软件验证的一般原则；

行业和 FDA 员工最终指南

文件发布时间：2002 年 1 月 11 日

本文件取代 1997 年 6 月 9 日发布的草案文件《软件验证的一般原则，版本 1.1》。

美国卫生与公众服务部
食品药品监督管理局
器械与放射健康中心
生物制品评估和研究中心

前言

公众评论

贵公司可以在任何时间向食品药品监督管理局，人力资源与管理服务办公室，系统与政策管理分部，文档管理部提交评论或建议（5630 Fishers Lane, Room 1061, （HFA-305）, Rockville, MD, 20852），供部门审议。在提交评论时请提交指导性文件的确切标题。可能直到文件下次修订或更新时，评论才会被机构受理。

如果对本指南的使用或解释有任何问题，请联系器械与放射健康中心（CDRH）的 John F Murry 先生，电话为（301）594-4659，邮箱为 jfm@cdrh.fda.gov。

如果对本指南的使用或解释有任何问题，请联系器械与放射健康中心（CDRH）的 Jerome Davis 先生，电话为（301）827-6220，邮箱为 davis@cdrh.fda.gov。

其他副本

CDRH

文件其他副本可以从网上下载，网址为：www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM085281.htm。贵公司也可以发送请求邮件至 dsmica@fda.hhs.gov 接收指南的电子版，或传真至 301-847-8149 来获取硬拷贝。请使用文件编号（938）来确认贵公司请求的指南。

CBER

其他副本可以从网址：http://www.fda.gov/cber/guidelines.htm 下载，也可以写书面申请至 CBER 交流、培训和制造商协助办公室[（HFM-40）, 1401 Rockville Pike, Rockville, Maryland 20852-1448]，或致电 1-800-835-5709 或 301-827-1800 获取。

目录

第 1 章 目的 1

第 2 章 范围 1

2.1 适用性 2
2.2 受众 2
2.3 最小负担方法 2
2.4 软件验证的法律要求 3
2.4 质量体系法规与上市前提交 4

第 3 章 软件验证的背景 5

3.1 定义与术语 5
3.1.1 要求与规格 5
3.1.2 确认与验证 6
3.1.3 IQ/ OQ/ PQ 7
3.2 软件开发作为系统设计的一部分 7
3.3 软件不同于硬件 8
3.4 软件验证的收益 9
3.5 设计审查 9

第 4 章 软件验证的原则 11

4.1 要求 11
4.2 缺陷预防 11
4.3 时间与努力 11
4.4 软件生命周期 11
4.5 计划 11
4.6 程序 11
4.7 变更后的软件验证 12
4.8 验证范围 12
4.9 审查的独立性 12
4.10 灵活性与责任 13

第 5 章 活动与任务 14

5.1 软件生命周期活动 14
5.2 支持软件验证的典型任务 14
5.2.1 质量计划 15
5.2.2 要求 16
5.2.3 设计 17
5.2.4 构造或编程 20
5.2.5 软件开发者进行的测试 21
5.2.6 用户现场测试 27
5.2.7 维护与软件变更 28

第 6 章 自动化处理器械和质量体系软件验证 30

6.1 需要多少验证证据？ 31
6.2 确定的用户要求 32
6.3 现成软件和自动化器械验证 33

附录 A – 参考文件 35

食品药品监督管理局参考文件 35
其它政府参考文件 36
国际和国家共识标准 37
生产过程软件参考 38
一般软件质量参考 39

附录 B –开发团队 43

软件验证的一般原则

本文件旨在提供指南。本指南代表食品药品监督管理局（FDA）对此主题的最新见解。其不会为任何人创造或赋予任何权利，也不对 FDA 或公众具有约束力。如果替代方法满足适用的法律、法规或其两者的要求，可以使用替代方法。

第 1 章 目的

本指南简要概述了食品药品监督管理局（FDA）认为适用于医疗器械软件的验证或用于设计、开发或生产医疗器械的软件验证的一般验证原则。最终指导性文件版本 2.0 将替代于 1997 年 6 月 9 日发布的草案文件《软件验证的一般原则，版本 1.1》。

第 2 章 范围

本指南描述了如何将医疗器械质量体系法规的某些条款应用于软件以及机构用于评估软件验证系统的最新方法。例如，本文件列出了 FDA 可接受的用于软件验证的内容；但本文件并未列出在所有情况下符合法律规定需要执行的所有行为和任务。

从某种程度上来说，本指南的范围要超过字面意义上所严格定义的验证范围。本指南中所讨论的软件计划、确认、测试、可追溯性、配置管理以及良好软件工程的其他方面均为重要的步骤，并且同样能够帮助得出软件经过验证的最终结论。

本指南建议将软件生命周期管理与风险管理行为相结合。根据待开发软件的预期用途以及与其相关的安全风险，软件开发人员应当明确具体方法、应用到的各种技术结合以及工作的投入规模。虽然本指南不会推荐任何特定的生命周期模式或特定的技术或方法，但指南建议在整个软件生命周期内开展软件验证和确认的工作。

如果软件的开发者为其他人员而非器械的生产商（即成品软件），则软件的开发者可能无法直接负责遵守 FDA 法规。

在这种情况下，需要承担法律责任的一方（及器械生产商）需要对成品软件开发者行为是否恰当做出评估，并明确软件经确认可以满足生产商预期用途还需要做出的努力。

2.1 适用性

本指南适用于：

- 作为医疗器械组件、部件或附件的软件；
- 本身即为医疗器械的软件（如血站软件）；
- 用于器械生产的软件（如器械生产过程中的可编程序控制器）；以及
- 用于实施器械生产商的质量控制体系的软件（如用于记录和存储器械历史记录的软件）。

本文件基于普遍认可的软件验证原则，因此可以适用于所有的软件。对于 FDA 而言，本指南适用于所有与法规监管范围内的医疗器械相关的任何软件，这些法规包括联邦食品药品与化妆品法案（法案）以及现行的 FDA 软件与法规政策。本文件并不特别指明软件是否在监管范围内。

2.2 受众

本指南为下列个体提供有用的信息和建议：

- 受到医疗器械质量体系监管的个体
- 负责医疗器械软件的设计、开发或生产的个体
- 负责设计、开发、生产或采购用于设计、开发或生产医疗器械的自动化工具，或用于实施质量管理体系本身的软件工具的个体
- FDA 研究者
- FDA 合规人员
- FDA 科学审查员

2.3 最小负担方法

我们认为我们应当在医疗器械法规的各个方面考虑最小负担方法。本指南可以体现出，我们对于相关的科学和法律要求进行了仔细的审查，并提供了我们认为符合这些要求的最小负担的方式。但如果贵公司认为有其他的方式能够带来更少的负担，那么请贵公司

与我们取得联系，我们可以考虑贵公司的观点。贵公司可以将书面评论寄送给本指南前言列出的联系人，或寄给 CDRH 监察专员。关于 CDEH 监察专员的详细信息，包括与监察专员的联系方式可以在下列网站找到：

http://www.fda.gov/cdrh/resolvingdisputes/ombudsman.html.

2.4 软件验证的法律要求

FDA 对 1992 年至 1998 年间的 3140 例医疗器械召回进行了分析，结果发现其中 242 例（7.7%）是由于软件故障。在这些与软件相关的召回中，有 192 例（79%）是由于器械最初生产和销售后对软件进行变更而引入的软件缺陷。在本指南中所讨论的软件验证及其他相关的良好软件工程规范是避免这种缺陷及其导致的召回的最主要方式。

软件验证是质量体系法规的要求，该法规于 1996 年 10 月 7 日于联邦公报发布，并于 1997 年 6 月 1 日起实行。（分别见于美国联邦法规（CFR）21 章第 820 部分，以及 61 联邦公报（52602）。）验证要求适用于作为医疗器械组件的软件、本身为医疗器械的软件以及用于器械的生产或实施医疗器械生产商质量体系的软件。

除非在分类法规中特别指明豁免，否则任何与 1997 年 6 月 1 日后开发的医疗器械软件产品，无论其器械类别如何，均适用于设计控制条例。（见 21 CFR §820.30。）该要求包括当前开发项目的完结、所有的新开发项目以及所有对已有医疗器械软件的变更。医疗器械软件验证的具体要求见 21 CFR §820.30（g）。医疗器械软件还要求其他的设计控制内容，例如计划、输入、验证以及审查。（见 21 CFR §820.30。）上述过程相应的记录结果能够为医疗器械软件确认通过的最终结论提供额外的支持信息。

根据 21 CFR §820.70（i）的要求，用于器械生产过程中的任何部分或用于质量体系管理的任何部分自动化的软件均应针对其预期用途进行验证。该要求适用于所有用于器械设计、测试、组件验收、生产、标签、包装、运输、投诉处理过程自动化的软件，以及用于质量管理体系的其他方面自动化的软件。

另外，用于生成、修改以及维护电子记录，以及用于管理电子签名的计算机系统均应符合验证要求（见 21 CFR §11.10（a）。）该计算机系统必须进行验证以确保准确性、可靠性、与预期性能一致，并能够识别无效的或改变的记录。

用于上述用途的软件可以在公司内部开发，也可以通过合同订单的方式开发。但通常针对特定的预期用途来购买现有的软件。所有的产品和/或质量控制体系软件，即使是购买现成软件，也应当明确记录其预期用途，以及可以用于比较的检测结果或其他证据的信息，以表明该软件经验证适用于其预期用途。

自动化医疗器械以及自动化生产和质量体系运行正越来越多地使用现成软件。现成软件可能有许多功能，而其中仅有一小部分是生产商所需要的。器械生产商有责任确保软件可以合适地应用于其医疗器械或医疗器械的生产。如果器械生产商购买了“现成”的软件，则其必须确保该软件能够在其选定的应用条件下按照预期工作。对用于生产或质量体系中的现成软件，本文件的 6.3 章节部分有其他的指南。对医疗器械中的软件，可以在 FDA 的文件《在医疗器械中使用现成软件的行业、FDA 审查者及符合性指南》中找到其他的有用信息。

2.4 质量体系法规与上市前提交

本文件针对涉及软件验证实施的质量体系法规提出了应对方案。其提供了管理及控制软件验证流程的指导意见。软件验证过程的管理与控制不应当与任何其他验证要求混淆，如自动化生产流程的验证。

