ICS 11.040.01
CCS C30

YY

中华人民共和国医药行业标准

YY/T 1406—20XX
代替 YY/T 1406.1—2016

医疗器械软件 GB/T 42062 应用于医疗器械软件的指南

Medical device software - Guidance on the application of GB/T 42062 to medical device software

（征求意见稿）

（本草案完成时间：2024-07-11）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

XXXX－XX－XX 发布　　XXXX－XX－XX 实施

国家药品监督管理局　　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。

本文件代替YY/T 1406.1—2016《医疗器械软件 第1部分：YY/T 0316应用于医疗器械软件的指南》，与YY/T 1406.1—2016相比，除结构调整和编辑性改动外，主要技术内容变化如下：

——更改标准名称为《医疗器械软件　GB/T 42062 应用于医疗器械软件的指南》；
——用 YY/T 0664—2020 及内容代替原 YY/T 0664—2008 及内容；
——用 GB/T 42062—2022 及内容代替原 YY/T 0316—2016 及内容；
——将第 1 章“总则”更改为第 1 章“范围”和第 2 章“规范性引用文件”；
——更改了文件的适用范围（见第 1 章，2016 版的 1.1）；
——将术语“安全相关软件”更改为“安全有关软件”（见 3.3，2016 版的 2.3）；
——将“风险管理通用要求”更改为“风险管理系统通用要求”（见第 4 章，2016 版的第 3 章）；
——将“包含软件的安全的系统的特征”更改为“包含软件的安全的系统的特性”（见 4.1.4，2016 版的 3.1.4）；
——将“人员资格”更改为“人员能力”（见 4.3，2016 版的 3.3）；
——将“编程经验和意向”更改为“编程经验和意见”（见 4.3.3，2016 版的 3.3.3）；
——将“软件开发计划(根据 YY/T 0664—2008) 的风险相关主题”更改为“软件开发计划(根据 YY/T 0664)风险相关的特定主题”（见 4.4.3，2016 版的 3.4.3）；
——更改了“风险管理文档”，增加了注（见 4.5，2016 版的 3.5）；
——将“医疗器械预期用途和与安全有关特征的识别”更改为“预期用途和可合理预见的误使用”（见 5.2，2016 版的 4.2），将“总则”更改为“预期用途”（见 5.2.1，2016 版的 4.2.1），增加了条标题“可合理预见的误使用”（见 5.2.2）；
——增加了“与安全有关的特性的识别”（见 5.3）；
——将“危险（源）识别”更改为“危险和危险情况的识别”（见 5.4，2016 版的 4.3）；
——将“估计每个危险情况的风险”更改为“风险估计”（见 5.4，2016 版的 4.3）；
——增加了按伤害的概率和严重度进行风险估计的内容（见 5.5.3）；
——删除了“降低风险”（见 2016 版的 6.1）；
——将“用设计方法取得固有安全”更改为“通过设计和制造获得固有安全”并增加相关内容（见 7.1.1.2，2016 版的 6.2.1.2）；
——将“安全信息”更改为“安全信息和用户培训”并增加相关内容（见 7.1.1.4，2016 版的 6.2.1.4）
——更改了“概述”中第一段有关采取硬件方面的措施的内容（见 7.1.2.1，2016 版的 6.2.2.1）；
——将“风险控制措施和软件结构性设计”更改为“风险控制措施和软件体系结构设计”（见 7.1.2.2，2016 版的 6.2.2.2）；
——将“防护性措施细节”更改为“防护措施细节”（见 7.1.2.3，2016 版的 6.2.2.3）；
——将“软件异常的风险控制措施”更改为“软件反常的风险控制措施”（见 7.1.2.5，2016 版的 6.2.2.5）；
——将“风险/受益分析”更改为“受益-风险分析”（见 7.4，2016 版的 6.5）；
——将“综合剩余风险的可接受性评价”更改为“综合剩余风险评价”（见第 8 章，2016 版的第 7 章）；
——将“风险管理报告”更改为“风险管理评审”（见第 9 章，2016 版的第 8 章）；
——增加了“总则”（见 10.1）、“信息收集”（见 10.2）、“信息评审”（见 10.3）和“措施”（见 10.4）；

——在资料性附录 B“软件原因的示例”部分的表 B.1 和表 B.2 中增加了与最新风险控制技术及风险管理实践有关的示例；
——在附录 D 的末尾增加了注。

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这些专利的责任。

本文件由国家药品监督管理局提出。

本文件由全国医疗器械质量管理和通用要求标准化技术委员会（SAC/TC221）归口。

本文件起草单位：

本文件主要起草人：

引　言

软件通常是医疗器械技术不可或缺的组成部分。建立包含软件的医疗器械的安全和有效性，需要知道软件的预期用途，同时要证明软件的实现满足这些预期用途且不引起任何不可接受的风险。

软件本身不是危险，但软件可能促成危险情况， 理解这一点很重要。宜总是以系统的视角考虑软件，软件的风险管理不能脱离系统孤立地进行。

复杂的软件设计可能涉及复杂的事件序列，这些序列可能促成危险情况。软件风险管理的任务主要包含识别那些能导致危险情况的事件序列，以及在这些事件序列中哪些位置可以中断序列，以预止伤害的发生或降低其发生的概率。

促成危险情况的软件事件序列可分为两类：

a)　事件序列表现为软件对输入的不可预见的响应(软件规范中的错误)；

b)　事件序列是由编码错误引起的（软件实现中的错误）。

由于正确地规范和实现复杂系统的难度以及完整验证复杂系统的难度，所以这些分类对软件来说是特有的。

因为很难估计会促成危险情况的软件反常的概率，并且因为软件在使用中不会因为损耗而随机失效，所以软件方面的风险分析宜关注可能导致危险情况的潜在软件功能和软件反常的识别——而不是估计概率。软件反常引发的风险大多数情况下仅需要评价伤害的严重度。

风险管理通常是有挑战性的，当涉及软件时更是如此。下面的条款包含了关于软件特性的额外的细节，这为从软件的视角理解 GB/T 42062—2022提供了指南。

本文件方框中的文本内容直接引用自GB/T 42062—2022标准，在文本的前面写明“GB/T 42062—2022原文”。

- **本文件的结构：**

本文件遵循了GB/T 42062—2022的结构，并为与软件有关的风险管理活动提供了指南。

由于软件生存周期中风险管理活动的迭代特性，在提供的信息中存在一些有意的冗余。

医疗器械软件　GB/T 42062 应用于医疗器械软件的指南

1　范围

本文件为GB/T 42062—2022中包含的要求应用于YY/T 0664—2020中所指的医疗器械软件（独立软件和软件组件）提供了指南，本文件不增加或改变GB/T 42062—2022或YY/T 0664—2020的要求。与此同时，GB/T 42062—2022和YY/T 0664—2020的内容为本文件提供了基础。

本文件在医疗器械/系统中包含软件时供需要实施风险管理的风险管理从业者，和需要理解如何满足GB/T 42062—2022中阐述的风险管理要求的软件工程师使用。

宜说明的是，虽然GB/T 42062—2022和本文件关注的是医疗器械，但本文件可用于对医疗保健环境中的所有软件实施安全风险管理过程，而无论其是否被归类为医疗器械。

本文件不涉及：

——已由现有标准或规划中的标准所覆盖的领域，如：报警、可用性工程、网络；

——生产或质量管理体系软件；

——软件开发工具。

不预期将本文件作为法规检查或认证评定活动的依据。

本文件中“宜”用来表示，在满足要求的若干可能中，推荐特别适合的一种，并未提及或排斥其他的可能性，或者用来表示某种做法更好但不是必须的要求。“宜”不应理解为要求。

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。

YY/T 0664—2020 医疗器械软件　软件生存周期过程（IEC 62304:2015，MOD）

GB/T 42062—2022 医疗器械　风险管理对医疗器械的应用(ISO 14971:2019,IDT)

3　术语和定义

GB/T 42062—2022和YY/T 0664—2020界定的以及下列术语和定义适用于本文件。

注：定义术语的索引开始于48页。

3.1

多样性　diversity

一种冗余的形式，其中冗余要素使用不同的（多样的）组件、技术或方法以降低共同原因导致所有要素同时失效的概率。

3.2

冗余　redundancy

提供多个组件或机制来完成同样的功能，使得一个或多个组件或机制的失效不会妨碍功能的执行。

3.3

安全有关软件　safety-related software

能够促成危险情况的软件，或用于实施风险控制措施的软件。

4　风险管理系统通用要求

4.1　风险管理过程

4.1.1　总则

GB/T 42062—2022 原文

4 风险管理系统通用要求

4.1 风险管理过程

制造商应建立、实施、形成文件和保持持续的过程，以用于：

a） 识别与医疗器械相关的危险和危险情况；

b） 估计和评价相关的风险；

c） 控制这些风险；

d） 监视风险控制措施的有效性。

此过程应应用于医疗器械的整个生命周期。

此过程应包括下列要素：

——风险分析；

——风险评价；

——风险控制；

——生产和生产后活动。

当存在形成文件的产品实现过程时，该过程应包括风险管理过程的适当部分。

注1：产品实现过程的描述，例如GB/T 42061—2022[7]的第7章。

注2：质量管理体系中形成文件的过程能用于系统性地处理安全问题，特别是能够早期识别复杂医疗器械中的危险和危险情况。

注3：风险管理过程的示意图见图1。基于特定的生命周期阶段，风险管理的各要素可能有不同的侧重点。此外，针对医疗器械的不同情况，风险管理活动可能迭代执行或在多个步骤中执行。附录B包含了风险管理过程中各个步骤更详细的概述。

用查看适当文件的方法检查符合性。

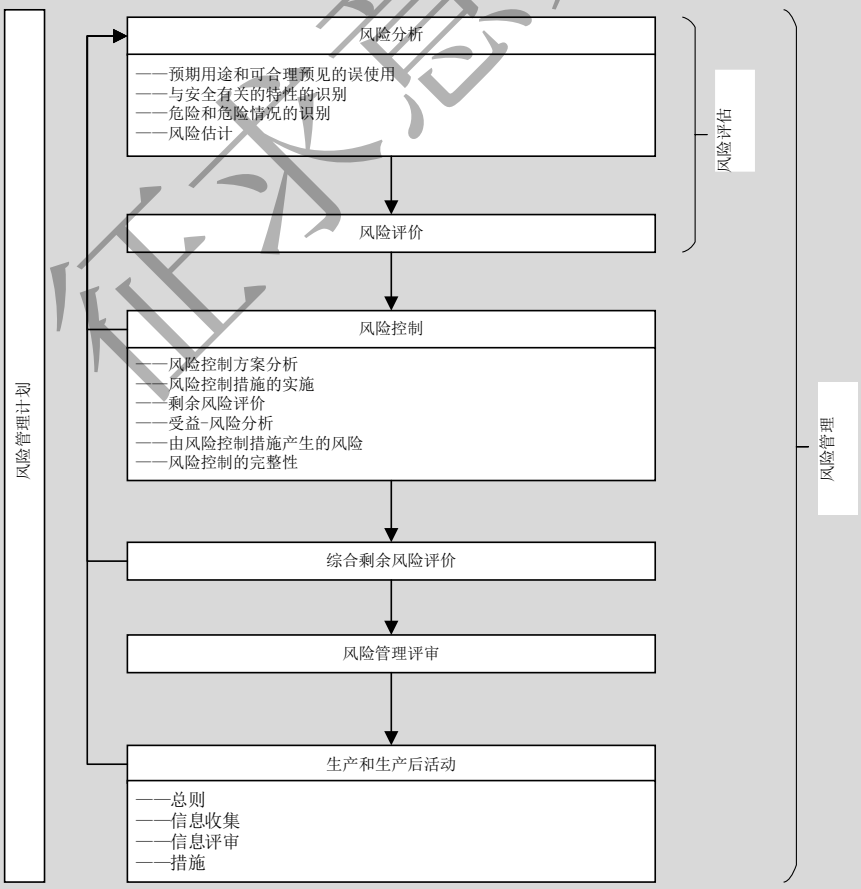


图 1：风险管理过程示意图

安全是系统（这里指整个医疗器械）的一个属性，该系统可能包括软件。风险管理宜在包括软件及其全部硬件环境的系统上进行。软件风险管理活动不宜脱离系统进行。

虽然软件方面的风险管理脱离整体医疗器械风险管理不能有效地进行，但有一些活动作为软件生存周期的不可或缺的组成部分由软件工程师完成可能是最好的。与用于整体医疗器械风险管理（GB/T 42062—2022）的要素相比，软件中的有些要素要求更多的关注并有不同的解释。为了做到有效，需要强调的是软件风险管理也需要关注医疗器械风险。

注1：脱离整体医疗器械的风险管理，软件方面的风险管理无法有效地进行，因为硬件失效、软件失效以及硬件和软件的风险控制措施的互相依赖。

注2：例如，所有的软件失效都是系统性的而非随机的（许多硬件失效/故障是随机的），且其失效的概率无法精确估计。因此，风险的概率要素应用于软件的方式是非常不同的，见本文5.5.3。

在医疗器械设计的早期阶段，软件工程师有很多机会致力于医疗器械的总体安全。宜在系统设计确定之前考虑软件在医疗器械安全方面的作用。

通过参与到医疗器械设计过程，随着设计进展，软件工程师有助于就软件有关风险做出与安全有关的决定。这些决定宜包括但不限于：

——提供足够的硬件资源以支持软件；

——硬件和软件间功能的划分；

——整体医疗器械的预期用途和软件用户界面的预期用途；

——避免非必要的复杂软件。

4.1.2 迭代

典型的软件开发生存周期经常使用迭代。迭代的使用允许：

——研究不同软件设计的可行性；

——在不同时期开发不同的软件项；

——软件不同版本的阶段性交付；

——纠正软件开发过程中产生的错误。

YY/T 0664—2020要求风险管理活动在整个软件生存周期中迭代并与系统设计活动相协调。例如，在软件开发期间，YY/T 0664—2020的5.2.4要求在软件需求确定后对医疗器械风险评估进行重新评价。这种重新评价可能导致系统需求规范和医疗器械风险评估的更新。风险评价宜在需求、体系结构设计和详细设计直至软件实现的所有阶段重复进行。

GB/T 42062没有规定设计和开发过程，其通常只要求在设计（包括风险控制措施）实现之前和之后（而不是设计期间）完成风险管理步骤。例如，当一个风险控制措施已被实施，GB/T 42062要求对其进行评审以确保其没有引起任何进一步的危险和危险情况。这不宜理解为仅在实施完成之后才要求进行这种评审——比较有利的做法是：一旦进一步的危险显现出来就要进行处置。这意味着在风险控制措施实施过程中进行迭代。

所有的交付物始终保持一致是很重要的。迭代对交付物的一致性是一种威胁。因此，应用严格的配置管理很重要，以保证变更的所有影响都被识别，且所有相关的交付物在变更后得到更新。当涉及软件时，这一点尤其重要，因为软件可能很快变更，一个看上去小的变更可能产生非预期的负面影响。与软件有关的所有信息都要是最新的，以避免工程师间的错误交流。宜检查软件变更提案的负面影响，特别是影响安全的负面影响。这可能导致风险管理过程的某些部分重复进行。

4.1.3 主动的或被动的安全设计方法

风险管理宜随着医疗器械规范的实质性输入尽早开始，在设计早期阶段考虑安全，也就是说，主动的设计方法比被动的设计方法更好。在主动方法中，安全与其他客户需求一起被考虑，并转化为早期的安全需求。尽管被动方法有时不可避免（例如在更新遗留产品时），但主动方法通常是获得安全的医疗器械最有效、最快捷和最经济的方法。

主动的安全设计的优点是：

——从一开始系统规范不仅包含医疗器械要做什么，也识别宜避免的系统行为以降低风险；

——从一开始就可以策划一个系统的体系结构，可以证明其在避免或防止不安全状态的同时提供想要的功能特性；

——因为体系结构已被精心细致地融入全部的设计中，能在避免返工的同时开发风险控制措施；

——能在早期作出安全方法和风险控制措施的选择（例如，可使通过设计的固有安全最大化，并使安全信息最小化）。

4.1.4 包含软件的安全的系统的特性

安全的系统高度期望的特性包括：
——使用简单的硬件安全机制以避免对与安全有关的软件项的过度需求；
——仅使用非常简单的安全有关软件项；
——安全有关软件项分布在若干独立的处理器中；
——足够的硬件资源，以在需要时运行所有安全有关软件，避免资源争夺；
——使用确定性软件时序设计；
——失效情况下的适当处理，例如：
- 警告用户有失效并留出知情干预的机会；
- 在失效的情况下提供简化功能；
- 在失效的情况下，可能时，安全地关断；
- 从失效中快速恢复；

