测试员培训入门教材

版本号<1.1>

中国测试员论坛 http://www.renju2002.com/cgi.asp

2003-3-17

1. 修订历史

创建者姓名：郭荣力
创建时间：2003-3-17
版本号：<1.0 >

修改者姓名：郭荣力
修改时间：2002-3-19
版本号：<1.1>
修改内容：增加测试案例的内容

2. 目录

3. 引言

3.1. 本文编写目的

这是为培训专业测试人员参加测试工作，
而编写的包含测试基础知识的入门培训教材。

3.2. 读者范围

将来参加测试工作的测试人员或者将来参加开发的程序员。

3.3. 专业术语说明

3.3.1. 软件缺陷

软件中含有符合下面 5 条规则之一的问题称为软件缺陷：

- 软件未达到产品说明书标明的功能。
- 软件出现产品说明书指明不会出现的错误。
- 软件功能超出产品说明书指明的范围。
- 软件未达到产品说明书未指出但应达到的目标。
- 软件测试人员或用户认为软件难以理解，不易使用，运行速度缓慢等问题。

3.3.2. 测试案例

测试用例的别名。

3.3.3. 黑盒测试

指测试人员通过各种输入和观察软件的各种输出结果来发现软件的缺陷，而不关心程序具体如何实现的一种测试方法。

3.3.4. 静态测试

指测试不运行的部分，例如测试产品说明书，对此进行检查和审阅。

3.3.5. 静态白盒测试

指在不执行的条件下有条理地仔细审查软件设计，体系结构和代码，从而找出软件缺陷的过程。有时称作结构分析。

3.3.6. 动态测试

通过运行和使用软件进行测试。

3.3.7. 探索测试

通常用于没有产品说明书的测试，这需要把软件当作产品说明书来看待，分步骤逐项探索软件特性，记录软件执行情况，详细描述功能，综合利用静态和动态技术来进行测试。

3.3.8. 等价区间

指测试相同目标或者暴露相同软件缺陷的一组测试用例。

3.3.9. 测试设计

提炼测试方法，明确指出设计包含的特性和相关测试。如果要求完成测试还明确指出测试案例和测试程序，指定特性通过/失败的规则。

3.3.10. 软件 QA

QA= Quality Assessment 质量评价。防止软件缺陷称为软件 QA。

3.3.11. TQM 或者 TQC 原理

TQM(全面质量管理)或者 TQC(全面质量控制)。其原理是，用集中的质量评判团队来负责质量是不实际的，因为工作的人不负责质量，所以他们不会设法实现质量评判目的。要想制造高质量产品，需要创立从管理开始自上而下的质量意识，使全体成员共同承担质量责任。

3.3.12. SQC

软件质量控制(SQC)是测试团队很常用的名称。该名称来源于制造行业，其中 QC 检验员对生产线上的产品进行采样、检测，如果测试失败，他有权停掉生产线或者整个工厂。测试团队很少有这种授权。软件 QC 团队也是如此。

3.3.13. Murphy 法则

永远不会有足够的时间把事情做好，但是总有时间返工。软件开发小组需要遵循一个过程，花费一些时间，变得有条理，一开始就设法作对。

3.4. 参考资料

《Software Testing》（美）Ron Patton 著 Copyright© 2001 by Sams Publishing

4. 测试人员的目标

找出软件缺陷，尽可能早一些，并保证其得到修复。

5. 测试工作过程要点

利用组织良好的测试计划、测试案例和测试报告正确交流和制定来完成的测试工作，是

测试员达到目标的保障。,

6. 检查代码

6.1. 静态白盒测试

进行静态白盒测试的首要原因是尽早发现软件缺陷,以找出动态黑盒测试难以揭示或遇到的软件缺陷。独立审查代码的人越多越好，特别是在开发过程初期从底层进行。另外可以为黑盒测试人员提供思路，他们不必了解代码的细节，但是根据审查备注，可以确定似乎有问题或者存在软件缺陷的特征范围。
开发小组没有专人负责白盒测试，一般由程序员组织和执行审查人员，软件测试人员被当作独立的观察者。也有测试人员是该任务执行人，要求编写代码的程序员和其他同事帮助审查。
静态白盒测试常见问题是不能善始善终。很多小组认为费用太高，没有产出。这是不正确的，很多公司已经招聘和培训程序员和测试员进行白盒测试了。