器械生产商可能使用同样的流程和记录来遵守质量体系和设计控制的要求，以及向 FDA 提交的上市前提交。本文不涉及任何与软件验证相关的安全性和有效性具体问题。本文也不解决设计方面的问题以及受监管软件的上市前提交所要求的文件资料。与安全性和有效性相关的具体问题，以及上市前提交所需要的文件资料应当由器械与放射健康中心（CDRH）的器械评估办公室（ODE）或生物制品评估和研究中心（CBER）的血液研究与审查办公室来确定。适用于上市前提交的 FDA 指导性文件见附录 A 的参考文件。

第 3 章 软件验证的背景

有许多人请求关于如何确保满足 FDA 所要求的确保在软件验证方面符合质量体系法规的具体指南。本文件所提供的关于软件验证的信息并非全新。根据第 4 章和第 5 章列出的原理和任务所进行的软件验证工作已经应用于软件行业多个领域超过 20 年。

由于医疗器械、工艺以及生产设施种类繁多，所以无法在一份文件内列出所有适用的具体验证内容。但几个较为具有一般适用性的初步概念可以很好地作为软件验证的指南。这些宽泛的概念为确定软件验证的详细方法提供了可接受的框架。其他具体信息可以参考附录 A 中列出的一系列参考文件。

3.1 定义与术语

除非由质量体系法规规定，或在下文中特别指明，否则本指南中所有使用的其它术语均根据当前版本的 FDA《计算机化系统和软件开发的术语表》确定。

医疗器械的质量体系法规（21 CFR 820.3（k））将“确定”定义为“明确、记录并实施”。本指南中出现的“确定”及“已确定”应当解释为相同的意思。

医疗器械质量体系法规中的一些定义与软件行业常用的词汇相比可能会存在混淆。例如要求、规格、确认及验证。

3.1.1 要求与规格

虽然质量体系法规声明必须记录设计输入的要求，且必须对特定的要求进行确认，但规范并没有进一步指明“要求”与“规格”两个术语之间的差别。**要求**指的是对于一个系统或其软件的任何形式的需要或期望。要求反映了消费者明确指出的或暗示的需要，可能是基于市场、合同或法律的，也可能是一个组织的内部要求。可能存在多种不同类型的要求（如设计、功能、实施、界面、性能或物理要求）。软件要求通常来自于涉及该系统分配给软件的系统功能方面的系统要求。软件要求通常以功能的方式进行描述，并且作为开发项目流程的形式进行确定、修改和更新。进行准确完整的软件要求记录是顺利完成软件验证的关键因素。

规格的定义为“说明要求的文件。”（见 21 CFR§820.3（y）。）其可能会涉及或包含图像、模型或其他相关文件，并且通常会指明检查器械是否符合要求的方法和标准。有多种不同类型的书面规格，例如系统要求规格、软件要求规格、软件设计规格、软件测试规格、软件集成规格等。所有的这些文件共同组成“规格要求”，也是各种验证工作所必须的设计输出。

3.1.2 确认与验证

在 ISO 8402:1994 中“确认”与“验证”是独立的不同的术语，而质量体系法规也与其协调一致。另一方面，许多软件工程期刊文章和教科书将“确认”与“验证”混合使用，或者有时候将软件的“确认、验证与检测（VV&T）”作为一个独立的概念，对这三者不进行区分。

软件确认提供了客观的证据来证明软件开发生命周期特定阶段的设计输出满足该阶段所有的具体要求。软件确认过程关注软件开发过程中的一致性、完整性、正确性及其支持的文件资料，并为软件已验证的后续结论提供支持。软件测试是众多确认过程中的一种，其旨在确认软件开发输出能够满足输入要求。其他的确认工作包括多种静态和动态分析、编码和文档检查、走查以及其他技术。

软件验证是针对成品器械的设计进行验证的一个部分，但在质量体系法规中没有单独定义。在本指南中，FDA 认为软件验证指的是**“通过检查和提供客观的证据来确认软件的规格符合用户需求以及预期用途，并且通过软件实施的特殊要求能够始终如一地实现。”**事实上，软件验证过程可能同时在软件研发周期的过程中和结束后进行，以确保所有的要求得到满足。因为软件通常作为大型硬件系统的一个组成部分，因此软件的验证通常要包含相应的证据来证明所有的软件要求已恰当、完整实施，并且可追溯到系统要求。软件是否通过验证的结论高度依赖于软件开发生命周期各个阶段开展的综合性软件测试、检查、分析以及其他确认工作。软件自动化器械的整体设计验证项目通常包含在模拟使用环境下的器械软件功能测试以及用户现场测试。

软件确认和验证十分困难，因为一名开发者无法永远进行测试，而却也很难知道多少证据是足够的。软件确认的关键在很大程度上是确定能够证明该器械满足了软件自动化功能及器械特征所有要求和用户期望的“置信水平”。如规格文件中的缺陷，剩余缺陷估计，测试范围和其他技术的措施都用于在运输产品之前开发可接受的置信水平。置信水平，也即需要的软件验证、确认和测试水平会随着该器械的自动化功能可能引起的安全性风险（危害）的不同而不同。关于软件安全风险管理的其他指南信息参见 FDA《医疗器械所含软件上市前提交内容指南》的第 4 章以及附录 A 中引用的国际标准 ISO/IEC

14971-1 和 IEC 60601-1-4。

3.1.3 IQ/ OQ/ PQ

很多年以来，FDA 和受管辖行业均试图在流程确认术语库的背景下理解和定义软件验证过程。例如，行业文件和其他的 FDA 验证指南有时候会描述使用安装确认（IQ）、操作确认（OQ）以及性能确认（PQ）的方式描述用户现场软件验证过程。关于 IQ/ OQ/ PQ 的定义以及其他信息参见 FDA 于 1987 年 5 月 11 日发布的《流程验证的一般原则》以及 1995 年 8 月发布的《计算机化系统和软件开发术语表》。

虽然 IQ/OQ/PQ 术语很好地达到了目的，是在用户现场组织软件验证任务的许多合法方式之一，但是许多软件专业人士可能不太清楚这个术语，并没有在本文档的其他地方使用。然而，FDA 人员和器械制造商都需要了解这些术语上的差异，因为这些术语要求并提供有关软件验证的信息。

3.2 软件开发作为系统设计的一部分

通常在系统设计过程中决定使用软件来实施系统功能。软件要求通常来自于整体的系统要求以及该系统中拟使用软件实施部分的设计要求。成品器械会有相应的用户需求和预期用途，但用户通常不会明确指出通过硬件、软件还是两者组合来满足这些要求。因此，软件验证工作必须在系统的整体设计验证背景下进行考虑。

要求规格文档代表了产品开发需要依据的用户需求和预期用途。软件验证的主要目标是证明所有完成的软件产品符合所有记录的软件和系统要求。系统要求和软件要求的正确性和完整性应当作为器械设计验证流程的一部分来完成。软件验证过程包括确认所有的软件规格的合规情况，以及确认所有的软件要求可追溯至系统规格要求。确认过程是整体设计验证的重要部分，用于确保医疗器械所有方面均符合用户要求和预期用途。

3.3 软件不同于硬件

虽然软件有很多工程任务与硬件相同，但软件具有一些非常重要的不同点。例如：

- 绝大多数的软件问题可追溯至设计和开发过程中的错误。虽然硬件产品的质量非常依赖于设计、开发和生产，但软件产品的质量主要取决于设计和开发过程，而对于软件生产的关注度较低。软件生产的过程由可以很容易进行确认的复制过程组成。生产数千份与初版功能完全相同的程序备份并不难；难点在于初版程序如何才能满足所有的规格要求。

- 软件最重要的一个特征是分支结构，即能够根据不同的输入执行不同序列的指令。这一特点也是软件的另一个特征—复杂性形成的主要因素。即使是很短的程序也非常复杂，很难完全理解。

- 通常来说，单独的测试过程无法完全确认软件是否完整和正确。除了测试，还需要其他的验证技术以及结构化的、记录的开发流程来确保全面的验证方法。

- 与硬件不同，软件不是一个物理实体，不会磨损。事实上，由于一些原先未被发现的缺陷发现和排除，软件可能随着使用不断改进。但软件是不断更新和改变的，这种改进有时候会因为更改过程中引入了新的缺陷而抵消。

- 与一些硬件故障不同，软件的故障发生前不会有征兆。软件的分支结构使得软件能够在执行中遵循不同的路径，这可能会掩盖掉一些未被发现的缺陷，直到软件产品进入市场一段时间后才被发现。

- 与软件特性相关的另一个问题是其更改的速度和难易程度。这一因素会使得软件和非软件专家认为软件问题能够很容易纠正。再加上对软件的理解不够深入，管理者可能会认为软件并不需要与硬件类似的严格控制的工程管理。而事实上，结果恰恰相反。**由于软件的复杂性，软件开发过程应当有比硬件更为严格的控制，这样才能避免出现在开发过程中难以检测到的问题。**

- 软件程序中一些看上去不重要的软件代码的更改能够引起非常严重的非预期问题。软件开发过程应当进行充足的计划、控制和记录，从软件更改过程中检测和纠正非预期结果。

- 由于软件专业人员需求量大、劳动力流动性强，对软件进行维护性更新的软件人员可能没有参与到最初的软件开发过程中。因此，准确和详细的记录十分重要。

- 在过去，软件组件部分可能不像硬件组件一样多次标准化和可交换。但医疗器械软件的开发者已经开始使用基于组件的开发工具和技术。面向对象的方法和使用现成软件组件有望进行快速、低成本的软件开发。但基于组件的方法要求在集成过程中非常小心。在进行集成前，需要花时间完整地定义和开发可重复使用的软件代码，并深入理解现成软件组件的行为。

基于上述和其他原因，软件工程需要比硬件工程更高水平的管理审查和控制。

3.4 软件验证的收益

软件验证是用于确认器械软件质量和软件自动化操作的重要工具。软件验证能够增加器械的实用性和可靠性，能够降低故障率、减少召回和纠正措施、降低对患者和用户的风险，并降低器械生产商的责任风险。软件验证能够通过更为便捷、低成本的方式来对软件进行可靠的更改及再次验证这些更改，从而减少长期成本。软件维护成本占据了软件整个生命周期内的很大一部分总成本。已经确定的全面软件验证流程能够通过降低后续软件发布的验证成本的方式帮助降低软件的长期成本。