——防止软件代码在其运行环境中通过自行修改或作为数据输入的结果被修改的方法；
——检出和/或防止与安全有关的数据损坏的方法。

4.2 管理职责

GB/T 42062—2022原文

4．2 管理职责

最高管理者应通过确保下列活动对风险管理过程的承诺提供证据：
——提供充分的资源；
——为风险管理分配能胜任的人员（见4.3）。
最高管理者应规定用于建立风险可接受性准则的方针并形成文件。此方针应提供一个框架确保准则基于适用的国家或地区法规和相关标准，并考虑可获得的信息，例如普遍公认的最新技术水平和已知的利益相关方关注点。
注1：制造商用于建立风险可接受性准则的方针能够规定风险控制的方法：合理可行地将风险降至最低，合理可实现地将风险降至最低，或对受益-风险比无不利影响的情况下尽可能降低风险。关于规定方针的指南见YY/T 1437[10]。
最高管理者应按照策划的时间间隔评审风险管理过程的适宜性，以确保风险管理过程的持续有效性，并且应将任何决定和采取的措施形成文件。如果制造商具有已建立的质量管理体系，这些评审可作为质量管理体系评审的一部分。
注2：对生产和生产后信息的评审结果能作为风险管理过程适宜性评审的输入。
注3：本条款描述的文件能整合到制造商质量管理体系产生的文件中，且风险管理文档中能引用这些文件。
用查看适当文件的方法检查符合性。

GB/T 42062—2022和YY/T 0664—2020 都设定已建立质量管理体系。GB/T 42062—2022 的4.2列出了风险管理对最高管理者的要求。

注：GB/T 42062—2022的4.1规定风险管理可以是质量管理体系的一个不可或缺的组成部分，并且YY/T 0664—2020的4.1规定可以通过使用符合GB/T 42061[2]或国家法规要求的质量管理体系来证明制造商始终如一地满足客户需求和适用的法规要求的能力。YY/T 0664—2020的附录B.4中也提供了关于4.1条款的指南，说明有必要作为质量管理体系的一个不可或缺的组成部分建立风险管理过程,作为应用适当的软件工程方法和技术的整体框架。

为实现有效的风险管理过程以及对医疗器械软件的安全设计和维护，最高管理者负责将必要的组织结构、充分的资源、职责和培训（见本文4.3）落实到位。

制造商可以考虑将软件开发或维护过程活动外包（例如设计、实现、测试或维护）。在这些情况下，最高管理者仍然对确保外包的软件开发或维护过程活动进行适当的风险管理活动，并对确保风险控制措施的适当应用负全部责任。

当软件开发外包时，制造商宜通过适当的合同来确保对软件及其设计有充分的控制，以确保GB/T 42062中要求的全部风险管理在医疗器械整个生存周期中都得到执行，包括软件发布后软件反常的纠正。

制造商宜考虑对供方设定实施要求（见针对供方控制的GB/T 42061的7.4），比如要求供方证实：

——符合 GB/T 42062 的有效的风险管理；

——符合 YY/T 0664 的有效的软件工程实践；

——提供持续满足顾客要求和适用的法规要求的医疗器械软件的能力。

如果有应用于外包过程或产品的风险控制措施，那么风险控制措施及其重要性宜形成文件并在合同中向供方明确传达。

4.3 人员能力

4.3.1 总则

GB/T 42062—2022 原文

4．3 人员能力

基于与所分配的任务相适应的教育、培训、技能和经验，执行风险管理任务的人员应是能够胜任的。适当时，这些人员应具备特定医疗器械（或类似医疗器械）及其使用的知识和经验、有关的技术或风险管理技术。应保留适当的记录。

注：风险管理任务可能由几个职能的代表执行，每个代表贡献其专业的知识。

用查看适当记录的方法检查符合性。

参与软件系统开发和维护的小组成员宜具有与其承担的任务相适应的知识和经验。对承担风险管理相关任务的人员的基本要求是具有必需的风险管理知识。包括临床专家（诸如临床支持和技术服务专家以及其他相关学科的专家）、软件工程师、系统设计师、可用性/人因工程专家和领域专家的多学科团队的参与，以及他们与软件工程人员和测试人员互动的程度和类型也宜从风险管理角度加以考虑。

这可能要求为每一个人制定培训计划以确保对所要求活动的完全理解。

同样，也宜考虑风险管理小组成员在软件方面的资格而且可能需要专门的培训。

以下条提出了宜被考虑的所需知识领域的综述。

4.3.2 预期用途/领域知识

在医疗器械设计的所有阶段，有效利用预期用途的知识很重要。这对于软件设计者和软件风险管理人员尤其重要。软件的复杂行为很容易促成用户误用或混淆，导致以前未预见的危险和危险情况。对临床实践彻底的领会能使风险管理者识别危险和危险情况，使软件工程师避免危险和危险情况，或设计出风险控制措施。

制造商宜确保临床专家（诸如临床支持和技术服务专家及其他相关学科专家）能参与设计活动和风险管理活动，或至少对这些活动提供建议。

另外，制造商宜考虑对软件工程师和风险管理者提供医疗器械临床应用方面的培训。

4.3.3 编程经验和意见

有经验的软件开发人员和测试人员能逐渐认识到在测试中发现所有软件缺陷的难度，因此也能认识到测试后还存在一定比例的软件缺陷。在软件开发团队中包含有经验的人员并且给予其适当的权限来指导、监督和质疑经验少的人员是很重要的。

为以下软件任务分配有经验的人员尤其重要：

——软件可能的失效方式的识别；

——与软件失效相关的风险分析；

——风险控制措施的识别；

——发布后的问题报告分析；

——变更的设计和实现，特别是软件发布以后。

在所有这些任务中，通过经验可以意识到软件及软件开发过程什么会出错，并且认识到进行变更的同时保持软件设计的完整性的难点。

4.4 风险管理计划

4.4.1 总则

GB/T 42062—2022 原文

4.4 风险管理计划

风险管理活动应得到策划。对于所考虑的特定的医疗器械，制造商应按照风险管理过程建立风险管理计划并形成文件。风险管理计划应是风险管理文档的一部分。

此项计划至少应包括下列内容：

a）策划的风险管理活动范围；

b）识别和描述医疗器械和计划中每个要素所适用的生命周期阶段；

c）评审风险管理活动的要求；

d）风险可接受性准则，基于制造商用于确定可接受风险的方针，包括在伤害发生概率不能估计时的接受风险的准则；

注1：风险可接受性准则对于风险管理过程的最终有效性至关重要。对于每个风险管理计划，制造商需要建立适合特定医疗器械的风险可接受性准则。

e）综合剩余风险评价的方法和基于制造商确定可接受风险的方针的综合剩余风险的可接受准则；

注2：综合剩余风险评价的方法可能包括收集和评审所考虑的医疗器械和市场上类似医疗器械的数据和文献，以及由具备应用知识和临床专业知识的跨职能专家团队进行判断。

f）风险控制措施实施及其有效性的验证活动；

g）与相关的生产和生产后信息的收集和评审有关的活动。

注3：制定风险管理计划和建立风险可接受性准则的指南见YY/T 1437[10]。

注4：并非计划的所有部分都需要同时制定，可能随着时间而制定计划或计划的一部分。

在医疗器械的生命周期内如果计划发生更改，应在风险管理文档中保留更改的记录。

用查看风险管理文档的方法检查符合性。

风险管理计划宜通过下列各项阐述软件是医疗器械一部分的事实：

——医疗器械的描述，包括医疗器械的何种功能将由软件实现；

——软件将依据 YY/T 0664 开发的声明；

——对软件风险管理特有的软件开发内容（见注）的引用；

——软件导致或软件控制的风险的可接受性准则（如果与医疗器械其他组件的可接受性准则不同）。

注：对软件开发计划的引用也许是表明包含软件风险管理特有的软件开发内容的最简单方式。见4.4.2和4.4.3，其讨论了风险管理计划和软件开发计划的关系及依据YY/T 0664标准的软件开发计划中与风险相关的特定主题。

软件导致或软件控制的风险的可接受性准则可能与其他组件的风险可接受性准则不同，一个原因是其伤害的概率不能估计。在这种情况下，风险可接受准则宜基于伤害的严重度（见本文5.4.3关于软件导致伤害的概率的讨论）。如果能断定危险的实际后果很小，风险就可以判定为可接受，也不必有风险控制措施。然而，对于重大危险，即能造成高严重度伤害的危险，任何级别的暴露水平都不能被认为其相应的风险低到可接受。在这种情况下，需要实施风险控制措施。

当概率无法估计时，剩余风险的可接受性准则宜考虑已实施的风险控制措施及其在降低伤害发生概率方面的有效性。风险控制措施宜是所有合理可行措施的组合，满足适用的标准和法规，并符合最新技术水平（见YY/T 1437—2023[3]）。

当策划与生产和生产后信息的收集和评审有关的活动时，对于软件宜考虑以下特定要求：

——如果使用未知来源软件（SOUP），宜对积极监视和评价可公开获得的反常清单及关于 SOUP 实际性能的信息作出规划。如可能，在获得 SOUP 的同时，宜与 SOUP 供应商达成协议以支持以上活动。如果医疗器械的使用者可自行（有意地或无意地）修改医疗器械的 SOUP（例如，SOUP 补丁或升级包），那么宜特别关注和监视市场上提供的 SOUP 新版本。关于 SOUP 及生产后监视见第 10 章。

——制造商宜使投诉发起人有可能识别和报告软件的版本。

4.4.2 风险管理计划与软件开发计划的关系

GB/T 42062对风险管理计划和YY/T 0664对软件开发计划的要求不宜被视为要求具有特定标题的特定文档。计划的要素可以包含在任何文档中以适应制造商的质量管理体系，只要：

——计划的多文档结合满足两个标准要求并可验证；

——所有计划彼此一致；

——所有计划能及时提供使用；

——所有计划保持更新以反映不断变化的环境。

4.4.3 软件开发计划(根据 YY/T 0664)风险相关的特定主题

软件开发计划宜确保软件开发过程，与软件开发相关的标准、方法和工具（依据YY/T 0664—2020的第5章，在软件开发计划中描述）都是有效的风险控制措施（见7.1.2.6关于过程作为风险控制措施的讨论）。这可以通过由其他组织、供方和组织内的其他项目提供证据来实现。否则，就在本项目内策划并验证其有效性。

当建立医疗器械风险管理过程时，宜考虑软件风险管理特有的方面，如安全编码标准、验证方法(如：形式化证明、同行评审、走查、仿真等），以及语法和逻辑检查器的使用。如果考虑将这些方面作为风险控制措施，也宜对其进行验证（见表B.2 验证风险控制措施的示例）。

适当时，宜在计划、程序或培训中对医疗器械开发每个阶段的软件风险管理活动进行说明。

4.5 风险管理文档

GB/T 42062—2022 原文

4.5 风险管理文档

对所考虑的特定医疗器械,制造商应建立和保持风险管理文档。除本文件其他章节的要求外，风险管理文档应提供对于每个已识别的危险的可追溯性：

——风险分析；

——风险评价；

——风险控制措施的实施和验证；

——剩余风险评价的结果。

注1：构成风险管理文档的记录和其他文件，能作为其他文件和文档（例如制造商的质量管理体系所要求的文件和文档）的一部分。风险管理文档不需要物理上包含所有的记录和其他文件，然而至少需要包含所有要求文件的引用或指向，以便制造商能够及时地收集风险管理文档中引用的信息。

注2：风险管理文档能采用任何形式或类型的媒介。

注3：对未使用ISO 14971设计的组件和器械，建立风险管理文档的指南见YY/T 1437[10]。

软件过程宜建立体系以使以下可追溯性成为可能，即从软件有关的危险和软件风险控制措施开始，并追踪其实施，直至相应的安全有关软件的需求以及满足那些需求的软件项。

上述所有活动都宜能够追溯至其验证（见YY/T 0664—2020的 7.3.2）。

因为软件在开发期间可频繁变更，并因为其可以不同的版本发布，与软件有关的那部分风险管理文档也可能变更并有多个版本。

表1列出了除GB/T 42062—2022的要求外，宜包括在风险管理文档中的，YY/T 0664—2020对文件的要求。

表1 除 GB/T 42062—2022 的要求外，宜包括在风险管理文档中的文件要求

YY/T 0664—2020 的条	风险管理文档文件
4.3 b）	赋予每个软件系统的软件安全级别
4.3 e）	对于软件系统中不执行安全相关功能的软件项采用较低软件安全级别（相比软件系统）的理由的说明

7.1.4	软件项促成危险情况的潜在原因
7.2.1	为软件项促成危险情况的每个潜在原因，规定风险控制措施
9.5	保留问题报告及其解决情况的记录，包括对其验证的记录。 适当时，更新风险管理文档。

注：软件安全级别的判定准则见 YY/T 0664—2020 的 4.3。此外，相关法规指南文件[7]中给出了对医疗器械软件安全进行分级的其他方法。

5 风险分析

5.1 风险分析过程

GB/T 42062—2022 原文

5.1 风险分析过程

制造商应按照5.2～5.5中的描述对特定的医疗器械进行风险分析。策划的风险分析活动的实施和风险分析的结果应记录在风险管理文档中。

注1：如果有类似医疗器械的风险分析或者其他有关信息可获得时，则该分析或信息能够作为新的风险分析的起点。这种有关程度取决于医疗器械之间的差别，以及这些差别是否会造成新的危险，或者造成输出、特性、性能或结果的重大差异。对于现有风险分析的利用程度取决于这些差异可能对发生危险情况影响的系统性评价。

注2：关于供选择的风险分析技术和体外诊断医疗器械风险分析技术的指南见YY/T 1437[10]。

除了5.2～5.5中要求的记录以外，风险分析实施和结果的文件还应至少包括：

a） 识别和描述所分析的医疗器械；

b） 识别实施风险分析的人员和组织；

c） 风险分析的范围和日期。

注3：风险分析的范围可能非常宽泛（例如对于新医疗器械的开发，制造商知之甚少或没有经验），也可能对范围进行限制（例如分析更改对现有医疗器械的影响，该器械的更多信息已经存在于制造商的文档中）。

用查看风险管理文档的方法检查符合性。

如GB/T 42062—2022所述，术语风险分析包含三项不同的活动：

——预期用途的识别；

——已知或可预见的危险（及其原因）的识别；

——估计每个危险和危险情况的风险。

必须认识到风险分析是整个软件开发过程不可或缺的一部分，而不是一个或两个独立的事件，这对于其有效性是非常重要的，因为危险和失效模式信息在软件开发生存周期过程中不断累积并需要在设计的每个阶段加以考虑。

因为很难估计能促成危险情况的软件反常的概率，软件方面风险分析的关注点主要是对可能导致危险情况的潜在的软件功能和反常的识别—而不是估计概率。关于估计概率的更多详细描述见5.4.3。

软件作为促成因素的最坏情况下伤害的严重度是确定软件开发过程严格程度的主要输入（见 YY/T 0664—2020的4.3）。5.2、5.3和5.4中提供的信息旨在帮助识别有效的风险管理过程中软件特有的方面。另外，软件方面的风险分析在形成的文档中宜是可识别的，且宜包括用来对硬件失效实施风险控制措施的软件，以及导致危险的软件原因和相关的风险控制措施。

5.2 预期用途和可合理预见的误使用

GB/T 42062—2022 原文

5.2 预期用途和可合理预见的误使用

制造商应将所考虑的特定的医疗器械的预期用途形成文件。

预期用途宜考虑的信息包括如预期的医学适应证、患者群体、与医疗器械交互的身体部位或组织类型、用户特征、使用环境和工作原理。

制造商也应将可合理预见的误使用形成文件。

该文件应保留在风险管理文档中。

注1：使用规范（见IEC 62366-1：2015，3.23[14]）能作为确定预期用途的一项输入。

注2：关于确定预期用途时需要考虑的因素和对可合理预见的误使用的解释见YY/T 1437[10]。

用查看风险管理文档的方法检查符合性。

5.2.1 预期用途

每个医疗器械都有其预期用途，同时还宜考虑到对这些用途存在有意或无意的误用的可能。尽管这不是软件特有的问题，但软件的使用可导致误用风险的增加，因为：

——医疗器械的行为更加复杂，因此更加难以掌握或理解；

——使用者可能对软件过分依赖，而不理解其局限性；

——医疗器械可能是可配置的，而使用者可能没有注意到当前配置；

——医疗器械可能与其他医疗器械或非医疗器械以其制造商无法详细预期的方式进行通信。

负责生成系统需求的人员和软件工程师有一个共同的责任——将包括软件在内的系统的预期用途以及与安全和安全使用有关的所有系统和软件需求一起记录在风险管理文档中。软件工程师具体负责识别在系统水平上过于细微而不明显的预期用途部分。

