在开发的最早阶段发现并修复安全缺陷，成本最低，效率最高，是实现“安全左移”的关键实践。

7.3.2.2 接口故障注入与熔断实现技术

7.3.2.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在开发阶段通过代码手段主动模拟接口依赖服务的故障（如延迟、中断、脏数据），验证系统的容错能力与熔断机制的有效性；
b) 核心原理：利用模拟对象框架或服务网格故障注入功能，拦截接口调用并人为引入网络抖动或错误响应，触发应用内的重试、降级或熔断逻辑；
c) 主要优势：
   1) 韧性验证：在非生产环境下低成本地验证系统对级联故障的防御能力；
   2) 逻辑覆盖：覆盖常规测试难以触发的异常分支代码。

7.3.2.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 微服务架构中的远程调用客户端；
   2) 涉及外部第三方接口集成的代码模块。
b) 实施步骤：
   1) 识别关键依赖：梳理所有核心业务强依赖的外部接口与服务；
   2) 编写注入策略：在集成测试代码中配置故障模拟策略（如让接口响应延迟特定时间或返回乱码）；
   3) 实现降级逻辑：编写备用逻辑代码，确保在主接口失效时提供兜底数据或友好的服务降级；
   4) 验证熔断状态：监控熔断器状态机的转换是否符合预期。
c) 实施要点：
   1) 核心目标：验证接口在极端或错误条件下的健壮性，确保故障不会在系统内蔓延；
   2) 关键方法：结合模拟技术与故障注入思想进行破坏性测试；
   3) 主要产出：包含故障模拟场景的集成测试套件及接口降级策略实现代码；
   4) 最终价值：提升系统的反脆弱能力，防止因单个接口异常导致整个分布式系统雪崩。

7.3.3 实现阶段模型级信息安全支撑技术

7.3.3.1 设计模型到代码实现的一致性校验技术

7.3.3.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在确保经过审慎分析和设计的安全模型（如威胁模型、访问控制模型）被忠实、完整地反映在最终的源代码实现中，防止设计与实现脱节；
b) 核心原理：通过建立从设计文档到代码模块之间的可追溯链接，进行双向的一致性校验：正向核实设计的每项措施是否都已实现，逆向审查代码逻辑是否与设计规约一致；
c) 主要优势：
   1) 确保设计实现：通过系统性比对和审查，确保安全理念从抽象模型到具体代码的准确传递；
   2) 有效验收：防止因开发过程中的误解、遗漏或随意变更导致的安全设计失效，是实现阶段对安全设计成果的有效验收手段。

7.3.3.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 源代码中对业务逻辑、状态机、权限控制等关键模型的实现；

2) 需求和设计阶段创建的威胁模型、攻击树、访问控制模型等。

b) 实施步骤：

1) 追溯映射：将代码中的关键实现模块，与设计文档中的相应模型进行一一对应；
2) 逻辑比对：人工审查代码实现，判断其逻辑是否与设计模型所描述的行为、状态转换和安全约束完全一致；
3) 检查遗漏：核实设计模型中定义的所有安全控制措施（如威胁缓解措施）是否都在代码中得到了正确和完整地实现；
4) 记录偏差：记录所有代码实现与设计模型不一致的地方，评估其安全影响，并提交给开发团队进行修正。

c) 实施要点：

1) 核心目标：确保最终的代码实现没有偏离或遗漏设计阶段经过审慎分析的安全模型；
2) 关键方法：通过正向追溯和逆向分析，建立设计与实现之间的双向链接，进行一致性校验；
3) 主要产出：一份模型实现一致性审查报告，指出设计与实现之间的偏差项；
4) 最终价值：保证安全设计的初衷被忠实地转化为可执行的代码。

7.3.3.2 安全模型驱动的策略代码生成技术

7.3.3.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在利用设计阶段构建的安全模型（如访问控制矩阵），自动化生成可执行的安全策略代码或配置文件，减少人工配置错误；

b) 核心原理：采用“策略即代码”理念，编写转换引擎，将结构化的安全模型（如统一建模语言图表）直接编译为防火墙规则、身份访问管理策略或数据库脚本；

c) 主要优势：

1) 一致性保障：消除人工翻译模型到实施过程中的偏差；
2) 变更管理：安全策略的变更可通过修改模型并重新生成代码来实现，纳入版本控制。

7.3.3.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 复杂的权限管理系统；
2) 微服务间的网络隔离策略。

b) 实施步骤：

1) 模型数字化：确保设计模型以机器可读的格式存储；
2) 开发转换工具：编写脚本将模型元素映射为目标系统的配置语法；
3) 集成构建流水线：在持续集成过程中自动执行转换工具，生成最新的安全配置；
4) 校验与部署：对生成的策略代码进行语法检查后，自动部署至测试或生产环境。

c) 实施要点：

1) 核心目标：实现安全模型到技术实现的自动化流转，消除手动配置带来的低级错误；
2) 关键方法：建立模型到代码的自动化映射与编译机制；
3) 主要产出：自动生成的安全策略文件、配置脚本及转换工具链；
4) 最终价值：极大地提高安全配置的准确性与效率，实现安全策略的标准化与版本化管理。

7.3.4 实现阶段软件系统级信息安全支撑技术

7.3.4.1 基础设施即代码静态合规性分析技术

7.3.4.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在确保软件赖以运行的基础设施环境（如操作系统、数据库、容器）的配置是安全的，消除因配置不当而引发的安全风险；
b) 核心原理：定义一套标准化的“安全配置基线”，然后通过自动化工具扫描“基础设施即代码”文件或线上环境，将当前配置与基线进行比对，自动发现不合规项；
c) 主要优势：
   1) 自动化与持续性：高效地增强软件运行环境的安全性，消除常见的配置弱点（如默认口令、不必要端口）；
   2) 减少攻击面：有助于减小系统的整体攻击面，防止攻击者利用底层环境的配置错误绕过应用层防御。

7.3.4.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 软件运行所依赖的操作系统、中间件、数据库、容器平台等基础设施；
   2) 以代码形式管理的基础设施配置。
b) 实施步骤：
   1) 定义配置基线：基于行业最佳实践和组织安全要求，为各类基础设施组件定义一套标准的安全配置基线；
   2) 自动化扫描：使用自动化工具定期扫描线上或预发布环境的基础设施配置，将其与已定义的基线进行比对；
   3) 生成差异报告：输出配置现状与安全基线之间的差异项，并按风险等级进行排序；
   4) 驱动修复闭环：将发现的配置弱点作为任务分配给运维或平台团队进行修复，并再次扫描以验证修复效果。
c) 实施要点：
   1) 核心目标：确保软件运行环境的配置符合安全标准，消除因配置不当导致的安全隐患；
   2) 关键方法：将安全配置要求标准化、文档化，并通过自动化工具进行持续性的合规性检查；
   3) 主要产出：基础设施配置安全合规性报告，以及待修复的配置项列表；
   4) 最终价值：硬化软件的运行环境，减少系统的攻击面，防止攻击者利用常见的配置错误来入侵系统。

7.3.4.2 容器镜像最小化与加固技术

7.3.4.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在构建阶段对软件运行载体（容器镜像）进行精简与硬化，移除所有非必要的组件与漏洞，减小攻击面；
b) 核心原理：采用多阶段构建技术仅保留编译产物，使用微型基础镜像，并移除命令行终端、包管理器等潜在攻击工具，同时禁用特权用户运行；
c) 主要优势：

1) 攻击面收敛：即使攻击者入侵容器，因缺乏系统工具，其横向移动难度剧增；
2) 漏洞消减：显著减少操作系统级依赖库的数量，从而降低已知漏洞的检出率。

7.3.4.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 所有基于容器技术交付的微服务与应用组件；
   2) 容器描述文件及构建流程。
b) 实施步骤：
   1) 基础镜像选型：强制使用官方认证的、最小化的基础镜像；
   2) 多阶段构建：在描述文件中分离构建环境与运行环境，丢弃编译器与源码；
   3) 权限降级：在构建脚本末尾创建专用低权限用户，并指定以该用户启动进程；
   4) 内容信任签名：构建完成后，使用私钥对镜像层进行签名，确保镜像未被篡改。
c) 实施要点：
   1) 核心目标：交付极其精简且默认安全的运行环境，最大程度限制攻击者在容器内的活动能力；
   2) 关键方法：实施最小化构建策略、非特权运行及镜像签名机制；
   3) 主要产出：经过加固的、体积最小化的容器镜像及标准化的构建模板；
   4) 最终价值：从基础设施层面构建坚固的防御壁垒，使应用具备内生的抗渗透能力。

7.4 软件信息安全性测试阶段支撑技术

7.4.1 测试阶段代码级信息安全支撑技术

7.4.1.1 静态应用安全测试方法

7.4.1.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：在不实际运行程序的情况下，通过分析其源代码或字节码来发现潜在的安全漏洞；
b) 核心原理：利用自动化分析引擎，通过模式匹配、数据流分析（污点分析）等技术，追踪不可信数据从输入源到危险操作点的流动路径，从而发现注入类、信息泄露等编码缺陷；
c) 主要优势：
   1) 早期介入：可以在编码阶段或CI/CD流水线中非常早地执行，以极低成本发现并修复问题；
   2) 高覆盖率：能够一次性扫描完整的代码，包括那些在动态测试中可能难以触发的逻辑分支。

7.4.1.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 软件项目的完整代码库；
   2) 持续集成流水线。
b) 实施步骤：
   1) 配置扫描规则：根据应用的技术栈和安全需求，选择并定制化静态分析的规则集；
   2) 执行代码分析：利用自动化分析引擎，对“静止”的代码进行扫描，通过语法分析、数据流分析等技术，识别代码中与已知漏洞模式匹配的缺陷；

3) 结果聚合与去重：对扫描结果进行处理，合并重复发现，并与历史扫描结果进行对比；
4) 审核与分诊：由安全人员或开发人员审核扫描报告，识别并过滤误报，对确认的漏洞根据其严重性和可利用性进行优先级排序。

c) 实施要点：
1) 核心目标：在不运行程序的情况下，通过分析代码来发现潜在的安全漏洞；
2) 关键方法：应用模式匹配、污点分析和控制流分析等技术，在整个代码库中寻找不安全的编码实践；
3) 主要产出：一份包含漏洞位置、类型、风险等级和修复建议的静态代码分析报告；
4) 最终价值：能够在开发周期的早期，以较低成本大范围地发现编码层面的安全缺陷，有效减少流入后期阶段的漏洞数量。

7.4.1.2 面向业务逻辑的白盒安全测试方法

7.4.1.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在深入代码实现层面，发现那些与特定业务流程紧密相关、自动化工具难以感知的逻辑漏洞（如支付漏洞、越权操作）；
b) 核心原理：测试人员在具备代码完全可见性的前提下，结合对业务功能的理解，主动设计并执行“非正常”但可能的业务操作序列，验证在边缘或恶意场景下代码的行为是否符合安全预期；
c) 主要优势：
1) 深度与针对性：能够精准地识别根植于复杂业务逻辑中的高价值缺陷，可有效弥补自动化扫描工具的不足；
2) 保障核心业务：有助于保障支付、订单流转等核心业务流程的安全性，防止因逻辑漏洞造成直接经济损失。

7.4.1.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
1) 实现复杂业务逻辑或安全机制的代码模块；
2) 对自动化工具不可见的、与业务流程紧密相关的安全漏洞。

b) 实施步骤：
1) 理解业务与安全需求：深入理解被测模块的业务功能和相关的安全需求；
2) 设计测试场景：基于对业务的理解，设计可能绕过安全控制的特定场景和异常操作序列；
3) 编写测试代码或手动执行：通过白盒测试的方式，直接调用内部函数或构造特定对象状态，来验证在这些边缘或恶意场景下，代码的行为是否符合安全预期；
4) 验证状态与权限：重点检查代码在处理关键操作时，是否对用户的身份、角色、权限和对象的所有权进行了正确且充分的校验。

c) 实施要点：
1) 核心目标：以白盒视角，深入代码实现，发现因业务逻辑复杂性而产生的安全漏洞；
2) 关键方法：将业务理解与代码分析相结合，模拟攻击者利用业务流程缺陷进行攻击的思维方式；
3) 主要产出：一份业务逻辑安全漏洞报告，描述可被利用的逻辑缺陷场景和复现步骤；
4) 最终价值：有效发现自动化扫描工具无法检测的业务逻辑漏洞和权限绕过问题。

7.4.2 测试阶段接口级信息安全支撑技术

7.4.2.1 模糊测试技术

7.4.2.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过向应用程序提供大量非预期的、畸形的输入数据，来发现其在处理异常情况时的健壮性问题和未知安全漏洞；

b) 核心原理：利用自动化工具生成大量的、半随机或基于规则变异的输入，并持续注入目标接口或文件解析器中，同时监控应用程序是否出现崩溃、服务拒绝、内存泄漏等非预期行为；

c) 主要优势：

1) 发现未知漏洞：作为一种探索性测试，不依赖已知漏洞模式，能有效发现边界条件缺陷、内存错误等常规测试难以预料的问题；

2) 提升鲁棒性：能有效发现导致程序崩溃、拒绝服务的缺陷，提升系统的整体稳定性和健壮性。

7.4.2.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 应用程序的接口；

2) 文件解析器、协议解析器等处理复杂输入的模块。

b) 实施步骤：

1) 确定测试目标：选择要进行模糊测试的接口或数据输入点；

2) 生成畸形输入：自动化工具生成大量非预期的、半合法的或完全随机的输入数据；

3) 持续发送输入：将生成的畸形数据持续、高速地发送给测试目标；

4) 监控应用响应：密切监控应用程序是否出现崩溃、服务拒绝、内存泄漏或未处理的异常等健壮性问题。

c) 实施要点：

1) 核心目标：通过提供非预期的输入，来发现软件在处理异常数据时的健壮性问题和未知漏洞；

2) 关键方法：自动化生成并注入大量畸形数据；

3) 主要产出：导致系统崩溃或异常的特定输入数据，以及相关的崩溃日志或内存转储；

4) 最终价值：能够发现开发人员和常规测试人员难以预料到的未知漏洞和稳定性问题，提升系统的鲁棒性。

7.4.2.2 交互式应用安全测试技术

7.4.2.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在结合SAST和DAST的优点，在应用运行时提供高精度的漏洞检测；

b) 核心原理：在应用程序的运行环境中部署一个“探针”，该探针能同时监控外部请求和应用内部的代码执行路径。当外部测试触发应用时，探针可从内部实时确认一个不安全的数据流是否真实存在并构成漏洞；

c) 主要优势：

1) 极高准确性：能同时看到外部攻击请求和内部代码执行路径，可以精确验证漏洞，基本消除误报；

2) 丰富上下文：发现漏洞时能直接报告问题代码行号、完整请求和数据流路径，为漏洞复现与修复提供全面的依据。

7.4.2.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 运行在测试环境中的网络应用和API服务；
   2) 需要在运行时上下文中进行分析的复杂应用。
b) 实施步骤：
   1) 插桩部署：在应用服务器的运行环境中部署一个代理或探针；
   2) 触发应用功能：通过手动测试、自动化功能测试或DAST扫描来正常地使用和访问应用功能；
   3) 内部监控与分析：代理在应用内部监控代码执行、数据流动、库函数调用等，当外部测试流量触发应用时，代理能够实时分析内部的代码行为；
   4) 关联并报告漏洞：当代理检测到不安全的数据流或危险的函数调用时，它会将此内部行为与触发该行为的外部请求关联起来，并报告一个精确到代码行级的漏洞。
c) 实施要点：
   1) 核心目标：结合DAST的外部视角和SAST的内部视角，在应用运行时进行安全分析；
   2) 关键方法：通过在应用内部插桩，获取运行时上下文，从而精确地验证漏洞是否存在，减少误报；
   3) 主要产出：一份高精度的漏洞报告，包含完整的请求和响应对、漏洞触发的内部代码路径和行号；
   4) 最终价值：极大地提高了漏洞检测的准确性，减少了误报分析的成本，并能提供丰富的上下文信息帮助开发人员快速定位和修复问题。

7.4.2.3 基于权限模型的越权攻击模拟技术

7.4.2.3.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在系统性地验证应用的访问控制机制是否被正确、完整地实现，以防止水平越权（访问同级用户资源）和垂直越权（用低权限执行高权限操作）漏洞；
b) 核心原理：基于设计阶段定义的权限矩阵或访问控制模型，构造全面的测试用例集，使用不同角色和身份的用户账户，系统性地尝试访问其权限内和权限外的所有功能和资源，并验证系统响应是否符合预期；
c) 主要优势：
   1) 系统性与覆盖度：是一种基于模型的、矩阵式的测试方法，力求覆盖所有角色与所有资源之间的访问关系组合，而非随机探索；
   2) 保障数据隔离：能够确保多用户、多租户系统的数据隔离和安全性，是保障此类系统的核心测试手段。

7.4.2.3.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 系统中所有涉及权限判断和数据隔离的接口；
   2) 基于角色、属性或多租户的访问控制模型。

b) 实施步骤：
   1) 梳理权限矩阵：明确不同角色或用户类型对不同功能和数据资源的访问权限；
   2) 构造测试场景：设计一系列测试用例，模拟不同身份的用户尝试执行其权限范围内和范围外的操作；
   3) 执行越权测试：使用低权限用户账户，尝试访问高权限用户才能访问的接口或功能，使用普通用户账户，尝试访问或修改属于其他同级用户的数据资源；
   4) 验证测试结果：检查每次尝试的返回结果，确认所有未经授权的访问都被系统正确地拒绝，且已授权的访问能够成功。
c) 实施要点：
   1) 核心目标：验证系统的访问控制机制是否被正确且完整地实现，不存在任何绕过或提权的缺陷；
   2) 关键方法：基于权限模型，系统性地、矩阵式地进行越权尝试；
   3) 主要产出：一份访问控制测试报告，详细记录所有测试用例的执行结果，并高亮显示失败用例；
   4) 最终价值：确保系统的访问控制机制坚实有效，防止数据泄露和未授权操作。

7.4.3 测试阶段模型级信息安全支撑技术

7.4.3.1 业务逻辑并发测试技术

7.4.3.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在发现那些只有在高并发条件下才会暴露的、由于竞争条件而导致的业务逻辑漏洞（如商品超卖、资金计算错误）；
b) 核心原理：识别出业务流程中涉及共享资源或状态更新的非原子性关键操作，然后使用自动化工具在极短时间窗口内模拟大量用户同时执行这些操作，人为制造时序冲突，以验证系统在并发压力下的数据一致性；
c) 主要优势：
   1) 发现隐蔽漏洞：能够发现常规功能测试无法触及的严重漏洞，这些漏洞在低并发时完美隐藏，但在生产高负载时可能导致严重后果；
   2) 模拟真实压力：通过模拟真实的线上压力场景，能够提前暴露并修复这些隐蔽且危害巨大的缺陷。

7.4.3.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 设计阶段定义的、涉及共享资源或状态的业务流程模型；
   2) 上述模型的代码实现，特别是多步骤、非原子性的业务逻辑。
b) 实施步骤：
   1) 识别关键业务节点：分析业务流程，找出可能存在竞争条件的关键步骤；
   2) 设计并发场景：设计模拟多个用户或请求在同一时刻或极短时间窗口内，并发执行同一个或相互关联的业务操作的场景；
   3) 实施并发请求：使用自动化测试工具，同时向服务器发送大量并发请求，触发设计的并发场景；

4) 验证数据一致性与状态正确性：在测试后，检查后台数据是否依然保持一致和正确，是否存在数据错乱或状态异常。

c) 实施要点：

1) 核心目标：发现在高并发条件下，由于竞争条件而导致的业务逻辑漏洞；
2) 关键方法：通过模拟高并发流量，暴露时序相关的程序缺陷；
3) 主要产出：一份并发测试报告，描述能够复现数据不一致或逻辑错误的并发场景；
4) 最终价值：发现并修复那些在单用户测试环境下无法暴露的、由并发执行引起的严重安全漏洞，确保在高负载下业务逻辑的正确性。

7.4.3.2 状态机逻辑遍历测试技术

7.4.3.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在验证核心业务对象的生命周期管理是否严格遵循了预先设计的状态机模型，防止因状态转换混乱而导致的安全问题；

b) 核心原理：以业务对象的状态机图为测试依据，系统性地设计测试用例，覆盖所有合法的状态转换路径，并重点测试所有非法的状态转换路径，验证系统能否正确拒绝非法转换。

c) 主要优势：

1) 逻辑严密性：通过将测试与设计模型直接挂钩，可以确保代码实现与设计意图的高度一致；
2) 保障业务完整性：能够系统性地发现逻辑漏洞，保障核心业务流程的完整性和严谨性，防止攻击者通过操纵对象状态来绕过业务规则。

7.4.3.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 具有明确状态定义和转换规则的业务对象；
2) 设计阶段定义的状态机模型。

b) 实施步骤：

1) 建模或获取状态机：依据设计文档或代码实现，梳理出被测对象的状态图，包括所有状态和合法的状态转换路径；
2) 设计遍历路径：设计测试用例，覆盖所有合法的状态转换路径，确保系统按预期工作；
3) 设计非法转换：设计测试用例，尝试执行所有非法的状态转换，验证系统是否会正确地拒绝这些操作；
4) 验证边界状态：重点测试初始状态和终结状态的处理逻辑，以及在每个状态下可以执行的操作权限是否正确。

c) 实施要点：

1) 核心目标：验证业务对象的状态转换逻辑与设计完全一致，不存在非预期的状态跳转或在错误状态下执行操作的漏洞；
2) 关键方法：将状态机模型作为测试地图，系统性地对所有可能的状态路径进行探索和验证；
3) 主要产出：一份状态机测试报告，确认所有状态转换的正确性，并指出任何与设计不符的实现；
4) 最终价值：保障核心业务流程的完整性和逻辑严密性，防止攻击者通过操纵对象状态来绕过业务规则，造成经济损失或数据混乱。

7.4.4 测试阶段软件系统级信息安全支撑技术

7.4.4.1 动态应用安全测试技术

7.4.4.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在从外部攻击者的黑盒视角，对一个正在运行的应用程序进行攻击性测试，以发现其可被利用的安全漏洞；

b) 核心原理：模拟真实攻击者行为：自动化扫描器先通过爬虫发现所有可访问的页面和接口，然后向每个发现的输入点系统性地注入大量预先构造的攻击载荷，最后通过分析应用的HTTP响应来推断是否存在漏洞；

c) 主要优势：

1) 真实性与低误报率：测试的是一个完整部署、真实运行的系统，因此发现的漏洞通常是实际可利用的，能准确反映系统在公网上的风险状况；

2) 验证整体安全：能够验证多个组件集成后的整体安全性，发现因环境配置不当或组件交互而产生的问题。

7.4.4.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：在测试环境中完整部署并运行的软件系统。

b) 实施步骤：

1) 爬取与探查：自动化扫描器首先爬取整个应用，发现所有可访问的页面、接口和参数，构建应用结构图；

2) 注入攻击载荷：扫描器向发现的每个输入点系统性地注入大量预定义的、针对各类漏洞的攻击载荷；

3) 分析应用响应：扫描器分析应用对恶意载荷的响应，通过观察返回内容、HTTP状态码、响应时间等变化，来判断是否存在漏洞；

4) 生成漏洞报告：将所有基于响应推断出的潜在漏洞进行汇总，生成报告。

c) 实施要点：

1) 核心目标：从外部攻击者的黑盒视角，模拟对正在运行的应用进行攻击，以发现可利用的安全漏洞；

2) 关键方法：通过“爬取—攻击—分析响应”的循环，自动化地对应用进行攻击性测试；

3) 主要产出：一份动态扫描报告，列出从外部视角发现的潜在漏洞及其利用入口；

4) 最终价值：能够发现应用在真实运行环境中暴露出的、可被外部利用的漏洞，验证多组件集成后的整体安全性。

7.4.4.2 基础设施漏洞扫描技术

7.4.4.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在自动化地发现并管理承载软件运行的基础设施层面（如服务器、操作系统、网络设备、数据库等）存在的已知安全漏洞；

b) 核心原理：利用漏洞扫描器对目标网络或主机进行主动探测，通过“指纹识别”技术精确判断其上运行软件的名称和版本号，然后与持续更新的全球漏洞数据库进行比对，从而找出所有未打补丁的组件；

c) 主要优势：

1) 广度与效率：能够快速、全面地盘点基础设施资产的安全状况，自动发现因未及时更新而存在的安全隐患；
2) 保障运行基础：确保软件的运行基础是相对安全的，有效防止攻击者利用底层平台的已知漏洞直接绕过应用层安全控制。

7.4.4.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 承载应用的服务器、网络设备、操作系统、数据库、中间件等；
   2) 生产环境或与生产环境高度一致的预发布环境。
b) 实施步骤：
   1) 资产发现：扫描指定网段，发现所有存活的主机和开放的网络服务端口；
   2) 服务识别与指纹探测：对开放的端口进行探测，识别其上运行服务的类型和版本；
   3) 漏洞匹配：将识别出的服务版本与持续更新的漏洞数据库进行比对，找出存在的已知漏洞；
   4) 生成与评估报告：生成包含受影响主机、漏洞详情、风险等级和修复建议的报告，并由安全团队进行风险评估和分发。
c) 实施要点：
   1) 核心目标：识别并管理软件运行环境基础设施层面存在的已知安全漏洞；
   2) 关键方法：通过主动扫描和版本比对，发现未打补丁的软件或系统组件；
   3) 主要产出：一份基础设施漏洞扫描报告；
   4) 最终价值：确保软件的运行基础是安全的，防止攻击者利用底层平台的已知漏洞绕过应用层安全控制，直接入侵服务器。

7.5 软件信息安全性发布与维护阶段支撑技术

7.5.1 发布与维护阶段代码级信息安全支撑技术

7.5.1.1 软件制品完整性与来源验证技术

7.5.1.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在为最终发布的软件制品（如安装包、容器镜像）提供一种可靠的完整性和来源真实性验证机制；
b) 核心原理：利用密码学的哈希函数和数字签名。发布方对合格的软件制品计算一个唯一的哈希值，并用其私钥对哈希值进行签名；使用方在部署前通过验证哈希值和签名来确认制品未被篡改且来源可信；
c) 主要优势：
   1) 抵御篡改：可确保用户获取的软件制品与发布方提供的原始状态一致，未被中间环节植入后门或恶意代码；
   2) 保障末端安全：是保障软件供应链末端安全、验证软件来源与完整性的重要控制措施，能抵御供应链投毒攻击。

7.5.1.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 经过测试、准备发布的软件包、容器镜像、固件等软件制品；

2) 从分发渠道下载或在部署前接收的软件制品。

b) 实施步骤：

1) 生成哈希或签名：在软件构建流程的最后一步，对最终的软件制品计算一个唯一的哈希值，并可选地使用私钥对其进行数字签名；

2) 安全发布凭据：将计算出的哈希值和签名文件，通过一个安全的、独立的渠道发布；

3) 部署前校验：在部署软件制品之前，用户或自动化部署系统需要重新计算下载的制品的哈希值，并与官方发布的哈希值进行比对；如果提供了签名，则使用公钥验证其有效性；

4) 阻断不一致部署：如果哈希值不匹配或签名验证失败，则立即中止部署流程，并发出安全警报。

c) 实施要点：

1) 核心目标：确保从构建到部署的整个过程中，软件制品未被恶意篡改或损坏；

2) 关键方法：利用密码学哈希的唯一性和数字签名的不可抵赖性，提供一种可靠的校验机制；

3) 主要产出：与每个软件制品相关联的哈希值和（或）数字签名；

4) 最终价值：保障软件供应链的末端安全，防止用户部署被植入后门或病毒的软件包，是抵御供应链投毒攻击的关键控制点。

7.5.1.2 第三方组件漏洞持续监控技术

7.5.1.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在解决软件发布后，其依赖的第三方组件新爆出漏洞所带来的安全风险；

b) 核心原理：建立一个动态的风险监控和响应循环：维护一份线上所有产品的动态SBOM，并将其与实时的漏洞情报源进行自动化关联分析，一旦发现正在使用的组件存在新漏洞，立即触发告警和应急响应；

c) 主要优势：

1) 应对动态风险：软件在发布后，其依赖的组件仍可能发现新的漏洞，此技术可有效应对这类因时间推移而产生的动态风险；

2) 实现持续安全：通过“动态资产清单”与“动态威胁情报”的结合，实现了从“发布即终止”到“全生命周期持续安全”的转变。

7.5.1.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 已部署在线上运行的软件系统；

2) 组织内所有产品使用的第三方组件清单。

b) 实施步骤：

1) 维护动态物料清单：持续更新已部署应用的SBOM，反映线上环境的真实情况；

2) 订阅漏洞情报：接入一个或多个漏洞情报源，实时获取全球范围内新披露的开源或商业软件漏洞信息；

3) 自动化关联分析：建立自动化流程，将新的漏洞情报与维护的物料清单进行实时比对，一旦发现已使用的组件存在新漏洞，立即触发告警；

4) 风险评估与应急响应：安全团队对告警进行快速评估，判断漏洞在当前系统中的实际影响，并启动应急响应流程进行修复或缓解。

c) 实施要点：
   1) 核心目标：在软件发布后，持续跟踪其依赖的第三方组件的安全状况，对新出现的漏洞做出快速反应；
   2) 关键方法：将动态的软件资产清单与动态的威胁情报相结合，实现风险的持续发现；
   3) 主要产出：实时的第三方组件漏洞告警，以及相应的风险评估报告；
   4) 最终价值：解决软件发布后因“时间差”导致的安全风险（即发布时安全的组件后续爆出漏洞），确保软件在整个生命周期内的持续安全。

7.5.2 发布与维护阶段接口级信息安全支撑技术

7.5.2.1 接口流量异常监测技术

7.5.2.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在为生产环境中的API提供一层实时的、动态的运行时保护，以发现并阻止各类攻击和滥用行为；
b) 核心原理：通过基于签名的引擎匹配已知攻击特征，或通过基于行为的引擎学习应用程序编程接口的正常调用模式以建立基线，用于监测并告警任何偏离基线的异常行为；
c) 主要优势：
   1) 运行时防护：能够在攻击真实发生时进行检测和阻断，弥补了静态防御措施的不足；
   2) 检测未知攻击：基于异常行为的检测不依赖已知攻击签名，能够发现针对业务逻辑的滥用和新型自动化攻击。

7.5.2.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 生产环境中的API网关或暴露在公网的API端点；
   2) 所有进出系统的API调用流量。
b) 实施步骤：
   1) 建立流量基线：在正常运行期间，采集和分析API流量，学习正常调用的模式，包括请求频率、参数分布、用户行为序列等，建立行为基线；
   2) 实时流量分析：实时捕获所有API请求和响应，并与已建立的基线进行比较，或应用预设的攻击模式进行检测；
   3) 识别异常行为：检测偏离基线的行为、已知的攻击特征以及非法的业务操作序列；
   4) 告警与阻断：一旦检测到可疑或恶意流量，系统应立即生成告警，并可根据策略自动执行阻断、限流或要求二次验证等响应动作。
c) 实施要点：
   1) 核心目标：在运行时实时监测API流量，发现并阻止针对API的攻击和滥用行为；
   2) 关键方法：结合基于签名的检测和基于行为异常的检测；
   3) 主要产出：实时的API攻击告警、可疑活动日志以及自动化的防护动作记录；
   4) 最终价值：为API提供一层动态的、自适应的运行时保护，有效防御各类自动化攻击、业务逻辑滥用和数据窃取企图。

7.5.2.2 影子接口发现与治理技术

7.5.2.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过流量分析自动发现生产环境中未被记录、未受管制的“影子接口”及废弃接口，消除资产盲区；
b) 核心原理：旁路镜像生产流量，利用聚类分析或规范比对技术，识别出实际流量中存在但未在网关注册或文档中缺失的接口路径与参数；
c) 主要优势：
   1) 资产可见性：实时绘制真实的接口攻击面地图，发现绕过网关的隐蔽通道；
   2) 数据合规：发现未受监管的敏感数据传输接口，防止数据通过未知路径泄露。

7.5.2.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 所有进出数据中心的通信流量；
   2) 接口网关日志及负载均衡器日志。
b) 实施步骤：
   1) 流量采集：在网关或核心交换机处配置流量镜像，将数据包复制到分析引擎；
   2) 基线比对：导入现有的接口文档，将实时流量中的请求路径与文档进行自动比对；
   3) 异常识别：标记“有流量无文档”的影子接口及“有文档无流量”的僵尸接口；
   4) 处置闭环：对发现的影子接口触发告警，要求开发团队限期注册、下线或补全安全策略。
c) 实施要点：
   1) 核心目标：消除接口资产盲区，确保所有对外接口均处于安全监控与管控之下；
   2) 关键方法：流量分析与文档规范比对相结合，持续审计接口清单；
   3) 主要产出：影子接口与僵尸接口清单、接口资产覆盖率报告及差异分析告警；
   4) 最终价值：封堵因文档滞后或管理疏漏导致的安全漏洞，实现接口资产的动态清零与全生命周期治理。

7.5.3 发布与维护阶段模型级信息安全支撑技术

7.5.3.1 基于威胁情报的模型校准与验证技术

7.5.3.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在建立一个反馈闭环，利用来自现实世界的真实攻击数据和威胁情报，来持续验证、修正和完善系统在设计阶段建立的抽象安全模型（如威胁模型、攻击树）；
b) 核心原理：将外部结构化的威胁情报与内部具体的系统设计模型进行自动化关联映射，分析新出现的攻击技术是否会影响原有模型的假设、是否会开辟新的攻击路径，从而动态评估真实风险；
c) 主要优势：
   1) 动态对抗演进：确保安全设计模型能够随安全威胁的演变而动态更新，维持其长期有效性；
   2) 主动精准防御：使防御策略的调整、监控规则的优化等工作能够基于最新的、情景化的风险评估来进行，使安全工作更主动和精准。

7.5.3.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 外部的、结构化的威胁情报；

2) 系统在设计阶段建立的安全模型；
3) 系统的日志、告警以及资产信息。

b) 实施步骤：
1) 情报与模型映射：将外部威胁情报中描述的攻击技术和攻击路径，与系统自身的威胁模型和攻击树进行比对映射；
2) 模型有效性验证：评估新出现的、真实的攻击手法是否在原有模型的覆盖范围之内，分析其是否会使模型中的某些攻击路径变得更容易实现，或开辟了全新的攻击路径；
3) 模型动态更新：根据现实世界的攻击数据，迭代和修正原有的安全模型，使其能够反映当前最新的威胁状况，保持模型的有效性；
4) 驱动风险再评估：基于更新后的模型和情报预警，重新评估系统面临的风险，并指导调整监控规则、优化防御策略或启动预防性加固。

c) 实施要点：
1) 核心目标：建立一个反馈闭环，利用真实的攻击数据来持续验证、修正和完善系统的安全设计模型；
2) 关键方法：将抽象的、外部的威胁情报，与具体的、内部的系统模型相结合，进行情景化分析，评估现实威胁对特定设计的影响；
3) 主要产出：一份更新后的威胁模型或攻击树，以及基于新情报的风险评估与处置建议报告；
4) 最终价值：使静态的安全设计模型动态更新，从而让系统安全防御的顶层设计始终保持前瞻性和有效性。

7.5.3.2 基于数字孪生的攻防演练复盘技术

7.5.3.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在利用数字孪生技术构建与生产环境高度一致的虚拟靶场，对真实发生的攻击事件或红蓝对抗进行高保真复盘与推演；
b) 核心原理：采集生产环境的配置、拓扑及流量数据，在隔离环境中动态生成“数字孪生体”，在此基础上重放攻击流量，分析攻击路径、验证防御策略有效性并推演潜在后果；
c) 主要优势：
1) 零干扰评估：在不影响生产业务的前提下，对高危攻击场景进行反复实验与破坏性测试；
2) 精准溯源：提供全视角的攻击链路回放能力，精确还原攻击者的每一步操作与系统的响应状态。

7.5.3.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
1) 关键基础设施或高价值业务系统；
2) 历史安全事件日志及攻防演练数据。

b) 实施步骤：
1) 孪生体构建：基于基础设施即代码配置与实时监控数据，自动化拉起与生产环境一致的仿真环境；
2) 攻击重放：将捕获的攻击流量或红队操作指令注入孪生环境；
3) 防御验证：在孪生环境中调整防火墙规则或防护策略，验证其是否能有效阻断该类攻击；
4) 策略同步： 将验证有效的防御策略反向同步至生产环境，实现安全闭环。

c) 实施要点：
   1) 核心目标：通过高保真的仿真复盘，持续优化安全防御体系，提升对未知威胁的响应能力；
   2) 关键方法：构建生产环境的动态数字孪生体，并实施攻击流量重放与防御推演；
   3) 主要产出：攻防演练复盘报告、经验证的防御策略配置建议及系统脆弱性分析报告；
   4) 最终价值：将被动的事件响应转化为主动的防御优化，构建“持续验证、持续改进”的安全运营闭环。

7.5.4 发布与维护阶段软件系统级安全支撑技术

7.5.4.1 运行时应用自我保护技术

7.5.4.1.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在赋予应用程序在运行时进行自我诊断和自我保护的能力，作为应用内部的一项纵深防御措施；
b) 核心原理：将一个轻量级的安全探针直接嵌入应用程序的运行环境内部，实时监控其内部行为（如函数调用、数据库查询），当检测到明确的攻击行为时，能即刻采取拦截、告警等响应动作；
c) 主要优势：
   1) 高精准度：由于身处应用内部并能理解上下文，能以极低的误报率精确识别并拦截已绕过边界防御的攻击；
   2) 主动精准防御：内置的防护机制使应用程序具备主动防御能力，能够在攻击发生时即时响应，是纵深防御的重要一环。

7.5.4.1.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 正在生产环境中运行的应用程序；
   2) 存在已知漏洞但暂时无法修复（如维护周期长的遗留系统或无源码的第三方组件）的应用程序。
b) 实施步骤：
   1) 嵌入防护能力：将RASP的安全防护能力以探针或库的形式，直接嵌入到应用程序的运行环境内部；
   2) 监控内部行为：RASP实时监控应用的内部行为；
   3) 上下文感知检测：基于对应用上下文的理解，当检测到明确的、正在发生的攻击行为时，立即采取行动；
   4) 实时响应：响应动作可以是拦截恶意操作、终止用户会话、记录详细攻击信息或向监控系统发送告警。
c) 实施要点：
   1) 核心目标：让应用程序具备在运行时进行自我诊断和自我保护的能力，实现精准、实时的内部防御；
   2) 关键方法：将防护能力深度嵌入应用内部，通过实时监控和上下文感知，在攻击行为发生瞬间予以精准拦截或响应；
   3) 主要产出：对绕过边界防御的攻击进行实时拦截的记录和告警；

4) 最终价值：可作为一项纵深防御措施，使应用程序具备运行时的自我诊断和自我保护能力。

7.5.4.2 运行时完整性监控技术

7.5.4.2.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在持续地监控生产环境中关键系统文件和进程的完整性，以快速检测系统是否已被非法篡改或攻陷；

b) 核心原理：基于“基线比对”。首先为监控目标建立一个“已知良好”的安全基线（加密哈希值），然后定期或实时地重新计算当前状态并与基线进行比对；

c) 主要优势：

1) 高可靠性：对文件或内存的任何内容修改都会导致哈希值改变，能准确发现入侵行为；

2) 及时告警：能够为安全事件响应提供及时、准确的高优先级告警，是检测系统是否被攻陷的关键技术手段。

7.5.4.2.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：

1) 生产环境中软件系统的可执行文件、库文件、核心配置文件；

2) 关键进程的内存区域。

b) 实施步骤：

1) 建立安全基线：在系统初始部署或更新后，为所有受监控的关键文件和内存区域计算并安全地存储其哈希值或数字签名，建立一个“已知良好”的基线；

2) 持续监控：监控系统定期或在特定事件（如文件修改）触发时，重新计算目标的哈希值或校验签名；

3) 比对基线：将计算结果与已知的安全基线进行比对；

4) 告警与响应：一旦发现任何未经授权的篡改，系统立即生成高优先级告警，并可触发自动化的响应动作。

c) 实施要点：

1) 核心目标：在生产环境中持续监控系统的完整性，防范攻击；

2) 关键方法：基于“基线比对”的原理，发现非授权变更；

3) 主要产出：针对文件或内存被篡改的高优先级告警；

4) 最终价值：是检测系统是否已被攻陷的关键技术手段。

7.5.4.3 凭证与密钥自动化轮换方法

7.5.4.3.1 核心阐述

核心阐述涵盖以下方面：

a) 定义与目标：旨在通过自动化流程，定期更换系统中的各类敏感凭证和加密密钥，以降低因凭证泄露而带来的长期风险；

b) 核心原理：建立一套自动化的调度和执行机制，该机制能自动生成、分发、部署新凭证，并通过健康度检查验证服务正常，最后安全地废弃旧凭证；

c) 主要优势：

1) 解决运维难题：解决了手动轮换复杂、高风险、低频的问题，确保了轮换过程的平滑、无中断和高频次；

2) 缩短攻击窗口：极大地缩短了凭证因泄露而可能被滥用的时间窗口，是维持系统长期安全性的关键运维实践。

7.5.4.3.2 应用实施要点

应用与实施涵盖以下方面：

a) 适用对象：
   1) 系统中使用的各类长期凭证；
   2) 用于数据加密和签名的各类密钥。
b) 实施步骤：
   1) 建立轮换策略与流程：为不同类型的凭证和密钥，制定明确的轮换周期和自动化的轮换流程；
   2) 执行轮换操作：通过自动化脚本或密钥管理系统，在预定时间生成新凭证或密钥，并将其安全地分发和部署到所有使用方；
   3) 功能验证：在新凭证或密钥部署后，立即执行一系列自动化测试，验证所有依赖该凭证或密钥的服务和功能是否依然正常工作；
   4) 废弃旧凭证：在确认系统在新凭证或密钥下运行稳定后，从系统中彻底移除并废弃旧的凭证或密钥，确保其不再可用。
c) 实施要点：
   1) 核心目标：定期更换系统中的敏感凭证和密钥，以缩短其可能被泄露和滥用的时间窗口；
   2) 关键方法：将轮换过程自动化、流程化，并包含必要的验证步骤，以确保轮换过程的平滑和无中断；
   3) 主要产出：凭证和密钥轮换的成功执行记录，以及功能验证通过的报告；
   4) 最终价值：显著降低因凭证或密钥泄露带来的长期风险，是维持系统长期安全性的关键运维实践。

8 软件信息安全性全生命周期活动和适应阶段

8.1 需求阶段活动

本阶段的主要目标是识别系统面临的信息安全威胁，界定合规性边界，并将模糊的信息安全愿景转化为可度量、可验证的需求规约。需求阶段应开展以下主要活动：

a) 定义信息安全目标与风险基线：
   1) 目的：在项目启动之初，将抽象的信息安全概念与具体业务价值对齐，明确信息资产的保护对象和防护优先级；
   2) 技术要求：应运用安全需求基线生成与裁剪方法来确立基础安全标准，并采用密码算法选型与敏捷性规约技术明确系统允许使用的加密算法集合及更迭策略。通过STRIDE威胁分析技术等方法，系统性地发现潜在信息安全威胁点，并结合风险评估，明确安全设计的重点；
   3) 解决问题：确保信息安全资源的投入能够与实际风险等级精确匹配，避免因信息安全措施缺失或过度设计导致资源浪费，为后续所有信息安全工作提供方向性指引。
b) 分析攻击路径与细化信息安全需求：
   1) 目的：从攻击者视角审视系统，并将泛化的信息安全目标转化为具体、可验证、可执行的技术与功能需求；
   2) 技术要求：应利用攻击树分析技术将高层攻击目标分解为具体步骤；基于第三方组件选型与准入控制方法，明确对软件供应链的管控要求；同时，通过安全编码规范定义与基

线化方法、接口安全契约规约技术以及接口防自动化与速率限制规约技术，将安全规则前置，明确防范自动化滥用的具体指标；

3) 解决问题：使信息安全需求更具针对性和可操作性，确保软件设计满足内外部的合规性要求，并从源头前置管理日益复杂的软件供应链信息安全风险。

8.2 设计阶段活动

本阶段的主要目标是将需求阶段确立的信息安全需求，转化为稳健的系统架构和可靠的技术实现方案，构建出易于防御、难以攻破的系统蓝图。设计阶段应开展以下主要活动：

a) 构建内生信息安全的系统架构：

1) 目的：从顶层设计上降低系统性风险，致力于构建一个易于防御、难于攻破，且具备良好故障隔离能力的系统蓝图；

2) 技术要求：应开展攻击面分析与最小化设计方法以系统性地识别并收缩暴露点，采用基于信任域的架构隔离与访问控制技术以实现故障遏制和损害控制，并设计密钥管理基础设施集成与自动化技术为敏感数据提供端到端保护；

3) 解决问题：从源头上减少被攻击的可能性，并设计好兜底策略，防止单点信息安全失陷演变为对整个系统的连锁攻击，保障核心数据的全流程信息安全性。

b) 设计安全可靠的组件与接口：

1) 目的：确保系统内部的核心逻辑模块与对外交互的接口具备明确、形式化的信息安全属性，将信息安全规则固化于设计之中；

2) 技术要求：应在接口设计中落实访问凭证与会话管理设计方法，遵循零信任原则。通过不可信数据处理与输出编码设计方法防御注入类攻击，并采用异常处理与日志脱敏设计方法确保故障状态下的隐私安全。对保障高信息安全等级的系统，宜使用基于形式化方法的安全属性证明技术对关键算法和协议进行严谨证明，并应用端到端数据流加密与密钥交换技术保障传输安全；

3) 解决问题：将信息安全需求精确地融入系统架构和接口定义中，消除“内部可信”等不安全的设计假设，为高风险模块提供最高级别的信息安全保证。

8.3 实现阶段活动

本阶段的主要目标是将信息安全设计蓝图转化为高质量、低风险的软件代码和配置，确保每一项业务逻辑和数据处理流程都严格遵循既定的信息安全规范。实现阶段应开展以下主要活动：

a) 遵循编码规范与管理组件依赖：

1) 目的：在代码编写和构建过程中，实时防范低级信息安全缺陷，并自动化地控制由第三方组件引入的信息安全风险；

2) 技术要求：应通过安全代码人工审查方法确保代码符合规范。在构建过程中集成SCA，用以扫描并识别依赖项中的已知漏洞。开发人员应利用接口实现的安全单元测试方法验证代码安全性，并采用接口故障注入与熔断实现技术在开发阶段验证系统的容错与降级能力；

3) 解决问题：有效提升代码的原生信息安全性，将信息安全检查左移至编码阶段，自动化地管理和控制软件供应链风险，防止携带漏洞的组件进入制品库。

b) 确保设计与实现的一致性：

1) 目的：确保所有设计阶段确立的安全机制和业务逻辑被忠实、完整地翻译为代码，防止因实现偏差或遗漏导致的安全漏洞；

2) 技术要求：应开展设计模型到代码实现的一致性校验技术，将代码实现与设计模型进行比对。同时，应用安全模型驱动的策略代码生成技术，实现安全策略的自动化生成与部署；
3) 解决问题：关闭设计与实现之间的“鸿沟”，保证安全设计的初衷被忠实地转化为可执行的代码，防止在编码环节引入新的逻辑漏洞。

c) 确保运行环境的配置安全：
1) 目的：将基础设施的信息安全配置纳入版本控制和自动化流程，在部署到生产环境之前，消除环境配置中的已知信息安全隐患；
2) 技术要求：应运用基础设施即代码静态合规性分析技术，对服务器配置进行检查；同时采用容器镜像最小化与加固技术，精简运行环境，移除不必要的组件与特权；
3) 解决问题：实现信息安全配置的左移和自动化，提升部署效率和环境的一致性，从源头上杜绝因环境配置错误导致的信息安全事件。

8.4 测试阶段活动

本阶段的主要目标是通过系统性的验证与评估活动，从代码、接口、业务逻辑到系统整体等多个维度，主动发现并修复潜在的信息安全缺陷，确保软件在发布前达到预定的信息安全质量标准。测试阶段应开展以下主要活动：

a) 开展代码层面的静态与结构化验证：
1) 目的：从源代码层面系统性地发现深层逻辑漏洞、编码缺陷以及不安全的第三方组件依赖；
2) 技术要求：应执行全面的SAST，并通过面向业务逻辑的白盒安全测试方法验证关键信息安全逻辑；
3) 解决问题：系统性地验证代码实现是否满足既定的信息安全规范，发现自动化工具难以捕捉的业务逻辑层面的编码缺陷。

b) 进行接口与运行时的动态与交互式验证：
1) 目的：从外部攻击者和内部运行时的双重视角，评估应用程序在真实运行状态下的健壮性、防御能力和实际信息安全表现；
2) 技术要求：应使用模糊测试技术探测接口的健壮性，利用IAST精准定位运行时漏洞，并开展基于权限模型的越权攻击模拟技术来模拟越权攻击；
3) 解决问题：高效发现运行时环境中暴露的各类信息安全缺陷，验证接口对合法、非法及恶意输入的响应是否安全、得当。

c) 验证业务逻辑模型与系统互动：
1) 目的：确保复杂的业务流程、状态转换机制和后端数据处理严格按照信息安全设计实现，发现自动化脚本难以覆盖的深层业务逻辑漏洞；
2) 技术要求：应采用业务逻辑并发测试技术发现竞争条件漏洞，并运用状态机逻辑遍历测试技术探索状态机中的逻辑错误；
3) 解决问题：深入验证系统在复杂应用场景和数据交互下的信息安全性，发现与业务紧密相关的信息安全风险。

d) 实施系统级的综合性信息安全评估：
1) 目的：在软件发布前，模拟真实世界的攻击，对整个系统的综合防御体系、韧性和恢复能力进行最终的、全面的信息安全验收；
2) 技术要求：应运行DAST与基础设施漏洞扫描技术以发现已知风险，并通过渗透测试等手段尝试串联漏洞形成攻击链；

3) 解决问题：全面评估系统在真实运行环境中的综合信息安全水位，验证其是否满足业务连续性和数据完整性等核心信息安全目标。

8.5 发布与维护阶段活动

本阶段的主要目标是在生产环境中对软件进行持续的威胁监控与主动防御，保障信息安全措施在整个运营周期内的长期有效性，并对新出现的信息安全风险进行快速响应与修复。发布与维护阶段应开展以下主要活动：

a) 保障软件制品的完整性与供应链的持续信息安全：
   1) 目的：确保交付给用户的软件是来源可信且未经篡改的，并在其整个生命周期内，对其中包含的第三方组件信息安全状况进行持续监控；
   2) 技术要求：应在发布制品时使用软件制品完整性与来源验证技术，并于部署后通过第三方组件漏洞持续监控技术对新发现的漏洞进行预警；
   3) 解决问题：有效防止软件在分发与部署过程中被恶意篡改，实现对已部署软件供应链风险的长期、动态管理，为快速应急响应提供支持。
b) 实施运行时的持续监控与主动防御：
   1) 目的：在生产环境中实时检测、响应和阻止各类信息安全攻击，并保障系统的持续稳定运行，将信息安全能力从被动响应转向主动防御；
   2) 要求：应部署接口流量异常监测技术来识别攻击行为，利用影子接口发现与治理技术消除未纳管的API资产。同时，运用基于威胁情报的模型校准与验证技术主动发现潜在入侵，部署RASP实现应用内的主动防御，并利用运行时完整性监控技术防止系统被恶意篡改；
   3) 解决问题：极大地提升系统在生产环境中的信息安全可见性和自我保护能力，在攻击发生的瞬间进行精准、实时地拦截，降低信息安全事件造成的影响。
c) 执行长期的信息安全运维与策略迭代：
   1) 目的：通过持续的运维活动，确保关键信息安全措施的长期有效性，并根据环境变化不断迭代和优化；
   2) 技术要求：应严格执行凭证与密钥自动化轮换方法，包括密钥的定期轮换、安全归档与销毁等操作。同时，宜采用基于数字孪生的攻防演练复盘技术，在虚拟靶场中对安全事件进行高保真推演；
   3) 解决问题：有效维持密码学保护措施的长期有效性，防止因密钥泄露或算法过时导致的历史数据泄漏风险，是保障系统信息安全水平不随时间推移而降低的关键工作。

软件信息安全性活动见表 1。

表 1 软件信息安全性全生命周期主要活动列表

活动阶段	活动名称	支撑技术与方法
需求阶段	定义信息安全目标与风险基线	安全需求基线生成与裁剪方法、STRIDE威胁分析技术、密码算法选型与敏捷性规约技术
	分析攻击路径与细化信息安全需求	攻击树分析技术、第三方组件选型与准入控制方法、接口安全契约规约技术、安全编码规范定义与基线化方法、接口防自动化与速率限制规约技术

表 1 软件信息安全性全生命周期主要活动列表（续）

活动阶段	活动名称	支撑技术与方法
设计阶段	构建内生信息安全的系统架构	攻击面分析与最小化设计方法、基于信任域的架构隔离与访问控制技术、密钥管理基础设施集成与自动化技术
	设计安全可靠的组件与接口	访问凭证与会话管理设计方法、不可信数据处理与输出编码设计方法、基于形式化方法的安全属性证明技术、端到端数据流加密与密钥交换技术、异常处理与日志脱敏设计方法
实现阶段	遵循编码规范与管理组件依赖	安全代码人工审查方法、SCA、接口实现的安全单元测试方法、接口故障注入与熔断实现技术
	确保设计与实现的一致性	设计模型到代码实现的一致性校验技术、安全模型驱动的策略代码生成技术
	确保运行环境的配置安全	基础设施即代码静态合规性分析技术、容器镜像最小化与加固技术
测试阶段	开展代码层面的静态与结构化验证	SAST、面向业务逻辑的白盒安全测试方法
	进行接口与运行时的动态与交互式验证	模糊测试技术、IAST、基于权限模型的越权攻击模拟技术
	验证业务逻辑模型与系统互动	业务逻辑并发测试技术、状态机逻辑遍历测试技术
	实施系统级的综合性信息安全评估	DAST、基础设施漏洞扫描技术
发布与维护阶段	保障软件制品的完整性与供应链的持续信息安全	软件制品完整性与来源验证技术、第三方组件漏洞持续监控技术
	实施运行时的持续监控与主动防御	接口流量异常监测技术、影子接口发现与治理技术、基于威胁情报的模型校准与验证技术、RASP、运行时完整性监控技术
	执行长期的信息安全运维与策略迭代	凭证与密钥自动化轮换方法、基于数字孪生的攻防演练复盘技术

参 考 文 献

[1] GB/T 8566—2022 系统与软件工程 软件生存周期过程

[2] GB/T 22239—2019 信息安全技术网络安全等级保护基本要求

[3] GB/T 25000.10—2016 系统与软件工程 系统与软件质量要求和评价系统与软件质量模型

[4] GB/T 25000.51—2016 系统与软件工程 系统与软件质量要求和评价（SQuaRE） 第51部分：就绪可用软件产品（RUSP）的质量要求和测试细则

[5] GB/T 32857—2016 保护层分析（LOPA）应用指南

[6] GJB/Z 157—2011 军用软件安全保证指南

[7] GJB 5236—2004 军用软件质量度量

[8] T/CICC 35008—2025 复杂软件系统可靠性技术要求

[9] IEEE Std 828-2012. "IEEE Standard for Configuration Management in Systems and Software Engineering." IEEE Computer Society, 2012.

[10] IEEE 730-2014 IEEE Standard for Software Quality Assurance Processes.

[11] IEEE Std 1633-2016. "IEEE Recommended Practice on Software Reliability." IEEE Computer Society, 2016.

[12] IEEE 1856-2017 IEEE Standard Framework for Prognostics and Health Management of Electronic Systems.