3.5 设计审查

设计审查是针对软件设计的全面系统的已记录检查，用于评估设计要求是否恰当，以及评估设计是否满足这些要求，以及发现问题。虽然在软件项目内开发团队可能有许多非正式的技术审查，但正式的技术审查更为结构化，并且包含了开发团队外的参与者。规划式设计审查可能参考或纳入来自于其他正式或非正式审查的结果。设计审查可能单独针对软件进行，或者在软件与硬件集成入系统后进行，也可能两者均进行。设计审查应当包含对开发计划、要求规格、设计规格、测试计划和流程、所有的其它与项目相关的文件和工作、确定的生命周期每个阶段的确认结果以及整体器械的验证结果的审查。

设计审查是管理和评估开发项目的主要手段。例如，通过规划式设计审查管理者能够确保达到软件验证计划确定的所有目标。质量体系法规要求在器械设计流程中至少开展一项规划式设计审查。但我们推荐进行多项设计审查（例如在每个软件周期的设计工作结束后，且在准备继续进行下一项工作前）。在要求的设计工作结束或临近结束，主要资源用于特定的设计方案前，规划式设计审查尤其重要。在这个时间点发现的问题能够很容易解决，节省时间和金钱，并降低错过关键问题的可能性。

在规划式设计生产过程中会对一些重要问题的解答进行记录。其中包括：

- 软件生命周期的每个环节是否已经确定了恰当的任务以及预期结果、输出或产品？

- 软件生命周期的每个环节的任务以及预期结果、输出或产品是否：

  ✓ 在正确性、完整性、一致性和准确性方面是否符合其它软件生命周期工作的要求？

  ✓ 满足该项任务的标准、惯例和协议？

  ✓ 为启动下一项软件生命周期任务确定恰当的基础？

第 4 章 软件验证的原则

本章列出了软件验证需要考虑的一般原则。

4.1 要求

软件要求规格文件提供了软件验证和确认的基线。软件验证过程需要确定软件要求规格才能完成（参考文件：21 CFR 820.3（z）和 （aa）和 820.30（f）和 （g））。

4.2 缺陷预防

软件的质量保证需要注意不要在软件开发过程中引入缺陷，不要试图在软件代码编写完成后再引入“检测质量”。软件测试在软件代码中显露缺陷方面的能力十分有限。例如，大多数软件的复杂度很高，因此无法彻底进行测试。**软件测试是一项必要的工作。但在大多数情况下测试本身不足以使我们确信软件适合其预期用途。**为了建立这种确信，软件开发者需要使用多种方法和技术来防止软件错误并且检测出未发生的软件错误。方法的“最佳组合”取决于许多因素，包括开发环境、应用、项目大小、编程语言及风险。

4.3 时间与努力

完成软件验证的方案需要时间与努力。软件验证的准备工作应当尽早展开，例如在设计阶段、开发计划阶段以及设计输入阶段。软件经过验证的最终结论应要建立在整个软件生命周期中执行的既定工作所收集的证据上。

4.4 软件生命周期

软件的验证工作在确定的软件生命周期环境下展开。软件生命周期包含软件工程任务以及支持软件验证工作必要的文档记录。另外，软件生命周期还包含适用于该软件预期用途的具体确认和验证工作。本指南不推荐任何特定的生命周期模型—需要在软件开发项目中进行选择和使用。

4.5 计划

软件验证流程通过使用计划来确定和控制。软件验证计划明确了在软件验证工作中“什么”是要完成的。软件验证计划是一种重要的质量体系工具。软件验证计划确定了工作领域，如范围、方法、资源、进度表，以及活动、任务和工作项目的类型和程度。

4.6 程序

软件验证的过程通过使用程序来执行。这些程序确定了"如何"来进行软件验证工作。程序应当明确完成每一项验证活动、任务和工作项目所必须采取的特定行动或行动顺序。

4.7 变更后的软件验证

由于软件的复杂性，一些看起来很微小的局部改变就有可能对整个系统产生重大影响。在软件发生任何变更时（即使是很小的改变），软件的验证状态需要重新确定。**如果软件发生了变更，需要进行验证分析，不仅为了验证每一处改变，也需要确定整个软件系统改变的程度和影响。**在这一分析的基础上，软件开发者随后应当进行恰当水平的软件回归测试，来证明系统中未发生改变但容易受到影响的部分没有受到负面影响。设计控制和恰当的回归测试能够提供信心表明软件在发生变更后仍然可以通过验证。

4.8 验证范围

验证的范围应当基于软件的复杂性和安全风险确定—而不是公司规模或资源限制。验证活动、任务和工作项目的选择应当符合软件设计的复杂性以及与软件特定预期用途使用相关的风险。对于低风险器械，可能仅需要基础的验证活动。随着风险的增加，需要增加其他的验证活动以包含额外的风险。验证记录文件应当足够证明所有的软件验证计划和流程均成功完成。

4.9 审查的独立性

验证活动应当在最基本的质量保证原则"审查的独立性"的基础上进行。自我验证非常困难。如果可能的话，独立的评估更好，尤其对于高风险的应用而言。一些公司以签约承包的方式寻找第三方机构进行独立的验证和确认，但这种解决方案并非总是可行的。另外一种方式是由未参与到特定设计或实施中，但对于项目的评估以及执行验证和确认活动有充足知识的内部员工完成。较小的公司为了保证审查的内部独立性可能需要在如何组织和分配任务方面更有创造性。

4.10 灵活性与责任

在具体实施这些软件验证原则时，在不同情况下可能会有很大差别。医疗器械的生产商在选择如何应用这些验证原则时有一定灵活性，但是仍然要对证明软件通过验证承担最终责任。

软件的设计、开发、验证和监管在多种环境下进行，适用的医疗器械种类繁多，风险水平也各不相同。FDA 监管的医疗器械应有包括下列情况的软件：

- 作为医疗器械的一个组件、部件或附件；
- 本身即为医疗器械；或
- 用于生产、设计、开发或质量体系的其它部分。

在每种环境下，可能会使用不同来源的软件组件来确定应用（如自行开发的软件、现成软件、合同软件、共享软件）。另外，软件组件来源形式也有所不同（如应用软件、操作系统、编译器、调试器、配置管理工具及许多其他形式）。在这些环境下进行软件验证是一项复杂的工作；因此，恰当的做法是在设计软件验证流程的时候就考虑到所有的这些软件验证原则。按照原则制订出的软件验证流程应当与系统、器械或工艺相关的安全风险相适应。

软件验证活动和任务可能是分散的，由不同的组织在不同的地点开展。但无论任务分配、合同关系、组件来源或开发环境如何，医疗器械的生产商或规格的制定者都需要对确保软件经过验证负有最终责任。

第 5 章 活动与任务

软件的验证工作通过软件开发生命周期不同阶段计划和执行的一系列活动与任务完成。这些任务可能是一次性的，也可能多次迭代，这取决于所使用的生命周期模型以及随着软件项目的进展而发生改变的范围。

5.1 软件生命周期活动

本指南不推荐使用任何特定的软件生命周期模型。软件开发者应当确定适合于其产品和组织的软件生命周期模型。软件生命周期模型的选择应当覆盖软件从产生到报废的整个周期。典型的软件生命周期模型活动包括下列内容：

- 质量计划
- 系统要求确定
- 详细的软件要求规格
- 软件设计规格
- 构架或编写代码
- 测试
- 安装
- 操作与支持
- 维护
- 报废

支持软件验证的确认、测试及其他任务包含在上述活动中。软件生命周期模型以多种方式组织这些软件开发活动，并提供了监测和控制软件开发项目的框架。各种软件生命周期模型（如瀑布式开发、螺旋式开发、快速原型开发、增量开发等）可见于 1995 年 8 月 FDA 发布的文件《计算机化系统与软件开发词汇表》。附录 A 引用了上述及多种其他类型的软件开发生命周期模型。

5.2 支持软件验证的典型任务

对于每一种软件生命周期活动，存在支持软件已验证结论的特定“典型”任务。但需要执行的具体任务、执行顺序以及执行的迭代和计时都要受到具体选用的软件生命周期模型和与软件应用相关的安全风险的决定。对于风险非常低的应用，可能某些任务完全不需要。但软件开发人员至少应当对每一项任务进行考虑，并应当确定并记录那些任务是否适合具体应用。下面的讨论非常宽泛，不会规定任何特定的软件声明周期模型或任务执行的特定顺序。

5.2.1 质量计划

设计和开发计划的最后应当给出一个计划，该计划明确了必要的任务、异常情况报告和解决流程、必要的资源，以及包括规划式设计审查在内的管理审查要求。软件生命周期模型以及相关的活动应当予以明确，每个软件生命周期活动必要的任务也应当明确。计划应当包含：

- 每项生命周期活动的具体任务；
- 列举出重要的质量因素（如可信性、可维修性和可用性）；
- 每项任务的方法和流程；
- 任务验收标准；
- 确定和记录产品输出是否符合输入要求评估的标准；
- 每项任务的输入；
- 每项任务的输出；
- 每项任务的作用、资源和责任；
- 风险和假设；以及
- 用户需求文档。

管理者必须确定并提供恰当的软件开发环境和资源。（见 21 CFR §820.20（b）（1）和（2）。）一般来说，每项任务均需要人力和物力资源。计划应当明确每项任务的人力、设施和器械资源，以及风险（危害）管理要起到的作用。应当开发配置管理计划，用于引导和控制多项平行开发活动，以及确保恰当的交流和记录。为了确保构成软件系统的所有规格文件获批版本、源代码、目标代码以及测试套件能够正确恰当地对应，有必要采取控制。控制也应该确保能够准确识别并有权使用当前的批准版本。

需要确定用于报告和解决在验证或其他活动过程中所发现软件异常的流程。管理部分应当识别这些报告并指出每项报告中的内容、格式以及负责的组织要素。审查和软件开发结果的审批也有必要建立流程，包括相应审查和批准的责任组织要素。

<u>典型任务—质量计划</u>

- 风险（危害）管理计划
- 配置管理计划
- 软件质量保证计划
  - 软件确认和验证计划

- ❑ 确认和验证任务及验收标准
- ❑ 计划表和资源分配（用于软件确认和验证行为）
- ❑ 报告要求

- 正式设计审查要求
- 其他技术审查要求

- 报告的问题和解决流程
- 其他支持行为

5.2.2 要求

要求开发包括医疗器械及其预期用途相关信息的确定、分析和记录。特别重要的部分包括系统功能在硬件/软件中的分配、操作状态、用户特性、潜在危害和预期任务。另外，要求还应当清晰地说明软件的预期用途。