5.2.2 可合理预见的误使用

5.2.2.1 用户界面

软件使设计更加灵活的用户界面成为可能，而这可影响用户的行为，导致可合理预见的误使用的新的形式。通常的误用源于对过于复杂的用户界面的误解和为避免错误和不安全状态而过分依赖软件。重要的是预见这些误使用和改变设计以尽可能地避免这些误使用。

这包括多语言标记的实现，特别是当这种标记是风险控制措施时。以下几点宜特别注意：

a) 不同语言对存储空间的不同需求；

b) 不同字符集的使用；

c) 使用字符代替符号；

d) 使用不同的单位可能需要对数据结果进行额外的换算；

e) 日期格式和数字标点；

f) 不同语言和/或字符集不同的布局要求；

g) 对确认的支持。

作为对GB/T 42062的补充，可用性过程的内容见IEC 62366-1[5]。

5.2.2.2 医疗器械的相互连接

医疗器械软件的使用使医疗器械和非医疗器械之间各种相互连接和相互通信成为可能。这些连接和通信很可能产生由医疗器械和所连接器械组成的系统的新应用（和误用）。虽然容易预见到可发生这种新的应用和误用，但如果这些连接和通信不受限制，医疗器械制造商就不易识别所有这种应用和误用。

因此，制造商对医疗器械的通信接口规定有限的预期用途，并在设计接口时尽可能将连接和通信限制在安全的范围，这是很重要的。

例如，软件利用医疗器械内置接口，基于对用户、患者的识别以及数据产生的前后关系，检查输入的治疗数据的一致性和合理性。如果数据由外部产生并通过网络连接输入到医疗器械中，则可能无法进行同样的检查。在这种情况下，制造商可考虑将这种软件检查作为一种网络应用，使之对网络用户可用，和/或将输入的数据限制在可信任的数据源，并为在临床环境中负责网络连接的人员编写一份全面的手册。

IEC 80001-1[6]包含了临床环境下医疗器械与IT网络的集成，其具体描述了制造商和将医疗器械接入IT网络的人员的责任。

5.3 与安全有关的特性的识别

GB/T 42062—2022原文

5.3 与安全有关的特性的识别

对所考虑的特定的医疗器械，制造商应识别可能影响医疗器械安全的定性和定量的特性并形成文件。适当时，制造商应规定这些特性的界限。该文件应保留在风险管理文档中。

注1：能用作识别影响安全的医疗器械特性的指南的问题清单见YY/T 1437[10]。

注2：与医疗器械临床性能的丧失或降低有关的可能导致不可接受风险的特性，有时被称为基本性能（见GB 9706.1[1]）。

用查看风险管理文档的方法检查符合性。

没有用于软件的补充指南。

5.4 危险和危险情况的识别

GB/T 42062—2022原文

5.4 危险和危险情况的识别

制造商应基于预期用途、可合理预见的误使用以及正常状态和故障状态下的与安全有关的特性，识别已知的和可预见的与医疗器械相关的危险并形成文件。

对于每个已识别的危险，制造商应确定能够造成危险情况的可合理预见的事件序列或组合，并对导致的危险情况进行识别和形成文件。

注1：在生命周期的所有阶段都可能触发事件序列，例如，在运输、贮存、安装、维护、常规检查、最终停用和处置期间。

注2：附录C给出了“危险”、“危险情况”和“伤害”之间关系的解释和示例。

注3：风险分析包括检查与可能导致不同危险情况的单一危险有关的不同的事件序列或组合。每个危险情况可能导致不同类型的伤害。

注4：在识别以前未识别出的危险情况时，风险分析能采用覆盖特定情况的系统性技术。YY/T 1437[]提供了一些可用技术的指南。

这些文件应保留在风险管理文档中。

用查看风险管理文档的方法检查符合性。用查看风险管理文档的方法检查符合性。

危险识别的目的是分析所有可预见的危险，并设计和实施有效的风险控制措施。

与热能、电能或者悬挂物不同，软件本身不是危险（一种潜在的伤害源）；与软件接触不会造成损伤。然而，软件可能导致人体暴露在危险下，换句话说，它可能促成危险情况。软件失效（任何形式）常常促使危险转化为危险情况。

因此虽然软件很少引入新的危险，但其经常改变危险情况。更重要的是对制造商而言，它可将避免危险情况的责任由用户转移到制造商。

例如，解剖刀存在显而易见的切伤危险。然而，对这种危险制造商按惯例不承担人体工程学设计以外的责任，因为这种危险被假定完全在外科医生的控制范围内。如果解剖刀是远程外科手术系统的一部分，同样的危险仍然存在，但现在避免切伤危险的责任要由提供解剖刀控制软件的制造商分担。

这意味着一些在没有软件时仅依靠器械的专业使用进行风险控制的危险，现在转为由制造商通过软件风险管理来控制。

一个重要的案例是由数据误处理带来的误诊的危险。这的确是一种危险，但当这些数据由临床医生处理时，制造商就没有责任。现在许多医疗器械利用软件来生成、存储、处理或使用数据，这使得该危险部分成为制造商的责任。

软件促成危险情况可有几种方式，包括以下几种（也见附录B）：

——软件可能正确地实现了某个不安全的系统需求，导致具有危险性质的行为直到实际伤害发生才被察觉；

——软件规范可能错误地实现了系统需求，导致非预期的行为，虽然这种行为符合软件规范；

——软件的设计和实现可能出错，导致软件的行为与规范相违背。明显的错误可能源于对软件规范的误解以及将规范转换为代码时的差错。而较不明显的错误可能源于软件项间和软件与其基础设施（包括硬件和操作系统）间的非预期交互。

对包含软件的医疗器械，进行仔细和全面的危险识别可带来（在风险管理过程的后期）以下重要的结果：

——防止软件导致伤害的硬件风险控制措施；

——将潜在有害的软件功能从设计规范中删除；

——使用软件来防止伤害的风险控制措施（见 YY/T 0664—2020，5.2.3）；

——识别软件中必须以低错误率实现的部分和软件规范中必须进行特殊测试的部分（见 YY/T 0664—2020，4.3）；

——识别较高安全级别的软件项，其必须与其他软件项（较低的软件安全级别）隔离以防止非预期负面影响产生的伤害（见 YY/T 0664—2020，4.3 和 5.3.5）。对此进一步的讨论见 7.1.2.2.4。

为充分识别危险,医疗器械的临床使用必须得到充分理解。另外,软件的复杂性带来了特别的挑战，包括可能的复杂用户界面。因此，软件危险的识别不能孤立地进行，其宜由多学科的团队在系统层面开展，团队包括临床专家（如临床支持和技术服务专家）、软件工程师、系统设计师以及可用性/人因工程专家（见4.3）。

危险识别宜考虑能由医疗器械的性质导致的伤害（如切伤、辐射或者电死患者）及与软件使用有关的额外的危险。后者可能包括：

——向临床医生或患者提供错误信息；

——对患者的错误识别（在医疗器械存储患者详情和处方的情况下）；

——由软件反常引起的治疗延误或治疗无法进行。

注：对许多医疗器械来说，治疗延误或治疗无法进行不被认为是对患者的伤害。

识别的危险宜包括与按其规范操作的软件有关的危险，和与软件反常有关的危险（见7.1）。

通常软件使得医疗器械的用户界面更加复杂。特别是包括软件的医疗器械经常处理信息，在根据患者的受益判别合理性的同时，宜考虑与错误信息及错误使用信息有关的额外的危险，例如：

——错误的数据输入；

——引起用户误读的显示；

——用户对报警的误解和忽视；

——由于数据或报警过多使用户应接不暇（见 IEC 62366-1[5]）。

5.5 风险估计

5.5.1 总则

GB/T 42062—2022原文

5.5 风险估计

对每个已识别的危险情况，制造商应利用可获得的信息或数据估计相关的风险。对于伤害发生概率不能估计的危险情况，应列出可能的后果，以用于风险评价和风险控制。这些活动的结果应记录在风险管理文档中。

用于对伤害发生概率和伤害严重度进行定性或定量分类的系统应记录在风险管理文档中。

注1：风险估计包括伤害发生概率和伤害严重度的分析。依据应用领域，可能只有风险估计过程的特定要素需要详细考虑。例如，当伤害非常小时，初始的危险和后果分析可能是足够的；或者当可获得的信息或数据不充分时，对发生概率的保守估计可能给出一些风险指示，见YY/T 1437[10]。

注2：风险估计可能是定性的或定量的。在YY/T 1437[10]中描述了风险估计的方法（包括由系统性故障产生的风险），并提供了体外诊断医疗器械风险估计的有用信息。

注3：从以下来源能够获得用于估计风险的信息或数据：

——已发布的标准；

——科学或技术研究；

——已在使用中的类似医疗器械的现场资料,包括可公开获得的事故报告；

——典型用户进行的可用性测试；

——临床证据；

——有关研究或模拟的结果；

——专家意见；或

——对体外诊断医疗器械的外部质量评估方案。

用查看风险管理文档的方法检查符合性。

要估计与软件有关的风险，首先必须识别包括软件在内的危险情况。软件既可能是导致危险情况的事件序列的初始原因，也可能位于事件序列的其他位置，就像预期检出硬件故障的软件的情况。软件可能包含未知来源软件（SOUP）组件或重用的先前开发的组件。

风险估计基于每个已识别的危险情况所导致的伤害的概率和严重度。因为很难估计软件反常引发伤害的概率（见5.5.3）,所以在估计导致伤害的事件序列中包含软件反常的危险情况的风险时，须谨慎使用软件反常发生的概率。

5.5.2 识别的方法

有多种方法可用来识别软件在危险情况中的潜在作用。这些技术采用不同的途径，在软件开发的不同阶段发挥作用。其中的任何一种都不是唯一恰当的方法。关于风险分析的一些可获得的技术信息见YY/T 1437—2023的附录B。

故障树分析（FTA）是一种传统的自上而下的方法（见IEC 61025[4]），其通常从医疗器械整体开始分析。FTA主要用于分析伤害的起因。其假定伤害发生，利用布尔逻辑来识别伤害发生必需的事件或条件。以逐步细化的方式分析事件或条件，直到能防止伤害的一个或多个风险控制措施得到识别。FTA可以用于识别导致危险情况的事件序列中包含的软件项。

失效模式和效应分析（FMEA）是一种自下至上的方法（见GB/T 7826[1]），其从组件或子系统（对软件来说就是YY/T 0664中的软件项）开始，并提出问题：如果该要素失效，那后果是什么？

考虑到预见每个软件项中存在哪些软件缺陷的难度，FMEA的起始点会是列出每个软件项的安全有关需求，并考虑这个问题：如果该需求无法满足，那后果是什么？

这使得失效会引起伤害的软件项，以及需要防止的失效类型得到识别。

当识别能导致危险情况的事件序列或事件组合时，关注直接与医疗器械基本性能有关的软件（例如计算血液葡萄糖水平的算法）以及危险有关的具体原因是最容易的。考虑能导致不明显的失效模式并因此导致一个或多个医疗器械危险的软件原因也很重要。软件原因的示例见附录B。

注：具体原因是软件中的缺陷，而软件的功能明显与器械临床功能有关，并导致器械的危险。例如，计算试验结果的算法缺陷。

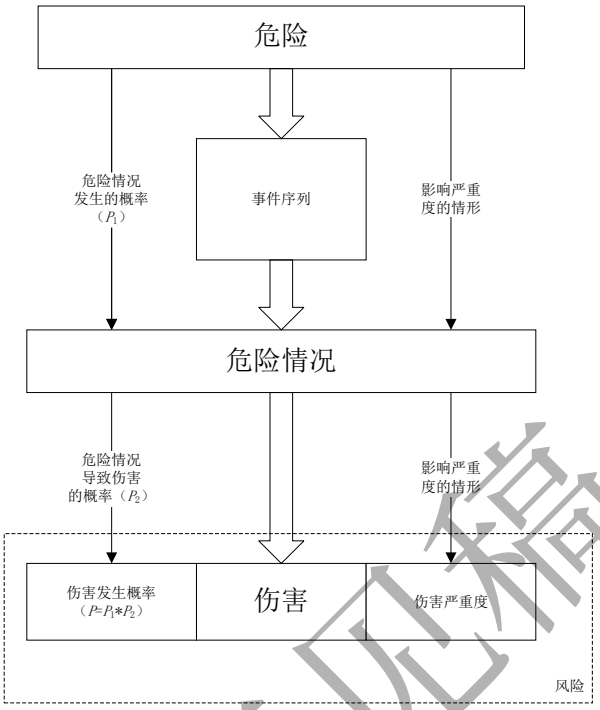
精确地预见在软件项中发生什么失效是困难的，但识别缺陷的类别是可能的，每一类都有众所周知的风险控制措施。例如，数据崩溃类故障可以通过求和校验程序来检出和防止。软件原因示例及建议的处理方式见附录B。制造商宜保持自己的与其产品相关的软件缺陷类别清单。

5.5.3 概率

特定版本软件中的反常在该软件的所有拷贝中都会出现。然而因为软件的每个单独拷贝输入的随机性，软件反常导致软件失效的概率很难估计。

对于估计软件失效发生概率的方法不存在共识。如果软件出现在导致危险情况的事件序列中，在估计危险情况的风险时不宜考虑软件失效发生的概率。此时考虑最不利情况下的概率是适当的，并宜将软件失效发生的概率设为1。当可以做到估计序列中其余事件（如果其不是软件）的概率时，该概率可用作危险情况发生的概率（图1中的P1）。如果无法做到，危险情况发生的概率宜设为1。

估计危险情况导致伤害的概率（图1中的P2）通常需要临床知识，以区分临床实践有可能防止伤害发生的危险情况和更可能导致伤害的危险情况。



注：P1-危险情况发生概率
P2-危险情况导致伤害的概率

图1　危险、事件序列、危险情况和伤害之间的关系图示示例——取自 GB/T 42062—2022 附录 C

在许多情况下，可能无法估计伤害发生的概率，宜仅依据伤害的严重度估计风险。在这些情况下的风险估计宜关注危险情况引发的伤害的严重度。

虽然可能无法估计软件失效发生的概率，但很明显许多风险控制措施降低了这种失效导致危险情况的概率。例如，软件反常导致的内存崩溃。内存求和校验会检出失效并降低危险情况的概率。该求和校验不能保证所有可能的崩溃都被检出，然而，这种手段将检出绝大部分此类崩溃并因此把风险降低到可接受水平。虽然在实施求和校验之前或之后都无法估计危险情况发生的概率，但可以断言，求和校验执行后，危险情况发生的概率比实施求和校验之前要低。制造商有责任证实风险控制措施对于剩余风险满足风险管理计划规定的可接受性准则是有效的。

总之，软件的风险估计宜主要关注严重度以及如果失效发生引发伤害的相对概率，而不是试图估计每个可能的软件失效的概率。

注：这有助于区分相同严重度的危险，以便更加关注那些具有更高实际伤害概率的危险。

尽管如此，如果有以往类似软件的设计开发经验、历史数据或上市前软件测试数据，或类似软件上市后的反馈信息，也可用来估计当前软件可能发生的伤害的概率，并和伤害的严重度一起考虑。

5.5.4　严重度

对软件导致风险的严重度的估计，影响所采用的软件开发过程。根据YY/T 0664，过程的严格程度取决于软件可能导致的伤害的严重度。

为了按照GB/T 42062进行风险评价，如何定义严重度等级取决于制造商，但与YY/T 0664 中软件安全分级相联系对定义严重度等级是有帮助的。否则可能必需两次定义严重度，一次为了整个风险管理过程所需的风险评价，另一次为依据YY/T 0664确定软件安全级别。

6　风险评价

GB/T 42062—2022 原文

6 风险评价

对于每个已识别的危险情况，制造商应使用风险管理计划中定义的风险可接受性准则评价估计的风险，并确定此风险是否可接受。

如果风险可以接受，则7.1～7.5中给出的要求不适用于此危险情况（即前进到7.6），此估计的风险应视为剩余风险。

如果风险不可接受，制造商应执行7.1～7.6中所述的风险控制活动。

风险评价的结果应记录在风险管理文档中。

用查看风险管理文档的方法检查符合性。

如5.5.3中所述，很难估计软件失效的概率。当这导致伤害的概率不能估计时，宜仅基于伤害的严重度评价风险。

7 风险控制

7.1 风险控制方案分析

GB/T 42062—2022原文

7 风险控制

7.1 风险控制方案分析

制造商应确定适于将风险降低到可接受水平的风险控制措施。

制造商应按下列优先顺序，使用一种或多种风险控制方案：

a) 固有安全的设计和制造；

b) 医疗器械本身或制造过程中的防护措施；

c) 安全信息和适当时的用户培训。

注1：选择风险控制方案的优先顺序的原理说明见 A.2.7.1。

注2：风险控制措施可能降低伤害严重度或伤害发生概率，或两者都降低。

注3：提供安全信息的指南见YY/T 1437[10]。

有关标准宜用作风险控制方案分析的一部分。

注4：许多标准阐述了医疗器械的固有安全、防护措施和安全信息。此外，某些医疗器械标准融入了风险管理过程的要素（例如电磁兼容性、可用性、生物学评价）。关于标准在风险管理中的作用的信息见YY/T 1437[10]。

