| **上海脑虎科技有限公司** | | 起草人 |  |
| --- | --- | --- | --- |
| **软件设计流程规范** | | 审核人 |  |
| 文件编号 | NH-RD-WG-002 | 批准人 |  |
| 版本号 | A/0 | 颁发部门 |  |
| 生效日期 | 2024.10.14 | 分发部门 |  |

1. 目的

本程序的目的把软件开发过程中逐渐形成的作业规范，制定成一套软件开发规范，可以科学有效的规范软件开发流程，更好的防范系统的隐患。

1. 适用范围

本程序适用于质量管理体系下的所有医疗器械和生产场所。本程序适用于产品软件（系统和子系统）的策划、设计、开发和变更活动。

1. 定义与缩写词

SQA：Software Quality Assurance（软件质量保证）

SQC：Software Quality Control (软件质量控制)

1. 职责和权限

见NH -COP-050<软件开发控制程序>>。

1. 流程

流程图（维度：软件生命周期）：需求分析(软件需求规格书)—>开发计划—>概要设计—>详细设计—>编码和单元测试—>集成测试—>系统测试—>软件发布—>系统运维。见NH -COP-050<软件开发控制程序>>。

1. 规范内容

## 软件需求分析

软件需求设计是要完成《软件需求书》，即描述软件开发要“做什么”而不是“怎样实现” ；建立一个可以实现的、可确认的、可验证的基本依据。如果被开发软件只是一个大系统中的一个元素，那么软件代表根据《系统设计规格书》进行分析分解的，整个大系统也包括在软件需求的描述之中 ，必须包括系统运行环境 。软件需求分析是软件设计的开端，是最重要的一个节点，为项目计划、软件设计、实现和验证提供了基础保障。

### 软件需求包含的内容

1. 使用者需求和预期用途
2. 人机交互/易用性要求，例如包含：

* 手动操作
* 人机交互
* 对人员限制
* 需要人员高度注意的区域。
* 国际化

1. 用户文件需求
2. 使用者维护要求
3. 安全需求
4. 报警、警告、操作者信息。
5. 网络安全需求；例如包含：

* 授权和认证
* 通讯完整性
* 审核跟踪
* 敏感信息加密

1. 法规和标准要求
2. 功能性能需求
3. 功能和性能需求，例如包含：

* 性能和可扩展性。
* 物理特性例如代码语言、操作系统与平台。
* 环境特性例如硬件、网络基础设施、和内存大小。
* 与其它设备、版本、第三方软件或软件升级的兼容性需求。

1. 软件系统输入和输出，例如包含：

* 数据特性，如字符或数字的格式。
* 范围、限制和默认值。

1. 软件和其他系统间的接口要求；例如包含：

* 互操作性要求(DICOM, HL7, IHE)
* 硬件和网络接口

1. 数据定义和数据库要求，例如，格式、数据型式和功能。
2. 安装要求和验收标准。
3. 操作、维护和维修要求。
4. 对硬件故障、故障处理（错误定义、隔离和恢复）和潜在软件缺陷的软件措施。
5. 风险管理
6. 风险评估和控制文件中的软件风险降低措施。

注意：如果软件系统是独立系统（软件设备），软件系统需求和产品需求可能是相同的。

注意：在软件需求经过评审和审批后，软件需求或风险降低措施的变动，应执行设计变更。

### 软件需求分析流程

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 步骤 | 负责人 | 行动 |
| **设计输入** | 1 | 软件代表（工程师） | * 依据系统设计规格书设计软件需求书确保软件规格是： * 可以实现系统需求并降低风险； * 明确的和不矛盾的； * 可以验证和测试的； * 唯一识别； * 可追溯到系统需求或其他来源。 * 组织评审软件需求书 * 确保在设计历史文卷中对软件需求书进行归档。 |
| 2 | 软件经理 | * 对软件需求进行评审。 |
| 3 | 系统工程师 | * 根据软件需求分析的结果，更新需求跟踪表和产品风险管理文件。 * 批准软件需求书 |

### 需求分析的方法

1）功能分解方法。

2）结构化分析法。

3）信息建模方法。

4）面向对象的分析方法。

### 软件需求开发的表达方式

1）效果图描述

2）逻辑图

3）关系图

5）原型图

6）UML

### 软件需求书评审

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 软件代表 | 系统工程师、交互与UI工程师、测试代表、质量代表、产品经理、所有参与软件开发的人员。 | 《软件需求书》 |