软件要求规格文档应当包含关于软件功能的书面定义。在没有预先确定并记录软件要求的情况下进行软件验证是不可能的。典型的软件要求规定了如下内容：

- 所有的软件系统输入；
- 所有的软件系统输出；
- 软件系统要执行的全部功能；
- 软件需要满足的所有性能要求，（如数据吞吐量、可信性以及定时）；
- 所有外部和用户接口，以及内部软件与系统接口的定义；
- 用户如何与系统交互；
- 错误的构成以及处理错误的方法；
- 所需的相应时间；
- 如果是设计限制的话，应当指出软件预期的运行环境（如硬件平台、操作系统）；
- 所有软件可接收的范围值、限制值、默认值以及特定值；以及
- 软件拟实施的所有与安全相关的要求、规格、特征或功能。

软件安全要求来自于与系统要求开发流程紧密相连的技术风险管理流程。软件要求规格应当明确确定系统软件故障能够引发的潜在危害，以及软件需要实施的安全要求。需要对软件故障的结果进行评估，同时也要对减少该故障的措施进行评估（如通过硬件减少故障、防错性程序设计等）。通过这些分析，应当能够确定预防危害最恰当的必要措施。

质量系统法规要求建立解决不完整、模糊或相互冲突的要求的机制。（见 21 CFR 820.30（c）。）软件要求规格中所确定的每项要求（如硬件、软件、用户、操作员界面以及安全性）应当进行准确性、完整性、一致性、可检测性、正确性和清晰性方面的评估。例如，软件要求应当进行评估并确认以下内容：

- 各种要求间不存在内部不一致性；
- 系统的所有性能要求均清晰阐明；
- 错误耐受度、安全性和保障性要求均完整恰当；
- 软件功能的分配准确完整；
- 软件要求适合于系统危害情况；以及
- 所有的研究均使用可测量的或可客观确认的语言叙述。

应当执行软件要求的可追溯性分析来将软件要求追溯至（以及从）系统要求和风险分析结果。除了各种用于确认软件要求的其他分析和记录外，还建议在软件设计工作全面展开前进行一项规划式设计审查来确认要求明确且恰当。要求可以分批审批和发布，但需要注意软件（和硬件）要求之间的交互和接口应当进行恰当地审查、分析和控制。

<u>典型的任务—要求</u>

- 初步的风险分析
- 可追溯性分析
  - 从软件要求到系统要求（反之亦然）
  - 从软件要求到风险分析
- 用户特征描述
- 列举主副存储器的特点和局限性
- 软件要求评估
- 软件用户界面要求分析
- 系统测试计划生成
- 验收测试计划生成
- 模糊审查或分析

5.2.3 设计

在设计过程中，软件要求规格转化为针对待实施软件的逻辑性和实际性的描述。软件设计规格是软件要做什么以及软件如何做的描述。由于项目的复杂性或者为了让不同等级、承担不同技术责任的人员清晰地理解设计信息，设计规格可能会同时包含设计的高度概括信息以及详细的设计信息。软件设计规格完成后会将程序员/编程员限制在商定的要求

和设计范围内。完成的软件设计规格能够避免工程师进行点对点设计决策。

软件设计需要解决人为因素。由于设计过于复杂，或者与用户操作过程中的直观期望相反而引起的使用错误是 FDA 遇到的最为常见和严重的问题之一。通常软件的设计是造成这种使用错误的因素之一。人因工程应当融入整个设计和开发流程，包括器械设计要求、分析和测试。在开发流程图、状态图、原型工具和测试计划中均应当考虑器械安全性和可用性因素。同时还应当执行任务和功能分析、风险分析、原型测试和审查以及完整的可用性测试。在实施这些方法时应当纳入来自用户人群的参与者。

软件设计规格应当包括：

- 软件要求规格，包括预先确定的软件验收标准；
- 软件风险分析；
- 开发流程和编程指南（或其他编程流程）；
- 描述程序拟运行的系统背景的系统文档(如语言描述或环境描述图)，包括硬件、软件与物理环境的关系；
- 使用的硬件；
- 测量或记录的参数；
- 逻辑结构（包括控制逻辑）和逻辑流程步骤（如算法）；
- 数据结构和数据流程图；
- 变量定义（控制和数据）和这些变量使用环境的描述；
- 错误、报警和警告信息；
- 支持性软件（如操作系统、驱动程序及其他应用软件）；
- 通信线路（软件内部模块间的连接、与支持性软件的连接、与硬件的连接、与用户的连接）；
- 安全措施（物理和逻辑安全措施）；以及
- 上述内容未指明的其他局限性。

上述的前四条内容通常为独立的已有文件，软件设计规格可以直接引用。软件要求规格在关于软件风险分析的前述章节已有讨论。书面开发流程能够作为组织的指南，书面编程流程能够作为每位程序员的指南。因为软件的验证无法脱离其所发挥功能的环境知识，因此还需要参考系统文档。如果上述的内容有部分不包含与软件中，则清晰地陈述这些情况（如程序中没有错误信息）对于将来软件的审查者和维护者将会有所帮助。

在软件设计过程中的这些工作有几个原因。进行软件设计评估是为了确定设计是否完整、正确、一致、准确、可行以及可维护。在设计过程中适当考虑软件结构（如模块结构）能够降低将来需要对软件进行更改时的验证工作规模。软件设计评估可能包括对控制流、数据流、复杂度、定时、选型、内存分配、关键性分析以及许多设计其他方面的分析。需要进行可追溯性分析来确认软件设计满足了所有的软件要求。作为一种识别不满足需求部分的技术，可追溯性分析也应当确认设计的所有方面均可追溯至软件要求。需要进行通信线路的分析来评估提出的设计与硬件、用户及相关的软件要求的关系。软件风险分析应当重新进行检查，确定是否有新的危害发现以及设计是否引入了新的危害。

在软件设计工作的最后、实施设计之前，应当进行一项规划式设计审查来确认该设计正确、一致、完整、准确和可测试。设计的各个部分可以分多次批准和发布用于实施；但应当注意对不同部分之间的交互和通信线路进行适当的审查、分析和控制。

大多数软件开发模型是迭代式的。这可能导致软件要求规格和软件设计规格都有若干个版本。所有获批的版本都应当根据确定的配置管理流程实现和控制。

典型任务—设计

- 更新的软件风险分析
- 可追溯性分析—从设计规格到软件要求（反之亦然）
- 软件设计评估
- 设计通信线路分析
- 模块测试计划生成
- 集成测试计划生成
- 测试设计生成（模块、集成、系统和可接受性）

5.2.4 构造或编程

用于新应用的软件可以通过编写代码（如编程）或通过将已有的软件组件组装（如从代码库、现成软件等）的方式来构造。编写代码是将详细的设计规格通过源代码的形式实施的过程。编写代码是软件开发流程概念的最低抽象水平。这是分解软件需求、将模块规格转化为编程语言的最后一步。

编写代码通常会使用高级的编程语言，但对于时间要求严格的操作也可能使用汇编语言（或微代码）。源代码在应用于目标硬件平台时可能会进行编译或解释。在选择编程语言和软件构建工具（汇编器、连接器和编译器）时应当考虑与之对应的质量评估任务（如选择的编程语言是否有调试和测试工具）的影响。一些编译器提供了可供选择的用于错误检查的水平和指令，用于辅助代码调试。不同等级的错误检查工具在整个代码编写过程中均可使用，编译器的警告或其他信息也可能会记录或不记录。但在代码编写和调解阶段的最后，通常会进行最为严谨的错误检查，用于记录软件中仍然存在的编译错误。如果在软件最后的源代码转译过程中没有使用最为严格的错误检查方式，则需要记录使用较为宽松的转移错误检查方式的正当理由。同时，在最后的编译过程中应当记录编译过程及结果，包括所有的警告或来自编译器的其他信息以及解决方式，或者记录不解决这些问题的正当理由。

公司经常使用特定的代码指南，这些指南规定了与软件代码编写过程相关的质量策略和流程。需要对源代码进行评估，以确认其符合特定的代码指南。上述指南应当包括与清晰性、代码风格、复杂度管理以及注释相关的编码惯例。代码注释应当提供模块有用的描述性信息，包括预期的输入和输出、引用的变量、预期的数据类型以及要进行的操作。应当对源代码进行评估已确认其符合相应的详细设计规格。已经完成并准备好进行集成和测试的模块应当记录其是否符合代码指南及其他适用的质量政策和流程。

源代码评估常采用代码检查和代码走读的形式进行。这种静态分析为代码执行前检测错误提供了非常有效的方法。通过这种方法可以单独检查每一项错误，还可以帮助聚焦随后的软件动态测试。公司可以使用手动（人工）检查以及恰当的控制来确保一致性和独立性。源代码的评估应当扩展到不同模块和层（水平和垂直接口）之间的内部连接的验证，以及其是否符合设计规格。所使用的流程记录以及源代码评估的结果应当作为设计确认的一部分。

源代码的可追溯性分析是一种确认所有代码均连接到已确定规格和测试流程的重要工具。源代码可追溯性分析应当执行并记录以下内容的确认情况：

- 软件设计规格的每个部分均在代码中实现；
- 代码实现的模块和功能能够追溯到软件设计规格和风险分析中的内容；
- 模块和功能的测试可以追溯到软件设计规格和风险分析中的内容；以及
- 模块和功能的测试可以追溯到同一模块和功能的源代码。

典型任务—构建或代码

- 可追溯性分析
  - 从源代码到设计规格（反之亦然）
  - 从源代码的测试用例到设计规格
- 源代码和源代码文件的评估
- 源代码接口分析
- 测试流程和测试用例生成（模块、集成、系统和验收）

5.2.5 软件开发者进行的测试

软件测试需要在已知的某些情况下运行软件产品，使用特定的输入值并且记录到与预先设定的结果相同的输出值。这是一项耗时、困难并且无法尽善尽美的工作。因此，为了有效且高效地完成，需要进行早期计划。

测试计划和测试用例应当在软件开发过程尽可能早的阶段完成。其应当明确日程表、环境、资源（人力、工具等）、方法、方案（输入、流程、输出、预期结果）、记录和报告标准。在测试过程中付出的劳动成本与复杂度、关键性、可信度和/或安全因素相关（例如需要通过关于容错功能的加强测试来产生关键结果的功能或模块来挑战）。软件分类及软件测试工作的描述见于文献，例如：