6.1.1. 正式审查

6.1.1.1. 正式审查有四个要素：

- 确定问题。审查的目标是找出软件问题，包括出错项目和遗漏项目。
- 遵守规则。审查需要固定的规则，如审查代码的行数，花的时间，那些内容需要备注等。
- 准备。每个合作者需要知道自己的职责，很多问题是在准备期间发现的。
- 编写报告。必须有书面报告，使报告便于开发小组使用。

6.1.1.2. 同事审查

这是一种最简单的方法，一般由一两个程序员和测试员一起进行，为了不至于成为闲聊，需要遵守正式审查的四个要素。这种聚集起来讨论代码也可以找出软件缺陷。

6.1.1.3. 公开陈述

编写代码的程序员向 5 人小组或者其他类似程序员和测试员正式表述。审查人员之中应该有一名资深程序员是很重要的。

6.1.1.4. 检验

最正式的审查类型，参与者称为检验员，职责从不同角度包括用户，测试员和产品

支持人员角度来审查产品。有些检验员被委任为会议主席和会议记录，保证检验过程遵守规则及审查。会议后可能检验员要碰头讨论发现的不足，程序员进行修改。最后由主席检验修改结果。检验被证明为在设计文档和代码中发现软件缺陷最有效的方法。

6.2. 编码规范和标准

可以运行并且测试中也表现稳定的代码被称为有问题，令人不易理解。一般有三个重要原因需要坚持标准和规范：

- 可靠性。事实证明按照某种标准或者规范写的代码更加可靠。
- 可读性/维护性。符合设备标准的规范代码容易阅读、理解和维护。
- 移植性。如果代码符合设备标准，移植将很轻松。

6.3. 静态白盒测试可能遇到的问题类型

- 数据引用出错
- 数据声明错
- 计算错误
- 比较错误
- 控制流程错误
- 子程序参数错误
- 输入/数出错误
- 其他

6.4. 动态白盒测试

6.4.1. 基本测试内容

- 直接测试底层功能、过程、子程序和库。
- 以完整程序方式从顶层测试软件，然后根据软件运行了解和调整测试案例。
- 从软件获得读取变量和状态信息的访问权，以便确定测试与预期结果是否相等。
- 估算执行测试时命中的代码量和具体代码，然后调整测试。

6.4.2. 动态白盒测试和调试

两者不能混为一谈，虽然会有交叉，调试是程序员做的，目的是修复问题。测试是为了找到缺陷。测试员可能要使用代码级调试器单步执行，观察变量，设置断点等。对于要求合法性检查的独立代码模块，还要编写测试程序进行测试。

6.4.3. 黑盒与白盒

进行白盒测试之前，要根据说明书建立黑盒测试案例，这种方法可以真正理解测试用途。否则会偏向模块的工作方式，程序员的说明也许包含错误，所以测试案例可能出现问题。

6.4.4. 数据范围

白盒测试合理的方法是把软件分成数据和状态（或者程序流程）。同时可以把白盒信息映射到已经写完的黑盒案例中。
首先考虑数据：包括所有变量、常量、数组、数据结构、键盘鼠标输入、文件、屏幕输入输出。以及调制解调器、网络等其他设备的输入/输出。
数据可以分成如下类型：

- 数据流。观察数据在各个模块甚至整个程序中的各种状态值。
- 次边界。寻找次边界条件。比如 2 的乘方。
- 公式和等式。比如被 0 除问题。
- 错误强制。有的错无条件不好模拟，需要制造错误。但是不要制造实际无法出现的错误。