## 软件概要设计

概要设计用于需求和设计之间搭建的一个桥梁，将软件需求转换成具体的设计解决方案的重要阶段。软件概要设计详细介绍了软件的结构组件、组件之间的关系和接口。 概要（架构）设计阶段：把软件按照一定的原则分解为模块层次，赋予每个模块一定的任务，并确定模块间调用关系和接口。

### 软件概要设计主要目标

产生软件概要设计说明书，说明系统模块划分、选择的技术路线等，整体说明软件的实现思路。并且需要指出关键技术难点等。

### 软件概要设计内容

软件概要设计应：

a) 将软件规格转化为概要设计文档，它描述了软件架构并定义软件模块。

b) 软件概要设计应定义软件模块之间，以及与外部部件之间的接口。

c) 对预期用途所必需的第三方软件，要定义其功能和性能要求。

d) 定义第三方软件需要的硬件及软件要求。

e) 定义对风险控制至关重要的软件架构策略（例如隔离和冗余）。

### 软件概要设计流程

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 步骤 | 负责人 | 行动 |
| **设计输出** | 1 | 软件代表（工程师） | * 创建软件概要设计并组织评审： * 概要设计实现了软件需求。 * 定义与风险控制，软件和硬件接口相关的要求。 * 确保在设计历史文卷中对软件概要设计进行归档。 |
| 2 | 软件经理（主管） | * 对软件概要设计进行评审和审批。 |

### 软件概要设计方法

1）总述：需求和目标、环境、局限；

2）总体设计：从全局的角度阐释，组织结构、功能、处理流程、有哪些模块、模块间的关系，运行环境等。（输出图：软件系统结构图、软件系统流程图、软件数据流程图/数据库类型等）

3）外部接口：说明外部用户、软件、硬件接口；

4）模块设计：每个模块的功能，简要说明“怎么做”（输入、输出、处理逻辑以及和其他模块的接口），在整个系统处于什么逻辑位置。

### 概要设计评审

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 软件代表 | 系统工程师、测试代表、所有参与软件开发的人员。 | 《软件概要设计说明书》 |

## 详细设计

软件详细设计是将软件概要设计分解为软件组件/单元及其接口。详细设计，是[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B" \t "_blank)中软件开发的一个步骤，就是对[概要设计](https://baike.baidu.com/item/%E6%A6%82%E8%A6%81%E8%AE%BE%E8%AE%A1)的一个细化，就是详细设计每个模块实现算法，所需的局部结构。在[详细设计](https://baike.baidu.com/item/%E8%AF%A6%E7%BB%86%E8%AE%BE%E8%AE%A1)阶段，主要是通过需求分析的结果，设计出满足用户需求的软件系统产品。

### 软件详细设计内容

详细的设计规范：核心算法、数据表示法、和软件单元间的接口、数据结构和软件产品的打包。

详细的设计应包含：

a) 软件概要设计的持续修正，直到它可以由软件单元表示。

b) 每一个软件单元的详细设计开发。

c) 软件单元和外部组件（硬件或软件）间接口的详细设计开发。

### 软件详细设计流程

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 步骤 | 负责人 | 行动 |
| **设计输出** | 1 | 软件代表（工程师） | * 创建和撰写软件详细设计。 * 满足软件详细设计的结构和方法 * 能够实现概要设计和软件需求。 * 与软件概要设计没有矛盾。 * 降低风险的措施能够实施。 |
| 2 | 软件经理（主管） | * 对软件详细设计进行评审和审批。 |

### 软件详细设计的方法

传统[软件开发方法](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E6%96%B9%E6%B3%95)的详细设计主要是用[结构化程序设计](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)法。

详细设计的表示工具有图形工具和语言工具。图形工具有业务流图、[程序流程图](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E6%B5%81%E7%A8%8B%E5%9B%BE" \t "_blank)。语言工具有伪代码等。

### 软件详细设计评审

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 软件代表 | 系统工程师、测试代表、所有参与软件开发的人员。 | 《软件详细设计说明书》 |