- NIST 特种文献 500-23，《结构化测试：一种使用圈复杂度度量的测试方法》；
- NUREG/CR-6293，《高度集成化系统的确认和验证指南》；**以及**
- IEEE 计算机协会出版社，《软件可靠性工程手册》

软件测试计划应当明确开发的每个阶段要执行的具体任务，并包括对相应的完成标准所代表的工作规模等级的合理说明。

软件测试存在着局限性，必须在计划特定软件产品的测试时予以识别和考虑。除非最简单的程序，否则软件无法完全测试。通常来说无法使用所有可能的输入量对软件产品进行测试，也不可能测试程序执行过程中能够出现的所有数据处理路径。没有哪一种测试或测试方法能够确保完全测试具体软件产品。对所有程序的功能进行测试并不是说所有的程序都经过了测试。测试所有程序的代码并不意味着程序中存在所有必需的功能。所有程序功能和所有程序代码的测试并不意味着程序是 100％正确！没有发现错误的软件测试不应解释为意味着软件产品中不存在错误；这可能意味着测试不够深入。

软件测试用例最核心的部分是预期结果。这是对实际测试结果进行客观评估的重要细节。这项必要的测试信息来自于相应的预先明确的定义或规格。软件规格文档必须明确，应当做什么、何时做、如何做以及为什么做，并且具备工程级（例如，可测量或可以客观验证）的细节才要达到可以确认通过测试的水平。进行有效的软件测试最主要的工作是确定测试什么而不是测试的实施。

软件测试流程应当建立在能够促进软件产品有效检查的基础上。适用的软件测试原则包括：

- 预先确定预期的测试结果；
- 好的测试用例有将错误暴露出来的较高可能性；
- 成功的测试能够发现错误；
- 与代码编写独立；
- 雇用应用（用户）和软件（编程）专家；
- 测试者使用与代码编写者不同的工具；
- 仅检查常规情况是不够的；
- 测试文档可以重复使用，并且在后续的审查中对测试输出结果的通过/未通过状态进行独立确认。

一旦前期任务（如代码检查）成功完成，软件测试随即开始。软件测试从单位水平的测试开始，从系统水平的测试结束。可能有一个单独的集成水平测试。软件产品应该使用基于其内部结构和外部规格的测试用例进行测试。这些测试应当能够提供深入而严格的检查，检查软件产品是否符合其功能、性能、接口的定义与要求。

基于代码的测试也成为结构化测试或“白箱”测试。其确定的测试用例基于源代码、详细的设计规格及其他开发文档获得的内容。这些测试用例对程序作出的控制决定进行挑战；

程序数据结构包括配置表格。结构化测试能够找出程序运行中从来没有执行过的“死”代码。结构化测试主要通过单元（模块）水平的测试实现，但可以扩展到软件测试的其他水平。

结构化测试的程度能够通过使用特定度量来进行评估，这些度量旨在显示结构化测试过程中已经评估的软件结构百分比。这些度量通常被称为“覆盖率”，是测试选择标准完成率的指标。结构化覆盖率的值应当与软件风险等级相当。使用术语“覆盖”通常指的是 100%覆盖。例如，如果一项测试程序达到了“声明覆盖”，则意味着软件 100%的声明至少执行了一次。常见的结构化覆盖度量包括：

- **声明覆盖**—该项标准要求每项程序声明进行充分的测试，要求执行至少一次；但完成声明覆盖不足以提供软件产品行为的可信度。
- **决策（分支）覆盖**—该项标准要求每项待执行的程序决策或分支均进行充分的测试，每项可能的结果出现至少一次。对于大多数软件产品来说这是最低程度的覆盖，但仅完成决策覆盖对于高度集成的应用是不够的。
- **状态覆盖**—该标准需要在程序决策中为每个情况提供足够的测试用例，至少一次采取所有可能的结果。只有必须评估多个条件才能做出决定时，其才与分支覆盖不同。
- **多状态覆盖**—该标准需要足够的测试用例来执行程序决策中所有可能的条件组合。
- **循环覆盖**—该标准需要足够的测试用例，对所有程序循环进行充分测试，要求执行 0 次、1 次、2 次级多次迭代，包括初始化、典型线程和终止（边界）状态。
- **路径覆盖**—该标准对每项可行的路径、基本路径等需要足够的测试用例，要求从确定的程序段从开始到退出进行测试，至少执行一次。因为软件程序中可能的路径数量非常多，因此路径覆盖率一般是无法达到的。路径覆盖率通常基于待测试软件的风险或关键性来确定。
- **数据流覆盖**—该项标准要求对每种可能的数据流需要足够的测试用例，要求每种数据流至少执行一次。有多种可用的数据流测试策略。

基于定义或基于规格的测试也被称为功能测试或“黑箱”测试。其基于软件产品（无论其为单元模块还是完整的程序）的预期用途来确定测试用例。这些测试用例针对程序的预期用途或功能、内部和外部接口进行测试。功能测试可以应用于所有的软件测试水平，从单元到系统水平的测试。

下列各种类型的功能软件测试涉及了工作规模从低到高的测试方法：

- **常规方案**—使用常规输入进行测试是必要的。但仅使用预期、有效的输入值对软件进行检测并不能完全测试软件产品。使用这种方法，常规方案的测试不能提供软件产品可靠性的充足可信度。
- **输入强制**—选择测试输入以确保测试能够产生所选的（或所有的）软件输出。
- **稳健性**—软件测试应当能够证明软件产品在输入非预期的无效数据时仍然能够正确运行。确定满足要求的测试用例的方法包括等价类划分法、边界值分析法、特殊方案确定（错误猜测）。虽然这些技术很重要也很必要，但这些技术不能确保软件产品所有适合的测试项目均经过确认和测试。
- **输入的组合**—功能测试方法明确了上述所有强调的独立或单项测试输入。大多数软件产品在使用状态下会接受多个输入。全面的软件产品测试应当考虑软件单元或系统在使用中可能会遇到的输入组合。错误猜测可以扩展到输入组合的确认，但这是一种点对点的技术。因果分析图表是一种可用于测试用例、能够系统地识别软件产品输入组合的功能软件测试技术。

功能性和结构性软件测试用例的确定提供了测试特定的输入，而非任意的输入。这些技术的一个缺点是很难将结构性与功能测试的完成标准与软件产品的可信度进行联系。高级的软件测试方法，如统计测试，可以用来提供软件可信性的进一步的保证。统计测试使用从基于操作剖面（如软件产品的预期使用、危害性使用或恶意使用）确定的分布中随机生成的测试数据。可以生成大量的测试数据并用于覆盖特定领域或问题，从而有较高的可能性发现软件产品设计者或测试者预期之外的单个或多个罕见的运行情况。统计测试也提供了很高的结构覆盖率。其确实需要一个稳定的软件产品。因此，结构性和功能测试是进行软件产品的统计测试的前提。

软件测试的另一个方面是软件变更的测试。在软件开发过程中会有频繁的变更。这些变更的原因是 1）调试中发现了错误并改正，2）新的要求或要求变更（“要求渐变”），和 3）由于发现了更为有效的或效率更高的实现方法并修改了设计。一旦软件产品成为基线产品（批准），软件的任何变更都应该有其自己的“微生命周期”，包括测试。变更的软件产品的测试需要额外的工作规模。不仅要证明这种变更的实施是恰当的，还应该通过测试证明这些变更没有对软件产品的其它部分产生不利影响。可以使用回归分析和测试的方法来确保变更没有在软件产品的其它部分产生问题。回归分析是基于对相关文档的评估来确定变更影响的方法（例如软件要求规格、软件设计规格、源代码、测试计划、测试用例、测试脚本等），目的是确定要运行的必要的回归测试。回归测试会再次运行先前程序可以正确执行的测试用例并将当前的结果与先前结果进行比较，来检测软件变更是

否造成了非预期的效应。在使用集成方法来建立软件产品时应当使用回归分析和回归测试来确保新集成的模块不会影响先前已经集成的模块运行。

为了对软件产品进行深入和严格的检查，开发测试通常会分不同的等级来进行组织。例如，软件产品的测试可以在单元、集成和系统水平进行测试。

1） 单元（模块或组件）水平的测试关注子程序功能的早期检查，能够保证对系统层面无法观察到的功能进行测试。单元测试确保了高质量软件单元来进行最终软件产品的集成。

2） 集成水平的测试关注数据和控制通过程序内部和外部接口的转移。外部接口是与其他软件（包括操作系统软件）、系统硬件以及用户的接口，可以描述为通信线路。

3） 系统水平的测试可以证明所有具体的功能均满足，并且软件产品可以信赖。该测试确认了装配完成后的程序软件产品在特定操作平台上的的功能与性能的要求。系统水平的软件测试解决了功能方面的问题以及器械软件与预期用途相关的下列问题：

- 性能问题（例如响应时间、可靠度测量）；
- 压力状态应对，例如在最大载荷、连续使用下的运行情况；
- 内部和外部安全特征的操作；
- 复原程序的有效性，包括灾难复原；
- 可使用性；
- 与其他软件产品的相容性；
- 在确定的硬件配置下的工作情况；以及
- 软件记录的准确性。

应当使用控制措施（如可追溯性分析）来确保达到预期的覆盖度。

系统水平的测试也显示了软件产品在预期的操作环境下的工作情况。这种测试的地点取决于软件开发者搭建目标操作环境的能力。根据环境条件，可能会在（潜在的）客户使用地点进行模拟和/或测试。测试计划应当明确必要的控制，确保达在非软件开发者直接控制的环境下开展系统水平的测试时能够到预期的覆盖率，且准备做好恰当记录。同时，对于本身为医疗器械或医疗器械组成部分、在 FDA 许可前要用于人体的软件产品，涉及受试者的测试可能需要研究用器械豁免政策（IDE）或伦理委员会（IRB）批准。

测试流程、测试数据和测试结果的记录应当能够用于进行客观的通过/不通过决策。还需要适合测试后续的审查和客观决策，也要适合各种后续的回归测试。在测试中检测到的错误要记录日志、分类、审查并在软件发布前改正。开发生命周期中收集和分析的软件错误数据可能会用于确定软件产品是否适合商业销售的发布。测试报告应当与相应测试计划的要求一致。