[13] IEEE Std 2675-2021 DevOps: Building Reliable and Secure Systems Including Application Security, Configuration Management, Provisioning, and Deployment

[14] ISO/IEC/IEEE 12207:2017. "Systems and software engineering -- Software life cycle processes." International Organization for Standardization, 2017.

[15] ISO/IEC 25389:2025 Information technology — The safe framework

[16] ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection — Information security management systems — Requirements.

[17] ISO/IEC 29100:2024 Information technology — Security techniques — Privacy framework

ICS 35.080
CCS L 77

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35002—2026

复杂软件系统保障性技术要求

Technical requirements for supportability of complex software systems

2026-02-28 发布　　2026-02-28 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。

本文件起草参与单位：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）、中国科学院声学研究所、中国航空综合技术研究所、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、赵诣深、郝程鹏、吴梦丹、林聪、廖力鸣、王英凡、曾福萍、代国良。

复杂软件系统保障性技术要求

1 范围

本文件规定了复杂软件系统保障性的定性要求与定量指标、支撑技术与方法，并规定了复杂软件系统各阶段活动中的相关实践。

本文件适用于复杂软件系统全生命周期的各项保障性活动。

2 规范性引用文件

下列文件中的有关条款通过引用而成为本文件的条款。凡注日期或版次的引用文件，其后的任何修改（不包括勘误的内容）或修订版本都不适用于本文件，凡不注日期或版次的引用文件，其最新版本适用于本文件。

GB/T 11457—2006 信息技术 软件工程术语

GB/T 25000.10—2016 系统与软件工程 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第 10 部分：系统与软件质量模型

GB/T 25000.21—2019 系统与软件工程 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第 21 部分：质量测度元素

GB/T 30998—2014 信息技术 软件安全保障规范

GB/T 39834.1—2013 系统与软件维护性 第 1 部分：指标体系

T/CICC 35008—2025 复杂软件系统可靠性技术要求

3 术语和定义

GB/T 11457—2006、GB/T 25000.21—2019和T/CICC 35008—2025确立的以及下列术语与定义适用于本文件。

3. 1

软件保障 software support

确保软件生存周期过程及产品符合需求、标准和规程要求的一组有计划的活动。

[来源：GB/T 30998—2014，3.1.26]

3.2

软件保障性 software supportability

软件在设计、开发和维护过程中通过系统化措施保障其长期可靠运行及抵御风险的综合能力。

3.3

软件质量保障 software quality assurance

通过一系列有计划、有系统的活动，确保软件开发过程符合既定技术需求，并向管理层提供足够信心，最终提高软件质量、减少缺陷并提升用户满意度。

3.4

软件安全保障 software security assurance

确保所识别的软件的需求、设计、实现、验证及运行规程能使潜在的危险最小化或将其排除。

[来源：GB/T 30998—2014，3.1.24，有修改]

3.5

软件维护 software maintenance

在交付以后,修改软件系统以排除故障、改进性能或其他属性或适应变更了的环境的过程。

[来源：GB/T 29834.1—2013，3.9，有修改]

3.6

软件维护性 software maintainability

软件系统或其部件能修改以排除故障、改进性能或其他属性或适应变更了的环境的容易程度。

[来源：GB/T 29834.12—013，3.4，有修改]

3.7

软件保密性 software confidentiality

软件系统确保数据只有在被授权时才能被访问的程度。

[来源：GB/T 25000.10—2016，4.3.2.6.1，有修改]

3.8

软件合规性 software compliance

软件系统遵循与特定领域相关的标准、法规或约定的程度。

3.9

软件完整性 software integrity

软件系统防止未授权访问、篡改计算机程序或数据的程度。

[来源：GB/T 25000.10—2016，4.3.2.6.2，有修改]

3.10

软件可信性 software trustworthiness

软件系统执行正确且可预期的功能的能力。

3.11

软件可用性 software usability/availability

软件系统在需要使用时能够进行操作和访问的程度。

[来源：GB/T 25000.10—2016，4.3.2.5.2，有修改]

4 缩略语

下列缩略语适用于本文件。

API	应用程序接口（Application Programming Interface）
CA	证书颁发机构（ Certificate Authority）
CCB	变更控制委员会（Change Control Board）
CISSP	信息系统安全专业认证（Certified Information System Security Professional）
CPU	中央处理器（Central Processing Unit）
DAST	动态代码安全分析（Dynamic Application Security Testing）
FMEA	故障模式与影响分析（Failure Mode and Effects Analysis）
FTA	故障树分析（Fault Tree Analysis）
Git LFS	Git大文件存储（Git Large File Storage）
HSM	硬件安全模块（Hardware Security Module）
HTTPS	超文本传输安全协议（Hyper Text Transfer Protocol Secure）
IaC	基础设施即代码（Infrastructure as Code）
IDS	入侵检测系统（Intrusion Detection System）
IP	互联网协议（Internet Protocol）

KMS	密钥管理系统（Key Management System）
MFA	多因素认证（Multi-Factor Authentication）
OCL	对象约束语言（Object Constraint Language）
OMG	对象管理组织（Object Management Group）
QA	质量保障（Quality Assurance）
RBAC	角色的访问控制（Role-Based Access Control）
REE	主操作系统（Rich Execution Environment）
SAST	静态代码安全分析（Static Application Security Testing）
SBOM	软件物料清单（Software Bill of Materials）
SCCs	标准合同条款（Standard Contractual Clauses）
SCMP	软件配置管理计划（Software Configuration Management Plan）
SFTP	SSH 文件传输协议（SSH File Transfer Protocol）
SLSA	软件制品供应链级别（Supply-chain Levels for Software Artifacts）
SSD	固态硬盘（Solid State Disk）
SSH	安全外壳协议（Secure Shell）
SSL	安全套接字层（Secure Sockets Layer）
TEE	可信执行环境（Trusted Execution Environment）
TLS	传输层安全（Transport Layer Security）
UML	统一建模语言（Unified Modeling Language）

5 软件保障对象

5.1 软件模型

软件模型是对软件系统结构、行为和交互的抽象描述（如UML图、架构图），是软件开发过程中的设计蓝图。

模型保障的目的在于通过审查和验证模型，确保架构设计的正确性，防止严重缺陷的引入，降低后期修复缺陷的成本。

5.2 软件测试工具、设备与测件

在软件测试的过程中，测试工具是用于自动化执行测试、生成报告的软件；测试设备是执行测试所需的硬件或仿真环境；测件是测试的具体对象，通常指软件本体或其特定功能模块。

测试保障的目的在于通过标准化的测试环境，验证软件在各种条件下的功能正确性和性能表现。

5.3 软件安装文件

软件安装文件指用于在目标环境中自动完成软件安装、升级、卸载、初始化与验证的一组可执行脚本或脚本集合。安装文件通过声明或编排安装步骤，负责完成软件组件投放、依赖检查与获取、目录/权限设置、服务注册与启动、配置生成或注入、数据库/中间件初始化、健康检查与回滚等操作，确保软件在规定环境中按预期可重复地部署。

软件安装文件保障的目的在于确保安装包的完整性和一致性，防止安装失败或关键文件缺失，保障安装包未被篡改，防止“后门”或病毒植入。

5.4 软件可执行文件

软件可执行文件是计算机系统可直接运行的二进制代码，是软件功能的核心载体。

可执行文件保障的目的在于保障可执行在传输和存储过程中的完整性，防止文件被损坏或篡改，确保软件按预期运行。对于商业软件，可执行文件保障同时承担保护知识产权和防止逆向工程的责任。

5.5 软件配置文件

软件配置文件指用于描述和控制软件在构建、部署、运行、运维过程中行为与资源绑定关系的文件集合，其内容以可配置参数形式存在，能够在不修改源代码或仅最小化修改的情况下改变软件的运行方式。

软件配置文件保障的目的是通过对配置文件这一类关键软件资产进行受控管理与技术防护，确保软件在全生命周期中能够按预期、可重复、可追溯、可恢复且安全合规地运行。

5.6 软件数据

软件在运行期间产生或处理的数据信息，包括用户数据、日志文件、数据库内容等。

软件数据保障的目的在于确保数据的保密性（防止泄露）、完整性（防止被篡改）和可用性（防止丢失），是用户隐私保护的核心，以及系统进行故障排查和安全追溯的基础。

5.7 软件源代码

源代码是软件开发者编写的、包含算法逻辑和功能描述的文本文件。

软件源代码保障的目的在于保护源代码的安全，防止源代码泄露。确保其来源清晰可追溯，以及保护商业机密和核心算法。

5.8 用户手册

软件用户手册是指导用户理解并正确操作软件的说明文档，包含功能介绍、操作步骤和故障排除指南。

用户手册保障的目的在于通过提供正确的使用指南，防止用户因误操作导致的安全事故或数据损坏。在部分领域（如医疗、金融），合规的用户手册也是满足监管要求、确保产品合法性的必要条件。

6 软件保障性定量指标

与软件保障性相关的定量指标包括管理保障、使用保障和人员保障三个方面，具体指标含义及其计算方法如表1所示。

表 1 软件保障性指标含义及计算方法

序号	类别	指标	含义	计算方法	说明
1	管理保障	平均保障延误时间	在规定的时间内，保障事件延误时间的平均值。	X=A/B A 表示保障延误的总时间； B 表示保障事件总数。	0≤X，X 越小越好。
2	管理保障	平均管理延误时间	在规定的时间内，管理延误时间的平均值。	X=A/B A 表示管理延误的总时间； B 表示保障事件总数。	0≤X，X 越小越好。
3	管理保障	保障设备满足率	在规定的时间内，能够提供使用的保障设备数与需要提供的保障设备总数之比。	X=A/B A 表示在规定的时间内，能够提供使用的保障设备的数量； B 表示需要提供的保障设备总数。	0≤X，X 接近 1 越好。
4	管理保障	备件满足率	在规定的时间内，能够提供使用的备件数与需要提供的备件总数之比。	X=A/B A 表示在规定的时间内，能够提供使用的备件数量； B 表示需要提供的备件总数。	0≤X，X 越接近 1 越好。

表 1 软件保障性指标含义及计算方法（续）

序号	类别	指标	含义	计算方法	说明
5	管理保障	备件利用率	在规定的时间内，实际使用的备件数与实际拥有的备件数之比。	X=A/B A 表示在规定的时间内，实际使用的备件数量； B 表示实际拥有的备件数量。	0≤X≤1。
6	管理保障	平均故障修复时间	故障报告到完全恢复的平均时间。	X=A/B A 表示所有故障事件中，从故障报告到完全恢复的总时间； B 表示故障事件总数。	0≤X，X 越小越好。
7	管理保障	计划执行率	计划中的维护/升级任务实际完成的比例。	X=A/B A 表示实际完成的维护/升级任务； B 表示计划中的维护/升级任务。	0≤X≤1，X 越接近 1 越好。
8	管理保障	故障定位效率	从故障报告到确定根本原因的平均耗时。	X=A/B A 表示故障报告到确定根本原因的总耗时； B 表示故障事件总数。	0≤X，X 越小越好。
9	使用保障	变更成功率	变更（如补丁、配置调整）是否导致故障的比例。	X=A/B A 表示未导致故障的变更总数； B 表示系统变更的总数。	0≤X≤1，X 越接近 1 越好。
10	使用保障	需维护频次	系统在长期运行过程中，单位时间内（如每年）所需的维护次数。	X=A/B A 表示一定时间内的维护次数； B 表示时间长度。	0≤X。
11	使用保障	热更新成功率	系统在不中断服务运行的前提下，成功完成热更新（如模型、模块、算法、配置文件等升级）操作的比例。	X=A/B A 表示热更新成功次数； B 表示热更新总次数。	0≤X≤1，X 越接近 1 越好。
12	使用保障	升级回滚成功率	系统升级失败或出现异常后，能否成功恢复到升级前稳定版本的比例。	X=A/B A 表示回滚成功次数； B 表示升级失败总次数。	0≤X≤1，X 越接近 1 越好。
13	使用保障	自动化升级比率	在所有升级任务中，由系统自动完成、无需人工干预的升级比例。	X=A/B A 表示自动升级的次数； B 表示系统升级的总次数。	0≤X≤1，X 越接近 1 越好。
14	使用保障	平均升级耗时	从升级操作正式开始（即升级触发/指令下发）到系统完成升级并恢复正常服务状态之间所消耗的平均时间。	X=A/B A 表示所有升级事件的耗时总数； B 表示升级事件总数。	0≤X，X 越小越好。

表 1 软件保障性指标含义及计算方法（续）

序号	类别	指标	含义	计算方法	说明
15	使用保障	监控覆盖率	系统对所有关键组件和服务的监控覆盖程度。	X=A/B A 表示实际被监控的组件数； B 表示组件总数。	0≤X≤1。
16	使用保障	指标采样频率	系统监控数据的采样频率，即每秒钟或每分钟收集的数据点数量。	X=A/B A 表示一定时间内采样点数量； B 表示时间间隔。	0≤X。
17	使用保障	系统可用性	系统正常运行的时间比例。	X=A/B A 表示系统正常运行的总时间； B 表示系统的总运行时间。	0≤X≤1，X 越接近 1 越好。
18	使用保障	功能完整性	软件在运行中是否能完整使用所有预期功能。	X=A/B A 表示实际能够实现的功能数； B 表示软件系统预期实现的功能总数。	0≤X≤1，X 越接近 1 越好。
19	使用保障	资源利用率	CPU、内存、磁盘等资源使用情况。	X=A/B A 表示软件系统对某资源的使用量； B 表示该资源的总量。	0≤X≤1。
20	使用保障	错误率	单位时间内出现的错误或异常次数。	X=A/B A 表示一段时间内出现的错误或异常次数； B 表示时间长度。	0≤X，X 越小越好。
21	人员保障	人员培训覆盖率	团队成员接受专业培训的比例。	X=A/B A 表示接受培训的成员数； B 表示团队成员总数。	0≤X≤1，X 越接近 1 越好。
22	人员保障	人员流动率	一定时间内团队成员离职的比例。	X=A/B A 表示离职的员工数； B 表示团队成员总数。	0≤X≤1。
23	人员保障	响应时效	人员对故障报告的平均响应时间。	X=A/B A 表示所有故障报告的总响应时间； B 表示故障报告的总数。	0≤X，X 越小越好。
24	人员保障	保障能力	人员对系统问题的处理能力。	X=A/B A 表示保障人员能够处理的问题总数； B 表示软件系统出现的问题总数。	0≤X≤1，X 越接近 1 越好。

7 软件保障性定性要求

7.1 软件开发阶段保障

7.1.1 质量保障

质量保障是贯穿于开发全过程的系统性工程，其目的是确保软件产品满足既定技术规范和用户期望。开发阶段的质量保障要求包括：

a) 文档评审：对需求规格说明书、设计文档等进行严格评审，确保其清晰、完整、无歧义且可验证；

b) 代码检查：通过同行评审或工具扫描，检查代码是否遵循编码规范、是否存在逻辑错误、潜在漏洞和可维护性问题；
c) 过程审核：定期审核开发过程是否遵循已定义的流程和标准，确保过程的稳定性和一致性；
d) 缺陷管理：建立并执行严格的缺陷跟踪、分析和管理流程，确保每一个被发现的缺陷都能得到有效处理；
e) 质量保障组织：为确保客观性和权威性，质量保障组织（QA团队）应在组织架构上独立于开发团队，保证质量保证活动的稳定执行和持续改进，避免因项目进度压力而牺牲质量标准；
f) 特定领域要求：在嵌入式等高可靠性、高安全性领域，质量保障要求更为严格，保障活动应当贯穿于需求分析、架构设计、详细设计、编码、构建、测试和发布的每一个环节，形成完整的证据链。

7.1.2 安全保障

开发阶段，软件安全保障方面的要求包括：

a) 开发人员应当基于系统级的危害分析（Hazard Analysis）建立软件安全模型，分析潜在的危险和攻击向量，并将其转化为明确、可验证的安全需求，清晰地标识在软件需求规格说明书中；
b) 开发方应制定全面的安全性保证策略，以系统性地排除或最大限度降低潜在风险；
c) 设计阶段应当包含明确的安全设计特征和方法，以实现已识别的安全关键需求，例如，采用防御性编程、最小权限原则、安全设计模式等，并确保这些安全设计的实现是可测试、可验证的；
d) 开发人员应当遵循安全编码标准（如避免使用不安全的函数、进行严格的输入验证、正确处理异常和错误等），以减少在代码实现阶段引入的安全漏洞；
e) 开发阶段所需要的软件工具应通过官方镜像与仓库进行下载，不应随意添加不明来源的第三方仓库。

7.1.3 合规性

合规性确保开发活动遵循公认的标准、法规和合同约定，合规性要求包括：

a) 软件开发过程应遵循国际或国家标准，如GB/T 8566、ISO/IEC/IEEE 12207等标准对软件生命周期过程的规定；
b) 对开发过程进行周期性的评审和审计，以验证其是否符合预定义的软件过程和安全要求，确保合规性得到持续满足；
c) 代码需避免侵犯第三方专利/著作权，使用开源组件时应当遵守许可证；
d) 交付物包含《知识产权声明文件》，明确版权归属及使用限制。

7.1.4 配置管理

配置管理确保软件开发过程的可控性、可追溯性与一致性，覆盖版本控制、基线管理及变更控制，要求包括：

a) 制定专门的配置管理计划（SCMP），明确管理目标、组织结构、职责分工以及具体的操作流程；
b) 建立软件基线，在开发过程中，软件经历的多个基线（如需求基线、设计基线等），每一个基线都是一个“冻结”状态，未经授权不得修改；
c) 制定统一的命名规则和版本号规则，确保每个软件构件（源代码、文档、配置文件等）都有唯一的标识；
d) 将所有配置项存放在受控的仓库中，确保统一管理；

e) 任何代码或文档的修改应当通过变更控制评审（CCB），变更申请需要经过评估、审批、实现、验证和归档等完整流程，未经批准的修改应当被阻止；

f) 建立审计机制，定期或在关键节点（如里程碑审查）对配置管理过程进行检查和审计，验证是否符合预定的管理计划和标准。

7.2 软件测试阶段保障

7.2.1 测试的核心目标与范围

测试阶段的主要目标是提供客观证据，证明软件产品在功能和非功能方面均达到了预定要求。其范围应覆盖：

a) 需求验证：确保所有在需求规格说明书中定义的功能和约束条件都得到了正确实现和满足；

b) 非常规条件测试：验证系统在异常输入、高负载、高压力以及其他非正常运行条件下的行为是否正确、健壮和安全。

7.2.2 可靠性、安全性与效率指标与评估

对软件保障性是一个受多重因素影响的复杂特性，对其评估需要一个综合性的指标体系，要求：

a) 建立综合评估框架，采用基于多指标体系的综合评价方法，结合定性与定量分析，对软件保障性进行评估；

b) 测量与软件保障活动相关的软件质量特性指标，包括：

1） 可靠性指标：衡量软件在规定时间和条件下无故障运行的能力；

2） 安全性指标：衡量软件保护信息和数据，使其免受未授权访问、使用、泄露、破坏、修改或销毁的能力，具体指标可细分为：保密性、完整性、抗抵赖性、可核查性、真实性等；

3） 效率指标：衡量软件在适当的资源配置情况下，能够以更快速度完成更多任务的能力，包括响应时间、吞吐率和资源利用率等指标。

7.2.3 测试验证方法与标准

为了实现上述保障性目标，需要采用多样化的测试与验证方法，包括：

a) 安全测试方法：除了常规的功能测试，还需引入专门的安全分析与测试方法，例如：

1） 危害分析（Hazard Analysis）、 故障模式与影响分析（FMEA）、故障树分析（FTA）等，用于在测试前识别潜在的风险点和测试重点；

2） 渗透测试、模糊测试、静态/动态代码安全分析（SAST/DAST）等，用于主动发现和验证安全漏洞；

b) 测试覆盖度分析：使用代码覆盖率、需求覆盖率等指标来度量测试的完备性，确保测试活动充分覆盖软件的关键部分，测试覆盖度报告应提供给安全和质量人员进行分析；

c) 非测试验证方法：对于某些难以通过传统测试手段验证的保障性需求（如某些架构级安全策略），可以通过正式评估、技术审查或演示等方式进行验证；

d) 标准化：测试活动应参考相关标准，如GB/T 30998、ISO/IEC/IEEE 29119、GB/T 38634等标准对测试活动的规定。

7.3 软件运行阶段保障

7.3.1 部署前与部署过程保障

此阶段的保障性要求重点在于确保服务的持续可用性、应对突发故障的快速恢复能力以及对运行状态的有效监控，要求：

a) 在部署前，应当制定详尽的软件保障方案和计划，该计划应明确保障的目标、范围、所需资源（人员、备件、工具等）、职责划分以及具体的实施流程；

b) 部署过程中，需确保软件的发布版本正确无误，配置项完整，所有在测试阶段发现的关键缺陷都已修复或有明确的后续处理计划，同时，需要进行发布前的最终验证，确保软件在目标环境中能够正常安装和启动。

7.3.2 运行保障

系统上线后，保障工作的重心转向维持其稳定运行，要求：

a) 建立全面的监控体系，实时收集系统健康状况、性能指标和业务指标，设定合理的告警阈值，确保在问题发生或出现前兆时，运维人员能第一时间获得通知；

b) 建立标准化的故障处理流程，包括：

1） 故障记录与分析：详细记录故障现象、发生时间、影响范围，并进行根本原因分析；

2） 故障隔离与恢复：故障发生时，快速定位故障模块或服务，并采取措施将其隔离，避免影响扩大，同时，启动恢复程序，尽快恢复服务；

3） 多层次技术支持：根据故障的严重程度，提供包括远程技术支持、派遣工程师现场处理、软件降级使用等在内的多层次支持策略。

7.3.3 灾难恢复

灾难恢复是软件运行保障的延伸，旨在应对大规模或毁灭性的故障，要求：

a) 数据备份与恢复：建立定期、可靠的数据备份机制，备份数据应异地存储，并定期进行恢复演练，确保备份的有效性；

b) 高可用架构：如负载均衡、多活数据中心等，通过冗余设计来抵御单点故障；

c) 快速恢复能力：设计和演练在主站点发生灾难时，能够快速切换到备用站点的能力，并明确恢复时间目标和恢复点目标。

7.4 软件升级维护阶段保障

7.4.1 软件维护

软件生命周期中，升级维护的效率和质量，取决于于软件本身设计的可维护性与保障性。此阶段的保障性要求核心在于以可控、安全、合规的方式管理变更，确保系统在演进过程中持续稳定。

7.4.2 变更管理与控制

所有变更，无论是修复缺陷、增加功能还是适应环境，都应当纳入严格的变更管理流程，具体要求包括：

a) 建立覆盖从变更请求、影响分析、风险评估、审批、实现、测试到部署的标准化全流程管理机制；

b) 对于任何变更，特别是对安全关键软件的变更，应当进行深入的安全影响分析，评估变更是否会引入新的危险、是否影响现有的安全控制措施、是否会改变系统的安全等级，所有对安全有影响的变更，应当得到软件安全人员的审查和批准；

c) 变更管理应当与配置管理紧密集成，确保每一次变更都与特定的软件版本相关联，所有文档和代码的更新都受控，并保留完整的变更历史记录以备审计和追溯；

d) 确保变更不违反安全策略和合规要求，特别是对于涉及用户数据的修改；

e) 应当制定明确的回退策略，以便在变更失败或出现异常时，能够快速恢复到原始状态；

f) 在生产环境实施前，应当在测试环境完成功能验证和安全测试，确保变更的可行性。

7.4.3 业务连续性与回滚策略

升级维护操作本身可能引入风险，应当有预案来保障业务连续性，具体要求包括：

a) 采取预防性维护，如主动打补丁、升级系统组件等，以避免未来发生重大故障、保障业务的连续性；

b) 为重要的升级或变更准备回滚计划，一旦新版本出现严重问题，应当能够快速、安全地恢复到变更前的稳定状态，系统和数据的恢复程序应事先准备，并确保可用。

7.4.4 合规与审计

整个软件维护周期内都应当保障维护过程满足合规性与审计要求，具体要求包括：

a) 维护记录与审计：应当对所有维护活动进行详细记录，形成维护档案，内容包括故障信息、错误原因、变更详情、维护时间、操作人员等；

b) 审核策略：应建立并执行审核策略，定期确认配置信息的连续完整性和信息安全状况，确保所有变更都符合内部政策和外部法规的要求。

7.5 软件培训阶段保障

7.5.1 人员培训

软件培训保障的目的与意义在于通过系统性的知识传授与规范落地，将抽象的合规要求转化为员工的日常操作能力。培训的目的包括：

a) 让人员掌握最新的开发工具、测试方法和运维脚本，提升解决实际问题的能力，从而减少因技术盲区导致的缺陷和故障；

b) 将组织的安全标准、代码审查流程和变更审批机制内化为团队的行为准则，确保所有制品都符合内部审计和外部监管的要求；

c) 提升人员的安全防御意识，使其能够主动识别潜在的安全威胁和质量风险，从根本上提升组织的业务连续性。

7.5.2 培训内容与方式

培训内容应具有针对性和实用性，培训课程应覆盖：

a) 系统与环境知识：针对特定系统及其运行环境的深入培训，使其了解系统的架构、功能和限制；

b) 安全与质量意识：关于常见安全漏洞、系统危害、组织安全实践和质量规范的通用培训，提升全员的风险防范意识；

c) 操作与维护技能：针对运维和支持人员，提供软件的部署、使用、维护、性能调优和常见问题处理等实操培训。

7.5.3 认证与评估机制

确保培训效果需要有效的评估和认证机制，要求包括：

a) 通过考试、实操考核、工作表现评估等方法来验证受训人员是否掌握了必要的知识和技能；

b) 项目的软件安全计划或质量保证计划中，应当明确包含对相关角色的培训要求；

c) 鼓励或要求关键岗位（如安全架构师、高级开发人员）持有特定的国际认证体系（如CISSP等）的专业认证，特别是在金融、医疗等强监管行业。

8 软件保障性支撑技术与方法

8.1 模型保障

软件模型（如UML图、流程图、数据流图、架构图等）是软件工程的核心抽象工具，通过可视化表达系统结构、行为及交互关系，在需求分析阶段消除歧义并精准对齐各方理解，在设计阶段提前暴露逻辑矛盾（如状态机死锁）以降低编码缺陷率，是驾驭系统复杂性、保障功能可靠性与演化可控性的重要工具。对软件模型进行保障的技术方法有：

a) 交叉映射验证：通过矩阵工具检查序列图消息与类图方法的对应关系，确保交互行为均有实现方法，消除未实现接口或冗余方法；

b) 扩展场景：在协作图中验证对象协作逻辑是否与类图操作一致，避免行为断层；

c) 分层平衡机制：对模型进行层级分解验证，高层输入/输出需与底层完全匹配（例：上下文图实体与一级图实体一致）；
d) 使用部署图验证物理架构与逻辑分层的兼容性；
e) 使用自动化工具（如NuSMV、SPIN）检测状态图死锁、时序冲突等问题，生成差异报告；
f) 使用形式化方法（如OCL约束）验证状态转换的数学完备性，确保无歧义逻辑；
g) 通过自动化工具（如Enterprise Architect）自动生成代码骨架，保持模型与代码实现同步；
h) 对状态图转换条件进行数学逻辑检验（如使用Z语言或TLA+），确保状态机无冲突路径；
i) 强制采用统一符号库（如UML图的OMG标准），规避线条交叉、标签歧义等问题；
j) 使用标准化绘图工具（如draw.io、PlantUML）确保符号一致性。

8.2 测试保障

8.2.1 测试工具保障

选用合适的测试工具能显著提升测试效率，减少人为错误，确保测试覆盖的全面性。保障测试工具的技术方法包括：

a) 安全来源验证：仅从官方或认证渠道获取测试工具，验证数字签名确保完整性；
b) 沙盒隔离运行：在独立环境（如Docker容器）中部署测试工具，监控异常行为，防止工具被恶意篡改或感染病毒；
c) 定期漏洞扫描：使用静态分析工具（如SonarQube）扫描测试工具代码，使用动态分析工具（如OWASP ZAP）监控运行时行为；
d) 版本控制与更新：通过配置管理工具统一管理工具版本，自动推送安全补丁；
e) 工具有效性验证：每次使用前校验工具功能（如校验测试脚本哈希值），异常时追溯至上次有效状态；
f) 权限最小化：基于必要性限制工具访问权限，将测试工具仅授权给必要角色。

8.2.2 测试设备保障

测试设备包括仿真环境和专用硬件配置，确保测试环境与生产环境高度一致，保障测试设备的技术方法包括：

a) 设备分级选型：针对不同的测试场景选用不同的测试设备，例如负载测试采用企业级服务器，兼容性测试覆盖主流设备型号；
b) 冗余容灾架构：关键设备（如测试服务器）配置双电源+RAID10磁盘阵列，网络链路采用BGP多线接入；
c) 基础设施即代码（IaC）：以代码（而不是图形化界面或命令行脚本）形式对计算基础设施进行描述、部署和维护，使用Terraform定义硬件配置，确保测试环境与生产环境拓扑一致；
d) 环境快照技术：通过VMware/Vagrant等工具创建环境镜像，故障时快速恢复至基准状态。

8.2.3 备用测件保障

备用测件（如测试套件中的冗余用例）在测试失败时提供快速切换机制，保障测试连续性，保障备用测件的技术方法包括：

a) 数据版本控制：测试数据集纳入Git LFS管理，保留历史版本供回滚；
b) 环境快速重建：基于Kubernetes部署测试集群，故在障时自动调度备用节点；
c) 流量分级切换：新版本测试通过后，先导流部分真实流量至沙盒环境，监控一定时间无异常再全量发布；
d) 自动化回滚：配置Prometheus告警规则，关键指标（如错误率>1%）触发自动回滚。

8.3 软件安装文件保障

8.3.1 完整性保障

软件安装文件的完整性保障是指在软件安装包从开发者发布到最终用户安装的整个生命周期中，确保安装文件没有被篡改、破坏或误传的过程。

完整性保障的核心目标是确保用户下载的安装文件与开发者发布时的原始文件完全一致，未在传输或存储过程中被篡改或损坏，软件安装文件的完整性保障可采取以下方法：

a) 哈希校验：开发者在发布软件时，使用哈希函数（如SHA-256、SHA-3）对安装文件进行运算，生成唯一哈希值并随发布渠道公开。用户下载后在本地计算哈希并与官方值比对，一致则证明文件完整。鉴于MD5、SHA-1已存在碰撞风险，宜优先使用SHA-256、SHA-3。用户可使用操作系统工具（Windows certutil、Linux sha256 sum等）验证。

b) 数字签名：开发者使用私钥对安装文件（或其哈希/清单）进行签名，用户或操作系统安装时用公钥验证签名，验证通过即可同时证明文件未被篡改且来源与签名主体一致。对脚本类安装文件（如PowerShell、Shell脚本）亦宜采用签名机制，并在执行策略中强制验签。

c) 安全传输：分发过程优先采用加密传输协议（如HTTPS/TLS），并通过下载站点的完整性校验能力降低中间人攻击与传输损坏风险，对镜像/包管理仓库可启用“清单校验+拉取校验”以避免拉取到被替换的文件。

8.3.2 来源可信性保障

来源可信性保障的目的是验证安装文件的发布者身份，确保文件确实来源于声称的开发者，而非冒名顶替的攻击者，软件安装文件的来源可信性保障可采取以下方法：

a) 供应链签名：软件厂商对安装包/安装脚本/安装清单进行签名，并要求安装端仅接受来自受控发布链路产生且签名有效的安装文件，对关键发布可附带来源证明用于审计追溯。

b) 代码签名证书：软件厂商向可信证书颁发机构（CA）申请代码签名证书，对安装包进行签名，用户验证时由系统检查证书链是否由受信任CA签发、证书是否吊销、签名是否有效。对于长期分发的安装文件，宜配合时间戳签名，降低证书到期对验证结果的影响。

c) 可信分发渠道与安全离线介质：通过官方站点、受控应用商店、企业内部制品库、受信任的包管理仓库分发，避免非官方镜像站/网盘等不受控来源。对敏感环境可采用具备防篡改包装与受控流转的离线介质（防篡改U盘/光盘）进行交付，并配套交接记录与入库验签流程。

8.4 软件配置文件保障

8.4.1 完整性保障

软件配置文件的完整性保障是指在配置文件从编制、评审、入库、发布、下发、加载到运行的全生命周期中，确保配置内容未被篡改、破坏、误覆盖或误分发的过程。

完整性保障的核心目标是确保运行环境中实际生效的配置与发布基线完全一致，且任何差异都可被发现与阻断。软件配置文件的完整性保障可采取以下方法：

a) 哈希校验与基线清单：对发布配置生成SHA-256/SHA-3等哈希值，形成配置基线清单，在部署下发、容器启动或服务加载前对配置文件重新计算哈希并比对，不一致则拒绝加载或阻断发布。

b) 数字签名与启动时验签：对配置基线清单或配置包进行签名（使用发布方私钥签名），部署侧或运行时使用公钥验签，验签失败视为配置不可信并拒绝启用，从机制上防止配置被替换。

c) 安全传输与受控分发：配置分发链路优先使用HTTPS/TLS、内网专线或受控制品库下发，避免明文传输与中间人篡改，对分发节点启用校验与重放保护。

8.4.2 来源可信性保障

来源可信性保障的目的是验证配置文件的提供者身份与发布链路，确保配置确实来源于授权的组织/流水线，而非冒名顶替或越权人员投放的配置。软件配置文件的来源可信性保障可采取以下方法：

a) 受保护分支与强制评审：配置仓库启用受保护分支、强制合并请求评审、双人复核与最小权限，禁止直接向生产分支推送，配置变更必须关联工单/审批记录。
b) 提交签名/发布签名：对配置提交启用签名，并对发布产物（配置包/基线清单）进行供应链签名，部署侧仅接受签名可信且证书链有效的配置。
c) 发布链路证明与制品准入：配置发布必须由受控流水线产生并推送至受控制品库/配置中心，运行环境只允许从白名单来源拉取（禁止“现场拷贝”“临时网盘”等非受控来源），并保留发布来源证明与审计记录。

8.4.3 正确性与一致性保障

正确性与一致性保障是指确保配置内容在语法、结构、取值范围与依赖关系上正确，且在不同环境/节点上表现一致、可复现，避免因配置错误导致功能异常、性能退化或不可用。可采取以下方法：

a) 格式校验：为配置建立机器可校验约束（如JSON Schema/YAML Schema/参数清单），在集成、交付和部署中执行必填项、类型与范围校验，不通过则阻断合并与发布。
b) 语义规则与策略校验：对互斥/依赖项、阈值上下限、端点格式、超时与限流安全默认值等做语义校验，可引入策略引擎（如基于规则的校验或策略即代码）对高风险配置变更直接拦截。
c) 预检查与灰度验证：部署前进行连通性、证书有效期和依赖可用性预检查，对关键配置变更采用灰度生效与自动回滚策略，验证通过再全量推广。

8.5 软件可执行文件保障

8.5.1 完整性与防篡改保障

完整性与防篡改保障旨在保护可执行文件免受篡改、逆向工程和非法利用，要求软件在安装后以及运行过程中，其二进制代码不被恶意修改，防止功能被劫持或植入后门。软件可执行文件的完整性与防篡改保障可采取以下方法：

a) 运行时完整性校验：在软件启动或运行时，对自身的关键代码段或库文件进行哈希校验，以检测是否被篡改；
b) 代码混淆与加壳：通过对代码进行混淆或使用专业的加壳工具，增加静态分析和逆向工程的难度，从而间接保护代码的完整性，防止被轻易篡改；
c) 可信执行环境（TEE）：TEE（如ARM TrustZone）通过硬件隔离技术，创建一个独立于主操作系统（REE）的安全环境，将核心的可执行代码和数据置于TEE中运行，利用硬件级别的保护机制，防止来自REE的恶意软件对其进行访问和篡改；
d) 安全启动：通过构建一个从硬件信任根开始的信任链，在设备启动的每个阶段（从Bootloader到操作系统内核，再到驱动和应用程序）都对其数字签名进行验证，只有签名验证通过的软件组件才被允许加载和执行，能够有效地防止恶意软件或被篡改的系统文件在启动阶段被加载，从而确保整个软件栈的初始完整性。

8.5.2 保密性保障

保密性保障主要关注保护可执行文件中包含的商业秘密和知识产权，如核心算法、专有协议等，防止被非法窃取或逆向分析。可执行文件的保密性保障可采取以下方法：

a) 代码混淆：对二进制指令序列进行变形处理，使其难以理解，具体方法包括：
   1） 控制流混淆：通过添加无意义的分支、循环或扁平化结构，打乱代码的正常执行路径，防止静态分析工具还原出清晰的函数调用；

2） 数据混淆：对关键数据进行加密或打乱，如使用数组替换、指针混淆等，防止逆向者通过修改数据包或硬编码查找关键常量；
3） 模糊判定：插入看似有逻辑但永远为真或假的判断语句，迷惑分析工具；
4） 代码虚拟化：将代码转换为一种自定义的虚拟机指令集，只有对应的虚拟内存（VM）才能运行；

b) 反调试与反跟踪：在代码中嵌入检测调试器的指令（如检测硬件断点、软件断点、调试器进程等），如果检测到调试器则自毁或进入死循环，阻断动态分析；
c) 敏感数据加密：对软件中硬编码的密钥或敏感配置进行加密存储，只有在运行时解密，防止攻击者通过内存转储或静态扫描直接获取关键信息；
d) 软件水印：在代码中嵌入特定的标识（如特定的指令序列或数据块），即使代码被篡改或压缩，标识仍能被检测，用于软件溯源，防止他人盗版或伪造软件。

8.5.3 可用性保障

可用性确保软件能够在需要时被用户正常访问和使用，软件可用性保障主要通过高质量的软件工程实践来实现，以应对各种预料之外的运行环境和输入，避免程序崩溃或无响应。可执行文件的可用性保障可采取以下方法：

a) 冗余架构设计：采用高可用性架构，如双机热备、集群部署和负载均衡等，避免单点故障。例如，关键组件（服务器、网络设备、存储）应配置物理或逻辑冗余，确保在单个节点失效时系统自动切换至备用节点，维持服务连续性。
b) 故障转移机制：建立无缝故障检测与转移方案，包括心跳监控、选举超时优化和自动重连功能。系统应定期验证节点状态（如每30秒检测一次），并设置灵敏的故障阈值（例如选举超时≤500ms），以快速触发转移。
c) 容错与预防性设计：引入事务处理、预测模型和组件隔离策略，减少潜在故障点。例如，通过模块化设计和分布式架构提升系统容错性，确保部分节点故障时不影响整体服务。

8.6 软件数据保障

8.6.1 软件数据类别

软件数据包括用户在使用过程中产生的业务数据、应用程序自身的配置数据等。根据其状态，可分为静态数据、运行数据和传输数据。

8.6.2 静态数据保障

静态数据是指存储在硬盘、SSD或数据库中的数据，保障措施主要针对数据被窃取或非法读取。静态数据保障的核心技术是加密。对于静态数据中的敏感数据（如个人信息、财务数据、凭证等），应使用强大的、业界公认的加密算法（如AES-256）进行加密存储。对于配置文件中的密钥、密码等敏感信息，也应加密处理，禁止明文存储。

加密的有效性高度依赖于密钥管理，可以使用以下方法来强化密钥管理：

a) 部署专用密钥管理系统（KMS）或硬件安全模块（HSM）：将密钥与数据物理分离存储于独立系统（如云KMS或本地HSM），通过硬件级保护防止密钥被窃取或篡改。
b) 实施密钥轮换策略：定期更换加密密钥（如每90天或每日自动轮换），降低长期密钥泄露风险。轮换频率应根据数据敏感度动态调整。
c) 强化访问控制与审计：结合基于角色的访问控制（RBAC）和多因素认证（MFA），限制密钥操作权限（如仅授权管理员可访问密钥），并记录全生命周期日志（包括生成、分发、使用和销毁），实现操作可追溯性。异常行为（如高频密钥访问）应触发实时告警。

d) 分层加密：对高敏感数据实施信封加密，即用主密钥加密数据密钥，再以非对称加密（如RSA）保护主密钥，实现混合加密架构。

e) 密钥生命周期管理：自动化密钥生成、分发、撤销和销毁流程，确保密钥在生成、存储及废弃阶段的安全。

8.6.3 运行数据保障

运行数据是指程序运行时加载到内存中的数据。保障措施主要针对内存读取和侧信道攻击。运行数据的保障可采用以下方法：

a) 内存加密：通过CPU或操作系统层面的加密机制，确保数据在内存中始终保持加密状态，只有在CPU的安全执行环境中，数据才会被解密用于计算，内存加密能够有效防止攻击者通过物理接触（如冷启动攻击）或内存侧信道攻击读取内存中的敏感数据；

b) 白盒密码学：通过混淆和混合算法，将密钥和算法深度融合进代码逻辑中，即使攻击者拿到完整的二进制文件，也难以逆向出有效的密钥或算法，防止攻击者通过调试器、逆向工程或内存转储提取硬编码在程序中的加密密钥；

c) 安全审计与日志保存：对运行中的关键操作（如管理员登录、密码修改、数据导出）进行详细记录和实时监控，日志数据采用加密存储和防篡改技术进行保护，确保审计信息的真实性和可靠性，即使攻击者成功获取了运行数据，也能通过审计日志追踪责任人。

8.6.4 传输数据保障

传输数据是指在网络中传输的数据，保障措施主要针对数据在传输过程中被窃取或篡改。传输数据的保障可采用以下方法：

a) 采用加密传输协议：使用SSL/TLS、HTTPS或SFTP等安全协议，对传输通道进行端到端加密，防止数据在传输过程中被窃听或截获；

b) 禁用不安全算法：避免使用MD5、DES-CBC、SHA1等弱加密算法，优先选用AES-256或TLS 1.3等强加密标准，以抵御暴力破解攻击；

c) 协议兼容性与更新：确保协议版本与最新安全标准同步，定期更新以修补已知漏洞，例如，通过TLS协议实现前向保密性；

d) 端到端数据加密：对敏感数据（如个人信息或核心代码）实施字段级或文件级加密，使用对称加密（如AES）或非对称加密（如RSA），确保即使数据被截获也无法解读；

e) 完整性校验机制：通过数字签名、哈希函数（如SHA-256、SHA-3）或校验码验证数据未被篡改，例如在传输前后生成哈希值比对，或使用数字签名确保数据来源可信；

f) 抗抵赖性与时间戳：结合数字签名和时间戳技术，防止数据发送方否认传输行为，并设置超时机制（如请求有效期≤5分钟）抵御重放攻击；

g) 双向身份认证：对通信双方实施严格身份验证，例如，通过证书、Token或Session机制，确保仅授权用户或系统可参与数据传输；

h) 分级授权管理：根据数据敏感度设置访问权限，例如，3级以上数据需审批授权并加密传输，4级数据原则上禁止对外传输；

i) 终端与网络隔离：部署防火墙、入侵检测系统（IDS）及网络分区隔离，限制未授权终端接入，并实施准入控制。

8.7 软件源代码保障

8.7.1 保密性与访问控制

源代码是软件的原始设计蓝图和核心知识产权。保护源代码的安全、确保其来源清晰可追溯，是构建可信软件的根基。软件源代码的保密性保障与访问控制可以采用以下方法：

a) 最小权限原则：为开发者分配仅满足其工作所需的最小代码访问权限；

b) 基于角色的访问控制（RBAC）：通过定义不同的角色（如开发者、测试者、访客）并为其分配预设的权限集，来简化和规范权限管理；
c) 安全存储：源代码仓库应存储在受严格物理或网络隔离保护的服务器上，并对仓库本身进行加密；
d) 身份验证：访问源代码仓库应当经过强身份验证，使用多因素认证。

8.7.2 完整性与审计追踪

源代码完整性是指确保代码未被篡改、准确反映开发者意图的状态。审计追踪是指对代码变更和审查过程的系统化记录与分析，旨在识别漏洞并确保合规。源代码完整性保障与审计追踪可以采用以下方法：

a) 使用Git等版本控制系统，并强制要求开发者对每一次代码提交进行签名，以确保每次变更的来源可信且不可否认；
b) 对源代码仓库的所有访问和操作行为（如克隆、拉取、推送、权限变更）进行详尽的日志记录和实时监控，以便及时发现异常行为并进行追溯；
c) 在源代码中嵌入不可见的、唯一的标识信息，一旦代码泄露，可以根据水印追溯泄露源头。

8.7.3 供应链安全与来源追溯

软件的供应链安全与来源追溯目的是确保代码从开发到交付的每个环节都安全可靠，并能清晰追踪其来源，可采用的方法包括：

a) 软件物料清单（SBOM）：SBOM是一份详细列出软件产品所有组件（包括开源库、第三方包及其版本、许可证等）的表单，能够有效地提升软件的透明度，为识别漏洞、管理依赖、和响应安全事件提供帮助；
b) 软件工件的供应链层级（SLSA）：SLSA是一个由谷歌提出的供应链安全框架，旨在通过一系列从低到高（Level 1-4）的安全要求，来保证软件制品从源代码到最终用户全过程的完整性和可追溯性；
c) Sigstore：Sigstore是一个由谷歌等组织发起的开源项目，提供一套简化的工具链和公共基础设施，帮助开发者为软件制品（如容器镜像、二进制文件、SBOM等）生成数字签名，并通过透明日志进行验证；
d) 零信任架构：将零信任原则应用于软件供应链，对每一次访问、每一次构建、每一次部署都进行严格的身份验证和授权，持续验证其安全性。

8.8 用户手册保障

8.8.1 内容合规性

软件用户手册的合规性旨在确保内容合法、真实且不产生不必要的法律责任，合规性保障要求：

a) 用户手册应当包含潜在风险、使用限制、紧急处理步骤等内容，确保用户在使用前能明确知晓潜在的危险和注意事项，防止因误操作导致事故；
b) 用户手册应明确哪些情况是免责的（如误用导致的后果），以及哪些情况下制造商/开发者不承担责任；
c) 用户手册描述的功能应当与软件的实际功能一致，不得出现“功能漂移”或描述与实际不符的情况；
d) 对于核心功能、特殊功能、异常情况处理（如“如何处理软件卡死”）应当有详细描述，不得遗漏重要操作步骤；
e) 用户手册应当包含软件版本号和手册版本号，并注明发布日期；
f) 需要有修改记录或变更日志，明确每次更新的内容，确保用户能了解更新带来的变化；

g) 用户手册中的图例、文字说明应与实际操作界面保持一致，防止因版本差异造成误导；
h) 用户手册应当使用非专业术语或提供术语解释，语言应清晰、简洁，避免模糊不清的表述；
i) 用户手册需包含版权声明、许可证协议、合规声明（如符合特定行业标准）等法律文本；
j) 如果软件涉及数据收集，用户手册中应包含数据处理说明和隐私政策的引用或概要；
k) 对于特定领域的软件（如无人驾驶航空器系统），应当明确哪些不适用的场景，并提供精度、性能等详细技术参数。

8.8.2 敏感信息处理

在软件用户手册中处理敏感信息（如用户个人隐私数据、系统默认密码、加密密钥等）的核心目的是保护数据安全和合规防护，同时确保手册本身不成为泄露风险的渠道，要求：

a) 在编写手册，尤其是在提供配置示例或截图时，应当对可能出现的敏感信息（如个人身份信息、IP地址、用户名、密码、API密钥等）进行脱敏处理，可以使用专业的数据脱敏工具，通过遮蔽、替换、加密等方式处理这些信息；
b) 对于系统默认密码、默认管理员账户、加密密钥等信息，不应明文展示，可以使用“请使用系统生成的随机密码”或“请参阅安全指南获取”进行说明；
c) 涉及个人隐私信息时，应当以显著、清晰的方式进行告知，明确列出要收集的敏感信息类型（如“地理位置”、“健康数据”），说明收集这些信息的具体目的、必要性，并强调已采取的加密传输和存储保护措施；
d) 如果手册涉及敏感数据的导出或存储（如配置文件下载），应当进行加密；
e) 对不再使用的敏感信息及时删除，在手册中说明，当用户选择注销或删除账号时，系统会立即进行彻底删除或匿名化处理，不保留任何恢复痕迹。

8.8.3 合规存档与跨境传输

软件用户手册的合规存档目的是确保手册内容的真实性、完整性和可追溯性，防止因文档缺失或篡改导致的法律纠纷或监管处罚。合规存档保障要求：

a) 企业应根据相关法律法规和内部政策，对用户手册的不同版本进行合规存档，确保其在规定期限内的可追溯性和可用性；
b) 存档内容真实准确，不能有虚构或篡改，任何修改都应当留下痕迹，以便日后核查；
c) 存档应当由相关负责人（如法务、项目负责人）签字确认，确保每一份文档都有明确的责任人；
d) 存档应当保留一定的期限（如3年或更久），并妥善保存以备监管机构审查或法律诉讼使用。

软件用户手册在跨境传输时，应当遵守目标国家的数据主权和安全法规，跨境传输保障要求：

a) 在跨境传输前，应当取得个人的明确同意，并告知传输的目的、范围和接收方；
b) 对于敏感信息（如用户个人信息、企业机密），需要进行脱敏或匿名化处理；
c) 需要与接收方签订符合当地法规的数据传输协议（如标准合同条款SCCs），并建立审计日志，记录传输行为。

9 软件保障性的主要活动

9.1 制定保障性方案

保障性方案是实施软件保障性工作的纲领性文件，通过在项目早期形成软件保障性的总体策略与技术路线，明确保障性目标、保障对象、保障边界、实施机制与验证方法，为后续计划与资源落实提供依据。

软件保障性方案需明确保障性总体目标、工作框架及各方职责，重点协调跨部门（开发、运维、用户支持）及跨组织（订购方、承制方、第三方服务商）的协作关系。具体内容应包含以下方面：

a) 软件保障性目标（如可用性、可维护性、可支持性）及技术路径；
b) 保障性管理组织架构、跨职能团队职责及沟通机制；
c) 保障性需求的论证方法与验收标准；
d) 保障性数据（运行日志、性能指标、用户反馈）的采集、分析及共享规范；
e) 对承制方代码质量、文档完备性、技术响应能力的监督措施；
f) 结合软件里程碑（需求评审、版本发布）的保障性专题评审节点；
g) 保障性指标的量化评价方法与工具；
h) 线上监控、用户反馈闭环及持续优化机制；
i) 资源预算（人力、工具链、云资源）与阶段里程碑。

9.2 制定保障性工作计划

软件保障性工作计划是将保障性方案分解为可执行的任务、里程碑与资源安排，确保保障活动可落地、可跟踪、可度量，其核心目标是在满足既定保障性要求且最小化全生命周期成本的前提下，确保软件设计符合保障性标准。保障性工作计划应当包括：

a) 明确产品保障性要求及合同规定的全部保障性工作项目，确保无遗漏；
b) 定义各项目的目标、内容、执行范围、操作流程、交付成果及验收评价标准；
c) 规定管理/实施机构的职责边界，配置必要的组织架构、专业人员及经费支持；
d) 阐述与开发、测试等研制计划中其他工作的接口规则与协作流程；
e) 制定数据资料的获取途径、传递方式、存储规范及访问权限控制机制；
f) 设立阶段性评审计划，明确参与方职责、输入输出物及决策准则；
g) 识别可能影响保障性目标的关键问题，提出解决路径与应急预案；
h) 设定各工作项的里程碑与时间表，确保与整体研发进度严格对齐。

9.3 规划与研制保障资源

规划、获取并验证保障软件交付与持续运行所必需的资源，目的是确保维护活动具备工具、资料、环境与人员能力支撑。该工作涵盖人力和人员、技术资料、保障设备等维度，通过系统性整合需求与约束，确保资源供给与软件生命周期保障需求的有效匹配，主要包括以下活动：

a) 人员与人力规划：确定保障所需的人力资源类型、数量及技能要求，包括开发、维护、技术支持等岗位配置，确保人员能力与保障任务匹配；
b) 训练与训练保障：设计培训体系，开发培训教材、模拟工具及考核机制，保障维护人员掌握软件操作、故障处理等技能；
c) 供应保障规划：规划软件交付后的资源供应链，包括备件管理、版本更新、许可证分配及消耗品补给，确保资源持续可用；
d) 技术资料开发：编制用户手册、维护指南、接口文档等全套技术资料，提供操作规范、故障诊断流程及维护标准；
e) 保障设备与设施研制：开发专用测试工具、诊断设备及软硬件支持环境，建设适配的保障设施（如数据中心、实验室），确保资源与软件兼容；
f) 设计接口管理：定义软件与硬件、外部系统的接口规范，确保保障资源（如诊断工具）能有效集成并协同工作。

9.4 建立软件维护组织

软件维护组织是保障软件在交付运行后的持续可用、可维护、可支持的组织基础，通过建立覆盖“事件响应—故障处置—缺陷修复—变更发布—配置管理—安全处置—知识沉淀”的职责体系与协作机制，确保维护活动有明确的责任主体与统一的流程规范。

软件维护组织应明确组织架构、岗位职责、接口关系与工作机制，重点协调跨部门（研发、测试、运维、安全、用户支持）以及跨组织（订购方、承制方、第三方服务商）的协作关系。软件维护组织的工作职责应包含以下方面：

a) 维护组织架构与职责分工（如运维支持、研发支持、质量支持、安全支持、配置与发布管理），以及各角色的权限边界；
b) 维护分级体系与升级路径，明确受理、定位、处置、升级、关闭的责任分工及时限要求；
c) 明确事件/故障管理机制（告警受理、分级响应、应急处置、恢复验证、通报机制），以及重大事件指挥与沟通流程；
d) 明确变更与发布过程（变更评审、发布窗口管理、回滚策略与演练、发布后验证），确保变更可控、可追溯、可回退；
e) 明确配置管理流程（配置基线、变更审批、漂移检测、权限控制与审计），确保配置一致性与合规性；
f) 进行维护资源与维护能力建设（值班制度、应急预案、人员培训、备件/账号/证书管理），保证持续支撑能力；
g) 建立维护文档与知识管理机制（运行维护手册、安装升级指南、故障案例库、交接清单），确保知识可继承与可复用。

9.5 建立保障性数据收集、分析和纠正措施系统

保障性数据收集、分析和纠正措施系统是实现软件保障性可度量、可分析、可改进的闭环机制，通过对运行维护数据的规范采集、统计分析与根因定位，形成纠正与预防措施并验证有效性。

该系统应覆盖数据采集、数据治理、指标体系、分析方法、纠正措施流程与审计追溯，重点解决多源数据分散、口径不一致、问题不闭环等风险。具体工作内容应包含以下方面：

a) 明确保障性数据范围定义与数据源清单，如运行日志、告警事件、性能与容量指标、发布与配置变更记录、安装/升级结果、缺陷与工单、用户反馈、安全审计与漏洞信息；
b) 建立数据采集与接入规范，包括字段定义、数据格式、时间同步、脱敏要求、采集频率、传输与存储方式，确保数据完整、准确、可用；
c) 建立数据存储、保留期限与访问控制机制，确保数据安全合规与可追溯；
d) 建立保障性指标体系，明确指标计算方法与评价准则；
e) 建立分析与评审机制（周期性趋势分析、异常波动分析、Top问题清单、专项治理），并规定报告频次、参与方与决策规则；
f) 明确根因分析方法与流程（问题分级、定位方法、证据链要求、复现与验证），形成可复核的分析结论。
g) 建立纠正与预防措施闭环机制，明确措施制定、责任人、完成期限、验证方法、关闭准则与复发防控要求，并将措施落实到设计/代码/配置/流程/培训等可执行改进项；
h) 建立有效性验证与固化机制，将验证通过的措施固化为标准流程、自动化规则或技术控制（如配置护栏、发布准入、漂移检测、证书临期告警）。

10 软件保障性的适应阶段

10.1 软件保障性的适应阶段

软件保障活动过程一般划分为部署前软件保障、移交和部署保障以及部署后软件保障三个阶段，如图 1 所示。

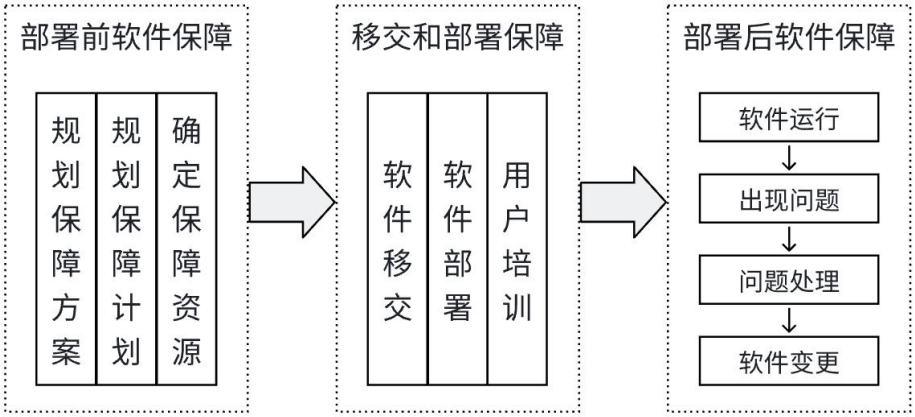

图 1 软件保障活动过程

10.2 部署前软件保障

部署前的软件保障从软件的需求分析阶段开始，并随着软件的设计、实现、集成、测试等各阶段不断深化和细化软件保障内容。部署前的软件保障活动主要包括三个方面：

a) 规划软件保障方案：规定软件保障范围、软件交付后维护过程、估算软件保障生命周期费用等，为软件保障组织构建一套全面的风险防控与质量控制体系，确保软件在全生命周期内符合安全、合规及质量的统一要求。

b) 制定软件保障计划：将安全策略落地为具体的操作标准和时间节点，规定从需求立项阶段的安全需求预审，到发布前的代码审计、渗透测试及合规审计，再到上线后的审计日志实时监控和持续合规监测，通过这一系列的审计与监控措施，确保所有上线软件均经过严格筛选，杜绝安全漏洞与合规风险。

c) 确定软件保障资源：应根据不同的保障模式确定需要的资源，为制定的方案和计划提供必要的人力、物力和数据支撑，配置合规审计员、安全工程师及数据管理员作为保障人员，部署漏洞扫描工具、渗透测试框架及日志审计系统作为技术装备，建立持续更新的审计报告库作为数据支撑，通过多方面的保障，确保软件保障机制具备可持续运行的能力。

10.3 移交和部署保障

软件移交与部署保障的目的在于确保交付物符合标准，并且能够安全、稳定地运行在目标环境中。该阶段的保障内容包括：

a) 软件移交：核对交付清单，确保所有的可执行文件、数据库脚本、配置文件和第三方依赖库均已完整提交且版本号一致。审计移交文档（如运维手册、接口文档、故障处理指南）的准确性，确保运维团队具备接管系统的能力。进行审计日志的配置和审查，确保所有关键操作均留痕可查，为后续的运维审计和安全追溯提供依据。

b) 软件部署：执行安全审计，检查部署制品中是否包含未授权的第三方组件、硬编码密码或后门程序，确保上线系统符合国家网络安全法及行业合规要求。实施环境一致性验证，核对生产环境的硬件、网络及中间件配置是否与预发布环境一致，防止因环境差异导致的故障。执行系统验证测试，确认系统在真实环境中的性能指标和功能需求均符合上线标准。

c) 用户培训：组织安全培训，重点讲解系统的安全操作规范，如强密码策略、访问控制和数据备份流程，防止因用户操作不当导致的数据泄露或系统异常。提供合规培训，明确业务数据的合规要求（如个人信息的收集和存储范围），确保用户在业务操作中不触犯《网络安全法》

或《个人信息保护法》等法律法规。建立用户操作日志审计机制，确保所有关键业务操作均被记录和审计，以便在出现安全事件时能够快速定位和追溯。

10.4 部署后软件保障

部署后软件保障是软件保障活动的主要阶段，依据软件保障方案和软件保障计划，实施两类保障活动：软件运行保障和软件维护保障。

软件运行保障主要体现在软件投入使用后的持续监控与管理，此方面的保障内容包括：

a) 对软件的运行状态进行全天候的监控，及时发现潜在的性能瓶颈和安全风险；
b) 在软件运行受阻或出现故障时迅速响应，采用预案进行处理以恢复正常运行；
c) 记录用户在使用过程中提出的新需求，并形成详实的需求文档和需求变更记录，最终整理报告向上级管理层反馈，以便为软件的后续演进和版本升级提供决策支持。

软件维护保障是针对软件运行期间突发故障的应急响应和修复机制，此方面的保障内容包括：

a) 遵循标准化的维护流程路径，包括故障申报、原因分析、维护计划制定到审批、实施修复、重新部署等环节；
b) 建立完善的维护过程档案管理体系，详细记录故障信息、错误定位、处理耗时、维护类型以及涉及的源代码文件变更等数据。

参 考 文 献

[1] GB/T 8566 系统与软件工程 软件生存周期过程

[2] GB/T 30998 信息技术 软件安全保障规范

[3] GB/T 38634.1 系统与软件工程 软件测试 第 1 部分:概念和定义

[4] GB/T 38634.2 系统与软件工程 软件测试 第 2 部分:测试过程

[5] GB/T 38634.3 系统与软件工程 软件测试 第 3 部分:测试文档

[6] GB/T 38634.4 系统与软件工程 软件测试 第 4 部分:测试技术

[7] ISO/IEC/IEEE 12207 Systems and software engineering — Software life cycle processes

[8] ISO/IEC/IEEE 29119-1 Software and systems engineering — Software testing Part 1: General concepts

[9] ISO/IEC/IEEE 29119-2 Software and systems engineering — Software testing Part 2: Test processes

[10] ISO/IEC/IEEE 29119-3 Software and systems engineering — Software testing Part 3: Test documentation

[11] ISO/IEC/IEEE 29119-4 Software and systems engineering — Software testing Part 4: Test techniques

ICS 35.080
CCS L 77

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35003—2026

复杂软件系统故障预测与健康管理技术要求

Technical requirements for prognostics and health management of complex software systems

2026-02-28 发布　　2026-02-28 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。

本文件起草参与单位：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）、中国科学院声学研究所、长龙航空维修工程有限公司、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、张耀星、侯展意、郝程鹏、曾福萍、杨穗利、廖力鸣、吴梦丹、李璇、冯吉开。

复杂软件系统故障预测与健康管理技术要求

1 范围

本文件规定了复杂软件系统故障预测与健康管理（PHM）的适用对象、指标体系、技术模型方法、数据管理、评价方法、全生命周期实施过程和风险管理要求。

本文件适用于关键软件系统、大型分布式软件系统、长期运行软件系统的PHM技术应用，涵盖软件生命周期各阶段的健康管理活动。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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 8566-2022 系统与软件工程 软件生存周期过程.