## 编码及规范

编码必须以设计文档为基础，必须保证所有的设计都被编码实现。当设计发生变更时，必须修改相关代码。

### 编码原则

1）必须保证设计文件和代码逻辑一致性。当代码修改，需要调整设计文件时，及时更新设计文档；  
 2）编码符合相应的编码规范具体语言参建具体语言的编程规范；

3）可维护性；

4）可复用性；

5）变量语义化；

6）设计模式的应用；

7）封装思想应用（模块化）；

8）避免全局变量；

### 编码的流程管理

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 步骤 | 负责人 | 行动 |
| **设计输出** | 1 | 软件工程师 | * 编写代码； |
| 2 | 软件经理（主管） | * 定期组织评审代码（关键核心逻辑和模块）； |
| 3 | 软件工程师 | * 提交代码； |

### 编码的规范

NH-RD-WG-003《C++语言编程规范》

NH-RD-WG-004《Java语言编程规范》

NH-RD-WG-005《JavaScript语言编程规范》

NH-RD-WG-006《Python语言编程规范》

NH-RD-WG-007《Verilog语言编程规范》

### 源代码管理规范（Git）

Git管理内容为软件项目所涉及到的源代码和软件项目文档、设计文档，均使用Git进行版本管理。

#### 存储库的结构

每个项目建立两个存储库：

Document：文档存储库；

SourceCode：工程源代码存储库。

* Document

Document下面按文档类型建立文件夹，软件文档包含软件项目中所编写的研发、测试、设计文档、通常文档由开发提交，供研发过程或后面查询使用。

一般文档的归档分为以下类别：

Design、SRS、Plan、HLD、DLD、Test、Other。

其中，“Design”存档交互与UI设计师输出的文档，一般包括：产品原型设计文档，产品效果图，产品UI切图、UI切图标注;

SRS、HLD、DLD分别对应软件研发过程中的需求分析、概要设计和详细设计；

Plan存储开发的进度表和开发计划；

Other用于存储项目中的其他文件，如注册文档等；

Test目录存储测试相关的文件，通常包括测试计划、测试报告、测试用例。

* SourceCode

软件项目的代码使用一个工程，采用main和develop分支的管理方式，建立main分支作为生产代码分支，生产代码分支是软件正式发布的代码分支，通过main分离出develop分支，develop分支是主要的日常开发分支，其他临时分支从develop分支分离。

一般代码文件夹分为以下类别：

Tools、XxxApp。

Tools文件夹存储工装软件，工具软件等源码；

XxxApp文件夹存储产品源码。

#### 权限

配置管理的流程及权限设置

|  |  |  |
| --- | --- | --- |
| 步骤 | 负责人 | 行动 |
| 1 | 软件代表 | * 申请产品配置目录建立 * 建立main\develop，其他人从develop分离创建其他开发分支 * 上传项目文档 |
| 2 | 源码管理人员 | * 按照申请建立配置目录 |
| 3 | 开发工程师 | * 编写源代码、上传 * 编译可执行代码，上传 * 文档上传 |

#### 分支的管理

其他分支的定义有两种：

1. 修改Bug后要合并到主干develop分支；
2. 特殊（定制版本）分支不合并到develop主干；

分支的命名：

1. 修改Bug分支以版本号命名：
2. 特殊分支以客户名称或功能名称命名。
3. 临时特性开发分支以feature-前缀及功能简称命名，临时分支合并到develop主分支后，及时删除。

版本发布：

1. 待发布版本从develop分离出RC\_A.B.C分支，其中A.B.C为完整版本号，基于该分支打包转测，在该分支上进行测试，修正Bug，后续Bug修正版仍在该分支上进行，并升级.C版本号，直到测试通过；
2. 每个打包均需要以完整版本进行Tag；
3. 测试通过的RC\_A.B.C合并到main生产分支，并进行tag，同时合并到develop分支并删除RC\_A.B.C分支。
4. 已发布上市版本的Bug修正，从main分支的对应版本如1.0.3建立分支1.0.4，多轮测试与Bug修正，版本号升级到最新版本如1.0.6，tag发布后，该分支保留不合并到main分支。

#### 注意事项

1. 代码要及时上传到代码管理服务器；
2. 提交前进行代码评审（关键模块）；
3. 提交代码写清楚修改原因（具体的修改描述要清楚，修改Bug的要列明Bug编号和Bug标题）；
4. 编译的中间文件不要提交代码管理服务器；
5. 不要上传自己的用户设置；
6. 文档由软件代表收集上传到Document库；
7. 文件提交错，及时进行回滚，必要时进行版本回退；

### 代码评审

代码评审也称代码复查，是指通过阅读代码来检查源代码与编码标准的符合性以及代码质量的活动，是软件开发的日常评审工作，由软件经理（主管）定期根据各项目的情况选择核心模块组织相关人员参与。代码评审内容：