医疗器械或其产品中发挥有效功能的软件产品通常十分复杂。常使用软件测试工具来确保软件产品测试过程的一致性、全面性和有效性，以及满足预期测试工作的要求。这些工具可能包括内置的支持软件来辅助进行单元（模块）测试、后续的集成测试（如驱动和桩模块）及商用软件测试工具。这些工具的质量水平应当不低于用其开发的软件产品。适当的文档记录能够提供相应的信息证明这些软件工具可以适用于其预期用途（见本指南的第 6 章）。

典型任务—软件开发者测试

- 测试计划
- 结构测试用例确认
- 功能测试用例确认
- 可追溯性分析—测试
  - 从单元（模块）测试到详细设计
  - 从集成测试到较高水平的设计
  - 从系统测试到软件要求
- 单元（模块）测试执行
- 集成测试执行
- 功能测试执行
- 系统测试执行
- 可接受性测试执行
- 测试结果评估
- 错误评估/解决方案
- 最终测试报告

5.2.6 用户现场测试

在用户现场进行测试是软件验证的核心部分。质量体系规范对安装和检查流程（包括适用的测试）以及证明软件安装恰当的检查和测试的记录做出了要求。（见 21 CFR §820.170。）同样，生产器械必须符合具体要求，并且自动化系统必须针对其预期用途进行验证。（分别见于 21 CFR §820.70（g）和 21 CFR §820.70（i））。

用户现场测试相关的术语容易混淆。在描述用户现场测试时使用了各种术语如 Beta 测试、地点验证、用户可接受性测试、安装确认以及安装测试等。在本指南中，术语“用户现场测试”综合了上述所有概念以及任何在开发者控制环境外进行的测试。该项测试应当在用户现场展开，并且使用实际作为安装系统配置一部分的硬件和软件进行。该项测试要在软件预期功能的环境中，通过真实的或模拟的软件使用来完成。

这部分的指导内容一般都很基本，适用于任何的用户现场测试。但在某些情况下（例如血站系统），在用户现场测试的计划过程中需要考虑具体地点验证问题。制定测试计划的人应当核查 FDA 中心对于相应产品管理的规定，来确定用户现场测试有没有额外的法规要求。

用户现场测试应当遵循预先前确定的书面计划、正式的测试总结和正式验收记录。应当记录所有测试流程、测试输入数据及测试结果。

需要提供证据证明硬件与软件按照预定的要求安装与配置。应当采取措施确保所有的系统组件在测试中均运行过，并且这些组件的版本是指定的版本。测试计划应在整个操作条件下明确测试，并应明确持续足够的时间，使系统遭遇广泛的情况和事件，以便检测在更正常的活动中不明显的任何潜在故障。

在早些时候由软件开发者在开发者地点进行的一些评估也应当在实际使用地点进行重复。这些测试可能包含涉及大量数据、较高负载或压力、安全性、故障检测（故障避免、故障检测、故障耐受和故障恢复）、错误信息以及安全要求实施相关的测试。开发者可能要向用户提供一些用于上述目的的测试数据集。

除了软件预期系统功能正常运行的评估，还需要对系统的用户是否能够理解和恰当进行界面操作进行评估。操作者应当能够执行预期的功能，并恰当及时地应对所有的警告、警示和错误信息。

在用户现场测试中，需要同时对正常的系统表现以及遇到的任何系统故障进行记录。在用户现场测试中针对检测到故障的系统修改应当遵循与其他软件变更相同的流程和控制。

软件的开发者可能或可能不参与用户现场测试。如果开发者参与了，则其要无缝执行用户现场测试与最后部分的设计水平系统测试。如果开发者没有参与，则更为重要的是用户中要有人能够理解细致的测试计划、预期测试结果的确定以及所有测试输出的记录的重要性。

典型任务—用户现场测试

- 可接受性测试执行
- 测试结果评估
- 错误评估/解决
- 最终测试报告

5.2.7 维护与软件变更

维护这一术语在应用于软件时与应用于硬件时含义不同。硬件和软件的操作性维护是不同的，因为其故障/错误机制不同。硬件维护通常包括预防性硬件维护工作、部件更换以及纠正性变更。软件维护包括纠正性、完善性和适应性维护，但不包括预防性维护工作或软件组件的更换。

用于纠正软件中的错误和故障的变更叫做纠正性维护。为了提高性能、可维护性及其它软件系统属性对软件进行变更是完善性维护。使软件系统能够在改变的环境中使用的软件变更称为适应性变更。

当软件系统发生改变时，如论在最初的开发过程中还是发布后的维护，均应当进行充足的回归分析和测试，来证明软件中没有发生变更的部分没有受到负面影响。还要进行用于评估实施的变更是否恰当的测试。

每种软件特定的必要的验证工作取决于变更的类型、受到影响的开发产品以及软件运行中产品受到的影响。对设计结构和不同模块、接口等之间进行详细和完整的记录能够减少变更时验证的工作规模。完全验证变更的工作规模水平也取决于原始软件验证的记录和归档情况。例如，在进行后续的回归测试时需要测试记录、测试用例以及先前的验证和确认测试结果。如果无法对这些信息进行归档用于后续使用的话，会极大地增加软件

变更后再验证的工作规模和支出。

除了作为标准软件开发流程一部分的软件验证和确认任务，还需要完成下列其他的维护任务：

- **软件验证计划修订—**对于先前已经经过验证的软件，应当对已有的软件验证计划进行修订来支持变更后的软件验证。如果没有已有的软件验证计划，则应当建立一个计划来支持变更后软件的验证。
- **异常评估—**软件公司通常会进行各种记录，如描述发现的软件异常情况的软件问题报告，以及解决异常情况采取的纠正性行为。但更多的时候错误会重复出现，因为开发者不会进一步确定问题的根本原因并采取必要的措施更改处理或流程来避免问题再次出现。应当对软件异常的严重性及对系统运行和安全性的影响进行评估，但软件异常应当被看做质量体系中流程缺陷的表现来对待。对异常情况的根本原因的分析能够明确特定的质量体系缺陷。当识别到某种趋势时（如类似的软件异常情况重复出现），需要实施并记录适当的纠正和预防性措施来避免类似质量问题再次出现。（见 21 CFR 820.100）。
- **问题识别与解决方案追踪—**软件维护过程中发现的所有问题都要进行记录。应当对每项问题的解决方案进行追踪，根据历史参照以及未来趋势的情况，确保其确实解决的问题。
- **对拟进行的变更进行评估—**所有拟进行的修订、优化或添加都要进行评估，确定每项变更对系统的影响。该项信息应当确定确认和/或验证工作需要迭代的程度。
- **任务迭代—**对于已经批准的软件变更，需要实施所有必要的确认和验证任务，确保预计的变更实施正确，所有的文档完整且最新，并且软件性能没有不可接受的改变出现。
- **文档更新—**应当对记录进行仔细审查，确定那些记录部分受到了变更的影响。所有受到影响的已批准文件（例如规格、测试流程、用户手册等）均应当根据配置管理流程进行更新。规格应当在任何维护和软件变更执行前更新。

第 6 章 自动化处理器械和质量体系软件验证

质量体系法规要求"当使用了计算机或自动化数据处理系统作为产品或质量体系的一部分，则【医疗器械】生产商应当根据已确定的规范针对其预期用途对计算机软件进行验证。"（见 21 CFR §820.70（i））。自 1978 年以来，这是 FDA 的医疗器械良好生产规范（GMP）法规的监管要求。

除了上述验证要求，作为医疗器械生产商产品流程或质量体系（或用于生成和存储 FDA 法规要求的记录）一部分的计算机系统也应遵循电子记录；电子签名法规。（见 21 CFR 第 11 部分。）该项法规对电子化记录的生产和存储提出了其他的安全性、数据集成和验证要求。这些来自第 11 部分的额外要求应当仔细考量并纳入到所有的自动化记录存储系统的系统要求和软件要求中。系统验证和软件验证应当证明满足了所有第 11 部分的要求。

计算机和自动化器械广泛应用于医疗器械设计、实验室测试和分析、产品检查和验收、生产和流程控制、环境控制、包装、标签、可追溯性、文档控制、投诉管理以及质量体系的许多方面。自动化车间系统的运行越来越多地涉及嵌入式系统的使用，这些应用包括：

- 可编程式逻辑控制器；
- 数字功能控制器；
- 统计过程控制；
- 监视控制和数据采集；
- 机器人技术；
- 人机接口；
- 输入/输出器械；以及
- 计算机操作系统。

在自动化医疗器械软件的设计、开发和测试过程中会频繁地使用软件工具。在质量体系实施过程中还会用到许多其它商业化的软件应用，如文字处理器、表格、数据库以及流程图软件。所有的这些应用均要满足软件验证的要求，但每种应用所使用的验证方法可以有很大差别。

无论软件产品或质量体系软件由医疗器械生产商自主开发、由承包商开发或购买现成软件，其均应当根据本指南其它部分所规定的基本原则进行开发。医疗器械生产商在确定软件如何达到验证标准时有一定的选择和灵活度，但软件验证应当作为决定如何以及由

谁来进行软件开发或从何处购买软件的主要考虑因素。软件开发者确定生命周期模型。一般来说下列内容支持软件验证：

- 软件开发生命周期每个阶段的输出确认；以及
- 开发完成的软件在医疗器械生产商预期使用的环境下正常运行的检查。

6.1 需要多少验证证据？

验证工作规模水平应当与自动化运行带来的风险相一致。除了其他的风险因素，如流程软件的复杂性以及器械生产商依赖于自动化流程进行安全和有效器械生产的程度，还需要确定验证工作所需要的测试内容和程度。自动化流程的需求文档和风险分析有助于确定证明软件在预期用途下有效所需要的证据范围。例如，如果器械生产商能够证明在产品发布前运行输出会根据规格进行充分确认，则自动化铣床可能只需要很少的测试。另一方面，下列情况可能需要大规模的测试：

- 工作间范围的电子记录和电子化签名系统；
- 用于灭菌循环的自动化控制器；或
- 用于检查生命维持/生命支持器械的电路板检查和可接受性分析的自动化测试器械。

质量系统可能会应用多种多样的商业化软件应用（例如用于质量体系统计的表格或统计软件包，用于趋势分析的画图软件包或者用于记录器械历史记录或投诉管理的商业化数据库）。这些软件所需要的验证证据等级取决于器械生产商记录的软件预期用途。例如，器械生产商选择不使用该软件供应商所提供的所有功能，则仅需要对需使用的功能、器械生产商依赖其作为生产或质量体系一部分的软件结果的功能进行验证。但高风险的应用不应当与未经验证的软件应用在相同的运行环境下运行，即使这些功能不会使用也不允许。当高风险应用和低风险应用在相同的运行环境里共同使用时可能需要考虑使用风险缓解技术，如存储区分或其他用于资源保护的方法。当软件升级或发生任何变更时，器械生产商应当考虑这些变更会如何影响软件“所用的部分”，并且必须重新确认和验证软件使用的部分。（见 21 CFR §820.70（i））。

