ISTQB®软件测试人员认证

高级大纲
测试经理

2012 版

（中文版 20150528）

国际软件测试认证委员会

中文版的翻译编辑和出版统一由 ISTQB®授权的 CSTQB 负责

中文版本号：V3.3

英文版权声明

如果此文档的来源是公认的，则可以拷贝此完整的文档或部分。

中文版权声明

版本历史

版本	日期	说明
ISEB v1.1	2001 年 9 月 4 日	ISEB 从业人员大纲
ISTQB® 1.2E	2003 年 9 月	ISTQB® 高级大纲（EOQ-SG）
V2007	2007 年 10 月 12 日	认证高级测试工程师大纲 2007 版
D100626	2010 年 6 月 26 日	纳入 2009 年同意的修改，把每章的内容分至每个单独的模块。
D101227	2010 年 12 月 27 日	接受不影响语句含义的格式变化和修正。
D2011	2011 年 10 月 31 日	拆分大纲，修改学习目标和相应的文字。增加 BO。
Alpha 2012	2012 年 2 月 09 日	纳入了所有成员委员会对 10 月发布版本的意见。
Beta 2012	2012 年 3 月 26 日	纳入了所有成员委员会对 Alpha 版本的意见。
Beta 2012	2012 年 4 月 7 日	Beta 版提交 GA
Beta 2012	2012 年 6 月 8 日	正文编辑版发布给各成员委员会。
Beta 2012	2012 年 6 月 27 日	加入 EWG 和术语意见
RC 2012	2012 年 8 月 15 日	候选发布版 – 最终包括成员委员会的编辑
GA 2012	2012 年 10 月 19 日	最终编辑和整理版提交成员大会（GA）
中文发布版	2013 年 6 月 30 日	最终编辑和整理提交 CSTQB
中文更新版本 V3.0	2014 年 11 月 17 日	周震漪负责修改和完善，建立新版本 V3.0
中文更新版本 V3.3	2015 年 5 月 28 日	熊晓虹负责大纲与“软件测试专业术语中英文对照表 v2.4 修订版本 1”的一致性修订

目录

版本历史 3
目录 4
0. 课程大纲引言 7
0.1 目的 7
0.2 概述 7
0.3 预期学习目标 7
1. 测试过程 — 420 分钟. 8
1.1 简介 9
1.2 测试计划、监督和控制 9
1.2.1 测试计划 9
1.2.2 测试监督和控制 10
1.3 测试分析 11
1.4 测试设计 12
1.5 测试实施 12
1.6 测试执行 13
1.7 评估出口准则和报告 13
1.8 测试结束活动 14
2. 测试管理 – 750 分钟. 15
2.1 简介 17
2.2 测试管理的背景 17
2.2.1 了解测试干系人 17
2.2.2 其它的软件开发生命周期活动及工作产品 18
2.2.3 测试活动和其它生命周期活动的整合 18
2.2.4 管理非功能测试 20
2.2.5 管理基于经验的测试 20
2.3 基于风险的测试和其它测试优先级设定和工作量分配的方法 21
2.3.1 基于风险的测试 21
2.3.2 基于风险的测试技术 25
2.3.3 选择其它测试技术 27
2.3.4 测试过程中的测试优先级设定和工作量分配 28
2.4 测试文档和其它工作产品 28
2.4.1 测试方针 29
2.4.2 测试策略 29
2.4.3 主测试计划 31
2.4.4 级别测试计划 31
2.4.5 项目风险管理 32
2.4.6 其它的测试工作产品 32
2.5 测试估算 33
2.6 定义和使用测试度量 34
2.7 测试的商业价值 38
2.8 分布式测试、外包测试以及内包测试 39
2.9 管理行业标准的使用 39
3. 评审 – 180 分钟. 41
3.1 简介 42

3.2 管理评审和审计 43
3.3 对评审进行管理 43
3.4 评审度量 45
3.5 管理正式评审 45
4. 缺陷管理 – 150 分钟. 47
4.1 简介 48
4.2 缺陷生命周期和软件开发生命周期 48
4.2.1 缺陷工作流程和状态 48
4.2.2 管理无效和重复缺陷 49
4.2.3 跨职能缺陷管理 49
4.3 缺陷报告信息 49
4.4 使用缺陷报告信息评估过程能力 51
5. 改进测试过程 – 135 分钟. 52
5.1 简介 53
5.2 测试改进过程 53
5.2.1 过程改进的介绍 53
5.2.2 过程改进的类型 53
5.3 改进测试过程 54
5.4 使用 TMMi 改进测试过程 55
5.5 使用 TPI Next 改进测试过程 55
5.6 使用 CTP 改进测试过程 56
5.7 使用 STEP 改进测试过程 56
6. 测试工具及自动化 – 135 分钟 57
6.1 简介 58
6.2 选择工具 58
6.2.1 开源工具 58
6.2.2 定制工具 58
6.2.3 投资回报（ROI） 59
6.2.4 选择流程 60
6.3 工具生命周期 61
6.4 工具度量 62
7. 人员技能 – 团队构成 – 210 分钟. 63
7.1 简介 64
7.2 个人技能 64
7.3 测试团队动力 65
7.4 使测试适合组织 66
7.5 激励 67
7.6 沟通 68
8. 参考资料 69
8.1 标准 69
8.2 ISTQB® 文档 69
8.3 商标 70
8.4 书籍 70
8.5 其它引用 71
9. 索引 72

致谢
本文件由国际软件测试认证委员会负责高级测试经理模块的高级子工作组的核心团队于 2010 至 2012 年间编制，他们包括：Rex Black （主席）、Judy McKay（副主席）、Graham Bath、Debra Friedenberg、Bernard Homès、Paul Jorgensen、Kenji Onishi、Mike Smith、Geoff Thompson、Tsuyoshi Yumoto。