编码规范问题：如命名不规范；

代码结构问题：如重复代码、巨大的方法和类、分层不当、紧耦合；

核心模板逻辑讲解评审，并提交代码评审表，见附件1《代码评审表》。

### 编译版本

软件开发的最终输出为可执行的软件程序。软件开发工程师做过单元测试后需要编译可执行程序，并编写软件说明。程序与软件版本说明一起提交集成测试。

软件校验码：软件校验码可以唯一地代表软件的版本特征数字化，并且是完整性验证。因此软件工程师在编译 后要用MD5工具生成校验码，并填写在软件版本说明中。

关于软件版本说明的格式要求见附件6《软件发布版本说明-模版》

## 软件测试

### 单元测试

软件在编码完成后进入单元测试。单元测试着重于软件以源代码形式实现地的各个单元，是针对软件设计的最小组成单位-程序模块，进行正确性检验的测试工作，它将在与程序的其他部分相隔离的情境下进行测试。其目的在于发现每个程序模块的内部在逻辑上和功能上可能存在的错误和缺陷。单元测试能验证程序和详细设计说明的一致性，通常在编码阶段进行，是软件工程师编写单元测试工具并执行，由软件工程师确保实施。

### 单元测试报告

通过测试工具输出单元测试报告，主要评估核心功能的验证及测试覆盖度。

### 集成测试用例

测试用例，指的是在测试执行之前设计的一套测试方法，包括测试环境（条件）、测试步骤、测试数据和预期结果。测试覆盖：功能测试、性能测试、负载测试、压力测试、关键接口的测试。软件代表在集成测试前需要完成测试用例编写，并评审：

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 软件代表 | 系统工程师、测试代表、软件经理（或主管） | 测试用例 |

### 集成测试及评审

集成测试着重于对软件的体系结构的设计和构造的测试，由软件工程师依照评审过的测试用例进行自测试，并输出集成测试报告，测试报告需要经过评审，是提交系统验证的依据。评审：

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 软件代表 | 系统工程师、测试代表、软件经理（或主管） | 测试报告 |

### 系统测试用例

系统测试用例，指的是在测试执行之前设计的一套测试方法，包括测试环境（条件）、测试步骤、测试数据和预期结果。测试覆盖：功能测试、性能测试、并发测试、压力测试、接口测试、内存测试、兼容性测试、用户界面测试、安装卸载测试、安全测试。测试代表在集成测试前需要完成测试用例编写，并评审：

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 测试代表 | 系统工程师、测试代表、软件经理（或主管） | 测试用例 |

### 系统测试及评审

系统测试着重于软件系统（由若干软件项组成）进行测试，通常采用黑盒测试，由测试工程师依照评审过的测试用例进行测试，并输出系统测试报告，测试报告需要经过评审，是提交系统验证的依据。评审：

|  |  |  |
| --- | --- | --- |
| 组织者 | 评审人员 | 工具 |
| 测试代表 | 系统工程师、测试代表、软件经理（或主管） | 测试报告 |