6.4.5. 代码覆盖

为了全面，必须测试程序的状态及其中的程序流程，设法进入和退出每一个模块，执行每一行代码，追踪每一条逻辑和决策分支。
最简单的方法是利用编译环境单步执行。大多数程序要用专门的代码范围分析器。
需要注意的是，全部语句都执行一遍，不等于走遍了软件的所有路径。所以需要分支覆盖。分支覆盖也不完全，还要考虑条件范围问题。
如果测试了所有可能的条件，达到了分支覆盖，顺便达到了语句覆盖。

7. 配置测试

因为无法保证用户的设备都符合通用的标准，所以需要把软件放在用户使用比较广泛的硬件上进行测试。

测试要点：

- 确定需要测试的硬件类型。
- 明确硬件标准。
- 确定可能的硬件属性，模式和选项。
- 分离配置缺陷。
- 等价分配。
- 只测试各种硬件不交叉的部分。

8. 文档测试

软件测试员不仅要测试软件同时需要测试文档，因为他要负责整个软件产品的各个部分的质量。

8.1. 文档的类型

- 包装文字和图形。
- 市场宣传资料、广告和其他资料。
- 授权/注册登记表。
- EULA 即最终用户许可协议，一般要求用户不经同意不可以复制软件，如果受到软件缺陷的损害，不得向生产厂家起诉。
- 标签和不干胶。
- 安装和设置指导。
- 用户手册。
- 联机帮助。
- 指南，向导和 CBT(计算机基础训练)。
- 样例、示例和模板。
- 错误提示信息。

8.2. 文档测试的重要性

如果安装指导有问题，不正确的错误提示信息把用户引入歧途，他们会认为这是软件缺陷。文档和代码对于用户来说是一样的。

8.3. 文档测试问题类型

- 文档面对的听众级别是否合适。
- 术语是否适用于听众，是否用法一致，所有术语是否可以被正确索引。
- 内容和主题是否有遗漏或者多余。材料深度是否合适。
- 所有信息是否准确。产品说明是否过时，技术支持网站链接和电话是否准确。
- 逐步执行。象用户一样来按步骤使用，看看是否遗漏某些说明。
- 图表和屏幕抓图。来源和表现是否正确。
- 样例和示例。向客户一样使用样例，如果是代码就要执行。样例如果不能运行给客户印象特别不好。
- 拼写和检查。

9. 其他测试

以下测试内容被省略：

- 兼容性测试

- 本地化测试
- 易用性测试
- 网站测试
- 自动测试及测试工具

10. 借助他人测试

不同人的角度不同，可以有效打破杀虫剂现象。
请产品支持或者客户服务帮助测试是很有效的一种方法。

10.1. 测试共享

如果几个测试员来测试，常用方法是在一定时间内简单互换测试任务。

10.2. 测试轰炸

测试人员同时停下工作，选择软件的某一块区域，集中进行测试。称为测试轰炸。

10.3. Beta 测试

他是一种外部测试方法。该过程中，软件分发给选定的潜在用户，他们在实际环境中使用软件。

Beta 测试需要考虑的几个问题：

10.3.1. 谁是 Beta 测试者

Beta 测试有不同的目标，所以必须了解测试者的类型。比如测试人员想要找出测试中残存的易用性错误，如果 Beta 测试者是技术人员的话，将更关心底层的技术，对于易用性不是很关心，这样测试效果将不是很好。

10.3.2. 如何知道 Beta 测试者用过软件

1000 个 Beta 测试者拿到 Beta 一个月后，没有任何错误报告，是否可以认为没有缺陷还是用户没有收到软件，还是过些时间才开始测试。测试执行者需要追问参加者，保证他们在使用软件并符合计划的目标。

10.3.3. Beta 测试的优缺点

Beta 测试可以成为寻找配置和兼容性软件缺陷的好方法。
Beta 测试对易用性测试非常有好处。
Beta 测试对于寻找其他软件缺陷方面出人意料的差。

10.3.4. 第三方测试

提交给其他公司进行转包测试，看上去比较麻烦，费用也高，但是如果做的好，可能成为共享测试的有效途径。

11. 计划测试工作

测试过程不可能在真空里工作，程序员编写代码，不说明它的功能，如何工作，何时完成，执行测试任务就难了。测试员之间不交流测试的对象，需要什么资源，进度如何安排，项目很难成功。
计划测试工作主要目的是交流软件测试小组的意图、期望以及对将要执行的任务的理解。而编制的测试计划通常成为空架子，以后不会有人看。所以计划工作的目标应该从建立文档转移到计划建立过程。

11.1. 测试计划主题

由于测试一般都有模板，所以很容易作出计划文档，然而如果只有文档，当产品小组的人没人知道测试员在做什么，或者为什么做的时候，测试的弊端就出来了。

11.1.1. 明确测试目标

测试计划过程和软件测试计划的目的是什么？程序员和技术作者及管理部门知道吗？
测试的产品是什么？是一个完全重写一个产品还是仅仅维护和升级，是独立程序还是很多小程序的组合？是自行开发的还是外包出去的，它的到底是什么东西？
产品质量和可靠性目标是什么？
测试计划过程的结果必须是产品质量和可靠性目标的清晰、简洁和一致通过的定义。

11.2. 测试的组织工作

测试计划中应该包括项目中所有主要人员的清单和联系方式。同时，文档存放的位置，测试工具从那里得到，使用什么硬件如果必要都需要指出。这些内容最好类似于“ 测试新手问题指南 ”。通常是新手负责的绝好部分，找到所有问题答案，把发现记录下来。

11.3. 明确定义

常用一些定义需要明确：
版本生成：测试计划应该定义构建版本的频率（每天或每周等等）以及预定义的质量等级。
版本发布文档：程序员提供的声明新特性、不同特性、修复问题和准备测试的内容。
Alpha 版：对少数主要客户和市场进行数量有限的分发，用于演示的目的软件版本。使用 Alpha 版的所有人应该了解确切的内容和质量等级。
Beta 版：向潜在用户广泛分发的正式版本。
说明书完成：说明书预计完成并且不再修改的计划安排。实际工作中，这个期限是无法实现的，但是需要确定下来，以后只会进行小的改动。
软件缺陷会议：由测试人员、项目管理员、开发管理员和产品支持组成的团队，审查缺陷，决定那些需要修改，那些不需要。

11.4. 需要和不需要测试的部分

实际工作中有不需要测试部分存在的可能性。

11.5. 定义测试阶段

明确测试进入和退出的规则。计划中应该确定每个预定的阶段。否则就会成为无头绪的单个工作。

11.6. 决定测试策略

使用白盒还是黑盒，如果需要结合在一起，如何做。是否使用工具，是否需要提交给其他专业公司测试。这项工作需要资深测试员来做。

11.7. 资源要求

- 人员
- 设备
- 办公空间
- 软件
- 是否或如何选择测试公司
- 其他供应：软盘，电话，参考书，培需资料等。

11.8. 测试人员的任务分配

11.9. 测试进度

如果一个特性虽然看起来很容易设计和完成，但是如果需要很多时间来测试，就要考虑是否要真正实现。
另外测试工作量在软家开发过程中是不平均分配的，比如从测试说明书和代码审查开始，测试的任务，人员和工作量不断增长，到产品发布之间会形成短暂的高峰。结果是不断受到先前事件影响，如果前提条件拖延，那么应该压缩测试时间还是把项目推迟，这称为进度危机。为了避免这种危机，
建议使用进入退出标志，使用相对进度：
例如测试计划任务在说明书完成之后 7 天。期限 4 周。
测试案例开始于测试计划完成需要 12 周。
第一阶段测试开始于代码完成可测试版本，期限 6 周。
第二阶段测试开始于 Beta 版完成，期限 6 周。
第三阶段测试开始于软件发布版本完成，期限 4 周。

11.10. 测试案例

计划决定使用什么方法编写测试案例，那里保存，如何使用和维护。

11.11. 缺陷报告

决定如何记录和跟踪缺陷。

11.12. 频度和统计

测试计划中应该明确收集那些信息，做什么决定，谁来收集，例如：
项目期间每天发现的软件缺陷总数。
仍然需要修复的软件缺陷清单。
根据严重程度对当前软件缺陷评级。
每个测试人员找出软件缺陷的总数。
从每个特性或者区域发现的软件缺陷数目。

11.13. 风险和问题

明确指出项目的潜在问题或者风险区域，对测试工作的影响之处。

12. 测试案例的编写和跟踪

12.1. 测试案例计划的目标

测试员拿到测试计划马上坐下来想出测试案例并开始计算是不正确的，正如程序员马上拿到产品说明书马上编码一样。
需要仔细计划测试案例的原因：
组织性 :正确计划案例，可以组织好测试案例并且使全部测试员和其他成员有效审查和使用。
重复性：项目期间需要多次执行同样测试，需要有个计划。
跟踪：可以回答计划执行多少个案例，最终版本需要多少案例，多少个通过，有被忽略等问题。
测试证实：高风险行业，测试小组必须正式确实执行了某些测试，发布忽略某些测试案例的软件实际是危险和不合法的。

12.2. 测试案例的要点

ANSI/IEEE 829 标准：
标识符：测试设计过程和测试程序说明引用的唯一标识符。
测试项：被测试的详细特性、代码模块等。如“ 加法运算的上限溢出错误 ”。
输入说明：列举送到软件执行测试案例的所有输入内容或者条件。
输出说明：描述进行测试案例的预期结果。
环境要求：指执行测试必须达到的软、硬件、测试工具、实用工具、人员等要求。
特殊要求：描述执行测试必须做到的特殊要求。
案例之间的依赖性：如果一个测试案例依赖于其他案例，应该在此注明。

实际上只能把上面的参考资料作为规范，而不是标准。大多时候使用简便方法。
需要发挥创造力，使用最有效的方法，使用简单列表，大纲甚至状态图或者数据流程图之类的图表。设法与他人交流测试案例，并且使用最有效的方法，但不要偏离编写测试案例的目的。

12.3. 测试脚本说明

该文档定义了执行测试案例的每一步步骤：

标识符：把测试过程与相关测试案例和测试设计捆绑在一起的唯一表示符。
目的：程序的目的及要执行的测试案例的引用信息。
特殊要求：执行程序需要的其他程序、技术或者特殊设备。
程序步骤：执行测试的详细步骤。
日志：指出用什么方式、方法记录结果和现象。
设置：说明如何准备测试。
启动：说明启动测试的步骤。

程序：描述用于运行测试的步骤。
衡量标准：描述如何判断标准－例如用秒表或者凭眼睛判断。
关闭：说明由于意外原因推迟测试的步骤。
中止：描述测试正常停止的步骤。
重置：说明如何把环境恢复到测试前的状态。
偶然事件：说明如何处理计划之外的情况。

对于测试过程只说：“ 尝试执行所有测试案例并回报发现的问题等 ” 是不够的。
应该使用详细的程序说明，要测试什么，如何测试都是一目了然的。

12.4. 细节和真实

“ 适可而止 ”特别适用于测试案例计划。通常不可能需要按照最细致的程度编写测试案例。无比详尽的测试案例说明减少了随意性，使测试很好重复，使无经验的测试员按照预定想法执行测试。缺点是，编写如此细致的测试案例说明需要化费很多时间和精力，增加升级难度。由于细节繁多，阻碍了测试工作，造成执行测试时间变长。
开始编写测试案例时，最好的机会是采用当前项目的标准。

12.5. 测试案例的组织和跟踪

一般应该考虑下面的问题：

- 计划执行多少个测试案例？需要执行多少时间？
- 能否挑选出测试相关案例组测试某些特性或者软件部分？
- 执行测试案例时，能否记录那一个通过，那一个失败。
- 失败的案例中，那些在上次执行时也失败了。
- 上次执行测试案例通过的百分比是多少。

12.6. 跟踪方式

- 用脑子记是最不可取的。
- 书面文档。使用纸和笔填写的含检查清单的表格和框图，提供了包含程序员签字的书面检查清单是法庭上证明测试执行过程的极佳证据。
- 使用电子表格是很流行的方法。
- 自定义数据库也是非常好的方法。

13. 评价成效

利用从缺陷跟踪工具库得到的数据可以回答类似下面的问题：

- 正在测试的软件各个区域缺陷分布情况。
- 某个测试员已经关闭的缺陷数。
- 某个时间段找出的缺陷数。
- 所有优先级为 1 的缺陷清单。
- 软件是否可以按正常发布期限发布。

13.1. 日常测试中使用的指数

一般可以给出下面这些问题的答案：

- 交给我来关闭的已解决软件缺陷的 ID 是什么？
- 该项目输入了多少软件缺陷？针对用户界面输入那些软件缺陷以不修复形式解决？
- 发现的软件中有多少严重级为 1 或者严重性为 2 的？
- 全部缺陷中修复了多少？推迟了多少？重复了多少。

各种统计指数分析：
测序员发现的缺陷的优先级比重饼图。
测试员发现的缺陷如何处理的直方图。（修复、未报告、重新提交、不是问题、推迟修复）。

13.2. 常用项目级指数

显示软件每个主要功能区发现软件缺陷数目的项目级饼图。
发现缺陷的时序图（按每天发现缺陷情况）可以发现很多问题。
上面的图加上累加软件缺陷趋势图。
随时间推迟的打开，解决和关闭的软件缺陷图。该图表一般使用彩色表示：
红色代表打开的软件缺陷，黄色代表解决的软件缺陷，绿色代表关闭的软件缺陷。

14. 软件质量评判

14.1. 制作高质量产品的费用

软件缺陷发现的越晚，处理费用越高。

14.2. 软件测试

找出软件缺陷，有效地描述他们，通知相应的人，并跟踪软件缺陷直到解决。

14.3. 质量评判

测试无法保证产品质量。QA 团队是对项目进行近似完全地控制，建立标准和方法论，有条理地仔细监视和评价软件开发过程，对发现地软件缺陷提出解决建议，执行某些测试（或者忽略），拥有决定产品何时发布的权。让一个管理员力争实现“ 无缺陷 ”的目标，而不是使软件如期发布或者低于预算。
软件 QA 的目标是防范软件缺陷，在产品说明书、设计文档和代码上执行静态测试，就算一种软件 QA，因为这样防止了软件缺陷出现。

14.4. 测试人员的其他任务

软件测试团队进行配置管理或者构造软件等是不正常的。这样作有两个问题：
占用了应该用于测试产品的资源
测试团队的最终目的是破而不是立，承认软件的构造过程形成利益冲突。
最好让程序员或者独立小组构造软件。测试应该专注于寻找软件缺陷。