所选择的风险控制措施应记录在风险管理文档中。

在风险控制方案分析中，如果制造商确定降低风险不可行，制造商应进行剩余风险的受益-风险分析（前进到7.4）。

用查看风险管理文档的方法检查符合性。

7.1.1 复杂系统风险控制方案的选择

7.1.1.1 总则

在复杂系统中，可能有许多能导致危险情况的事件序列。不可能或没有必要对这种序列中的每个事件都应用风险控制措施。对所选事件应用风险控制措施就足以把伤害的总概率降到可接受水平。

在以下三条中，给出了如何在软件中实施三种风险控制措施的概述。另外还讨论了到底哪些事件需要风险控制措施（见7.1.1.5）。

7.1.1.2 通过设计和制造获得固有安全

通过设计获得固有安全常常通过消除医疗器械的不安全的功能特性来实现，或者通过改变设计用更安全的方法（即以避免或最小化危险情况的方法）来实现功能特性。这通常有简化设计的效果，使设计更易实现、用户更易操作。

对于在软件中实现的功能特性更是如此。不加辨别地在软件系统中包含所有可能的客户期望是一种诱惑。这可能导致软件组件间能交互的方式大量增加，引入意想不到的危险情况。通过在医疗器械及其软件的开发过程的早期实施风险管理，可以避免这些问题同时仍然使大部分客户满意。

在大多数情况下，对于软件通过设计的方法取得固有安全涉及：

——除去不必要的功能特性；

——改变软件体系结构以避免导致危险情况的事件序列；

——简化用户界面以降低使用中人为错误的概率；

——规定软件设计规则以避免软件反常。

后者的示例包括：

——仅使用静态内存分配以避免与动态内存分配有关的软件反常；

——使用限定的编程语言版本以避免可能导致编程错误的结构。

关于通过制造获得固有安全，宜特别关注软件拷贝过程所用软件、工具及设备的确认和维护，确保所拷贝的医疗器械软件的完整性，并防止对软件安全产生不良影响。

7.1.1.3 防护措施

使用软件的医疗器械的防护措施能通过硬件或软件实施。防护措施的设计宜证明防护措施独立于其所应用部分的功能。如果在硬件上应用软件防护措施，相对容易实施，反之亦然。

在选择以软件方式实施防护措施并应用于软件时，避免同一个原因导致多重失效的可能性是重要的。如果一个防护措施检出和/或防止一个危险情况，制造商宜证明该防护措施和提供基本性能的软件功能特性具有充分的隔离。

例如，为患者提供治疗的软件可在一个处理器上运行，而实施软件防护措施的软件在另一个独立的处理器上运行。

7.1.1.4 安全信息和用户培训

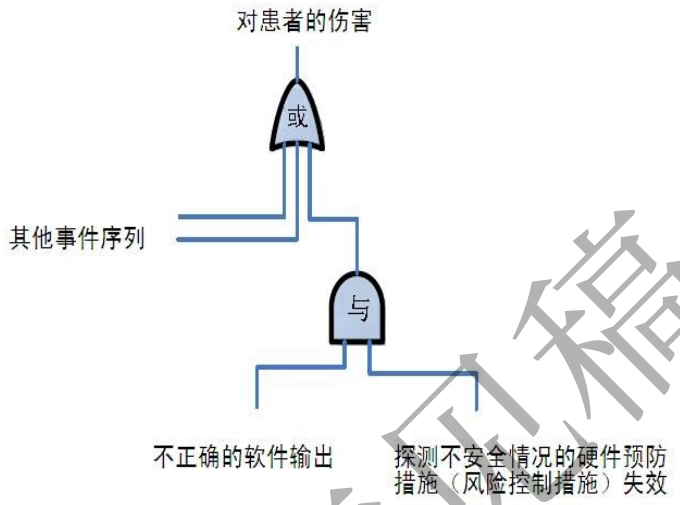
医疗器械中软件的使用很可能导致比用户所看到的更加复杂的行为。这很可能导致对安全信息的更多依赖，从简单的屏幕警告到复杂的用户手册和明确定义的培训课程。通过注重良好的用户界面设计可以降低这些书面材料的量和复杂程度。

医疗器械软件的使用经常涉及培训，如果将用户培训作为风险控制手段，则培训资料属于安全信息；此外，制造商还宜确保在软件预期使用寿命内具备相应的培训能力。具体见（见IEC 62366-1[5]）。

7.1.1.5 哪些事件需要风险控制措施

许多事件序列都可能导致危险情况。不可能或不必要对这些序列中的每个事件应用风险控制措施。有选择地对一些事件应用风险控制措施，就足以把总的伤害概率降到可接受水平。

在决定哪些事件宜被检出、阻止或使其发生的可能性降低时，绘制出能导致危险情况的事件序列显然是有帮助的。虽然故障树分析（见5.5.2）不显示事件序列，但可以通过它来识别这些序列。风险控制措施的正确操作就像是“与门”的一个“非”输入，不管“与门”其他的输入是什么，伤害都会被阻止。图2表示FTA图的一部分，图中一个不正确的软件输出（其本身是一个事件序列的输出）由旨在检测不安全情况的风险控制措施所阻止，防止其对患者造成伤害，并防止输出有害影响（例如通过终止一个动作）。只有当风险控制措施失效时，不正确的软件输出才能导致对患者的伤害。需要注意的是，导致不正确软件输出的事件序列并不需要被详细探究以确保其不能导致对患者的伤害。



风险控制措施阻止不正确软件输出引发伤害的FTA图示

明显可以应用风险控制措施的点包括：

——软件系统作为一个整体的输入；

——软件系统作为一个整体的输出；

——软件模块之间的内部接口。

限制软件输入范围的风险控制措施能防止不安全的输出。它也能降低输入导致伤害（由于软件反常）的可能性，因为其降低了软件以可能未经测试的非预期方式运行的概率（见5.5.3），但这种降低不是太明显。

限制软件输入范围可以通过软件或硬件风险控制措施来实现，例如：

——能检查输入并且拒绝不安全或不一致值的软件风险控制措施；

——硬件风险控制措施可能包括一个上锁的房间以防止未经授权的人输入数据。

设置于医疗器械或其软件输出的风险控制措施可以检查软件输出值是否在安全范围内和是否具有内部一致性以防止伤害发生，这可以通过以下方法实现，例如：

——能检查输出值并防止其偏离安全范围的软件风险控制措施；

——限制施加给患者的能量的硬件风险控制措施；

——由警告标签和硬线连接的“停止”开关组成的风险控制措施。这种风险控制措施假定有能力的操作者能够检出危险情况。

除了应用于医疗器械或其软件的输入和输出的风险控制措施外，也可以对软件组件的输入和输出应用风险控制措施。这样可以检查软件更小组件的输入和输出并且防止伤害。

可能无法规定医疗器械安全运行的单一参数范围，然而可以规定一个“安全操作范围”，换句话说是一个参数组合，其组成医疗器械安全运行的边界。软件可以用来评估医疗器械的操作是否在安全操作范围内。例如，软件可以通过对应用部分测得的温度和暴露时间的组合来检出灼伤患者的可能性。

在有些案例中，临床医生知道软件输入和输出的安全范围，但在医疗器械的设计中却无法预期。在这些案例中，可以通过风险控制措施确保医疗器械严格按照临床医生规定来运行。可以用硬件或软件的风险控制措施来检出软件输出与输入间的不一致。

例如，临床医生可开出使用医疗器械治疗的处方，其对不同的患者有很大不同。仅通过分析输入值和输出值不能检出危险情况。尽管如此，仍可应用软件风险控制措施确保医疗器械输出与输入（预期的处方）的准确匹配。

7.1.2 风险控制方法

7.1.2.1 概述

为了对软件有效地实施适当的风险控制措施，宜深入理解产品开发和软件生存周期。一些类型的风险控制措施在设计早期很容易实施，但在开发过程晚期不可能实施或成本太高。如果没有在产品开发过程的早期从风险管理角度仔细考虑软件，为了医疗器械的安全，可能会作出使用硬件措施的决定，无意中造成对正确软件操作的过度依赖。

隔离软件项并为软件项赋予软件安全级别可能是有用的，这样可以将高度关键的软件项（例如，如有缺陷可以导致死亡的软件项）与不会影响安全的软件项区分开来。见YY/T 0664—2020 标准的4.3软件安全分级。

赋予软件安全级别，可以作为对更关键的软件项进行更严格控制、重点验证和配置管理活动的基础。如果这样做，宜仔细考虑其副作用，对关键性较低的软件项宜与受其影响的任何更关键软件项有相同的评级。宜注意YY/T 0664允许在同一个活动或任务中使用不同的方法（见YY/T 0664—2020的5.1.4关于规定方法的要求）。制造商可以制定方案来区分软件安全分类为C级的软件项。例如，制造商可对复杂程度高的软件项使用更正式的验证方法（例如，相对于代码评审而采用代码检查）。

需要注意的是，软件项可能在初期被定为与安全有关，然后通过某些风险控制措施或设计选择，将其作为关键性较低的软件项对待。合理地执行风险管理可以通过隔离和固有安全设计，使安全有关软件的子集尽可能减少到最小。

确保软件安全需要贯穿产品开发生存周期的各种活动。可靠性技术（例如正式的故障分析方法）并不能构成完整的风险管理方法。可靠性和安全，尽管非常相关，但它们并非同一个属性，认识到这一点也很重要。关注可靠性的软件生存周期过程并不一定能达到足够的安全。

7.1.2.2、7.1.2.3、7.1.2.4、7.1.2.5和7.1.2.6中更详细地描述了一些具体的风险控制措施，并就如何处置风险的具体原因给出了指南。

7.1.2.2 风险控制措施和软件体系结构设计

7.1.2.2.1 概述

软件体系结构宜描述通过固有安全设计来控制风险的软件功能特性，以及通过防护措施来降低风险的软件机制。

7.1.2.2.2 通过体系结构特性提供的固有安全设计

与软件控制功能相关的危险可以避免，例如通过硬件实现功能（类似地，与硬件功能相关的危险（磨损、疲劳）可以通过使用软件来避免）。

有时某个危险可以通过高层级的设计决定完全避免。例如，从硬件的方面看，使用电池代替交流电源可以完全排除触电死亡的风险。同样的，导致危险的整个编程错误可以通过高层级的设计决定来消除。例如，内存泄漏可以通过仅使用静态数据结构来避免。

在使用软件的系统中有个特殊的问题，认为软件能不受限度地共享硬件基础设施。这种观念是错误的。

系统设计中有一个通用规则，系统宜包括足够的资源，以在需要时执行所有必要的任务。这个规则对软件和硬件都适用。如果一个软件项和安全有关，风险评估宜阐述以下问题：

——该安全有关软件项需要时能否访问它的处理器？

——在不安全状况发展成事故前，安全有关软件项能否确保足够的处理器占用时间以完成其任务？

——能否证明没有其他软件项能破坏或干扰安全有关软件项？

如果安全有关软件不得不与非安全有关的软件共享处理器，那么以上的问题尤为重要，因为安全功能会和非安全功能竞争资源（关于隔离见7.1.2.2.4）。

开发方法的选择宜确保所有上述问题对设计者可见。例如，将安全有关软件项设计为一个过程，并在操作系统有空闲时才运行，这样的设计是不够的。开发方法宜支持计划安排、优先级和时序的审慎设计。

7.1.2.2.3 容错体系结构

医疗器械需要有许多功能，以确保患者或用户的安全。这些功能可包括不能被中断或延迟的临床功能，以及实施防护性风险控制措施的功能。

容错设计是一种非常通用的方法，用以提高医疗器械可靠性（参考软件工程实践者，包括Pullum[8]和 Banatre[9]）。容错设计的目的是确保安全有关的功能在组件故障（包括软件反常）发生时仍然继续运行。

容错设计常常利用冗余。这可是一个至关重要组件的简单复制，以便在一个组件失效时系统能继续运行，或者由额外组件组成，以便检出失效并将运行切换到替代模式中，这也许限制了部分功能。

使用容错设计能在软件失效时使至关重要的功能继续运行。在这种情况下，使用同一软件多重拷贝的简单冗余可能是不充分的，因为同样的缺陷在软件的每个拷贝中都存在。

在这种情况下，需要使用多样性。例如，使用额外的软件检出软件错误并执行恢复程序。额外的软件宜避免与被监控的软件共享任何特性，以排除一个软件缺陷导致两个软件都失效的可能性。

在更关键的情况下，可由两个或更多软件项完成同样的功能，但他们从一个通用的规范开始，被独立设计和实现。这就是“多样性编程”。然而需要注意，不同开发工程师有犯同样错误的趋势，这会使多样性失效。还要注意通用规范可能包含不正确的需求。最终，必须使用一些方法，比如表决，来确保故障软件不会产生任何影响。至少需要三个不同的软件项来实现表决机制。

当使用冗余来实现容错设计时，不管是否使用多样性，告知用户发生了失效很重要。否则，具有容错设计的医疗器械可能看起来运行正常，但实际上是在降低安全运行。

7.1.2.2.4 通过隔离降低软件原因带来的风险

软件缺陷可能使在相硬件上运行的不相关的软件发生错误。制造商宜选择隔离安全有关软件项和非安全有关软件项的方法，使非安全有关软件项不能干扰安全有关软件项的运行（见YY/T 0664—2020的5.3.5），并宜证明隔离是有效的。这包括证明软件项使用适当的资源（物理资源或时间）以避免软件项间非预期的争夺。

软件项间的有效隔离必须处理以下几种软件项发生非预期交互的可能方式。

——当软件争夺共享硬件（例如处理器、存储设备和其他输入/输出设备）的占有时间时，软件项可能以非预期的方式交互。这能妨碍软件项在预期的时间运行。提供充分的硬件资源是一种体系结构特性（见 7.1.2.2.2），宜有足够的规范和策划来确保在需要时有充分的时间来运行所有的软件项。

——多个软件项可能共同存在于同一个内存中；这会导致一个软件项非预期地改变属于另一个软件项的数据。在极端情况下，一个软件项可能偶然地改变了另一个软件项的编码。许多处理器和操作系统提供硬件辅助隔离内存使用的方法。当有这些方法时，就宜使用。许多这样的方法会防止非预期的交互，即使其中一个软件项存在缺陷。

——当软件项共享变量（包括全局变量、环境变量和操作系统参数）时，也可能存在非预期的交互；这会在其中的一个软件项存在缺陷时，导致软件项间非预期的通信。软件项间变量共享宜做到最小化。必要的话，向所有工程师发布规则，以确保只有少数规定的软件项才能改变共享变量，而所有其他的软件项仅能读取共享变量，而不能修改它们。

最强的隔离方式是在不同的处理器上运行不宜交互的软件项。然而，如上推荐的谨慎的体系结构设计可以在单一处理器上提供适当程度的隔离。

在实验室环境中测试系统，对于给定的测试用例，可能物理和时间资源都是充分的，然而现场的应用负荷或执行环境（其他过程在同一地点运行）使软件以某种方式失效从而导致伤害。

另一方面，如果实验室中的测试用例确实表现出低性能并采用无效措施草率地提升软件速度，这些措施可能破坏设计并通过不可预见的副作用增加其他风险。

有效的隔离宜证明在正常操作下：

a) 能防止数据流崩溃：非安全有关软件项不能修改安全有关数据；

b) 能防止控制流崩溃：
- 安全有关功能总能在正确的时间执行，不被非安全有关软件项的行为影响；
- 非安全有关软件项不能修改安全有关软件项；

c) 能防止执行环境崩溃：安全有关软件项和非安全有关软件项（例如，处理器寄存器、设备寄存器、内存访问特权）使用的软件系统的部分不会发生崩溃。

导致违背上述原则的事件（例如，硬件失效）宜能被检出并且使系统采取必要的行动来确保持续安全。

7.1.2.3 防护措施细节

在很多情况下，通过固有安全设计来避免所有危险或者对所有潜在失效执行容错是不切实际的。在这些情况下，防护措施是次优的管理潜在危险的方法。这些措施通常通过检出潜在危险情况来运行，或者自动干预以减轻后果，或者产生报警使用户可以干预。

例如， X射线治疗系统可有一个使用软件逻辑或硬件的互锁系统，当通往该设施的任何门打开时，关闭X射线发生器。互锁功能对治疗没有作用。它的唯一目的就是减轻非预期的射线暴露的伤害。