### 缺陷管理

开发过程中，任何与设计输入或设计输出不符的，或不可接受的风险，都视为缺陷，缺陷处理流程如下：



## 软件版本管理

版本是指源代码译后的可执行代码，版本的管理和建立仍是由软件代表负责。软件工程师负责提交源代码或执行程序。

### 版本定义及分类

软件版本：以字母、符号和数字构成的用于标识软件状态即视为软件版本。

根据产品本身的特点和质量管理体系的要求，同时考虑到监管的要求，即软件版本命名规则能够区分软件更新类型。

软件版本可分为软件发布版本和软件完整版本：

1. 软件完整版本：体现重大增强类软件更新、轻微增强类软件更新、纠正类软件更新、软件构建；

2. 软件发布版本：软件发行所用的标识版本，仅体现重大增强类软件更新（即重大软件更新）。

软件版本命名规则同样遵循风险从高原则，即同时发生重大、轻微软件更新按重大软件更新处理，同时发生增强类、纠正类软件更新按增强类软件更新处理。

### 命名规则

软件完整版本号分为3个部分，表示为： A.B.C。

软件发布版本号分为2个部分，表示为： A.B。

其中：

A为主版本号，表示重大增强类软件更新，初始值为0，当软件进行了重大增强类软件更新，该号码加1。

B为辅版本号，表示轻微增强类软件更新，初始值为0，当软件进行了轻微增强类软件更新，该号码加1。

C为修正版本号，表示纠正类软件更新，初始值为0，当软件进行了纠正类软件更新，该号码加1。

## 软件开发的输出

软件开发最终输出交付成果如下：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文件名称 | 文件负责人 | M0/M1 | M2 | M3 | M4 | M5-M7 | 注册检测文件 | 注册审评文件 | 注册体考文件 | 备注 |
| 软件开发计划 | 软件代表 |  | 归档 |  |  |  |  |  | ● |  |
| 软件需求书 | 软件代表 |  | 归档 |  |  |  |  |  | ● |  |
| 软件概要设计说明书 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件详细设计说明书 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件单元测试用例 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件单元测试报告 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件安装包 | 软件代表 |  |  |  | 归档 |  |  |  | ● |  |
| 软件集成测试用例 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件集成测试报告 | 软件代表 |  |  | 归档 |  |  |  |  | ● |  |
| 软件系统测试用例 | 测试代表 |  |  |  | 归档 |  |  |  | ● |  |
| 软件系统测试报告 | 测试代表 |  |  |  | 归档 |  |  |  | ● |  |
| 软件发布版本说明 | 测试代表 |  |  |  | 归档 |  |  |  | ● |  |
| 软件风险管理分析报告 |  |  |  |  |  |  |  | ● | ● |  |
| 软件可追溯性分析报告 |  |  |  |  |  |  |  | ● | ● |  |
| 软件技术要求 |  |  |  |  |  |  | ● | ● | ● | 独立软件时 |
| 软件使用说明书 |  |  |  |  | 归档 |  | ● | ● | ● | 独立软件时 |
| 软件研究资料 |  |  |  |  |  |  |  | ● | ● |  |
| 网络安全描述文档 |  |  |  |  |  |  |  | ● | ● |  |

## 软件质量保证

软件质量保证是建立一套有计划，有系统的方法，它通过对软件产品和活动进行评审和审计来验证软件是合乎标准的。软件质量保证的措施：

1）在开发标准和质量评价准则的指导下，软件设计时应用好的技术方法、制作高质量的软件需求书和设计文件

2）文档评审

3）代码评审

4）单元测试

5）集成测试

6）系统测试

7）控制变更

## 软件升级的管理

软件上市后发生变更，要基于软件的安全性和有效性判定是否是重大软件更新，采取相应的升级措施：

（1）重大软件更新：影响到医疗器械安全性或有效性的增强类更新，即重大增强类软件更新，应申请变更注册。

（2）轻微软件更新：不影响医疗器械安全性与有效性的增强类更新、纠正类更新，包括轻微增强类软件更新、纠正类软件更新，通过质量管理体系进行控制，无需申请变更注册，待下次变更注册时提交相应注册申报资料。

此外，软件构建是指软件编译生成一个工作版本，符合软件更新定义，通过质量管理体系进行控制，注册申报资料要求与纠正类软件更新相同。召回相关软件更新包括软件更新导致召回、召回措施所用软件更新，无论增强类更新还是纠正类更新均属于重大软件更新，按照医疗器械召回相关法规要求处理，不属于本指导原则讨论范畴。

软件更新遵循风险从高原则，即同时发生重大、轻微软件更新按重大软件更新处理，同时发生增强类、纠正类软件更新按增强类软件更新处理。软件更新需考虑引入回滚机制，以保证医疗业务的连续性，特别是对高风险软件。

重大软件更新判定原则如下：

软件更新若影响到医疗器械的预期用途、使用场景或核心功能原则上均属于重大软件更新，其判定原则包括但不限于：

（1）完善型软件更新：若影响到用户决策（含决策能力、决策结果、决策流程、用户行动）或人员（含患者、用户、其他相关人员）安全则属于重大软件更新，如软件的输入输出数据类型、体系结构、用户界面关系、物理拓扑、核心算法、核心功能、诊疗流程或预期用途等发生改变，软件系统、高风险软件项/软件单元进行代码重构，安全性级别改变，调整报警方式等；而运算效率单纯提高、诊疗流程或工作语言可配置化（即用户可保留原有诊疗流程或工作语言）、用户界面文字性修改、中低风险软件项/软件单元的代码重构等情况通常不视为重大软件更新，除非影响到医疗器械的安全性或有效性。