GB/T 25000.1-2021 系统与软件工程 系统与软件质量要求和评价(SQuaRE)第 1 部分：SQuaRE 指南.

GB/T 25000.23-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE)第 23 部分：系统与软件质量产品质量测量.

GB/T 25000.51-2016 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第 51 部分：就绪可用软件产品(RUSP)的质量要求和测试细则

GB/T 37739-2019 信息技术 云计算平台即服务部署要求.

T/CICC 35008-2025 复杂软件系统可靠性技术要求

3 术语与定义

GB 25000-2021、GB 25000-2019、T/CICC 35008—2025、GB 8566-2022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

软件故障预测 software fault prediction

通过对软件运行数据的持续监测与分析，识别性能退化趋势和潜在故障征兆，预测未来故障发生的时间、类型及剩余使用寿命的技术方法。

3.3

软件健康管理 software health management

基于健康监测和故障预测结果，制定和执行维护策略、任务规划等决策的管理过程。

3.4

软件健康监测 software health monitoring

通过日志等手段持续或定期获取软件运行参数，实时评估软件健康状态的过程。

3.5

软件状态监测　software condition monitoring

对软件的关键参数进行连续或定期监测，以判断其工作状态是否正常的技术。

3.6

软件健康状态　software health state

系统或部件在特定时刻的整体健康水平，通常用健康等级或数值表示。

3.7

软件健康指数　software health index

用于量化表征软件健康状态的综合指标，通常为 0 到 1 之间的数值。

3.8

软件故障　software fault

软件偏离规定功能的状态，但尚未完全失去工作能力。

3.9

软件失效　software failure

软件完全丧失规定功能的状态。

[来源：GB 25000-2021，4.2，有修改、GB 25000-2019，4.3，有修改]

3.10

软件故障时间　software fault time

从软件开始运行到发生故障的时间间隔。

3.11

软件故障检测　software fault detection

识别软件是否存在故障的过程。

3.12

软件故障诊断　software fault diagnosis

确定故障类型、位置和严重程度的分析过程。

3.13

软件故障隔离　software fault isolation

确定软件故障发生的具体位置或部件的过程。

3.14

软件异常检测　software anomaly detection

识别软件行为偏离正常模式的技术。

3. 15

软件剩余使用寿命　remaining useful life

从当前时刻到软件发生失效或性能退化到不可接受水平的预期时间。

3.16

虚警率　false alarm rate

错误预测故障的比例。

3.17

漏警率　missed alarm rate

未能预测到实际故障的比例。

3.18

基线　baseline

系统正常运行状态下各项指标的参考值或标准值。

[来源：GB 25000-2021，4.1，有修改、GB 8566-2022，3.2，有修改]

3.19

软件退化　software degradation

软件系统性能、可靠性等质量特性随时间推移性能逐渐衰退的现象。

[来源：GB 25000-2019]

3.20

软件生命周期　software life cycle

软件从需求提出、开发、测试、运行、维护到退役的全过程及其管理活动的总称。

[来源： GB 8566-2022，3.1，有修改]

4　缩略语

下列缩略语适用于本文件：

API　应用程序编程接口（Application Programming Interface）

ABAC　基于属性的访问控制（Attribute-Based Access Control）

ARIMA　自回归积分移动平均模型（Autoregressive Integrated Moving Average）

CNN　卷积神经网络（Convolutional Neural Network）

DFT　可测试性设计（Design for Testability）

FaaS　函数即服务（Function as a Service）

GRU　门控循环单元（Gated Recurrent Unit）

HI　健康指数（Health Index）

HM　健康管理（Health Management）

LSTM　长短期记忆网络（Long Short-Term Memory）

MCMC　马尔可夫链蒙特卡洛（Markov Chain Monte Carlo）

MTBF　平均故障间隔时间（Mean Time Between Failures）

MTTF　平均失效时间（Mean Time To Failure）

MTTR　平均修复时间（Mean Time To Repair）

PCA　主成分分析（Principal Component Analysis）

PHM　预测与健康管理（Prognostics and Health Management）

QoS　服务质量（Quality of Service）

REST　表述性状态传递（Representational State Transfer）

RBAC　基于角色的访问控制（Role-Based Access Control）

RNN　循环神经网络（Recurrent Neural Network）

RUSP　就绪可用软件产品（Ready to Use Software Product）

SLA　服务级别协议（Service Level Agreement）

SoS　软件体系系统（System of Systems）

SOA　面向服务的架构（Service-Oriented Architecture）

SQuaRE　系统与软件质量要求和评价（Systems and Software Quality Requirements and Evaluation）

TRL　技术成熟度等级（Technology Readiness Level）

VAR　向量自回归（Vector Autoregression）

5　软件PHM基础

5.1　软件 PHM 基本概念

软件PHM是将传统硬件系统的预测与健康管理技术扩展到软件领域的综合技术体系。它通过持续监测软件系统的运行状态、性能指标和质量特性，识别软件老化、性能退化等异常现象，预测软件故障的发生时间和类型，并提供相应的维护决策支持。

软件PHM的核心目标是：提高软件系统的可用性和可靠性，降低软件维护成本，延长软件系统的有效使用寿命，优化软件维护策略。

5.2 软件 PHM 架构

5.2.1 架构总览

软件PHM系统应采用分层模块化架构设计，自下而上由数据采集与监控层、健康分析与预测层及决策与管理层构成，各层级通过标准化接口进行交互。

架构逻辑如下：

a) 数据采集与监控层：作为感知终端，利用探针及日志采集器，从系统级、服务级、组件级及代码级全方位获取运行数据；

b) 健康分析与预测层：作为核心计算引擎，基于机理模型与数据驱动算法，执行健康度量、故障诊断、故障预测及剩余寿命预测；

c) 决策与管理层：基于分析结果与维护知识库，生成运维策略与资源优化指令，实现从状态感知到主动防御的闭环管理。

5.2.2 数据采集与监控层

数据采集与监控层作为整个PHM系统的基础支撑层，负责全方位、多维度的软件系统状态感知和数据获取。该层主要包括以下几个核心组件：

a) 应用监控模块

实时监控系统的 CPU 使用率、内存占用、磁盘 I/O、网络流量等关键性能指标，通过多维度指标采集技术建立全面的性能监控体系，确保对系统运行状态的全方位感知。

b) 系统监控模块

收集系统运行日志、应用程序日志、错误日志等各类日志信息，运用实时数据流处理技术对大量日志数据进行实时采集和初步处理，为后续分析提供丰富的行为数据。

c) 代码监控模块

监测代码复杂度、代码覆盖率、静态分析结果等代码质量相关指标，集成多源数据融合技术将来自不同监控工具和数据源的信息进行统一整合和标准化处理。

d) 用户行为监控模块

跟踪用户操作模式、功能使用频率、交互响应时间等用户行为数据，通过数据质量保障机制确保采集数据的准确性、完整性和时效性，为健康管理评估提供可靠的数据基础。

e) 运行时监控与静态分析

运行时监控通过在线数据采集技术实时获取系统动态运行数据；静态分析通过代码审查、配置检查等方式获取系统的静态特征信息，两者相结合为上层分析提供全面的数据基础。

5.2.3 健康分析与预测层

健康分析与预测层是整个 PHM 系统的核心计算引擎，承担着从原始监控数据到健康状态评估和故障预测的关键转换任务。该层主要包括以下功能模块：

a) 数据预处理模块

对采集的原始数据进行清洗、归一化、去噪等预处理操作，确保数据质量和一致性，运用特征工程技术从原始数据中提取和构造具有预测价值的特征向量。

b) 健康状态评估模块

运用多维度评估算法，综合分析系统的功能健康度、性能健康度、资源健康度等多个维度，形成综合性的健康状态评估结果，通过模型训练技术运用机器学习、深度学习等算法构建健康评估模型。

c) 寿命预测模块

通过分析软件系统的老化趋势和性能退化模式，预测系统在当前运行模式下的剩余有用寿命，基于时间序列预测技术利用历史数据的时序模式预测未来的系统健康状态演化趋势。

d) 故障诊断模块

运用模式识别和异常检测算法，及时识别系统中出现的各类故障征兆和异常模式，结合自回归预测技术通过建立系统状态的自相关模型实现短期和中期的状态预测。

5.2.4 决策与管理层

决策与管理层作为 PHM 系统的最高决策层，基于健康分析与预测层的输出结果，制定和执行相应的维护决策和管理策略。该层主要包括以下核心模块：

a) 维护策略模块

根据系统健康状态和故障预测结果，制定包括预防性维护、纠正性维护、完善性维护等在内的综合维护计划，并提供代码重构优化功能，根据代码质量分析结果提供代码改进建议。

b) 资源优化模块

通过分析系统资源使用模式和性能瓶颈，提供资源配置优化建议，包括硬件资源调配、软件参数调优等，同时具备资源重新分配功能，基于资源使用预测动态调整资源分配策略。

c) 风险管理模块

评估系统面临的各类风险，制定相应的风险缓解和应急响应措施，集成应急响应功能和故障隔离恢复功能，在检测到严重故障征兆时触发应急处理流程，通过隔离故障组件和启动备用资源实现系统的快速恢复。

6 软件PHM应用对象和场景

6.1 系统级

6.1.1 监测对象

监测对象如下：

a) 完整的企业级软件应用系统，如 ERP 系统、CRM 系统、财务管理系统等；
b) 大型分布式软件系统，如电商平台、社交网络平台、云计算平台等；
c) 软件系统集群，包括多个相互协作的软件系统组成的业务平台；
d) 软件产品线，涵盖同一产品族下的多个软件版本和变体；
e) 跨平台软件生态系统，包括主系统及其相关的插件、扩展和第三方集成；
f) 端到端业务流程系统，涵盖完整业务链条中的所有软件组件。

6.1.2 应用场景

应用场景如下：

a) 企业核心业务系统常态化运维；
b) 大型分布式集群健康状态集中管理；
c) 跨平台软件生态协同监测；
d) 软件产品线全版本健康追溯；
e) 端到端业务流程连续性保障。

6.1.3 核心监测数据

监测数据如下：

a) 结构化数据
    1) 系统整体资源指标：CPU 利用率、内存占用量、磁盘 I/O 总量、网络总流量；

2) 业务运行指标：核心业务交易量、交易成功率、端到端响应时间、业务并发量；
3) 可用性指标：系统运行时长、计划外停机时长、故障发生频次、MTBF、MTTR；
4) 配置参数：系统全局配置项、集群节点配置、跨平台集成参数。

b) 半结构化数据
1) 系统日志：系统启动 / 停机日志、集群状态同步日志、全局错误日志、跨平台交互日志；
2) 业务日志：核心业务流程执行日志、交易状态变更日志、批量处理任务日志；
3) 配置文件：JSON/XML 格式的系统配置文件、集群部署配置文档、第三方集成配置参数。

c) 非结构化数据
1) 故障报告：系统级故障诊断报告、跨平台兼容性问题描述、集群异常事件分析文档；
2) 运维记录：系统升级维护操作记录、故障应急处置预案、健康状态评估报告；
3) 外部依赖文档：第三方服务接口说明、跨平台集成协议文档、安全合规审计报告。

6.2 服务级

6.2.1 监测对象

监测对象如下：

a) 微服务架构中的独立微服务，如用户服务、订单服务、支付服务等；
b) SOA 架构中的业务服务组件和技术服务组件；
c) Web 服务，包括 RESTful API、SOAP 服务、GraphQL 服务等；
d) 消息队列服务，如 Kafka、RabbitMQ、ActiveMQ 等消息中间件服务；
e) 数据库服务，包括关系数据库、NoSQL 数据库、缓存数据库等；
f) API 网关和服务网格中的代理服务；
g) 容器化服务，如 Docker 容器中运行的应用服务；
h) 函数即服务 (FaaS)，如 AWS Lambda、Azure Functions 等无服务器函数。

6.2.2 应用场景

应用场景如下：

a) 微服务架构下独立服务性能监控与故障预警；
b) API 接口稳定性保障与调用质量优化；
c) 消息队列 / 数据库服务资源负载管理；
d) 容器化服务弹性伸缩健康支撑；
e) 无服务器函数执行状态监测与优化。

6.2.3 核心监测数据

监测数据如下：

a) 结构化数据
1) 服务性能指标：API 调用量、调用成功率、接口响应时间、超时次数、并发请求数；
2) 资源占用指标：服务独立 CPU / 内存占用、连接池活跃数、消息队列堆积量、数据库查询耗时；
3) 可用性指标：服务在线时长、服务重启次数、故障恢复时间、SLA 达标率；
4) 契约指标：接口参数校验通过率、版本兼容适配率、服务依赖可用性。

b) 半结构化数据
1) 服务日志：接口调用日志、错误堆栈日志、依赖服务调用日志、契约校验日志；
2) 中间件日志：消息队列生产 / 消费日志、数据库连接日志、缓存命中日志；

3) 配置文件：服务配置参数、容器部署配置、API 网关路由配置。

c) 非结构化数据

1) 故障分析文档：服务熔断 / 降级触发报告、接口兼容性问题分析、依赖服务异常影响评估；

2) 运维记录：服务版本迭代记录、配置变更操作日志、性能优化方案文档；

3) 契约文档：API 接口契约说明、服务依赖关系图谱、SLA 协议文本。

6.3 组件级

6.3.1 监测对象

监测对象如下：

a) 业务功能模块，如登录模块、权限管理模块、报表生成模块等；

b) 技术框架组件，如 Spring 框架组件、Hibernate 组件、Struts 组件等；

c) 第三方库和依赖包，如 Apache Commons、Jackson、Log4j 等开源库；

d) 中间件组件，如应用服务器、消息中间件、缓存组件等；

e) 数据访问层组件，包括 DAO 层、ORM 组件、数据库连接池等；

f) 用户界面组件，如前端页面组件、控件、插件等；

g) 安全组件，如认证组件、加密组件、防火墙组件等；

h) 工具类和公共组件，如日志组件、配置管理组件、监控组件等。

6.3.2 应用场景

应用场景如下：

a) 核心业务功能模块健康状态监测；

b) 技术框架 / 第三方组件兼容性与安全性管理；

c) 数据访问层性能优化与故障定位；

d) 安全组件防护效果评估；

e) 公共工具类可用性保障。

6.3.3 核心监测数据

监测数据如下：

a) 结构化数据

1) 组件运行指标：模块调用频次、执行耗时、错误发生率、配置参数有效性；

2) 质量指标：代码复杂度、静态分析告警数、测试通过率、第三方库版本号及漏洞评分；

3) 资源指标：组件内存占用、线程池活跃线程数、数据库连接池使用率；

4) 安全指标：认证成功率、加密算法执行效率、权限校验通过率。

b) 半结构化数据

1) 组件日志：功能模块报错日志、数据访问操作日志、安全组件审计日志、框架运行日志；

2) 配置文件：组件配置参数、第三方库依赖配置、连接池配置；

3) 测试日志：单元测试执行日志、集成测试结果日志、性能测试报告摘要。

c) 非结构化数据

1) 故障诊断文档：组件异常堆栈信息、第三方库兼容性问题描述、框架报错分析报告；

2) 开发文档：组件设计说明书、接口调用说明、代码注释文档；

3) 安全文档：漏洞扫描报告、安全合规检查记录、加密算法应用说明。

6.4 进程级

6.4.1 监测对象

监测对象如下：

a) 应用程序进程，如 Java 应用的 JVM 进程、.NET 应用的 CLR 进程等；
b) 系统服务进程，如 Web 服务器进程、数据库服务器进程等；
c) 后台守护进程，如定时任务进程、监控代理进程、日志收集进程等；
d) 虚拟机实例，如 Docker 容器实例、Kubernetes Pod 等；
e) 线程和线程池，包括工作线程、I/O 线程、调度线程等；
f) 数据库连接和连接池中的具体连接实例；
g) 网络连接和 Socket 连接实例；
h) 内存区域和堆栈，如 JVM 堆内存、栈内存、方法区等；
i) 文件句柄和系统资源句柄；
j) 进程间通信对象，如共享内存、管道、信号量等。

6.4.2 应用场景

应用场景如下：

a) 关键进程运行状态实时监控；
b) 虚拟机 / 容器实例资源优化；
c) 线程死锁 / 内存泄漏检测；
d) 系统资源句柄泄漏防护；
e) 进程间通信稳定性保障。

6.4.3 核心监测数据

监测数据如下：

a) 结构化数据
   1) 进程状态指标：进程 PID、CPU 占用占比、内存使用量、线程数及状态、进程优先级；
   2) 资源句柄指标：文件句柄数量、数据库连接实例数、Socket 连接数、端口占用状态；
   3) 内存指标：堆内存 / 栈内存使用量、内存泄漏趋势值、垃圾回收频率及耗时；
   4) 线程指标：线程池活跃数、阻塞线程数、线程执行耗时、死锁检测结果。
b) 半结构化数据
   1) 进程日志：进程启动 / 崩溃日志、JVM/CLR 运行日志、容器实例生命周期日志；
   2) 资源监控日志：内存使用趋势日志、CPU 负载波动日志、网络连接状态日志；
   3) 配置文件：进程启动参数、虚拟机配置参数、容器资源限制配置。
c) 非结构化数据
   1) 故障调试文档：进程崩溃核心转储文件、线程死锁分析报告、内存泄漏堆栈信息；
   2) 运维记录：进程重启操作记录、资源调整日志、故障应急处置步骤；
   3) 诊断报告：进程性能剖析报告、系统资源冲突分析文档、进程间通信异常排查记录。

7 软件PHM指标体系

7.1 软件 PHM 定性要求

7.1.1 健康监测定性要求

对软件健康监测能力提出要求，包括监测覆盖度、监测精度、监测实时性等，具体内容如下：

a) 根据软件架构复杂性，明确代码级、模块级、系统级的全层次监测覆盖要求；
b) 根据关键业务功能，明确核心组件的重点监测与异常状态识别要求；
c) 根据性能基线建立需求，明确正常运行模式下的健康基准数据采集要求；
d) 应考虑监测开销对系统性能的影响，明确轻量化监测与深度分析的平衡要求；

e) 必须明确监测数据的完整性与准确性验证机制，确保健康评估的可靠性。

7.1.2 故障预测定性要求

对软件故障预测能力提出要求，包括预测时效性、预测准确性、预测可解释性等，根据预测结果可信度，明确不确定性量化与置信度支撑方法，具体可使用以下方法：

a) 可使用统计置信区间法,基于统计分布模型，结合 Bootstrap 重采样，构建故障发生时间、退化程度的置信区间，量化随机误差不确定性；
b) 可使用贝叶斯概率量化法，通过贝叶斯网络及 MCMC 参数采样，以概率形式输出故障发生置信度，量化模型参数波动影响；
c) 可使用集成模型方差法，利用随机森林、GBM 等集成模型的预测方差反映不确定性，结合加权置信度提升结果可靠性；
d) 可使用蒙特卡洛模拟法，对输入数据与模型参数随机采样模拟，输出预测结果分布特征与置信水平，明确不确定性边界。

7.1.3 故障诊断定性要求

对软件故障诊断能力提出要求，包括诊断准确性、诊断效率、诊断深度等，具体内容如下：

a) 根据故障类型多样性，明确功能故障、性能故障、资源故障的统一诊断要求；
b) 根据故障定位精度，明确从系统级到代码行级的逐层细化诊断要求；
c) 根据诊断时效性需求，明确实时诊断与快速响应能力要求；
d) 应考虑复杂故障场景，明确多重故障的关联分析与根因识别要求；
e) 应在诊断过程中提供证据链，明确诊断结论的可追溯性与可验证性要求。

7.1.4 健康评估定性要求

对软件健康状态评估能力提出要求，包括评估全面性、评估准确性、评估一致性等，具体内容如下：

a) 根据健康维度完整性，明确功能健康、性能健康、安全健康的综合评估要求；
b) 根据健康指标量化需求，明确健康得分计算与等级划分的科学性要求；
c) 根据评估结果的可比性，明确跨时间、跨版本的健康状态对比要求；
d) 应考虑评估的客观性，明确基于数据驱动的量化评估与主观偏差控制要求；
e) 应在评估中体现业务价值，明确健康状态与业务影响的关联度评估要求。

7.1.5 集成协同定性要求

对软件 PHM 系统集成协同能力提出要求，包括系统集成、数据协同、流程协同等，具体内容如下：

a) 根据现有工具链集成，明确与开发工具、测试工具、运维工具的无缝对接要求；
b) 根据数据共享需求，明确与监控系统、日志管理系统的数据融合要求；
c) 根据流程协同效率，明确与 CI/CD 流水线、运维流程的自动化集成要求；
d) 应考虑接口标准化，明确 API 设计的规范性与可扩展性要求；
e) 应在集成中保证数据安全，明确访问控制与数据保护要求。

7.1.6 退化趋势分析定性要求

对软件退化趋势分析能力提出要求，包括趋势识别准确性、退化速度评估、退化路径预测等，具体内容如下：

a) 根据软件老化特征，明确性能退化、可靠性下降的趋势识别要求；
b) 根据退化机理理解，明确资源累积消耗、代码质量恶化的分析要求；
c) 根据退化影响评估，明确用户体验、业务功能的退化程度量化要求；
d) 应考虑退化的非线性特征，明确突变式退化与渐进式退化的区分要求；

e) 应在趋势分析中考虑环境因素，明确负载变化、配置调整对退化趋势的影响要求。

7.1.7 生命周期健康管理定性要求

对软件全生命周期健康管理能力提出要求，包括阶段性管理、连续性管理、演化性管理等，具体内容如下：

a) 根据开发阶段差异，明确设计、编码、测试、部署各阶段的健康管理要求；
b) 根据运行周期特点，明确上线初期、稳定运行期、老化期的差异化管理要求；
c) 根据版本演进需求，明确版本升级、功能迭代中的健康状态传承要求；
d) 应考虑生命周期的完整性，明确从开发到退役的全过程健康跟踪要求；
e) 应在生命周期管理中体现前瞻性，明确未来演进的健康风险预防要求。

7.2 软件 PHM 定量指标

7.2.1 故障识别准确率

故障识别准确率表示 PHM 系统正确识别故障的比例，反映系统故障检测算法的准确性和可靠性。该指标是评估 PHM 系统核心能力的关键指标。计算方法见公式（1）：

$$Acc = \frac{x}{m} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（1）$$

式中：

Acc——故障识别准确率；

x——正确识别的故障数量，单位为个；

m——总故障数量，单位为个。

7.2.2 虚警率

虚警率表示 PHM 系统错误报告故障的频率，反映系统误报的控制水平。过高的虚警率会降低用户对系统的信任度。计算方法见公式（2）：

$$FAR = \frac{f}{n} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（2）$$

式中：

FAR——虚警率；

f——误判为故障的正常状态数量，单位为次；

n——系统监测到的正常状态总数量，单位为次。

7.2.3 漏警率

漏警率表示 PHM 系统未能检测到实际故障的比例，反映系统故障检测的完整性。该指标直接影响系统的安全性和可靠性。计算方法见公式（3）：

$$MAR = \frac{r}{n} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（3）$$

式中：

MAR——漏警率；

r——未被识别的故障数量，单位为个；

n——软件实际发生的总故障数量，单位为个。

7.2.4 预测提前时间

预测提前时间表示 PHM 系统在故障实际发生前多长时间给出预警，反映系统的预警时效性。该指标决定了维护决策的时间窗口。计算方法见公式（4）：

$$PRT = ft - fp \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（4）$$

式中：

PRT——预测提前时间，单位为秒；

ft——故障实际发生时间，单位为秒；

fp——故障预测报警时间，单位为秒。

7.2.5 健康指数计算精度

健康指数计算精度表示 PHM 系统计算的健康指数与专家评估或标准值的一致性程度。该指标反映健康状态评估算法的有效性。计算方法见公式（5）：

$$Ha = 1 - \frac{|Hc - Hr|}{Hr} \quad \cdots\cdots (5)$$

式中：

Ha——健康指数计算精度，取值范围为 [0,1]，越接近 1 表示精度越高；

Hc——PHM 系统计算得出的健康指数，取值范围为 [0,1]；

Hr——健康指数参考标准值，可通过专家评估、标准测试场景标定或行业基准数据确定，取值范围为 [0,1]。

7.2.6 故障修复成功率

故障修复成功率表示基于 PHM 系统诊断结果成功修复故障的比例，反映 PHM 系统诊断准确性对维护活动的指导效果。该指标是评估 PHM 系统实际价值的重要指标。计算方法见公式（6）：

$$sr = \frac{ff}{fd} \times 100\% \quad \cdots\cdots (6)$$

式中：

sr——故障修复成功率，取值范围为 [0%,100%]；

ff——基于 PHM 诊断结果首次修复成功的故障数量，单位为个；

fd——PHM 系统诊断识别出的故障总数，单位为个。

7.2.7 功能正确率

功能正确率是衡量软件系统执行预定功能准确性的关键指标，反映系统在实际运行中按照设计要求正确完成功能操作的能力。该指标通过对比系统实际执行结果与预期结果，评估功能实现的准确程度。计算方法见公式（7）：

$$facc = \frac{nc}{nt} \times 100\% \quad \cdots\cdots (7)$$

式中：

$facc$——功能正确率，取值范围为 [0%,100%]；

nc——在测试或运行周期内正确执行的功能数量，单位为个；

nt——软件系统设计的功能总数，单位为个。

7.2.8 功能覆盖率

功能覆盖率表示软件系统当前可用功能相对于设计规划功能的完整程度，是评估系统功能完备性的重要指标。该指标反映了系统功能实现的全面性和开发进度的完成情况。计算方法见公式（8）：

$$Cov = \frac{na}{nd} \times 100\% \quad \cdots\cdots (8)$$

式中：

Cov——功能覆盖率，取值范围为 [0%,100%]；

na——当前可正常使用的功能数量，单位为个；

nd——软件系统设计阶段规划的功能总数，单位为个。

7.2.9 接口调用成功率

接口调用成功率衡量系统各个接口在实际调用过程中成功响应的比例，是评估系统服务质量(QoS)和接口稳定性的核心指标。该指标直接反映了系统对外服务的可靠程度。计算方法见公式（9）：

$$SA = \frac{nas}{nat} \times 100\% \quad \cdots\cdots (9)$$

式中：

SA——接口调用成功率，取值范围为 [0%,100%]；

nas——接口调用成功的次数，单位为次；

nat——接口调用的总次数，单位为次。

7.2.10 平均响应时间

平均响应时间是指软件系统处理用户请求从接收到返回结果的平均时间长度，是衡量系统性能表现的基础指标。该指标直接影响用户体验和系统吞吐能力。计算方法见公式（10）：

$$ART = \frac{\sum_{i=1}^{n} T_i}{n} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（10）$$

式中：

ART——平均响应时间，单位为毫秒（ms）；

T_i——第 i 次请求的响应时间，单位为毫秒（ms）；

n——统计周期内的请求总数，单位为次。

7.2.11 CPU 利用率

CPU 利用率反映处理器资源在特定时间段内的使用百分比，是评估系统计算资源消耗和性能负载的关键指标，该指标帮助识别计算瓶颈和资源优化需求。计算方法见公式（11）：

$$U = \frac{Tb}{Tt} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（11）$$

式中：

U——CPU 利用率，取值范围为 [0%,100%]；

Tb——统计周期内 CPU 的忙碌时间，单位为秒（s）；

Tt——统计周期的总时长，单位为秒（s）。

7.2.12 系统可用性

系统可用性表示系统正常运行时间占总运行时间的比例，是衡量系统稳定性和服务连续性的核心指标。该指标直接影响用户服务体验和业务连续性。计算方法见公式（12）：

$$Av = \frac{Tu}{Tu+Td} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（12）$$

式中：

Av——系统可用性，取值范围为 [0%,100%]；

Tu——统计周期内系统正常运行的时间，单位为小时（h）；

Td——统计周期内系统因故障或维护导致的停机时间，单位为小时（h）。

7.2.13 故障频次

故障频次反映单位时间内系统发生故障的次数，是评估系统稳定性和可靠性水平的重要指标。该指标通过故障日志统计和异常事件记录进行监测。计算方法见公式（13）：

$$FR = \frac{nf}{T} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（13）$$

式中：

FR——故障频次，单位为次/小时（次/h）或次/天（次/天）；

nf——统计周期内系统发生的故障总次数，单位为次；

T——统计周期的时长，单位为小时（h）或天（天）。

7.2.14 MTBF（平均故障间隔时间）

MTBF 表示系统连续正常运行的平均时间长度，是衡量系统可靠性的经典指标。该指标反映了系统在正常运行期间的稳定程度。计算方法见公式（14）：

$$MTBF = \frac{Ta}{Tn} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（14）$$

式中：

$MTBF$——平均故障间隔时间，单位为小时（h）；

Ta——系统总运行时间，单位为小时（h）；

Tn——系统发生的故障总次数，单位为次。

7.2.15 MTTF（平均失效时间）

MTTF 表示系统从开始运行到首次发生失效的平均时间，反映系统在无维修情况下的可靠性水平。该指标主要用于评估不可修复系统或组件的可靠性表现，对于系统设计和可靠性评估具有重要意义。计算方法见公式（15）：

$$MTTF = \frac{\sum_{i=1}^{k} T_i}{k} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（15）$$

式中：

$MTTF$——平均失效时间，单位为小时（h）；

T_i——第 i 个样本从运行到首次失效的时间，单位为小时（h）；

k——测试或统计的样本总数，单位为个。

7.2.16 MTTR（平均修复时间）

MTTR 表示系统故障修复所需的平均时间，反映系统故障恢复的效率和维护能力。该指标对于评估系统维护水平和故障影响程度具有重要意义。计算方法见公式（16）：

$$MTTR = \frac{\sum_{i=1}^{n} T_i}{n} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（16）$$

式中：

$MTTR$——平均修复时间，单位为小时（h）或分钟（min）；

T_i——第 i 次故障的修复总时间，单位为小时（h）或分钟（min）；

n——故障修复的总次数，单位为次。

7.2.17 部署成功率

部署成功率衡量软件部署操作成功执行的比例，反映部署流程的稳定性和自动化水平。该指标对于评估持续集成和持续部署能力具有重要意义。计算方法见公式（17）：

$$SD = \frac{nc}{nt} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（17）$$

式中：

SD——部署成功率，取值范围为 [0%,100%]；

nc——成功完成的部署次数，单位为次；

nt——部署操作的总次数，单位为次。

7.2.18 配置变更成功率

配置变更成功率表示系统配置变更操作成功执行的比例，反映配置管理流程的有效性和系统配置的稳定性。该指标用于评估配置管理能力和变更风险控制水平。计算方法见公式（18）：

$$SC = \frac{nc}{nt} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（18）$$

式中：

SC——配置变更成功率，取值范围为 [0%,100%]；

nc——成功完成的配置变更次数，单位为次；

nt——配置变更操作的总次数，单位为次。

8 软件PHM技术方法

8.1 软件内技术

8.1.1 软件健康监测技术

采用嵌入式监测，外部工具和数据融合三类核心技术，实现对软件运行状态的全维度、实时感知，具体技术如下：

a) 运行时监测技术

运行时监测通过在软件执行过程中插入监测代码或使用外部监测工具，实时收集系统性能指标、资源使用情况和业务指标数据。主要技术包括：字节码插桩技术，在字节码层面插入监测代码；动态代理技术，通过代理模式拦截方法调用进行监测；AOP（面向切面编程）技术，在横切关注点插入监测逻辑。

b) 日志挖掘技术

通过分析系统日志、应用日志和业务日志，提取有价值的健康状态信息。技术要点包括：日志解析和结构化处理，异常模式识别和分类，时间序列日志分析，分布式日志聚合和关联分析。

c) 性能剖析技术

深入分析软件系统的性能瓶颈和资源消耗情况：CPU 性能剖析，识别高 CPU 消耗的代码段；内存剖析，检测内存泄漏和不合理的内存使用；I/O 性能分析，监测磁盘和网络 I/O 性能。

8.1.2 软件故障诊断技术

故障诊断技术旨在快速准确地识别软件系统中发生的故障类型、故障位置和故障原因，为故障修复提供指导，下面是故障诊断的通用技术：

a) 基于规则的诊断

通过预定义的规则集合和专家知识进行故障诊断方法：症状-故障映射规则，决策树诊断算法，专家系统方法。

b) 基于模型的诊断

建立软件系统的行为模型，通过模型推理进行故障诊断方法：有限状态机模型诊断，Petri 网模型诊断，依赖图模型诊断。

c) 基于统计的诊断

利用统计学方法分析系统行为偏差方法：异常检测算法，聚类分析方法，概率推理诊断。

d) 基于机器学习的诊断

采用机器学习算法自动学习故障模式方法：监督学习分类算法，无监督学习异常检测，深度学习故障识别。

8.2 软件外技术

8.2.1 软件故障预测技术

故障预测技术通过分析历史数据和当前状态，预测软件系统未来可能发生的故障、性能退化和健康状态变化。根据建模方法的不同，可分为数据驱动方法和模型驱动方法：

a) 数据驱动预测方法

数据驱动方法直接从大量运行数据中学习和提取知识，无需显式建立系统物理模型，是软件 PHM 的重要技术路径。

b) 机器学习预测方法

下面是常见的机器学习预测方法：

1) 监督学习预测：利用标记数据训练预测模型，包括决策树、随机森林、支持向量机（SVM）、神经网络等方法，用于故障分类预测和回归预测；
2) 无监督学习预测：从无标记数据中发现异常模式，包括 K-means 聚类、主成分分析（PCA）、自编码器异常检测等方法；
3) 深度学习预测：采用深度神经网络处理复杂的非线性关系，包括卷积神经网络（CNN）用于模式识别、循环神经网络（RNN）用于序列预测、长短期记忆网络（LSTM）用于时间序列预测；
4) 强化学习：通过与环境交互学习最优预测策略，适用于动态环境下的自适应预测。

c) 时间序列预测方法

下面是常见的时间序列预测方法：

1) 经典时间序列：ARIMA 模型预测、指数平滑方法、季节性分解预测等传统统计方法；
2) 现代时间序列：Prophet 算法、向量自回归（VAR）模型、状态空间模型等先进方法；
3) 深度时间序列：LSTM、GRU、Transformer 等深度学习方法在时间序列预测中的应用。

d) 统计分析预测方法

下面是常见的统计分析预测方法：

1) 回归分析：线性回归、多项式回归、支持向量回归、随机森林回归等方法建立预测模型；
2) 生存分析：Kaplan-Meier 估计、Cox 比例风险模型等方法预测系统剩余使用寿命；
3) 贝叶斯方法：贝叶斯网络、动态贝叶斯网络等概率推理方法。

e) 大数据分析技术

下面是常见的大数据分析技术：

1) 分布式数据处理：MapReduce、Spark 等框架处理大规模历史数据；
2) 流式数据处理：Kafka、Storm、Flink 等技术实现实时预测。

f) 模型驱动预测方法

模型驱动方法基于对软件系统的深入理解，建立数学模型来描述系统行为，并基于模型进行健康管理和预测。

g) 随机过程预测方法

下面是常见的随机过程预测模型方法：

1) 马尔可夫链模型：建立系统状态转移的概率模型，预测未来状态变化；
2) 半马尔可夫过程：考虑状态停留时间分布的随机过程模型；
3) 泊松过程：建模故障发生的随机时间间隔；
4) 布朗运动模型：描述系统性能的随机游走过程；
5) 隐马尔可夫模型：处理不可直接观测的内部状态变化。

h) 性能预测模型方法

下面是常见的性能预测方法：

1) 排队论模型：分析系统吞吐量、响应时间等性能指标；
2) Petri 网模型：建模并发系统的动态行为和性能特征；
3) 随机 Petri 网：结合概率因素的 Petri 网扩展；
4) 流体模型：连续化建模大规模系统的性能行为；
5) 马尔可夫决策过程：优化性能管理策略。

i) 退化预测方法

下面是常见的退化预测方法：

1) 威布尔分布模型：建模组件失效时间分布；
2) Gamma 过程：建模单调递增的退化过程；
3) Wiener 过程：建模连续时间的退化轨迹；
4) 分数布朗运动：考虑长程相关性的退化建模。

j) 混合预测建模方法

下面是常见的混合预测方法：

1) 灰盒建模：结合物理机理模型和数据驱动方法；
2) 数字孪生技术：构建软件系统的虚拟映射模型；
3) 多尺度建模：集成不同时空尺度的建模方法；

4) 物理-数据融合模型：融合领域知识和历史数据；
5) 混合神经网络模型：将物理约束嵌入神经网络；
6) 集成学习方法：组合多种预测模型提高预测精度。

8.2.2 软件故障定位与处理

采用“精准定位 + 影响评估 + 根因分析 + 自动化恢复”的全流程技术体系，在故障诊断基础上明确故障范围、追溯根本原因，并提供高效处理与恢复机制，具体技术如下：

a) 故障定位技术

故障定位技术通过多种技术手段精确确定故障发生的具体位置、影响范围和传播路径。主要技术包括：

1) 调用链路追踪技术，通过分布式追踪系统跟踪请求在系统中的完整执行路径；
2) 依赖关系分析，构建组件间的依赖图谱识别故障传播路径；
3) 异常传播分析，分析异常在系统中的扩散模式和影响边界。

b) 故障影响评估

故障影响评估技术量化分析故障对业务和系统的影响程度，为处理决策提供依据。评估内容包括：

1) 业务影响评估，分析故障对关键业务流程和用户体验的影响程度；
2) 系统影响评估，评估故障对系统性能、可用性和稳定性的影响；
3) 风险扩散评估，预测故障进一步扩散的可能性和潜在危害；
4) 恢复时间评估，估计不同处理策略下的故障恢复时间和资源成本。

c) 根因分析技术

根因分析技术深入挖掘故障发生的根本原因，避免仅处理表面症状。技术要点包括：

1) 多层次分析方法，从表象、直接原因到根本原因的逐层深入分析；
2) 关联性分析，识别故障与系统变更、环境因素、负载变化等的关联关系；
3) 时间序列分析，通过时间维度的数据分析发现故障的触发条件和演化过程。

8.3 模型选择策略

8.3.1 选择维度

需要从以下维度考虑：

a) 数据可用性

包括数据规模（海量/稀缺）、数据类型（结构化/半结构化/非结构化）、数据质量（完整性/准确性）、历史数据积累情况。

b) 实时性要求

包括故障预警响应时间（秒级/分钟级/小时级）、分析延迟容忍度。

c) 系统复杂度

包括软件架构（单体/分布式/微服务）、模块耦合度、运行环境（云原生/嵌入式/桌面端）。

d) 精度需求

包括故障预测准确率、健康指数计算精度的最低可接受阈值；

e) 资源约束

包括计算资源（CPU/内存）、部署成本、维护难度、技术团队能力。

8.3.2 场景化选择建议

下面是不同场景的选择策略：

a) 数据充足、实时性要求高、系统复杂度高场景

优先选择数据驱动模型如下：

1) 故障分类/回归预测：随机森林、支持向量机（SVM）、神经网络；
2) 时间序列预测（如性能退化趋势）：LSTM、Transformer、Prophet 算法；
3) 异常检测：自编码器、K-means 聚类、主成分分析（PCA）。

b) 数据稀缺、机理清晰、实时性要求中等

优先选择模型驱动方法：

1) 性能预测：排队论模型、Petri 网模型；
2) 退化预测：威布尔分布模型、Gamma 过程、Wiener 过程；
3) 故障概率预测：马尔可夫链、贝叶斯网络。

c) 数据中等、精度要求高、系统复杂度中等

优先选择混合建模方法：

1) 物理-数据融合模型：将软件运行机理（如内存泄漏规律）嵌入神经网络；
2) 灰盒建模：结合故障模式与影响分析和机器学习算法；
3) 集成学习：组合 2 种以上单一模型提升鲁棒性。

8.3.3 约束适配要求

下面是不同约束下的选择策略：

a) 嵌入式软件（资源有限）：优先选择轻量化模型（如朴素贝叶斯、简单线性回归），避免深度神经网络；
b) 关键业务软件（如金融交易系统）：优先选择可解释性强的模型（如决策树、规则库模型），拒绝“黑箱模型”；
c) 长期运行软件（如服务器运维软件）：优先选择支持增量训练的模型（如在线学习LSTM、增量SVM），适配数据动态更新需求。

9 软件PHM的数据管理

9.1 监测数据类型

9.1.1 结构化数据

结构化数据具有固定的数据模式和明确的字段定义，便于存储和查询分析。常见的结构化监测数据包括性能指标数据（CPU 利用率、内存使用量、响应时间）、系统配置参数、业务度量数据（交易量、用户数、成功率）、资源利用统计等。结构化数据是 PHM 量化分析和趋势预测的主要数据基础。

9.1.2 半结构化数据

半结构化数据具有一定的组织结构但不完全规范化，通常包含元数据信息。典型的半结构化监测数据包括系统日志文件、应用程序日志、JSON 格式的事件数据、XML 配置文件、数据库审计日志等。这类数据需要通过解析和提取技术转换为可分析的格式。

9.1.3 非结构化数据

非结构化数据缺乏预定义的数据模式，需要通过文本挖掘和模式识别技术进行处理。主要包括错误堆栈跟踪信息、用户反馈文本、代码注释文档、故障报告描述、运维操作记录等。非结构化数据虽然处理复杂，但往往包含丰富的故障诊断和经验知识信息。

9.2 数据采集要求

采样策略应满足数据质量优先、系统开销可控要求：

a) 需按指标重要性及变化频率，制定固定频率、自适应、事件驱动三类差异化采样规则；
b) 高频性能指标应采用滑动窗口平均技术减少数据量，异常事件必须完整捕获上下文信息；
c) 采样频率不得低于指标变化频率的 2 倍，确保数据有效性。

9.3 数据处理与存储

9.3.1 流式数据处理

流式数据处理需满足如下要求：

a) 应采用分布式流处理平台（如 Apache Kafka、Storm、Flink 等）构建流式处理架构，需支持在线数据清洗、转换及初步分析功能；
b) 应采用事件驱动模式，确保对系统状态变化的快速响应和异常模式的及时识别；
c) 需具备容错机制与状态管理能力，保障数据处理的连续性与一致性；
d) 关键指标处理延迟不得超过 10 秒，非关键指标不得超过 5 分钟。

9.3.2 分层存储架构

应按数据访问频率与重要性建立热、温、冷三级分层存储体系：

a) 热数据应存储于高性能存储设备，查询响应时间不得超过 500 毫秒；
b) 温数据应存储于标准存储设备，满足日常分析高频访问需求；
c) 冷数据应归档至低成本存储介质，保存期限不得少于 3 年；
d) 存储层应支持数据自动分层迁移，基于预设策略（如访问频率、存储时长）实现全生命周期自动化管理，迁移过程不得影响数据可用性。

9.4 数据质量控制

9.4.1 数据质量评估框架

数据质量评估框架可通过以下措施：

a) 应满足多维度的数据质量评估体系，包括准确性、完整性、一致性、及时性、有效性等关键维度；
b) 需制定每个维度的量化指标和评估方法，如数据缺失率、重复率、异常值比例等；
c) 需通过自动化的监控和数据采集(SCADA)工具持续评估数据质量状况，建立质量报告和告警机制。

9.4.2 异常检测与清洗

异常检测与清洗可通过以下措施：

a) 部署智能化的数据异常检测算法，能够识别各种类型的数据异常，包括数值异常、模式异常、时序异常等。
b) 基于统计方法、机器学习和规则引擎的组合检测数据异常。
c) 对于检测到的异常数据，采用适当的清洗策略，包括删除、修正、插值等方法，同时保留原始数据以备审计。

9.4.3 数据血缘与影响分析

数据血缘与影响分析可通过以下措施：

建立完整的数据血缘关系图，跟踪数据从源头到最终使用的完整链路，当发现数据质量问题时，能够快速定位影响范围和根本原因。

支持数据变更的影响分析，评估数据修改对下游分析和决策的潜在影响。

建立数据质量事件的处理流程和责任机制。

9.5 数据安全与隐私保护

9.5.1 访问控制与权限管理

访问控制与权限管理遵循以下要求：

a) 应实施基于角色的访问控制（RBAC）和基于属性的访问控制（ABAC）机制，确保只有授权用户才能访问相应的数据资源；
b) 建立细粒度的权限管理体系，支持数据级、字段级和操作级的权限控制；

c) 实现动态权限管理，根据用户角色变化和业务需求及时调整访问权限；
d) 部署统一的身份认证和单点登录系统，简化用户访问流程的同时确保安全性。

9.5.2 数据加密与脱敏

数据加密与脱敏遵循以下要求：

a) 应对敏感数据实施全生命周期加密保护，包括传输加密、存储加密和使用时加密；
b) 采用行业标准的加密算法和密钥管理机制，确保加密密钥的安全存储和定期轮换；
c) 对于包含个人信息或商业机密的数据，实施数据脱敏和匿名化处理，在保护隐私的前提下支持数据分析和挖掘；
d) 建立数据分类分级体系，根据数据敏感性采用不同强度的保护措施。

9.5.3 审计与合规管理

审计与合规管理遵循以下要求：

a) 应建立完整的数据访问审计机制，记录所有数据操作的详细日志，包括访问者、访问时间、操作内容等信息。
b) 实施实时的安全监控和异常行为检测，及时发现和响应潜在的安全威胁。
c) 确保数据处理活动符合相关法律法规要求。
d) 建立数据安全事件的应急响应流程，包括事件检测、影响评估、处置措施和事后总结。

10 软件PHM评价方法

10.1 评价准则

10.1.1 准确性准则

准确性是 PHM 系统最核心的评价准则，要求系统能够正确识别软件健康状态、准确诊断故障原因、可靠预测未来趋势。需建立分层的准确性评价标准，包括检测准确性、诊断准确性、预测准确性，且针对不同类型的软件系统和应用场景，制定差异化的准确性要求。

10.1.2 实时性准则

实时性准则要求 PHM 系统能够在规定时间内完成数据处理和分析任务，满足不同场景的时效性需求：

a) 对于关键故障检测，要求秒级响应；
b) 对于趋势分析，要求分钟级完成；
c) 对于深度分析，要求小时级交付结果。

建立实时性评级体系，根据响应时间将系统划分为优秀、良好、合格、不合格等级别。

10.1.3 可扩展性准则

可扩展性准则评估 PHM 系统适应业务增长和需求变化的能力。包括数据规模扩展能力、功能扩展能力、架构扩展能力。建立可扩展性测试方法和评估框架。

10.2 评价指标

10.2.1 技术性能指标

技术性能指标主要评估 PHM 系统在数据处理、算法执行和系统运行方面的表现：

a) 数据处理指标包括数据采集完整率、数据质量得分、数据处理实时性等；
b) 算法性能指标包括故障检测准确率、误报率、漏报率、预测精度、预测时间窗口等；
c) 系统运行指标包括系统可用性、响应时间、吞吐量、资源利用率等。

这些指标能够直接反映 PHM 系统的技术能力和运行效果。

10.2.2 业务价值指标

业务价值指标主要包括以下方面：

a) 业务价值指标衡量 PHM 系统对软件系统运维和管理带来的实际效益；

b) 主要包括故障平均修复时间(MTTR)、系统平均故障间隔时间(MTBF)、计划外停机时间减少率、运维成本降低比例、用户满意度提升等；
c) 这些指标直接关联到 PHM 系统的商业价值和投资回报率，是评估系统成功与否的重要标准。

10.3 评价流程

10.3.1 评价规划

实施过程遵循以下要求：

a) 制定详细的评价计划，明确评价目标、范围、方法和时间安排；
b) 识别评价的利益相关者，包括系统开发人员、运维人员、管理人员等，确定各方的评价需求和关注点。选择合适的评价方法和工具，建立评价环境和测试数据集；
c) 制定评价风险管理计划，识别可能影响评价结果的风险因素并制定应对措施。

10.3.2 数据收集与分析

实施过程遵循以下要求：

a) 应按照预定的评价指标体系收集相关数据，包括系统运行日志、性能监测数据、用户反馈信息等；
b) 建立数据质量检查机制，确保收集数据的准确性和完整性；
c) 采用适当的统计分析方法处理评价数据，包括描述性统计、趋势分析、对比分析等；
d) 建立数据可视化机制，以图表形式直观展示评价结果。

10.3.3 综合评估与报告

实施过程遵循以下要求：

a) 应基于收集的数据和分析结果，对照评价准则进行综合评估；
b) 采用加权评分方法计算各维度得分和总体评价结果；
c) 识别系统的优势和不足，分析问题的根本原因；
d) 编制详细的评价报告，包括评价方法、过程、结果和改进建议；
e) 建立评价结果的跟踪机制，监督改进措施的实施效果。