6.2 确定的用户要求

软件验证一个非常重要的关键问题是明确了一下内容的用户要求规格：

- 软件或自动化器械的“预期用途”；以及
- 在生产质量合格的医疗器械时器械生产商依赖于软件或器械的程度。

器械生产商（用户）需要明确预期的运行环境，包括任何要求的硬件和软件配置、软件版本、实用程序等。用户还需要：

- 记录系统性能、质量、错误处理、开机、关机、安全性等的要求；
- 明确所有与安全相关的功能或特征，例如传感器、警告、联锁系统、逻辑处理步骤或命令序列；以及
- 明确可接受性能的客观标准。

验证过程必须根据文档形式的规范展开，验证结果必须进行记录。（见 21 CFR §820.70（i）。）测试用例应当进行记录，并运行系统，根据预先确定的标准测试其性能，尤其是最关键的参数。测试用例应当包括错误和警告状态、开机、关机、所有适用的用户功能和操作控制、潜在的操作错误、允许的最大和最小值以及适用于器械预期用途的压力条件。应当执行测试用例并记录结果，评估并确定结果是否支持软件已验证适用于其预期用途的结论。

器械生产商可能会用自己的员工进行验证，也可能依赖于第三方人员，如器械/软件供应商或顾问。无论什么情况，器械生产商均应对确保产品和质量体系软件的下列方面最终负责：

- 根据书面流程，在特定预期用途下软件通过验证；以及
- 在选择的应用场景中的运行情况符合预期。

器械生产商应当有下列文档：

- 确定用户需求；
- 使用的验证规范；
- 验收标准；
- 测试用例和结果；以及
- 验证总结

上述文档客观确认了软件的预期用途经过验证。

6.3 现成软件和自动化器械验证

大多数用于器械生产的自动化器械和系统均由第三方供应商提供，并购买现成软件（OTS）。器械生产商有责任确保 OTS 软件开发者适用的产品开发方法是恰当的，并且满足器械生产商预期使用 OTS 软件的要求。对于 OTS 软件和器械，器械生产商可能能够获得供应商的软件验证文档，也可能无法获得该文档。如果供应商能够提供关于其系统要求、软件要求、验证流程以及验证结果的信息，则医疗器械生产商能够使用这些信息作为验证文档的起点。供应商的生命周期文档，例如测试规范和结果、源代码、设计规格以及要求规格，能够确定软件经过验证。但很多时候这些文档无法从商业器械供应商处获得，或者供应商可能会拒绝提供其专有信息。

如果可能，并且取决于涉及的器械风险，器械生产商应该考虑对供应商的设计以及 OTS 软件开发中使用的方法进行审查，并且应当对 OTS 软件生成的开发和验证文档进行评估。该审查过程可以由器械生产商或具有资质的第三方来完成。审查应当证明供应商针对 OTS 软件的验证和确认流程和结果是恰当的，并且足够证明使用该软件生产的医疗器械满足安全性与有效性的要求。

一些供应商不习惯在监管环境下的操作，可能没有能够支持器械生产商验证要求的生命周期流程记录。其他的一些供应商可能不同意审查。如果无法从供应商获得必要的验证信息，则器械生产商需要进行足够的系统水平的“黑箱”测试，确定软件满足其“用户需求和预期用途”。对于很多应用来说，单独进行黑箱测试是不够的。根据器械产生的风险、OTS 软件在流程中的作用、对供应商进行审查的能力以及供应商提供的信息是否充足等信息，使用 OTS 软件或器械可能是适合的，也可能是不适合的，尤其是当存在合适的其它替代方案时。器械生产商应还当考虑一旦供应商停止支持后后续的维护和 OTS 软件支持的问题（如果存在的话）。

对于一些现成软件开发工具，例如软件编译器、连接程序、编辑器以及操作系统，由器械生产商进行全面的黑箱测试可能不太实际。不进行这些测试—验证工作的重要部分—可能无法验证这些软件工具。但可以通过其他的方式对运行情况进行可靠的推断。例如，第三方测试机构会经常对编译器进行认证，商业化软件产品可能会有“漏洞列表”，供应商提供的系统要求和其他操作信息可能与器械生产商预期使用情况进行比较，帮助集中“黑箱”测试工作。现成操作系统不需要作为单独的程序进行验证。但应用软件系统水平的验证测试应当包括使用所有操作系统服务，包括可能是用于应用程序预期用途的最大负载状态、文件操作、系统故障处理情况以及内存限制。

更详细的信息请参阅附录 A 中的生产和处理软件参考文件。

附录 A – 参考文件

食品药品监督管理局参考文件

医疗器械制造商设计控制指南，器械与放射健康中心，食品药品监督管理局，1997 年 3 月。

设计实现，医疗器械人力因素介绍，器械与放射健康中心，食品药品监督管理局，1997 年 3 月。

电子记录；电子签名最终规定，62 联邦公报 13430（1997 年 3 月 20 日）。

计算机化系统和软件开发术语表，实地调查部，区域行动办公室，法规事务办公室，食品药品监督管理局，1995 年 8 月。

医疗器械所含软件的上市前提交内容指南，器械评估办公室，器械与放射健康中心，食品药品监督管理局，1998 年 5 月。

医疗器械中现成软件使用的行业、FDA 工作人员和符合性指南，器械评估办公室，器械与放射健康中心，食品药品监督管理局，1999 年 9 月。

流程验证的一般原则指南，药品与生物制品中心，器械与放射健康中心，食品药品监督管理局，1987 年 5 月。

医疗器械；现行良好生产规范（CGMP）最终规定；质量体系法规，61 联邦公报 52602（1996 年 10 月 7 日）。

血站计算机软件上市前通告提交的审查员指南，生物制品评估和研究中心，食品药品监督管理局，1997 年 1 月。

学生手册 I，课程 INV545，计算机系统验证，人力资源发展部，法规事务办公室，食品药品监督管理局，1997 年。

技术报告，软件开发工作，实地调查部，区域行动办公室，法规事务办公室，食品药品监督管理局，1987 年 7 月。

其它政府参考文件

W. Richards Adrion, Martha A. Branstad, John C. Cherniavsky. *NBS Special Publication 500-75, Validation, Verification, and Testing of Computer Software,* Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S.Department of Commerce, February 1981.

Martha A. Branstad, John C Cherniavsky, W. Richards Adrion, *NBS Special Publication 500-56, Validation, Verification, and Testing for the Individual Programmer*, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, February 1980.

J.L. Bryant, N.P. Wilburn, *Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry*, NUREG/CR-4640, U.S. Nuclear Regulatory Commission, 1987.

H. Hecht, et.al., *Verification and Validation Guidelines for High Integrity Systems*. NUREG/CR-6293. Prepared for U.S. Nuclear Regulatory Commission, 1995.

H. Hecht, et.al., *Review Guidelines on Software Languages for Use in Nuclear Power Plant Safety Systems, Final Report*. NUREG/CR-6463. Prepared for U.S. Nuclear Regulatory Commission, 1996.

J.D. Lawrence, W.L. Persons, *Survey of Industry Methods for Producing Highly Reliable Software*, NUREG/CR-6278, U.S. Nuclear Regulatory Commission, 1994.

J.D. Lawrence, G.G. Preckshot, *Design Factors for Safety-Critical Software*, NUREG/CR-6294, U.S. Nuclear Regulatory Commission, 1994.

Patricia B. Powell, Editor. *NBS Special Publication 500-98, Planning for Software Validation, Verification, and Testing,* Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, November 1982.

Patricia B. Powell, Editor. *NBS Special Publication 500-93, Software Validation, Verification, and Testing Technique and Tool Reference Guide*, Center for Programming Science and Technology, Institute for Computer Sciences and Technology, National Bureau of Standards, U.S. Department of Commerce, September 1982.

Delores R. Wallace, Roger U. Fujii, *NIST Special Publication 500-165, Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards*, National Computer Systems Laboratory,

National Institute of Standards and Technology, U.S. Department of Commerce, September 1995.

Delores R. Wallace, Laura M. Ippolito, D. Richard Kuhn, *NIST Special Publication 500-204, High Integrity Software, Standards and Guidelines*, Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, September 1992.

Delores R. Wallace, et.al. *NIST Special Publication 500-234, Reference Information for the Software Verification and Validation Process*. Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, March 1996.

Delores R. Wallace, Editor. *NIST Special Publication 500-235, Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. Computer Systems Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, August 1996.

国际和国家共识标准

ANSI/ANS-10.4-1987，《核工业科研和工程用计算机程序验证和确认指南》，美国国家标准协会，1987年。

ANSI/ASQC标准D1160-1995，《正式设计评论》，美国质量控制协会，1995。

ANSI/UL 1998：1998，《可编程组件中的软件安全标准》，Underwriters Laboratories, Inc.,，1998年。

AS 3563.1-1991，软件质量管理体系，第1部分：要求。由澳大利亚标准协会（澳大利亚标准协会）出版，1 The Crescent，Homebush，NSW 2140。

AS 3563.2-1991，软件质量管理体系，第2部分：实施指南。由澳大利亚标准协会（澳大利亚标准协会）出版1 The Crescent，Homebush，NSW 2140。

IEC 60601-1-4：1996，医疗电气设备，第1部分：安全性的一般要求，4。并列标准：可编程电气医疗系统。国际电工委员会，1996年。

IEC 61506：1997，工业过程测量与控制-应用软件文件。国际电工委员会，1997年。

IEC 61508：1998，电气/电子/可编程电子安全相关系统的功能安全。国际电工委员会，1998年。

IEEE Std 1012-1986，软件验证和确认计划，电气及电子工程师学会，1986年。

IEEE标准收集，软件工程，电气及电子工程师学会，Inc.，1994.ISBN 1-55937-442-X。

ISO 8402：1994，质量管理和质量保证-词汇。国际标准化组织，1994年。

ISO 9000-3：1997质量管理和质量保证标准-第3部分：ISO 9001:1994计算机软件的开发、供应、安装和维护应用指南。国际标准化组织，1997年。

ISO 9001：1994质量体系-设计、开发、生产、安装和维修的质量保证模式。国际标准化组织，1994年。

ISO 13485：1996质量体系-医疗器械-ISO 9001应用的特殊要求，国际标准化组织，1996年。

ISO/IEC 12119：1994，信息技术-软件包-质量要求和测试，联合技术委员会ISO/IEC JTC 1，国际标准化组织和国际电工委员会，1994年。

ISO/IEC 12207：1995，信息技术-软件生命周期流程，联合技术委员会ISO/IEC JTC 1，小组委员会SC 7，国际标准化组织和国际电工委员会，1995年。

ISO/IEC 14598：1999，信息技术-软件产品评估，联合技术委员会ISO/IEC JTC 1，小组委员会SC 7，国际标准化组织和国际电工委员会，1999年。

ISO 14971-1：1998，医疗器械-风险管理-第1部分：风险分析的应用。国际标准化组织，1998年。

机载系统和设备认证中的软件注意事项。特别委员会167 RTCA。RTCA Inc.，Washington，D.C.电话：202-833-9339。文件编号RTCA/DO-178B，1992年12月。

生产过程软件参考

The Application of the Principles of GLP to Computerized Systems, Environmental Monograph #116, Organization for Economic Cooperation and Development (OECD), 1995.

George J. Grigonis, Jr., Edward J. Subak, Jr., and Michael Wyrick, "Validation Key Practices for Computer Systems Used in Regulated Operations," *Pharmaceutical Technology*, June 1997.

Guide to Inspection of Computerized Systems in Drug Processing, Reference Materials and

Training Aids for Investigators, Division of Drug Quality Compliance, Associate Director for Compliance, Office of Drugs, National Center for Drugs and Biologics, & Division of Field Investigations, Associate Director for Field Support, Executive Director of Regional Operations, Food and Drug Administration, February 1983.

Daniel P. Olivier, "Validating Process Software", *FDA Investigator Course: Medical Device Process Validation*, Food and Drug Administration.

GAMP Guide For Validation of Automated Systems in Pharmaceutical Manufacture, Version V3.0, Good Automated Manufacturing Practice (GAMP) Forum, March 1998:
Volume 1, Part 1: User Guide
Part 2: Supplier Guide
Volume 2: Best Practice for User and Suppliers.

Technical Report No. 18, Validation of Computer-Related Systems. PDA Committee on Validation of Computer-Related Systems. PDA Journal of Pharmaceutical Science and Technology, Volume 49, Number 1, January-February 1995 Supplement.

Validation Compliance Annual 1995, International Validation Forum, Inc.

一般软件质量参考

Boris Beizer, *Black Box Testing, Techniques for Functional Testing of Software and Systems*, John Wiley & Sons, 1995. ISBN 0-471-12094-4.

Boris Beizer, *Software System Testing and Quality Assurance*, International Thomson Computer Press, 1996. ISBN 1-85032-821-8.

Boris Beizer, *Software Testing Techniques*, Second Edition, Van Nostrand Reinhold, 1990. ISBN 0-442-20672-0.

Richard Bender, *Writing Testable Requirements, Version 1.0*, Bender & Associates, Inc., Larkspur, CA 94777, 1996.

Frederick P. Brooks, Jr., *The Mythical Man-Month, Essays on Software Engineering*, Addison-Wesley Longman, Anniversary Edition, 1995. ISBN 0-201-83595-9.

Silvana Castano, et.al., *Database Security*, ACM Press, Addison-Wesley Publishing Company, 1995. ISBN 0-201-59375-0.

Computerized Data Systems for Nonclinical Safety Assessment, Current Concepts and Quality Assurance, Drug Information Association, Maple Glen, PA, September 1988.

M. S. Deutsch, *Software Verification and Validation, Realistic Project Approaches*, Prentice Hall, 1982.

Robert H. Dunn and Richard S. Ullman, *TQM for Computer Software*, Second Edition, McGraw-Hill, Inc., 1994. ISBN 0-07-018314-7.

Elfriede Dustin, Jeff Rashka, and John Paul, *Automated Software Testing – Introduction, Management and Performance,* Addison Wesley Longman, Inc., 1999. ISBN 0-201-43287-0.

Robert G. Ebenau and Susan H. Strauss, *Software Inspection Process*, McGraw-Hill, 1994. ISBN 0-07-062166-7.

Richard E. Fairley, *Software Engineering Concepts*, McGraw-Hill Publishing Company, 1985. ISBN 0-07-019902-7.

Michael A. Friedman and Jeffrey M. Voas, *Software Assessment - Reliability, Safety, Testability*, Wiley-Interscience, John Wiley & Sons Inc., 1995. ISBN 0-471-01009-X.

Tom Gilb, Dorothy Graham, *Software Inspection*, Addison-Wesley Publishing Company, 1993. ISBN 0-201-63181-4.

Robert B. Grady, *Practical Software Metrics for Project Management and Process Improvement*, PTR Prentice-Hall Inc., 1992. ISBN 0-13-720384-5.

Les Hatton, *Safer C: Developing Software for High-integrity and Safety-critical Systems,* McGraw-Hill Book Company, 1994. ISBN 0-07-707640-0.

Janis V. Halvorsen, *A Software Requirements Specification Document Model for the Medical Device Industry*, Proceedings IEEE SOUTHEASTCON '93, Banking on Technology, April 4th -7th, 1993, Charlotte, North Carolina.

Debra S. Herrmann, *Software Safety and Reliability: Techniques, Approaches and Standards of Key Industrial Sectors,* IEEE Computer Society, 1999. ISBN 0-7695-0299-7.

Bill Hetzel, *The Complete Guide to Software Testing*, Second Edition, A Wiley-QED Publication, John Wiley & Sons, Inc., 1988. ISBN 0-471-56567-9.

Watts S. Humphrey, *A Discipline for Software Engineering*. Addison-Wesley Longman, 1995. ISBN 0-201-54610-8.

Watts S. Humphrey, *Managing the Software Process*, Addison-Wesley Publishing Company, 1989. ISBN 0-201-18095-2.

Capers Jones, *Software Quality, Analysis and Guidelines for Success*, International Thomson Computer Press, 1997. ISBN 1-85032-867-6.

J.M. Juran, Frank M. Gryna, *Quality Planning and Analysis*, Third Edition, , McGraw-Hill, 1993. ISBN 0-07-033183-9.

Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley Publishing Company, 1995. ISBN 0-201-63339-6.

Cem Kaner, Jack Falk, Hung Quoc Nguyen, *Testing Computer Software*, Second Edition, Vsn Nostrand Reinhold, 1993. ISBN 0-442-01361-2.

Craig Kaplan, Ralph Clark, Victor Tang, *Secrets of Software Quality, 40 Innovations from IBM*, McGraw-Hill, 1995. ISBN 0-07-911795-3.

Edward Kit, *Software Testing in the Real World*, Addison-Wesley Longman, 1995. ISBN 0-201-87756-2.

Alan Kusinitz, "Software Validation", *Current Issues in Medical Device Quality Systems*, Association for the Advancement of Medical Instrumentation, 1997. ISBN 1-57020-075-0.

Nancy G. Leveson, *Safeware, System Safety and Computers*, Addison-Wesley Publishing Company, 1995. ISBN 0-201-11972-2.

Michael R. Lyu, Editor, *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, McGraw-Hill, 1996. ISBN 0-07-039400-8.

Steven R. Mallory, *Software Development and Quality Assurance for the Healthcare Manufacturing Industries*, Interpharm Press,Inc., 1994. ISBN 0-935184-58-9.

Brian Marick, *The Craft of Software Testing*, Prentice Hall PTR, 1995. ISBN 0-13-177411-5.

Steve McConnell, *Rapid Development*, Microsoft Press, 1996. ISBN 1-55615-900-5.

Glenford J. Myers, *The Art of Software Testing*, John Wiley & Sons, 1979. ISBN 0-471-04328-1.

Peter G. Neumann, *Computer Related Risks*, ACM Press/Addison-Wesley Publishing Co., 1995. ISBN 0-201-55805-X.

Daniel Olivier, *Conducting Software Audits, Auditing Software for Conformance to FDA Requirements*, Computer Application Specialists, San Diego, CA, 1994.

William Perry, *Effective Methods for Software Testing*, John Wiley & Sons, Inc. 1995. ISBN 0-471-06097-6.

William E. Perry, Randall W. Rice, *Surviving the Top Ten Challenges of Software Testing*, Dorset

House Publishing, 1997. ISBN 0-932633-38-2.

Roger S. Pressman, *Software Engineering, A Practitioner's Approach*, Third Edition, McGraw-Hill Inc., 1992. ISBN 0-07-050814-3.

Roger S. Pressman, *A Manager's Guide to Software Engineering*, McGraw-Hill Inc., 1993 ISBN 0-07-050820-8.

A. P. Sage, J. D. Palmer, *Software Systems Engineering*, John Wiley & Sons, 1990.

Joc Sanders, Eugene Curran, *Software Quality*, Addison-Wesley Publishing Co., 1994. ISBN 0-201-63198-9.

Ken Shumate, Marilyn Keller, *Software Specification and Design, A Disciplined Approach for Real-Time Systems*, John Wiley & Sons, 1992. ISBN 0-471-53296-7.

Dennis D. Smith, *Designing Maintainable Software*, Springer-Verlag, 1999. ISBN 0-387-98783-5.

Ian Sommerville, *Software Engineering*, Third Edition, Addison Wesley Publishing Co., 1989. ISBN 0-201-17568-1.

Karl E. Wiegers, *Creating a Software Engineering Culture*, Dorset House Publishing, 1996. ISBN 0-932633-33-1.

Karl E. Wiegers, *Software Inspection, Improving Quality with Software Inspections*, Software Development, April 1995, pages 55-64.

Karl E. Wiegers, *Software Requirements*, Microsoft Press, 1999. ISBN 0-7356-0631-5.

附录 B –开发团队

器械与放射健康中心

合规办公室	Stewart Crumpler
器械评估办公室	James Cheng, Donna-Bea Tillman
健康与行业项目办公室	Bryan Benesch, Dick Sawyer
科学与技术办公室	John Murray
监测与生物统计办公室	Howard Press

药品评估和研究中心

医疗政策办公室	Charles Snipes

生物制品评估和研究中心

合规与生物制品质量办公室	Alice Godziemski

法规事务办公室

区域行动办公室	David Bergeson, Joan Loreng