在某些情况下（即医疗器械功能丧失不会产生危险），安全可能以未完成任务为代价获得。例如，实验室血液分析仪未能提供结果，在某些情况下这可能并不危险，但提供错误结果可能是危险的。在这个示例里，当保护性检验程序显示非预期的故障时关闭分析仪，与继续工作相比降低了风险。在失效-安全（fail-safe）体系结构中，系统或组件故障，或其他危险条件，可导致功能丧失，但在某种程度上保护了操作者和患者的安全。在失效-可操作（fail-operational）系统中，系统可继续安全操作，但性能较低（例如降低容量或减慢响应时间）。

7.1.2.4 即时防止和公布危险情况

一类重要的风险控制措施是提高防止危险情况的可能性。

除了防止伤害，还宜考虑向用户公布检出的情况。否则风险控制措施随后的失效可能使伤害发生。

宜考虑软件风险控制措施运行的频率。软件风险控制措施宜运行得足够频繁以在导致伤害之前检出危险情况。

7.1.2.5 软件反常的风险控制措施

软件存在一个特别的难点：导致危险情况的序列中的一些事件可由未发现的软件反常导致，很难预期这些反常会在哪里发生或其后果是什么。

风险控制措施能降低软件反常引发伤害的概率。不管先前事件的性质怎样，通常在软件体系结构中总存在着一些位置，可以施加风险控制措施以降低伤害的概率。如果谨慎地做到这些，就没必要通过准确预测软件反常的性质来防止他们引发伤害。

在这种方法不可行的情况下，例如在软件中实施防护措施，则宜使用确保软件完整性的方法（见7.1.2.6）。

7.1.2.6 过程作为风险控制措施

如果软件反常可能促成导致危险情况的事件序列，可能无法设计风险控制措施来防止伤害的发生。这种情况下，最好的解决办法是通过设计的固有安全，使软件反常不能导致危险情况。

当固有安全设计不可行时，可以利用有效的软件开发过程来降低软件反常发生的概率。一致的共识是，当与其他类型的风险控制措施一同考虑，并且详细定义时，过程风险控制措施是有益的方法。

如果能对软件可靠地完成预期功能建立高度信心，并且对软件无故障的信心很高，则软件可被视为高完善性组件。为达到这种高度信心，制造商必须证明软件开发过程能预期产生高度可靠、无故障的软件。应用这样的过程则可以声称降低了软件反常发生的概率。

一般认为，提高软件开发过程的严格程度可以降低软件反常的数量。需要注意的是虽然医疗器械软件测试可以降低软件反常的数量，但不能假定当软件通过所有计划的测试时，就没有软件反常了。这是因为在临床使用中，对软件的输入包含测试计划外的序列。由于医疗器械软件太复杂以至于不能无遗漏地测试，所以严格的测试只能看作是降低危险情况概率的一种方法。然而测试本身，不足以建立软件可被视为高完善性组件的信心。

在软件开发过程中包含YY/T 0664中规定的活动和任务，可以作为定义严格软件开发过程的起始点。开发严格的软件开发过程时需要考虑的其他项目可包括：

a) 人员能力——技能、资质、经验和培训（谁开发软件？）；
b) 方法——规范、设计、编码和测试方法的适用性（开发过程是什么？）；
c) 评审及检查的严谨性、正式性和范围（进行了多少静态分析？）；
d) 工具——诸如编译器、需求的可追溯性工具和配置管理工具等的质量（软件开发期间使用什么工具？）。

开发高完善性软件的可预测性取决于具有一个始终得到遵循的可重复的过程。

当通过实施严格的开发过程来降低由软件反常导致危险情况的风险时，宜通过收集和分析表明软件反常导致的失效频率的数据来证明风险控制措施的有效性。要声称某个过程产生了高完善性软件，宜有证据来支持没有或很少有软件失效。

7.1.3 对未知来源软件（SOUP）的考虑

使用SOUP的决定通常是在系统设计阶段做出的。医疗器械潜在风险越高，对SOUP的潜在故障模式分析就越严格并确定风险控制措施。通常不能通过更改SOUP来添加监控或隔离SOUP所需的新风险控制措施,来防止其失效引发危险或危险情况。也没有足够的内部设计信息可以用来识别所有由SOUP引起的潜在危险。所以系统和软件体系结构设计时宜提供必要的风险控制措施监控或隔离SOUP，以防止SOUP失效时产生危险。

当医疗器械包含SOUP组件时，必须注意该软件不能危及医疗器械安全。这种情况可能需要引入软件“封装”或中间件体系结构。中间件可以：

a) 阻止不希望使用的 SOUP 功能特性的使用；
b) 执行逻辑检查以确保正确的信息在 SOUP 和医疗器械软件之间传送；
c) 提供医疗器械需要的附加信息。

还有一个与SOUP有关的关键问题就是商用操作系统和通讯系统的使用。宜基于全面的风险评估，在不影响安全的同时，建立一种允许对软件平台进行更改（例如稳定性，信息安全）的体系结构。这种风险评估宜包括必要的变更频次分析以确保医疗器械安全的完整性；例如安装网络信息安全补丁。

7.2 风险控制措施的实施

GB/T 42062—2022 原文

7.2 风险控制措施的实施

制造商应实施在7.1中选择的风险控制措施。

应对每个风险控制措施的实施予以验证，此验证应记录在风险管理文档中。

注1：风险控制措施实施的验证可能作为质量管理体系中设计和开发验证或过程鉴定的一部分。

应对风险控制措施的有效性予以验证。验证结果应记录在风险管理文档中。

注2：有效性验证可能作为质量管理体系中设计和开发确认的一部分，并且可能包括用户测试，见A.2.7.2。

注3：如果风险降低的有效性与设计和开发验证或过程鉴定结果之间的关系已知，则有效性验证也可能作为设计和开发验证或过程鉴定的一部分。

示例1：某产品性能特性的设计验证，例如一个药物注射器的剂量准确度，可能作为确保安全药物剂量的风险控制措施有效性的验证。

示例2：过程鉴定可能作为对生产输出变化导致的风险有关的风险控制措施的有效性的验证。

注4：关于设计与开发验证和确认的更多信息，见GB/T 42061—2022[7]。更多指南见YY/T 1437[10]。

用查看风险管理文档的方法检查符合性。

一旦确定了风险控制措施，那么这些措施就需要得到实施，并验证其有效性。

验证风险控制措施已正确实施并且对风险控制有效，这对软件来说是非常基本的。分析和测试可能都是必要的。要考虑的关键方面包括：

a) 可追溯性，以确保所有安全有关软件项得到识别，且在软件所有相关版本和变体（例如，对于不同的平台、语言或医疗器械型号）中，所有安全有关功能得到规定、执行和测试；
b) 测试风险控制措施时更严格和更大的覆盖范围，包括大范围的反常和压力条件下的测试；
c) 当发生变更时，即使这些变更预期不影响安全，也要关注对风险控制措施和安全有关功能的回归测试。

即使所用的过程被认为严格到足以创建高度完善的软件，结果仍必须被验证。

如果进行了根本原因分析且跟踪和评价了安全有关反常的关键性评级（见YY/T 0664—2020的9.1），在验证和确认活动中收集的反常信息常常是有用的。可评价有安全后果的反常以确定在风险评估中是否被识别，以及已识别的风险控制措施一旦实施是否充分。软件反常的数据可用来证明软件开发过程的有效性，或用来识别软件开发过程哪些方面需要改进以便声称其降低了风险。

软件风险控制措施的充分性可能不如硬件风险控制措施的充分性那样明显。因此当处理软件时，宜考虑风险评估文档是否需要传统形式外的其他形式。一个可能有用的方式是编写“安全用例”（见附录E）。

7.3 剩余风险评价

GB/T 42062—2022原文

7.3 剩余风险评价

在实施风险控制措施后，制造商应使用风险管理计划中定义的风险可接受性准则对剩余风险进行评价。此评价结果应记录在风险管理文档中。

如果使用这些准则，判定某个剩余风险是不可接受的，应研究进一步的风险控制措施（返回到7.1）。

用查看风险管理文档的方法检查符合性。

软件产生的剩余风险需要包括在与医疗器械相关的系统级的剩余风险中。在难以估计软件反常的概率的情况下，剩余风险评价通常包括：确定是否所有导致不可接受风险的事件序列都有风险控制措施，以降低其发生的概率，或将伤害的严重度限制在风险管理计划中定义的可接受水平（见4.4.1）。

识别到的（在验证和确认活动期间）任何未纠正的软件反常都宜通过分析来确定其是否影响安全有关软件（见YY/T 0664—2020的5.8.3）。如果影响，就需要评价这些反常引起的风险，并用来评价这些反常能影响的任何危险情况的剩余风险。

7.4 受益-风险分析

GB/T 42062—2022 原文

7.4 受益-风险分析

如果使用风险管理计划中建立的准则，判定某个剩余风险是不可接受的，而进一步的风险控制又不可行，制造商可收集和评审数据和文献，以便确定预期用途的受益是否超过该剩余风险。

如果此项证据不支持受益超过剩余风险的结论，则制造商可考虑修改医疗器械或其预期用途（返回到5.2）。否则，该风险是不可接受的。

如果受益超过该剩余风险，则前进到7.5。

受益-风险分析的结果应记录在风险管理文档中。

注：实施受益-风险分析的指南见YY/T 1437[10]。

用查看风险管理文档的方法检查符合性。

没有用于软件的补充指南。

7.5 由风险控制措施产生的风险

GB/T 42062—2022 原文

7.5 由风险控制措施产生的风险

制造商应评审风险控制措施的影响，考虑是否：

——引入新的危险或危险情况；或
——风险控制措施的引入对以前已识别的危险情况所估计的风险产生影响。

对任何新的或增大的风险应按照5.5～7.4进行管理。

评审结果应记录在风险管理文档中。

用查看风险管理文档的方法检查符合性。

严格的软件配置管理过程，包括严格的变更控制（见YY/T 0664—2020的8.2）是必须的，以便所引入的软件风险控制措施对医疗器械其他部分的影响得以仔细地检查（见YY/T 0664—2020的7.4）。

GB/T 42062并未规定设计和开发过程。这样的一个结果是：当某个风险控制措施已经实施，GB/T 42062只是简单地要求对这个措施进行评审以确保其没有导致进一步的危险。这不宜理解为仅在实施完风险控制措施后才要求检查问题。

对软件风险控制措施，特别重要的是不能等到软件实现完成后才进行评审。一旦规定了软件风险控制措施，就宜将其置于配置管理之下并予以评审以发现不良的副作用，包括无意中产生的新的危险或危险情况。

风险控制措施的实施大大地增加了软件设计的复杂程度，这可增加额外的软件反常的可能性或导致新的危险情况。风险控制措施宜尽可能的简单可行，并宜始终接受新的风险评估。

这种评审至少宜在软件设计后以和软件系统测试后重复进行。

7.6 风险控制的完整性

GB/T 42062—2022原文

7.6 风险控制的完整性

制造商应评审风险控制活动，以确保所有已识别的危险情况产生的风险已经得到考虑，并且所有的风险控制活动已经完成。

评审的结果应在记录在风险管理文档中。

用查看风险管理文档的方法检查符合性。

当软件是危险情况的促成因素时，宜考虑将YY/T 0664—2020的7.3.2纳入风险控制的完整性活动中。

8 综合剩余风险评价

GB/T 42062—2022 原文

8 综合剩余风险评价

在所有风险控制措施经实施并验证后，制造商应使用风险管理计划中定义的方法和综合剩余风险可接受性准则[见4.4 e）]，考虑所有剩余风险的影响，与预期用途的受益相比较，评价医疗器械造成的综合剩余风险。

如果综合剩余风险判定为可接受，制造商应向用户告知重大的剩余风险并且在随附文件中包括必要的信息以公开这些剩余风险。

注1：关于公开重大剩余风险的原理说明见A.2.8。
注2：关于综合剩余风险评价和剩余风险公开的指南见YY/T 1437[10]。

与预期用途的受益相比较，如果综合剩余风险被判定为不可接受，制造商可考虑实施其他的风险控制措施（返回到7.1），或修改医疗器械或其预期用途（返回到5.2）。否则，综合剩余风险仍不可接受。

综合剩余风险评价的结果应记录在风险管理文档中。

用查看风险管理文档和随附文件的方法检查符合性。

综合剩余风险的评价要求实施所有风险控制措施。这包括在其使用环境的每个不同的系统配置下对软件的评价。

系统测试活动（关于所有软件功能和硬件风险控制）的结果宜与可接受性准则一起评价。所有剩余软件反常需要记录在风险管理文档中，并予以评价以确保其不会促成不可接受的风险（见YY/T 0664—2020的5.8.2和5.8.3）。必要时，通过独立的含临床/应用专家的多学科评审进行的评价是可接受的。在随附文件中包括相关信息也是有必要的。

9 风险管理评审

GB/T 42062—2022 原文

9 风险管理评审

在医疗器械商业销售发布前，制造商应对风险管理计划的执行情况进行评审。评审应至少确保：

——风险管理计划已被适当地实施；
——综合剩余风险是可接受的；
——已有适当方法收集和评审生产和生产后阶段的信息。

评审结果应作为风险管理报告予以记录和保持，并应包括在风险管理文档中。

在风险管理计划中，应将评审的责任分配给具有适当权限的人员[见4.4 b）]。

用查看风险管理文档的方法检查符合性。

宜考虑将YY/T 0664—2020的第6章和7.3.2作为风险管理过程评审的一部分。

10 生产和生产后活动

10.1 总则

GB/T 42062—2022原文

10 生产和生产后活动

10.1 总则

制造商应建立、形成文件并保持一个系统，以在生产和生产后阶段主动地收集和评审与医疗器械有关的信息。在建立此系统时，制造商应明确收集和处理信息的适当的方法。

注1：见GB/T 42061—2022[7]的 7.3.3、8.2.1、8.4 和 8.5。
注2：关于生产和生产后活动的指南见YY/T 1437[10]。

用查看适当文件的方法检查符合性。

软件风险管理在软件生存周期内持续进行，其中包括软件维护过程（见YY/T 0664—2020的第6章）和软件问题解决过程（见YY/T 0664—2020的第9章）。

YY/T 0664—2020 的第6章要求制造商建立软件维护计划，阐明在医疗器械软件发布以后接收、形成文件、评价、解决和跟踪反馈的程序。维护计划还宜阐明所使用的软件风险管理过程和问题解决过程，用来分析和解决医疗器械软件发布后发生的问题。

10.2 信息收集

GB/T 42062—2022 **原文**

10.2 信息收集

适用时，制造商应收集以下信息：

a） 生产期间和生产过程的监视期间所产生的信息；

b） 用户产生的信息；

c） 负责医疗器械安装、使用和维护的人员产生的信息；

d） 供应链产生的信息；

e） 可公开获得的信息；

f） 与普遍公认的最新技术水平有关的信息。

注：与普遍公认的最新技术水平有关的信息可能包括：新的或修订的标准，所考虑的医疗器械特定应用的已发布的确认数据，替代医疗器械和/或治疗方法的可获得性以及其他信息（见 YY/T 1437[10]）。

制造商还应明确主动收集和评审市场上可公开获得的类似医疗器械和其他类似产品信息的需要。

用查看适当文件的方法检查符合性。

除了对已发布软件本身涉及的生产和生产后信息进行收集外，SOUP也是软件维护计划和生产后风险管理活动的一个重要方面。一些SOUP根据其特性（如防毒软件）可能需要经常更新，制造商宜在软件维护计划中考虑这一点。

SOUP的失效或非预期结果以及SOUP的退市（停止技术支持）可能影响医疗器械综合剩余风险的可接受性。因此，有必要在软件系统的开发和维护中实施SOUP监视活动。该活动宜阐述SOUP更新、升级、Bug修复、补丁和退市。制造商宜主动监视公开可获得的反常清单和关于SOUP现场性能的信息。

10.3 信息评审

GB/T 42062—2022原文

10.3 信息评审

制造商应评审所收集的可能与安全有关的信息，特别关注：
——是否有先前未识别的危险或危险情况出现；
——是否由危险情况产生的已估计的风险不再是可接受的；
——与预期用途的受益相比较，是否综合剩余风险不再是可接受的；或
——是否普遍公认的最新技术水平已经发生变化。
评审的结果应记录在风险管理文档中。
用查看风险管理文档的方法检查符合性。

软件问题解决过程（见YY/T 0664—2020的第9章）的使用整合了软件问题调查中和问题安全相关性评价中的风险管理活动。在调查和评价问题时组成包括临床专家、软件工程师、系统设计人员和可用性/人因工程专家的多学科团队是非常重要的（见4.3节）。

对于监视活动收集到的任何公开可获得的反常清单和关于SOUP现场性能的信息，制造商宜予以评价，并主动确定是否有任何已知的反常导致能引发危险情况的事件序列（见YY/T 0664—2020 的6.1f）、7.1.2c）、7.1.3和7.4.2）。