14.5. 测试管理与组织结构要点

小型开发小组（小于 10 人）常用组织结构：
测试团队向管理员报告，他同时管理程序员的工作。这种组织形式的缺陷在于编写代码的人和寻找缺陷的人向同一个报告，可能引起利益冲突。
开发管理员的目标是促使其小组开发软件，测试员报告软件缺陷只会妨碍这个过程。如果管理员为测试员提供很多资源和经费，他们会找出更多缺陷，但他们找到的缺陷越多，就越妨碍管理员制作软件的目标。
但是如果开发管理员的经验很丰富，认识到其目标不仅是制作软件，同时要制作高质量软件，这个结构也可以运行的很好。管理员会一视同仁，有利于相互交流，管理层次也较少。
另一种组织结构：
测试团队和开发团队都向项目管理员报告。测试团队一般有自己的负责人和管理员，其利益和注意力集中在测试小组及其工作上，这种独立性在对软件质量做重大决定时极有好处。测试小组的意见与产品制作者的意见同等重要。然而，该组织结构的缺点是项目管理员做最终决定。在高风险或者任务要求严格的系统开发中，由更高等级拥有质量意见最终决定权是有益的。
最后一种组织结构：
该组织中，负责软件质量的小组直接向高级管理员报告，相对独立，拥有与各项目同等的项目等级。该团队的独立性允许他们建立标准和规范，评价结果，采取跨越多个项目的处理措施。关于质量优略的信息可以直达最高层。这种授权，并不意味着他们可以在软件项目和用户不要求的前提下，建立不合理和难以实现的质量目标。这个独立的质量组织必须设法面对所有的项目，利用发布软件的经验稳定盲目追求质量的热情。同时必须保持程序员和其他小组成员的紧密合作关系。随着交流路线的分散，这一点越来越难

以控制。

14.6. 能力成熟度模型（CMM）

美国国防部指导，由软件开发委员会和软件工程学会(SEI)和卡内基梅隆（Carnegie Mello）大学共同开发。

5 级 CMM 成熟度：简述和测试有关的方面。

- 初步。测试与其他活动混在一起。
- 可重复级。基本项目管理代替了项目费用，进度等跟踪，以前的项目经验可以被应用。测试方面，使用了基本软件测试行为，例如测试计划和测试案例。
- 明确。通用管理和工程活动被标准化和文档化。测试开始之前，审查和证实测试文档和计划。测试与开发相互独立。测试结果用于确定软件发布时间。
- 可控。组织过程处于统计控制之下。产品质量事先由数量决定，软件达到目标之前不可以发布。开发过程中收集开发和测试资料，不断进行调整，使项目按计划进行。
- 不断优化。尝试新的技术和处理过程，评价结果，不断变革以期望达到质量更高的等级。

14.7. ISO 9000

国际标准化组织（ISO）为所有制造业设立标准。
他的目标在于开发过程，而不是产品。它关心的是进行工作的组织方式，而不是工作成果。质量是相对的，主观的。公司的目标应该是达到满足客户要求的质量等级。利用质量开发过程可望实现该目标。
ISO 9000 只决定过程的要求是什么，而不管如何达到。
ISO 9000 针对软件软件的部分是 ISO9001 和 ISO 9000-3。9001 负责处理设计、开发生产、安装和服务产品。ISO 9000-3 负责处理开发、供应安装和维护计算机软件。
ISO 9000-3 的要求如下：

- 开发详细的质量计划和程序控制配置管理、产品验证和合法性检查（测试），不规范行为（软件缺陷）和修正措施（修复）。
- 准备和接受软件开发证实，包括项目定义、产品目标清单、项目进度、产品说明书、如何组织项目的描述、风险和假设的讨论，以及控制策略。
- 使用客户容易理解、测试时容易进行合法性检查的用于来表述说明书。
- 计划、开发、编制和实施软件设计审查程序。
- 开发控制软件设计随着产品的生命周期中而变化的程序。
- 开发和编制软件测试计划。
- 开发检测软件是否满足客户要求的方法。
- 实施软件合法性检查和验收测试。
- 维护测试结果的记录。
- 控制调查研究和解决软件缺陷的方式。
- 证明产品在发布之前已经就绪。
- 开发控制产品发布过程的程序。
- 明确指出和规定应该收集何种质量信息。
- 使用统计技术分析软件开发过程。
- 使用统计技术评估产品质量。