10.3.4 动态迭代

建立动态迭代机制以确保体系效能：

a) 实施分类频率控制：技术指标高频调参、业务指标中频调权、全维度体系年度重构；
b) 由版本更新、环境变更或指标不达标等异常情形触发迭代；
c) 经多方反馈、体系更新、模型校准、报告输出闭环流程，实现评价方法的持续调优。

11 软件PHM全生命周期实施过程

11.1 需求分析与论证阶段

11.1.1 健康管理需求分析

健康管理需求分析遵循以下要求：

a) 在系统论证阶段深入分析目标软件系统的健康管理需求，识别关键的可靠性和可用性指标；
b) 通过对类似系统的故障模式分析、运行环境评估和用户需求调研，确定 PHM 系统需要重点关注的健康状态参数和监测指标；
c) 建立需求优先级矩阵，区分必需的核心 PHM 功能和可选的扩展功能；
d) 分析 PHM 需求对系统架构、性能和成本的影响，为技术方案选择提供依据。

11.1.2 技术需求定义

技术需求分析遵循以下要求：

a) 基于业务需求转化为具体的技术需求规格，包括监测指标体系、数据采集频率、处理实时性要求、存储容量需求等；
b) 分析现有软件系统的技术架构和接口能力，评估 PHM 系统的集成可行性和技术约束；
c) 定义系统的非功能性需求，如可扩展性、可维护性、安全性等质量属性要求。

11.1.3 可行性评估

从技术可行性、经济可行性和组织可行性三个维度评估 PHM 实施的可行性。

a) 技术可行性评估包括数据采集能力、计算资源需求、算法适用性等方面；
b) 经济可行性分析实施成本与预期收益的对比；
c) 组织可行性考虑人员技能、管理流程、文化接受度等因素。

基于可行性分析结果制定实施策略和风险缓解措施。

11.2 系统设计与研制阶段

11.2.1 架构设计

PHM 系统应采用分层微服务架构，自下而上划分为数据采集、处理、分析、知识管理和用户接口五层，模块间通过标准 API 解耦。通过数据备份、服务冗余与故障转移机制保障高可用，并预留扩展接口以支持系统持续演进。

11.2.2 数据流设计

数据流设计需要做好下面几点：

a) 需设计端到端的数据流架构，从原始数据采集到最终的健康状态展示；
b) 需定义数据的标准格式和传输协议，确保不同组件间的数据兼容性；
c) 需设计实时数据流和批处理数据流的并行处理机制，满足不同时效性要求的分析任务；
d) 需建立数据质量检查点和异常处理机制，确保数据流的可靠性和完整性。

11.2.3 可测试性设计

在软件系统设计中融入可测试性(Design for Testability, DFT)设计理念，为 PHM 数据采集和状态监测提供便利：

a) 设计内置的测试点和监测接口，支持运行时状态信息的获取；
b) 实现软件组件的可观测性设计，包括详细的日志记录、性能计数器、状态变量暴露等；
c) 建立标准化的监测数据格式和接口协议，便于 PHM 系统的集成和扩展。

11.2.4 容错设计与冗余机制

设计多层次的容错机制，包括以下功能：

a) 实现关键组件的冗余设计，包括热备份、冷备份、负载均衡等不同的冗余策略；
b) 设计优雅降级机制，在部分功能故障时能够维持核心功能的正常运行；
c) 建立检查点和回滚机制，支持系统状态的保存和恢复。
d) 实现自适应的资源管理，根据系统负载和健康状态动态调整资源分配。

11.3 实施部署与集成阶段

11.3.1 迭代开发与集成

实施过程遵循以下要求：

a) 采用敏捷开发方法，将 PHM 系统功能分解为多个迭代周期，逐步构建和完善系统功能；
b) 建立持续集成和持续部署（CI/CD）流水线，自动化代码构建、测试和部署过程；
c) 实施严格的代码审查和质量检查，确保代码质量和系统稳定性；
d) 建立开发、测试和生产环境的一致性，减少环境差异导致的问题。

11.3.2 数据迁移与初始化

实施过程遵循以下要求：

a) 制定详细的数据迁移计划，将现有的监测数据和历史数据迁移到 PHM 系统中；
b) 建立数据验证和清洗流程，确保迁移数据的质量和完整性；
c) 配置初始的健康基线和阈值参数，基于历史数据分析和专家经验确定合理的初始值；
d) 建立数据备份和恢复机制，防止数据迁移过程中的意外损失。

11.3.3 系统集成与配置

实施过程遵循以下要求：

a) 实现 PHM 系统与目标软件系统的集成，配置数据采集接口和监测代理；
b) 部署分布式的 PHM 组件，配置负载均衡和服务发现机制；
c) 建立系统监控和日志收集机制，确保 PHM 系统自身的健康状态可见；
d) 配置告警规则和通知机制，建立多层次的异常响应体系。

11.4 测试验证与试验阶段

11.4.1 功能验证

设计全面的测试用例覆盖 PHM 系统的各项功能，包括数据采集、处理、分析和展示等核心功能。需执行单元测试、集成测试和系统测试，验证各个组件的正确性和整体系统的协调性，且模拟各种异常场景和边界条件，测试系统的容错能力和异常处理机制。

11.4.2 故障注入测试

采用故障注入技术模拟各种软件故障和异常情况，验证 PHM 系统的检测和诊断能力：

a) 设计多种故障注入方法，包括代码级注入、数据级注入、环境级注入等；
b) 建立故障模型库，涵盖常见的软件故障模式和失效机制；
c) 执行故障注入实验，评估 PHM 系统的故障覆盖率和检测精度；
d) 分析故障注入结果，优化检测算法和诊断规则。

11.4.3 性能确认

在模拟的生产环境中进行性能测试，验证系统在预期负载下的响应时间、吞吐量和资源利用率：

a) 执行压力测试和负载测试，确定系统的性能边界和瓶颈点；
b) 测试系统的扩展能力，验证在负载增长情况下的性能表现；
c) 确认系统满足服务级别协议（SLA）中定义的性能指标要求。

11.4.4 准确性评估

使用历史数据和已知案例验证 PHM 系统的健康评估、故障诊断和预测算法的准确性：

a) 建立基准测试数据集，定期评估算法性能的变化趋势；
b) 对比 PHM 系统的分析结果与实际发生的事件，评估预测的准确性和误报率；
c) 建立持续的算法优化和模型调整机制，不断提升系统的准确性。

11.5 运行维护阶段

11.5.1 持续健康监测

建立持续监测机制，实时跟踪软件系统的健康状态变化：

a) 实现多层次的监测体系，包括系统级、服务级、组件级和代码级的监测；
b) 建立动态的监测策略，根据系统负载和重要性调整监测频率和深度；
c) 实现智能化的异常检测，减少误报和漏报；
d) 建立监测数据的长期存储和趋势分析能力，支持历史回溯和模式发现。

11.5.2 预防性维护决策

基于 PHM 分析结果制定预防性维护计划，在故障发生前采取预防措施：

a) 建立维护决策支持系统，综合考虑故障风险、维护成本、业务影响等因素；
b) 实现维护资源的优化配置，提高维护效率和效果；

c) 建立维护知识库，积累维护经验和最佳实践；

d) 实现维护活动的自动化和智能化，减少人工干预和错误。

12 风险管理

12.1 PHM 相关风险识别

12.1.1 技术风险

技术风险是 PHM 系统面临的核心挑战，它紧密关联 PHM 监测的范围与深度、其采用的方法论自身的局限性，以及时间维度上的策略选择。具体技术风险见表 1。

表 1 技术风险类别

风险类型	风险点	描述
监测覆盖与完备性风险	PHM 对象不全面	PHM 系统可能无法覆盖被监测软件的所有模块、组件甚至相关联的外部依赖服务，导致健康管理存在盲区。
	监测点不全	即使覆盖了部分对象，也可能存在关键性能指标、日志事件或资源利用率等监测数据点缺失，无法形成全面的健康画像。
	未知风险	由于软件系统复杂性和动态性，总会存在现有 PHM 方法无法预见或捕获的异常模式、新型故障或极端情况，导致 PHM 失去预警能力。
PHM 方法与模型风险	PHM 方法自身的技术局限	所采用的健康评估算法或预测模型（如剩余寿命预测）可能存在固有缺陷，如模型泛化能力不足、对训练数据过度依赖、或无法有效处理非线性、非平稳的软件退化规律。
	模型准确性与鲁棒性问题	在面对软件运行环境变化、负载波动或出现新的故障模式时，模型可能产生大量误报或漏报，降低 PHM 系统的可信度。
时间维度策略风险	PHM 检测频率不合理	过长的检测周期可能导致无法及时捕获瞬时性异常或快速退化趋势，错过最佳预警时机；而过短的周期则可能产生大量冗余数据，增加处理负担。
	预测间隔长度	预测结果的输出间隔设置不合理，可能导致预测过于频繁而缺乏实际指导意义，或间隔过长而丧失预警价值。
PHM 系统架构与工程风险	系统扩展性不足	由于技术债务等原因，PHM 系统自身架构无法应对未来需要监测更多种类的软件系统，或处理更大规模监测数据的需求。
	故障自我诊断与恢复能力弱	PHM 系统自身发生组件故障或数据处理异常时，未能及时发现并进行自我修复或优雅降级，从而导致其丧失对被监测软件的健康管理能力。

12.1.2 应用风险

应用风险关注 PHM 系统对组织应用目标和运营效果的潜在负面影响，需考虑如下几种风险：

a) 需求风险包括用户需求理解偏差、需求变更频繁、期望管理不当等问题；

b) 依赖风险包括对特定供应商的过度依赖、技术锁定、第三方服务中断等外部依赖问题；

c) 业务风险的管理需要从战略高度统筹考虑，平衡技术投入与业务价值。

12.1.3 合规风险

合规风险涉及 PHM 系统在法律法规、行业标准、内部政策等方面的合规性问题，需考虑如下几种风险：

a) 法规风险包括数据保护法规违反、行业监管要求不满足、知识产权侵权等法律风险；
b) 标准风险包括国际标准不符合、行业最佳实践偏离、质量标准不达标等规范性风险；
c) 审计风险包括内部审计发现问题、外部监管检查不通过、合规证明材料不完整等风险；
d) 合规风险的管理需要建立持续的合规监控机制和定期的合规评估流程。

12.2 风险评估方法

12.2.1 定性风险评估

定性风险评估通过专家判断和经验分析，对风险进行等级划分和相对比较：

a) 建立风险评估矩阵，将风险按照发生概率和影响程度划分为不同的风险等级，如极高、高、中、低等类别；
b) 采用德尔菲法收集多位专家的独立判断，通过多轮调研和反馈达成风险评估共识；
c) 使用故障模式与影响分析(FMEA)方法，系统分析各种故障模式的发生原因、影响后果和检测方法；
d) 建立风险评估标准和指导原则，确保评估过程的一致性和可重复性。

12.2.2 定量风险评估

定量风险评估通过数学模型和统计方法，计算风险的期望损失和概率分布：

a) 采用蒙特卡罗仿真方法，模拟各种不确定因素的随机变化，评估风险的累积影响；
b) 使用决策树分析复杂的风险决策问题，考虑不同决策路径的概率和收益；
c) 建立风险指标体系，通过关键风险指标(KRI)的监控来预警潜在风险；
d) 采用敏感性分析识别对风险结果影响最大的关键因素，优化风险控制的重点。

12.2.3 动态风险评估

建立动态的风险评估机制，随着项目进展和环境变化持续更新风险评估结果：

a) 实现风险监控的自动化，通过实时数据收集和分析及时发现风险变化；
b) 建立风险预警机制，当风险指标超过预设阈值时自动触发预警和应对措施；
c) 定期执行风险评估更新，结合新的信息和经验调整风险评估结果；
d) 建立风险学习机制，从历史风险事件中总结经验教训，改进风险评估方法。

12.2.4 集成风险评估

考虑多种风险因素的相互作用和关联影响，进行综合的风险评估：

a) 建立风险关联性分析模型，识别风险之间的因果关系和传导路径；
b) 采用层次分析法(AHP)处理多目标的风险决策问题，权衡不同风险因素的相对重要性；
c) 建立风险组合分析框架，评估风险投资组合的整体风险水平；
d) 实现跨领域的风险集成评估，统筹考虑技术、运营、业务、合规等各类风险的综合影响。

12.3 风险控制措施

12.3.1 风险缓解措施

通过采取主动措施降低风险发生的概率或减少风险的影响程度，可通过技术缓解、流程缓解、人员缓解和资源缓解来实施：

a) 技术缓解包括实施冗余设计、容错机制、备份方案等技术手段；
b) 流程缓解包括建立标准操作程序、质量检查机制、审批流程等管理措施；
c) 人员缓解包括加强培训、建立专家团队、制定激励机制等人力资源措施；
d) 资源缓解包括增加预算储备、时间缓冲、设备冗余等资源保障措施。

12.3.2 风险接受与监控

风险接受与监控遵循以下要求：

a) 对于低风险或经济上不值得采取其他措施的风险，采用接受策略并建立监控机制；

b) 建立风险监控体系，持续跟踪风险状态变化，及时调整风险策略；

c) 制定应急响应计划，明确风险发生时的处理流程和责任分工。

参 考 文 献

[1] GB/T 25000.1-2021 系统与软件工程 系统与软件质量要求和评价(SQuaRE)第1部分：SQuaRE 指南.

[2] GB/T 25000.23-2019 系统与软件工程 系统与软件质量要求和评价(SQuaRE)第23部分：系统与软件质量产品质量测量.

[3] GB/T 8566-2022 系统与软件工程 软件生存周期过程.

[4] GB/T 25000.51-2016 系统与软件工程 系统与软件质量要求和评价(SQuaRE) 第51部分：就绪可用软件产品(RUSP)的质量要求和测试细则

[5] GB/T 37739-2019 信息技术 云计算平台即服务部署要求.

[6] T/CICC 35008-2025 复杂软件系统可靠性技术要求

[7] ISO 13372：2012 Condition monitoring and diagnostics of machines — Vocabulary.

[8] ISO 17359：2011 Condition monitoring and diagnostics of machines − General guidelines.

[9] NIST SP 800-218, Secure Software Development Framework V1.1 ： Recommendations for Mitigating the Risk of Software Vulnerabilities

[10] ISO 13379-1：2012 Condition monitoring and diagnostics of machines − Data interpretation and diagnostics techniques − Part 1： General guidelines.

[11] NIST SP 800-161 Rev. 1 Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations

[12] EN 50128：2011/A2：2020 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems

[13] ISO/IEC/IEEE 29119-1：2013. Software and systems engineering -- Software testing -- Part 1： Concepts and definitions.

[14] ISO/IEC 25010：2023 Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

[15] SAE JA6268_201710. HM Development Guide for Aerospace Applications.

[16] ISO/IEC 25040：2011. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Evaluation process.

[17] IEEE Std 828-2012. IEEE Standard for Configuration Management in Systems and Software Engineering.

[18] IEEE Std 2675-2021 DevOps： Building Reliable and Secure Systems Including Application Security, Configuration Management, Provisioning, and Deployment

[19] IEEE Std 1633-2016. IEEE Recommended Practice on Software Reliability.

[20] ISO/IEC/IEEE 12207：2017. Systems and software engineering -- Software life cycle processes.

[21] IEEE Std 2857-2021 Privacy Engineering for System Life Cycle Processes

[22] IEC 62506：2013. Methods for product accelerated testing.

[23] IEEE Std 1012-2016. IEEE Standard for System, Software, and Hardware Verification and Validation.

[24] ISO/IEC 20000-1：2018.Information technology -- Service management -- Part 1： Service management system requirements.

[25] ISO/IEC 25051：2023 Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Requirements for quality of Ready to Use Software Product (RUSP) and instructions for testing

[26] ISO 18435-2：2012 Industrial automation systems and integration – Diagnostics, capability assessment and maintenance applications integration – Part 2： Descriptions and definitions of application domain matrix elements

[27] ATA MSG-3：2013 MSG-3： Operator/Manufacturer Scheduled Maintenance Development, Volume 1 – Fixed Wing Aircraft

[28] US Army ADS-79C-HDBK：2012 Aeronautical Design Standard Handbook for Condition Based Maintenance Systems for US Army Aircraft

[29] ISO/IEC 25010：2011 Systems and software Quality Requirements and Evaluation (SQuaRE)

ICS 03.120.01
CCS L 05

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35004—2026

复杂智能系统信息安全性技术要求

Technical requirements for security of complex intelligent systems

2026-02-28 发布　　2026-02-28 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。

本文件起草参与单位：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）、中国航空研究院/智航院、中国科学院声学研究所、中国兵器工业软件工程与评测中心、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、张北、吴梦丹、郝程鹏、侯展意、刘波、司昌龙、王栓奇、王若、刘佳、盛珂、王树泰、程建新、刘磊、孙舒凡。

复杂智能系统信息安全性技术要求

1 范围

本文件规定了面向复杂智能系统的信息安全性技术要求，描述了复杂智能系统通用质量特性中有关信息安全性方面的相关指标以及测试方法。

本文件适用于面向复杂智能系统的信息安全性预估、设计和测试。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件;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 36344-2018 信息技术 数据质量评价指标

GB/T 44662-2024 健康管理 终端设备数据采集与传输协议

GB/T 45225-2025 人工智能 深度学习算法评估

GB/T 45087-2024 人工智能 服务器系统性能测试方法

GJB 3181A-2020 军用软件支持环境选用要求

3 术语和定义

下列术语适用于本文件：

3.1

数据泄露率 data leakage rate

每年因安全问题导致数据泄露的次数。

3.2

威胁检测响应时间 threat detection and response time

网络威胁或入侵行为发生后，从检测到响应的时间间隔。

3.3

入侵检测成功率 intrusion detection success rate

系统对网络异常或攻击行为的正确识别率。

3.4

入侵成功率 intrusion success rate

攻击者成功绕过防护机制并控制系统的频率。

3.5

权限错误配置率 permission misconfiguration rate

由于错误配置导致发生权限滥用的实例比率。

3.6

漏洞修复时间 vulnerability remediation time

从被识别为漏洞到漏洞修复完成的平均时间。

3.7

抗对抗样本攻击率 defense rate against adversarial attacks

系统对对抗样本的防御成功率。

3.8

内生安全 endogenous security

指人工智能系统因其自身的技术特点、内在缺陷或脆弱性（如算法、模型、数据、框架等方面）而产生的安全问题。

3.9

衍生安全　exogenous security

指因人工智能系统的应用或其行为（无论正确与否）对外部其他系统、环境或社会造成的安全性影响或风险。

3.10

对抗样本攻击　adversarial attack

一种闪避攻击（Evasion Attack），指攻击者通过向模型的正常输入数据中添加人眼难以察觉的微小扰动，从而欺骗模型做出错误分类或判断的攻击方法。

3.11

数据投毒攻击　data poisoning attack

一种在模型训练阶段发起的攻击，指攻击者通过向训练数据集中故意注入少量精心构造的恶意数据以破坏最终训练出的模型完整性，使其产生特定错误行为或植入后门。

3.12

模型窃取攻击　model stealing / extraction attack

指攻击者在仅能有限访问（如通过公开 API）模型的情况下，通过构造大量查询并观察其输出，来逆向复现或窃取一个与目标模型功能相似的副本模型的攻击行为。

3.13

模型后门攻击　model backdoor attack

一种特殊的投毒攻击，指攻击者在模型中植入一个隐藏的“触发器”（Trigger）。该模型在正常输入下表现正常，但一旦输入中包含特定触发器，就会执行攻击者预设的恶意任务。

3.14

模型鲁棒性　model robustness

模型在面对输入数据的微小扰动、噪声或分布变化时，仍能保持其预测结果稳定性和准确性的能力。

4　缩略语

下列缩略语用于本文件：

ABAC	基于属性的访问控制（Attribute-Based Access Control）
AES	高级加密标准（Advanced Encryption Standard）
ASID	地址空间标识符（Address Space Identifier）
AV	防病毒/反恶意软件（Anti-Virus）
C2PA	内容来源和真实性联盟（Coalition for Content Provenance and Authenticity）
CI/CD	持续集成/持续部署（Continuous Integration/Continuous Deployment）
CV	计算机视觉（Computer Vision）
EDR	端点检测与响应（Endpoint Detection and Response）
GAN	生成式对抗网络（Generative Adversarial Networks）
GPU	图形处理器（Graphics Processing Unit）
HIDS	基于主机的入侵检测系统（Host-based Intrusion Detection System）
HSM	硬件安全模块（Hardware Security Module）
LOF	局部异常因子（Local Outlier Factor）
MFA	多因素认证（Multi-Factor Authentication）

NPU 神经网络处理器（Neural Processing Unit）
PAM 特权访问管理（Privileged Access Management）
PGD 投影梯度下降（Projected Gradient Descent）
PoLP 最小权限原则（Principle of Least Privilege）
RLHF 基于反馈的强化学习（Reinforcement Learning from Human Feedback）
SDL 安全开发生命周期（Security Development Lifecycle）
SEV 安全加密虚拟化（Secure Encrypted Virtualization）
SGX 软件防护扩展（Software Guard Extensions）
SIEM 安全信息与事件管理（Security Information and Event Management）
SVD 奇异值分解（Singular Value Decomposition）
TCM 可信密码模块（Trusted Cryptography Module）
TLS 传输层安全协议（Transport Layer Security）

5 定量指标

5.1 数据加密完备性

数据加密完备性是衡量实现数据加密的完备程度的度量，指已按照规定采用加密保护的敏感数据项数量与规格说明中要求实施加密保护的敏感数据项总数之比，计算见公式（1）：

$$X_1 = \frac{A_1}{B_1} \quad \cdots\cdots (1)$$

式中：

X_1：数据加密完备性比例；

A_1：评审中已证实的已加密敏感数据项数量；

B_1：规格说明中要求加密的敏感数据项总数。

5.2 漏洞修复及时率

漏洞修复及时率是衡量安全漏洞被及时响应和修复能力的度量，指在规定时间窗口内成功修复的漏洞数量与指定时期内发现的需要修复的漏洞总数之比，计算见公式（2）：

$$X_2 = \frac{A_2}{B_2} \quad \cdots\cdots (2)$$

式中：

X_2：漏洞修复及时率；

A_2：在规定时间窗口内成功修复的漏洞数量；

B_2：指定时期内需要修复的漏洞总数。

5.3 威胁响应及时率

威胁响应及时率是衡量安全运营团队对安全告警响应效率的度量，指在规定时间窗口内得到初步遏制的安全告警数量与指定时期内产生的高置信度安全告警总数之比。计算见公式（3）：

$$X_3 = \frac{A_3}{B_3} \quad \cdots\cdots (3)$$

式中：

X_3：威胁响应及时率；

A_3：在规定时间窗口内得到初步遏制的安全告警数量；

B_3：指定时期内产生的高置信度安全告警总数。

5.4 抗入侵性

抗入侵性是衡量系统在模拟攻击下整体防御能力的度量，指在授权渗透测试中成功抵御的模拟攻击场景数量与模拟攻击场景总数之比。计算见公式（4）：

$$X_4 = \frac{A_4}{B_4} \quad \cdots\cdots (4)$$

式中：

X_4：抗入侵性比例；

A_4：授权渗透测试中成功抵御的模拟攻击场景数量；

B_4：模拟攻击场景总数。

5.5 模型抗攻击性

模型抗攻击性是衡量人工智能模型抵御对抗样本攻击能力的度量，指模型在面对对抗样本集测试时能够保持正确决策的样本数量与对抗样本集中的样本总数之比。计算见公式（5）：

$$X_5 = \frac{A_5}{B_5} \quad \cdots\cdots (5)$$

式中：

X_5：模型抗攻击性；

A_5：模型能够正确处理的对抗样本数量；

B_5：对抗样本集中的样本总数。

5.6 入侵检测准确性

入侵检测准确性是衡量安全检测系统识别攻击流量能力的度量，指安全检测系统在评测中正确识别的已知攻击实例数量与评测中使用的已知攻击实例总数之比。计算见公式（6）：

$$X_6 = \frac{A_6}{B_6} \quad \cdots\cdots (6)$$

式中：

X_6：入侵检测准确性；

A_6：系统正确识别的已知攻击实例数量；

B_6：评测中使用的已知攻击实例总数。

5.7 权限配置正确性

权限配置正确性是衡量系统遵循最小权限原则的符合程度的度量，指在审计中被证实符合最小权限原则的权限配置项数量与被审计的权限配置项总数之比。计算见公式（7）：

$$X_7 = \frac{A_7}{B_7} \quad \cdots\cdots (7)$$

式中：

X_7：权限配置正确性；

A_7：审计中证实符合最小权限原则的配置项数量；

B_7：被审计的权限配置项总数。

5.8 攻击成功率

攻击成功率是衡量对抗攻击算法“欺骗”模型能力的度量，指在评测中，成功使模型产生错误分类的对抗样本数量（A）与评测中使用的原始样本总数（B）之比。计算见公式（8）：

$$X_8 = \frac{A_8}{B_8} \quad \cdots\cdots (8)$$

式中：

X_8：攻击成功率；

A_8：模型产生错误分类的对抗样本数量；

B_8：评测中使用的原始样本总数。

5.9 鲁棒准确率

鲁棒准确率是衡量模型抵御对抗攻击能力的度量，指模型在遭受攻击后，仍能正确分类对抗样本的数量（A）与评测中使用的对抗样本总数（B）之比。计算见公式（9）：

$$X_9 = \frac{A_9}{B_9} \quad \cdots\cdots (9)$$

式中：

X_9：鲁棒准确率；

A_9：遭受攻击仍能正确分类对抗样本的数量；

B_9：评测中使用的对抗样本总数。

6 定性要求

6.1 身份鉴别与访问控制要求

本节要求确保所有访问系统资源的实体（用户、设备、服务）都经过严格的身份验证，并被授予完成其任务所必需的最小权限，要求如下：

a） 身份标识唯一性与管理：系统应为所有人类用户、系统服务账户、应用程序和设备分配全局唯一的身份标识。身份标识的生命周期（创建、分配、吊销）应被严格管理和审计；

b） 凭证安全策略：应实施并强制执行严格的密码策略，包括最小长度、复杂度要求、历史记录限制和定期轮换。所有静态凭证（如密码、API 密钥）在存储和传输时必须经过强哈希加盐或非对称加密处理，严禁明文存储；

c） 多因素认证(MFA)：对于所有远程访问、特权账户访问以及对关键数据或功能的访问请求，必须强制启用多因素认证；

d） 机器与服务身份认证：对于服务间通信、API 调用和自动化流程，应采用基于证书、API 密钥或服务网格等现代化的、非静态凭证的强认证机制，避免在代码或配置文件中硬编码凭证；

e） 访问控制策略实施：系统应具备强制执行访问控制策略的能力。建议优先采用基于属性的访问控制或基于角色的访问控制模型，以实现对资源访问的精细化管理；

f） 最小权限原则(PoLP)：所有账户和服务的权限授予严格遵循最小权限原则。默认访问策略应为“全部拒绝”，仅明确授予执行特定任务所必需的最小权限集。权限应具备时效性，避免永久授权；

g） 特权访问管理(PAM)：必须对管理员账户等特权身份进行严格管控。应采用即时权限提升机制，对特权会话进行实时监控和录像，并在每次使用后自动撤销权限或轮换凭证；

h） 定期权限审计：应建立定期的权限审查流程，由资源所有者或其主管对账户的权限配置进行审核和确认，及时移除不再需要或过度的权限。

6.2 数据安全与隐私保护要求

本节要求在数据的整个生命周期中保护其机密性、完整性和可用性，并严格遵守隐私保护法规。要求如下：

a） 数据分类分级：在数据创建之初，必须根据其敏感性和业务价值进行分类分级（如公开、内部、秘密、绝密）。不同的安全级别应对应不同的安全保护策略；

b） 传输中加密：系统所有内部和外部的网络通信，特别是承载敏感数据的通信，必须使用行业标准的强加密协议（如 TLS1.2 及以上版本）进行端到端加密，并禁用不安全的旧版协议和弱加密套件；

c） 静态加密：所有存储于持久化介质（如硬盘、数据库、对象存储）的敏感数据和个人隐私数据，必须采用强加密算法进行静态加密。应根据风险评估，采用包括卷加密、数据库加密、文件加密或应用层加密在内的多层次加密方案；

d） 密钥管理：用于加密的密钥必须通过安全的密钥管理系统或硬件安全模块进行全生命周期管理。密钥的生成、分发、轮换和销毁过程必须安全、可控且有审计记录。密钥访问权限应遵循最小权限原则；

e） 数据完整性保护：系统应采用强哈希算法或数字签名技术，对关键数据、配置文件、软件代码等生成校验值，以在传输和存储过程中检测任何未经授权的篡改；

f） 数据脱敏：在非生产环境（如开发、测试环境）中使用生产数据前，必须对敏感信息和个人隐私数据进行有效的脱敏处理，如数据屏蔽、截断、标记化等，以降低数据泄露风险；

g） 安全删除：当数据不再需要时，应有机制确保其被安全、彻底地删除，防止数据被恢复。应根据数据敏感度采用标准的数据擦除技术或密码学擦除方法；

h） 隐私设计：系统的设计和开发必须从一开始就融入隐私保护的考量，主动采取技术和组织措施保护个人信息。应提供用户友好的隐私设置选项，允许用户控制其个人数据；

i） 知情同意与透明度：系统在收集、使用或共享个人数据前，必须以清晰、易懂的方式告知用户，并获得其明确的知情同意。隐私政策应公开透明，详细说明数据处理的目的、范围和方式。

6.3 系统与架构安全要求

本节要求通过安全的设计、加固和隔离，构建一个具有纵深防御能力的、能够抵御现代网络攻击的系统架构，要求如下：

a） 安全设计原则：整体架构设计应遵循“安全默认”和“纵深防御”原则。应绘制数据流图并进行威胁建模，以识别潜在攻击面和设计缺陷，并据此设计缓解措施；

b） 网络分段与隔离：应根据业务功能和信任级别，将网络划分为不同的安全域（如生产区、测试区、DMZ 区），并使用防火墙、安全组或微隔离技术在各区域间实施严格的访问控制策略，限制横向移动；

c） 主机与端点安全：所有服务器、虚拟机和容器镜像都必须经过安全基线加固（如遵循CISBenchmarks），禁用不必要的服务和端口，并安装和配置防病毒/反恶意软件（AV）、主机入侵检测系统（HIDS）和端点检测与响应（EDR）等工具；

d） 应用安全与安全开发生命周期(SDL)：系统的开发过程必须遵循安全开发生命周期，如 SDL、DevSecOps 规范。应强制执行安全编码规范，并使用静态应用安全测试、动态应用安全测试和交互式应用安全测试工具对代码和运行中的应用进行持续检测；

e） 漏洞与补丁管理：必须建立正式的资产清单和漏洞管理流程。应使用自动化工具持续扫描系统和第三方组件的漏洞，根据风险等级确定修复的优先级和时间窗口，并对补丁安装过程进行有效管理和验证；

f） 供应链安全管理：必须对所有第三方供应商和开源组件进行安全风险评估。应维护一份软件物料清单，持续监控其依赖项的漏洞情报，并与供应商签订包含安全责任和事件通知要求的合同条款；

g） 配置管理与漂移检测：应建立并维护一套安全配置基线，并使用自动化工具对所有系统的配置进行持续监控，及时发现并告警任何未经授权的配置“漂移”，防止因配置错误导致的安全风险。

6.4 模型安全要求

本节专门针对智能系统核心的 AI 模型，提出抵御针对性攻击、确保其可靠性和可信性的要求，要求如下：

a） 训练数据安全与溯源：AI 模型的训练数据是核心资产，必须确保其在采集、标注、存储和使用过程中的机密性、完整性和来源可追溯性。应对数据提供者进行背景审查，并记录数据处理的每一步操作，以防御源头污染；

b） 模型抗投毒攻击：在模型训练阶段，应实施有效的数据清洗和异常检测机制，识别并剔除可能含有恶意样本或带有偏见的脏数据，以防止模型逻辑被污染；

c） 模型抗规避攻击(对抗性鲁棒性)：在模型推理阶段，系统应对输入数据进行严格的校验和净化，以识别和过滤对抗性样本。应通过对抗性训练、防御性蒸馏等技术手段，提升模型在面对恶意构造输入时的鲁棒性和稳定性；

d） 模型机密性与防窃取：核心 AI 模型应被视为商业机密进行保护。应通过严格的 API 访问控制、请求速率限制、输出结果加噪或模型水印等技术，防止攻击者通过大量查询来逆向工程或完整窃取模型（模型窃取）；

e） 模型可解释性与公平性审计：对于做出关键决策的 AI 模型（如金融、医疗领域），应具备一定的可解释性，使其决策逻辑能够被人类理解和审查。必须建立机制，定期对模型的输出进行公平性审计，检测并纠正其中可能存在的算法偏见；

f） 推理攻击防御：对于处理敏感数据的模型，应采取技术措施（如差分隐私）来防御推理攻击，防止攻击者通过模型的公开输出反推出训练集中的个人隐私信息（成员推理、属性推理等）。

6.5 可信计算环境要求

可信计算环境是 AI 系统安全体系的信任基础，其可靠性直接决定了上层所有安全措施的有效性。在多用户共享的 AI 计算中心场景下，必须从硬件及底层系统软件层面构建安全保障，确保数据和模型在存储、传输和处理过程中的机密性、完整性与可用性。要求如下：

a） 使用专用加密加速技术：利用芯片内置的高性能加密引擎，对 AI 模型、训练数据等核心资产在持久化存储和网络传输前进行加密处理。应配合使用独立的密钥管理基础设施对加密密钥进行全生命周期管理，确保只有经过身份认证和授权的计算任务才能访问密钥，从而在源头上保障数据所有权，并显著降低加密操作对性能的影响；

b） 使用机密计算技术：为缓解数据在内存处理过程中因明文存在而面临的泄露风险，应部署基于硬件可信执行环境的机密计算方案。通过将模型和数据加载到由 CPU 或 NPU 等硬件强制隔离的安全区域内进行运算，可有效阻止包括宿主机操作系统、虚拟机监视器在内的高权限实体对运行中数据和代码的非授权访问，实现“数据可用不可见”的运行时保护；

c） 使用可信互联技术：大规模分布式模型训练依赖于计算节点间的高速互联。为保障跨节点通信的安全性，应采用硬件级的安全互联协议。通过在底层硬件层面直接建立加密安全通道，实现 GPU 集群间“零开销”或极低开销的安全数据传输，高效防御中间人攻击和数据窃听，同时确保训练性能不受影响；

d） 进行平台完整性验证：为防止计算平台的固件、操作系统及驱动程序等基础软件被恶意篡改或植入后门，应建立基于硬件信任根的静态和动态完整性度量与验证机制。从系统引导开始，构建一条完整的信任链，逐层度量并验证关键软件组件未被篡改，确保整个计算平台运行在可信状态，有效抵御高级持续性威胁（APT）和供应链攻击。

6.6 安全运维要求

本节要求建立强大的监控、响应和恢复能力，并充分考虑人的因素在安全体系中的作用。要求如下：

a） 统一日志与安全监控：系统应将所有关键组件（包括服务器、网络设备、应用、数据库、AI 模型）的安全日志集中采集到安全信息与事件管理（SIEM）平台。日志内容必须包含足够的信息以支持事件溯源（遵循 5W1H 原则）；

b） 威胁检测与告警关联：SIEM 平台应配置有效的关联分析规则，能够基于海量日志和威胁情报，自动检测潜在的安全威胁和异常行为，并生成高质量、低误报的告警，通知安全响应团队；

c） 事件响应计划与演练：必须制定、维护并定期演练一套正式的事件响应计划。该计划应明确定义不同类型安全事件的响应流程、角色职责、沟通机制和处置步骤；

d） 高可用性与拒绝服务攻击防护：系统应采用负载均衡、冗余设计等架构，确保高可用性。应部署专业的抗拒绝服务攻击解决方案，以保障业务在面临大流量攻击时的连续性；

e） 备份与恢复：必须对关键数据和系统配置进行定期、自动化的备份，并存储于安全的异地位置。恢复流程必须被完整记录和定期测试，以确保在发生灾难性事件时，能够按预定目标快速恢复服务；

f） 持续性安全意识培训：必须为所有员工（特别是开发者和运维人员）提供持续的、与角色相关的安全意识培训和教育项目，内容涵盖密码安全、数据保护、风险识别等；

g） 安全变更管理：所有对生产环境的变更（包括代码发布、配置修改、系统升级）都必须遵循严格的变更管理流程，经过充分的测试和审批，并保留完整的变更记录，以防范因无序变更引入的安全风险。

6.7 预防数据投毒攻击的要求

6.7.1 数据污染鲁棒性

本节要求从算法和模型架构层面提升其对恶意数据的内在抵抗力。通过优化损失函数、改进聚合策略及采用集成学习方法，确保模型在训练数据可能受污染的情况下，仍能保持预期的性能表现。要求如下：

a） 使用鲁棒的损失函数：替换标准的损失函数（如交叉熵），使用对异常值不敏感的函数。例如，Huber 损失或修剪损失，这些函数会自动降低或忽略那些导致巨大损失值的样本（很可能是投毒样本）对梯度更新的贡献；

b） 采用鲁棒的聚合与优化算法：在每次迭代更新模型参数时，对来自各个数据样本的梯度进行筛选。例如，可以计算每个梯度的范数，并剔除掉那些范数过大或过小的梯度（即梯度裁剪或修剪），因为投毒样本通常会产生异常的梯度；

c） 利用集成学习：训练多个子模型，每个模型使用不同数据子集（例如 Bagging）。由于投毒样本只存在于部分子集中，因此只会污染少数子模型。在最终决策时，通过投票或平均的方式，未被污染的模型可以压制被污染模型带来的错误影响，从而得到一个鲁棒的最终结果。

6.7.2 训练数据可审查性

本节要求建立针对训练数据质量的深度检测与清洗机制。通过统计分析、影响评估及特征聚类等技术手段，主动识别并剔除潜在的投毒样本，从源头上保障模型训练数据的纯净度。要求如下：

a） 实施异常值检测：将训练数据在特征空间中进行表示，并使用统计学或机器学习方法寻找离群点。例如，可以使用聚类算法（如 DBSCAN），投毒样本可能会形成一个或多个孤立的小簇；或者计算每个样本与邻近样本的距离，距离过远的样本有可能是异常点；

b） 进行影响函数分析：用于估算“如果移除某个训练样本，模型会发生多大变化”。通过计算每个训练样本对一个干净验证集性能的影响分数，可以识别出那些对模型产生巨大负面影响的样本，这些样本极有可能是投毒数据；

c） 分析激活模式：运行整个训练集通过一个预训练或部分训练的模型，并提取每个样本在网络中间层的激活向量。理论上，属于同一类的正常样本会有相似的激活模式，而被植入后门或投毒的样本，即使标签相同，其激活模式也会与正常样本形成差异。通过对这些激活向量进行聚类，可以将可疑的样本簇分离出来。

6.8 预防模型窃取攻击的要求

6.8.1 低信息泄露

本节要求在模型服务接口层面实施最小化信息披露原则。通过控制 API 返回信息的粒度、精度及引入隐私保护技术，降低单次查询所暴露的模型内部特征，从而增加攻击者获取模型知识的难度。要求如下：

a） 设计模型的外部查询接口（API），使其在提供服务的同时，最大限度地减少关于模型内部结构和参数信息的泄露；

b） 限制返回信息的粒度：API 不应返回完整的概率分布向量（即每个类别的置信度得分）。最严格的做法是只返回最终的预测类别。这样，攻击者无法利用详细的概率信息通过知识蒸馏的方式来训练克隆模型；

c） 修改置信度输出：如果业务要求必须返回概率，可以通过温度缩放等方法来平滑输出的概率分布。提高温度（T>1）会让概率分布更趋于均匀，降低攻击者获取模型真实“信心”的精确度；

d） 应用差分隐私：在模型的输出（logits）或权重上添加经过精确计算的随机噪声。这提供了一种可证明的安全保证，即任何单次查询的输出都不会过多地暴露模型训练数据或参数的隐私信息，从而增加了窃取难度。

6.8.2 查询行为对抗策略

本节要求引入动态防御机制以混淆攻击者的视听。通过在输出中增加随机性和非确定性因素，以及根据威胁感知动态调整响应策略，显著提高攻击者构建有效替代模型的成本。要求如下：

a） 通过引入随机性或非确定性，使攻击者其难以构建准确的输入-输出对偶数据集；

b） 输出中引入随机性：对于分类任务，当置信度最高的两个类别的概率非常接近时，可以按概率随机返回其中一个作为结果；

c） 动态调整防御策略：结合下述的异常检测，一旦发现可疑的查询行为，可以临时性地增加输出噪声的强度或提高温度缩放的系数，动态增加窃取成本。

6.8.3 异常查询模式检测

本节要求构建针对API调用行为的实时监控与分析体系。通过识别高频、非典型分布及特定探测模式的查询行为，结合被动限制与主动溯源手段，及时发现并阻断模型窃取企图。要求如下：

a） 建立一个监控系统，像防火墙一样实时分析 API 的调用行为，识别出非正常用户的查询模式并采取行动；

b） 设置速率限制和配额：对单个 IP 或用户 ID 在单位时间内的查询次数进行限制。模型窃取需要大量查询，显著拖慢攻击进程；

c） 分析查询数据分布：监控并分析用户提交查询的数据特征。模型窃取攻击者经常会系统性地构造查询来探测模型的决策边界。如果发现大量查询集中在特征空间的某些非典型区域，或呈现出梯度下降式的探索模式，即可判定为异常；

d） 部署模型水印：在模型中嵌入“数字水印”，即让模型对某些特定的输入产生特定的输出。如果之后在某个疑似被窃取的模型上复现了这个水印，就获得了窃取的证据。

6.9 预防对抗样本攻击的要求

6.9.1 模型鲁棒性增强

本节要求通过强化训练过程和预处理手段来提升模型自身的防御壁垒。利用对抗训练、输入正则化及鲁棒优化算法，使模型能够学习到平滑且稳定的决策边界，从根本上抵御微小扰动的影响。要求如下：

a） 对抗训练：在训练集中人为生成并加入对抗样本（如利用 FGSM、PGD 攻击生成），使模型在训练阶段就暴露于可能的攻击输入中，从而学习到更加鲁棒的决策边界。该方法可以显著提高模型对常见攻击的抵抗力；

b） 输入正则化与预处理：在输入数据送入模型前增加随机化或去噪处理，如图像高斯模糊、JPEG 压缩、随机缩放与裁剪等。这样即使攻击者构造精细扰动，也可能被预处理环节削弱或破坏；

c） 鲁棒优化方法：在训练目标中显式融入鲁棒性约束，例如采用对抗风险最小化框架：最小化样本在最坏扰动情况下的损失；或在损失函数中加入对输入扰动敏感度的正则项，从而抑制模型的过度敏感性。

6.9.2 对抗样本检测能力

本节要求部署专门的检测器以在推理前识别潜在攻击。基于输入数据分布、梯度敏感度及深度特征的一致性分析，区分正常样本与恶意构造的对抗样本，实现对攻击流量的精准拦截。要求如下：

a） 输入分布一致性检测：建立正常数据分布模型（如高斯混合建模、核密度估计），在检测阶段对输入进行似然估计。若输入的概率密度显著低于正常训练数据，则可能是对抗样本；

b） 梯度敏感性分析：对输入样本轻微扰动，观察模型输出的变化幅度。由于对抗样本依赖输入空间的敏感方向，其梯度往往异常大。基于此可训练一个检测器来识别潜在对抗输入；

c） 特征空间分析与激活一致性：提取中间层特征或激活向量，比较输入样本与已知正常样本的相似度。对抗样本虽然在输入空间几乎不可区分，但在深层表示空间可能会表现出异常模式，可以借助聚类或异常检测方法识别。

6.9.3 推理阶段的防御机制

本节要求在模型实际运行环境中引入实时防护与认证措施。通过随机化平滑、形式化验证及动态降级策略，确保模型在面对未被检测出的对抗攻击时，仍能输出可靠或安全可控的预测结果。要求如下：

a） 随机化预测：在预测阶段对输入添加轻微的噪声（如高斯噪声），并多次运行模型，最终输出多数投票结果。这种方法可以在理论上为模型提供认证鲁棒性保证，有效抵御小扰动攻击；

b） 认证防御：采用基于凸松弛、区间约束传播、线性规划等方法，对模型的决策边界提供形式化的鲁棒性证明，即保证在某个范数范围内的扰动不会改变预测结果；

c） 动态防御响应：在推理过程中实时监控输入信号。如果检测到可能的对抗扰动，可以触发降级处理策略，例如提高预处理强度、增加随机化、或者请求人工复核。

7 智能系统信息安全性支撑技术及方法

7.1 数据投毒的检测技术

7.1.1 异常值检测法

针对训练数据集中，在特征空间或数据分布上偏离良性样本群体的投毒数据，通过度量单个数据点与整体数据分布的偏差来识别潜在的投毒样本。具体实现应遵循以下技术路径：

a） 基于距离的检测：计算每个数据点与其 K 个最近邻样本的平均距离或加权距离。设定距离阈值，超过阈值的数据点应被标记为异常；

b） 基于密度的检测：采用局部异常因子（LOF）算法，计算每个数据点的局部密度与其邻域内其他数据点局部密度的比值，以识别位于低密度区域的异常点。或采用 DBSCAN 等密度聚类算法，将无法归入任何高密度簇的数据点识别为噪声点（即异常点）；

c） 基于集成学习的检测：利用孤立森林等模型，通过构建多棵随机树来隔离数据点。异常点因其稀有性，通常具有较短的平均隔离路径长度，据此计算其异常得分。

7.1.2 数据影响与价值评估法

针对对模型参数和决策边界产生关键影响，降低模型泛化性能或植入后门的投毒数据，量化每个训练样本对最终模型性能的贡献度或影响度来识别恶意样本。具体实现应遵循以下技术路径：

a） 影响函数分析：采用影响函数来近似计算移除单个训练样本后，模型在验证集上损失函数的变化量。将导致验证集损失显著降低（即具有高度负面影响）的样本标记为潜在的投毒数据；

b） 数据价值评估：采用基于合作博弈论的 Shapley 值或其近似计算方法，评估每个数据点对模型在特定性能指标上（如准确率）的边际贡献。将贡献度为负值或显著低于平均水平的数据点识别为低质量或恶意数据。

7.1.3 模型行为分析法

针对在输入特征上难以区分，在模型训练或推理过程中引发异常内部行为的投毒数据，通过监控和分析模型在处理不同数据时的内部状态来识别异常。具体实现应遵循以下技术路径：

a） 激活模式聚类：在模型训练完成后，提取所有训练样本在指定中间层（如全连接层或卷积层）的神经元激活向量。对这些激活向量进行聚类分析，将在激活空间中偏离其所属类别正常簇中心的样本识别为异常；

b） 谱特征分析：计算由训练数据在模型某层激活值构成的协方差矩阵。通过对该矩阵进行奇异值分解（SVD），分析其谱特征。由特定投毒数据（尤其是后门攻击样本）引发的异常谱特征可用于识别相关样本。

7.2 对抗样本的检测技术

7.2.1 输入变换一致性检测法

针对通过添加微小、高频扰动而生成，且对特定变换操作敏感的对抗样本，通过对输入样本施加一个或多个预定义的变换，比较变换前后模型输出的一致性来判断其是否为对抗样本。具体实现应遵循以下技术路径：

a） 特征压缩：对输入数据执行降维操作，如降低图像的色彩位深或应用空间平滑滤波器。比较原始输入与压缩后输入的模型预测结果，若二者差异超过预设阈值，则判定为对抗样本；

b） 随机化变换：对输入样本进行小幅度的随机变换，如随机缩放或旋转。对同一输入进行多次随机变换并获取预测结果分布。若该分布的方差或熵值超过阈值，则判定为对抗样本。

7.2.2 模型不确定性度量法

针对将模型推向其决策边界附近，从而引发模型高预测不确定性的对抗样本。通过量化模型对输入样本预测的不确定性程度来进行检测。具体实现应遵循以下技术路径：

a） 蒙特卡洛 Dropout：在模型推理阶段保持 Dropout 层激活，并对同一输入进行多次前向传播，得到一个预测结果集。计算该结果集的统计指标（如预测类别的方差或熵）。若不确定性指标超过预设阈值，则判定为对抗样本；

b） 贝叶斯神经网络：采用贝叶斯神经网络模型进行推理。该模型直接输出预测分布，从中提取认知不确定性的度量。若认知不确定性显著高于基线水平，则判定为对抗样本。

7.3 后门攻击的检测技术

7.3.1 神经元分析法

针对通过控制模型中少数特定神经元的行为来实现的后门攻击。该方法通过检查和分析模型内部神经元的激活状态与功能来发现后门。具体实现应遵循以下技术路径：

a） 激活状态分析：在大量良性输入下，识别出持续处于低激活或“休眠”状态的神经元。随后，通过优化方法寻找能强烈激活这些休眠神经元的输入模式，该模式即为潜在的后门触发器；

b） 剪枝扫描：对模型的神经元进行渐进式剪枝。如果在剪枝过程中，模型在某个特定目标类别上的性能发生远超其他类别的断崖式下降，则表明被剪除的神经元与此后门功能高度相关。

7.3.2 扰动鲁棒性检测法

针对被后门触发器激活后，模型预测结果对额外扰动表现出异常鲁棒性的样本。该方法通过在输入端叠加扰动并观察模型输出的稳定性来检测后门。具体实现应遵循以下技术路径：

a） 输入叠加与熵分析：将待测输入与一系列来自不同类别的良性图像进行叠加混合。对于每个混合后的输入，计算模型输出的预测熵；

b） 若一个输入在经过多种叠加扰动后，其预测熵始终维持在极低水平，则判定该输入已激活后门。

7.4 伪造与生成式内容的检测技术

7.4.1 物理与生理不一致性检测法

针对未能完全模拟物理规律或人类生理特征的伪造内容，通过检测伪造内容与先验知识的矛盾之处进行识别。具体实现应遵循以下技术路径：

a） 生理信号分析：提取并分析视频中人脸区域的生物信号，检测是否存在符合正常生理范围的周期性微弱颜色变化（遥测光电容积描记法信号），或分析其眨眼频率和模式是否符合人类统计规律；

b） 物理一致性分析：分析图像或视频中，主体对象与环境之间的物理交互是否合理，如检测人脸或物体的光照方向、阴影形状、高光反射是否与场景光源一致。

7.4.2 生成模型指纹检测法

针对因特定生成模型（如 GAN、扩散模型）的体系结构或训练过程而在生成内容中留下的系统性、可识别的数字痕迹，通过训练分类器来识别这些生成伪影。具体实现应遵循以下技术路径：

a） 频域分析：将图像内容变换到频域，分析其频谱图，识别由生成模型（特别是基于上采样的模型）引入的特定周期性伪影；

b） 通用伪影分类器：使用包含大量真实图像和多种生成模型伪造图像的数据集，训练一个深度神经网络分类器，使其学习区分真实内容和生成内容的通用伪影特征。

7.4.3 内容溯源与认证法

通过在内容生命周期源头引入认证机制，为内容的真实性提供可验证的依据，不依赖于对内容本身的分析，而是验证其附带的来源声明。具体实现应遵循以下技术路径：

a） 数字水印验证：检查内容中是否嵌入了由可信来源添加的、抗攻击的不可见数字水印，通过解码水印信息来验证内容的来源和完整性；

b） 内容来源标准符合性检查：遵循如C2PA等行业标准，检查数字内容是否附带有加密签名的元数据清单，通过验证该清单的签名和履历来确认内容的真实性。

7.5 部署环境的安全加固

7.5.1 治理与架构

明确治理责任：建立清晰的治理结构，明确AI系统的网络安全负责人与整个组织的网络安全负责人保持一致。识别所有利益相关方角色、职责和问责机制，要求如下：

a） 获取与利用威胁模型：要求 AI 系统的主开发商提供其系统的威胁模型。部署团队应利用此威胁模型来指导安全措施的实施、评估潜在威胁并规划缓解策略；

b） 安全边界保护：在IT环境与AI系统之间建立并强化安全边界（如防火墙、网关），并审查边界防护的盲点；

c） 零信任架构：秉持零信任的原则设计部署架构，假定失陷已然发生，对所有访问请求进行严格的身份验证和授权；

d） 数据源保护：识别并保护所有用于模型训练或微调的专有数据源。维护一个受信任和有效的外部数据源目录，以防范数据投毒或后门攻击。

7.5.2 配置硬化

本节要求通过实施技术管控措施来增强部署环境的防御纵深。采取环境隔离、补丁管理、加密保护以及强认证机制等多重手段，系统性地减少攻击面并提升数据安全性。要求如下：

a） 环境隔离：将运行机器学习模型的环境进行沙箱化处理，例如将其置于经过安全强化的容器或虚拟机中，以限制其潜在影响范围；

b） 持续的漏洞与补丁管理：密切关注硬件供应商（GPU、CPU 等）和软件的安全通告，及时应用安全补丁和更新，以最小化已知漏洞被利用的风险；

c） 静态数据加密：对所有敏感的 AI 信息（如模型权重、输出日志、训练数据）在存储时进行加密；

d） 密钥安全存储：将用于解密的密钥存储在硬件安全模块（HSM）中，实现密钥与加密数据的物理分离和安全管理；

e） 传输中数据加密：采用最新版本的传输层安全协议（TLS）来加密所有传输中的数据；

f） 强化认证机制：全面实施抗网络钓鱼的多因素认证（MFA），特别是对访问关键信息、服务和管理员账户的场景。

7.6 密码与加密系统的安全检测

7.6.1 自动化密码逻辑漏洞检测

本节要求引入智能化工具以提升密码系统审计的效率与准确性。结合自动化分析方法与大语言模型辅助技术，深入挖掘并修复加密实施过程中的逻辑缺陷与潜在风险。要求如下：

a） 使用自动化密码逻辑方法，核心通过知识库指导，借助大语言模型加以辅助，用于检测加密算法中的逻辑缺陷和编码错误；

b） 使用检测系统结合思维链（CoT）提示和检索增强生成（RAG）技术，借助数据量充足且完善的密码知识库指导，识别出密码算法中存在的数据安全问题。

7.6.2 轻量化密码认证技术

本节要求针对特定场景部署高效且安全的身份验证方案。通过融合轻量化算法与全生命周期的认证管理策略，确保设备接入与运行过程中的可信度与安全性。要求如下：

a） 使用轻量化密码认证技术结合轻量化密码算法、身份认证协议、证书管理机制与可信根部署策略；

b） 提出设备接入前认证、运行中身份保持、远程重认证、密钥更新等核心技术要求。

7.7 隐私保护与数据的安全检测

7.7.1 成员推理攻击检测

本节要求建立针对模型记忆泄漏的检测机制。利用统计分析方法评估模型生成内容与训练数据的相似性，从而识别模型是否过度拟合隐私信息，保障训练数据的机密性。要求如下：

a） 使用 N-gram 覆盖方法，通过分析 AI 模型生成的文本内容就能检测其是否记住了训练数据，无需访问模型内部信息；

b） 通过使用 N-gram 覆盖"术来比较生成内容与原始文档的相似程度，设计覆盖度、创新度指数和最长公共子串三种计算方法。

7.7.2 数据生命周期安全检测

本节要求实施全流程的数据安全管控策略。从分类分级到动态监测，通过技术手段与管理体系的结合，确保数据在采集、存储、处理及销毁等各个环节的安全性。要求如下：

a） 在数据处理全周期采取加密传输、访问控制等手段保障敏感信息不被非法获取或篡改；

b） 建立完善的数据分类分级管理体系，根据不同类型数据的重要性设置相应的防护策略；

c） 通过数据流图谱构建、访问日志分析和异常行为检测等技术，确保数据在整个生命周期中得到适当保护。

7.8 AI威胁检测技术

7.8.1 大语言模型特有攻击检测

针对大语言模型的特点，在检测针对该模型的攻击中，要求如下：

a） 提示注入防御：检测输入中是否含有恶意指令，如“忽略之前命令”，包含提示词中的直接指令或附带隐藏指令的文件；

b） 越狱攻击识别：监控输出是否突破安全护栏，如暴力破解 RLHF 限制；

c） 训练数据提取：防御通过反复查询重建训练数据，如“抽词攻击”。

7.8.2 多模态系统攻击检测

在同时处理多种数据模态的系统中，传统的单模态安全检测方法难以应对跨模态安全威胁。常见的检测方法有：

a） 模态一致性验证：对比不同模态解析结果的逻辑一致性，如图文匹配度；

b） 跨模态特征分析：检测特征空间中的异常关联，如 CLIP 模型的图文嵌入距离异常。

7.9 模型安全设计技术

7.9.1 鲁棒性增强技术

本节要求在模型设计与训练阶段主动注入防御属性。通过对抗训练、数学认证及反逆向工程技术，提升模型在面对恶意干扰时的稳定性与抗攻击能力。要求如下：

a） 对抗训练：针对 PGD 等攻击样本注入训练；

b） 随机平滑：使用 CertifiedRobustness 保证方式，增强鲁棒性；

c） 梯度掩码：采取梯度掩码防御模型逆向工程模式。

7.9.2 全水印技术

安全水印技术是智能系统知识产权保护和防伪溯源的核心手段，通过在模型参数、输出结果或数据中嵌入隐蔽标识，实现所有权验证、盗版追踪和篡改检测，关键技术如下：

a） 白盒水印：模型参数中植入特定模式，适用于模型提供方完全可控的场景；

b） 黑盒水印：触发集验证所有权，仅通过模型输入输出交互验证水印，适用于云 API 服务等无法接触模型内部的场景。

8 基于模型架构类型的信息安全性要求

8.1 基于大模型的智能系统信息安全性要求

8.1.1 外部信息安全性要求

外部信息安全性要求如下：

a） 大模型训练集与验证集应采用多源校验与完整性检测，防止数据投毒；

b） 应支持跨数据中心的传输加密（TLS1.3 及以上），保障参数同步的机密性与完整性；

c） 宜具备抗模型窃取攻击措施，如 API 限制、访问审计与输出加噪。

8.1.2 使用信息安全性要求

使用信息安全性要求如下：

a） 模型推理接口应强制启用身份认证与访问控制，防止未授权调用；

b） 运行时应提供对抗样本检测与异常输入屏蔽机制；

c） 宜部署模型水印或指纹技术，用于检测与追溯模型泄露与窃取行为。

8.1.3 风险控制要求

风险控制要求如下：

a） 部署过程中应定期执行渗透测试，发现潜在攻击面；

b） 宜建立模型快照与版本库，保证在安全事件后可快速恢复。

8.2 基于联邦学习的智能系统信息安全性要求

8.2.1 外部信息安全性要求

外部信息安全性要求如下：

a） 终端与服务器之间的参数/梯度传输必须采用端到端加密；

b） 联邦聚合过程应使用差分隐私或安全多方计算，防止单点窃取敏感信息；

c） 宜支持节点身份认证与可信执行环境，防止恶意节点参与。

8.2.2 使用信息安全性要求

使用信息安全性要求如下：

a） 客户端应具备安全自检机制，确保其满足系统安全规范后方可参与训练；

b） 应通过异常梯度监测与鲁棒聚合防御，降低投毒攻击风险；

c） 宜保留完整的联邦训练日志，用于安全审计和追溯。

8.3 基于深度神经网络的智能系统信息安全性要求

8.3.1 外部信息安全性要求

外部信息安全性要求如下：

a） 网络训练数据通道必须保证机密性与防篡改性；

b） 应对第三方依赖库进行安全验证，防范供应链攻击；

c） 宜提供动态模型参数完整性校验机制。

8.3.2 使用信息安全性要求

使用信息安全性要求如下：

a） 应启用输入检测，识别异常输入与对抗性扰动；
b） 在云端部署时应限制 API 调用频度，防止模型窃取与爆破；
c） 宜支持推理阶段的差分隐私防御，降低训练数据被推理攻击推断的风险。

8.4 基于迁移学习的智能系统信息安全性要求

8.4.1 外部信息安全性要求

外部信息安全性要求如下：
a） 源模型应具备可信证明，避免使用来源不明或篡改的预训练模型；
b） 应验证迁移数据集的合法性，防止预训练时植入后门。

8.4.2 使用信息安全性要求

使用信息安全性要求如下：
a） 应具备对新任务数据的鲁棒检测与清洗机制；
b） 在迁移与微调阶段，应采用参数加密存储与防篡改措施。

8.5 基于Transformer模型的智能系统信息安全性要求

8.5.1 外部信息安全性要求

外部信息安全性要求如下：
a） 训练语料库应进行安全筛查，防止数据泄密或恶意样本混入；
b） 应具备大规模训练数据的访问权限隔离机制。

8.5.2 使用信息安全性要求

使用信息安全性要求如下：
a） 模型推理输出应具备内容安全过滤，防止生成攻击性、违规或敏感信息；
b） 宜部署异常查询检测，及时阻断模型窃取行为。

8.6 基于扩散模型的智能系统信息安全性要求

8.6.1 外部信息安全性要求

外部信息安全性要求如下：
a） 应验证生成模型训练集来源，防止引入恶意伪造数据；
b） 模型参数应具备存储加密与访问控制机制。

8.6.2 使用信息安全性要求

使用信息安全性要求如下：
a） 应提供合规性检测，防止生成违法违规或侵权内容；
b） 宜启用内容溯源与数字水印，以追踪生成结果来源。

8.7 基于强化学习的智能系统信息安全性要求

8.7.1 外部信息安全性要求

外部信息安全性要求如下：
a） 在与真实环境交互时，应隔离训练与实际控制信号，防止意外安全事件；
b） 宜部署环境接口验证机制，避免恶意控制命令注入。

8.7.2 使用信息安全性要求

使用信息安全性要求如下：
a） 在训练策略更新时，应具备安全回退机制，防止未经验证的策略直接上线；
b） 宜支持对奖励函数的异常值检测，避免被利用进行策略操纵。

8.8 基于多智能体系统的智能系统信息安全性要求

8.8.1 外部信息安全性要求

外部信息安全性要求如下：

a） 智能体间的通信应采用加密信道，防止窃听与篡改；

b） 宜具备容错机制，应对单一智能体恶意或失陷带来的扩散风险。

8.8.2 使用信息安全性要求

使用信息安全性要求如下：

a） 协同决策应具备一致性保证，防止恶意节点投毒影响整体决策；

b） 宜在任务过程中建立异常检测与隔离策略，及时剥离失陷智能体。

9 基于功能类型的信息安全性要求

9.1 感知功能

感知功能信息安全性要求如下：

a） 应启用传感器数据完整性校验，防止数据篡改；

b） 宜对输入数据执行异常检测，识别恶意干扰信号（如对抗样本、激光干扰）；

c） 应提供冗余验证机制，降低单一传感器被攻击导致的安全风险。

9.2 认知功能

认知功能信息安全性要求如下：

a） 应在多模态融合时执行数据一致性与可信性检测，防止恶意注入假信息；

b） 推理时输出应具备可解释性与安全审计能力，防止偏见或攻击操纵结果；

c） 宜采用知识库更新验证，防止数据污染或恶意注入错误规则。

9.3 决策功能

决策功能信息安全性要求如下：

a） 在多执行单元中，应防止假数据或异常输入影响集中调度；

b） 决策结果应记录和追溯，提供安全审计证据；

c） 宜具备安全降级和冗余机制，保障在部分节点失陷后仍可维持基本功能。

9.4 控制功能

控制功能信息安全性要求如下：

a） 控制指令传输应全程加密，防止劫持；

b） 控制环路应内置异常信号过滤，避免误操作导致设备损坏；

c） 宜具备实时入侵检测与自动阻断机制。

9.5 执行机构

执行机构信息安全性要求如下：

a） 应对执行机构的控制信号执行数字签名与鉴权，防止伪造控制指令；

b） 宜在关键执行环节启用“安全停机”机制，防止遭受恶意操控；

c） 应建立执行机构健康监测与告警机制，防止长时间异常运行引发灾害。

10 智能系统信息安全性活动全生命周期过程

10.1 需求阶段

在需求阶段，即项目的初始可行性与需求确认环节，应将信息安全作为智能系统一项关键的非功能性需求予以明确。其核心活动包括：

a） 安全需求分析：识别智能系统的关键信息资产与潜在威胁源，结合行业标准和合规性要求，分析可能的攻击面，如数据投毒、模型窃取、未经授权访问等；

b） 安全目标设定：基于风险评估，设定量化和定性的安全性目标；

c） 安全策略与架构初步设计：在顶层架构设计中，预先规划防御层级与安全机制，例如零信任访问控制、加密通道、日志审计、模型安全加固等，使安全性要求成为与功能需求同等重要的约束。

10.2 设计阶段

设计阶段是将安全性规划转化为具体架构与实现的过程，应落实以下原则：

a） 安全架构集成：在系统架构中引入纵深防御机制，包括网络分区隔离、访问控制引擎、可信计算环境和硬件安全模块；

b） 模块化与最小暴露设计：通过功能解耦降低攻击面，避免敏感接口直接暴露，采用最小权限原则确保组件调用的安全；

c） 安全开发生命周期（SDL）：在编码过程中全面采用安全编码规范（如规避常见漏洞 OWASP Top 10），集成静态代码分析、依赖库漏洞扫描和自动化安全测试工具；

d） 模型与数据保护：对模型训练数据实施防投毒清洗，采用差分隐私或加噪策略保护敏感数据，并在模型参数存储中启用文件签名与加密。

10.3 训练与测试阶段

训练与测试阶段不仅要验证功能正确性，更要对信息安全性进行系统评估与度量：

a） 安全性指标验证：确认在论证阶段设定的目标达成情况，例如数据加密覆盖率、入侵防御成功率、漏洞修复时间等；

b） 攻防演练与渗透测试：通过白盒/黑盒渗透测试、红队对抗测试，对系统抵御入侵、DDoS 攻击和对抗样本攻击能力进行验证；

c） 缺陷分析与漏洞修复闭环：对发现的安全漏洞和缺陷进行根因分析，建立漏洞披露与修复流程，保证其能在规定时间内完成修复，并将结果反馈到研制团队，形成安全性改进闭环。

10.4 维护与更新阶段

在系统投入运行后，需持续监控与改进安全性，确保长期抵御新型威胁：

a） 安全运维监控：建立集中日志、监控与 SIEM 平台，实时检测可疑访问行为或异常模型输出，实现快速响应；

b） 安全退化监测：防止随着代码迭代引入新的安全风险。通过 CI/CD 流水线自动检查，确保覆盖率不降低、漏洞扫描通过率不下降；

c） 补丁与威胁情报管理：建立全面的漏洞响应机制，自动追踪第三方组件与模型依赖的漏洞，结合威胁情报库动态更新防御规则；

d） 人员与流程安全：强化特权访问管理（PAM）、多因素认证机制，对运维人员定期开展安全培训，减少人为操作失误带来的风险；

e） 应急响应与恢复：制定应急响应预案并定期演练，确保在遭受攻击或发生安全事件时，系统能迅速恢复至可信状态。

参 考 文 献

[1] GB/T 40647-2021 智能制造 系统架构

[2] GB/T 43782-2024 人工智能 机器学习系统技术要求

[3] GB/T 45079-2024 人工智能 深度学习框架多硬件平台适配技术规范

[4] GB/T 45225-2025 人工智能 深度学习算法评估

[5] NASA-STD-8729.1 Nasa reliability and maintainability standard for spaceflight and support systems

[6] NASA-STD-8739.8 Software assurance and software safety standard

[7] NPR 7150.2 NASA software engineering requirements

[8] NASA-GB-8719.13 NASA Software Safety Guidebook

[9] ISO/IEC 23053:2022 Framework for AI systems using machine learning

[10] ISO/IEC 24668 信息技术 人工智能 大数据分析的过程管理框架

[11] ISO/IEC TR 24030-2024 信息技术 人工智能 使用案例

[12] AIOSS-01-2018 人工智能 深度学习算法评估规范

ICS 03.120.01
CCS L 05

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35005—2026

复杂智能系统保障性技术要求

Technical requirements for supportability of complex intelligent systems

2026-02-28 发布　　2026-02-28 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》的规定编写。

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）、中国科学院声学研究所、中国航空工业集团公司沈阳飞机设计研究所、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、李晓亮、郝程鹏、周怡婧、刘东、代国良、邵麒、姚琪、司昌龙、吴梦丹、刘磊。

复杂智能系统保障性技术规范

1 范围

本文件规定了复杂智能系统保障性的适用对象、技术要求、支撑技术方法及全生命周期等内容，描述了复杂智能系统通用质量特性中有关保障性方面的相关指标及方法。

本标准适用于在开放、不确定环境中运行的复杂智能系统的保障性要求、测评与验证活动。

2 规范性引用文件

下列文件中的有关条款通过引用而成为本文件的条款。凡注日期或版次的引用文件，其后的任何修改（不包括勘误的内容）或修订版本都不适用于本文件，凡不注日期或版次的引用文件，其最新版本适用于本文件。

GB/T 42018—2022　信息技术 人工智能　平台计算资源规范

GB/T 41867—2022　信息技术 人工智能　术语

GB/T 45958—2025　网络安全技术 人工智能计算平台安全框架

GJB 451A—2005　可靠性维修性保障性术语

GJB 1909A—2009　装备可靠性维修性保障性要求论证

T/ CICC 35012—2025　复杂智能系统可靠性技术要求

3 术语与定义

GB/T 41867—2022、GB/T 42018—2022、GB/T 45958—2025、GJB 451A—2005、GJB 1909A—2009和T/ CICC 35012—2025确立的以及下列术语和定义适用于本文件。

3.1

复杂智能系统　complex intelligent systems

由感知、认知、决策与执行等功能模块构成，采用机器学习等方法，在不确定、开放环境下执行任务的人机环协同系统。其复杂性体现在多源异构数据、动态场景、要素耦合以及全生命周期演化。

3.2

智能系统保障　support of intelligent systems

智能系统部署前后为确保其持续可靠运行并满足任务需求而实施的所有活动的总和。

3.3

智能系统保障性　supportability of intelligent systems

智能系统通过合理的设计特性与规划的保障资源，在任务周期内持续、可靠、安全地运行，并在异常或突发情况下具备自恢复、人工干预与升级演化能力的一种系统保障能力。

3.4

AI 平台　platform for AI

为AI应用提供计算、存储、网络与开发运维能力的软硬件与服务集成，包括但不限于CPU、GPU、AI加速器、系统软件、中间件、框架与接口等。

3.5

保障资源　support resources

使用与维护智能系统所需的硬件、软件、人员、数据及算力资源的统称。

3.6

训练集　training set

用于估计和更新模型参数的数据集。

注：训练集应与验证集、测试集在样本层面互不重叠；对时间序列或主体相关数据，应采取时间/主体隔离以防信息泄漏。

[来源：GB/T 41867—2022，3.2.34，有修改]

3.7

测试集　test set

用于在模型训练与选择完成后，对模型在未见数据上的性能进行客观评估的独立数据集。

注：测试集仅用于最终评估，不应用于任何形式的训练或调参决策。

[来源：GB/T 41867—2022，3.2.3，有修改]

3.8

验证集　validation set

用于模型开发阶段选择模型结构与调参等的独立数据集，不参与模型参数的最终训练。

注：验证集应与训练集、测试集互不重叠，避免信息泄漏。

[来源：GB/T 41867—2022，3.2.35，有修改]

3.9

算力资源　computing power resources

用于执行数值运算、逻辑推理和算法处理的处理单元集合，体现为对计算任务的处理能力。

3.10

模型　model

基于数据、知识或规则，经形式化表达并能通过算法运行，用以表征问题域中的对象关系、状态映射或决策过程的数学化或逻辑化结构。

3.11

训练　training

在给定的训练数据及约束条件下，通过优化算法调整模型参数以最小化（或最大化）预定目标函数的过程。

[来源：GB/T 42018—2022，3.11，有修改]

3.12

测试　testing

为验证和确认系统、子系统或模型性能是否满足设计要求，对其在特定环境下进行功能性、性能、可靠性及安全性评估的活动。测试通常包括测试方案制定、数据准备、测试执行、结果分析与迭代验证。

4 缩略语

下列缩略语适用于本文件。

ACL	访问控制列表（Access Control List）
AI	人工智能（Artificial Intelligence）
BN	批量归一化（Batch Normalization）
CI/CD	持续集成/持续部署（Continuous Integration/Continuous Deployment）
CMDB	配置管理数据库（Configuration Management Database）
CNN	卷积神经网络（Convolutional Neural Network）
CPU	中央处理器（Central Processing Unit）
DPU	数据处理器（Data Processing Unit）
GPU	图形处理器（Graphics Processing Unit）
IPSec	互联网协议安全（Internet Protocol Security）
K8s	容器编排引擎（Kubernetes）
ML	机器学习（Machine Learning）
MTBF	平均无故障时间（Mean Time Between Failures）
MTTR	平均修复时间（Mean Time To Repair）
NPU	神经处理器（Neural Processing Unit）
QoS	服务质量（Quality of Service）
RNN	循环神经网络（Recurrent Neural Network）
RL	强化学习（Reinforcement Learning）
SGD	随机梯度下降（Stochastic Gradient Descent）
TLS	传输层安全（Transport Layer Security）
TPU	张量处理器（Tensor Processing Unit）
VM	虚拟机（Virtual Machine）
VPN	虚拟专用网络（Virtual Private Network）

5 智能系统保障性适用对象

5.1 数据资源保障

数据资源保障围绕智能系统运行所需的训练、测试及业务应用数据，建立覆盖采集、存储、处理、传输、使用全流程的管理规范与运行机制，保障数据质量、安全与合规性，为模型训练、性能验证及运行决策提供可靠的数据支撑。

5.2 计算资源保障

计算资源保障为智能系统提供全场景、全周期的计算环境支持，通过对存储、网络、软件框架及基础设施等进行系统性统筹与保障，确保系统在各类运行环境下稳定、高效、连续、安全运行，满足系统长期运行与任务执行的可靠性要求。

5.3 算力资源保障

算力资源保障聚焦智能系统的算力供给需求，构建全寿命周期算力支撑机制，确保系统在规定任务与预期环境下，持续获得可靠、稳定、安全的算力供给，保障各类算法运算与任务处理高效开展。

5.4 模型保障

模型保障面向智能系统核心模型，建立覆盖设计、开发、训练、部署、迭代全生命周期的管理规范与管控机制，通过全流程质量控制与合规审查，确保模型运行精准、稳健、安全，为系统完成使命任务提供核心技术支撑。

5.5 训练保障

训练保障针对智能系统模型训练环节，建立覆盖训练数据准备、训练环境搭建、训练过程执行、训练结果验证的全流程规范与机制，保障模型训练过程科学、可控，确保训练结果有效可用。

5.6 测试保障

测试保障面向智能系统从模块组件到整体功能的全维度测试工作，建立覆盖测试方案设计、测试数据构建、测试执行、缺陷管理、测试结果评估的全流程规范与机制，全面验证系统性能，确保系统满足设计指标与使用要求。

5.7 使用保障

使用保障针对智能系统的实际应用与运行使用环节，建立覆盖使用准入、操作规范、运行监控、异常处置、使用评估及运维支撑的全流程规范与机制，保障系统使用过程的合规性、安全性、便捷性，确保系统在实际应用场景中稳定发挥功能，满足用户使用需求与长期可靠运行要求。

6 智能系统保障性定性要求

6.1 数据资源保障性定性要求

6.1.1 训练数据保障性定性要求

训练数据为智能系统建模核心资源，须满足覆盖性、统一性、合规性要求，保障模型学习科学稳健，支撑系统使命任务。具体要求如下：

a) 覆盖管控：依据典型场景、边界条件及异常样本分布，明确数据覆盖范围；校验分布合理性，规避偏置风险，防止模型失效；

b) 格式标注规范：制定样本采集、标注全流程统一格式与标注规范；明确异构处理要求，避免模型歧义，保障数据一致可用；

c) 全生命周期追溯：建立数据采集、存储、处理等全流程追溯机制；明确各环节追溯要素，确保过程可追溯、状态可核查；

d) 安全防护：针对对抗样本、恶意篡改等风险，制定安全规范；明确防篡改、防污染等措施，建立风险监测机制，保障数据安全完整；

e) 合法合规：遵循法律法规与行业规范，建立数据来源与使用合规审查机制；明确隐私保护、版权核查要求，确保数据合法合规。

6.1.2 测试数据保障性定性要求

测试数据为智能系统性能验证核心资源，须契合系统高复杂度、多场景适配特点，满足覆盖性、真实性、安全性及合规性要求，保障测试结果有效，支撑系统验收交付。具体要求如下：

a) 覆盖管控：依据系统全功能、典型场景、极端工况，明确测试数据覆盖范围；涵盖正常、异常、边缘及对抗样本，规避测试盲区，验证性能边界；
b) 真实一致：贴近真实运行数据特征，明确真实性校验标准；统一数据格式、标注规则，保障兼容性，避免测试偏差；
c) 全生命周期管控：建立测试数据全流程管理机制；明确各环节要点，实现可追溯、状态可控；
d) 安全防护：针对数据泄露、篡改等风险，制定防护规范；明确加密、访问控制等措施，建立风险监测机制，保障数据安全；
e) 合法合规：遵循法规与行业规范，建立合规审查机制；明确隐私脱敏、版权核查要求，确保数据合法使用。

6.1.3 运行数据保障性定性要求

运行数据是智能系统运行中影响性能与安全的关键要素，须契合系统实时响应、长期稳健运行特点，满足实时性、稳定性、准确性及安全性要求，为系统全生命周期可靠运行与精准决策提供可信数据支撑。具体要求如下：

a) 实时性管控：依据系统使命任务，明确数据采集、传输、处理的实时性指标；建立监测预警机制，确保时效满足响应要求，规避延迟风险；
b) 稳定性保障：结合实际运行环境，建立冗余备份、容错的数据获取与传输机制；明确链路可靠性校验标准，防止数据丢失、传输中断影响运行；
c) 准确性校验：依据数据输入准确性规范，建立使用前校准、清洗与验证流程；明确校验标准与容错阈值，确保数据与真实环境一致，保障决策精准；
d) 监测溯源：针对运维与审计需求，在数据流全链路设置监测节点与日志功能；明确日志要素、留存周期及权限，支撑异常检测、故障定位与溯源；
e) 全生命周期有效性保障：契合长期运行需求，建立数据在线质量动态评估与持续验证机制；定期复盘数据质量与性能匹配度，确保全生命周期数据稳定、性能有效。

6.2 计算资源保障定性要求

计算资源保障旨在确保智能系统在各类运行环境下具备稳定、高效、连续和安全的计算支持，覆盖存储、网络、软件框架及基础设施等方面，满足长期运行与任务可靠性的要求。计算资源保障的定性要求包括：

a) 算力保障：计算单元配置应满足智能任务的最低性能需求，并支持扩展性与可替代性，避免单点故障或供应链限制造成系统停滞；
b) 存储保障：存储系统应确保容量充足、访问高效与数据一致性，同时具备冗余与校验机制，避免因存储异常导致数据丢失或损坏；
c) 网络保障：计算节点间通信应保证带宽、延迟与安全性，具备同步一致性机制及链路冗余能力，在局部故障下仍可维持整体任务连续性；
d) 软件框架保障：运行软件与框架应具备接口兼容、跨平台迁移及版本可控能力，并支持容器化或虚拟化部署，以保证系统的可移植性与可维护性；
e) 基础设施保障：供电、散热与运行环境应具备冗余与容灾能力，关键节点必须配备备用电源及温控监测机制，确保计算资源长期稳定；
f) 综合保障：计算资源应支持多任务调度与统一监控，能够在资源受限条件下提供压缩或量化模型运行，并在断电、断网场景下具备应急运行与数据恢复能力。

6.3 算力资源保障定性要求

算力资源保障旨在确保智能系统在全寿命周期内，能在规定任务和预期环境下获得可靠、持续和安全的算力支撑。算力保障应从以下方面提出定性要求：

a) 算力应在规定时间内可获取，并通过优先级分配机制确保关键任务的算力需求，不得因调度失当导致任务中断；
b) 算力应在长时运行和高负载条件下保持稳定，具备容灾与冗余措施，保证任务在局部算力失效时仍可维持运行；
c) 算力应能根据任务变化和环境条件进行动态调配，支持异构资源协同，避免因架构或任务变动导致算力不可用；
d) 算力资源应具备横向和纵向扩展能力，扩展过程不得中断系统运行，扩展后需保持资源一致性和高效利用；
e) 算力应具备访问控制、使用隔离和异常检测，防止越权调用和恶意占用，保障算力与调度指令的完整性；
f) 算力应具备监测、诊断和恢复机制，能够在异常或失效后于规定时间内恢复任务支撑，并支持模块化替换与升级维护。

6.4 模型保障性定性要求

为确保智能系统在全寿命周期内能够可靠、有效地支撑任务执行，应满足以下保障性定性要求：

a) 模型应能够按照设计目标实现输入与输出的正确映射；训练与验证过程应确保模型在预期任务域内具备可接受的精度和一致性；
b) 模型应在输入扰动、噪声或环境变化情况下保持稳定性能，不得因微小干扰导致显著偏差或失效；
c) 模型应具备在合理范围内推广至未见数据或扩展任务场景的能力，并能够通过再训练或迁移学习机制适应任务需求变化；
d) 模型应具备可解释性，对关键输出结果应可提供溯源依据或决策逻辑说明，以支持验证、审计和信任；
e) 模型应能适配不同算力平台和运行环境，具备跨系统、异构硬件架构下的移植和扩展能力；
f) 模型应支持生命周期管理，包括版本控制、更新升级、性能评估与淘汰替换，确保长期运行中的可靠性和演进性；
g) 模型的开发与使用应符合相关标准、法规和伦理要求，不得在敏感任务中出现不符合规范的数据偏差或决策偏差。

6.5 训练保障定性要求

为确保智能系统能够可靠、有效地训练，应满足下列训练方面的保障性定性要求：

a) 训练所用数据应具备真实性、完整性、代表性，并经过清洗与标注，防止因数据缺陷引入系统性偏差；
b) 训练流程应可跟踪、可复现，包含记录数据版本、模型版本、超参数配置等，确保训练结果可验证；
c) 训练过程应防范数据投毒、后门注入、对抗性样本攻击等风险，训练环境应具有访问控制和隔离措施；
d) 训练机制应支持模型再训练、增量学习或迁移学习，以便应对新任务或数据分布变化；
e) 训练方案应在保证效果的前提下，合理配置算力和存储资源，避免过度消耗与资源浪费。

6.6 测试保障定性要求

为确保智能系统测试过程顺利进行并获得可信结果，应满足下列测试方面的保障性定性要求：

a) 应制定完整测试计划，明确测试目标、范围、方法、资源及优先级；
b) 测试环境应可控、可复现，并与应用环境保持等效性；环境应隔离外部干扰；
c) 测试数据应包括典型场景、边界条件与异常情况，确保覆盖系统预期任务域；
d) 测试应按既定用例和流程执行，不得随意更改；关键步骤应进行记录与追溯；

e) 测试工具与平台应符合标准规范，并定期验证正确性与有效性；
f) 测试过程应防止数据、模型泄露，测试平台应具备访问控制和隔离措施；
g) 应对系统精度、稳定性、时效性等性能指标进行验证，并开展干扰和异常工况下测试；
h) 当系统更新时，应执行回归测试，验证系统性能不退化；
i) 测试结果应可解释，并应具备完整记录以支持审计、合规与认证。

6.7 功能保障性定性要求

6.7.1 感知保障定性要求

为确保智能系统感知功能持续、可靠、安全运行，应满足下列感知方面的保障性定性要求：
a) 系统应具备多源感知与信息融合能力，以保证环境信息获取的完整性和准确性；
b) 感知结果应在时效性、分辨率和精度上满足任务需求；
c) 感知数据处理过程应具备容错与异常检测机制，防止因单一传感器故障造成功能丧失；
d) 感知模块应防范外部干扰与攻击，不得因信号欺骗、干扰或仿真样本导致失效；
e) 感知能力应支持自检与状态监测，能够在性能下降时发出告警。

6.7.2 认知保障定性要求

为确保智能系统认知功能能够准确表征环境语义、适配动态场景，认知保障应满足下列保障性定性要求：
a) 系统应具备对感知信息的分类、识别、推理与理解能力，确保其能够正确表征环境语义；
b) 认知过程应能够处理不确定性与模糊性，维持在多种环境条件下的鲁棒性；
c) 认知结果应可解释，可通过可视化或指标表征验证正确性；
d) 认知功能应避免固有偏差，保障公平、公正与合规性；
e) 认知处理模块应支持动态学习与适应，满足系统在不同任务下的演进需求。

6.7.3 决策保障定性要求

为确保智能系统决策功能贴合任务目标、规避运行风险，决策保障应满足下列保障性定性要求：
a) 系统应能基于认知结果生成合理、有效的决策方案，并满足任务目标；
b) 决策过程应包含多方案权衡、风险评估和冲突解决机制；
c) 决策结果应在安全约束和法律法规允许的范围内生成，不得产生违法或高风险行为；
d) 决策逻辑应具备透明性和可追溯性，以支持事后审计和责任认定；
e) 系统应在关键任务中提供人工干预接口，保证高风险场景下人的可控性。

6.8 执行应用保障性定性要求

6.8.1 控制保障定性要求

为确保智能系统控制功能安全可控、适配动态环境，控制保障应满足下列保障性定性要求：
a) 控制模块应对执行机构提供准确、稳定、鲁棒的控制信号，确保执行过程安全可控；
b) 控制系统应具备实时监测与反馈机制，能够根据环境变化自适应调节；
c) 控制逻辑应考虑极端情况和故障模式，并提供冗余与容错能力；
d) 控制接口应符合安全通信与防护要求，防止恶意篡改和非法控制；
e) 控制行为不得引发越界操作或危害人员与设备安全。

6.8.2 使用保障定性要求

为确保智能系统使用便捷、合规、可维护，使用保障应满足下列保障性定性要求：
a) 系统应提供符合人体工学和任务需求的使用界面，保证用户理解性和可操作性；
b) 使用过程应支持操作提示、告警机制和安全防护措施，避免误用和滥用；
c) 系统功能和权限应分级配置，不得允许未经授权的功能访问；

d) 在使用阶段，系统应提供必要的数据保护与隐私保护措施，符合适用法律法规；

e) 系统应支持用户反馈收集与故障报告，便于持续改进和维护。

7 智能系统保障性定量要求

7.1 升级回滚成功率

用于度量系统升级失败或出现异常后，成功恢复到升级前稳定版本的比例，计算方法见公式（1）。

$$\mathrm{L}=\frac{a}{b}\times 100\% \quad \text{.....（1）}$$

式中：

L——升级回滚成功率，无单位；

a——回滚成功次数，单位为次；

b——升级失败总次数，单位为次。

7.2 升级耗时

用于度量从升级操作正式开始（即升级触发/指令下发）到系统完成升级并恢复正常服务状态之间所消耗的总时间，计算方法见公式（2）。

$$\mathrm{T}=a-b \quad \text{.....（2）}$$

式中：

T——升级耗时，单位为小时（h）或分钟（min）；

a——升级结束时间点，单位为时间戳；

b——升级开始时间点，单位为时间戳。

7.3 模型更新成功率

用于度量系统进行模型更新时，能够成功实现更新并保持模型性能稳定的比例，计算方法见公式（3）。

$$\mathrm{S}=\frac{a}{b}\times 100\% \quad \text{.....（3）}$$

式中：

S——模型更新成功率，无单位；

a——模型更新成功次数，单位为次；

b——模型更新总次数，单位为次。

7.4 迭代周期时间

用于度量系统完成一次完整的模型迭代过程所需的平均时间，计算方法见公式（4）。

$$\mathrm{T}=\frac{1}{N}\sum_{i=1}^{N}T_i \quad \text{.....（4）}$$

式中：

T——迭代周期时间，单位为小时（h）或天（d）；

N——总迭代次数，单位为次；

T_i——第i次迭代的时间，单位为小时（h）或天（d）。

7.5 迁移成功率

用于度量在平台迁移过程中，系统成功适应新平台并正常运行的比例，计算方法见公式（5）。

$$\mathrm{S}=\frac{a}{b}\times 100\% \quad \text{.....（5）}$$

式中：

S——迁移成功率，无单位；

a——成功迁移次数，单位为次；

b——总迁移次数，单位为次。

7.6 迁移时间

用于度量从一个平台迁移到另一个平台所需的总时间，计算方法见公式（6）。

$$T = a - b \quad \text{.....（6）}$$

式中：

T——迁移时间，单位为小时（h）或分钟（min）；

a——系统迁移后稳定运行时间点，单位为时间戳；

b——迁移开始的时间点，单位为时间戳。

7.7 平台适配次数

用于度量系统在迁移过程中需要调整或修改才能适应目标平台的次数，计算方法见公式（7）。

$$N = c \quad \text{.....（7）}$$

式中：

N——平台适配次数，单位为次；

c——适应目标平台的调整次数，单位为次。

7.8 跨平台运行稳定性

用于度量系统在不同平台上运行时的稳定性，计算方法见公式（8）。

$$R = (1 - \frac{a}{b}) \times 100\% \quad \text{.....（8）}$$

式中：

R——跨平台运行稳定性，无单位；

a——系统崩溃次数，单位为次；

b——不同平台的运行次数，单位为次。

7.9 配置管理频率

用于度量系统在一段时间内进行配置调整的次数，计算方法见公式（9）。

$$P = \frac{a}{b} \quad \text{.....（9）}$$

式中：

P——配置管理频率，单位为次/时间单位（如：次/天、次/周）；

a——配置管理次数，单位为次；

b——时间区间，单位为天（d）、周（w）等。

7.10 人员培训覆盖率

用于度量团队成员接受专业培训的比例，计算方法见公式（10）。

$$R = \frac{a}{b} \times 100\% \quad \text{.....（10）}$$

式中：

R——人员培训覆盖率，无单位；

a——接受培训的成员数，单位为人；

b——团队总成员数，单位为人。

7.11 人员流动率

用于度量一定时间内团队成员离职的比例，计算方法见公式（11）。

$$H = \frac{a}{b} \times 100\% \quad \text{.....（11）}$$

式中：

H——人员流动率，无单位；

a——离职的员工数，单位为人；

b——团队总成员数，单位为人。

7.12 硬件资源利用率

用于度量系统硬件资源的实际使用情况与其总容量的比例，计算方法见公式（12）。

$$R = \frac{\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N} b_i} \times 100\% \quad \text{.....（12）}$$

式中：

R——硬件资源利用率，无单位；

a_i——第i个硬件资源实际性能使用量，单位为硬件性能单位；

b_i——第i个硬件资源总性能，单位与对应硬件实际性能使用量单位一致；

N——硬件资源总数量，单位为个。

7.13 冗余设备可用率

用于度量系统中冗余设备的可用性，计算方法见公式（13）。

$$R = \frac{a}{b} \times 100\% \quad \text{.....（13）}$$

式中：

R——冗余设备可用率，无单位；

a——可用冗余资源数，单位为个；

b——总资源数，单位为个。

7.14 数据完整性达标率

用于度量训练、测试或运行数据中符合完整性要求（无缺失、无无效值、格式规范）的样本比例，反映数据质量核心指标，计算方法见公式（14）。

$$D = \frac{m-n}{m} \times 100\% \quad \text{.....（14）}$$

式中：

D——数据完整性达标率，无单位；

m——数据总样本数，单位为个；

n——缺失关键字段、存在无效值或格式错误的样本数，单位为个。

7.15 数据标注准确率

用于度量标注数据中符合预设标注规范的样本比例，保障标注数据可靠性，计算方法见公式（15）。

$$A = \frac{k}{t} \times 100\% \quad \text{.....（15）}$$

式中：

A——数据标注准确率，无单位；

k——标注结果与标准一致的样本数，单位为个；

t——参与标注校验的样本总数，单位为个。

7.16 算力负载均衡度

用于度量多算力单元（CPU/GPU/TPU等）的负载分布均衡性，反映算力调度合理性，计算方法见公式（16）。

$$B = （1 - \frac{\mu}{\sigma}） \times 100\% \quad \text{.....（16）}$$

式中：

B——算力负载均衡度，无单位（取值范围0%-100%）；

σ——各算力单元负载率的标准差，无单位；

μ——各算力单元负载率的平均值，无单位（取值范围0-1）。

7.17 模型性能变化率

用于度量模型在输入扰动（噪声、对抗样本）下的性能变化程度，反映模型抗干扰能力，计算方法见公式（17）。

$$D = \frac{c-h}{c} \times 100\% \quad \text{.....（17）}$$

式中：

D——模型性能变化率，无单位（百分比表示）；

c——模型在无扰动输入下的基准性能（如准确率、F1值），单位为模型性能指标对应的单位；

h——模型在预设扰动输入下的实际性能，单位与基准性能单位一致。

8 智能系统保障性设计技术

8.1 数据资源保障性设计

数据资源保障性设计是针对智能系统运行所需的训练、测试及使用数据，建立覆盖采集、存储、处理、传输和使用全流程的规范与机制，确保数据质量、安全与合规性，为模型训练、性能验证和运行决策提供可靠支撑。相关方法如下：

a) 数据治理方法：建立统一的数据标准、格式规范、标注规范，保证一致性；
b) 数据质量控制：采用采样检测、冗余比对、噪声过滤与缺失修复等技术，保证数据完整性与准确性；
c) 数据全生命周期管理：建立数据采集、存储、转移、归档、销毁全流程记录与可追溯机制；
d) 数据安全防护：采用加密存储、访问控制列表（ACL）、水印、脱敏等措施，防范数据泄露与篡改。

8.2 计算资源保障性设计

计算资源保障性设计旨在确保智能系统在各类运行环境下具备稳定、高效、连续和安全的计算支持，覆盖存储、网络、软件框架及基础设施等方面，满足长期运行与任务可靠性的要求。相关方法如下：

a) 高可用架构：采用分布式架构、故障转移、冗余备份技术，提升平均无故障时间（MTBF），降低平均修复时间（MTTR）；
b) 性能调度：配置任务优先级调度和负载均衡机制，防止计算瓶颈；
c) 网络与通信安全：采用VPN、TLS/IPSec、零信任认证方式，保障传输安全；
d) 虚拟化与容器化：支持容器化、虚拟机（VM）、微服务管理框架，基于容器编排引擎（K8s）实现资源调度，提高跨平台部署能力；
e) 基础设施监控：在电力、散热、机房环境等方面配备传感与告警机制，保障信息技术（IT）基础设施长期可靠运行。

8.3 算力资源保障性设计

算力资源保障性设计旨在确保智能系统在全寿命周期内，能在规定任务和预期环境下获得可靠、持续和安全的算力支撑。相关方法如下：

a) 算力池化与调度：采用资源池化技术和智能调度平台，支持弹性扩展，实现CPU、GPU等算力资源的统一管理；
b) 异构算力适配：支持 CPU/GPU/TPU/DPU/NPU 等多种算力单元的协同运行；
c) 任务优先级分配：使用QoS策略，保证关键任务优先获得算力资源；
d) 容灾与冗余：建立跨机房或跨节点的冗余配置，防止单点故障；
e) 算力安全：通过访问控制、使用隔离、算力使用监测与异常检测，防范算力资源的恶意占用、越权调用与非法篡改。

8.4 模型保障性设计

为确保智能系统模型在全寿命周期内能够可靠、有效地支撑任务运行，要求在模型设计阶段采取以下措施提升保障性，相关方法如下：

a) 模型设计规范：采用模块化、可解释性设计，并配套参数可追溯机制，针对卷积神经网络（CNN）、循环神经网络（RNN）等主流模型制定统一设计标准；
b) 模型验证方法：使用仿真测试、交叉验证、形式化验证来检查正确性与稳定性；
c) 模型保护技术：采用水印、哈希、加密、模型访问控制防止非法复制或攻击；
d) 调整与再训练机制：支持模型迁移学习、增量学习、在线更新机制，基于深度学习和强化学习（RL）算法适配环境变化，提升模型更新成功率；
e) 生命周期管理：对模型版本、性能指标、配置参数建立档案，支持升级与淘汰依托配置管理数据库（CMDB）实现全流程管控。

8.5 训练保障性设计

训练保障性设计是为确保模型训练过程科学、可靠且安全，要求在训练数据、流程、环境及资源配置等方面实施管理与控制。相关方法如下：

a) 训练全程追踪：记录训练数据版本、模型版本、超参数配置及运行日志，支持可复现；
b) 数据与标注审查：建立人工审核与自动检测机制，减少偏差与污染；
c) 安全训练技术：采用数据投毒检测、对抗训练、防后门检测机制，保障安全性；
d) 资源分配合理化：通过调度优化提升算力使用效率，避免资源浪费，依托硬件资源利用率指标监控算力使用情况，采用随机梯度下降（SGD）、批量归一化（BN）等算法优化训练资源消耗；；
e) 环境仿真训练：在不同物理或虚拟环境场景下开展训练，提高模型鲁棒性。

8.6 测试保障性设计

测试保障性设计是为确保智能系统测试过程顺利进行并获得可信结果，而对测试计划、环境、数据、过程及管理实施的规范和控制。相关方法如下：

a) 测试计划制定：覆盖功能、性能、鲁棒、安全性与可靠性指标；
b) 多环境测试：应在典型场景、极端环境和资源受限等条件下测试；
c) 自动化测试平台：采用自动化工具、CI/CD 测试框架，提高效率与一致性；
d) 安全性测试：采用渗透测试、对抗样本测试、异常工况注入；
e) 回归与持续测试：系统更新后执行回归测试，确保性能不退化；
f) 结果可追溯：测试记录、数据和结果应完整归档并支持审查与认证。

8.7 使用保障性设计

使用保障性设计旨在为复杂智能系统建立覆盖人机交互、操作运维、权限管控、安全使用的全流程保障机制，相关方法如下：

a) 人机交互适配：贴合人体工学与任务场景设计简洁操作界面，对接系统核心模块，设置定制化面板及操作引导，降低使用门槛；
b) 操作管控：建立操作日志留存、追溯与审计机制，关联配置变更，设置操作校验与二次确认，明确责任归属；
c) 权限管控：基于ACL构建分级权限体系，按最小权限分配角色权限，实现权限动态分配回收，规范申请审批流程；
d) 安全防护：集成异常操作监测告警，联动IDS防护，对敏感数据脱敏溯源，高风险操作设置双重验证；
e) 故障处置：集成故障上报与反馈模块，建立闭环处置机制，嵌入自助排查指南，设置异常预警指标；

f) 全周期适配：设计可扩展架构，适配系统升级迭代，搭建定制化培训体系，依托用户反馈优化使用体验；
g) 多场景兼容：适配本地、远程、移动等场景，采用VPN、TLS保障安全，实现终端自适应，提升跨平台稳定性。

9 智能系统保障性全生命周期活动

9.1 需求阶段

在需求阶段，应将智能系统的风险分析和保障策略纳入系统建设与使用的决策过程。此阶段需综合考虑信息安全保障要素，以实现系统建设与保障工作的同步规划、同步实施和同步使用。保障性任务主要包括：

a) 基于任务目标，明确安全保障需求，定义完成系统任务所需的保障目标及场景（如极端情况、边界条件）；
b) 确保任务定义具备可验证性与可追溯性，并建立初步的保障性指标体系。

9.2 开发阶段

在开发阶段，应赋予智能系统全面的保障能力。需开展以下关键工作：

a) 系统需求分析：明确安全与保障需求，将其纳入系统架构设计；
b) 引入关键设计特性：在体系架构与模块层面加入可观测性、可解释性和模块热替换能力等特性，提升系统的可靠性和维护性；
c) 执行项目管理活动：包括预算规划、资源协调及相关的管理保障。

9.3 验证与确认阶段

验证与确认阶段专注于对系统是否满足设计阶段提出的保障性目标进行全面测试与确认。保障性任务包括：

a) 对系统功能及非功能性保障目标进行系统性验证，确保其与设计规范一致；
b) 通过覆盖全面的测试用例验证系统在不同场景（包括边界条件）下的稳定性与安全性。

9.4 部署阶段

部署阶段的关键任务是确保保障能力的落地，通过全面资源配套和系统设定实现系统的安全可靠运行。保障性任务包括：

a) 部署与之匹配的保障资源，包括硬件支持、智能系统管理工具及人员能力提升等；
b) 对管理、运维和使用人员进行全面能力培训，确保其能够胜任相关保障工作。

9.5 运维阶段

在运维阶段，应对系统的运行状态进行实时监测以确保其持续稳定性和有效性。保障性任务包括：

a) 持续监测系统运行状态及外部输入情况，确保模型始终处于安全高效的状态；
b) 及时识别异常行为并采取对应措施，防止异常扩散或升级成更大问题。

9.6 重新评估阶段

重新评估阶段的重点是基于系统任务、数据和环境的变化，系统性地审查现有的保障机制与体系。保障性任务包括：

a) 定期评估模型性能、保障体系和外部环境的适配性，及时更新和优化现有方案；
b) 根据系统运行中积累的数据，动态调整其保障性配置。

9.7 报废阶段

在生命周期结束时，必须确保智能系统的报废过程安全有序，以避免因系统遗留问题而引发的隐患。保障性任务包括：

a) 对系统下线过程进行全面审核，确保所有安全保障手段在报废环节得以严格落实；
b) 彻底脱敏和销毁系统中存储的敏感数据，同时对算法模型进行安全销毁，防止非法滥用。

参 考 文 献

[1] ISO/IEC 5259-1:2024 人工智能分析和机器学习（ML）的数据质量 第1部分
[2] ISO/IEC 5259-3:2024 人工智能分析和机器学习（ML）的数据质量 第3部分
[3] ISO/IEC 5338-2023 信息技术 人工智能 人工智能系统生命周期过程
[4] ISO/IEC 23053:2022(E) 采用机器学习（ML）的人工智能（AI）系统框架

ICS 35.240.01
CCS L 05

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35006—2026

复杂智能系统故障预测与健康管理技术要求

Technical requirements for prognostics and health management of complex intelligent systems

2026-02-28 发布　　2026-02-28 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）、四川航天系统工程研究所、中国科学院声学研究所、中国船舶集团有限公司综合技术经济研究院、中国航空工业集团公司沈阳飞机设计研究所、长龙航空维修工程有限公司、可靠性与环境工程技术国家级重点实验室、北京航空航天大学可靠性工程研究所。

本文件主要起草人：杨顺昆、范珈铭、苟晓冬、郝程鹏、吴梦丹、林焱辉、李璇、刘东、吴敏、周怡婧、刘杰、刘焱、闫戈、姚琪、许丹、刘磊、邓新蕴、徐雅丽、于功也、杨穗利、曾康。

复杂智能系统故障预测与健康管理技术规范

1 范围

本文件规定了面向复杂智能系统的故障预测与健康管理的技术要求，描述了复杂智能系统中故障预测与健康管理方面的总体架构、指标体系、全生命周期过程以及支撑技术。

本文件适用于面向复杂智能系统的故障预测与健康管理。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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 7826-2012 系统可靠性分析技术 失效模式和影响分析（FMEA）程序

GB/T 36344-2018 信息技术 数据质量评价指标

GB/T 43782-2024 人工智能 机器学习系统技术要求

GB/T 44442-2024 智能制造 远程运维系统 评价指标体系

GB/T 44662-2024 健康管理 终端设备数据采集与传输协议

GB/T 45087-2024 人工智能 服务器系统性能测试方法

GB/T 45225-2025 人工智能 深度学习算法评估

GB/T 45288.2-2025 人工智能 大模型 第2部分：评测指标与方法

GB/T 45401.2-2025 人工智能 计算设备调度与协同 第2部分：分布式计算框架

GB/T 45579-2025 机器人智能化视觉评价方法及等级划分

GB/T 45907-2025 人工智能 服务能力成熟度评估

GB/T 46315-2025 工业互联网平台 设备健康管理规范

AIIA/T 0221-2025 MaaS模型即服务技术与应用要求 第7部分：大模型API服务性能测试方法

3 术语和定义

GB/T 7826—2012，GB/T41867—2022， GB/T 45087—2024， GB/T 45579—2025界定的以及下列术语和定义适用于本文件。

3.1

复杂智能系统故障预测与健康管理 prognostics and health management of complex intelligent systems

是一种综合利用多源数据、人工智能算法及专家知识，对复杂智能系统及其组成部件的当前状态进行持续监测、评估与诊断，并对其未来演化趋势及潜在故障进行提前预警的技术体系。

3.2

状态特征量 condition characteristic quantity

从原始监测数据中提取、可直接反映特定故障模式或健康退化便于后续诊断与预测的特征量。

3.3

健康指数 health index

健康指数是将多个状态特征量或异构信息融合后得到的单一无量纲指标，用于量化系统整体健康水平。健康指数随时间单调退化，常被映射为剩余使用寿命估计。

3.4

预测有效期 prognostic horizon

从预测首次达到可接受精度开始，到实际失效为止这一段时间跨度。

4 缩略语

下列缩略语适用于本文件。

CNN	卷积神经网络 （Convolutional Neural Network）
CPT	条件概率表（Conditional Probability Table）
DAG	有向无环图（Directed Acyclic Graph）
DDM	漂移检测方法（Drift Detection Method）
ETL	提取、转换和加载（Extract， Transform， Load）
FAR	故障误报率（False Alarm Rate）
FDR	故障检测率（Fault Detection Rate）
FLOPS	每秒浮点运算次数（Floating-point Operations Per Second）
FLR	故障定位率 （Fault Localization Rate）
FMECA	失效模式、影响及危害性分析（Failure Mode， Effects， and Criticality Analysis）
GAN	生成对抗网络（Generative Adversarial Network）
HI	健康指数（Health Index）
LSTM	长短期记忆网络（Long Short-Term Memory）
MAR	故障漏报率（Missing Alarm Rate）
OOD	分布外（Out-of-Distribution）
PH	预测有效期（Prediction Horizon）
PHM	故障预测与健康管理 （Prognostics and Health Management）
ROC	接受者操作特性曲线 （Receiver Operating Characteristic）
RUL	预期剩余寿命（Remaining Useful Life）

5 复杂智能系统故障预测与健康管理技术框架

5.1 概述

复杂智能系统故障预测与健康管理（PHM）的层级划分遵循从数据感知到决策反馈的逻辑演进规律，旨在建立一套涵盖物理实体、逻辑软件及人工智能模型全维度的监控与管理框架。本章定义的七个层级构成了智能系统PHM的核心技术闭环，通过层级间确定的数据接口与逻辑调用，支撑系统在全生命周期内的安全性、可靠性与任务效能优化。

5.2 数据采集层

数据采集层负责对复杂智能系统运行过程中的全要素原始观测数据进行获取与初步汇聚。采集对象涵盖物理传感器输出、执行机构反馈、系统运行日志以及智能算法的内部状态参数。该层级通过边缘计算节点执行高频采样与硬件级时间戳标记，确保多模态数据在时空基准上的一致性。

5.3 数据处理层

数据处理层通过预定义的算法流程将原始数据转化为具备分析价值的结构化特征。该层级执行数据的提取、转换与加载操作，包括噪声平滑、异常点剔除及缺失值补全等预处理工序。针对物理部件，利用信号处理技术提取时频域特征；针对智能模型，通过高维映射提取反映推理性能的特征向量。

5.4 状态检测层

状态检测层通过比对实时特征与基准模型，识别系统当前的运行工况并判定是否存在偏差。该层级集成异常检测算法与逻辑推理机制，不仅监测物理组件的突发性故障，亦针对复杂智能系统特有的模型性能衰退进行监控，具体包括分布外检测、输入数据漂移及概念漂移监测。

5.5 健康评估层

健康评估层利用多源信息融合技术，对复杂智能系统的总体状态进行定量化评定。该层级通过综合物理退化指标、软件运行效能及模型置信度，计算系统整体的健康指数。评估逻辑基于多准则决策分析，考虑不同子系统对全局可靠性的权重影响，将系统健康状态划分为由正常至失效的连续等级，为系统的运行风险控制提供量化的状态依据。

5.6 故障诊断层

故障诊断层主要负责对已被检测出的异常状态或已确认的健康退化进行深层致因分析与模式确认。该层级利用因果推理、多模态关联分析及知识图谱技术，解析物理失效与逻辑错误的耦合关系。针对复杂智能系统，该层级重点区分并定位传感器偏差、执行机构硬件损耗、计算资源瓶颈、输入数据分布漂移及模型概念漂移等不同维度的故障诱因。

5.7 预测评估层

预测评估层基于历史趋势数据与物理演化机理，对系统未来的失效退化路径进行时序外推。该层级负责预测物理部件及智能模型达到预设失效阈值的剩余有效时间。预测结果包含剩余正常工作时长的期望值及不确定性量化指标。

5.8 决策生成层

在决策生成层根据健康评估结果与预测效能曲线，自动生成针对性的维护方案或系统调节指令。该层级在资源约束、成本目标及任务优先级的多目标优化框架下，制定视情维护阶段的计划，并触发智能系统的自适应调整逻辑。

6 复杂智能系统故障预测与健康管理指标体系

6.1 定性要求

6.1.1 数据完整性

数据完整性确保数据从修改、存储、传输到使用的全过程中，不会因人为错误、系统故障、恶意攻击或软硬件缺陷而遭到破坏或失真。

6.1.2 数据准确性

数据准确性表示数据值与其所代表的物理世界真实状态的符合程度，关注的是数据与客观事实之间的偏差。使用不准确的数据训练模型，会导致模型学习到一个与真实物理世界有系统性偏差的模型。

6.1.3 数据一致性

数据一致性衡量的是数据在智能系统内是否遵循已定义的内部逻辑规则和约束，关注数据与数据之间、数据与系统规则之间是否存在矛盾。

6.1.4 数据时效性

数据时效性衡量的是数据从产生到可用于分析和决策的时间延迟是否满足智能系统PHM分析需求。数据到达的延迟会减慢AI模型的在线学习或周期性更新的频率，使其无法快速适应设备状态的新变化。

6.1.5 数据规范性

数据规范性衡量的是数据是否遵循预定义的格式、语法和结构标准，保证数据处于正确的格式，确保数据可以被系统正确地解析和处理。不规范的数据会导致整个数据流的停滞。

6.1.6 数据可访问性

数据可访问性衡量的是被授权的用户或系统能够在需要时便捷、可靠地获取到数据的能力。它关注的是数据获取的过程是否通畅，以及是否有合适的工具和权限来发现和使用数据。

6.1.7 可扩展性

是系统在处理不断增长的工作负载时，仍能保持其性能和效率的能力。可扩展性包括通过增加硬件资源或增加节点数量来提升系统处理能力。

6.1.8 结果可解释性

当智能系统评估出健康度下降时，需要能追溯导致下降的主要原因。

6.1.9 预测时效性

预测时效性是衡量智能系统PHM 系统性能的核心指标之一，用于评价可靠的剩余使用寿命预测是否能够在必须采取行动的最后期限之前足够早地生成，从而为维护决策、资源调配及干预措施预留充足的准备时间。