制造商发布的SOUP软件补丁或更新可能包含额外的功能，这些功能对医疗器械的安全和有效性不是必要的。对于这些SOUP更新宜分析可以从医疗软件发布中去除的多余组件，从而避免可能导致危险情况的非预期的变化。

与任何软件项目更改一样，制造商宜了解哪些软件项受SOUP更新影响并执行回归测试（见YY/T 0664—2020的7.4、8.2和9.7）。

10.4 措施

GB/T 42062—2022原文

10.4 措施

如果所收集的信息确定与安全有关，则需要执行以下措施：

1） 关于特定的医疗器械，

——制造商应评审风险管理文档并确定是否有必要对风险进行重新评估和/或对新风险进行评估；

——如果某个剩余风险不再可接受，应对先前实施的风险控制措施的影响进行评价，并宜考虑将其作为修改医疗器械的输入；

——制造商宜考虑对市场上的医疗器械采取措施的需要；

——任何决定和措施应记录在风险管理文档中。

2） 关于风险管理过程：

——制造商应对先前实施的风险管理活动的影响进行评价；

——应将评价结果作为最高管理者评审风险管理过程适宜性的输入（见4.2）。

注：生产后监视的一些要求是某些国家法规的内容。在此情况下，可能需要附加的措施（例如后续的生产后评价）。

用查看风险管理文档和其他适当文件的方法检查符合性。

没有用于软件的补充指南。

附 录 A
（资料性）
定义的讨论

危险和伤害是GB/T 42062—2022中一对关键的术语。对这两个术语的理解，对于理解本文件很重要。伤害是对人健康的损伤或损害，或对财产或环境的损害。危险被定义为可能导致伤害的潜在根源。危险有许多原因。

根据GB/T 42062—2022的定义，只有当事件序列或其他情形（包括正常使用）导致危险情况（见5.5.3 图1）时，危险才能导致伤害。事件序列既包括单一事件也包括事件组合。YY/T 1437提供了危险和危险情况的指南。GB/T 42062—2022的附录C提供了关于危险、可预见事件序列和危险情况示例的指南。

危险或危险情况的起因是任何事件序列,这些事件的组合可合理预见会导致危险情况。给定的危险可能有一个、几个或许多可能的原因（事件序列）。

与热能、电能或悬挂物不同，软件本身不是危险（一种潜在的伤害源）；与软件接触不会造成伤害。然而，软件可能导致人体暴露在危险下，换句话说，它可能促成危险情况。软件失效（任何形式）常常促进危险转化为危险情况。

软件促成危险情况主要通过促成暴露危险并产生危险情况的事件序列。

表A.1 危险、可预见事件序列、危险情况和可发生的伤害的关系

危险	涉及软件的可预见事件序列	初始原因	危险情况	伤害
电能	用于控制眼部植入物电流的软件输出过高。	（1）软件算法有局限。 （2）正确地规定了软件，但有反常。	过量电流通过植入物施于患者眼部	严重灼伤 视力损伤
临床功能丧失	软件无法按照设计提供生命支持功能	(1)软件无法处理不常用的输入数据 (2)软件未检出设备的不正确设置 (3)硬件没有提供足够的资源以支持软件的及时运行	(1)器械不能在需要时提供治疗 (2)器械在不正确设置下运行 (3)器械未对威胁生命的情况做出警告	患者状况恶化 死亡
临床人员对患者疏忽	软件输入和输出混淆或误导用户	(1)软件用户界面造成用户迷惑 (2)软件输出超过用户的反应能力 (3)用户不了解软件的限制	(1)对患者的不当治疗 (2)对紧急事件缺乏及时反应 (3)对软件的过分依赖取代人的主动性	患者状况恶化 死亡

附 录 B
（资料性）
软件原因的示例

表B.1列出了通常与危险有关的软件功能领域，并提供了危险潜在原因情形的示例。同时提供了在软件开发中有益于改善风险控制的提问的示例。表中的部分信息可能并不适用于所有医疗器械软件，对特定医疗器械的相关性取决于该医疗器械的预期用途、系统级设计以及软件在该医疗器械中的作用和其他因素。此表仅预期作为分析的起始点。

此表并不详尽，但有助于考虑如何开发安全有效的软件。

表B.1 不同功能领域软件原因的示例

软件功能领域	危险原因示例	提问
报警和警示		
优先级	低优先级报警掩盖了高优先级报警的显示或可听见的输出 关键报警不持续	规范是否识别了系统如何对多报警情况做出反应？ 有多级报警吗？ 高级别报警的声音超过低级别报警的声音吗？ 任何报警宜持续直到用户承认报警吗？
保护性措施	每个报警情形或报警种类的防护措施不清晰 没有规定一旦报警解除就取消防护措施	防护措施会造成可用性问题吗？即用户能安全地从防护措施跳转出来吗？
关机/安全模式/恢复	安全模式措施不充分 安全模式措施引发新危险	安全模式措施对预期用途合适吗？ 临床人员评审过安全模式场景吗？ 安全模式状态对用户来说显而易见吗？
用户界面	拥挤的或糟糕（不良）的用户界面掩盖“真实的”报警状态 响应报警要采取的措施不清晰	临床人员评审过防护措施的可用性吗？
运行日志	持续的错误表明有未解决的故障 运行日志报警与错误的患者相关联	发现的错误在运行日志中记录了吗？ 运行日志空间够大吗？ 运行日志存储可靠吗？ 运行日志如何清除？ 运行日志清除时使用者知道吗？
可听性	背景噪声掩盖了报警声音 报警声音太吵以至于操作者找到其他方式将报警关闭 声音系统失效且用户未意识到	声音报警设计考虑了预期使用环境吗？ 用户参与了用户界面设计的需求开发吗？ 怎样通过上电或由患者验证声音系统？
数据通讯	重要数据源通讯断开导致错过重要报警或数据	重要数据源通讯断开是否有报警提示？
临界电源运行状态		
数据完整性	关机时非易失性的写操作正在进行中 关键参数未被保护以在重启电源时恢复治疗 关键参数在运行期间被潜在的软件反常所覆盖	断电时，进行中的内存写入过程会发生什么？软件知道马上要断电吗？ 通电时非易失性存储被验证了吗？ 在使用前要检查关键参数吗？
复位	意外复位后，未能与组件同步 持续的复位未被检出，而这可能是即将发生故障的信号	复位用作风险控制措施吗？ 复位操作时会危及输入/输出操作吗？ 用户意识到复位吗？
恢复	通电初始化时医疗器械无法用于预期用途 通电时非易失性失效－你该怎么做？ 用户不知道关键参数被恢复到出厂设置	恢复时间是安全问题吗？ 医疗器械可得性是安全问题吗？ 失效安全的防护措施如何影响非易失性存储器？

表 B.1　不同功能领域软件原因的示例 （续）

软件功能领域	危险原因示例	提问
电源模式	在低功率状态时，可听功能或其他关键用户界面不可用 向低功率转换时，无意间关闭了关键中断功能	低功率模式期间，风险控制措施受影响了吗？ 是否已将软件从低功率模式的恢复作为一种可能的启动状态用于验证和确认活动？
关键用户控制/可用性		
调整/导航/选择	当用户界面（UI）软件中的进程处理了值的变化，但控制进程从未接收到新值	如果调整新值但没有选择或确认，用户是否被告知？ 正在被调整的参数是否需要两步操作以实现更改？
数据输入	用户输入了范围之外的值 用户输入在范围内但是非预期值	要提示用户确认吗？ 软件检查数据输入的有效性吗？ 要求监督用户登录以确认非常关键的输入或错误覆盖吗？
可访问性	隐藏停止控制 戴着外科手套时触摸屏控制不起作用	用户要达到安全相关功能必须通过多少“层”？
屏幕变化	警告“自动地”引起屏幕显示转换	所有的自动屏幕转换都被评价了吗？
用户界面设计	颜色的使用没有考虑到常见的色盲 用户不能确定哪种情况会导致报警 界面上使用的标识与常规认知存在差异 界面上叠加显示的区域透视度不适当 层次复杂、路径设置不符合常规	色盲操作者会如何解释错误信息？ 用户界面设计的开发需求有用户参与吗？ 当前所用的标识是否有业界常用显示图形？ 在设计透视度时是否考虑了用户的极端情况？例如，色弱等。 在设计时是否考虑了动作步骤数量的限制，同时是否都有明确的顺序指导？
用户操作	界面操作设计存在在同一位置不停点击触发一些机器功能的运行	是否有在界面同一位置点击触发机器功能运行导致安全风险
显示		
诊断图像	方向相反 患者关联错误 方向标识与实际方向不一致 定位线位置显示错误 图像与报告关联错误	有技术方法去保证正确的图像方向吗？ 图像如何与患者联系起来？ 在旋转、镜像等操作后,是否考虑了更新计算方向？ 在发送、传输等操作后,是否考虑了更新计算位置？ 图像如何与报告关联起来？
诊断波形	不合适的显示滤波器 图形失真、图像变形、缩放比例错误、时间基数错误、有损压缩	需要显示哪些频率成分？ 临床人员评审过需求了吗？ 显示滤波器的特征被完整描述了吗？即拒绝什么？通过什么？超过输入的完全范围吗？
硬件控制		
算法 电机控制	积分饱和、混叠、定时、溢出、端口错误（串行端口/并行端口）	采样率是多少？ 如果使用比例-积分-微分（PID）控制，积分增益是否受限？算法是否对制造的所有硬件变化进行了表征？ 如果使用反馈控制，对反馈信号的有效性进行哪些检查？ 微处理器和编译器使用的所有数据类型都评价了吗？
应用能量	没有在初始时和治疗过程中持续地检验所有的能量“阈值” 安全系统失效但是用户不知道	所有的认定是在按计划持续被验证吗？ “普通模式”错误可在治疗控制软件和安全监控软件中存在吗？ 每次通电或每个患者都做安全监测验证吗？

表 B.1　不同功能领域软件原因的示例 （续）

软件功能领域	危险原因示例	提问
离散	位粘滞； 由于轮询间隔以比特为单位的变化未被检出	软件检出位粘滞（从不变更）了吗？ 轮询速度与系统或硬件工程师讨论过了吗？
校准/自检 检验范围 分析校准 （专用软件校准）	差的使用说明书使得使用者未能对医疗器械进行正确地校正，导致错误的校正常数 对非零信号进行了自动校零操作，即非预期的袖带或线上压力，或传感器受力	软件执行了合理或有效的校准值检查吗？即斜率或偏移量？ 用户知道自动校准或自动校零吗？
硬件故障检验	检测到硬件故障但没有报告给用户 医疗器械继续在这种条件下使用 通电后硬件故障发生； 软件只对供电时对硬件故障进行检查	所有的硬件故障都报告给用户了吗？ 硬件故障检查宜在通电时、每次治疗或间断的或连续的（如每秒一次）进行吗？
自清洁	周期中，用户中止断了清洁或消毒过程	软件强制该周期完成吗？ 软件能检出被终止而未完成的清洁或消毒过程吗？
液体输送	检出不适当的标定 没有在初始时和治疗过程中持续地检验所有的液体“阀门”	所有的认定是在按计划不断验证吗？ 系统安全能被破坏吗？即泵在安全卡中没有管子的情况下在运行。
生命支持	从未定义安全状态 在许多关断路径中，有一条路径不能禁止中断 生命支持功能没有备份	对于目标人群（例如：成人、婴儿）范围,全面定义和分析安全状态了吗(包括治疗延误和安全关机顺序的影响)？ 软件能支持“有限功能”模式并且报告使用者情况吗？
电脑或服务器崩溃	运行关键服务的没有硬件看门狗的电脑或服务器崩溃导致服务失效	是否有故障转移集群（热备机器）？
监视		
决策	监控软件中普通模式错误 竞态条件引发错误的决策结果	治疗控制和治疗监控软件是独立开发的吗？ 对于这个决策点，软件设计是否有排除或使竞态状态的可能性最小化了吗？
未激活	控制系统不知道监控系统已经关闭子系统 网络化系统日志参数在医疗器械中未被激活	控制子系统知道监控子系统的活动吗？ 如何告知用户或网络系统未激活的参数
显示	显示值没有更新但用户未意识到 在两个或更多的优先级执行显示写操作	如何让用户意识到“冻结”的显示？ 在被取代前，视频的“前后内容”保存了吗？
测量	错误的数据采样时间或采样率 对图像进行标准测量，测量结果显示错误	采样率对于信号的频谱合适吗？ 测量值是贯穿所有的软件层以统一的单位储存吗？ 是否按照规定的标准及参数进行转换？
接口		
不良的参数传递	函数以微升为单位进行量值传递，而驱动系统的期望值是以毫升为单位 不好的指针传递 被传递的指针指向临时的内存区域，但在处理之前内存里的值已经丢失	每个软件函数验证所传递的参数吗？ 软件语言支持更强大类型的安全检查吗？ 软件是否被设计为在整个软件包内使用统一的量值单位？ 参数是在更高优先处理层上修改吗？
网络	软件进入死循环，等待主机响应 网络上的医疗器械重“名”导致数据丢失 网络数据处理占据了中央处理器（CPU）处理周期，致使没有资源给安全或预期用途功能使用	软件被设计成与任何物理网络连接条件兼容吗？ 远程连接能通过重复地发命令或假数据降低系统性能吗？ 医疗器械检查网络名是否已被使用了吗？

表 B.1　不同功能领域软件原因的示例 （续）

软件功能领域	危险原因示例	提问
数据		
临床信息	系统访问错误患者记录，而用户界面显示没能使其显而易见 系统以错误的存档方式储存患者数据 意外断电等因素影响了正在处理中数据，造成图像数据失真 不正确的图像信息被保存到DICOM图像中 图像的计算和测量错误 图像显示的信息不正确 图像标注测量错误	能有多个独立的标识符显示以让用户了解检出的混淆吗？ 关键标识符可以嵌入真实数据用作交叉核对吗？ 设计时是否考虑数据完整显示的保证手段？例如，先生成数据后添加数据库记录，没有完成的或者半成品的内容不会存在于系统中。 有技术方法保证正确的图像方向/图像计算和测量/图像显示的信息吗？ 有参考DICOM 标准定义需要保存到图像中的内容和格式吗？
报告	报告提供错误的数据或以错误的顺序标识或没有单位	什么报告会被用于临床？ 如果数据不正确伤害的严重度有多大？ 临床医生有多少可能注意到这个问题？
数据库	由于系统级失效的副作用或未知来源软件（SOUP）副作用导致数据崩溃	怎样在使用前发现数据崩溃？ 可以在每次使用时检查而不只是在启动时检查吗？
数据完整性	数据无法及时存盘而丢失 数据损坏	设计时是否有保证数据保存的实时性的设计手段？
诊断		
决策	伪影侦测指示抑制了显示器上的心脏停搏指示	警报指示的层级得到充分评审吗？并与临床人员进行了评审吗？
数据转换	运算精度错误导致无效结果 运算使用或显示了错误的单位 非法数据导致计算错误 图像传输或存储中出现数据错误	需要什么样的运算精度？ 宜如何进行数学公式编码以保证充分的精度？ 在设计时是否考虑了对非法数据的限制？ 有技术方法保证图像无损压缩吗？ 数据上传或下载过程中如何校验正确性？
自动的预防性维护	后台诊断更改了临时数据但是与此同时应用代码正在检索实际使用的数据 后台诊断干扰了正常的时序 自动功能输出的结果不正确	诊断中，应用过程在适当的时间被锁定了吗 关键的定时周期内诊断被锁定了吗？ 是否采用了可信的算法？是否提供了手动纠正的功能？
信息安全		
配置选项	对关键配置参数或数据的访问没有防护或防护不充分	什么数据是关键的，用户不宜更改或者只有在监督授权时才可以更改？ 需要跟踪监控吗？
功能性访问	对访问治疗控制或仪器运行没有防护或防护不充分	操作前操作者需要登录吗？ 患者能不经意地操作医疗器械吗？

表 B.1　不同功能领域软件原因的示例（续）

软件功能领域	危险原因示例	提问
接口访问	对通过通信接口或网络提交的数据或指令没有防护或防护不充分	什么是被允许远程操作的？ 远程系统设想的控制可靠吗？如果可靠，为什么？
病人数据	新病人数据混在已解除的病人数据中	解除病人操作与数据存储是否同步？
数据安全	通过设备泄露病人隐私信息或安全密钥信息	病人隐私信息、安全密钥是否加密存储？ 日志中是否记录病人隐私信息、安全密钥等信息？ 是否使用了固定密码？
性能		
容量/负载/响应时间	峰值负载时关键时序受到影响 处理/输入/输出的顺序受到影响或峰值负载下数据丢失 在系统峰值负载下电机控制受到影响 由于数据库访问失败，导致软件无法启动	峰值负载或当达到容量限值时，数据或时序会在无察觉的情况下丢失或受影响吗？ 峰值负载下，输入和输出会在正确的顺序下排队等候吗？ 关键的功能和风险控制措施在这些压力条件下测试了吗？ 对探测限值实施了风险控制措施吗？ 能否设置中断将临界时间受限的功能与其他功能隔离？ 设计时是否考虑数了数据库的备份恢复机制？
维修保养		
错误的设备故障诊断信息	故障定位错误 无法定位故障	有什么设计确保故障定位准确？ 能否定位所有潜在故障？
系统校正	系统无法校正 系统校正不准确	有什么设计能够提供必要的系统校正？ 有什么设计能够确保系统校正准确？
错误配置	错误配置触发非正常功能	怎么使用户知道设置被配置错误了？
固件烧录中断	固件烧录中断，嵌入设备无法使用	是否可以复原？
定期维修提醒	必要的系统维修（或系统校正）超过时效，可能带来系统功能失准	有什么设计可以提醒用户需要进行必要的系统维修，例如检测必要校准的时效性？
远程维修	进行远程维修时，此时进行临床使用可能会引起异常	有什么设计可以提示用户当前系统处于远程维修，不建议临床使用？
人工智能		
预测和推理	病灶结果的漏检 病灶结果的误检 预测参数偏差	训练数据是否足够，以便能覆盖所有预期的预测结果？ 是否有更好的算法模型提高预测准确性？ 是否有方法提示用户，这是人工智能预测结果，需要用户最后确认？
数据隐私和安全	患者隐私数据泄露	如何保证用户数据在训练，存储，传输，预测时不被泄露？

除了表B.1显示的安全有关软件功能中危险的潜在软件原因，还有一些类型的软件原因可导致对与发生失效的软件无关的其他软件产生副作用。如果某种软件缺陷对软件的安全有关部分有不可预期的影响，就宜识别这种可能性，并开发风险控制策略及具体的风险控制措施。

表B.2识别了这些危险的潜在软件原因的示例和每个示例要考虑的可能的风险控制措施。基于需求的系统测试，在识别这些软件原因或者验证相关的风险控制措施及风险控制措施的有效性方面常常是无效的。表中的右列提供了适用于每个示例的验证方法类型的指南，包括静态或动态等类型。表B.3提供了静态和动态等验证方法的示例。

表B.2 带来副作用的软件原因示例

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
算法				
除以零	运行时间错误陷阱，防御性编码	◆	◆	D
数字上溢/下溢	范围检查，浮点数表示	◆	◆	D
浮点数取整	鲁棒算法	◆		
不合适的范围/边界检查	防备性编码	◆	◆	S
差一错误(OBO)	防备性编码	◆	◆	
内存泄漏	集中控制内存管理，使用工具排查	◆	◆	D
内存读写异常	内存采坑工具引入，执行严格的编码规范	◆	◆	D
依赖库不一致异常	配置管理角度对于引入库进行统一管理	◆		D
死锁	控制时序执行，进行超时处理	◆	◆	D
系统库异常	异常控制处理		◆	D
第三方库不兼容	隔离控制，版本受控	◆		D
精度不够	合理的数据类型采用	◆		
不同编译器数字格式不一致	使用明确的格式语义定义	◆		T
内存分配碎片	优化内存使用	◆		T
连续内存分配异常	内存分配连续时考虑上限	◆	◆	T
内存占用时间过长	考虑内存声明周期及时释放			T
并行性差、执行效率低	考虑并发设计	◆	◆	
使用不支持异步的数据结构	采用支持重入的数据结构	◆	◆	
未初始化	进行初始化，避免野指针	◆	◆	
字符串处理无编码考虑	考虑操作系统或者语言的字符串编码转换	◆	◆	

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
执行库与头文件不一致	配置管理对于库进行监控	◆		
句柄资源泄漏	统一资源管理，监控工具	◆	◆	
运行时程序域不当	运行视图下的一致性进行保障	◆		
资源释放时机错误	遵循资源申请释放规范	◆	◆	T
缓冲区溢出	考虑缓冲区 buffer，进行异步处理	◆		T
硬件相关				
电可擦只读存储器（EEPROM）用法： 长时间访问，磨损	使用特殊的访问模式（页面模式/突发模式），只在数据变更时写操作，只在失电时高速缓冲存储器写操作和更新电可擦只读存储器（EEPROM）。	◆		T
中央处理器（CPU）/硬件失效	开机中央处理器（CPU）检查，映像程序循环冗余码校验（CRC）检查，随机存取存储器（RAM）测试，时钟检查，看门狗检查，非易失存储器检查，对硬件响应的超时和合理性原因检查，传感器与电源/接地短路检查，用已知信号测试传感器响应		◆	
噪声	防抖数字输入，滤波器模拟输入，全部中断－使用的和未使用的－都有中断服务程序(ISRs)			
外围接口异常	对于模数转换器（ADC）/数模转换器（DAC）启动延迟，验证时序和其他接口需求总是匹配，合理性检查	◆		T
硬件驱动				
驱动与函数 API 不匹配	监控硬件与驱动版本，及时更新		◆	
驱动与操作系统不匹配	针对不同的系统版本，动态执行驱动更新		◆	
驱动设置不当	需要按照驱动要求进行严格的参数设置	◆	◆	
时序				
竞争条件	更新时识别和保护（锁定）共享资源，执行单一的、不能重新进入的进程来处理对共享资源、共享资源分析的所有访问			S

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
错过时间最后时限	特定的时序需求，适当的实时设计算法，通过设计排除优先反转和死锁问题，避免非确定的时序结构，在完整代码中验证所有的时序假定 注：非确定时序结构包括：递归程序，等待来自硬件响应，动态内存分配，和虚拟内存（与硬盘交换存储页）的响应			T
错过中断功能	简单的中断结构（最少的优先级），中断服务程序(ISRs)简短快速，软件中断阻塞是罕见的并且持续时间短，适当的实时设计			T
输出抖动过大	中断服务程序(ISR)简短且执行快速，避免非确定的时序结构，适当的实时设计，在周期任务开始和高优先级的中断服务程序(ISR)时更新所有输出			T
看门狗超时	适当决定看门狗计时器重置和设置暂停时间之间的最长时间间隔，避免危险情况的同时还要尽可能地设置最长周期的暂停时间			T
模式				
异常终止	退出陷阱，运行错误陷阱，适当的看门狗计时器设计，对所有程序输入的有效性检查，上电自检，发布前删除程序中所有“调试”代码和其余非产品功能特性，确保不会无意中进入“特殊”模式（服务，制造）			D
电源丧失/恢复/排序问题	启动检查：中央处理器（CPU）、映像程序循环冗余码校验（CRC）、随机存取存储器（RAM）、时钟、看门狗、非易失存储器、外围设备等等。适当的状态设计，时间平均值初始化，外围设备重新初始化，从非易失性存储库中储存/恢复系统状态，外部电压监控/重置电路			
启动/关闭反常	启动检查（见上方），外围设备和数据适当初始化，适当时使用非易失性内存，适当的状态设计			
进入/退出低电量模式	适当的中断设计	◆		T
数据问题				
数据崩溃	映像随机存取存储器（RAM）、块循环冗余码校验（CRCs）或校验和，通过函数封装数据访问，全局数据减到最少，保持数据结构简单，知道编译器如何在结构中排列数据，避免类型转换（见下方的“错误指针”和“中间数据”）			S
资源竞争问题	共享资源分析（见上方的“竞态条件”）			

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
错误指针	防御性编码：取消定位前做有效性测试，使用强类型语言，将指针使用减到最少，避免指针转换	◆	◆	S
数据转换错误:数据类型转换，缩放	避免类型转换，使用浮点数表示	◆	◆	S
初始化错误	预设置时均变量，上电时所有数据存储器清零	◆	◆	S
范围外的平均数据	确保计算均值时有足够的采样（尤其是上电时），或者将均值预先初始化为已知（或最后一个）好的数值	◆		
翻转	合理性检查	◆	◆	
易失性数据	验证易失性存储器类别用于被硬件、中断服务程序（ISRs）或不同优先级任务更改的所有数据。	◆		S
意外混叠	采样数据的频率至少要大于信号最高频率成分频率的两倍（奈奎斯特准则），限制信号的带宽			
使用中间数据 注：当运算可以被中断或优先占用时,宜从不在全局变量(或共享）上进行一系列计算，而是使用暂时变量运行所有的计算并用单一的、不可中断的指令更新全局变量。	确保设定要及时同步的任意数据同时得到更新，共享资源分析	◆		
界面问题				
更新显示失败	持续更新而不是事件驱动			
人为因素：误用	重构场景的日志，上下文敏感的帮助，简单的用户界面设计			
网络问题，如：多用户	负荷测试			T
硬件/软件配置错误/驱动程序错误	软件开发流程，配置管理工具			
破损的补丁/更新	启动时映像程序循环冗余码校验(CRC)和版本检查，检查协议修订，产品有效期			

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
未知来源软件（SOUP）失效模式：中止/不回复，锁中断时间过长，等	检查 SOUP 勘误表，鲁棒设计（阻塞调用超时检查），锁住经常使用或被 ISR 使用的内存页，仅使用需要的 SOUP 功能特性：除去所有其他功能特性	◆		T
病毒	病毒检查器			
浏览器/网络不兼容	启动时集成版本检查、兼容性测试			
软件				
对象不释放	对象引用及时回收，引入工具排查	◆	◆	
非托管资源泄漏	区分不同资源，按照非托管资源及时处理	◆	◆	
信号量导致的死循环	超时控制，严格执行控制逻辑	◆	◆	
多线程时序问题	引入信号量等同步机制	◆	◆	T
后台线程访问界面	控制线程作用域	◆	◆	
数据库访问异常	异常控制处理	◆	◆	D
文件损坏容错	考虑自恢复机制，保证最小集运行状态	◆	◆	
资源监控异常处理	动态监控资源	◆	◆	
处理性能吞吐量不够	资源组合利用，提高响应时间	◆	◆	D
界面刷新速率与帧率不达标	多线程组合，资源池配置	◆	◆	
超高分辨率显示器不适配	根据显示器 DPI 进行布局与图标处理，考虑矢量图形	◆		
时间日期格式错误	按照标准格式化处理	◆	◆	
多语言环境数字格式错误	按照无关模式处理	◆	◆	
时区时间转换错误	按照 UTC 统一标准处理	◆	◆	
语言字符集未考虑	考虑字符集转换	◆	◆	
数据大对象设计不规范	控制对象内存尺寸	◆	◆	
网络 IP 监听缺失	多 IP 监听或者按需选择 IP 监听	◆		
运行环境加载无热启动	控制模块粒度，按需引入启动	◆	◆	T
全局变量过度使用导致问题复杂化	禁用全局变量	◆		
未按照行业标准提供接口服务	执行行业标准	◆	◆	

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
访问对象读取脏数据	控制读写并发时序	◆	◆	
未考虑下位机超时	引入异常控制机制，隔离封装	◆	◆	
数据传输未加密	使用非对称密钥进行加密处理考虑	◆	◆	S
数据传输未考虑压缩	结合性能考虑动态压缩	◆	◆	S
未启用并行加速	引入并行计算机制	◆	◆	S
未考虑不同操作系统接口差异化	进行差异化模块处理	◆	◆	S
运行环境				
未知来源软件选用不合适	选用稳定可靠长生存周期模块		◆	
操作系统自动更新后不可用	考虑禁用自动更新	◆	◆	
软件兼容环境不够	扩展模块兼容性			
代码执行与杀毒软件冲突	避免使用易被判断为恶意行为的代码，添加信任白名单			
硬件异常限制	根据硬件安全级别控制逻辑	◆	◆	
防呆设计缺失	引入异常控制机制，隔离封装	◆	◆	
可重复性不够	模块化处理	◆	◆	T
压力并发与资源匹配关系无设置	均衡化匹配设计	◆	◆	D
单一的系统依赖	业务模块的接口化与逻辑的提取封装	◆	◆	
配置项自构建触发	配置项异常提供自恢复能力	◆		
运行资源动态监控不足	动态监控资源	◆		T
输出结果一致性不够	按照标准输出结果	◆	◆	
杂项				
内存泄漏	避免动态内存分配	◆	◆	D
系统死锁	简单锁策略（在给定的时间内进程只能有一个资源锁住），死锁分析			S

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
重入	确保所有被中断所调用的（或不同优先级的多任务）功能，包括那些包含在那些第三方库里的功能，都能重入			D
堆栈溢出	堆栈执行时期保护，高位标识，堆栈分析			S
逻辑错误/语法	使用资源代码分析工具（如 Lint）和/或 Max.编译器警告级别 在临界控制点进行双重多样化和相互校验	◆	◆	S
无限循环	循环计数、循环超时设定、看门狗计时器	◆	◆	
资源泄漏	代码审核，资源泄漏工具检查（包括动态内存等）			D
资源重复释放	代码审核			D
浮点精度问题	检查是否用 float 表示大于等于 8e6 的值（数值越大，浮点精度越差）？ 浮点的比较是否考虑了精度？ 是否在不支持硬件浮点的 cpu 上，进行了大量、频繁的浮点运算？			S
存储介质寿命问题	是否超预期的频繁写入存储介质导致存储介质短期内失效？			S
整型溢出问题	通过 32bit 整数来保存一直递增的值（例如毫秒数，tick，收包总数等），是否存在反转问题？			D
代码损坏	启动和运行时映像程序循环冗余码校验（CRC）检查			
无作用程序代码	如未移除无作用程序代码而插入纠错，在无作用程序代码（在定制的或成品软件的组件中）运行后会警示或执行安全关机			D
错误条件代码	确保适当时和仅在需要时使用条件编译	◆		
非预期的宏副作用	在所有宏参数处使用括弧			S
资源枯竭	堆栈，堆和时序分析			T
错误警告/报警优先次序	压力测试			
未授权的功能特性（“镀金”、“后门”等）	需求和设计评审，跟踪矩阵			
操作/优先顺序错误	跟踪代码执行过程中的函数调用顺序，以及函数被调用的上下文信息			
安全状态	独立监视器			

表 B.2 带来副作用的软件原因示例（续）

	验证类型	分析：静态(static) /动态(dynamic)/时序(timing)		
		测试（单元、集成）		
		检查		
软件原因	风险控制措施			
人工智能				
预测和推理 结果不准确	训练数据要足够，以便能覆盖所有预期的预测结果 更好的算法模型提高预测准确性 是否有方法提示用户，这是人工智能预测结果，需要用户最后确认	◆	◆	D
患者隐私数据泄露	访问权限控制，数据存储传输加密	◆	◆	D

有许多方法有利于保证风险控制措施按预期执行，其中有些需要更密集的资源。没有任何单一的方法是充分的。见表B.3中识别的一些方法。

表B.3 有利于保证风险控制措施按预期进行的方法

静态分析	**动态测试**	**建模**
走查	功能测试	环境建模
设计评审	时序和内存测试	时序仿真
寄生电路分析	边界值分析	用例/用户工作流
	性能测试	
	压力测试	
	统计测试	
	错误猜测	
	基于线程的测试	
	基于使用的测试	
	群集测试	

测试宜考虑多种类型（例如，压力、边界、时序、失电、故障、SOUP失效等）以保证安全有关软件在足够大范围的条件下测试，而不是只关注基于需求的测试。

附 录 C
（资料性）
软件有关的潜在隐患

表C.1列出了需要在风险管理活动（针对GB/T 42062—2022的条款）和软件生存周期（针对YY/T 0664—2020的条款）中避免的与软件有关的潜在隐患。

表C.1 需要避免的与软件有关的潜在隐患

GB/T 42062—2022 的第5章：风险分析
——使用不切实际的低概率估计软件失效，造成不切实际的风险评级，导致不适当的风险控制措施。 ——增加了软件功能特性，而没有进行风险分析来确定是否有新的危险和危险情况或原因被引入该医疗器械，或者没有进行风险分析来确定现有的风险控制措施是否受到弱化（可以是在软件最初的开发阶段也可以在软件发布后作为维护的部分）。 ——医疗器械风险分析过程仅定义了系统和硬件层面，既没有充分地阐述软件与风险分析的关系，也没有要求特别考虑软件反常作为危险和危险情况的潜在原因。 ——风险分析和软件开发生存周期程序的严密性与医疗器械的潜在伤害不相适应
GB/T 42062—2022的 5.1 风险分析过程
——风险分析过程只定义了系统和硬件层面，软件仅在对硬件失效实施风险控制措施时被提及。 ——风险分析和软件开发生存周期程序的严格程度与医疗器械的潜在伤害不相适应。 ——软件仅在产品开发生存周期的末期被视为风险分析的一部分
GB/T 42062—2022 的 5.2 预期用途和可合理预见的误使用
——仅考虑用户环境的一个子集/仅考虑潜在的计算机系统平台的一个子集。 ——不考虑平台演变或不考虑对信息安全或其他未知来源软件（SOUP）补丁的需求。 ——对导致潜在危险的误用和使用错误考虑不充分，因而与其相应的风险控制措施没有被识别
GB/T 42062—2022 的 5.3 – 5.4 与安全有关的特性的识别及危险和危险情况的识别
——仅使用失效模式和效应分析（FMEA）或者故障树分析（FTA）方法，好像他们本身就足以进行充分的风险管理。 ——对硬件和软件孤立地进行失效模式和效应分析（FMEA）或者故障树分析（FTA）。 ——忽略整体的危险级别及原因如： • 有不可预见影响的软件错误； • 用于硬件失效风险控制措施的软件逻辑中的错误； • 用于医疗器械预期的临床用途的软件逻辑中的错误（如用于结果计算的算法）； • 软件平台失效－操作系统、库、SOUP； • 计算机组件和外围设备失效； • 通信接口失效； • 人为因素。 ——通过下面的假设进行原因识别： • 软件反常仅仅影响特定组件的功能，而对其他的软件项或数据没有副作用； • 软件会正确地运行； • 潜在的软件失效原因过多，并且在识别、检测或风险控制上不可预测； • 在导致危险情况的软件事件序列的开头或末尾使用风险控制措施就足够了

表 C.1 需要避免的软件相关的潜在隐患（续）

GB/T 42062—2022 的 5.5 风险估计
——假设单一故障条件的概念适用于软件设计问题和事件序列。 ——假设不详尽的测试将某个失效的概率降低至零。 ——基于功能，假设某些软件项与安全无关，而不考虑潜在的非预期的副作用。 ——在没有足够临床知识情况下，或者在没有具有危险对所有潜在用户和目标人群的影响的临床知识（人为因素）的人参与的情况下，就赋值严重度。 ——基于临床医生会发觉失效或错误信息的假设，而赋值低的严重度。 ——基于所有用户都正确地遵循器械标记和手册或者不会因疏忽造成错误的假设，而赋值低的严重度。 ——假设为某个危险计划的一些风险控制措施，并作为赋值初始严重度的一部分考虑。如果此假设有误，低初始严重度可能导致在以后识别的风险控制不充分。 ——仅使用潜在的对患者的直接伤害来判定严重度，而不考虑通过软件提供给用户信息的间接使用，不考虑延迟治疗和其他与医疗器械有效性和基本性能有关的因素。 ——假设临床医生总会反复核对软件提供的信息，或会发觉错误信息而赋值低严重度，且基于此而不实施其他的风险控制措施
GB/T 42062—2022 的第6章 风险评价
——通过对软件反常可能性的主观判断，确定其不需要风险控制措施。 ——消除因硬件特性而导致的软件风险，后来修改或者移除了涉及的硬件在某种程度上使软件成为可能的影响因素，但却未考虑额外的软件风险控制措施。 ——由于假设软件会按预期运行或者假设通过测试会发现所有反常，而没有考虑潜在的软件反常是导致危险的一个因素
GB/T 42062—2022的 7.2 风险控制措施的实施
——风险控制措施在正常条件或限定条件下被验证，而不是在大范围异常和压力条件下被验证。 ——用于完成风险控制措施的软件或数据处于其他软件易于访问到的组件中或位置，使潜在有害的副作用增大。 ——风险控制措施只在一个操作平台或程序变体上被验证。 ——因为很难强制发生（例如内存失效、竞争条件、数据崩溃、堆栈溢出），一些风险控制措施没有得到实际验证。 ——假设在开发过程能发现所有安全有关的反常，测试会保证其在现场正常工作。 ——风险控制措施的实施使软件设计明显更复杂，这种复杂性增加了额外软件反常的可能性或引发新危险
GB/T 42062—2022 的第10章 生产和生产后活动
——当引入另外的风险控制措施时，由于使用错误而忽略了潜在危险的现场事件。 ——没有评价现场信息就假设初始概率和严重度估计是正确的。 ——没有考虑到医疗器械用于非预期的用途，此时已实施的风险控制措施可能不充分。例如用于测试艾滋病病毒（HIV）的体外诊断试剂预期用于个人用途，但却被用于公众血液供应的筛查

表 C.1 需要避免的软件相关的潜在隐患（续）

YY/T 0664—2020 的5.1 软件开发策划
——在软件计划和生存周期过程中没有建立风险管理活动。 ——软件风险管理活动和整体医疗器械的风险管理活动不相关联。 ——软件风险评估只在生存周期的一个阶段进行。 ——软件开发者和测试者未经风险管理培训或没有风险管理经验。 ——假设常规的风险管理活动覆盖了软件风险管理。 ——软件风险没有得到规范化管理。 ——安全决策的可追溯性没有被确立
YY/T 0664—2020 对未知来源软件（SOUP）的考虑
——由于定义软件结构时没有考虑风险评估和风险控制，漏掉固有安全设计风险控制措施。 ——假设测试会使无效的体系结构足够安全。 ——未能识别体系结构安全有关的方面，当这些结构要素随后被改变或删除时，会导致未知的安全风险
YY/T 0664—2020 的 5.4 软件详细设计
——只关注正常情形的处理，且假设组件间的接口和参数的传递将会是正确的，而不是纳入多层级错误检查。 ——对详细设计的头脑风暴和后续评审中，没有考虑识别能导致危险和危险情况的潜在软件失效及相关的风险控制措施。 ——在风险管理活动中忽略软件失效原因（见附录B）
YY/T 0664—2020 的 5.5 软件单元的实现
——相信最好的编码和/或测试过程、惯例、工具或雇员，能弥补差的、固有不安全或过于复杂的设计。 ——在关键的代码开发中使用缺乏经验的开发者。 ——未能定义和要求明确的防御性编程惯例。 ——仅仅依赖动态测试而不执行代码检查或静态代码分析，尤其是对关键组件。 ——在不理解风险管理和设计需求的相关性的情况下而偏离设计。 ——在开发过程早期在关键组件上仅运行一次单元测试，而不把其当作回归测试的一部分重复测试。 ——将测试仅集中于动态、黑盒、系统层面的技术而不执行静态和动态的白盒验证
YY/T 0664—2020 的 5.6 – 5.7 软件集成和集成测试及软件系统测试
——不使用风险评估信息来策划测试或培训测试者。 ——尽管百分之百的测试是不可能做到的，但还是依靠测试作为风险控制措施。 ——没有把系统或软件故障模式的仿真作为测试的一部分来验证风险控制措施。 ——使用没有鉴定的和不受控的自动测试工具并且信任结果。 ——未能正确地分析代码来查明测试中不能发现的反常

表 C.1 需要避免的软件相关的潜在隐患（续）

YY/T 0664—2020 的 5.8 软件在系统级别应用的发布
——未能使已发布的软件和文档版本保持一致，以致对开发和测试团队未来产品的发布造成误导。不准确的文档和不准确的可追溯性会导致失去关联，从而导致忽略危险及其原因，失去风险控制措施或未能充分地验证安全有关软件。 ——未能使有足够临床经验的人参与到剩余反常的评价中。 ——剩余反常重要性的评价仅仅基于检测到的功能特性而非完整的根本原因分析，以判定各种条件下所有潜在的副作用。 ——忽略一旦第三方不再发布特定的SOUP版本，这些版本对于失效调查和现场纠正将不可获得这一事实。 ——因特定工具及其版本（如编译器）没有存档，失去创建特定软件版本的能力。 ——医疗器械的寿命可能会比当前的存档媒介长，当制造商使用新的存档媒介取代旧的存档媒介时，宜策划一个将旧的存档转移到新媒介的途径
YY/T 0664—2020 的 6.1 建立软件维护计划
——制定维护过程时，对于变更的风险管理没有清晰的方法。 ——制定变更的风险管理时，仅描述变更的预期功能，而没有涉及受影响的组件和与其相关的风险
YY/T 0664—2020 的 6.2-6.3问题和修改分析及修改的实施
——假设一个小的功能变化不会影响安全。 ——在没有重新检测现有的风险控制措施和用户界面适当性的情况下，将医疗器械的用途扩展至新的目标人群、新的病征、新的用户类型（如护士而非外科医生）或新的平台。 ——没有确定根本原因和潜在的副作用，基于已报告的现场问题征兆设定问题解决资源的优先级。 ——为非临床目的（如开账单）而设计的软件包含了临床数据，这些临床数据后续用于临床目的，这种情况没有适当的风险管理

附 录 D
（资料性）
生存周期/风险管理矩阵

表D.1列出了YY/T 0664—2020的开发活动和相关的软件风险管理活动。注意该表格并不打算展现严格的按次序排列的瀑布型生存周期。

表D.1 生存周期/风险管理关系表

YY/T 0664—2020 生存周期要求	GB/T 42062—2022		
	5 风险分析	6 风险评价	7 风险控制
5 软件开发过程			
5.1 软件开发策划	风险管理策划 (GB/T 42062—2022 的 4.4) 计划和文档： a) 策划的风险管理活动范围：识别和描述医疗器械和计划中每个要素所适用的生存周期阶段； b) 职责和权限的分配； c) 对风险管理活动评审的要求； d) 风险可接受性准则，基于制造商确定风险可接受性的方针。包括伤害发生概率不能估计时接受风险的准则； e) 综合剩余风险评价的方法和基于制造商确定可接受风险的方针的综合剩余风险的可接受准则； f) 验证活动； g) 与相关的生产和生产后信息的收集和评审有关的活动。		
5.2 软件需求分析	• 分析预期用途、合理可预见的误使用和用户以识别危险 • 识别已知的和可预见的危险与临床医生、患者、服务人员及其他任何接触该医疗器械的人的关系 • 考虑产品阶段：如安装和装配、培训、使用、升级和维护 • 识别能导致危险情况的事件序列或组合 • 估计每个已识别的危险情况的风险，考虑可能后果的严重度 • 4.3 软件安全分级(YY/T 0664 要求)	• 对已识别的风险，确定是否需要降低风险 • 确定仅软件风险控制措施是否就足够了，或硬件风险控制措施是否是必要的或期望的和可行的	• 为已识别的风险（如因硬件失效和使用错误造成）识别软件风险控制措施 • 对可能影响设计（如固有安全设计或防护措施）的软件失效进行软件风险控制措施的初步识别 • 识别用来提高危险情况的可检测度，降低其严重度和/或其发生概率的软件 • 评审新危险的风险控制措施
5.3 软件体系结构设计	• 识别关键数据和组件以及可导致危险的缺陷级别，要特别注意软件原因 • 识别相关危险 • 识别接口：通信内容是什么和何时进行通信 • 评价性能的准则和限值 • 4.3 软件安全分级(YY/T 0664要求)	• 从风险的角度重新评价软件被赋予的角色的可接受性 • 评价受非安全相关功能影响的控制措施的敏感性	• 识别体系结构风险控制措施以隔离关键组件，预防或检出危险特定的软件原因 • 特别注意要提供适当的冗余 • 识别与冗余相关的危险 • 识别用于检出和控制的全局方法

表D.1　生存周期/风险管理关系表（续）

活动	风险分析	风险评价	风险控制
5.4　软件详细设计	• 识别危险的其他潜在原因 • 假设数据、编码和传输错误 • 假设硬件失效	• 重新评价风险控制措施的充分性 • 辨别安全相关代码和非安全相关代码	• 实施特定的风险控制措施和防御性设计/编程惯例 • 完成可追溯性和覆盖率分析来确保风险控制措施得到实施 • 确定是否实现了未明确的功能
5.5软件单元的实现	• 识别危险的其他潜在原因 • 评价同类代码的每个测试失效	• 通过在一系列条件下，通过由有代表性的用户在有代表性的环境下的测试，质疑风险控制措施，重新评价风险控制措施的充分性	• 在一系列条件下，一系列平台上验证风险控制措施 • 最终发布之前进行风险控制措施的回归测试 • 完成可追溯性和覆盖率分析以确保风险控制措施已被实施和测试
5.6　软件集成和集成测试	同上		• 最终发布之前进行风险控制措施的回归测试 • 完成可追溯性和覆盖率分析以确保风险控制措施已被实施和测试
5.7　软件系统测试	同上		• 最终发布之前进行风险控制措施的回归测试 • 完成可追溯性和覆盖率分析以确保风险控制措施已得到实施和测试
5.8　软件在系统级别应用的发布	• 识别配置管理计划，包括配置项和依存关系		• 验证定制软件和SOUP的正确版本已发布 • 验证构建的环境处于配置控制之下
6 软件维护过程			
6.1　建立软件维护计划	• 策划对于变更、增强和修复如何进行风险管理，以及如何监视和分析现场使用信息，以评估风险控制的充分性和额外的风险降低机会。		
6.2　问题和修改分析	• 分析现场性能以识别先前未认识到的或其他的危险及这些危险的原因	• 风险评级和风险控制措施充分性的再评价	• 识别是否需要额外的风险控制措施或是否有必要修正已有的措施
6.3 修改的实施	• 与开发过程相似但要关注变化对以下各项的影响： 　• 影响已有的风险控制措施 　• 引入新的导致危险的原因 　• 引入新的预期用途功能，以至引入新的危险 　• 安全有关代码的回归测试 • 软件发布依照YY/T 0664—2020的5.8		

注：有关遗留软件的风险管理，依照YY/T 0664—2020，4.4.2的要求进行。

附 录 E
（资料性）
安全用例

安全用例是“一种通过提供令人信服的、易于理解的、有效的案例等大量证据支持的结构性的论点，用于表明医疗器械在给定的操作环境里对于给定的预期用途是安全的。”（改编自UK MoD Def Stan 00-56）。

在像军事系统、近海石油、铁路运输和原子能等行业，安全用例的概念是被广泛认可的，这种技术对于医疗器械行业不是强制的，这个附录也不是为了提出超出GB/T 42062之外的要求。

本文件提出的安全用例是一种结构化、文件化和沟通的方式以证明医疗器械的安全达到充分的水平。安全用例也能有助于确保安全在医疗器械整个生命期内得到保持。

安全用例使用风险管理过程的结果来说明为什么软件对于其预期用途足够安全，以及为什么软件满足所有的法规要求（同样适用于相关法规术语）。

可以把安全用例看成风险管理或剩余风险总结，该总结引用风险管理文档中更详细的文档支持信息和证据。安全用例也能包含交叉引用以说明所有风险控制措施的规范和测试覆盖率。

为实施安全用例，需要以下步骤：

——明确说明系统的一系列要求；

——提供支持的证据；

——将要求和证据联系起来的安全论点集合；

——支持论点的假设和判断；

——允许不同的观点和细节水平。

安全用例的主要要素是：

——要求：关于系统或一些子系统的性能；

——证据：用作安全论点的基础。可以是事实（例如基于已建立的科学原理和先前的研究）、假设或从低一层的子论点中产生的子要求；

——论点：将证据和要求联系起来，可以是确定性的、可能性的或定性的；

——推论：提供论点转化规则的机制。

关于安全用例要素和构造的更多信息，见Bishop等人[10]的文章。

Kelly[11]和Weaver等人[12]的两篇文章提供了安全用例介绍和结构化目标注释。

参 考 文 献

注：联合工作组并未认可下列的任何科技参考书的内容，这些书目是以ISO 14971对医疗器械软件要求的应用指南的相关附加信息的方式提供的。

[1] GB/T 7826 系统可靠性分析技术 失效模式和影响分析（FMEA）程序

[2] GB/T 42061—2022 医疗器械 质量管理体系 用于法规的要求

[3] YY/T 1437 医疗器械 GB/T 42062应用指南

[4] IEC 61025, Fault tree analysis (FTA)

[5] IEC 62366-1, Medical devices – Application of usability engineering to medical devices

[6] IEC 80001-1, Application of risk management to information technology (IT) networks incorporating medical devices – Part 1: Roles, responsibilities and activities

[7] 国家药监局器审中心关于发布医疗器械软件注册审查指导原则（2022年修订版）的通告（2022年第9号）

[8] PULLUM, L. Software fault tolerant techniques and implementation. Boston: Artech House,2001

[9] BANATRE, M., LEE, P. (Eds)., Hardware and Software Architectures for Fault Tolerance: Experiences and Perspectives. Berlin, Germany: Springer-Verlag, 1994

[10]BISHOP, P.,BLOOMFIELD,R.(1998),A Methodology for Safety Case Development, Safety-Critical Systems Symposium http://www.adelard.co.uk/resources/papers/pdf/sss98web.pdf

[11] KELLY, T. P., Systematic Approach to Safety Case Management, Proceedings of SAE 2004 World Congress, Detroit, March 2004 (Proceedings published by the Society for Automotive Engineers)

[12] WEAVER, R. A., KELLY, T. P., The Goal Structuring Notation – A Safety Argument Notation, Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004

定义术语的索引