（2）适应型软件更新：若软件运行环境跨越互不兼容的计算平台（含硬件配置、外部软件环境、必备软件、网络条件）则属于重大软件更新，如预期运行的操作系统软件由Windows变为iOS，更换浏览器内核、必备软件，网络条件由局域网变为广域网，计算平台由通用计算平台变为医用计算平台等；预期运行的系统软件、支持软件、通用中间件的兼容性版本更新、补丁更新通常不视为重大软件更新，除非影响到医疗器械的安全性或有效性。

## 软件安全级别

根据《[医疗器械软件注册技术审查指导原则](http://www.cirs-group.com/md/news/news/9699.html)》的要求，对于含软件的医疗器械，均应在注册时提交《[软件研究报告](http://www.cirs-group.com/md/news/news/9926.html)》。而软件研究报告的详略程度，直接由软件的安全性级别和复杂程度来决定。因此，确定在编制一个医疗器械软件的研究报告时，首先要确定的就是该软件的安全性级别。

根据YY/T 0664《医疗器械软件软件生存周期过程》的要求，制造商应按照软件系统引起的危害对于患者、操作者或者其他人员的可能影响，赋予每个软件系统一个软件安全性级别（A、B或C）。

 基于如下的严重度，应初步赋予软件相应安全性级别：  
         A级（轻微）：不可能对健康有伤害和损坏；  
         B级（中等）：可能有不严重的伤害；  
         C级（严重）：可能死亡或严重伤害。

确定一个具体的产品的安全性级别的方法如下：

（1）按产品预期用途、使用环境和核心功能进行判定。

预期用途主要考虑软件的临床用途（如诊断、治疗、监护、筛查等）和重要程度（如重要作用、辅助作用、补充作用等），使用环境主要考虑软件的使用场所（如医院、家庭等）、疾病类型（如严重性、紧迫性、传染性等）、患者人群（如成人、儿童、老年、女性等）和用户类型（如专业用户、普通用户、患者等），核心功能主要考虑软件的功能类型（如控制驱动、处理分析等）、实现方法（如CT图像重建采用滤波反投影算法还是迭代算法，异常识别采用常规图像处理算法还是人工智能算法等）和复杂程度（如算法规模、参数数量、运算速度等）。

（2）按风险管理所确定的风险等级进行判定

对每个软件系统，在赋予安全性级别以前。均应应用C级要求。应假定可能引起软件系统未能像规定的那样起作用的失效的概率为100%。  
 如果软件失效可以引起死亡或严重伤害的风险，执行与软件安全性级别相匹配的软件生存周期过程，包括软件开发过程、软件维护过程、配置管理过程、风险管理过程和问题解决过程。同时，采用良好软件工程实践完善质量管理体系要求，保证软件质量。另外，应保证软件自身的信息安全，确保健康数据的保密性、完整性和可得性。由以上风险控制措施将风险降低到可接受水平（如YY/T0316所规定），或者降低软件失效后果或者降低由失效引起的死亡或严重伤亡的概率，软件安全性级别可从C级降低到B级；如果软件失效引起的非严重伤害风险同样通过上述风险控制措施降低到可接受水平，软件安全性级别可从B级降低到A。

8.附则

本规范由江西脑虎科技有限公司负责解释。

9.相关表单（附件）

附件1：代码评审表

附件2：软件开发计划-模版

附件3：软件需求书-模版

附件4：软件概要设计说明书-模版

附件5：软件详细设计说明书-模版

附件6：软件发布版本说明-模版

10.参考文件

| 分类 | 文件编号 | 文件名称 |
| --- | --- | --- |
| 外部 | GBT 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》 |
| 外部 | 国家药品监督管理局2022年第9号 | 《医疗器械软件注册技术审查指导原则（2022年修订版）》 |
| 外部 | 国家药品监督管理局2022年第7号 | 《医疗器械网络安全注册技术审查指导原则（2022年修订版）》 |
| 外部 | 中华人民共和国工业和信息化部令第　9号 | 《软件产品管理办法》 |
| 内部 | NH-COP-050 | 《软件开发控制程序》 |
| 内部 | NH-COP-051 | 《软件更新与版本管理控制程序》 |
| 内部 | NH-RD-WG-003 | 《C++语言编程规范》 |
| 内部 | NH-RD-WG-004 | 《Java语言编程规范》 |

**修订履历**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 文件名称 | 修改章节 | 修改后版次 | 修改人 | 修改日期 | 备注 |
| 01 | 《软件设计流程规范》 | 首次编辑 | A/0 | 崔俊杰 | 2024.10.14 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |