第 3 章　软件质量与质量保证

【导读案例】 在软件测试计划中确定测试需求

写测试需求主要为了什么呢？项目中基本都有很细致的功能规格说明，还有其他一些相关的概念设计文档，我们总是会看到这些文档的最新版本。然而，项目多为迭代方式进行，分很多版本提交，如 1.0.1、1.0.2 等，并不是每个版本都要测试全部功能，往往只是测试其中的一部分。有的版本主要测业务流程，有的主要测性能。所谓测试需求，就是说明这个版本需要测试哪些东西。

测试需求按照功能性、可靠性、易用性、性能、可维护性、可移植性来分类，同时也要按照优先级来分类，有的是必须测试通过的，有的则可以协商。

除说明我们需要测试的内容以外，测试需求还有一个重要的作用：辅助说明测试接受标准。例如，某个版本的功能测试需求有 100 个功能点，其中 30 个必须实现，其他 70 个中实现 60 个即可。假如每个功能点 1 分，那么，功能测试的接受标准就是：总分 90 分以上并且 30 个重要功能点必须测试通过。假如没有达到这个接受标准（只有 85 分），我们就可以负责任地说：测试不通过，不能发布。如果要发布，则需要变更项目计划和测试计划。

测试需求最好能细致到功能点的粒度，这样对项目量化管理非常有好处，而且这应该在项目版本计划中进行说明。如果项目计划中没有说得很详细，那测试计划就要写得详细一些。

下面看一个例子（见表 3-1），这是某项目测试报告的一部分，其中列出了功能测试需求的执行情况。

表 3-1　项目测试报告（部分）

任　　务	说　　明	是否通过
添加专项工作	操作角色仅为系统管理员，正确新建专项工作	通过
编辑专项工作	操作角色仅为系统管理员，正确保存编辑后的专项工作	通过
删除专项工作	操作角色仅为系统管理员，正确删除所选的专项工作	通过
导出某专项工作的警情列表	按照同样的数据字典格式导出，保存为.xls 文件	通过

可以看出，这里的测试需求列得比较细，而且从用户角度进行说明。在实际项目中，需要写得多细，可以根据项目情况决定，只是不要忘记编制测试计划的主要目的。建议尝试把测试需求写细一些，体会一下量化管理的感觉。以后的测试例会上，可以把测试需求拿出来评审，比较一下不同项目的不同策略。

3.1 质量是免费的

1979年，菲利普·克罗斯比在《质量是免费的：使质量确定的艺术》一书中写道，制造高质量的产品比制造低质量产品实际上不需要额外开销（其实开销更小）。这似乎是不可能的，但研究表明，这确实是真的。软件缺陷发现得越晚，其处理费用就越高，并且按指数级激增。

菲利普·克罗斯比、约瑟夫·朱兰、W·爱德华兹·戴明都被称为“质量之父”，他们撰写了大量书籍，其实践结论也在世界范围内广泛使用。虽然他们的著作不是专门针对软件的，但是他们提出的概念常常适合于所有领域。

考虑把质量的费用分为两类：一致性费用和非一致性费用。一致性费用是指与一次性计划和执行测试相关的全部费用，用于保证软件按照预期方式运行。如果发现了软件缺陷，必须花时间分离、报告和回归测试以保证其得以修复，那么非一致性费用就会上涨。因为这些软件缺陷在发布之前被发现，所以这些费用归属于内部失败。

如果软件缺陷被遗漏并落到客户手里，结果就是代价昂贵的产品支持，可能还需要修复、重新测试和发布软件。在更糟糕的情况下，产品可能要召回或者卷入官司。这些外部失败的费用属于非一致性费用。

克罗斯比在他的书中介绍，由内部失败引起的一致性费用加上非一致性费用要小于由外部失败引起的非一致性费用。尽早提出软件缺陷，或者在理想情况下一开始就没有软件缺陷，那产品的开销就会比可能的情况要小。质量是免费的，这是普遍常识。

遗憾的是，一部分软件行业在项目开始时想法很好，然后随着问题的增多和时间进度的越来越近，规则和理由都被抛之脑后。现在的公司都认识到自己产品质量的费用太高。客户开始提出质量方面的要求，竞争对手也开始制作质量更好的软件——克罗斯比在数十年前为制造行业所写的话，今天在软件中得到实现。

3.2 软件质量的困境

一位编程语言专家伯特坦德·迈耶曾经这样论述所谓的软件质量困境：如果生产了一个存在严重质量问题的软件系统，你将受到损失，因为没有人想去购买。另外，如果你花费无限的时间、极大的工作量和高额的资金来开发一个绝对完美的软件，那么完成该软件将花费很长的时间，生产成本极其高昂，以至于破产。要么错过了市场机会，要么几乎耗尽所有资源。所以企业界会努力达到奇妙的中间状态：一方面，产品要足够好，不会立即被抛弃（比如在评估期）；另一方面，产品又不是那么完美，不需花费太长时间和太多成本。

软件工程师应该努力生产高质量的系统，但迈耶所讨论的情况也是现实的，甚至对于最好的软件工程组织也是如此。

3.2.1 “足够好”的软件

坦率地说，接受迈耶的观点，生产“足够好”软件是可接受的，事实上大的软件公司几乎都在这么做。这些大厂生产带有已知缺陷的软件，并发布给大量的最终用户。他们认识到，

1.0 版提供的一些功能和特性达不到最高质量，计划在 2.0 版改进。他们这样做，知道有些客户会抱怨，但他们认为上市时间胜过更好的质量，只要交付的产品“足够好”。

“足够好”的软件提供用户期望的高质量功能和特性，但同时也提供了其他更多的包含已知错误的难解或特殊的功能和特性。软件供应商希望最终用户会忽视错误，因为他们对其他的应用功能是如此满意。

诚然，“足够好”可能在某些应用领域和几个主要的软件公司起作用。毕竟，如果一家公司有庞大的营销预算，并能够说服足够多的人购买 1.0 版本，那么该公司已经成功地锁定了这些用户。可以认为，公司将在以后的版本提高产品质量。通过提供足够好的 1.0 版，公司垄断了市场。

但对于一些小公司来说，就要警惕这一观念，当你交付一个足够好（有缺陷的）的产品时，是冒着永久损害公司声誉的风险，你可能再也没有机会提供 2.0 版本了，因为不良言论可能会导致销售暴跌和公司关门。

在某个应用领域（如实时嵌入式软件）或者建造的是与硬件集成的应用软件（如汽车软件、电信软件），如果因为疏忽而交付了带有已知错误的软件，有可能使公司处于代价昂贵的诉讼之中。在某些情况下，甚至可以是刑事犯罪——没有人想要“足够好”的飞机航空电子系统软件！因此，在将“足够好”作为解决软件质量问题的捷径时，需要谨慎行事。

3.2.2 质量成本

质量成本包括追求质量过程中或在履行质量活动中引起的费用以及质量不佳引起的下游费用等所有费用。为了解这些费用，一个组织必须收集度量数据，为目前的质量成本提供一个基准，找到降低这些成本的机会，并提供一个规范化的比对依据。

质量成本可分为预防成本、评估成本和失效成本。

（1）预防成本：包括以下几种成本。

- 计划和协调所有质量控制和质量保证所需管理活动的成本。
- 为开发完整的需求模型和设计模型所增加的技术活动的成本。
- 测试计划的成本。
- 与这些活动有关的所有培训成本。

（2）评估成本：包括为深入了解产品“第一次通过”每个过程的条件而进行的活动。评估成本的例子包括以下几种。

- 对软件工程工作产品进行技术审查的成本。
- 数据收集和度量估算的成本。
- 测试和调试的成本。

（3）失效成本：是那些如果在将产品发给客户之前或之后没有错误就不会存在的费用。失效成本可分为内部失效成本和外部失效成本。内部失效成本发生在发货之前发现错误时，包括以下几种。

- 为纠正错误进行返工（修复）所需的成本。
- 返工时无意中产生副作用，必须对副作用加以缓解而发生的成本。
- 组织为评估失效的模型而收集质量数据，由此发生的相关成本。

外部失效成本是在产品已经发送给客户之后发现了缺陷时的相关成本。外部成本的例

子是解决投诉，产品退货和更换，帮助作业支持，与保修工作相关的人力成本。不良的声誉和由此产生的业务损失是另一个外部失效成本，这是很难量化但非常现实的。生产了低质量的软件产品时，不好的事情就要发生。

3.2.3 质量和安全

随着基于 Web 的系统和应用重要性的增加，应用系统的安全性已变得日益重要。简而言之，低质量的软件比较容易被攻击，会间接地增加安全风险，随之而来的是费用和问题。

安全专家加里·麦格劳这样评论：软件安全完全与质量有关系。必须一开始就在设计、构造、测试、编码阶段以及在整个软件生命周期（过程）中考虑安全性、可靠性、可得性、可信性。即使是已认识到软件安全问题的人们，通常也主要关注生命周期的晚些阶段，其实越早发现软件问题越好。有两种类型的软件问题，一种是隐藏的错误，这是实现的问题。另一种是软件缺陷，这是设计中的构造问题。人们对错误关注太多，却对缺陷关注不够。

要构造安全的系统，就必须注重质量，并在设计时就开始关注。通过消除架构缺陷（从而提高软件质量），将会使攻击软件变得愈加困难。

3.2.4 管理活动的影响

软件质量受管理决定的影响往往和受技术决定的影响是一样的，即使最好的软件工程实践也能被糟糕的商业决策和有问题的项目管理活动破坏。

（1）估算决策。在确定交付日期和制定总预算之前，给软件团队提供项目估算数据是很少见的。反而是团队进行“健康检查”，以确保交付日期和里程碑是合理的。在许多情况下，存在巨大的上市时间压力，迫使软件团队接受不现实的交付日期。结果抄了近路，可以获得更高质量软件的活动被忽略，使产品质量受到损害。如果交付日期不合理，那么坚持立场就是重要的。

（2）进度安排决策。在建立软件项目时间表时，会按照依赖性安排任务的先后顺序。例如，由于 A 构件依赖于 B、C 和 D 构件中的处理，直到 B、C 和 D 构件完全测试后，才能安排 A 构件进行测试。但是，如果时间很紧，为了做进一步的关键测试，A 必须是可用的。在这种情况下，可能会决定在没有其附属构件（这些附属构件的运行要稍落后于时间表）的情况下测试 A，这样，对于交付前必须完成的其他测试，就可以使用 A 了。毕竟，期限正在逼近。由此，A 可能存在隐藏的缺陷，只有到晚些时候才被发现，最终影响产品质量。

（3）面向风险的决策。风险管理是成功软件项目的关键特性之一。我们需要知道哪儿可能会出问题，并建立一项如果确实出问题时的应急计划。许多软件团队喜欢盲目乐观，在什么都不会出问题的假设下建立开发计划。更糟的是，他们没有办法处理真的出了差错的事情。结果，当风险变成现实时一片混乱，并且随着疯狂程度的上升，质量水平必然下降。

3.3 质量与软件质量

随着软件应用逐渐融入社会日常生活的各方面，人们越来越关注软件的质量，但是，却很难对软件质量给出一个全面的描述。多年来提出的各种各样的软件质量度量和因素，都试图定义一组属性，以推动实现较高的软件质量，这些质量因素特性包括可靠性、易用性、维

护性、功能性和可移植性等。

3.3.1 关于质量的观点

从本质上说，每个人都希望建立高质量的系统，但生产“完美”软件所需的时间和工作量在市场主导下是根本无法达到的。这个问题就转换为是否应该生产“足够好”的软件。此外，不管选择什么方法，从预防、评估和失效等方面来考虑，质量都是有成本的。

事实上，人们可以用不同的方式来看待质量。哈佛商学院的大卫·加文指出：“质量是一个复杂多面的概念”，可以从5个不同的观点来描述。

(1) 玄妙观点认为质量是马上就能识别的东西，却不能清楚地定义。

(2) 用户观点是从最终用户的具体目标来说的。如果产品达到这些目标，就显示出质量。

(3) 制造商观点是从产品的原始规格说明的角度来定义质量，如果产品符合规格说明，就显示出质量。

(4) 产品观点认为质量是产品的固有属性(如功能和特性)。

(5) 基于价值的观点根据客户愿意为产品支付多少费用来评测质量。

实际上，质量涵盖了所有这些观点。设计质量是指设计师赋予产品的特性。原料等级、公差和性能等规格说明决定了设计质量。如果产品是按照规格说明书制造的，那么使用了较高等级的原料，规定了更严格的公差和更高级别的性能，产品的设计质量就能提高。在软件开发中，设计质量包括设计满足需求模型规定的功能和特性的程度。符合质量关注的是实现遵从设计的程度以及所得到的系统满足需求和性能目标的程度。罗伯特·格拉斯认为设计质量和符合质量之间比较“直观的”关系符合下面的公式。

$$\text{用户满意度} = \text{合格的产品} + \text{好的质量} + \text{按预算和进度安排交付}$$

格拉斯认为质量是重要的。但是，如果用户不满意，其他任何事情就都不重要了。德马科同意这个观点，他认为：“产品的质量是一个函数，该函数确定了它在多大程度上使这个世界变得更好。”这个质量观点的意思就是：如果一个软件产品能给最终用户带来实质性的益处，他们可能会心甘情愿地忍受偶尔的可靠性或性能问题。

3.3.2 软件质量的定义

高质量的软件是一个重要目标。一般地，软件质量可以定义为：在一定程度上应用有效的软件过程创造有用的产品，为生产者和使用者提供明显的价值。

该定义强调了以下3个重要方面：

(1) 有效的软件过程为生产高质量的软件产品奠定了基础。软件工程实践允许开发人员分析问题、设计可靠的解决方案——二者皆为生产高质量软件的关键所在。最后，诸如变更管理和技术评审等普适性活动与其他部分的软件工程活动密切相关。

(2) 有用的产品是指交付最终用户要求的内容、功能和特征，但最重要的是，以可靠、无误的方式交付这些东西。有用的产品总是满足利益相关者明确提出的那些需求，同时也要满足一些高质量软件应有的隐性需求(例如易用性)。

(3) 通过为软件产品的生产者和使用者增值，高质量软件为软件组织和最终用户群体带来了收益。软件组织获益是因为高质量的软件在维护、改错及客户支持方面的工作量都

降低了，从而使软件工程师减少返工，将更多的时间花费在开发新的应用系统上，软件组织因此而获得增值。用户群体也得到增值，因为应用系统提供有用的能力，在某种程度上加快了业务流程。最后的结果是：① 软件产品的收入增加；② 当应用系统支持业务流程时，收益更好；③ 提高了信息可获得性，这对商业来讲是至关重要的。

3.3.3 加文的质量维度

戴维·加文建议采取多维的观点考虑质量，包括从符合性评估到抽象的（美学）观点。尽管加文的8个质量维度不是专门为软件制定的，但考虑软件质量时依然可以使用？

（1）性能质量。软件是否交付了所有的内容、功能和特性？这些内容、功能和特性在某种程度上是需求模型所规定的一部分，可以为最终用户提供价值。

（2）特性质量。软件是否提供了使最终用户满意的特性？

（3）可靠性。软件是否准确地提供了所有的特性和能力，当需要（使用该软件）时，它是否是可用的？

（4）符合性。软件是否遵从本地的和外部的与应用领域相关的软件标准，是否遵循了事实存在的设计惯例和编码惯例？例如，对于菜单选择和数据输入等用户界面的设计是否符合已接受的设计规则？

（5）耐久性。是否能够对软件进行维护（变更）或改正（改错），而不会粗心大意地产生意料不到的副作用？随着时间的推移，变更会使错误率或可靠性变得更糟吗？

（6）适用性。软件能在可接受的短时期内完成维护（变更）和改正（改错）吗？技术支持人员能得到所需的所有信息以进行变更和修正缺陷吗？

（7）审美。毫无疑问，大多数人都同意美的东西具有某种优雅、特有的流畅和醒目的外在，这些都很难量化，但显然是不可缺少的。

（8）感知。在某些情况下，一些偏见将影响人们对质量的感知。例如，有人给你介绍了一款软件产品，该软件产品是由过去曾经生产过低质产品的厂家生产的，你的自我保护意识将会增加，你对于当前软件产品质量的感知力可能受到负面影响。类似地，如果厂家有极好的声誉，你将能感觉到好的质量，甚至在质量实际并不存在的时候。

加文的质量维度提供了对软件质量的“软”评判。这些维度的多数（不是所有）只能主观地考虑。正因如此，也需要一套“硬”的质量因素，这些因素可以宽泛地分成两组：① 可直接测量的因素（如测试时发现的缺陷数）；② 只能间接测量的因素（如可用性或可维护性）。在任何情况，必须进行测量，应把软件和一些基准数据进行比较来确定质量。

3.3.4 麦卡尔的质量因素

麦卡尔、理查兹和沃尔特斯提出了影响软件质量因素的一种有用的分类。这些软件质量因素侧重于软件产品的3个重要方面，即操作特性、承受变更的能力以及对新环境的适应能力（见图3-1）。

针对图3-1中所提到的因素，麦卡尔及他的同事描述如下。

（1）正确性。程序满足其需求规格说明和完成用户任务目标的程度。

（2）可靠性。期望程序以所要求的精度完成其预期功能的程度。需要注意的是，还有更完整的可靠性定义。

(3) 效率。程序完成其功能所需的计算资源和代码的数量。

(4) 完整性。对未授权的人员访问软件或数据的可控程度。

(5) 易用性。对程序进行学习、操作、准备输入和解释输出所需要的工作量。

(6) 维护性。查出和修复程序中的一个错误所需要的工作量。

(7) 灵活性。修改一个运行的程序所需的工作量。

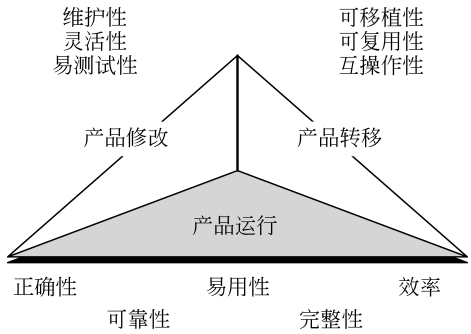

图 3-1 麦卡尔的软件质量因素

(8) 易测试性。测试程序以确保它能完成预期功能所需要的工作量。

(9) 可移植性。将程序从一个硬件和(或)软件系统移植到另一个环境所需要的工作量。

(10) 可复用性。程序(或程序的一部分)可以在另一个应用系统中使用的程度。这与程序所执行功能的包装和范围有关。

(11) 互操作性。将一个系统连接到另一系统所需要的工作量。

对这些质量因素的度量仅能间接地测量。不过,使用这些因素评估应用系统的质量可以真实地反映软件的质量。

3.3.5 WebApp 设计质量

早期的 WWW Web 站点仅包含链接在一起的一些超文本文件,这些文件使用文本和有限的图形来表示信息。随着时间的推移,一些开发工具(如 XML、Java)扩展了 HTML 的能力,使得 Web 工程师在向客户提供信息的同时也能提供计算能力,这就有了基于 Web 的系统和应用(WebApp)。WebApp 如今已经发展成为成熟的计算工具,不仅可以为最终用户提供独立的功能,还可以与公司数据库和业务应用集成在一起。

WebApp 是独特的一种软件类型,鲍威尔提出了基于 Web 的系统和应用"涉及印刷出版和软件开发之间、市场和计算之间、内部通信和外部关系之间以及艺术和技术间的混合"。绝大多数 WebApp 具备下列属性。

(1) 网络密集性:WebApp 驻留在网络上,服务于不同客户群体的需求。网络提供开放的访问和通信(如因特网)或者受限的访问和通信(如企业内联网)。

(2) 并发性:大量用户可能同时访问 WebApp。很多情况下最终用户的使用模式存在很大差异。

(3) 无法预知的负载量:WebApp 的用户数量每天都可能有数量级的变化。周一显示有 100 个用户使用系统,周四就可能会有 10 000 个用户。

(4) 性能:如果一位 WebApp 用户必须等待很长时间(访问、服务器端处理、客户端格式化显示),该用户就可能转向其他地方。

(5) 可用性:尽管期望百分之百的可用性是不切实际的,但是对于热门的 WebApp,用户通常要求能够全天候访问。澳大利亚或亚洲的用户可能在北美传统的应用软件脱机维护时间要求访问。

（6）数据驱动：许多 WebApp 的主要功能是使用超媒体向最终用户提供文本、图片、音频及视频内容。除此之外，WebApp 还常被用作访问那些存储在 Web 应用环境之外的数据库中的信息（如电子商务或金融应用）。

（7）内容敏感性：内容的质量和艺术性仍然在很大程度上决定了 WebApp 的质量。

（8）持续演化：传统的应用软件是随一系列规划好的时间间隔发布而演化的，而 Web 应用则持续地演化。对某些 WebApp 而言（特别是其内容），按分钟发布更新，或者对每个请求动态更新页面内容，这些都是司空见惯的事情。

（9）即时性：尽管即时性（也就是将软件尽快推向市场的迫切需要）是很多应用领域的特点，然而将 WebApp 投入市场可能只是几天或几周的事。

（10）安全性：由于 WebApp 是通过网络访问来使用的，因此要限制访问的最终用户的数量，即使可能也非常困难。为了保护敏感的内容，并提供保密的数据传输模式，在支持 WebApp 的整个基础设施上和应用本身内部都必须实施较强的安全措施。

（11）美观性：不可否认，WebApp 的用户界面外观很有吸引力。是否能将产品或思想成功地推向市场，界面美观和技术设计同样重要。

WebApp 几乎具备了上述的所有属性。软件质量的所有技术特征以及通用质量属性也都适用于 WebApp。其中一些最相关的通用特性——可用性、功能性、可靠性、效率及可维护性，为评估基于 Web 的系统的质量提供了有用基础。

奥尔西纳和他的同事设计了一个“质量需求树”，定义了一组可产生高质量 WebApp 的技术属性，包括可用性、功能性、可靠性、效率和可维护性（见图 3-2）。此后，奥弗特又对图 3-2 描述的质量属性进行了扩展。

图 3-2　质量需求树

（1）安全性：WebApp 已经和重要的公司及政府数据库高度集成。电子商务应用系统提取敏感的客户信息，然后将这些信息存储起来。因此，WebApp 的安全性变得极为重要。安全性的关键度量标准是 WebApp 和服务器环境拒绝非授权访问和（或）阻挡恶意攻击的能力。

（2）可用性：从技术的角度说，可用性是对 WebApp 的可用时间占总时间的百分比的

一种度量。一般最终用户期望 WebApp 7×24 小时都是可用的。但“正常运行时间”并不是可用性的唯一指标。奥弗特认为，使用仅限于在一种浏览器或平台上可用的特性，会使 WebApp 在那些具有不同浏览器或平台的配置中变得不可用，用户会毫无例外地转向其他地方。

(3) 可伸缩性：WebApp 及其服务环境和为其提供接口的系统能否承受用户访问数量上的巨大波动？响应速度是否会因此而剧减(或者完全停止)？开发成功的 WebApp 是远远不够的，开发能够成功调节负载(相当多的最终用户)的 WebApp 同样重要，而且会变得越来越重要。

(4) 投放市场时间：虽然这并不是真正的技术方面的质量属性，仅仅是从商业角度考虑的一种质量度量。但是，市场上的第一个 WebApp 往往能够吸引非常多的最终用户。

在 WWW 上查找信息的人们可以获得数以亿计的网页，要从这么多的信息源中选择所需要的信息，用户如何评价 WebApp 所展示内容的质量(如准确性、精确性、完整性、适时性)呢？蒂尔曼提出了评价内容质量的一组有用标准，具体如下。

(1) 能否很容易地判断内容的范围和深度，确保满足用户的要求？

(2) 是否容易识别内容作者的背景和权威性？

(3) 能否决定内容的通用性？最后的更新时间及更新内容是什么？

(4) 内容和位置是否稳定(即它是否一直保存在引用的 URL 处)？

这些也只是设计 WebApp 时应该考虑的问题中的一小部分。

3.4 软件质量保证


微课 3-1

良好的软件质量是良好的项目管理和扎实的软件工程实践的结果。帮助软件团队实现高质量软件的管理活动包括软件工程方法、项目管理技术，以及质量控制活动。

(1) 软件工程方法。建立高质量的软件必须理解所要解决的问题，还需要能够创造一个符合问题的设计，该设计同时还要有一些性质，以实现符合质量维度和因素的软件。

软件工程的一些概念和方法，可能产生对问题合理完整的理解和综合性设计，从而为构造活动建立了坚实的基础。如果应用这些概念并采取适当的分析和设计方法，那么创造高质量软件的可能性将大大提高。

(2) 项目管理技术。考虑下面这些措施：① 项目经理使用估算以确认交付日期是可以达到的；② 进度依赖关系是清楚的，团队能够抵抗走捷径的诱惑；③ 进行了风险规划，这样出了问题就不会引起混乱，软件质量将受到积极的影响。

(3) 质量控制活动。包括一套软件工程活动，以帮助确保每个工作产品符合其质量目标。评审模型以确保它们是完整的和一致的。检查代码，以便在测试开始前发现和纠正错误。应用一系列的测试步骤以发现处理逻辑、数据处理以及接口通信中的错误。当这些工作成果中的任何一个不符合质量目标时，软件团队能够借助测量和反馈的结合使用来调整软件过程。

3.4.1 关于质量保证

对于任何产品的生产企业来说，质量控制和质量保证都是必不可少的活动。随着时间

推移，大量生产技术逐渐普及，质量控制由生产者之外的其他人承担。

第一个正式的质量保证和质量控制方案于1916年由贝尔实验室提出，此后迅速风靡整个制造行业，出现了更多正式的质量控制方法，这些方法都将测量和持续的过程改进作为质量管理的关键成分。如今，每个公司都有保证其产品质量的机制。事实上，公司重视质量的明确声明已经成为过去几十年中市场营销的策略。

质量保证体系可以定义为：用于实现质量管理的组织结构、责任、规程、过程和资源。创建质量保证体系的目的是帮助组织以符合规格说明的方式，保证组织的产品和服务满足客户的期望。这些体系覆盖了产品整个生命周期的多种活动，包括计划、控制、测量、测试和报告，并在产品的开发和制造过程中改进质量等级。

相关的概念还有全面质量管理（TQM）或者全面质量控制（TQC），该方法的基本思想是：用集中的质量保证团队来负责质量是不实际的，因为从事代码工作的人员并不负责质量，他们不会设法实现质量保证的目的。要想制造高质量的产品，需要创立从管理开始自上而下的质量文化，使全体人员共同承担质量责任。

尽管TQM/TQC与现有的质量保证团队的工作有很大的关联，但是并不排除对软件测试的需要。完全相反，软件测试的作用在此环境中被更清晰地定义。无论过程中付出多么大的努力，软件终究是由人建立的，是人就会犯错误。所以仍然需要一个团队来专心寻找软件缺陷。这样，软件质量保证人员的主要职责就是检查和评价当前软件开发的过程，找出改进过程的方法，以达到防止软件缺陷出现的目标。

软件质量保证（Software Quality Assurance，SQA）是在软件过程中的每一步都进行的“普适性活动”。SQA包括：对方法和工具有效应用的规程，对诸如技术评审和软件测试等质量控制活动的监督，变更管理规程，保证符合标准的规程，以及测量和报告机制。

为了正确地进行SQA，必须收集、评估和发布有关软件工程过程的数据。基于统计的SQA有助于提高产品和软件过程本身的质量。软件可靠性模型将测量加以扩展，能够由所收集的缺陷数据推导出相应的失效率和进行可靠性预测。

总之，应该注意“SQA就是将质量保证的管理规则和设计规范映射到适用的软件工程管理和技术空间上”。质量保证的能力是成熟的工程学科的尺度。当成功实现上述映射时，其结果就是成熟的软件工程。

SQA的内容如下。

（1）SQA过程。

（2）具体的质量保证和质量控制任务（包括技术评审和多层次测试策略）。

（3）有效的软件工程实践（方法和工具）。

（4）对所有软件工作产品及其变更的控制。

（5）保证符合软件开发标准的规程（当适用时）。

（6）测量和报告机制。

其中的管理问题和特定的过程活动，使软件组织确保“在恰当的时间以正确的方式做正确的事情”的具体活动。

3.4.2 SQA的要素

软件开发质量保证的历史同步于硬件制造。SQA涵盖了广泛的内容和活动，这些内容

和活动侧重于软件质量管理，可以归纳如下。

(1) 标准：IEEE、ISO 及其他标准化组织制定了一系列广泛的软件工程标准和相关文件。标准可能是软件工程组织自愿采用的，或者是客户或其他利益相关者责成采用的。SQA 的任务是要确保遵循所采用的标准，并保证所有的工作产品符合标准。

(2) 评审和审核：技术评审是由软件工程师执行的质量控制活动，目的是发现错误。审核是一种由 SQA 人员执行的评审，意图是确保软件工程工作遵循质量准则。例如，要对评审过程进行审核，确保以最有可能发现错误的方式进行评审。

(3) 测试：软件测试是一种质量控制功能，它有一个基本目标——发现错误。SQA 的任务是要确保测试计划适当和实施有效，以便尽可能地实现软件测试的基本目标。

(4) 错误/缺陷的收集和分析：改进的唯一途径是衡量如何做。SQA 人员收集和分析错误和缺陷数据，以便更好地了解错误是如何引入的，以及什么样的软件工程活动最适合消除它们。

(5) 变更管理：变更是对所有软件项目最具破坏性的一个方面。如果没有适当的管理，变更可能会导致混乱，而混乱几乎总是导致低质量。SQA 确保进行足够的变更管理实践。

(6) 教育：每个软件组织都想改善其软件工程实践。改善的关键因素是对软件工程师、项目经理和其他利益相关者的教育。SQA 组织牵头软件过程改进，并是教育计划的关键支持者和发起者。

(7) 供应商管理：可以从外部软件供应商获得以下三种类型的软件。

① 简易包装软件包，如微软 Office。

② 定制外壳，提供可以根据购买者需要进行定制的基本框架结构。

③ 合同软件，按客户公司提供的规格说明书定制设计和构造。

SQA 组的任务是，通过建议供应商应遵循的具体的质量做法(在可能的情况下)，并将质量要求作为与任何外部供应商签订合同的一部分，确保高质量的软件成果。

(8) 安全管理：随着网络犯罪和新的关于隐私的政府法规的增加，每个软件组织应制定政策，在各层面保护数据，建立防火墙保护 Web 应用系统，并确保软件在内部没有被篡改。SQA 确保应用适当的过程和技术来实现软件安全。

(9) 安全：因为软件几乎总是人设计系统(例如，汽车应用或飞机应用)的关键组成部分，潜在缺陷的影响可能是灾难性的。SQA 可能负责评估软件失效的影响，并负责启动那些减少风险所必需的步骤。

(10) 风险管理：尽管分析和减轻风险是软件工程师考虑的事情，但是 SQA 组应确保风险管理活动适当进行，且已经建立风险相关的应急计划。

此外，SQA 还确保将质量作为主要关注对象的软件支持活动(如维修、求助热线、文件和手册)高质量地进行和开展。

3.4.3 SQA 的任务

SQA 是由与两个不同人群相联系的多种任务组成，分别是做技术工作的软件工程师和负有质量策划、监督、记录、分析和报告责任的 SQA 组。软件工程师通过采用可靠的技术方法和措施进行技术评审，并进行周密计划的软件测试来获得质量(和执行质量控制活动)。

（1）SQA组的行动纲领是帮助软件团队实现高品质的目标产品。质量保证活动，即从事质量保证策划、监督、记录、分析和报告，这些活动由独立的SQA组执行（和完成）。

（2）编制项目质量保证计划。该计划作为项目计划的一部分，并经所有利益相关者评审。软件工程组和SQA组进行的质量保证活动都受该计划支配。该计划确定要进行的评估、要进行的审核和评审、适用于项目的标准、错误报告和跟踪的规程、SQA组产出的工作产品以及将提供给软件团队的反馈意见。

（3）参与项目的软件过程描述的编写。软件团队选择完成工作的过程。SQA组审查该过程描述是否符合组织方针、内部软件标准、外部要求的标准（如ISO 9001），以及是否和软件项目计划的其他部分一致。

（4）评审软件工程活动，以验证是否符合规定的软件过程。SQA组识别、记录和跟踪偏离过程的活动，并验证是否已作出更正。

（5）审核指定的软件产品，以验证是否遵守了作为软件过程一部分的那些规定。SQA组审查选定的产品，识别、记录并跟踪偏差，验证已经作出的更正，并定期向项目经理报告其工作成果。

（6）确保根据文档化的规程记录和处理软件工作和工作产品中的偏差。在项目计划、过程描述、适用的标准或软件工程工作产品中可能会遇到偏差。

（7）记录各种不符合项并报告给高层管理人员。跟踪不符合项，直到解决。

除这些活动外，SQA组还协调变更的控制和管理，并帮助收集和分析软件度量。

执行上述SQA活动，以实现一套务实的目标。

（1）需求质量。需求模型的正确性、完整性和一致性将对所有后续工作产品的质量有很大的影响。SQA必须确保软件团队严格评审需求模型，以达到高水平的质量。

（2）设计质量。软件团队应该评估设计模型的每个元素，以确保设计模型显示出高质量，并且设计本身符合需求。SQA寻找能反映设计质量的属性。

（3）代码质量。源代码和相关的工作产品（如其他说明资料）必须符合本地的编码标准，并显示出易于维护的特点。SQA应该找出那些能合理分析代码质量的属性。

（4）质量控制有效性。软件团队应使用有限的资源，在某种程度上最有可能得到高品质的结果。SQA分析在评审和测试上的资源分配，评估是否以最有效的方式进行分配的。

3.4.4 SQA计划

SQA计划为SQA提供了一张路线图。该计划由SQA小组（或者软件团队）制定，作为各个软件项目中SQA活动的模板。

IEEE公布了一个SQA计划标准，该标准建议SQA计划应包括如下内容。

（1）计划的目的和范围。

（2）SQA覆盖的所有软件工程工作产品的描述（如模型、文档、源代码）。

（3）应用于软件过程中的所有适用的标准和习惯做法。

（4）SQA活动和任务（包括评审和审核）以及它们在整个软件过程中的位置。

（5）支持SQA活动和任务的工具和方法。

（6）软件配置管理的规程。

（7）收集、保护和维护所有SQA相关记录的方法。

(8) 与产品质量相关的组织角色和责任。

3.4.5 统计软件质量保证

统计质量保证反映了一种在产业界不断增长的趋势：质量的量化。对于软件而言，统计质量保证包含以下步骤。

(1) 收集软件的错误和缺陷信息，并进行分类。

(2) 追溯每个错误和缺陷形成的根本原因(如不符合规格说明、设计错误、违背标准、缺乏与客户的交流)。

(3) 使用帕累托原则(80%的缺陷可以追溯到所有可能原因中的20%)，将这20%("重要的少数")原因分离出来。

(4) 一旦找出这些重要的少数原因，就可以开始纠正引起错误和缺陷的问题。

"统计质量保证"这个比较简单的概念代表的是创建自适应软件过程的一个重要步骤，在这个过程中要进行修改，以改进那些引入错误的过程元素。

值得注意的是，改正行动主要是针对"重要的少数"。随着这些问题的解决，新的"重要的少数"随之出现，并提上改正日程。

3.4.6 软件安全与可靠性

软件安全是一种SQA活动，它主要用来识别和评估可能对软件产生负面影响并促使整个系统失效的潜在灾难。如果能够在软件过程的早期阶段识别出这些灾难，就可以指定软件设计特性来消除或控制这些潜在的灾难。

建模和分析过程可以视为软件安全的一部分。开始时，根据危险程度和风险高低对灾难进行识别和分类。例如，与汽车上的计算机巡航控制系统相关的灾难可能有以下几种。

(1) 产生失去控制的加速，不能停止。

(2) 踩下刹车踏板后没有反应(关闭)。

(3) 开关打开后不能启动。

(4) 减速或加速缓慢。

一旦识别出这些系统级的灾难，就可以运用分析技术来确定这些灾难发生的严重性和概率。为了达到高效，应该将软件置于整个系统中进行分析。例如，一个微小的用户输入错误(人也是系统组成部分)有可能会被软件错误放大，产生将机械设备置于不正确位置的控制数据，此时当且仅当外部环境条件满足时，机械设备的不正确位置将引发灾难性的失效。失效树分析、Petri网模型等分析技术可以用于预测可能引起灾难的事件链，以及其中各事件出现的概率。

一旦完成灾难识别和分析，就可以进行软件中与安全相关的需求规格说明。在规格说明中包括一张不希望发生的事件清单，以及针对这些事件所希望产生的系统响应。这样就指明了软件在管理不希望发生的事件方面应起的作用。

计算机程序的可靠性是整个质量的重要组成部分。如果某个程序经常不能执行，那么其他软件质量因素是不是可接受的就无所谓了。与其他质量因素不同，软件可靠性可以通过历史数据和开发数据直接测量和估算出来。按统计术语所定义的软件可靠性是："在特定环境和特定时间内，计算机程序正常运行的概率"。举个例子来说，如果程序X在8小时

处理时间内的可靠性估计为0.999，也就意味着，如果程序X执行1000次，每次运行8小时处理时间（执行时间），则1000次中正确运行（无失效）的次数可能是999次。

软件可靠性涉及一个关键问题，即术语“失效”。在讨论软件质量和软件可靠性时，失效意味着与软件需求的不符，但这一定义有等级之分。失效可能仅仅是令人厌烦的，也可能是灾难性的。有的失效可以在几秒之内得到纠正，有的则需要几个星期甚至几个月的时间才能纠正。更复杂的是，纠正一个失效可能会引入其他的错误，而这些错误最终又会导致其他的失效。

尽管软件可靠性和软件安全彼此紧密相关，但是弄清它们之间的微妙差异非常重要。软件可靠性使用统计分析的方法来确定软件失效发生的可能性，而失效的发生未必导致灾难。软件安全则考察失效导致灾难发生的条件。也就是说，不能在真空中考虑失效，而是要在整个计算机系统及其所处环境的范围内加以考虑。

3.4.7 ISO 9000质量标准

与软件质量有关的标准之一是国际标准化组织（ISO）的ISO 9000。要想获得ISO 9000认证决非轻而易举的事。

ISO 9000是关于质量管理和质量保证的一系列标准，它以通用的术语描述了质量保证体系要素，定义了一套基本达标的实践，帮助公司不断地交付符合客户质量要求的产品（或者服务）。无论公司是一家修理铺，还是拥有数十亿资金的集团公司，是制作软件、鱼饵，还是配送快餐，ISO 9000都适用。好的管理实践对它们同等适用。

某个公司要登记成为ISO 9000质量保证体系中的一种模式，该公司的质量体系和实施情况应该由第三方的审核人员仔细检查，查看其是否符合标准以及实施是否有效。成功注册之后，这个公司将获得由审核人员所代表的注册登记实体颁发的证书。此后每半年进行一次监督审核，以此保证该公司的质量体系持续符合标准。

3.5 能力成熟度模型

成熟度模型是一种用来评估组织在特定领域（如软件开发、项目管理、质量保证等）的能力和改进程度的框架。这些模型通常定义了一系列的级别或阶段，每个级别代表了组织能力的不同层次，从基础到高级逐步提升。

3.5.1 软件能力成熟度模型

软件的能力成熟度模型（Capability Maturity Model，CMM）是一个行业标准模型，用于定义和评价软件公司开发过程的成熟度，提供怎样做才能提高软件质量的指导。它是在美国国防部的指导下，由软件开发团体和软件工程学院（SEI）及卡内基-梅隆大学共同开发的。

CMM的特别之处在于它是通用的，同等适用于任意规模的软件公司。它的5个等级为评测公司软件开发的成熟度提供了简单的手段，确定了步入下一级成熟度的关键措施。

5级CMM成熟度描述如下。

1级：初始的。该等级的软件开发过程通常是临时且无序的。项目成功依靠个人英雄的行为和运气。过程没有通用的计划、监视和过程控制。开发软件的时间和费用无法预知。

测试过程与其他过程混杂在一起。

2级：可重复的。该等级成熟度的最好描述是项目级的思想，基本项目管理过程已经建立并得到执行，跟踪项目费用、进度、功能和质量。以前类似的项目经验可以应用到当前项目中。该等级有一定的组织性，使用了基本软件测试行为，如测试计划和测试用例。

3级：定义的。该等级具备了组织化思想，而不仅仅是针对具体项目。通用管理和工程活动被标准化和文档化，并在整个组织中推广。这些标准在不同的项目中被采用并得到证实。当压力增加时，不会放弃规则。在测试开始之前，要审查和批准测试文档和计划。测试团队与开发人员独立。测试结果用于确定软件完成的时间。

4级：可管理的。在该成熟度等级中，组织过程处于统计的控制之下，对过程性能进行测量和控制。产品质量事先以量化的方式指定(例如，产品直到每1000行代码只有0.5个以下错误才能发布)，软件在未达到目标之前不得发布。在整个项目开发过程中，收集开发过程和软件质量的详细情况，经过调整校正偏差，使项目按计划进行。

5级：不断优化的。该等级称为不断优化(不是“已经优化”)是因为它从4级不断提高。持续改进过程性能，通过创新和技术变革不断优化，采用提高和创新的变动以期达到质量更佳的等级。正当所有人认为已经达到最佳时，新的想法又出现了，再次提高到下一个等级。

针对上述5个等级必需的条件，思考以下问题：从当今社会软件公司的现状看，最多的成熟度为1级，多数成熟度为2级，少数成熟度为3级，极少数成熟度为4级，成熟度为5级的更是凤毛麟角。

能力成熟度模型集成(CMMI)是目前最广泛使用的成熟度模型之一，它整合了多个领域的最佳实践，旨在帮助企业提高其过程管理和系统工程的能力。

应该认识到，倡导公司提高软件开发成熟度不是软件测试员的事。在开始新的测试工作时，应该确定公司和新的小组处于哪一级的成熟度。了解所处的等级或者追求的等级，有助于定出期望值，更好地理解组织的期望。

3.5.2 测试成熟度模型

软件测试过程是软件过程的一部分。CMM提出测试是软件过程的关键组成部分，但并没有对测试过程管理作详细规定。1996年，伯恩斯坦等参照CMM提出了测试成熟度模型(Test Maturity Model，TMM)，将其作为CMM的补充，其目的在于帮助软件组织改进和评价其软件测试过程。

TMM是一个采用分级方法确定软件组织软件测试能力成熟度的参考模型，可以指导软件组织提高其软件测试能力。TMM将测试划分为5个级别，分别是初始级、阶段定义级、集成级、管理和度量级、优化与缺陷预防及质量控制级，每个低级别均要达到一定的测试目标才能上升到更高一级。

(1) 处于初始级的软件测试过程没有定义成熟度目标，软件测试是一个完全混乱的过程，测试在编码完成之后进行，并且与程序调试未加区分。

(2) 处于阶段定义级的测试已具备基本的测试技术和方法，测试已经与程序调试区分开来，并且定义为紧随软件编码完成之后的一个独立阶段，但测试计划往往在编码之后才制定，有悖于软件工程的要求。

(3) 处于集成级的测试不再仅是编码完成之后的一个阶段，而是集成到整个软件生命

周期中，测试人员在需求分析阶段便开始着手制订测试计划，并根据用户或客户需求建立测试目标，同时设计测试用例并制订测试通过准则。

（4）处于管理和度量级的测试，测试活动除包括测试被测程序外，还包括软件生命周期中各阶段的评审、审查和追查，使测试活动涵盖了软件验证和软件确认活动，测试过程不再只是定性的描述，已经是一个可量化度量的过程。

（5）处于优化与缺陷预防及质量控制级的测试是在前4级的基础上，缺陷预防和质量控制已经实施，可以监控测试成本和效率，测试过程的管理为持续改进产品质量和过程质量提供指导，已经具备了调整和连续改进过程的机制。

随着CMM发展成CMMI，TMM也升级为TMMi（测试成熟度模型集成）。TMMi专注于软件测试过程的成熟度，它基于CMMI的思想，但更具体地针对测试活动。

（1）初始级：测试过程缺乏一致性。

（2）已定义级：建立了基本的测试政策和程序。

（3）集成级：测试过程与整个开发过程紧密结合。

（4）管理与度量级：使用定量方法来管理和度量测试过程。

（5）优化级：持续改进测试过程，以实现更高的效率和效果。

此外，SPICE（软件过程改进和能力测定）也是一个国际标准，用于评估和改进软件过程的能力，它提供了一种通用的方法来衡量软件过程的质量，包括不完整级、已执行级、已管理级、已建立级、可预测级、优化级。

【习题】

1. 1979年，菲利普·克罗斯比提出，制造高质量的产品比制造低质量产品实际上不需要额外开销。因为软件缺陷发现得越晚，其处理费用（　　）。

A. 呈随机状态，不好把控　　B. 是个恒定数字，但数额巨大

C. 就越低，成倍下降　　D. 就越高，并且按指数级激增

2.（　　）都被称为“质量之父”，他们撰写了大量书籍，其实践结论也在世界范围内广泛使用，他们提出的概念常常适合于所有领域。

① 菲利普·克罗斯比　　② 约瑟夫·朱兰

③ 冯·诺依曼　　④ W·爱德华兹·戴明

A. ①②④　　B. ①③④　　C. ①②③　　D. ②③④

3. 考虑把质量的费用分为两类。其中的（　　）是指与一次性计划和执行测试相关的全部费用，用于保证软件按照预期方式运行。

A. 常规测试费　　B. 一致性费用　　C. 计划性费用　　D. 非一致性费用

4. 如果软件缺陷被遗漏并落到客户手里，结果就是代价昂贵的客服支持，需要修复、重新测试和发布软件，甚至召回或者对簿公堂。这些外部失败的费用属于（　　）。

A. 常规测试费　　B. 一致性费用　　C. 计划性费用　　D. 非一致性费用

5.“（　　）”的软件提供用户期望的高质量功能和特性，但同时也提供了其他更多的包含已知错误的难解或特殊的功能和特性。公司将在以后的版本继续提高产品质量。

A. 退役版本　　B. 最终版本　　C. 足够好　　D. 无缺陷

6. 质量成本包括追求质量过程中或在履行质量活动中引起的费用以及质量不佳引起的下游费用等所有费用，可分为(　　)。

① 增值成本　② 预防成本　③ 评估成本　④ 失效成本

A. ②③④　B. ①②③　C. ①②④　D. ①③④

7. (　　)包括计划和协调所有质量控制和质量保证活动的成本，为开发完整需求模型和设计模型的技术活动成本，测试计划成本以及与这些活动有关的所有培训成本。

A. 增值成本　B. 预防成本　C. 评估成本　D. 失效成本

8. (　　)是那些如果在将产品发给客户之前或之后没有错误就不会存在的费用。其中内部是指在发货之前发现错误，例如，为纠正错误进行返工(修复)所需的成本。

A. 增值成本　B. 预防成本　C. 评估成本　D. 失效成本

9. (　　)包括为深入了解产品“第一次通过”每个过程的条件而进行的活动。例如，对软件工程工作产品进行技术审查的成本，测试和调试的成本等。

A. 增值成本　B. 预防成本　C. 评估成本　D. 失效成本

10. 软件质量受管理决定的影响往往和受技术决定的影响是一样的，即使最好的软件工程实践也能被糟糕的商业决策和有问题的项目管理活动破坏。这样的管理活动有(　　)。

① 估算决策　② 沟通协调　③ 进度安排　④ 风险决策

A. ①②③　B. ①③④　C. ①②④　D. ②③④

11. 各种各样的软件质量度量和因素，都试图定义一组属性，以推动实现较高的软件质量，这些质量因素特性包括(　　)、功能性和可移植性等。

① 可靠性　② 易用性　③ 灵活性　④ 维护性

A. ①②④　B. ①②③　C. ②③④　D. ①③④

12. 麦卡尔等提出的影响软件质量因素侧重于软件产品的3个重要方面，即(　　)。

① 操作特性　② 价值水平　③ 变更能力　④ 适应能力

A. ②③④　B. ①②③　C. ①②④　D. ①③④

13. WebApp是一种独特的软件类型。软件质量的所有技术特征以及通用质量属性都适用于WebApp。其中(　　)、效率及可维护性等特性为评估Web系统的质量提供了有用基础。

① 可用性　② 功能性　③ 灵活性　④ 可靠性

A. ②③④　B. ①②③　C. ①②④　D. ①③④

14. 良好的软件质量是良好的项目管理和扎实的软件工程实践的结果。帮助软件团队实现高质量软件的四大管理活动是软件工程方法、(　　)。

① 数值计算方法　② 质量控制活动　③ 项目管理技术　④ 软件质量保证

A. ①②③　B. ②③④　C. ①②④　D. ①③④

15. 质量保证体系可以定义为：用于实现质量管理的组织结构、(　　)和资源。其目的是帮助组织以符合规格说明的方式，保证组织的产品和服务满足客户的期望。

① 责任　② 规程　③ 平台　④ 过程

A. ①②④　B. ①③④　C. ①②③　D. ②③④

16. 为了正确地进行SQA，必须(　　)有关软件工程过程的数据。基于统计的SQA有

助于提高产品和软件过程本身的质量。

① 收集　② 评估　③ 发布　④ 渲染

A. ①③④　B. ①②④　C. ①②③　D. ②③④

17. 计算机程序的可靠性是整个质量的重要组成部分。按统计术语所定义的软件可靠性是："在特定环境和特定时间内，计算机程序正常运行的(　　)"。

A. 场合　B. 时间　C. 条件　D. 概率

18. 成熟度模型是一种用来评估组织在特定领域(　　)的能力和改进程度的框架，通常定义了一系列的级别或阶段，代表了组织能力的不同层次。

① 盈利能力　② 软件开发　③ 项目管理　④ 质量保证

A. ①②③　B. ②③④　C. ①②④　D. ①③④

19. (　　)是一个行业标准模型，用于定义和评价软件公司开发过程的成熟度，提供怎样做才能提高软件质量的指导。

A. CMM　B. TMM　C. ISO　D. BASE

20. 软件测试过程是软件过程的一部分。伯恩斯坦等提出了(　　)，将其作为CMM的补充，其目的在于帮助软件组织改进和评价其软件测试过程。

A. CMM　B. TMM　C. ISO　D. BASE

【实验与思考】 软件质量维度的新思考

本章"实验与思考"的目的是：

(1) 熟悉关于软件质量和质量保证的基本概念和内容。

(2) 熟悉软件质量学术观点和相关质量标准，不断丰富软件质量最新知识。

(3) 通过网络资料的收集和分析，了解软件质量管理工具。

1. 实验内容与步骤

接受迈耶的观点，生产"足够好"的软件是可接受的，事实上人们也正是这么做的。请寻找并记录一个你认为是"足够好"的软件。

请记录：说出具体公司的名字以及你认为运用"足够好"思想开发的具体产品。

答：_____

_____

_____

_____

参见本章3.3.3节，加文建议采取多维的观点考虑质量。请思考，为加文提出的每个质量维度添加两个额外的问题。这些内容、功能和特性在某种程度上是需求模型所规定的一部分，可以为最终用户提供价值。

① 性能质量。软件是否交付了所有的内容、功能和特性？

新问题1：_____

_____

新问题2：_____

_____

② **特性质量**。软件是否首次提供了使最终用户满意的特性？

新问题 1：

新问题 2：

③ **可靠性**。软件是否无误地提供了所有的特性和能力，当需要（使用该软件）时，它是否是可用的，是否无错地提供了功能？

新问题 1：

新问题 2：

④ **符合性**。软件是否遵从本地的和外部的与应用领域相关的软件标准，是否遵循了事实存在的设计惯例和编码惯例？

新问题 1：

新问题 2：

⑤ **耐久性**。是否能够对软件进行维护（变更）或改正（改错），而不会粗心大意地产生意料不到的副作用？随着时间的推移，变更会使错误率或可靠性变得更糟吗？

新问题 1：

新问题 2：

⑥ **适用性**。软件能在可接受的短时期内完成维护（变更）和改正（改错）吗？技术支持人员能得到所需的所有信息以进行变更和修正缺陷吗？

新问题 1：

新问题 2：

⑦ **审美**。关于美的问题很难量化，但显然是不可缺少的。

新问题 1：

新问题 2：

⑧ **感知**。在某些情况下，一些偏见将影响人们对质量的感知。

新问题 1：

新问题 2：

2. 实验总结

3. 实验评价（教师）