6.1.10 决策准确性

决策准确性指的是系统生成的决策或建议的正确程度，智能系统需要基于历史和实时数据，利用先进的算法模型，准确地识别设备潜在的故障模式、定位故障原因和部位。

6.1.11 可执行性

可执行性是指智能系统生成的决策方案在现实世界中是否可行且易于操作。智能系统在生成决策时，必须充分考虑现实的资源限制。

6.2 定量指标

6.2.1 数据缺失率

数据缺失率（Missing）是指在预期的观测时间序列或数据集中，由于各种原因未能成功采集、传输或存储的数据点所占的比例。它是一个衡量数据完整性的关键质量指标。其具体形式如公式（1）所示：

$$Missing = \frac{X}{Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（1）$$

式中：

$Missing$——数据缺失率；

X——缺失数据点的数量；

Y——预期总数据点的数量。

6.2.2 数据丢包率

数据丢包率（Package Loss）定义为在数据传输过程中，成功发送的数据包中未能被接收端成功接收的数据包所占的比例。丢包通常以数据包为单位进行计算。直接反映了从传感器端到数据处理中心（边缘或云端）的通畅与可靠程度。其具体形式如公式（2）所示：

$$PackageLoss = 1 - \frac{X}{Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（2）$$

式中：

$PackageLoss$ ——数据丢包率；

X——成功接收的数据包数；

Y——发送的总数据包数。

6.2.3 故障诊断覆盖率

故障诊断覆盖率（Coverage）衡量的是一个智能系统PHM系统被设计用来能够检测和诊断的故障模式数量，占该设备所有已知或预期可能发生的故障模式总数的比例，是定义系统能力边界和验收系统资格的关键。应该对不同关键等级的设备/模块，设定不同的最低诊断覆盖率要求。具体形式如公式（3）所示：

$$Coverage = \frac{X}{Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（3）$$

式中：

$Coverage$ ——故障诊断覆盖率；

X——智能PHM系统设计规范中声明可以诊断的故障模式的数量；

Y——根据FMECA等分析得出的，该设备已有的、可能发生的故障总数。

6.2.4 故障检测率

故障检测率（FDR, Fault Detection Rate)是实际发生故障时，系统成功检测并报警的概率。具体形式如公式（4）所示：

$$FDR = \frac{X}{X+Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（4）$$

式中：

FDR ——故障检测率；

X——实际有故障，且系统成功报警的次数；

Y——实际有故障，但系统没报警的次数。

6.2.5 故障漏报率

故障漏报率（MDR, Missed Detection Rate）是实际发生故障时，系统未能检测到的概率。它与检测率是互补的。具体形式如公式（5）所示：

$$MDR = \frac{Y}{X+Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（5）$$

式中：

MDR ——故障漏报率；

X——实际有故障，且系统成功报警的次数；

Y——实际有故障，但系统没报警的次数。

6.2.6 故障虚警率

故障虚警率（FAR, False Alarm Rate）在系统处于正常状态时，系统错误发出故障警报的概率。具体形式如公式（6）所示：

$$FAR = \frac{X}{X+Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（6）$$

式中：

FAR ——故障漏报率；

X——实际没故障，但系统错误报警的次数；

Y——实际没故障，且系统没报警的次数。

6.2.7 准确率

准确率（Accuracy）衡量模型做出正确预测的样本占总样本的比例，具体形式如公式（7）所示。

$$Accuracy = \frac{X}{Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（7）$$

式中：

$Accuracy$ ——准确率；

X——被模型正确预测的正样本的样本数；

Y——总样本数。

6.2.8 精确率

精确率（Precision）也被称为查准率，衡量在所有被模型预测为故障的样本中，有多少是真正的故障。具体形式如公式（8）所示。

$$Precision = \frac{X}{X+Y} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（8）$$

式中：

$Precision$ ——精确率；

X——被模型正确预测的正样本的样本数；

Y——被模型错误预测的负样本的样本数。

6.2.9 召回率

召回率（Recall）也被称为查全率或者灵敏率，定义为在所有实际为正类的样本中，被成功找出的比例。具体形式如公式（9）所示。

$$Recall = \frac{X}{X+Y} \quad \cdots\cdots (9)$$

式中：

$Recall$ ——召回率；

X——被模型正确预测的正样本的样本数；

Y——被模型错误预测的正样本的样本数。

6.2.10 **漏检率**

漏检率（Missing Rate）是实际为正的样本里，被模型漏掉（判成负）的比例。具体形式如公式（10）所示。

$$MissingRate = \frac{X}{X+Y} \quad \cdots\cdots (10)$$

式中：

$MissingRate$ ——漏检率；

X——被模型错误预测的正样本的样本数；

Y——被模型正确预测的正样本的样本数。

6.2.11 **虚警率**

虚警率（False Alarm Rate）是实际为负的样本里，被模型误报（判成正）的比例。具体形式如公式（11）所示。

$$FalseAlarmRate = \frac{X}{X+Y} \quad \cdots\cdots (11)$$

式中：

FalseAlarmRate——虚警率；

X——模型错误预测的负样本的样本数；

Y——模型正确预测的负样本的样本数。

6.2.12 AU-ROC

AU-ROC是ROC曲线下方的面积。ROC曲线本身绘制的是在各种可能的分类阈值下，模型的真阳性率（TPR，即召回率）与假阳性率（FPR）之间的关系。AUC衡量的是模型将正负样本（故障/健康）区分开来的整体能力，且该能力独立于任何特定的分类阈值。

6.2.13 RMSE

RMSE是均方误差的平方根，衡量回归模型预测值与真实值之间的平均偏差大小。RMSE对模型的离群预测点非常敏感，具体形式如公式（12）所示：

$$S_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_{pred}(t_i) - Y_{true}(t_i)\right)^2} \quad \cdots\cdots (12)$$

式中：

S_{RMSE}——均方误差平方根；

n——样本数量；

$Y_{pred}(t_i)$——第i个时间点的预测值；

$Y_{true}(t_i)$——第i个时间点的真实值；

6.2.14 **决定系数**

衡量模型所解释的因变量方差占总方差的比例。它表明了模型的预测在多大程度上解释了真实结果的变化。具体形式如公式（13）所示：

$$S_{R2} = 1 - \frac{\sum(Y_{true} - Y_{pred})^2}{\sum(Y_{true} - Y_{mean})^2} \quad \cdots\cdots (13)$$

式中：

S_{R2}——决定系数；

Y_{pred}——预测值；

Y_{true}——真实值；

Y_{mean}——真实值均值。

其中，分子部分是模型预测误差的平方和（残差平方和），分母部分是总的平方和，即仅用均值进行预测时会产生的误差。

6.2.15 **激活值分布**

监控网络中关键层的神经元激活值的分布。分布的剧烈变化意味着模型可能正在进入一个不稳定的状态。

6.2.16 **健康神经元比例**

健康神经元比例是指在神经网络中，能够对输入数据产生有效响应（非恒定输出）的神经元占总神经元数量的比例。在长期运行或迁移学习过程中，部分神经元可能因梯度消失或激活函数特性而永久停止激活（死亡）或陷入饱和区。具体形式如公式（14）所示：

$$R_{health} = \frac{N_{active}}{N_{total}} \times 100\% \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（14）$$

式中：

R_{health}——健康神经元比例；

N_{total}——模型中神经元总数；

N_{active}——保持活性的神经元数量。

6.2.17 **数据漂移**

数据漂移是指用于训练机器学习模型的输入数据，其统计分布在模型部署后随时间发生显著变化，导致模型在生产环境中的预测性能下降的现象。本质属性是仅输入特征的概率分布P(X)发生变化，而输入与输出之间的条件关系 P(Y|X)保持不变。下面是常见用于监控数据漂移的相关指标：

a） KL散度（Kullback–Leibler Divergence），又称相对熵（Relative Entropy），是用一个概率分布来近似另一个真实分布时所损失的信息量。可用于量化基线训练数据和生产数据间的概率分布差异。当某个特征的KL散度超过了设定的阈值，即可判断发生了数据漂移。具体形式如公式（15）所示。

$$D_{KL}(P \parallel Q) = H(P,Q) - H(P) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（15）$$

式中：

$D_{KL}(P \parallel Q)$——分布P到分布Q的KL散度；

$H(P，Q)$ ——是分布P和分布Q之间的交叉熵；

$H(P)$——是分布P的熵。

b） 群体稳定性指数PSI：是一个常用于衡量两个群体分布稳定性的指标，它可以量化当前数据（实际分布）与某个基准（预期分布，通常是模型训练数据）之间的差异程度，它将分布差异转化为一个单一的数值，使得漂移程度可以被直观地比较和度量。PSI的计算基于对变量进行分箱。具体形式如公式（16）所示。

$$S_{PSI} = \sum_{i=1}^{k}(A_i - E_i) \cdot ln\left(\frac{A_i}{E_i}\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（16）$$

式中：

S_{PSI}——群体稳定性指数；

k ——为分箱数量；

E_i——为基准样本在第i 组的占比；

A_i ——为监控样本在第 i组的占比。

c） JS散度：JS散度是KL散度的一种改进形式。JS散度是对称的，并且其值总是有界的从而更具鲁棒性和可解释性。JS散度的值域在0到1之间：0表示分布完全相同，1表示分布完全不同，能有效地捕捉到两个概率分布之间的整体差异，使其对各种类型的分布变化都很敏感，能够检测到可能被其他指标忽略的细微漂移。具体形式如公式（17）所示。

$$D_{JS}(P \mid Q) = \frac{1}{2} D_{KL}(P \mid M) + \frac{1}{2} D_{KL}(Q \mid M) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（17）$$

式中：

$D_{JS}(P \mid Q)$——分布P和分布Q之间的JS散度；

M——两个待比较分布P和Q的平均分布。

6.2.18 **概念漂移**

概念漂移是指输入特征 X与输出标签 Y 之间的真实映射关系，即条件概率分布P(Y|X)，随时间或上下文发生了根本性的改变。这意味着模型在训练阶段学习到的判断规则在实际应用中已不再成立或已发生演变， 模型面临失效的风险。

常见的概念漂移检测方法如下：

a） 基于统计测试的检测方法：通过计算统计量（如均值、方差、偏度等），应用特定的统计检验（如KL散度或卡方检验等）；

b） 基于窗口的检测方法：将数据划分为多个固定大小或动态调整的窗口，并在每个窗口内单独计算统计量或模型性能。通过比较不同窗口的统计量或模型性能的变化来检测是否发生概念漂移；

c） 基于学习的检测方法：通过应用机器学习算法来学习数据的正常行为模式，并据此识别出异常或数据分布的变化。

6.2.19 **不确定性**

在模型评估阶段引入的不确定性量化，核心作用是将点预测扩展为区间或分布预测，从而提供对模型可靠性的直接度量。不确定性估计揭示模型在不同输入子空间中的可信程度，使得评估结果具备可解释性和可操作性。

6.2.20 **偶然不确定性**

是指由数据本身固有的、内在的随机性或噪声所引起的不确定性。它的根源在于用来描述一个系统的信息本身就不完整或含有噪声。主要来源有：

a） 传感器噪声与测量误差：智能系统中任何物理传感器都有其固有的精度限制和随机噪声；

b） 过程随机性：智能系统与外界交互的过程，以及智能系统内部的一些数据处理步骤是具有随机性的，会导致偶然不确定性。

6.2.21 **认知不确定性**

认知不确定性本质上是源于模型知识的缺乏。它反映了模型在其训练数据未能覆盖的区域进行预测时的不自信或困惑程度。通过提供更多、更具代表性的数据，或者改进模型结构，就可以让模型学习到更多知识，从而减少这种不确定性。认知不确定性的主要来源有：

a） 训练数据不足或不具备代表性：模型对其训练数据覆盖的区域很熟悉，但对数据稀疏或完全空白的OOD样本则一无所知；

b） 模型结构不当：选取的模型过于简单，无法捕捉数据中复杂的潜在关系，表达能力受限。

6.2.22 **响应时间**

智能系统处理输入并产生输出所需时间。

6.2.23 **吞吐量**

智能系统在单位时间内能处理的请求数量。

6.2.24 **资源利用率**

智能系统运行时消耗的计算资源以及硬件资源。

6.2.25 **健康指数**

健康指数是一个综合性的量化指标，通常是一个单一的数值，介于0和1之间，用于全面评估一个系统或设备的整体健康状况。通过融合多个状态特征量的信息，提供一个比单个指示物更全面、更鲁棒的健康评估。HI的计算方式并非一成不变。相同的状态特征量组合在不同的情景下动态调整权重获得不同的HI值。一种示例的计算方法如公式（18）所示。

$$HI_t = \frac{RUL_{(t)}}{RUL_{max}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（18）$$

式中：

HI_t——第t个循环时的健康指数；

RUL_t——在第t个循环时的剩余使用寿命；

RUL_{max}——系统在全新的状态下总设计寿命或在特定工作条件下的最大运行循环次数。

6.2.26 RUL预估

RUL 是智能系统PHM输出的核心预测结果之一，用于表示设备或部件从当前状态起到发生功能性失效之间的剩余时间。智能系统PHM在预测估计部分，必须对系统的RUL做出合理预估。

6.2.27 **相对准确度**

相对准确度是衡量预测误差相对于真实剩余使用寿命大小的指标。它将绝对误差转换为一个无量纲的百分比形式，从而能够公平地评估和比较在不同寿命尺度的设备上的预测性能。对于单个时间点t_i的预测，其相对准确度RA_i如公式（19）所示。

$$RA_i = \left(1 - \left| RUL_{true}(t_i) - \frac{RUL_{pred}(t_i)}{RUL_{true}(t_i)} \right| \right) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（19）$$

式中：

t_i——当前观测时刻；

$RUL_{true}(t_i)$——时间点t_i的剩余真实使用寿命；

$RUL_{pred}(t_i)$ —— 时间点t_i的预测剩余使用寿命。

6.2.28 **$\alpha-\lambda$性能**

$\alpha-\lambda$性能是一个更贴近实际应用需求的、基于“成功/失败”准则的评估度量。它定义一个围绕真实RUL的精度锥，并判断预测结果是否在要求的时间点进入并保持在这个锥形区域内。α边界定义了精度锥的宽度，表示允许的相对误差范围；λ时间点定义了评估的时间基准点。它是一个介于0和1之间的值，代表生命周期的某个百分比位置。对于给定的(α, λ)组合，预测是否合格的判断条件如公式（20）所示：

$$(1-\alpha)RUL_{true}(t_i) \le RUL_{pred}(t_i) \le (1+\alpha)RUL_{true}(t_i), \forall t_i > \lambda(t_{Eol}) \cdots\cdots\cdots（20）$$

式中：

t_i——当前观测时刻；

$RUL_{true}(t_i)$——时间点t_i的剩余真实使用寿命；

$RUL_{pred}(t_i)$ —— 时间点t_i的预测剩余使用寿命；

t_{Eol}——设备的总寿命（End-of-Life time）。

6.2.29 **预测有效期**

预测有效期用于衡量智能系统PHM模型能够多早地开始提供持续可靠的RUL预测。它定义为从预测误差首次进入并持续保持在预设的精度范围内的那个时间点起，到设备最终失效（End-of-Life， EoL）之间的时间跨度。

PH的计算步骤如下：

a） 确定t_{PH}：从设备的生命周期末端t_{Eol}向前回溯，找到最后一个使得预测RUL落在精度锥之外的时间点，这个时间点的下一个时间点$\mathrm{t_{PH}}$标志着预测开始持续可靠的时刻；

b） 计算PH：PH就是从 t_{PH}到t_{Eol}的时间。

6.2.30 收敛率

收敛率是一个衡量智能系统PHM模型预测趋向于真实RUL值的速度和稳定性的动态指标。它描述的是随着时间的推移，特别是当设备退化特征变得明显后，模型的预测误差的减小趋势。通常可以通过以下方式来量化：

a） 计算RUL预测误差的绝对值随时间变化的曲线的斜率；

b） 测量预测误差从一个较大的阈值降低到一个较小的阈值所花费的时间。

6.3 面向模型架构的定性要求和定量指标

6.3.1 大语言模型

大语言模型应用于多种任务中，所以用于评估大语言模型的性能指标涉及诸多方面，部分指标如下：

a） 功能正确性：功能正确性指的是模型生成的代码在给定输入下能否通过一组预定义的测试用例，并正确完成任务逻辑，用于大模型代码生成任务中；

b） F1分数：在大模型推理中，F1分数指对模型生成答案与参考答案之间推理链或推理结果是否匹配的综合度量，它把精确率与召回率折中成一个 0–1 之间的单值；

c） 一致性：在大模型的长上下文的场景下，一致性指的是模型在生成输出时，能够正确理解、记忆并整合分布在整个超长文本中所有相关信息的能力，确保其输出在事实、逻辑、角色和指令等多个维度上，与上下文中任何位置的信息都保持统一，不会出现矛盾或遗忘；

d） 困惑度：困惑度反映了语言理解问题中大模型对给定文本的预测能力。对于一个给定的序列，困惑度的计算公式如公式（21）所示：

$$\mathrm{PPL}(W) = \mathrm{P}(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} \qquad \cdots\cdots (21)$$

式中：

$\mathrm{PPL}(W)$——序列的困惑度；

$\mathrm{P}(w_1，\ w_2，\ \ldots，\ w_N)$——是序列的联合概率。

e） 幻觉率：幻觉率衡量大语言模型生成内容中包含非事实性错误或与给定上下文逻辑冲突的信息比例。具体形式如公式（22）所示：

$$R_{hal} = \frac{N_{hal}}{N_{total}} \qquad \cdots\cdots (22)$$

式中：

R_{hal}——幻觉率；

N_{hal}——被判定包含事实性错误、逻辑矛盾或无中生有信息的样本（或生成单元）数量；

N_{total}——参与评估的生成样本（或生成单元）总数。

f） 提示注入防御成功率：该指标用于评估模型抵御恶意提示攻击（如越狱指令、目标劫持、角色扮演攻击）的能力。它是衡量智能系统安全边界完整性的核心健康指标。具体形式如公式（23）所示：

$$R_{defense} = \frac{N_{blocked}}{N_{attack}} \qquad \cdots\cdots (23)$$

式中：

$R_{defense}$——提示注入防御成功率；

$N_{blocked}$——模型成功识别并拒绝执行恶意指令，或输出符合安全规范的响应次数；

N_{attack}——输入系统的恶意提示攻击样本总数。

6.3.2 强化学习智能系统

强化学习智能系统中关注的部分指标如下：

a）累计奖励：累积奖励衡量在一个完整的轨迹中，智能体获得的总奖励。通常计算平均累积奖励，即在大量测试轨迹上的回报均值。更高的平均回报通常意味着更好的性能。

b） 学习曲线：学习曲线是绘制智能体在训练过程中，累积奖励随训练步数（或轨迹数）变化的曲线，这条曲线可以揭示学习的速度、稳定性和最终性能；

c） 收敛速度：智能体需要多少训练步数或时间才能达到一个令人满意的性能水平。收敛速度越快，模型的学习效率越高。

6.3.3 联邦学习智能系统

联邦学习智能系统中关注的部分指标如下：

a） 泛化差异：用于评估联邦学习下模型在跨工况情况下的故障预测，具体形式如公式（24）所示：

$$GeneralizationGap = Loss_{unseen} - \frac{1}{N}\sum_{i=1}^{N} Loss_{i,train} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（24）$$

式中：

$GeneralizationGap$——泛化误差；

$\mathrm{Loss}_{\mathrm{unseen}}$——模型在未见过的新数据上的损失值，反映模型在未知场景下的预测性能；

$Loss_{i,train}$——第 i 个客户端在本地训练数据上的损失值。

b） 流程周期效率：用于评估联邦学习下模型优化过程中的有效时间占比，具体形式如公式（25）所示：

$$ProcessCycleEfficiency = \frac{T_{add}}{T_{total}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（25）$$

式中：

$ProcessCycleEfficiency$——流程周期效率；

T_{add}——增值时间，联邦学习中本地模型在边缘侧进行梯度计算和参数更新的实际耗时；

T_{total}——总周期时间，包含数据预处理、本地计算、网络传输延迟、服务端聚合以及节点等待的总时长。

c） 模型偏离度：衡量本地客户端模型参数更新方向与全局模型聚合方向的差异程度。用于诊断特定客户端是否存在数据异构性漂移或本地过拟合，具体形式如公式（26）所示：

$$D_i^{(t)} = \frac{||W_{global}^{(t)} - W_{local,i}^{(t+1)}||_2}{||W_{global}^{(t)}||_2} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（26）$$

式中：

$D_i^{(t)}$——第t轮通信中，客户端i的模型偏离度；

$W_{global}^{(t)}$——第t轮分发给客户端的全局模型参数向量；

$W_{local,i}^{(t+1)}$——客户端i经过本地训练后，准备上传的本地模型参数向量。

6.3.4 计算机视觉智能系统

计算机视觉智能系统中关注的部分指标如下：

a） 损坏误差：用于评估分类器在特定类型损坏视觉场景上的鲁棒性，考虑了不同损坏难度的差异。通过在特定扰动下的性能测试，评估系统对恶劣环境的抵抗力具体形式如公式（27）所示：

$$\mathrm{CE}_c^f = \frac{\sum_{s=1}^{N} E_{s,c}^f}{\sum_{s=1}^{N} E_{s,c}^{Baseline}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（27）$$

式中：

CE_c^f——分类器f在损坏类型c场景下的损坏误差；

$E_{s,c}^f$——分类器f在损坏类型c场景下，严重程度s（共有N个级别）下的Top-1错误率；

$E_{s,c}^{Baseline}$——基准分类器模型在相同损坏场景下的错误率。

b） 交并比：在计算机视觉的目标检测任务中，交并比定义为预测框与真实框的交集面积与并集面积之比。具体形式如公式（28）所示：

$$\mathrm{IoU} = \frac{|B_{\mathrm{pred}} \cap B_{\mathrm{gt}}|}{|B_{\mathrm{pred}} \cup B_{\mathrm{gt}}|} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（28）$$

式中：

IoU——交并比；

B_{pred}——预测框的面积，即模型输出的检测框的面积；

B_{gt}——人工标注目标框的面积。

c） 视频时空一致性误差：反映了系统在动态环境下的跟踪稳定性。高波动率通常预示着下游决策系统将发生故障。具体形式如公式（29）所示：

$$E_{flow} = ||M_t - Warp(M_{t-1}, F_{t-1\to t})|| \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots（29）$$

式中：

E_{flow}——视频时空一致性误差；

M_t——模型在当前时间t输出的掩码或检测结果；

M_{t-1}——模型在上一时间$t-1$输出的掩码或检测结果；

$F_{t-1\to t}$——描述像素点如何从时间$t-1$到时间t的运动向量场，由专门的光流算法计算；

$Warp$——变换函数。

7 复杂智能系统PHM应用对象范围

7.1 感知功能

在感知功能部分，智能系统通过传感器等硬件获取外界环境信息。此阶段智能系统PHM的主要对象是物理传感器和数据采集硬件。这包括但不限于摄像头、雷达、激光雷达、麦克风、温度传感器、压力传感器以及相关的信号处理电路等。适用范围如下：

a） 状态监测：实时监控传感器的性能参数，如精度、灵敏度、信噪比等，以确保数据输入的准确性和可靠性；

b） 故障诊断与预测：通过分析传感器输出数据的漂移、噪声增大或信号异常等现象，诊断现有故障并预测未来可能发生的性能退化或失效；

c） 健康管理：根据预测结果，为传感器的维护、校准或更换提供决策支持，从而保障整个智能系统信息输入的源头质量。

7.2 认知功能

认知功能负责处理和解释感知到的信息，进行理解、分析和学习。此阶段智能系统PHM的对象转向数据处理单元和算法模型。具体包括承载算法运行的处理器、存储器等硬件，以及算法本身的健康状态。适用范围如下：

a） 硬件健康管理：监测计算硬件的运行状态，如温度、功耗、内存使用率等，预测因硬件过热、老化等导致的计算性能下降或失效风险；

b） 算法性能监控：通过监控模型输出的准确率、稳定性和鲁棒性，评估算法是否因数据分布变化、模型过时或数据污染而出现性能退化；

c） 资源管理：基于对计算资源需求的预测，进行动态的资源调度与管理，确保认知任务的高效、稳定运行。

7.3 决策功能

智能系统决策功能负责基于认知结果，制定行动策略和方案。此阶段智能系统PHM关注的是决策支持系统和相关的软件、硬件平台。适用范围：

a） 决策系统可靠性保障：确保决策算法所在平台的稳定运行，防止因硬件故障导致决策中断或错误；

b） 决策质量评估：智能系统PHM提供的关于底层硬件和数据质量的健康评估信息，可以作为决策置信度的输入；

c） 风险管理：将系统自身的健康状态作为决策制定的一个重要考量因素。

7.4 控制功能

智能系统控制功能体现在根据决策指令，通过执行器来完成具体操作。适用对象：此阶段智能系统PHM的核心对象是执行器及其驱动系统。适用范围：

a） 执行器状态监控与预测：实时监测执行器的运行参数，诊断磨损、疲劳、老化等问题，并预测剩余执行器使用寿命；

b） 性能退化管理：在执行器出现性能退化但尚未完全失效时，智能系统PHM可以为控制算法提供信息，使其对控制指令进行补偿和调整，以维持系统整体的性能表现；

c） 视情维修：根据智能系统PHM对执行器健康状态的预测，制定视情维修计划，在部件真正失效前进行维修或更换，从而降低非计划停机带来的损失。

7.5 执行机构

智能系统的执行机构负责将决策指令转化为物理世界的实际动作。这些部件通常涉及机械运动、力传递和能量转换，是磨损、疲劳和失效的高发区域。智能系统PHM的适用对象是构成执行机构的各类物理组件，特别是那些对系统功能至关重要、容易发生性能退化或故障的部件。适用范围：

a） 通过设定阈值或利用机器学习算法，自动识别偏离正常运行状态的信号；

b） 判断执行机构故障的具体模式（如磨损、疲劳、裂纹、松动）并量化其严重程度；

c） 剩余可用时长预测：基于历史数据和物理模型，预测一个部件在需要更换前还能安全运行多长时间；

d） 预测关键性能指标（如效率、精度）随时间下降的趋势；

e） 量化在未来某个时间窗口内发生功能性故障的概率。

7.6 数据层面

在数据层面，智能系统PHM的目标是确保输入到模型的数据是高质量且一致的，因为数据是所有数据驱动模型的基础。适用对象包括用于模型训练和验证的静态数据集、模型在生产环境中接收到的实时动态数据流、负责数据采集、清洗、转换和传输的数据管道以及特征。适用范围如下：

a） 数据质量评估：持续监控数据的关键质量指标，如完整性（缺失值）、准确性（异常值或噪声）、一致性（格式或单位是否统一）和时效性；

b） 数据分布监测：跟踪关键特征的统计属性（如均值、方差、分布形态），以建立数据健康的基线；

c） 数据问题溯源：当监测到数据质量下降时，诊断问题的根源，例如是由于上游的传感器故障、数据采集错误还是数据处理管道中的缺陷；

d） 数据漂移检测：诊断并识别输入数据的分布是否发生了显著变化；

e） 预测数据质量退化：基于历史数据质量的变化趋势，预测未来可能出现的数据问题；

f） 预测数据漂移的发生：通过监测数据分布的微小、持续变化，预测未来发生显著数据漂移的可能性，为模型维护提供预警。

7.7 模型层面

在模型层面，智能系统PHM的目标是监控、诊断和预测模型本身的性能退化，确保其输出结果的可靠性，承载模型运行的软件和硬件基础设施。适用范围：

a） 性能指标：跟踪实时或定期监控模型的关键性能指标，如准确率、精确率、召回率、F1分数、均方根误差RMSE等；
b） 预测输出监控：监测模型预测结果的分布是否稳定，预测结果的漂移可以作为模型健康状态变化的早期信号。同时监控模型输出的公平性、不确定性和置信度，确保模型的输出处于高可信状态；
c） 概念漂移检测：诊断输入特征与目标变量之间的真实关系是否发生了变化；
d） 预测模型性能衰退：通过分析性能指标的下降趋势，预测模型性能将很快低于可接受阈值的时间点；
e） 预测剩余有效服务时间：预测一个模型在需要重新训练或更新之前，还能有效服务多长时间；
f） 主动维护建议：基于对数据漂移和概念漂移的监测与预测，主动触发模型再训练、更新或切换至备用模型报警；
g） 模型偏见监控：模型如果针对某一类特定输入在预测或决策过程中表现出系统性错误倾向，即模型偏见，同样代表模型处于非健康状态。

8 复杂智能系统PHM全生命周期过程

8.1 需求阶段

8.1.1 定义健康与可靠性指标

明确智能系统需要达到的性能、可靠性和可用性等关键指标，以定义模型精度的可接受下降范围和可容忍的故障发生频率。

8.1.2 确定健康监测数据需求

明确为了监测系统健康状态需要收集哪些数据，包括模型的输入输出数据、模型的置信度分数、系统资源使用率、以及用户的反馈数据等。

8.2 设计阶段

8.2.1 设计可监测的系统架构

设计支持健康数据的采集和传输的系统架构，用于状态因子的统计。

8.2.2 嵌入健康状态评估模块

设计用于实时或准实时评估系统健康状况的模块，用于评估各个阶段模型的健康状态。

8.2.3 设计故障注入和测试接口

为了在后续测试阶段验证智能系统PHM功能，需要预留用于模拟故障的接口和机制。

8.3 训练阶段

8.3.1 训练数据质量管理

对训练数据进行严格的清洗和标注，识别并处理可能导致模型的异常值、噪声和偏差，保证数据的质量。

8.3.2 监控训练状态

当智能系统训练过程中出现训练效果不佳，智能系统PHM应及时对训练进行干涉。

8.4 测试阶段

8.4.1 智能系统PHM功能验证

测试系统是否能按设计要求采集、处理和存储健康数据，以及健康状态评估模块的输出是否准确。

8.4.2 智能系统PHM性能验证

旨在通过全量健康数据注入与长期回溯比对，量化系统在不同工况下的健康评估精度、诊断延迟及误报率等性能，确认其满足设计指标。

8.4.3 故障模拟与诊断测试

通过预留的故障注入接口，模拟数据漂移、概念漂移、硬件资源受限等异常情况，测试系统能否准确地检测到异常、定位问题根源并发出警报。

8.4.4 系统验收与合格判定

在完成功能验证与性能验证后，需对智能系统 PHM 进行整体验收。验收测试应涵盖离线数据集回放与在线实时流测试，判定准则应综合考虑定性要求与定量指标。

8.5 运行阶段

8.5.1 实时状态监测

持续不断地监控系统各项状态指标。

8.5.2 异常检测与故障诊断

利用智能系统PHM算法自动检测偏离正常行为的异常模式。

8.6 维护与更新阶段

8.6.1 制定视情维护阶段策略

基于智能系统PHM系统对健康状态的评估和RUL的预测，来决定何时需要对模型进行再训练、更新或调整。

8.6.2 健康管理

利用智能系统PHM提出的建议进行健康管理，避免不必要的模型更新以及及时隔离故障。

8.7 退役阶段

当智能系统PHM持续预测到系统的维护成本将超过其产生的价值，或者其核心性能指标在多次更新后仍无法满足业务需求时，可以为系统退役提供数据支持。

9 复杂智能系统PHM支撑方法

9.1 知识驱动方法

9.1.1 专家系统

专家系统通过模拟人类专家的决策过程，并利用推理机进行逻辑推理，从而在智能系统中实现故障的自动化诊断与决策支持。这种方法的核心优势在于能够清晰地表达和利用专家经验，为复杂的诊断问题提供透明化的解决方案。

9.1.2 模糊逻辑

模糊逻辑通过引入隶属度函数的概念，有效处理智能系统监测数据中的不确定性与模糊性，使得系统能够模拟人类的近似推理能力。

9.1.3 贝叶斯网络

通过节点与有向边来表达变量间的因果关系和概率依赖，适用于智能系统PHM中的不确定性推理与故障诊断。它能够融合多源信息，在信息不完备的情况下进行概率预测，并能通过反向推理追溯故障的根本原因。

9.1.4 基于物理模型的故障机理建模

基于基础物理学原理建立设备健康状态的数学模型，从而精确描述系统性能退化的内在物理过程。

9.1.5 机理-参数化模型

失效物理（PoF）模型深入研究材料与组件在特定负载和环境应力下发生失效的根本物理化学过程，并建立描述其退化过程的参数化数学模型。

9.1.6 基于符号推理的诊断

该方法利用符号AI构建系统的结构、功能和行为模型，并通过知识图谱等技术将部件、故障模式及其因果关系显式地表达为符号网络。智能系统PHM通过在这个符号网络上进行逻辑推理，可以高效地隔离故障源头并解释诊断路径，尤其适用于具有复杂逻辑关联的系统。

9.1.7 故障树分析与事件树分析

故障树分析是一种自顶向下的演绎推理方法，通过逻辑门将系统顶层不期望的事件层层分解为底层基本事件的组合，用于识别故障的根本原因。而事件树分析则是自底向上的归纳方法，从一个初始事件出发，分析其可能引发的一系列后续事件序列及其后果。

9.1.8 失效模式与影响分析

FMECA通过逐一分析系统中各组成单元的潜在失效模式及其对系统功能的影响，并评估其严重度、发生率和探测度。在智能系统PHM中，其分析结果不仅指导了监测策略的制定，也为构建专家系统规则库和故障诊断知识图谱提供了宝贵的先验知识。

9.1.9 因果推理与关联分析

因果诊断技术通过构建结构因果模型或因果图谱，从统计关联中识别出具有物理意义的因果演化路径。该技术利用干预分析与反事实推理机制，通过将先验物理知识与观测数据中的因果发现算法相结合，为复杂智能系统的层级化故障隔离提供具备逻辑确定性的解释支撑。

9.2 数据驱动方法

9.2.1 时间序列分析

9.2.1.1 指数平滑

指数平滑通过分析历史数据点之间的依赖关系来预测未来的趋势。常被用于短期状态预测和构建动态阈值，当实际监测值偏离模型预测时即可发出早期预警。

9.2.1.2 小波变换/傅里叶变换特征提取

小波与傅里叶变换是将时间域信号转换到频率域或时频域的关键技术，实现从原始数据到关键故障特征提取。

9.2.2 回归分析

回归分析通过建立监测参数与设备健康状态或剩余寿命之间的数学映射关系，是智能系统PHM中进行性能退化建模和RUL预测的基础方法，其核心目标是量化退化趋势，为视情维修提供决策依据。

9.2.3 异常检测

9.2.3.1 统计异常检测

这类基于统计学原理的方法包括Z-Score、箱线图、PCA等，通过定义数据的“正常”分布范围来识别偏离常规模式的离群点，是实现早期故障检测的一线技术。它们计算简单、部署快速，非常适合在智能系统中对大量传感器数据进行实时监控，及时发现初始异常信号。

9.2.4 聚类方法

包括K-means、DBSCAN在内的聚类方法作为一种无监督学习方法，通过将数据点划分为不同的簇来识别系统运行的多种工况，而那些不属于任何正常工况簇的数据点则被视为异常。这种方法尤其适用于在缺乏故障标签的情况下，自动发现新的、未知的系统运行模式或故障类型。

9.2.5 一类分类器

一类分类器专门为数据不平衡场景设计，仅通过学习正常运行数据来构建一个正常边界，任何落在边界之外的数据都被判定为异常，可以很好地识别故障数据。

9.2.6 机器学习及深度学习方法

9.2.6.1 决策树

以决策树为基础的集成学习模型（如随机森林、XGBoost）是目前工业界应用最广泛、效果最好的机器学习方法之一，擅长处理复杂的非线性关系。它们在智能系统PHM中被广泛应用于高精度的故障诊断和剩余寿命预测。

9.2.6.2 支持向量机

支持向量机通过在特征空间中寻找一个最优超平面来区分不同类别的数据，尤其在高维特征空间中表现优异，泛化能力强，用于对提取出的复杂故障特征进行精准的模式识别。

9.2.6.3 自编码器

自编码器是一种无监督的神经网络，通过学习如何重构输入数据来自动提取数据中的核心特征，其重构误差可作为健康指标。

9.2.6.4 卷积神经网络

卷积神经网络CNN能够自动从原始信号中学习和提取具有空间层次性的特征，免去了繁琐的人工特征工程。

9.2.6.5 循环神经网络及其变体

循环神经网络（RNN）及其变体（LSTM， GRU）和Transformer模型是专为处理序列数据而设计的，能够有效捕捉数据在时间维度上的依赖关系和演变规律，通过学习设备从健康到失效的全生命周期数据，实现对未来退化趋势的精准预测。

9.2.6.6 注意力机制

注意力机制允许模型在处理信息时动态地聚焦于关键的部分，在智能系统PHM中，它不仅能提升模型在长时序预测中的性能，还能在融合多个传感器数据时，自动为更重要的传感器分配更高的权重，从而提高诊断的准确性和可解释性。

9.2.6.7 图神经网络

图神经网络将复杂系统抽象为由部件（节点）和它们之间的物理或逻辑连接（边）构成的图，从而在系统层面进行建模，用于分析故障在复杂系统拓扑结构中的传播路径，实现部件级与系统级的联合诊断与健康评估。

9.2.6.8 生成对抗网络

GAN通过两个网络的博弈来生成高度逼真的合成数据，有效解决了智能系统PHM领域中故障样本稀缺的难题。通过生成多样化的合成故障数据，GAN可以显著增强诊断模型的训练集，从而提升模型对罕见故障的识别能力和泛化性能。

9.2.7 可解释人工智能技术

可解释性技术提供模型输出结果与输入特征之间的逻辑透明度。该技术包括全局解释与局部解释两个维度：全局解释用于量化各传感器维度对系统健康评估的贡献度；局部解释则通过归因分析方法，揭示特定故障预警触发时的关键证据支持。

9.3 模型驱动方法

9.3.1 数字孪生方法

数字孪生为智能系统实体创建一个实时同步、高保真的虚拟副本，通过融合多物理场仿真、传感器数据、历史信息与模型内容，实现对设备全生命周期状态的精准映射与动态预测。

9.3.2 物理+数据混合建模方法

混合建模通过将基于物理机理的模型与强大的数据驱动模型深度融合，实现了优势互补，解决了单一方法的局限性。它利用物理模型提供先验知识和约束，同时借助机器学习模型从数据中学习未知或复杂的动态，从而构建出预测精度更高、可解释性更强且泛化能力更好的智能系统PHM模型。

9.3.3 虚实融合协同驱动技术

虚实融合技术通过物理实体与虚拟模型之间的双向实时数据交互，构建高保真的闭环演化环境。该技术利用物理空间获取的实时运行数据不断修正虚拟模型的边界参数，确保数字孪生体与实体的同步性；同时，利用虚拟空间的高仿真能力执行极端工况下的故障注入与应力仿真，弥补物理空间故障样本匮乏的问题。

9.4 多源信息融合技术

9.4.1 融合层级与总体框架

智能系统PHM中多源信息融合的目标是通过系统性的逻辑架构，将分布式的物理传感器信号与异构的数字逻辑数据转化为可追溯的、统一时空基准的健康证据链，利用多维冗余信息的集成显著降低系统健康监测的不确定性。按照融合体系划分为三级：

a） 数据级融合：该层级直接面向原始观测域，执行多模态数据的时空统一基准映射。通过信号调理、噪声抑制算法及基于统计推断的数据缺失补偿机制，消除数据采集过程中的瞬态扰动与非同步误差。

b） 状态级融合：利用多源协同估计理论对系统的隐藏健康状态进行实时反演。通过联合物理机理模型与多维度特征指标，在对齐后的数据上联合估计隐藏状态，在状态空间内执行非线性滤波或特征递归集成，并输出带协方差或置信度的健康状态向量；

c） 决策级融合：该层级处于PHM任务链的最顶层，负责对多个子系统、异构算法或独立监测源生成的局部评估结论进行高级别综合决策。通过引入证据理论、模糊逻辑或加权一致性算法执行决策冲突消解与可信度重构，为系统的维护干预与自主效能管理提供最终的执行依据。

9.4.2 数据级融合

数据级融合是多源信息融合体系的基础层，旨在将来自物理空间与逻辑空间的原始观测流转化为时空基准统一、物理意义一致且具备统计可信度的集成数据集。该过程的实施需遵循以下技术逻辑：

a） 观测对齐与同步：通过高精度时钟同步协议获取硬件级时间戳，并利用补偿算法，将异步采样的异构数据流重采样至同一时间离散序列。在空间维度，利用外参标定矩阵执行坐标变换，将数据映射至统一的系统本体坐标系中，消除因传感器布局差异引起的几何偏差。

b） 数据质量治理：在融合前需执行严格的预清洗工序，包括基于统计算法的离群点剔除或深度补全算法的缺失值恢复。针对传感器特有的噪声分布，利用自适应滤波器降低高频扰动，确保融合后的数据流具有较高的信噪比。

c） 多尺度冲突与量纲处理：针对不同物理意义和不同采样率的数据，融合逻辑需具备尺度感知能力。对于具备物理关联但量纲不一的特征，采用无量纲化处理实现数值层面的可比性。针对采样尺度差异巨大的数据源，需评估信息的混叠效应，必要时采用多分辨率分解方法，在不同频段或时间窗口内执行分层融合。

d） 互斥融合维度的识别处理：必须识别并处理具备物理互斥性的维度。当多源数据在语义层面存在根本冲突或在空间覆盖上完全不重叠时，系统需停止原始数据级的强制融合，转而执行冲突抑制逻辑。

e） 融合输出验证：融合后的数据集需经过统计一致性检验，确保集成后的数据流在时域与频域分布上符合系统的物理约束。

9.4.3 状态级融合

状态级融合通过在系统架构中并行运行一组具备不同噪声分布参数的候选模型库，通过引入贝叶斯模型概率评估机制，系统能够依据实时观测证据流对各局部候选模型的匹配度进行动态量化，并据此执行基于后验概率的加权集成逻辑。

该层级的最终产出为包含均值项与协方差矩阵的状态向量，该向量不仅高度集成了系统实时的量化健康指数，还深度涵盖了表征系统当前运行稳态的鲁棒性度量、统计不确定性水平以及剩余性能裕量等关键状态参数。

9.4.4 决策级融合

9.4.4.1 D-S证据理论

D-S（Dempster-Shafer）证据理论是一种处理不确定性信息的框架，它允许融合来自多个信息源（如不同算法或传感器）的、甚至可能是相互冲突的诊断证据，在智能诊断系统中，该理论能够量化各个证据的可信度与不确定度，并最终给出一个比任何单一来源都更可靠、更稳健的综合判断。

9.4.4.2 加权投票与贝叶斯模型平均技术

当各源输出为同一故障标签时，采用历史准确率或F1分数作为权重进行软投票；若输出为概率，则使用贝叶斯模型平均，将先验模型概率与似然乘积归一化，可证明在期望泛化误差意义下最优。

9.4.5 边缘计算技术

边缘计算实现了对设备的实时状态监测与快速响应，辅助需要瞬时决策的紧急故障保护和高频数据分析场景，是构建高效、敏捷的智能系统PHM的关键一环。

9.4.6 知识图谱技术

知识图谱技术将零散的数据中提取的特征模式、模型状态、历史输入输出日志等信息统一组织成一个结构化的语义网络。它在多源异构信息之间建立了清晰的关联。

参 考 文 献

[1] GB/T 7826-2012 系统可靠性分析技术 失效模式和影响分析（FMEA）程序

[2] GB/T 36344-2018 信息技术 数据质量评价指标

[3] GB/T 43782-2024 人工智能 机器学习系统技术要求

[4] GB/T 44442-2024 智能制造 远程运维系统 评价指标体系

[5] GB/T 44662-2024 健康管理 终端设备数据采集与传输协议

[6] GB/T 45087-2024 人工智能 服务器系统性能测试方法

[7] GB/T 45225-2025 人工智能 深度学习算法评估

[8] GB/T 45288.2-2025 人工智能 大模型 第2部分：评测指标与方法

[9] GB/T 45401.2-2025 人工智能 计算设备调度与协同 第2部分：分布式计算框架

[10] GB/T 45579-2025 机器人智能化视觉评价方法及等级划分

[11] GB/T 45907-2025 人工智能 服务能力成熟度评估

[12] GB/T 46315-2025 工业互联网平台 设备健康管理规范

[13] AIIA/T 0221-2025 MaaS模型即服务技术与应用要求 第7部分：大模型API服务性能测试方法

[14] US-SAE AIR7999 Condition monitoring and diagnostics of machine systems — Data processing，communication and presentation

[15] IEEE 1232-2010 IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments （AI-ESTATE）

[16] NIST NISTIR 8012 Standards Related to Prognostics and Health Management （PHM） for Manufacturing

[17] ISO 13374-4：2015 Condition monitoring and diagnostics of machine systems — Data processing，communication and presentation

[18] IEEE 1856-2017 IEEE Standard Framework for Prognostics and Health Management of Electronic Systems

[19] ISO/IEC TR 24027：2021 Information technology — Artificial intelligence （AI） — Bias in AI systems and AI aided decision making

[20] ISO/IEC 22989：2022 Information technology — Artificial intelligence — Artificial intelligence concepts and terminology

[21] ISO/IEC 23053：2022 Framework for Artificial Intelligence （AI） Systems Using Machine Learning （ML）

[22] ISO/IEC 5338：2023 Information technology — Artificial intelligence — AI system life cycle processes

[23] ISO/IEC TS 25058：2024 Systems and software engineering — Systems and software Quality Requirements and Evaluation （SQuaRE） — Guidance for quality evaluation of artificial intelligence （AI） systems

[24] IEEE 3110-2025 IEEE Standard for Computer Vision （CV）--Technical Requirements for Algorithms Application Programming Interfaces （APIs） of Deep Learning Framework

ICS 03.120.01
CCS V 00

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35007—2026

装备体系结构可靠性建模与预计

Architecture reliability modeling and prediction for equipment system of systems

2026-02-28 发布　　2026-02-28 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《标准化工作导则一第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：西北工业大学、无人飞行器技术全国重点实验室、北京航空航天大学、国防科技大学、中国运载火箭技术研究院战术军贸事业部、中国船舶集团有限公司综合技术经济研究院、中国航空综合技术研究院、中国空间技术研究院钱学森空间技术实验室、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）、西安现代控制技术研究所。

本文件主要起草人：陈志伟、李大庆、白光晗、张雨露、张罗庚、王凯旋、刘一萌、褚嘉运、王建峰、方晓彤、王宁、景永峰、马梓瑞、肖和业、林聪、王厚天、封雷、胡朗霄。

装备体系结构可靠性建模与预计

1 范围

本文件规定了开展装备体系结构可靠性建模与预计工作的基本原则、一般要求和详细要求。基本程序包括：装备体系结构与要素分析、装备体系结构可靠性数据收集、装备体系结构可靠性建模、装备体系结构可靠性预计、编制装备体系结构可靠性建模与预计报告，为装备体系结构可靠性建模与预计工作提供依据和指导。

本文件适用于装备体系结构可靠性建模与预计的相关工作。通过开展结构可靠性建模与预计工作，系统性识别制约装备体系结构可靠性水平的关键因素，给出装备体系结构可靠性水平的预计值，旨在支撑装备体系可靠性架构设计、资源配置优化及重构策略选择。具体建模与预计方法可结合不同装备体系特点进行适当剪裁。

2 引用文件

下列文件中的有关条款通过引用而成为本文件的条款。凡注明日期或版次的引用文件，其后的任何修改单（不包括勘误的内容）或修订版本都不适用本文件。然而，凡未注日期或版次的引用文件，其最新版本适用于本文件。

GJB 450B　　装备可靠性工作通用要求

GJB 451B　　可靠性维修性保障性术语

GJB 1909A　　装备可靠性维修性保障性要求论证

GJB 8113　　武器装备研制系统工程通用要求

3 术语和定义

GJB450B、GJB451B 确立的以及下列术语和定义适用于本文件。

3.1

体系 system of systems（SoS）

多个独立和有效系统的集合，通过共享资源与能力，构成一个提供独特功能的更大系统。

[来源：GJB 451B-2021，A.1.2]

3.2

装备体系 equipment system of systems（ESoS）

装备体系是根据任务需求、经济和技术能力，由一定数量和质量相互关联、功能互补的多种装备，按照装备的优化配置和提高整体作战能力的要求，综合集成的装备类别、结构和规模的有机整体。

3.3

装备体系结构可靠性 architecture reliability of ESoS

装备体系在规定的条件下和规定的时间内，保持其整体架构运行稳定并实现作战功能闭环的能力。

3.4

装备体系结构可靠度 degree of architecture reliability of ESoS

装备体系在规定的条件下和规定的时间内，保持其整体架构运行稳定并实现作战功能闭环的概率。

3.5

OODA 环 OODA loop

OODA（Observation、Orientation、Decision、Action）环指由具备探测、判断、决策、执行等功能的系统实现从探测、判断、决策到执行的一种闭环结构。

3.6

有效 OODA 环 effective OODA loop

装备体系在遭受内外部干扰（内部干扰指系统或设备自身的故障、失效；外部干扰指遭受人为的、有目的性的恶意攻击和环境条件变化）情况下仍可实现从探测、判断、决策到执行的一种闭环结构。

3.7

有效 OODA 网 effective OODA network

由装备体系中的有效 OODA 环交叉融合形成的有向网络结构。

4 基本原则

装备体系结构可靠性建模与预计工作应遵循以下原则：

a) 需求牵引

装备体系结构可靠性建模与预计工作应全面考虑装备体系自身功能稳定的需求，主要包括：

1) 应依据 GJB 8113 第 4 章中体系结构设计开展分层建模，明确装备体系拓扑结构、要素组成，以有序开展准确的结构可靠性建模与预计；
2) 在建模过程中，应充分考虑环境及装备状态的动态变化需求，确保模型能在装备体系结构调整、外部环境、要素性能退化等场景下预计装备体系结构可靠性。

b) 预防为主

装备体系结构可靠性建模与预计工作应遵循预防为主、早期介入的方针，主要包括：

1) 应定义在规定任务剖面下的数据采集规范，包括单装备可靠性数据（如故障率、修复率等），体系交互数据（如通信范围等），环境扰动数据（如蓄意攻击强度、区域参数等）。

c) 权衡协调

装备体系结构可靠性建模与预计工作应与体系自身特性及相关工作协调开展，主要包括：

1) 考虑到结构可靠性建模以构建包含异质节点要素的 OODA 环为目标，应明确装备类型及装备之间的作用顺序，明确装备之间的连接关系，以合理构建有效 OODA 网。

d) 演化重构

装备体系结构可靠性建模与预计工作应考虑装备体系动态演化性与可重构性，主要包括：

1) 考虑到任务、环境及装备状态的动态变化，结构可靠性建模期间应明确装备的退化机制以及外部环境的干扰机制，确保模型能反映装备体系在任务执行过程中的时变特性；
2) 考虑到单个装备失效情况以及新目标出现的可能性，结构可靠性建模期间应明确装备之间的替代可行性，规定 OODA 环重构的基本依据和具体策略，以实现可重构性。

e) 资源保障

装备体系结构可靠性建模与预计工作应配备必要的资源保障，主要包括：

1) 需求方与研制方应建立开放的信息交流制度，确保单装备可靠性数据、体系交互数据、环境扰动数据的及时共享与一致性校验；
2) 装备体系结构可靠性建模与预计工作应配备充分的人力和物力等资源，尤其应明确数据采集、模型建立与结构可靠性预计等专项资源需求，保障工作顺利开展。

5 一般要求

5.1 开展时机

一般在装备体系结构方案设计完成后开展结构可靠性建模与预计工作，并在研制阶段迭代开展。

开展装备体系结构可靠性建模与预计的阶段为：

a) 在装备体系结构方案设计通过技术评审后启动初步建模与预计；
b) 在工程研制等阶段动态开展，当出现以下情况时应重新预计：
   1) 体系结构变更影响关键要素；
   2) 环境威胁等级发生显著变化。

开展装备体系结构可靠性预计需达到以下前提条件：

a) 装备体系结构方案设计完成；
b) 装备体系结构状态基本固定；
c) 环境剖面确认；
d) 装备体系相关数据充分。

5.2 参数要求

依据 GJB 1909A 第 5 章装备可靠性维修性保障性定量要求、GJB 450B 第 4 章中可靠性参数、GJB 451B 第 5 章中可靠性术语参数等相关标准的规定，本文件明确了装备体系结构可靠性参数，一般包括单装可靠性、单装修复率、任务打击难度、任务失效判据、OODA 环重构时间等，参见表 A.1 和表 A.2。

应考虑可靠性参数作为开展装备体系可靠性设计与分析、试验与评价的基本依据。在装备体系的结构可靠性建模与预计阶段，通过结构可靠性建模和分析，预计结构可靠性参数值。

如果结构可靠性参数能达到目标值的要求，则可以保证装备体系在作战过程中多层次、多批次、多方向同时饱和攻击的战术需求；如果达不到维持基本作战能力的最低要求，则装备体系难以维持有效作战能力。

5.3 数据要求

为确保装备体系结构可靠性建模和预计工作的有效实施，在装备体系的结构可靠性建模与预计过程中，依照实际情况，明确一般数据要求：

a) 根据装备体系结构可靠性建模与预计过程需求，定义清晰的数据收集范围和指标；
b) 对输入数据统一数据格式，如日期、单位，以便于后续建模与预计过程实施；
c) 建立数据访问权限控制机制，实现对数据的安全管理。

6 详细要求

6.1 基本程序

装备体系结构可靠性建模与预计基本程序包括体系结构与要素分析、装备体系结构可靠性数据收集、装备体系结构可靠性建模、装备体系结构可靠性预计、编制装备体系结构可靠性预计报告。工作流程如图 1 所示。

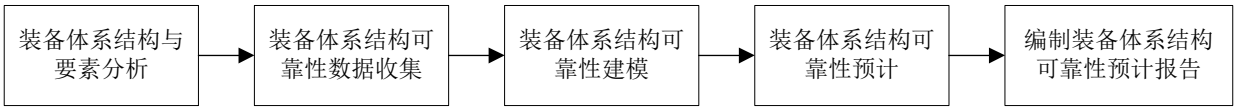

图1　结构可靠性建模与预计工作流程示意图

6.2　装备体系结构与要素分析

结合装备体系结构特点，确定装备体系结构层次划分方式，一般包括任务层、能力层与资源层。根据 OODA 环理论，对装备体系要素进行识别，一般包括探测类、决策类、打击类和通信类要素，参见附录 B。

6.3　装备体系结构可靠性数据收集

根据装备体系结构可靠性建模与预计需求，需要进行相应数据收集。一般包括：

a)　装备体系拓扑结构、要素组成数据；
b)　执行单元等各类要素的类型及要素数量；
c)　要素失效/扰动因素分析表，包括蓄意攻击、区域干扰等；
d)　装备体系中单个装备的可靠性参数，如可靠度、失效率、修复率等；
e)　装备体系外部干扰模式与强度等。

6.4　装备体系结构可靠性建模

结合装备体系所属装备、任务环境等特点，体系结构可靠性建模方法具体包括：确定装备体系要素模型、确定装备体系连边模型、确定有效 OODA 网模型。OODA 环是一种闭环结构，在现实战场中，装备体系通常需要多个 OODA 环交互耦合共同完成作战任务。因此，将装备体系有效 OODA 环数量作为衡量其结构可靠性的量化指标，通过将有效 OODA 环数量与要求阈值进行量化分析，进而建立装备体系结构可靠性模型，方法要点如下：

6.4.1　确定装备体系要素模型

应充分考虑装备体系要素异质性，通过分析要素属性和失效模式建立要素模型。装备体系要素类型一般包括探测类要素、决策类要素和执行类要素，要素属性涵盖要素类型、要素数量、位置坐标和性能等。要素建模具体过程参见附录 C.1。

6.4.2　确定装备体系连边模型

应充分考虑装备体系要素之间的数据传输和任务分配策略，建立装备体系连边模型。考虑到装备体系中的各类要素表现出不同的连接模式，建立从探测到决策、决策到执行等类型的连边模型，以形成装备体系 OODA 网络。连边建模具体过程参见附录 C.2。

6.4.3　确定有效 OODA 网模型

基于已构建的装备体系要素模型和连边模型，建立装备体系有效 OODA 网模型。在任务过程中，装备体系通常需要多个 OODA 环交互耦合，共同完成作战任务。在考虑内外部干扰因素对于 OODA 环要素与连边的影响，并融入动态重构策略提出有效 OODA 环模型。一个 OODA 环中的要素也可能出现在其他 OODA 环中，而且同类要素之间可以进行信息融合与资源共享。因此，OODA 环通过共享相似要素进行协作，从而形成有效 OODA 网。有效 OODA 网建模过程参见附录 C.3。

6.4.4　确定结构可靠性模型

根据装备体系结构与有效 OODA 网模型，建立装备体系结构可靠性模型和量化算法，具体过程参见附录 C.4。

6.5 装备体系结构可靠性预计

结合装备体系任务需求和相关数据，装备体系结构可靠性预计方法包括：确定有效 OODA 计算方法、确定装备体系重构策略、确定结构可靠度计算方法。方法要点如下：

6.5.1 确定有效 OODA 环计算方法

考虑到要素协作性，OODA 环通过交叉融合形成的异质有向网络结构，通过共享资源和整合信息来实现任务目标，有效 OODA 环的计算方法步骤具体包括：转移矩阵计算、邻接矩阵计算、要素存在概率、连边存在概率等。计算有效 OODA 环具体过程参见附录 D.1。

6.5.2 确定装备体系重构策略

应充分考虑不同类型要素在随机失效和蓄意攻击下的失效模式分析，确定三种重构策略：团簇/平台间重构、团簇/平台内重构和要素维修或生成，以快速提升体系杀伤能力。具体过程参见附录 D.2。

6.5.3 确定结构可靠度计算方法

结合邻接矩阵和到达矩阵，计算装备体系有效 OODA 环数量，从而进一步给出体系结构可靠度量化方法，参见附录 D.3，明确装备体系结构可靠度算法，算法伪代码参见表 D.1。

6.6 编制装备体系结构可靠性预计报告

装备体系结构可靠性预计完成后应编制预计报告，报告主要包含：预计目的、预计内容、预计所依据的标准号、装备体系结构可靠性相关定义、装备体系结构与要素、结构可靠性建模、结构可靠性预计结果、结论分析、问题与建议等。

附 录 A

（资料性）

结构可靠性数据需求及指标体系构建

应充分填写体系预计需求清单，以完成结构可靠性数据收集，需求清单具体如下。

表A.1 装备体系结构可靠性建模与预计需求清单

需求类别	具体需求	备注
作战任务需求信息	任务目标的探测难度、执行难度等	
	不同任务阶段体系的任务失败判据	
体系架构信息	装备体系的层级架构、任务执行类型、要素类型、要素数量等	
	要素类型（探测装备、决策装备、执行装备）、各类要素中装备的组成情况、要素初始连接情况（拓扑架构）	
	要素性能退化过程、要素随机失效类型、要素失效率	
	装备体系结构可靠性建模与预计指标体系（表 A.2）	
	要素修复率	
重构过程信息	任务执行单元内同类要素之间的重构、不同任务执行单元之间要素的重构、通过修复或新增要素节点进行重构	

表A.2 装备体系结构可靠性建模与预计指标体系

	指标级	评价指标
装备体系结构可靠性建模与预计指标	结构可靠性	有效 OODA 环数量等
	探测类要素	可靠度、修复率、探测距离、探测概率、位置坐标、通信距离、探测任务分配、漏警率等
	决策类要素	可靠度、修复率、指令响应时间、信息处理时间、位置坐标、通信距离、准确性、网络传输延时等
	执行类要素	可靠度、修复率、打击精度、执行任务分配、位置坐标、通信距离等
	通信类要素	空中/水面/水下传输距离、通信可靠度、数据丢包率等

附 录 B

（资料性）

装备体系结构与要素分析

B.1 装备体系结构及要素

在装备体系中，每个团簇或平台都需要执行一组任务（目标或子目标）以实现体系总体任务，依据装备体系组成结构关系，其结构层次划分如图 B.1 所示。

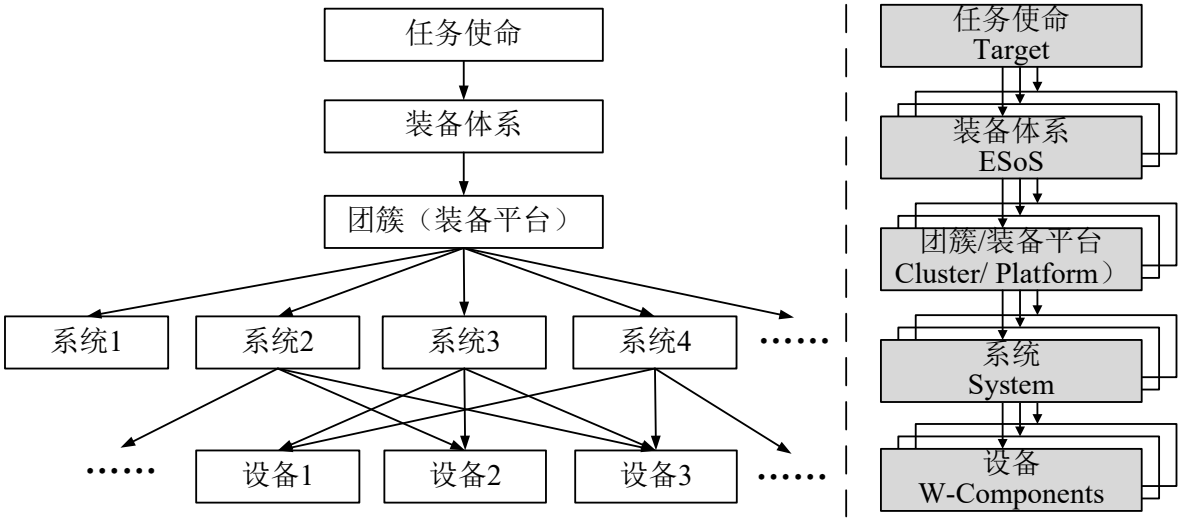

图 B.1 装备体系物理资源层次结构划分

依据装备体系结构和 OODA 理论，在装备设计、运行及其执行任务过程等各个阶段，按照影响装备体系的关键能力得到影响装备体系的四个要素：

a) 探测类要素

探测类要素指装备体系中用于感知、发现、识别和跟踪目标的各类传感器、侦察平台及相关信息处理能力的统称。在装备体系任务执行过程中，探测类要素肩负发现目标、识别目标信息等重要职责。

b) 决策类要素

在装备体系任务执行过程中，决策类要素肩负处理目标信息、下达指令等重要职责，是影响装备体系规划、形成与运用的关键因素。

c) 执行类要素

执行类要素构成装备体系的行动终端，承担将决策转化为实际效能的职责。该要素通过多样化的行动手段（包括火力压制、电子干扰、信息对抗等），在物理域或信息域实现任务目标。不同装备搭载的专用任务载荷与控制系统，赋予体系差异化的任务执行能力。作为装备体系能力输出的最终环节，执行类要素的性能直接决定任务链构建的实际效果。

d) 通信类要素

通信类要素是指装备体系中用于实现信息传输、交换与共享的通信设备、网络架构、协议标准的统称。在装备体系执行任务的过程中，通信系统肩负联合组网、信息传递等重要职责。

装备体系要素是作战力量的主要源泉，上述 4 类关键要素是实现作战活动 OODA 过程的基础，其他辅助系统对装备体系效能的影响程度较低，将探测类、决策类、执行类和通信类确定为装备体系要素，并对其进行建模与分析。同时，通信网络是实现各物理资源之间“资源与信息共享”的前提与基础，本文件将要素之间的通信等效为链路。装备体系要素和链路类型如表 B.1 所示。

表 B.1　装备体系要素和链路类型

装备体系要素和链路类型	要素功能
探测类要素（S）	对目标探测，获取任务目标情报，感知任务环境
决策类要素（D）	分析处理战场信息与态势，指挥与决策
执行类要素（W）	对敌方目标进行精确地打击和电子干扰等
通讯链路（E）	连接通信节点、用于传输信息的物理或逻辑通路

B. 2　装备体系拓扑分析

例如，某装备体系拓扑结构组成如图 B.2 所示，其主要组成包括若干小型无人机、小型无人艇、中大型无人机、中型无人艇、大型无人艇、岸基指挥中心等。

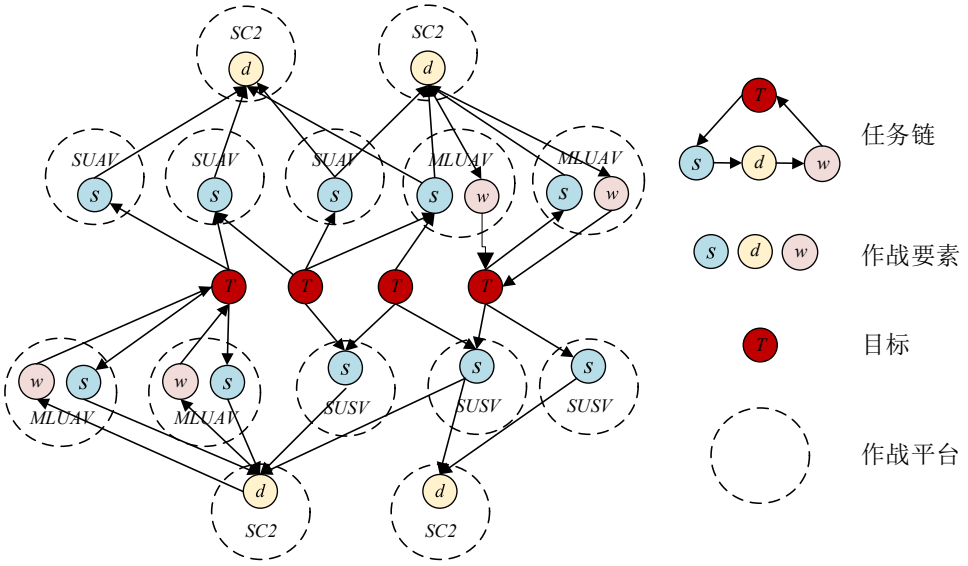

图 B.2　某装备体系拓扑结构

附 录 C

（资料性）

装备体系结构可靠性建模方法

C.1 要素模型

在装备体系化的背景下，分析要素的属性并考虑可用资源的限制是至关重要的。装备体系各要素都具有其特定属性，并受到其功能性能约束。通过考虑装备要素功能性能及其约束，可实现对装备体系能力的定量与定性分析。

在理想状态下，装备体系各要素可以相互连接，形成一个全连通的网络。在任务场景中，不同类型的要素会受到功能性能属性和作战资源约束。本文件定义 $s_i(i=1,2,...,I)$ 表示第 i 个探测要素，$d_j(i=1,2,...,J)$ 表示第 j 个决策要素，$w_m(m=1,2,...,M)$ 表示第 m 个执行要素，I,J,M 表示不同类型要素的数量；e_{s_i,d_j} 和 e_{d_j,w_m} 分别表示要素 s_i 到 d_j 和 d_j 到 w_m 的关联关系。基于 OODA 环的装备体系组成结构如图 C.1 所示。

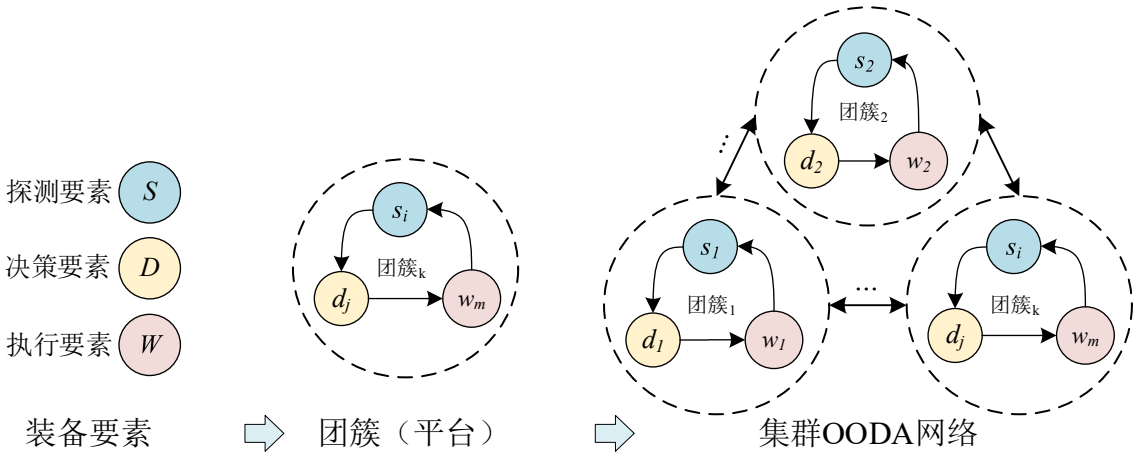

图 C.1 基于 OODA 环的装备体系组成结构

C.1.1 要素失效模型

针对要素失效建模，首先，无人装备在任务过程中会发生自然退化和随机失效。此外，在任务过程中要素或团簇容易遭受各种类型的外部冲击和干扰，如病毒、电磁冲击和打击等，不同冲击类型对不同要素造成的损伤程度也不相同，且会同时影响到体系中的部分要素或全部要素。同时，同平台或者团簇内的要素具有关联关系会产生共因失效。因此，需要同时考虑要素的上述失效模式建立要素失效模型。

随机失效以不可预测的方式发生，导致要素从体系中移除。考虑到装备固有的可靠性，采用指数分布或泊松分布结合蒙特卡罗方法来准确描述其失效行为。要素 s_i,d_j,w_m 的失效或消亡由参数为 $\lambda_i^s,\lambda_j^d,\lambda_m^w$ 的泊松分布来描述。要素 s_i,d_j,w_m 的修复或生成由参数为 μ_i^s,μ_j^d,μ_m^w 的指数或泊松分布等描述。

蓄意攻击是指对系统或其组件造成伤害或破坏的蓄意行为，该类型攻击一般都有特定的目标和策略。其中，最常见的蓄意攻击策略包含最大度攻击策略、最大介数攻击策略和树形攻击策略等。最大度攻击策略和最大介数攻击策略是分别根据节点要素的度和介数中心性从高到低排序的要素移除策略，树形攻击策略主要应用于网络病毒入侵等类型的外部干扰分析。本文件以最大度攻击策略为例分析其对体系的影响，设 k_i^s,k_j^d,k_m^w 表示要素 s_i,d_j,w_m 的度。

C.1.2 要素属性分析

对于探测类要素 S，其主要是雷达、侦查无人机等装备。由于侦查探测装备的型号不同，其可靠性也会有较大的差距。探测类要素 S 的属性模型建立如下：

$$s_i = (I, \lambda_i^s, \mu_i^s, k_i^s, cluster_k, \cdots) \quad \text{(C.1)}$$

其中，I 表示探测要素的数量，λ_i^s 表示探测要素故障率，μ_i^s 表示探测要素修复率，k_i^s 表示探测要素节点度值，$cluster_k$ 表示要素 s_i 是团簇 k 的成员。

对于决策类要素 D，其主要包括指挥决策系统和地面站等组成。决策类要素 D 的属性模型建立如下：

$$d_j = (J, \lambda_j^d, \mu_j^d, k_j^d, cluster_k, \cdots) \quad \text{(C.2)}$$

其中，J 表示决策要素的数量，λ_j^d 表示决策要素故障率，μ_j^d 表示决策要素修复率，k_j^d 表示决策要素节点度值，$cluster_k$ 表示 d_j 是团簇 k 的成员。

对于执行类要素 W，主要包括导弹、巡飞弹等。执行类要素 W 的属性模型建立如下：

$$w_m = (M, \lambda_m^w, \mu_m^w, k_m^w, cluster_k, \cdots) \quad \text{(C.3)}$$

其中，M 表示执行要素的数量，λ_m^w 表示执行要素故障率，μ_m^w 表示执行要素修复率，k_m^w 表示执行要素节点度值，$cluster_k$ 表示 W_{m} 是团簇 k 的成员。

C.2 连边模型

装备体系连边代表要素之间的通信和关联关系。连边承载的信息促进了体系内要素之间的数据传输、命令和任务分配。边的存在表明其所连接的要素之间存在直接联系和交互，其在实现体系要素之间的协调行动、协作和信息交换方面发挥着至关重要的作用，最终提升装备体系整体作战效能。体系要素之间通过有线局域网或无线数据链连接，要素间的通信表现为有向关系，将团簇内和团簇间的通信定义为具有不同权重的有向边，给出基于加权有向图的装备体系连边模型。

首先，给出基于 OODA 环的加权有向边模型：

$$E = \{e_{s_i,d_j}, e_{d_j,w_m}\} \quad \text{(C.4)}$$

连边 e_{s_i,d_j} 传输侦察任务指令，考虑了要素 s_i 和 d_j 之间的通信距离和可靠性。e_{s_i,d_j} 模型建立如下：

$$e_{s_i,d_j}(d_{s_i,d_j}, R^c_{s_i,d_j}) \quad \text{(C.5)}$$

其中，d_{s_i,d_j} 和 $R^c_{s_i,d_j}$ 分别为 s_i 和 d_j 的通信距离和可靠性。

连边 e_{d_j,w_m} 传输作战任务指令，其中考虑了要素 d_j 和 w_m 之间的通信距离和可靠性。e_{d_j,w_m} 模型建立如下：

$$e_{d_j,w_m}(d_{d_j,w_m}, R^c_{d_j,w_m}) \quad \text{(C.6)}$$

其中，d_{d_j,w_m} 和 $R^c_{d_j,w_m}$ 分别为 d_j 和 w_m 的通信距离和可靠性。

C.3 OODA网模型

装备体系组成系统和关联关系存在差异，本节采用异质网络模型建立有效 OODA 网模型，以赋予不同节点要素和连边实际意义。异质有向网络的定义如下：

异质有向网络：给定一个有向图 $D = (V, E, \varphi, \psi)$，该有向图有一个节点类型映射函数为 $\varphi(V) \to \xi$，其中 $v \in V$ 属于特定的节点类型 $\varphi(v) \in \xi$。图中每条边 e 都由一对有序节点 (u, v) 表示方向，其中 u 是起点，v 是终点。边类型映射函数为 $\psi : E \to \zeta$，其中每条边 $e \in E$ 都属于一个特定的关系 $\psi(e) \in \zeta$。若图的节点

类型$|\xi|>1$或边类型$|\zeta|>1$，则该网络模型为是异质有向网络。

本文件中，装备体系由不同类型系统组成，其中$V=(S,D,W)$，$E=\{e_{s_i,d_j},e_{d_j,w_m}\}$，其中$|\xi|=4$，$|\zeta|=4$，$S=\{s_1,s_2,...,s_I\}$，$D=\{d_1,d_2,...,d_J\}$，$W=\{w_1,w_2,...,w_m\}$。连边$e_{s_i,d_j}\in E$意味着从$s_i$到$d_j$的信息传输。

OODA 环是一种闭环结构，在实际任务过程中，装备体系通常需要多个 OODA 环交互耦合，共同完成作战任务。本节考虑内外部干扰因素对于 OODA 环节点要素与连边的影响，并融入动态重构策略提出有效 OODA 环模型。一个 OODA 环中的要素也可能出现在其他 OODA 环中，而且同类要素之间可以进行信息融合与资源共享。因此，OODA 环通过共享相似要素进行协作，从而形成如图 C.2 所示的有效 OODA 网。本文件给出有效 OODA 网模型（effective OODA network model）定义如下：

有效 OODA 网模型是通过对传统 OODA 环模型进行扩展，纳入了要素随机失效、蓄意攻击和体系动态重构等因素对 OODA 环完成任务实际效果的影响，装备体系中的线性 OODA 环交叉融合形成的异质有向网络结构，体系中的组成系统通过共享资源和整合信息来实现任务目标。有效 OODA 网络模型可用一个集合表示：

$$eOODA\_network=\{\mathrm{A},V,E\} \quad \text{.....(C.7)}$$

其中，$A=\{\mathbf{A_{SD}},\mathbf{A_{DW}}\}$表示描述不同要素之间连接的邻接矩阵集合，$V=(S,D,W)$表示装备体系中的要素，$E=\{e_{s_i,d_j},e_{d_j,w_m}\}$表示要素间的关联关系。例如，要素 S 和 D 的邻接矩阵如下所示：

$$\mathbf{A_{SD}}=\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \\ \cdots \\ s_I \end{array}\begin{array}{c} \begin{array}{ccccc} d_1 & d_2 & d_3 & \cdots & d_J \end{array} \\ \begin{bmatrix} x_{s_1,d_1} & x_{s_1,d_2} & x_{s_1,d_3} & \cdots & x_{s_1,d_J} \\ x_{s_2,d_1} & x_{s_2,d_2} & x_{s_2,d_3} & \cdots & x_{s_2,d_J} \\ x_{s_3,d_1} & x_{s_3,d_2} & x_{s_3,d_3} & \cdots & x_{s_3,d_J} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{s_I,d_1} & x_{s_I,d_2} & x_{s_I,d_3} & \cdots & x_{s_I,d_J} \end{bmatrix} \end{array} \quad \text{.....(C.8)}$$

其中，$x_{s_i,d_j}\,(i=1,2,...,I;j=1,2,...,J)$是邻接矩阵$\mathbf{A_{SD}}$的元素。

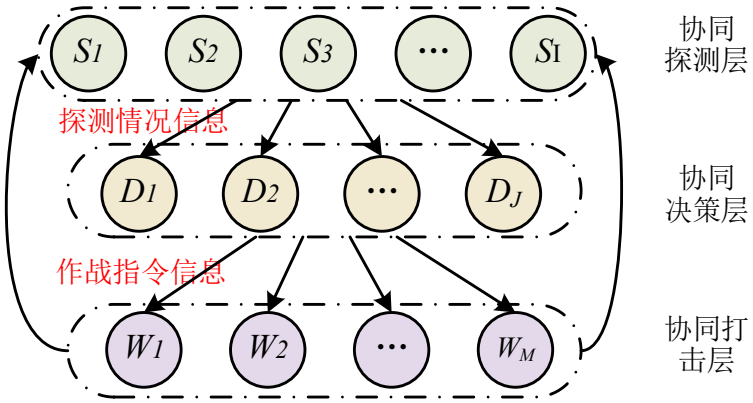

图 C.2　装备体系有效 OODA 网络模型

C.4　结构可靠性模型

装备体系结构可靠性模型为在规定的条件下和规定的时间内，装备体系在 t 时刻的有效 OODA 环数量与装备体系内 OODA 环总数之比，用式 C.9 表示：

$$R_{ESoS}(t)=\frac{N_{OODA}(0)-r_{OODA}(t)}{N_{OODA}(0)} \quad \text{.....(C.9)}$$

其中，$R_{ESoS}(t)$ 为在 t 时刻的装备体系结构可靠性；$N_{OODA}(0)$ 为装备体系初始具备的 OODA 环总数；$r_{OODA}(t)$ 为任务到 t 时刻 OODA 环断链数。

附 录 D

（资料性）

装备体系结构可靠性预计方法

D.1 有效OODA计算

本节利用转移矩阵计算装备体系的有效 OODA 环数量，具体计算步骤如下：

转移矩阵 $\mathbf{A}=\mathbf{A_{SD}}\bullet\mathbf{A_{DW}}$ 为装备体系异质有向图的转移矩阵（ $S\to D\to W$ ）。如果要素 s_i,d_j,w_m 存在，且要素之间连通，元素 $x_{s_i,d_j}=x_{d_j,w_m}=1$，若要素之间不连通，元素 $x_{s_i,d_j}=x_{d_j,w_m}=0$。可得 x_{s_i,d_j},x_{d_j,w_m} 的值为要素和连边示性函数的乘积，如下式所示：

$$\begin{cases}x_{s_i,d_j}=\alpha(s_i)\times\alpha(d_j)\times\alpha(e_{s_i,d_j})\\x_{d_j,w_m}=\alpha(d_j)\times\alpha(w_n)\times\alpha(e_{d_j,w_m})\end{cases}\quad\text{.....(D.1)}$$

其中，$\alpha(\bullet)$ 表示每个要素和连边的存在性。

要素 S 与 W 的邻接矩阵为 $\mathbf{A_{SW}}$：相邻的转移矩阵 $\mathbf{A_{SD}}$ 和 $\mathbf{A_{DW}}$ 被定义为 $\mathbf{A_{SD}}$ 的到达要素类型与 $\mathbf{A_{DW}}$ 的起始要素类型相匹配的矩阵，通过将相邻的转移矩阵相乘得到。$\mathbf{A_{SW}}$ 提供了有向图从一种要素类型转换到另一种要素类型的信息，可得 OODA 环数量为：

$$N_{OODA}=\sum_{j=1}^{J}x_{s_i,d_j}\times x_{d_j,w_m}\quad\text{.....(D.2)}$$

其中，N_{OODA} 为装备体系中有效 OODA 环的数量，$\mathbf{A_{SD}}$ 和 $\mathbf{A_{DW}}$ 的维数分别为 $I\times J$ 和 $J\times M$。进一步，t 时刻装备体系的有效 OODA 环数量 $N_{eOODA}(t)$ 计算如下：

在要素失效和生成条件约束下，各要素存在概率为：

$$\begin{cases}p(s_i)=1-F_i^s(t_{\lambda_i^s})+F_i^s(t_{\lambda_i^s})\times G_i^s(t_{\mu_i^s})\\p(d_j)=1-F_j^d(t_{\lambda_j^d})+F_j^d(t_{\lambda_j^d})\times G_j^d(t_{\mu_j^d})\\p(w_m)=1-F_m^w(t_{\lambda_m^w})+F_m^w(t_{\lambda_m^w})\times G_m^w(t_{\mu_m^w})\end{cases}\quad\text{.....(D.3)}$$

若要素的失效或生成服从指数分布，即有：

$$\begin{cases}p(s_i)=1-exp(-\lambda_i^s\times t_{\lambda_i^s})+exp(-\lambda_i^s\times t_{\lambda_i^s})\times exp(-\mu_i^s\times t_{\mu_i^s})\\p(d_j)=1-exp(-\lambda_j^d\times t_{\lambda_j^d})+exp(-\lambda_j^d\times t_{\lambda_j^d})\times exp(-\mu_j^d\times t_{\mu_j^d})\\p(w_m)=1-exp(-\lambda_m^w\times t_{\lambda_m^w})+exp(-\lambda_m^w\times t_{\lambda_m^w})\times exp(-\mu_m^w\times t_{\lambda_m^w})\end{cases}\quad\text{.....(D.4)}$$

若要素的失效或生成服从指数分布，且考虑要素最大度攻击失效，即有：

$$\begin{cases}p(s_i)=[1-exp(-\lambda_i^s\times t_{\lambda_i^s})+exp(-\lambda_i^s\times t_{\lambda_i^s})\times exp(-\mu_i^s\times t_{\mu_i^s})]\times\alpha(k_i^s)\\p(d_j)=[1-exp(-\lambda_j^d\times t_{\lambda_j^d})+exp(-\lambda_j^d\times t_{\lambda_j^d})\times exp(-\mu_j^d\times t_{\mu_j^d})]\times\alpha(k_j^d)\\p(w_m)=[1-exp(-\lambda_m^w\times t_{\lambda_m^w})+exp(-\lambda_m^w\times t_{\lambda_m^w})\times exp(-\mu_m^w\times t_{\lambda_m^w})]\times\alpha(k_m^w)\end{cases}\quad\text{.....(D.5)}$$

其中，$F_i^s(t_{\lambda_i^s})$，$F_j^d(t_{\lambda_j^d})$ 和 $F_m^w(t_{\lambda_m^w})$ 表示自各要素上次修复以来时间的故障累积分布函数；$G_i^s(t_{\mu_i^s})$，$G_j^d(t_{\mu_j^d})$ 和 $G_m^w(t_{\mu_m^w})$ 表示失效要素自故障以来时间的修复或要素生成累积分布函数；$t_{\lambda_i^s}$，$t_{\lambda_j^d}$ 和 $t_{\lambda_m^w}$ 表示各要素上次修复后的时间；$t_{\mu_i^s}$，$t_{\mu_j^d}$ 和 $t_{\mu_m^w}$ 表示各要素上次故障后的时间；$\alpha(k_i^s)$，$\alpha(k_j^d)$ 和 $\alpha(k_m^w)$ 表示在最

大度攻击下要素是否被移除的示性函数。

装备体系有效 OODA 环中通信链路的连通概率为：

$$\begin{cases} p(e_{s_i,d_j}) = \alpha(cd_{s_i,d_j}) \times R^c_{s_i,d_j} \\ p(e_{d_j,w_m}) = \alpha(cd_{d_j,w_m}) \times R^c_{d_j,w_m} \end{cases} \quad \text{.....(D.6)}$$

其中，$R^c_{s_i,d_j}$ 和 $R^c_{d_j,w_m}$ 为通信系统可靠性，由其分布函数计算得到；$\alpha(cd_{s_i,d_j})$ 和 $\alpha(cd_{d_j,w_m})$ 表示要素之间的距离在有效通信范围内，由示性函数表示如下：

$$\alpha(cd_{s_i,d_j}) = \begin{cases} 1 & cd_{s_i,d_j} \le \min(cd^{s_i}_{\max}, cd^{d_j}_{\max}) \\ 0 & cd_{s_i,d_j} > \min(cd^{s_i}_{\max}, cd^{d_j}_{\max}) \end{cases} \quad \text{.....(D.7)}$$

$$\alpha\left(cd_{d_j,w_m}\right) = \begin{cases} 1 & cd_{d_j,w_m} \le \min\left(cd^{d_j}_{\max}, cd^{w_m}_{\max}\right) \\ 0 & cd_{d_j,w_m} > \min\left(cd^{d_j}_{\max}, cd^{w_m}_{\max}\right) \end{cases} \quad \text{.....(D.8)}$$

D.2 重构策略

体系动态重构策略旨在提高装备体系在动态环境中的适应能力和任务成功率。动态重构策略使体系根据不断变化的任务需求、资源可用性和环境条件动态调整其配置和行为。

由于装备体系具有信息交互能力，可通过装备体系耦合网络拓扑调控来抑制不同类型与强度的内外部扰动。现阶段对装备体系耦合网络拓扑结构的防控策略主要关注防控策略和约束条件（经济和技术）等方面。资源层内所有感知与执行要素均围绕顶层任务进行多层决策，因此，一旦 OODA 中的某个要素遭到破坏，其决策机制能够快速反应，立即组织其余功能相似的要素进行重构，从而构成新的 OODA 对目标进行打击。通过上述分析，给出三类基于规则的装备体系重构策略，如图 D.1 所示。

重构策略 I：团簇/平台内重构，当团簇/平台内要素失效，同一团簇/平台内同类要素可进行协同重构，该策略使体系进行降级使用，保持在任务基线之上。

重构策略 II：团簇/平台间重构，当团簇/平台内要素失效，相邻团簇/平台同类要素可通过中继要素进行协同，该策略允许体系进行降级使用，保持在任务基线之上。

重构策略 III：当要素失效，可通过修复或新增要素进行重构，使体系恢复至完好状态，但该策略需要消耗额外资源与费用。

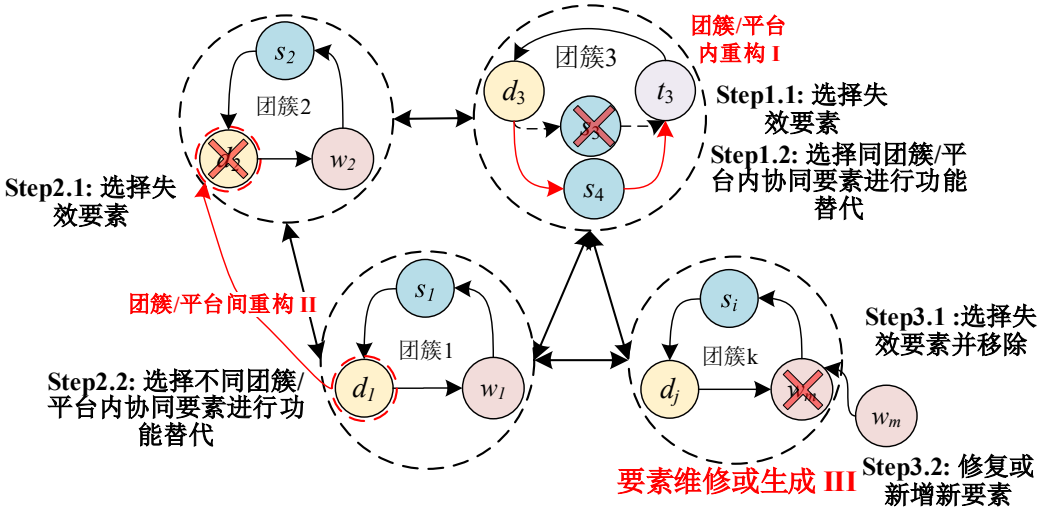

图 D.1 基于规则的装备体系拓扑重构策略

D.3 结构可靠度计算算法

本文件给出装备体系有效OODA环及结构可靠性仿真算法，N_{sim} 为模拟次数，伪代码如表D.1所示。

表 D.1 装备体系有效 OODA 环及结构可靠性仿真算法

	算法 1: 装备体系有效OODA环及结构可靠性仿真算法
1	输入: 初始网络 $D=(V,E,\varphi,\psi)$；要素 s_i,d_j,w_m 属性和连边属性：e_{s_i,d_j}，e_{d_j,w_m}；迭代次数 N_{sim}，仿真时间 T_{sim}.
2	输出: 体系结构可靠性：$R_{ESoS}(t)$ 和有效OODA环数量：$N_{eOODA}(t)$.
3	仿真初始化 $n_{sim}=0,t_{sim}=0$
4	$N_{OODA}(0)=\sum_{i=1}^{I}\sum_{m=1}^{M}x_{s_i,d_j}\times x_{d_j,w_m}$
5	**for** t_{sim} **to** T_{sim} 　**for** n_{sim} **to** N_{sim}
6	随机失效：通过蒙特卡罗仿真及要素失效率及其分布确定装备体系的要素失效数量，随机移除失效节点要素及其连边，并将失效要素添加至失效要素列表
7	最大度攻击失效：确定攻击模式，将体系中的系统按照要素度数降序排列，然后移除相应数量的失效要素及其连边
8	**for** 失效要素列表中的每个要素 　　团簇/平台内重构：集群间的相同要素可以相互替换 　　团簇/平台间重构：每个集群中的相同要素可以相互替换 　　要素维修或生成：通过蒙特卡罗方法添加新要素或修复失效要素，生成相应要素和连边 　　**end**
9	要素存在：基于示性函数和蒙特卡罗方法来确定要素的存在
10	连边存在：基于示性函数和蒙特卡罗方法来确定连边的存在
11	计算邻接矩阵元素：$\begin{cases}x_{s_i,d_j}=\alpha(s_i)\times\alpha(d_j)\times\alpha(e_{s_i,d_j})\\x_{d_j,w_m}=\alpha(d_j)\times\alpha(w_n)\times\alpha(e_{d_j,w_m})\end{cases}$
12	计算要素 S 与 W 的邻接矩阵为 $\mathbf{A_{SW}}$
13	该次仿真有效OODA环数量：$N_{eOODA}(t)=\sum_{i=1}^{I}\sum_{m=1}^{M}x_{s_i,d_j}\times x_{d_j,w_m}$
14	$R'_{ESoS}(t,n_{sim})=N_{eOODA}(t)/N_{OODA}(0)$
	end for
15	$R_{ESoS}(t)=\sum_{n_{sim}=1}^{N_{sim}}R'_{ESoS}(t,n_{sim})/N_{sim}$
	end for
16	**return** $N_{eOODA}(t)$ and $R_{ESoS}(t)$.

ICS 03.120.01
CCS V 00

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/CICC 35008—2026

装备体系任务可靠性建模与评估

Mission reliability modeling and assessment for equipment system of systems

2026-02-28 发布　　2026-02-28 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：西北工业大学、无人飞行器技术全国重点实验室、北京航空航天大学、国防科技大学、中国船舶集团有限公司综合技术经济研究院、中国运载火箭技术研究院战术军贸事业部、中国航空综合技术研究院、中国空间技术研究院钱学森空间技术实验室、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）。

本文件主要起草人：陈志伟、李大庆、白光晗、尹斯源、王凯旋、张罗庚、肖和业、刘一萌、王建峰、方晓彤、褚嘉运、马梓瑞、张雨露、吕亚强、崔巍巍、袁远、林聪、徐嘉。

装备体系任务可靠性建模与评估

1 范围

本文件规定了开展装备体系任务可靠性建模与评估工作的目的和原则、一般要求和详细要求。基本程序包括：装备体系结构与要素分析、装备体系任务可靠性数据收集、装备体系任务可靠性建模、装备体系任务可靠性评估、编制装备体系任务可靠性评估报告，为装备体系任务可靠性评估工作提供依据和指导。

本文件适用于装备体系任务可靠性建模与评估工作，是装备可靠性工程在体系层面的深化与应用。以单装可靠性为依据和输入，通过构建基于任务剖面的任务链模型，系统性地量化装备体系在面临内外部扰动及任务变更等复杂场景下，维持其核心功能与保持任务链完整的概率与能力。

2 引用文件

下列文件中的有关条款通过引用而成为本文件的条款。凡注明日期或版次的引用文件，其后的任何修改单（不包括勘误的内容）或修订版本都不适用本文件。凡未注日期或版次的引用文件，其最新版本适用于本文件。

GJB 450B　　装备可靠性工作通用要求

GJB 451B　　可靠性维修性保障性术语

GJB 1909A　　装备可靠性维修性保障性要求论证

3 术语和定义

GJB450B、GJB451B等标准确立的以及下列术语和定义适用于本文件。

3.1

体系　system of systems（SoS）

多个独立和有效系统的集合，通过共享资源与能力，构成一个提供独特功能的更大系统。

[来源：GJB 451B-2021，A.1.2]

3.2

装备体系　equipment system of systems（ESoS）

装备体系是根据任务需求、经济和技术能力，由一定数量和质量相互关联、功能互补的多种装备，按照装备的优化配置和提高整体作战能力的要求，综合集成的装备类别、结构和规模的有机整体。

3.3

装备体系任务可靠性　mission reliability of ESoS

在规定的条件下和规定的时间内，装备体系完成规定任务使命的能力。

3.4

装备体系任务可靠度　degree of mission reliability of ESoS

在规定的条件下和规定的时间内，装备体系完成规定任务使命的概率。

3.5

有效任务链　effective task chain

为完成特定目标任务，在遭受内外部扰动（内部扰动指系统或设备自身的故障、失效；外部扰动指遭受人为的、有目的性的恶意攻击和环境条件变化）情况下探测、决策、执行和目标类要素按照任务与时序逻辑依序构成的一种闭环结构。

3.6

有效任务网　effective task network

由多个有效任务链通过要素共享、信息融合而形成的协同异质网络结构。

4　基本原则

装备体系任务可靠性建模与评估工作应遵循以下原则：

a)　需求牵引：在建立装备体系任务可靠性模型时，应充分考虑不同任务的特征与需求，调整任务链结构以及任务需求等模型架构；

b)　任务剖面覆盖：应充分了解、分析当前装备体系任务的剖面，以实现任务可靠性模型对任务剖面中各阶段子任务的全覆盖，构建准确的任务链；

c)　可行性：所建立的装备体系任务可靠性模型和评估方法应有明确的验证方法与途径，以确保其可行性；

d)　数据真实性：装备体系任务可靠性评估过程中用到的所有参数（装备要素可靠度、功能性能、位置坐标等）的来源都要是明确的、可追溯、可检查的。

5　一般要求

5.1　开展时机

一般在装备体系结构方案设计完成后与任务准备阶段，结合任务剖面开展任务可靠性建模与评估工作，并在任务执行阶段对模型和评估结果进行迭代更新。

装备体系开展任务可靠性评估需达到以下前提条件：

a)　已完成装备体系的结构方案设计，各平台/装备的组成、功能、接口及网络连接关系明确、稳定，能够清晰描述出潜在的任务链/任务网结构；

b)　针对待评估的具体任务，其任务剖面已详细定义，包括任务的阶段划分、时序关系、成功/失败判据、环境条件以及各阶段涉及的主要装备及其使用强度；

c)　已收集并确认评估所需的基础数据与信息，详见附件表 A.1。

5.2　参数要求

依据 GJB 1909A 中的第五章装备可靠性维修性保障性的定量要求、GJB 450B 中的第四章第六节可靠性参数、GJB 451B 中的第五章第二节可靠性术语参数等相关标准的规定，本文件明确了装备体系任务可靠性参数，参见附录 A，附录 B。

作为装备体系任务执行使用要求的重要组成部分，任务可靠度参数为体系的任务可靠性设计、分析、试验与评价提供了统一的量化基准与验证标尺。在建模与评估阶段，应通过构建的模型与仿真分析，对该参数值进行精确计算与评估。

任务可靠性评估结果是衡量装备体系架构设计是否达标的关键判据。当评估结果达到或超越目标值时，表明该体系结构设计合理、冗余配置有效，在体系层面具备了在复杂对抗环境中应对高强度、多波次饱和任务需求的能力，能以较高的概率保证核心任务使命的达成。反之，当评估结果低于最低可接受值时，则证明该体系在特定任务剖面下存在结构性缺陷或能力短板，其整体任务执行能力不可靠，难以支撑预定的任务。评估结果所揭示的薄弱环节即为体系优化与设计迭代的直接依据。

5.3　数据要求

为确保装备体系任务可靠性建模和评估工作的有效实施，在装备体系的任务可靠性建模与评估过程中，依照实际情况，明确一般数据要求：

a)　根据装备体系任务可靠性建模与评估过程需求，定义清晰的数据收集范围和指标；

b)　所有数据需明确来源、采集方法与置信度，关键数据应具备可追溯性；

c)　数据格式须标准化、结构化，确保跨装备、跨模型的无歧义解析与集成；

d) 建立数据访问权限控制机制，实现对数据的安全管理。

数据要求参见附录 C。

6 详细要求

6.1 基本程序

装备体系任务可靠性建模与评估基本程序包括体系结构与要素分析、装备体系任务可靠性数据收集、装备体系任务可靠性建模、装备体系任务可靠性评估、编制装备体系任务可靠性评估报告。工作流程如图 1 所示。

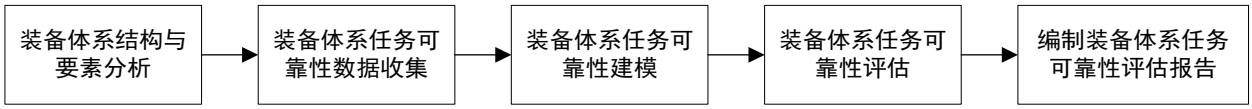

图1 装备体系任务可靠性建模与评估工作流程示意图

6.2 装备体系结构与要素分析

结合装备体系结构特点，确定装备体系结构层次划分方式，一般包括任务层、能力层与物理资源层。根据任务链理论，对装备体系要素进行识别，一般包括探测类、决策类、执行类、通信类要素，参见附录 D。

6.3 装备体系任务可靠性数据收集

根据装备体系任务需求，需要进行相应数据收集。一般包括：

a) 装备体系拓扑结构及装备要素数据；

b) 装备体系任务剖面，包括任务的定义及时序关系和持续时间、任务成功判定准则、各任务的相对频度以及任务涉及的主要装备及其使用强度；

c) 装备体系要素类型、要素数量等；

d) 装备体系中单个装备的可靠性参数，如可靠性、失效率、修复率等；

e) 装备体系中单个装备的性能参数，包括作用距离、任务成功概率等；

f) 装备体系内外部扰动模式与强度等；

g) 装备体系的重构策略。

6.4 装备体系任务可靠性建模

装备体系在执行不同的任务时，例如：探测任务、决策任务、打击任务、保障任务等，其架构与需求存在差异，需构建基于不同类型任务链（例如：探测任务链、决策任务链、打击任务链等）的任务可靠性模型。

结合装备体系架构与任务剖面，确定的任务可靠性建模方法具体包括：确定装备体系结构与任务剖面、确定装备体系要素模型、确定装备体系连边模型、确定有效任务网模型。一条有效任务链的形成代表了单个任务活动的有效执行。因此，将装备体系有效任务链数量作为衡量其任务可靠性的量化指标，通过将有效任务链数量与任务需求（阈值）进行量化分析，进而建立装备体系任务可靠性模型，具体流程如下：

6.4.1 确定装备体系结构与任务剖面

应充分明确装备体系的层次结构，明确不同装备之间以及装备体系与任务目标之间的交互关系，包括通信、探测、打击等。同时，应充分考虑装备体系当前执行任务的特征，明确任务的阶段划分、时序关系、成功/失败判据、环境条件以及各阶段涉及的主要装备及其使用强度。

6.4.2 确定装备体系要素模型

应充分考虑装备体系要素异质性，通过分析要素属性和失效模式建立要素模型。装备体系要素类型一般应包括探测类要素、决策类要素、执行类要素和目标类要素，要素属性涵盖要素类型、要素数量、位置坐标、可靠性和性能等。装备体系要素建模具体过程参见附录 E.1。

6.4.3 确定装备体系连边模型

应充分考虑装备体系要素之间的数据传输和任务分配，建立装备体系连边模型。考虑到装备体系中的各类要素表现出不同的连接模式，建立从目标到探测、探测到决策、决策到执行、执行到目标的连边模型，为形成装备体系任务网络提供连边模型。装备体系连边建模具体过程参见附录 E.2。

6.4.4 确定装备体系有效任务网模型

应基于已构建的装备体系要素模型和连边模型，建立装备体系任务网模型。结合异质要素属性特点并依次连边运作，构建对特定目标产生杀伤效果的有效任务链。装备体系中的有效任务链通过要素共享、信息融合而形成的协同异质网络结构，即有效任务网模型，其建模过程参见附录 E.3。

6.4.5 确定装备体系任务可靠性模型

应根据装备体系结构与有效任务网模型，结合装备体系任务需求（阈值），建立装备体系任务可靠性模型和量化算法，具体过程参见附录 E.4。

6.5 装备体系任务可靠性评估

结合装备体系任务需求和相关数据，确定的装备体系任务可靠性评估方法包括：确定有效任务链计算方法、确定装备体系失效与重构策略、确定任务可靠性计算方法，装备体系任务可靠性具体评估过程参见附录 F。

6.5.1 有效任务链计算

考虑到有效任务链是装备体系中的各类要素相互依存、依次运作对特定目标产生线性影响效果的链路，建立有效任务链计算方法，计算步骤具体包括：生成转移矩阵、生成到达矩阵、计算任务网有效任务链数量、计算目标任务链数量、判断任务是否成功、计算邻接矩阵存在概率、计算要素存在概率、计算连边存在概率。计算有效任务链具体过程参见附录 F.1。

6.5.2 装备体系要素失效与重构策略

应充分考虑不同类型要素在随机失效和蓄意攻击下的失效模式分析，确定四种重构策略：团簇/平台间重构、团簇/平台内重构、要素维修或生成和任务重分配，以快速提升体系杀伤能力。装备体系要素失效与重构的具体过程参见附录 F.2。

6.5.3 任务可靠性计算

应结合到达矩阵和蒙特卡罗算法，计算装备体系有效任务链数量，并基于此计算装备体系任务可靠性。任务可靠性仿真算法伪代码参见附录 F.3。

6.6 编制装备体系任务可靠性评估报告

装备体系任务可靠性评估完成后应编制评估报告，报告主要包含：评估目的、评估内容、评估所依据的标准号、装备体系任务可靠性相关定义、装备体系任务剖面、装备体系结构与要素、任务可靠性建模、任务可靠性评估结果、结论分析、问题与建议等。

附 录 A

（资料性）

装备体系任务可靠性参数与评估需求

在评估工作开始前应充分填写体系评估需求清单，以完成任务可靠性数据收集，需求清单具体如下。

表 A.1 体系评估需求清单

需求类别	具体需求	备注
任务需求信息	任务目标的探测难度、打击难度等	
	任务的主要剖面与各任务阶段所需要素	
	不同任务阶段体系的任务失败判据	
体系架构信息	装备体系的层级架构、执行类型、要素类型、要素数量等	
	要素类型（探测装备、决策装备、执行装备）、各类要素中装备的组成情况、要素初始连接情况（拓扑架构）	
	要素的任务执行能力（探测能力、探测概率、执行距离等）	
	要素性能退化过程、要素随机失效类型、要素失效率	
	装备体系任务可靠性建模与评估指标体系	
	要素修复率	
重构过程信息	执行装备内同类要素之间的重构、不同执行装备之间要素的重构、通过修复或新增要素进行重构、装备体系任务重构（任务发生变化）	

附 录 B

（资料性）

任务可靠性参数指标体系

表 B.1 装备体系任务可靠性参数指标体系

	体系层级	指标级	评价指标
装备体系任务可靠性建模与评估指标体系	体系层	基本可靠性	有效 OODA 环数量等
		任务可靠性	任务可靠度、任务成功率、体系适应性等
	能力层	探测链可靠性	探测覆盖可靠性、抗干扰可靠性、数据时效可靠性等
		决策链可靠性	决策链闭环时效可靠性、容错切换可靠性、智能决策可靠性等
		打击链可靠性	打击精度可靠性、火力持续可靠性、打击协同可靠性等
		信息链可靠性	动态拓扑连通可靠性、抗干扰传输可靠性、端到端时效可靠性等
	资源层	探测类要素	可靠度、修复率、探测距离、探测概率、可同时搜索目标数量、位置坐标、通信距离、探测任务分配、漏警率等
		决策类要素	可靠度、修复率、指令响应时间、信息处理时间、位置坐标、通信距离、准确性、网络传输延时等
		执行类要素	可靠度、修复率、打击精度、杀伤距离、毁伤概率、打击任务分配、位置坐标、通信距离等
		通信类要素	传输距离、通信可靠度、数据丢包率等

附 录 C

（资料性）

装备体系任务可靠性建模与评估数据

表 C.1 装备体系任务可靠性建模与评估数据要求

数据类型	详细数据
单装可靠性数据	平均故障间隔时间、平均修复时间、故障率分布函数、修复率、贮存寿命、使用寿命、性能退化模型及参数、任务前准备时间、再次出动准备时间、使用频度与强度等
体系交互数据	要素间最大通信距离、有效传输速率、通信覆盖范围、传输时延、数据丢包率、链路建立成功率、通信中断统计规律、链路可用度、通信任务可靠度、抗干扰恢复时间等
能力性能数据	探测距离、探测概率、识别概率、虚警率、多目标处理能力、决策周期、指令响应时间、目标分配准确率、协同规划时效、打击范围、毁伤概率、突防概率、弹药基数、重瞄时间等
环境与对抗数据	地形地貌、气象与电磁环境等影响装备性能的环境参数、电子干扰强度与样式、网络攻击特征、物理摧毁概率、隐身与伪装特性、任务目标特性、威胁等级、运动规律、防御能力、典型任务场景想定、红蓝双方作战条令、对抗强度与频率等

附 录 D

（资料性）

装备体系结构与要素分析

装备体系中的关键要素通过通信网络实现资源与信息的共享，使得装备体系形成一个有机整体。依据装备体系结构、OODA 环和杀伤链理论，在装备体系设计、运行及其执行任务过程等各个阶段，按照影响装备体系的关键能力得到影响装备体系的五个要素：

a）探测类要素

探测类要素指装备体系中用于感知、发现、识别和跟踪目标的各类传感器、侦察平台及相关信息处理能力的统称。在装备体系任务执行过程中，探测类要素肩负发现目标、识别目标信息等重要职责。

b）决策类要素

在装备体系任务执行过程中，决策类要素肩负处理目标信息、下达指令等重要职责，是影响装备体系规划、形成与运用的关键因素。

c）执行类要素

执行类要素构成装备体系的行动终端，承担将决策转化为实际效能的职责。该要素通过多样化的行动手段（包括火力压制、电子干扰、信息对抗等），在物理域或信息域实现任务目标。不同装备搭载的专用任务载荷与控制系统，赋予体系差异化的任务执行能力。作为装备体系能力输出的最终环节，执行类要素的性能直接决定任务链构建的实际效果。

d）通信类要素

通信类要素是指装备体系中用于实现信息传输、交换与共享的通信设备、网络架构、协议标准的统称。在装备体系执行任务的过程中，通信系统肩负联合组网、信息传递等重要职责。

e）目标类要素

目标类要素是指装备体系中所关注或作用的对象，包括敌方兵力、设施、武器平台等具有军事价值的实体及其特征、状态和行为属性。

装备体系要素是任务执行力量的主要源泉，上述五个关键要素是构建任务链的基础，其他辅助系统对装备体系效能的影响程度较低，将探测、决策、任务执行和通信确定为装备体系要素，并对其进行建模与分析。装备体系各装备的物理资源之间以信息为介质，以网络为载体，进行资源与信息的共享，以实现共同的目标，完成共同的使命。同时，通信网络是实现各物理资源之间“资源与信息共享”的前提与基础，本文件将要素之间的通信等效为链路。装备体系要素和链路类型如表 D.1 所示。

表D.1　装备体系要素和链路类型

装备体系要素和链路类型	要素功能
探测类要素（S）	对目标探测，获取任务目标情报，感知任务环境
决策类要素（D）	分析处理战场信息与态势，指挥与决策
执行类要素（W）	对敌方目标进行精确打击和电子干扰等
通讯链路（E）	连接通信节点、用于传输信息的物理或逻辑通路
目标类要素（T）	定义任务需求，并作为体系能力构建与评估的基准

附 录 E

（资料性）

装备体系任务可靠性建模方法

E.1 确定要素模型

装备体系中的每个要素都具有一定的属性，并受到特定的限制。通过研究这些属性和限制，对装备体系行为和能力的大小进行定量与定性分析。

E.1.1 探测类要素 S 的属性模型

探测类要素（S）之间可能存在显著的性能差异，设 $sd_{\max}^{s_i}$ 表示要素 s_i 的最大探测距离，如果与目标的距离超过 $sd_{\max}^{s_i}$，则探测类要素无法探测发现目标。探测类要素的探测概率是指探测类要素在其感知范围内成功探测到目标的概率。探测类要素 S 的属性模型建立如下：

$$s_i(I,\lambda_i^s,\mu_i^s,k_i^s,x_i^s,y_i^s,cluster_k,cd_{\max}^{s_i},sd_{\max}^{s_i},...) \quad \text{(E.1)}$$

式中 x_i^s,y_i^s 探测类要素 s_i 的二维位置坐标；$cluster_k$ 为表示 s_i 是装备体系 K 的成员；$cd_{\max_i}^{S}$ 为表示 s_i 的最大通信距离；$sd_{\max}^{s_i}$ 为表示最大探测距离。

E.1.2 决策类要素 D 的属性模型

决策类要素（D）主要由决策及类似要素组成，主要性能指标包括响应时间、吞吐量和准确性等。决策类要素的探测和执行任务分配结果会影响整个 OODA 循环和任务链的生成和效果。此外，决策类要素的通信能力直接影响其向其他要素发布任务的能力。决策类要素 D 的属性模型建立如下：

$$d_j(J,\lambda_j^d,\mu_j^d,k_j^d,x_j^d(t),y_j^d(t),cluster_k,cd_{\max}^{d_j},...) \quad \text{(E.2)}$$

式中 x_j^d,y_j^d 为 d_j 的二维位置坐标；$cluster_k$ 为 d_j 是装备体系 K 的成员；$cd_{\max}^{d_j}$ 为 d_j 的最大通信距离。

E.1.3 执行类要素 W 的属性模型

执行类要素（W）主要包括导弹、巡航导弹、无人机、电磁干扰设备等。不同类型的 W 具有不同的能力，主要性能指标包括任务完成精度、最大任务执行距离和任务完成概率等。W 的属性模型建立如下：

$$w_m(M,\lambda_m^w,\mu_m^w,k_m^w,x_m^w,y_m^w,cluster_k,cd_{\max}^{w_m},wd_{\max}^{w_m},...) \quad \text{(E.3)}$$

式中 x_m^w,y_m^w 为表示 w_m 的二维位置坐标；$cluster_k$ 为表示 w_m 是装备体系 K 的成员；$cd_{\max}^{w_m}$ 和 $wd_{\max}^{w_m}$ 为表示 w_m 的最大通信距离和任务执行距离。

E.1.4 目标类要素 T 的属性模型

目标类要素（T）主要性能指标包括目标价值、探测难度和目标执行难度。目标类要素的属性模型建立如下：

$$T_n(N,\lambda_n^t,\mu_n^t,x_n^t,y_n^t,cluster_k,value_n^t,...) \quad \text{(E.4)}$$

式中 N 为表示目标类要素的数量；x_n^t,y_n^t 为表示 t_n 的二维位置坐标；$cluster_k$ 为表示 t_n 是装备体系 K 的成员；$value_n^t$ 为表示目标价值。

E.1.5 要素的存在性函数

$$c_{s_i}=\begin{cases}1 & s_i\text{存在}\\ 0 & s_i\text{不存在}\end{cases}\quad c_{d_j}=\begin{cases}1 & d_j\text{存在}\\ 0 & d_j\text{不存在}\end{cases}\quad c_{w_m}=\begin{cases}1 & w_m\text{存在}\\ 0 & w_m\text{不存在}\end{cases}\quad\text{……(E.5)}$$

式中 c_{s_i} 为要素 s_i 的存在指示变量，当要素存在时为 1，当要素不存在时为 0；c_{d_j} 为要素 d_j 的存在指示变量，当要素存在时为 1，当要素不存在时为 0；c_{w_m} 为要素 w_m 的存在指示变量，当要素存在时为 1，当要素不存在时为 0。

E.2 连边模型

在实际任务过程中，探测类要素在发现任务目标方面发挥着至关重要的作用，其通过边将信息传送给决策类要素，决策类要素根据探测类要素和任务目标的属性分配不同的探测任务。然后，决策类要素根据探测信息分配执行任务，并将分配结果传输给执行类要素。执行类要素收到任务信息后执行任务。

E.2.1目标-探测边模型

边 e_{t_n,s_i} 考虑了探测任务的分配结果和探测概率。e_{t_n,s_i} 模型建立如下：

$$e_{t_n,s_i}(d_{t_n,s_i},result_{t_n,s_i},sp_{t_n,s_i},\cdots)\quad\text{……(E.6)}$$

式中 $result_{t_n,s_i}$ 为探测任务分配的结果；d_{t_n,s_i} 为 s_i 和 t_n 的距离；sp_{t_n,s_i} 为 s_i 对 t_n 的探测概率。

E.2.2 探测-决策边模型

边 e_{s_i,d_j} 考虑了要素 s_i 和 d_j 之间的通信可靠性和丢包问题。

$$e_{s_i,d_j}(d_{s_i,d_j},R^c_{s_i,d_j},plr^c_{s_i,d_j},\cdots)\quad\text{……(E.7)}$$

式中 d_{s_i,d_j}、$R^c_{s_i,d_j}$ 和 $plr^c_{s_i,d_j}$ 分别为 s_i 和 d_j 的距离、通信可靠性和丢包率。

E.2.3 决策-执行边模型

边 e_{d_j,w_m} 传输任务指令，其中考虑了要素 d_j 和 w_m 之间的通信可靠性和数据包丢失。

$$e_{d_j,w_m}(d_{d_j,w_m},R^c_{d_j,w_m},plr^c_{d_j,w_m},\cdots)\quad\text{……(E.8)}$$

式中 d_{d_j,w_m}、$R^c_{d_j,w_m}$ 和 $plr^c_{d_j,w_m}$ 分别为 d_j 和 w_m 的距离、通信可靠性和丢包率等。

E.2.4 执行-目标边模型

边 e_{w_m,t_n} 考虑了执行任务的分配结果、突防概率和毁伤概率等。

$$e_{w_m,t_n}(d_{w_m,t_n},result_{w_m,t_n},wp_{w_m,t_n},\cdots)\quad\text{……(E.9)}$$

式中 $result_{w_m,t_n}$ 为执行任务的分配结果；d_{w_m,t_n} 为 w_m 和 t_n 的距离；wp_{w_m,t_n} 为 w_m 对 t_n 的毁伤概率。

E.2.5 边存在性指示函数

$$c_{e_{s_i,d_j}}=\begin{cases}1 & e_{s_i,d_j}\text{存在}\\ 0 & e_{s_i,d_j}\text{不存在}\end{cases}\quad c_{e_{t_n,s_i}}=\begin{cases}1 & e_{t_n,s_i}\text{存在}\\ 0 & e_{t_n,s_i}\text{不存在}\end{cases}\quad\text{……(E.10)}$$

$$c_{e_{d_j,w_m}}=\begin{cases}1 & e_{d_j,w_m}\text{存在}\\ 0 & e_{d_j,w_m}\text{不存在}\end{cases}\quad c_{e_{w_m,t_n}}=\begin{cases}1 & e_{w_m,t_n}\text{存在}\\ 0 & e_{w_m,t_n}\text{不存在}\end{cases}$$

E.3 有效任务网模型

装备体系由多种类型的系统组成，每种系统都具有不同的功能和相互关系。简单的同构有向图网络不足以有效描述不同系统之间的关系。因此，基于异构有向图建立了一个有效任务网模型，为不同的要素和边赋予实际意义。在理想状态下，装备体系各要素相互连接，形成一个完全连通的网络。然而，在实际任务场景中，组成系统不同类型的要素会受到自身属性和运行资源的限制。令 $s_i(i=1,2,...,I)$ 代表第 i 个探测类要素， $d_j(i=1,2,...,J)$ 代表第 j 个决策类要素， $w_m(m=1,2,...,M)$ 代表第 m 个执行类要素， $t_n(n=1,2,...,N)$ 代表第 n 个目标类要素，I,J,M,N 代表不同类型要素的数量。异构装备体系由不同的系统组成，不同的系统体现为一个装备体系 $G=(V,E,\xi,\zeta)$，其中 $V=(S,D,W,T)$，$E=\{e_{t_n,s_i},e_{s_i,d_j},e_{d_j,w_m},e_{w_m,t_n}\}$，其中 $|\xi|=4$，$|\zeta|=4$，$S=\{s_1,s_2,...,s_I\}$，$D=\{d_1,d_2,...,d_J\}$，$W=\{w_1,w_2,...,w_m\}$，$T=\{t_1,t_2,...,t_n\}$。边 $e_{s_i,d_j}\in E$ 意味着从 s_i 到 d_j 的信息传输。

E.3.1 有效任务链

有效任务链是一种链式结构，在传统任务链基础上，考虑了装备体系任务执行过程中随机失效、蓄意攻击、任务分配结果、关键性能指标和任务执行资源限制等内外部因素的综合影响。

装备体系的有效任务链描述了为完成整体任务，各功能要素按照整体任务逻辑顺序进行协作的单一、完整流程。该模型考虑了要素故障、资源约束及外部干扰对任务链中各个功能段及其连接的影响。

有效任务链表示为：

$$eol=t_n\rightarrow s_i\rightarrow d_j\rightarrow w_m\rightarrow t_n \quad \text{.....(E.11)}$$

式中 s_i 为装备体系探测类要素； d_j 为装备体系决策类要素； w_m 为装备体系执行类要素； t_n 为装备体系目标类要素。

E.3.2 有效任务网

由于任务场景的复杂性，装备体系要产生多个有效任务链能完成任务，一条有效任务链中的要素也可能出现在多条有效任务链中（如果执行类要素是物理要素，则只能出现一次）。类似要素进行信息融合和资源共享。因此，任务链通过共享相似要素进行协作，从而形成有效任务网。有

有效任务网用一个集合来表示：

$$eon=\{A,V,E\} \quad \text{.....(E.12)}$$

式中 $A=\{\mathbf{A_{TS}},\mathbf{A_{SD}},\mathbf{A_{DW}},\mathbf{A_{WT}}\}$——描述不同要素之间连接的邻接矩阵集合。例如：要素 S 和 D 的邻接矩阵 $\mathbf{A_{SD}}$。

$$\mathbf{A_{SD}}=\begin{bmatrix} x_{s_1,d_1} & x_{s_1,d_2} & x_{s_1,d_3} & \cdots & x_{s_1,d_J} \\ x_{s_2,d_1} & x_{s_2,d_2} & x_{s_2,d_3} & \cdots & x_{s_2,d_J} \\ x_{s_3,d_1} & x_{s_3,d_2} & x_{s_3,d_3} & \cdots & x_{s_3,d_J} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{s_I,d_1} & x_{s_I,d_2} & x_{s_I,d_3} & \cdots & x_{s_I,d_J} \end{bmatrix} \quad \text{.....(E.13)}$$

式中 $x_{s_i,d_j}(i=1,2,...,I;j=1,2,...,J)$ 为邻接矩阵 $\mathbf{A_{SD}}$ 的元素。

E.4 任务可靠性量化模型

装备体系任务可靠度是装备体系任务可靠性的量化参数，可通过仿真算法进行量化。在典型任务剖

面下对装备体系执行任务能力进行仿真，装备体系在 t 时刻的有效任务链数量 $N_{eol}(t)$ 与任务需求（阈值）τ 进行比较，若 $N_{eol}(t) \geqslant \tau$，则认为该次任务成功，否则失败。通过 N_{sim} 次仿真，将装备体系在 t 时刻仿真任务成功的次数与总仿真次数的比值作为装备体系在 t 时刻的任务可靠度，表示：

$$MR_{ESoS}(t) = \frac{Num_{task-success}(t)}{N_{sim}} \quad \text{(A.1)}$$

式中 $MR_{ESoS}(t)$ 为在 t 时刻的装备体系任务可靠度；N_{sim} 为任务总数；$Num_{task-success}(t)$ 为在 t 时刻任务成功的次数。当 $N_{eol}(t) \geq \tau$ 时，$Num_{task-success} = Num_{task-success} + 1$。否则，$Num_{task-success} = Num_{task-success}$，$\tau$ 为任务成功时所需有效任务链的数量阈值。

附 录 F

（资料性）

装备体系任务可靠性评估方法

F.1 有效任务链计算

F.1.1 转移矩阵

$\mathbf{A}=\mathbf{A_{TS}}\bullet\mathbf{A_{SD}}\bullet\mathbf{A_{DW}}\bullet\mathbf{A_{WT}}$ 是装备体系关于异构有向图 $T\to S\to D\to W\to T$ 的转移矩阵。如果要素 s_i,d_j,w_m 和 t_n 存在，且要素之间存在关系，则元素 $x_{t_n,s_i}=x_{s_i,d_j}=x_{d_j,w_m}=x_{w_m,t_n}=1$，如果要素之间不存在关系，则 $x_{t_n,s_i}=x_{s_i,d_j}=x_{d_j,w_m}=x_{w_m,t_n}=0$。例如，元素 $x_{t_n,s_i},x_{s_i,d_j},x_{d_j,w_m}$ 和 x_{w_m,t_n} 表示为要素和边的指示函数的乘积，如下所示：

$$\begin{cases}x_{t_n,s_i}=\alpha(t_n)\times\alpha(s_i)\times\alpha(e_{t_n,s_i})\\x_{s_i,d_j}=\alpha(s_i)\times\alpha(d_j)\times\alpha(e_{s_i,d_j})\\x_{d_j,w_m}=\alpha(d_j)\times\alpha(w_n)\times\alpha(e_{d_j,w_m})\\x_{w_m,t_n}=\alpha(w_m)\times\alpha(t_n)\times\alpha(e_{w_m,t_n})\end{cases}\quad\text{……(F.1)}$$

式中 $\alpha(\bullet)$ 为示性函数，若示性函数里的元素满足设定条件，则函数值为 1；若示性函数里的元素不满足设定条件，则函数值为 0。

F.1.2 到达矩阵

相邻的转移矩阵 $\mathbf{A_{TS}}$ 和 $\mathbf{A_{SD}}$ 被定义为 $\mathbf{A_{TS}}$ 的到达要素类型与 $\mathbf{A_{SD}}$ 的起始要素类型相匹配的矩阵。通过在相邻的转换矩阵之间进行矩阵乘法，得到 $\mathbf{A_{TT}}=\mathbf{A_{TS}}\bullet\mathbf{A_{SD}}\bullet\mathbf{A_{DW}}\bullet\mathbf{A_{WT}}$，其中 $\mathbf{A_{TT}}$ 是一个方阵，代表要素类型 T 的到达矩阵。到达矩阵提供了在网络中从一种要素类型转换到另一种要素类型的概率信息。通过使用矩阵迹来计算任务网中的有效任务链数量。

$$N_{eol}=Trace(\mathbf{A_{TT}})\quad\text{……(F.2)}$$

式中 N_{eol} 为有效任务链数； $\mathbf{A_{TS}}$, $\mathbf{A_{SD}}$, $\mathbf{A_{DW}}$ 和 $\mathbf{A_{WT}}$ 的维数分别为 $N\times I$, $I\times J$, $J\times M$ 和 $M\times N$ 。

F.1.3 目标类要素任务链数量

为了更好的针对每一个目标类要素 T 考虑其各类影响因素对执行类要素数量和能力限制，首先应使用 $\mathbf{A_{TT}}$ 计算针对每个 T 要素的任务链数量。计算公式如下：

$$N_{t_n}=\mathbf{A_{TT}}(x_{t_n,t_n})\quad\text{……(F.3)}$$

式中 N_{t_n} 为针对要素 t_n 的任务链数。

F.1.4 任务成功公式

$$\alpha(task\ success)=\begin{cases}1 & if\ N_{eol}\geq\tau\\0 & if\ N_{eol}<\tau\end{cases}\quad\text{……(F.4)}$$

F.1.5 邻接矩阵元素的存在性

确保任务成功的重要因素之一是确保任务链是连通的，即任务链所含要素之间的邻接矩阵是存在的。邻接矩阵元素的存在概率如下：

$$\begin{cases}\alpha(x_{t_n,s_i})=\alpha(t_n)\times\alpha(s_i)\times\alpha(e_{t_n,s_i})\\ \alpha(x_{s_i,d_j})=\alpha(s_i)\times\alpha(d_j)\times\alpha(e_{s_i,d_j})\\ \alpha(x_{d_j,w_m})=\alpha(d_j)\times\alpha(w_n)\times\alpha(e_{d_j,w_m})\\ \alpha(x_{w_m,t_n})=\alpha(w_m)\times\alpha(t_n)\times\alpha(e_{w_m,t_n})\end{cases}\quad\text{……..……(F.5)}$$

F.1.6 要素存在性

要分别计算要素和边的存在概率，这与要素和连边的属性和资源限制有关。每个要素的存在概率表示为：

$$\begin{cases}\alpha(t_n)=\alpha(1-F_n^t(t_{\lambda_n^t})\times G_n^t(t_{\mu_n^t}))\\ \alpha(s_i)=\alpha(1-F_i^s(t_{\lambda_i^s})\times G_i^s(t_{\mu_i^s}))\\ \alpha(d_j)=\alpha(1-F_j^d(t_{\lambda_j^d})\times G_j^d(t_{\mu_j^d}))\\ \alpha(w_m)=\alpha(1-F_m^w(t_{\lambda_m^w})\times G_m^w(t_{\mu_m^w}))\end{cases}\quad\text{……..……(F.6)}$$

式中 $F_i^s(t_{\lambda_i^s})$，$F_i^s(t_{\lambda_i^s})$，$F_j^d(t_{\lambda_j^d})$ 和 $F_m^w(t_{\lambda_m^w})$ 为自每个要素上次修复以来的时间的累积分布函数（CDF）；$G_i^s(t_{\mu_i^s})$，$G_i^s(t_{\mu_i^s})$，$G_j^d(t_{\mu_j^d})$ 和 $G_m^w(t_{\mu_m^w})$ 为自每个要素上次失效以来的时间的累积分布函数（CCDF）。

F.1.7 边的存在性

$$\begin{cases}\alpha(e_{t_n,s_i})=\alpha(result_{t_n,s_i})\times\alpha(d_{t_n,s_i})\times\alpha(sp_{t_n,s_i})\\ \alpha(e_{s_i,d_j})=\alpha(cd_{s_i,d_j})\times\alpha(R^c_{s_i,d_j})\times\alpha(1-plr^c_{s_i,d_j})\\ \alpha(e_{d_j,w_m})=\alpha(cd_{d_j,w_m})\times\alpha(R^c_{d_j,w_m})\times\alpha(1-plr^c_{d_j,w_m})\\ \alpha(e_{w_m,t_n})=\alpha(result_{w_m,t_n})\times\alpha(d_{w_m,t_n})\times\alpha(wp_{w_m,t_n})\end{cases}\quad\text{……..……(F.7)}$$

式中 $result_{t_n,s_i}$ 和 $result_{w_m,t_n}$ 分别代表探测任务和执行任务的分配结果；cd_{s_i,d_j} 和 cd_{d_j,w_m} 分别代表探测与决策、决策与执行类要素之间的通信距离；$R^c_{s_i,d_j}$ 和 $R^c_{d_j,w_m}$ 分别代表探测与决策、决策与执行类要素之间的通信任务可靠性；d_{t_n,s_i} 和 d_{w_m,t_n} 分别代表探测类要素的探测距离与执行类要素的任务执行距离；$plr^c_{s_i,d_j}$ 和 $plr^c_{d_j,w_m}$ 分别代表探测与决策、决策与执行类要素之间的通信丢包率；sp_{t_n,s_i} 和 wp_{w_m,t_n} 分别代表探测成功概率与任务执行成功概率。

F.2 动态重构策略

动态重构是体系或系统通过调整其自身资源和结构配置来响应环境中的不同情况以改变其状态的能力。重构策略使装备体系能够根据不断变化的任务要求、资源可用性和环境条件动态调整其配置和行为，本文件建立了四种重构策略：簇间重构、簇内重构、要素维修或生成与任务重分配。

装备体系团簇/平台间重构：指调整或重构体系内不同团簇/平台装备之间的连接和关系的过程。这涉及修改装备体系不同团簇/平台间的通信模式和互动关系，以提高体系性能，实现预期目标。

装备体系团簇/平台内重构：涉及修改装备体系同一团簇/平台内要素之间的连接和关系。这一过程旨在优化装备体系的内部结构和协调，使要素之间的通信、资源分配和任务分配更加高效。

要素维修或生成：指在装备体系内维修或增加新要素。这可能涉及部署额外的装备体系或激活休眠要素，以提高体系整体的能力和容量。

任务重分配：涉及体系内要素间任务的重新分配，如临机任务发生、要素故障替代或修复失效、平衡负载等情况。任务重分配的目的是优化任务分配，最大限度地提高体系效率，确保完成优先任务目标。

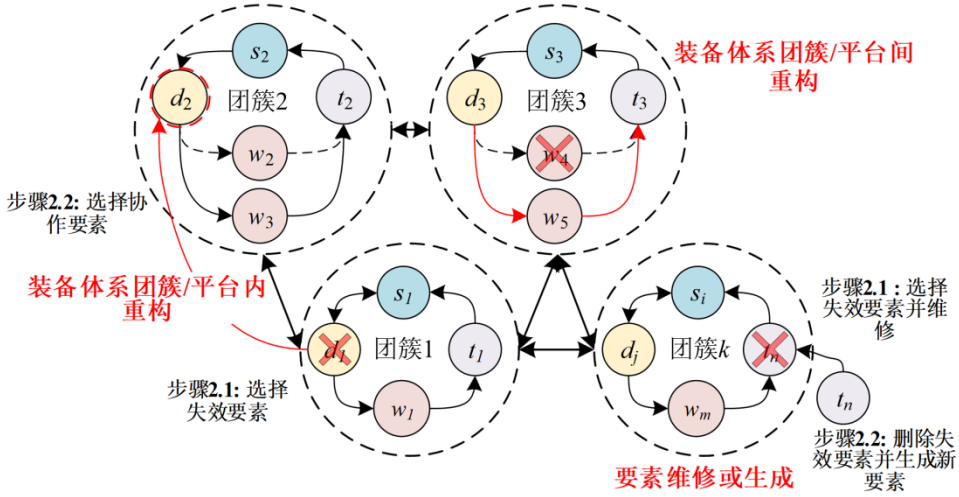

图 F.1　基于规则的任务网动态重构策略

重构策略 1-3 为基于规则的装备体系动态重构策略，如图 F.1 所示。基于任务的装备体系任务重分配，如图 F.2 所示，根据顶层任务和战场态势变化，利用实时任务规划调整装备体系拓扑结构，形成新的有效任务网。

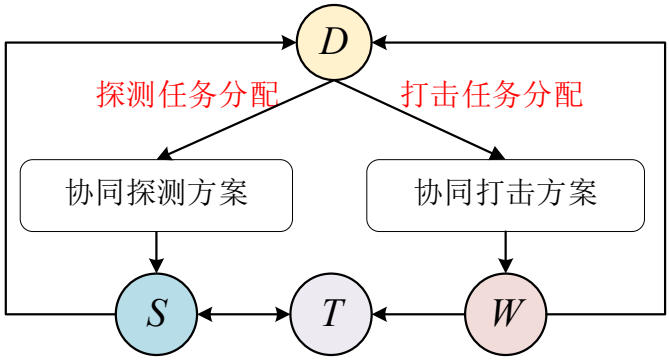

图 F.2　基于任务的装备体系任务重分配

F. 3　任务可靠性仿真算法

在计算装备体系任务链数量并建立动态重构策略后，充分考虑内外部因素对于任务链要素与连边的影响、要素失效与边失效、动态重构策略、任务成功率，建立有效任务链模型。

本文件给出装备体系有效任务链及任务可靠性仿真算法，N_{sim} 为仿真次数，伪代码如表 F.1 所示。

表 F.1　装备体系有效任务链及任务可靠性仿真算法

	算法 1：装备体系有效任务链及任务可靠性仿真算法：
1	输入：初始网络 $D=(V,E,\varphi,\psi)$. 要素属性 $s_i(I,\lambda_i^s,\mu_i^s,k_i^s,x_i^s,y_i^s,cluster_k,cd_{\max}^{s_i},sd_{\max}^{s_i})$，$d_j(J,\lambda_j^d,\mu_j^d,k_j^d,x_j^d(t),y_j^d(t),cluster_k,cd_{\max}^{d_j})$，$w_m(M,\lambda_m^w,\mu_m^w,k_m^w,x_m^w,y_m^w,cluster_k,cd_{\max}^{w_m},wd_{\max}^{w_m})$ 和 $t_n(N,\lambda_n^t,\mu_n^t,x_n^t,y_n^t,cluster_k,value_n^t)$； 边属性：$e_{t_n,s_i}(d_{t_n,s_i},result_{t_n,s_i},sp_{t_n,s_i})$，$e_{s_i,d_j}(d_{s_i,d_j},R_{s_i,d_j}^c,plr_{s_i,d_j}^c)$，$e_{d_j,w_m}(d_{d_j,w_m},R_{d_j,w_m}^c,plr_{d_j,w_m}^c)$，和 $e_{w_m,t_n}(d_{w_m,t_n},result_{w_m,t_n},wp_{w_m,t_n})$；迭代次数 N_{sim}，仿真时间 T_{sim}.
2	输出：任务网中有效任务链的数量 N_{eol} 与装备体系任务可靠性 $MR_{ESoS}(t)$.
3	仿真初始化 $n_{sim}=0$, $t_{sim}=0$
4	for t_{sim} to T_{sim} for n_{sim} to N_{sim}
5	随机失效：通过蒙特卡洛模拟根据要素失效率及其分布确定装备体系的要素失效数量，随机移除失效要素及其边，并将失效要素添加至失效要素列表
6	最大度攻击失效：确定攻击模式，将装备体系中的系统按照要素度数降序排列，然后移除相应数量的失效要素及其边
7	for 失效要素列表中的每个要素 装备体系间重构：装备体系间的相同要素相互替换 装备体系内重构：每个装备体系中的相同要素相互替换 要素维修或生成：通过蒙特卡洛方法维修失效要素或添加新要素，生成相应的要素和边 任务重分配：根据任务分配算法重构装备体系拓扑结构 end
8	要素存在：基于指示函数和蒙特卡洛方法来确定要素的存在
9	边存在：基于指示函数和蒙特卡洛方法来确定边的存在
10	计算邻接矩阵的条目 $x_{t_n,s_i},x_{s_i,d_j},x_{d_j,w_m}$ 和 x_{w_m,t_n}
11	计算到达矩阵 $\mathbf{A_{TT}}=\mathbf{A_{TS}}\times\mathbf{A_{SD}}\times\mathbf{A_{DW}}\times\mathbf{A_{WT}}$
12	有效任务链的数量：$Trace(\mathbf{A_{TT}})=N_{eol}$

13	if $N_{eol}(t)\geq\tau$ then
	$Num_{task\text{-}success}=Num_{task\text{-}success}+1$
	else:
	$Num_{task\text{-}success}=Num_{task\text{-}success}$
	end

14	任务可靠性：$MR_{ESoS}(t)=Num_{task\text{-}success}(t)/N_{sim}$
	end for
	end for
15	Return $N_{eol}(t)$ and $MR_{ESoS}(t)$.

ICS 03.120.01
CCS L 00

T/CICC

中 国 指 挥 与 控 制 学 会 团 体 标 准

T/ CICC 35009—2026

装备体系韧性建模与评估

Resilience modeling and evaluation for equipment system of systems

2026-02-28 发布　　2026-02-28 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。

本文件隶属《复杂体系任务可靠性与安全性》系列团体标准中的《装备体系任务可靠性与安全性系列标准》。《装备体系任务可靠性与安全性系列标准》已经发布了以下部分：

a) 装备体系安全性指标确定要求；
b) 装备体系安全性评估方法；
c) 装备体系基本任务可靠性建模与评估；
d) 装备体系任务可靠性建模与评估；
e) 装备体系韧性建模与评估；

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。

本文件中的附录A～附录D为资料性附录。

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：国防科技大学、军事科学院、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）、西北工业大学、西华大学、南昌航空航天大学、中国航空综合技术研究院、中国运载火箭技术研究院战术武器总体技术部。

本文件主要起草人：白光晗、李大庆、张小可、陈志伟、刘一萌、段东立、刘涛、许贝、李际超、王晓、林聪、胡朗霄。

装备体系韧性建模与评估

1 范围

本文件规定了开展装备体系韧性建模与评估工作的目的和原则、一般要求和详细要求。基本程序包括：装备体系要素分析、装备体系韧性数据收集、装备体系韧性建模、装备体系能力数据收集、装备体系韧性评估、编制装备体系韧性建模与评估报告，为装备体系韧性的建设、优化与决策提供依据和指导。

本标准适用于装备体系韧性建模与评估工作。

2 规范性引用文件

下列文件中的有关条款通过引用而成为本指南的条款。凡注明日期或版次的引用文件，其后的任何修改单（不包括勘误的内容）或修订版本都不适用本指南。凡未注日期或版次的引用文件，其最新版本适用于本指南。

GJB 450B 装备任务可靠性工作通用要求

GJB 451B 任务可靠性维修性保障性术语

GJB 1909A 装备任务可靠性维修性保障性要求论证

GJB 8113 武器装备研制体系工程通用要求

3 术语与定义

GJB451B、GJB450B 等标准确立的以及下列术语和定义适用于本文件。

3.1

体系 system of systems （SOS）

多个独立和有效体系的集合，通过共享资源与能力，构成一个提供独特功能的更大体系。

3.2

装备体系 equipment system of systems （ESoS）

装备体系是根据任务需求、经济和技术能力，由一定数量和质量相互关联、功能互补的多种装备，按照装备的优化配置和提高整体任务能力的要求，综合集成的装备类别、结构和规模的有机整体。

3.3

装备体系韧性 ESoS resilience

装备体系在整个生命周期中面对内外部扰动时，通过冗余、修复及重构等主被动手段，实现抵抗冲击、适应环境变化、快速恢复功能并持续完成任务的能力。

3.4

任务链 task chain

在装备体系内，为精确执行、探测监视等特定任务目的，将探测、决策、执行等装备通过信息链路有机串联，形成的一条环环相扣、高度协同的功能路径。

3.5

任务网 task network

由多个任务链通过要素共享、信息融合而形成的协同异质网络结构。

3.6

扰动 disturbance

对装备体系结构、功能或运行状态造成影响的外部或内部事件，包括但不限于故障、攻击、资源中断和环境恶化等。

3.7

任务能力　mission capability

装备体系在给定时间内、在特定任务（或任务）情景下，完成规定任务的能力，可通过任务完成率、体系效能等指标表征。

3.8

装备体系韧性度量　ESoS resilience metrics

装备体系在抵抗、适应以及恢复整个韧性过程中功能变化的表征。

4　基本原则

装备体系韧性建模与评估工作的目的是建立统一、可操作的建模框架，以及标准化的评估方法。通过标准化的方法，提升装备体系韧性研究的科学性、体系性和可比性，为后续优化设计、决策支持提供基础依据。

装备体系韧性建模与评估工作应遵循以下原则：

a)　需求牵引：在建立装备体系任务可靠性模型时，应充分考虑不同任务的特征与需求，调整任务链结构以及任务链阈值等模型架构；

b)　数据可信与多源融合：应在规定任务剖面下定义模型输入数据与参数库，包括装备清单及能力参数、维修/补给、对抗扰动/毁伤模式与强度等；数据来源需可追溯，开展质量评估与不确定性标注，必要时进行多源数据融合与一致性校准；

c)　合理性：应显式刻画韧性机理与因果关系，包含退化机理、损毁传播与功能削弱机理、恢复与补偿机理、资源约束与竞争机理等；明确模型假设、约束条件与参数适用范围；

d)　恢复性：应定义在局部失效或能力降级时的功能维持与恢复规则，明确切换条件与约束，形成闭环恢复流程，确保任务链能够快速启用备份路径并重构，以维持体系的持续任务能力。

e)　可验证性：应在具体任务场景中对韧性建模与评估开展仿真分析，撰写《装备体系韧性建模报告》以进一步分析与改进。

5　一般要求

5.1　开展时机

一般在装备体系结构方案设计完成后与运行阶段，开展韧性建模与评估工作，并在任务执行阶段对模型和评估结果进行迭代更新。

装备体系开展韧性建模与评估需达到以下前提条件：

a)　已完成装备体系要素确认，各平台/装备的组成、功能、接口及网络连接关系明确、稳定，能够清晰描述出潜在的扰动影响与恢复需求；

b)　已确认装备体系在使用环境下的扰动类型和强度，能够清晰描述不同扰动对装备体系的影响及其应对策略；

c)　已收集并确认评估所需的基础数据与信息，详见附录 A。

5.2　参数要求

本文件明确了装备体系韧性建模与评估参数，参见附录A表A.1。

韧性参数为装备体系的韧性设计、评估提供了统一的标准。在韧性建模与评估阶段，应通过模型建立以及仿真设计，对韧性参数值进行精确计算与评估。

5.3　数据要求

为确保装备体系韧性建模与评估工作的有效性，在建模与评估的过程中，需对数据做出要求：

a)　装备体系韧性建模与评估所用数据应与评估目的、任务情景和体系范围相一致。

b)　数据来源应明确，可包括试验测试、历史运行、仿真结果、设计文档及专家经验等。

c) 数据应满足完整性、准确性、一致性和可追溯性要求；当存在缺失或不确定性时，应说明处理方法及影响。
d) 应包括装备单元清单及主要属性，单元间拓扑关系（决策、通信、保障等）。
e) 应包括关键装备的任务相关能力参数（如探测、执行、处理能力等）。
f) 应包括典型任务类型、任务流程、任务需求（完成率、时效、资源约束等），以及主要任务/运行情景的环境与约束条件。
g) 应包括扰动类型、作用对象及发生时间、持续时间、强度等参数；宜包括典型故障模式、后果及防护/备用措施数据。

具体数据要求参见附录 A 表 A.2。

6 详细要求

6.1 基本程序

装备体系韧性建模的基本程序包括装备体系要素分析、装备体系韧性数据收集、装备体系韧性建模、装备体系能力数据收集、装备体系韧性评估、编制装备体系韧性建模与评估报告。应在装备体系要素分析后，收集装备体系的能力数据，并进行装备体系韧性评估及编制装备体系韧性建模与评估报告。如果装备体系的能力数据无法收集，应收集韧性数据，并进行装备体系韧性建模及仿真获取装备体系的能力数据。具体工作流程如图 1 所示。

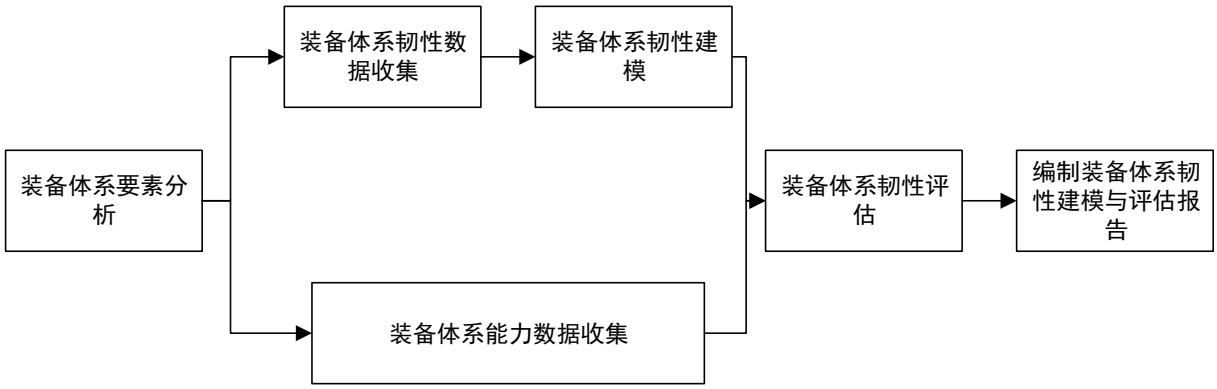

图1 装备体系韧性建模与评估工作流程示意图

6.2 装备体系要素分析

结合装备体系结构特点，确定装备体系结构层次划分方式，物理层次一般包括任务使命、火力单元、装备、设备。根据任务链理论，对装备体系要素进行识别，一般包括探测类、决策类、执行类和目标装备，详见附录 B。

6.3 装备体系韧性数据收集

根据装备体系任务需求，需进行相关数据收集，详见附录A。一般包括：

a) 装备体系拓扑结构、要素组成数据；
b) 各类要素中装备的组成情况、要素初始连接情况；
c) 要素能力退化过程、要素随机失效类型；
d) 装备体系外部扰动类型与强度等；
e) 装备体系的失效与具体的恢复策略，包括正常连接关系、失效后的连接关系和失效后的恢复策略等；
f) 扰动开始时间、恢复开始时间、恢复持续时间等；
g) 任务能力的变化。

6.4 装备体系韧性建模

依据装备体系结构对装备体系划分为探测单元（S）、决策单元（D）、执行单元（I）、目标实

体（T）。其中目标实体为敌方单元或敌方装备体系。确定了装备体系节点建模、任务网建模、任务链建模与计算、装备体系的扰动与恢复方法，详见附录 C。具体流程如下：

6.4.1 装备体系节点建模

应充分考虑装备体系组成的异质性，通过分析装备体系的要素，建立装备体系节点模型。装备体系节点模型一般应包括探测类节点、决策类节点、执行类节点和目标节点。装备体系节点建模具体过程参见附录 C.1。

6.4.2 装备体系任务网建模

基于已构建的装备体系结构模型，建立装备体系任务网模型。结合异质要素属性特点并依次连边运作，构建了探测、通信、决策、执行四层网络结构，其建模过程参见附录C.2。

6.4.3 装备体系任务链建模及计算

考虑到七种不同的任务链，并建立了任务链的计算方法。具体过程参见附录C.3-C.4。

6.4.4 装备体系的扰动与恢复方法

应充分考虑不同类型要素在随机失效和蓄意攻击下的失效模式分析，确定三种恢复策略，即任务链间恢复、任务链内恢复以及节点自恢复，以快速恢复体系能力。装备体系的扰动与恢复方法参见附录 C.5-C.6。

6.5 装备体系能力数据收集

应开展装备体系实际任务场景下的试验，收集相应的能力数据。如果无法收集可通过上述建模及仿真的方式获得。

6.6 装备体系韧性评估

应利用获取的装备体系能力数据，确定韧性度量算法并计算装备体系韧性。根据应用场景的不同，应选用适用的评估方法：能力驱动的韧性评估方法见附录 D.1，任务驱动的韧性评估方法见附录 D.2。

6.7 编制装备体系韧性建模与评估报告

报告主要包含：装备体系任务韧性相关定义、装备体系要素分析方法、装备体系数据清单、装备体系韧性建模方法、装备体系的能力数据特征及装备体系韧性评估结果等。

附 录 A

（资料性附录）

装备体系韧性参数及数据清单

在韧性建模与评估工作开始前，应明确装备体系韧性的参数体系，如表A.1所示。

表A.1 装备体系韧性建模与评估参数体系

	类别	参数名称	符号	含义说明	参数类型
装备体系韧性建模与评估参数体系	体系结构	装备单元数量	N_e	体系中装备单元（节点）的总数量	输入
	任务能力	装备体系任务开始前能力	Q_D	扰动发生前体系稳态的能力	输入
	任务能力	装备体系任务最低能力	Q_{R1}	扰动发生后体系的最低能力	输入
	任务能力	装备体系任务恢复后能力	Q_R	恢复后的体系能力	输入
	任务能力	最低能力要求	$Q_{min}(t)$	体系在 t 时刻的任务最低要求能力	状态
	任务能力	时刻任务能力	$Q(t)$	时刻 t 体系任务能力（能力），用于构造能力-时间曲线	状态
	韧性指标	任务时间韧性度量	R_0	基于任务时间的韧性度量	输出
	韧性指标	任务能力韧性度量	R_1	基于任务能力的韧性度量	输出
	韧性指标	体系综合韧性度量	R	基于任务时间韧性度量和任务能力韧性度量的综合韧性度量	输出
	韧性指标	韧性中的总能力因子	σ	扰动时间段内总能力相对于期望总能力的占比	输出
	韧性指标	韧性中的恢复因子	ρ	恢复后稳定的能力相对于期望能力占比	输出
	韧性指标	韧性中的吸收因子	δ	体系抵抗扰动的水平	输出
	韧性指标	韧性中的易损因子	ζ	扰动过程中能力的波动程度	输出
	韧性指标	韧性中的恢复时间因子	τ	恢复速度的快慢	输出
	韧性指标	切换参数	α	时间韧性和能力韧性之间的切换参数	输入
	时间指标	开始时间	T_0	体系开始工作时间	输入
	时间指标	扰动时间	T_D	体系遭受扰动的时间	输入
	时间指标	恢复开始时间	T_R	体系能力开始恢复的时间	输入
	时间指标	恢复完成时间	T_{SS}	体系能力恢复完成时间	输入

表A.1　装备体系韧性建模与评估参数体系（续）

装备体系韧性建模与评估参数体系	时间指标	任务结束时间	T	体系任务结束时间	输入
	时间指标	信噪比	SNR_{dB}	能力数据的信噪比	输入

应充分填写体系建模与评估的需求清单，以完成体系韧性数据收集，需求清单具体如表 A.2 所示。

表A.2　装备体系韧性建模与评估数据清单

数据类型	详细数据	备注
体系要素信息	装备体系的要素类型、要素数量等	
	要素类型（探测装备、决策装备、执行装备）、各类要素中装备的组成情况	
体系要素信息	要素初始连接情况（拓扑架构）	
任务需求信息	要素的能力值	
	任务目标的探测难度、执行难度等能力值	
	执行任务过程中的扰动因素、攻击方式等	
	目标的重要程度、需要的要素数量	
扰动信息	体系遭受的扰动类型、扰动方式、扰动开始时间	
恢复信息	体系要素的恢复方法、恢复开始时间、恢复结束时间	
韧性评估信息	体系的能力阈值	
	体系韧性侧重因子	

附 录 B

（资料性附录）
装备体系要素分析

装备体系的要素有：

（1）探测要素

装备体系执行任务过程中，探测要素主要负责执行搜索、识别及获悉敌情等任务，具体而言如探测机、雷达、天基探测监视体系等。因此，探测要素是装备体系的重要组成要素，也是影响任务能力的重要因素。

（2）决策要素

装备体系在执行任务过程中，决策要素主要在战场上执行信息传输和分析、决策、辅助决策等任务，具体而言如决策体系、地面决策中心、任务管理中心等。因此，决策要素也是影响任务能力主要因素之一。

（3）执行要素

装备体系在执行任务过程中，火力单元要素主要是在战场上执行火力及电磁扰动等任务，具体而言如导弹、轰炸机、电磁扰动雷达等。作为任务链中行动的执行机构，执行要素也是影响任务能力的主要因素之一。

（4）敌方目标

装备体系在执行任务过程中，敌方目标单元是在战场上为完成我方任务需执行和扰动的敌方目标实体，具体而言如敌方的探测、决策、执行实体、基础设施。

附 录 C

（资料性附录）
装备体系韧性建模方法

C.1 装备体系节点模型

在装备体系的韧性建模中，依据装备体系的结构组成，将装备作为节点，具体分为四类节点

1）探测装备（V_S）：主要承担战场信息的获取与感知任务，其节点属性参量可表示为向量$v_S=\left[v_{S_1},v_{S_2},v_{S_3},v_{S_4}\right]$。其中，$v_{S_1}$代表探测装备的空间位置，$v_{S_2}$代表探测装备的最大探测距离，$v_{S_3}$代表探测装备的最大通信距离，$v_{S_4}$则用来判断探测装备是否处于正常工作状态。

2）决策装备（V_D）：负责任务决策，其属性参量可表示为向量$v_D=\left[v_{D_1},v_{D_2},v_{D_3}\right]$。其中，$v_{D_1}$代表决策装备的空间位置，$v_{D_2}$代表决策装备的最大通信距离，$v_{D_3}$则用来判断决策装备是否处于正常工作状态。

3）执行装备（V_w）：用于实施火力执行与目标摧毁，其属性参量可表示为向量$v_I=\left[v_{W_1},v_{W_2},v_{W_3},v_{W_4}\right]$。其中，$v_{W_1}$代表执行装备的空间位置，$v_{W_2}$代表执行装备的最大通信距离，$v_{W_3}$代表执行装备的最大射程距离，$v_{W_4}$则用来判断执行装备是否处于正常工作状态。

4）目标装备（V_T）：为战场上被探测或执行对象，其属性参量可表示为向量$v_T=\left[v_{T_1},v_{T_2},v_{T_3}\right]$。其中，$v_{T_1}$表示目标装备的空间位置，$v_{T_2}$表示目标装备的价值，$v_{T_3}$为目标装备的信息不确定性参数，其取值反映战场迷雾的影响，值越大表示情报可信度越低。

C.2 装备体系任务网建模

设红方作战装备节点集合为$R=\left\{r_1,r_2,\ldots,r_n\right\}$，蓝方作战装备节点集合为$B=\left\{b_1,b_2,\ldots,b_m\right\}$，其中$r_i$为第$i$红方节点，$b_j$为第$j$个蓝方节点，$n$为红方节点个数，$m$蓝方节点个数，$1\le i\le n$，$1\le j\le m$。构建探测、通信、决策、执行四层网络拓扑结构。

1）探测网络拓扑结构

探测网络以有向二分图$G_s=\left(R,B,E_s\right)$为数学模型，其中边集E_s由探测关系矩阵$P_1\in\left\{0,1\right\}^{n\times m}$定义。矩阵$P_{1_{ij}}$满足：

$$P_{1_{ij}}=\begin{cases}1, & \text{红方节点}r_i\text{可覆盖蓝方节点}b_j\\ 0, & \text{其他}\end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(C.1)}$$

该网络拓扑结构通过非零元素定位具备探测能力的红方节点及其探测覆盖范围，形成对任务目标的情报感知体系。

2) 通信网络拓扑结构

通信网络采用有向图$G_c=\left(R,E_c\right)$描述，其邻接矩阵$A_2\in\left\{0,1\right\}^{n\times n}$定义边集$E_c$。矩阵主对角线元素$A_{2_{ii}}=1$，表示节点自身通信能力；非对角线元素$A_{2_{ii}}=1$满足：

$$A_{2_{ii}}=\begin{cases}1, & \text{节点}r_i\text{与节点}r_j\text{可通讯}\\ 0, & \text{否则}\end{cases} \quad \text{(C.2)}$$

该网络拓扑结构通过矩阵的连通分量分析，可识别通信子网与信息传输路径。

3) 决策网络拓扑结构

决策网络基于有向图 $G_{c2}=(R,E_{c2})$ 构建，邻接矩阵 $A_3\in\{0,1\}^{n\times n}$ 定义节点间的决策关系。矩阵元素 A_{3_i} 满足：

$$A_{3_{ij}}=\begin{cases}1, & \text{节点}r_i\text{可决策节点}r_j\\ 0, & \text{否则}\end{cases} \quad \text{(C.3)}$$

可决策是指节点间存在指挥关系。通过矩阵的幂运算 A_3^k 可分析决策层级与指令传递路径，非零元素的分布界定了各节点的决策域与受控范围。

4)执行网络拓扑结构

执行网络以有向二分图 $G_a=(R,B,E_a)$ 建模，由执行关系矩阵 $P_4\in\{0,1\}^{n\times m}$ 确定边集 E_a。矩阵元素 $P_{4_{ij}}$ 满足：

$$P_{4_{ij}}=\begin{cases}1, & \text{红方节点}r_i\text{可攻击蓝方节点}b_j\\ 0, & \text{否则}\end{cases} \quad \text{(C.4)}$$

上述四层网络拓扑结构通过信息、指令与火力的跨层交互，构成复杂适应任务体系。探测网络获取的情报经通信网络传输至决策网络，支撑任务决策；决策指令通过通信网络下达至执行网络，实现对蓝方目标的精准执行，形成“探测-决策-执行”的闭环任务链路，显著提升装备体系的整体效能与对抗优势。如图C.2所示。

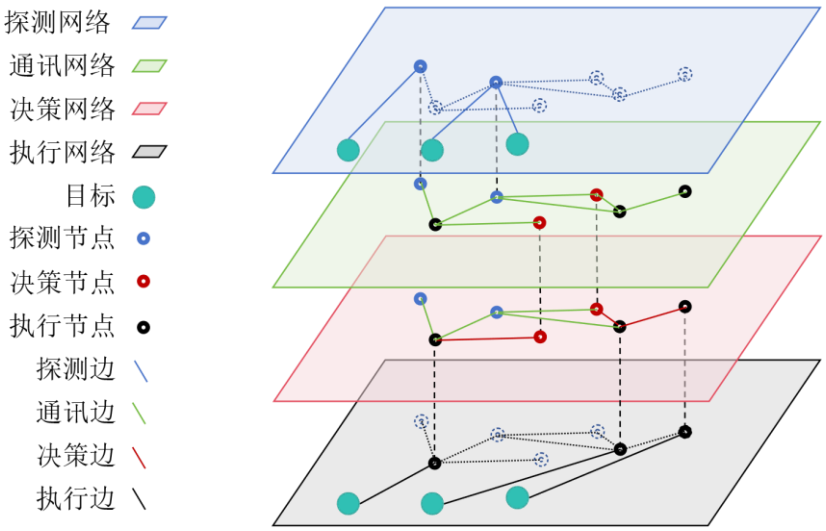

图 C.2　任务网模型

将探测（$P1$）、通信（$A2$）、决策（$A3$）、执行（$P4$）四个关系矩阵构成任务网络G，其中邻接矩阵A为 $A=\begin{bmatrix}P1 & & & \\ & A2 & & \\ & & A3 & \\ & & & P4\end{bmatrix}$。

C.3 装备体系任务链模型

根据装备体系中的实际连接关系，建立装备体系的任务链模型。

1） $T \to S \to D \to W \to T$ 典型任务链

2） $T \to S \to S \to D \to W \to T$ 具备信息共享的任务链

3） $T \to S \to D \to D \to W \to T$ 具备协同决策的任务链

4） $T \to S \to S \to D \to D \to W \to T$ 同时具备信息共享和协同决策的任务链

5） $T \to S \to D \to S \to D \to W \to T$ 具备信息反馈的任务链

6） $T \to S \to S \to D \to S \to D \to W \to T$ 具备信息共享和反馈的任务链

7） $T \to S \to D \to D \to S \to D \to W \to T$ 具备协同决策和信息反馈的任务链

C.4 装备体系任务链计算

依据七种装备体系任务链模型，使用深度优先搜索算法，计算装备体系的任务链，具体如表C.1所示。

表 C.1 装备体系任务链计算方法

	算法 1: 任务链计算算法：
1	**输入:** 初始网络 $D=(V,E,A)$， 装备集合(V_s,V_D,V_W,V_T)，网络的邻接矩阵 A；
2	**输出:** 任务网中的七种任务链
3	**for** (七种不同的任务链)**:**
4	**for** v_D **to** V_D **:** **if** (节点已访问) **continue**
5	**for** 当前任务链下的下一节点： **if** (不存在相连节点或已访问) 返回上一节点
6	**end for**
7	输出任务链
8	**end for**
9	**end for**

C.5 装备体系扰动方式

装备体系失效模式以随机失效与度失效模式两种，并分析两种扰动方式下的体系任务链变化。应记录装备体系的扰动开始时间。

C.6 装备体系能力恢复方法

围绕装备体系的冗余、修复、重构等手段，开展装备体系韧性恢复。

任务链间恢复：指任务链中某节点失效后，不同任务链间通过中继节点进行协同。该重构策略修改了装备体系间的通联方式，以恢复体系能力。

任务链内恢复：指任务链中某节点失效后，同一任务链内的同类进行协同恢复。该冗余策略修改了装备体系内的通联方式，以恢复体系能力。

节点自修复：指任务中某节点失效后，通过修复或新增节点提升体系能力的方法。按照节点失效前的度数大小顺序恢复。

应记录装备体系的恢复开始时间以及装备体系的恢复时间。

C.7 装备体系韧性量化模型

装备体系韧性的能力是量化装备体系韧性的关键参数，可通过仿真或者实际数据量化。具体计算方法为：

标准任务链能力：

$$C_{sc}=\frac{C_s*C_D*C_I}{C_T} \quad \text{(C.6)}$$

式中：

C_{sc}——标准任务链的能力；

C_s——探测装备的能力；

C_D——决策装备的能力；

C_I——执行装备的能力；

C_T——敌方目标装备的能力

广义任务链的能力：

$$C_{bc}=\frac{\left(1-\prod_{i=1}^{n}\left(1-C_{s_i}\right)\right)*\left(1-\prod_{j=1}^{m}\left(1-C_{D_j}\right)\right)*C_I}{C_T} \quad \text{(C.7)}$$

式中：

C_{bc}——广义任务链的能力；

C_{S_i}——广义任务链中的第 i 个探测装备的能力；

C_{D_j}——广义任务链中的第 j 个决策装备的能力。

装备体系的整体能力

$$Q(t)=\sum_{k=1}^{N_{op}}C_{bc_k}(t) \quad \text{(C.8)}$$

式中：

N_{op}——任务链的数量；

$C_{bc_k}(t)$——第 k 条任务链的能力。

附 录 D

（资料性附录）
装备体系韧性评估方法

D.1 能力驱动的韧性评估方法

装备体系的韧性曲线如图D.1所示

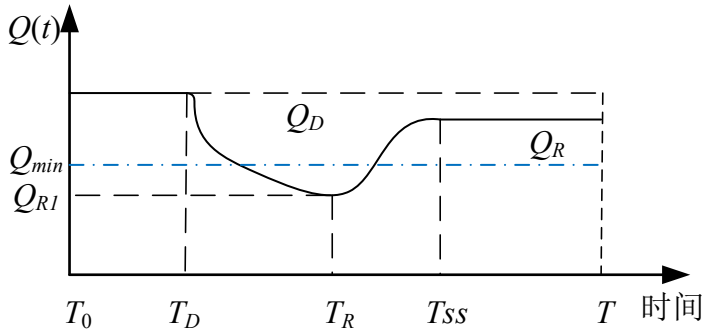

图 D.1 装备体系韧性曲线

其韧性值为：

$$R=\begin{cases}\sigma\rho\left[\delta+\zeta+1-\tau^{(\rho-\delta)}\right] & \text{如果}\rho-\delta\geq 0\\ \sigma\rho\left(\delta+\zeta\right) & \text{否则}\end{cases} \quad \text{(D.1)}$$

其中

$$\sigma=\frac{\sum_{T_0}^{T_{final}}Q(t)}{\left[Q_D\left(T_{final}-T_0\right)\right]} \quad \text{(D.2)}$$

σ：描述了扰动时间段内总能力相对于期望总能力的占比。

$$\rho=\frac{Q_{R1}}{Q_D} \quad \text{(D.3)}$$

ρ：恢复后稳定的能力相对于期望能力占比。

$$\delta=\frac{Q_R}{Q_D} \quad \text{(D.3)}$$

δ：描述了体系抵抗扰动的水平，δ越大抗毁性越强，δ越小抗毁性越弱。

$$\tau=\frac{T_{SS}-T_0}{T_{final}-T_0} \quad \text{(D.4)}$$

τ：描述了恢复速度的快慢，其越小，恢复越快。

$$\zeta=\frac{1}{1+\exp[-0.25\left(SNR_{dB}-15\right)]} \quad \text{(D.5)}$$

ζ：波动因子，描述了扰动过程中能力的波动程度，$0<\zeta<1$在正常理想情况下，$\zeta=1$。

D.2 任务驱动的韧性评估方法

任务驱动的韧性评估的计算方法构建如下：

$$R_{\alpha}(T)=\frac{\int_{T_0}^{T}Q(t)^{\alpha}\left[Q(t)-Q_{min}(t)\geq 0\right]dt}{\int_{T_0}^{T}Q_D(t)^{\alpha}\,dt} \quad \text{(D.6)}$$

式中：

$[P]$——艾弗森括号，仅当P为真时$[P]=1$，否则$[P]=0$；

$\alpha\in\{0,1\}$——为时间韧性和能力韧性之间的切换参数。

当$\alpha=0$时，$R_0(T)$为任务时间韧性度量：

$$R_0(T)=\frac{\int_{T_0}^{T}\left[Q(t)-Q_{min}(t)\geq 0\right]dt}{T-T_0} \quad \text{(D.7)}$$

表示在T_0到T的时间段内体系满足最低任务要求的时间总和与当前已进行任务时间段的比值。

当$\alpha=1$时，$R_1(T)$为任务能力韧性指标：

$$R_1(T)=\frac{\int_{T_0}^{T}Q(t)\left[Q(t)-Q_{min}(t)\geq 0\right]dt}{\int_{T_0}^{T}Q_D(t)\,dt} \quad \text{(D.8)}$$

该指标表征体系在任务时间内实际有效能力累积与最优（期望）能力累积比值。

因此结合时间和能力两个维度的韧性指标，根据对任务的评价倾向设计如下的体系综合韧性指标：

$$R(T)=\beta * R_0(T)+(1-\beta)R_1(T) \quad \text{(D.9)}$$

其中$0\leq\beta\leq 1$，为体系韧性侧重因子，有$0\leq R(T)\leq 1$。当体系工程师侧重于了解体系在遭受扰动后能够保持最低需求能力继续工作的能力时，可设置较大的β值，使综合韧性指标侧重于任务时间韧性；当体系工程师更加想了解体系遭受扰动后，在任务时间内能力恢复程度及累积情况可设置较小的β值，表示综合韧性值侧重于任务能力韧性。