核心团队向评审团队和所有测试委员会成员提出的建议和帮助表示感谢。

直至高级课程大纲截稿时，其工作组成员如下（按姓氏字母排序）：
Graham Bath、Rex Black、Maria Clara Choucair、Debra Friedenberg、Bernard Homès（副主席）、Paul Jorgensen、Judy McKay、Jamie Mitchell、Thomas Mueller、Klaus Olsen、Kenji Onishi、Meile Posthuma、Eric Riou du Cosquer、Jan Sabak、Hans Schaefer、Mike Smith （主席）、Geoff Thompson、Erik van Veenendaal、Tsuyoshi Yumoto。

下列成员参与了评审、评论和大纲表决工作:

Chris van Bael、Graham Bath、Kimmo Hakala、Rob Hendriks、Marcel Kwakernaak、Rik Marselis、Don Mills、Gary Mogyorodi、Thomas Mueller、Ingvar Nordstrom、Katja Piroué、Miele Posthuma、Nathalie Rooseboom de Vries、Geoff Thompson、Jamil Wahbeh 和 Hans Weiberg
本文由 ISTQB®大会于 2012 年 10 月 19 日正式发布。

参加本大纲前期翻译的有（按姓氏拼音排序）：李雪、凌劲锋、王华、文燕、熊文杰、郑丹丹等。

参加本大纲最终翻译的 CSTQB 专家有（按姓氏拼音排序）：柴阿峰、陈耿、李硕、王轶、徐文叶、周震漪（组长）等。

0. 课程大纲引言

0.1 目的

本大纲根据国际软件测试认证委员会对高级大纲（测试经理）的要求进行编写，ISTQB®提供此大纲的主要目的：

1. 国家认证委员会需将大纲翻译成当地语言并分发给培训机构，并可根据特定语言对大纲进行适度润色、修改，以保证语句通顺可读。
2. 考试委员会可根据特定语言，按照高级大纲（测试经理）的学习目标设计考题。
3. 培训机构需根据高级大纲（测试经理）准备课程并选择最适宜的教学方法。
4. 需要认证的考生，根据高级大纲准备考试（作为培训课程的一部分或独立使用）。
5. 国际软件和系统工程领域，应以此大纲为基础，推进软件和系统测试蓬勃发展。并以此为基础著书和出文章。

0.2 概述

高级大纲由以下 3 个单独的部分组成：

- 测试经理
- 测试分析师
- 技术测试分析师

高级大纲的概述文档 [ISTQB®_AL_OVIEW] 包括以下信息：

- 每个大纲的学习成果
- 每个大纲摘要
- 各大纲之间的关系
- 认知程度描述 （K 级别）
- 附录

0.3 预期学习目标

预期学习目标是学习成果的前提保证，同时也被作为高级测试经理认证考试题目编写的基础。被标记为 K1 级的知识点，需要考生牢记、回忆、识别和认知。K2，K3，K4 级的知识点会在之后相关章节出现。

1. 测试过程 — 420 分钟.

关键词

出口准则（exit criteria），测试用例（test case），测试结束（test closure），测试条件（test condition），测试控制（test control），测试设计（test design），测试执行（test execution），测试实施（test implementation），测试日志（test log），测试计划（test planning），测试规程（test procedure），测试脚本（test script），测试总结报告（test summary report）

测试过程的学习目标

1.2 测试计划、监督和控制

TM-1.2.1 (K4) 为了计划测试活动和工作产品来实现测试目标，必须对一个系统的测试要求进行分析

1.3 测试分析

TM-1.3.1 (K3) 使用可追溯性来检查与测试目标、测试策略和测试计划相关的已定义测试条件的完整性和一致性

TM-1.3.2 (K2) 解释可能影响所规定测试条件详细程度的因素，以及详细规定测试条件的优点和缺点

1.4 测试设计

TM-1.4.1 (K3) 使用可追溯性来检查与已定义测试条件相关的所设计测试用例的完整性和一致性

1.5 测试实施

TM-1.5.1 (K3) 使用风险、优先级、测试环境和数据依赖以及限制条件来制定测试执行的进度，该进度针对测试目标、测试策略和测试计划是完整和一致的

1.6 测试执行

TM-1.6.1 (K3) 使用可追溯性来监督测试进展与测试目标、测试策略和测试计划的一致性和完整性

1.7 评估出口准则和报告

TM-1.7.1 (K2) 解释在测试过程中准确和及时信息收集的重要性，以便支持准确的报告和对照出口准则进行评价

1.8 测试结束活动

TM-1.8.1 (K2) 总结四组测试结束活动

TM-1.8.2 (K3) 实现项目回顾以评价过程和发现改进领域

1.1 简介

在 ISTQB® 软件测试基础级认证大纲中已描述了基本的测试过程包括以下活动：

- 计划和控制
- 分析和设计
- 实施和执行
- 评估出口准则和报告
- 测试结束活动

基础级大纲认同这些活动虽然有逻辑顺序，但过程中的某些活动可能重叠，或并行进行。

对于高级大纲，为了让该过程更为精练和优化、更适合软件开发生命周期以及促进有效的监督和控制，某些活动被视为独立的。所考虑的活动如下：

- 计划、监督和控制
- 分析
- 设计
- 实施
- 执行
- 评估出口准则和报告
- 测试结束活动

1.2 测试计划、监督和控制

本节将着重讨论测试计划、监督和控制过程。如基础级大纲所述，这些活动是测试管理的职责。

1.2.1 测试计划

每个测试级别的测试计划在这一级别的测试过程的初始阶段开始，并贯穿整个项目，直至这一级别的测试结束活动完成。计划包括识别满足测试策略中定义的任务和目标所需的活动和资源。测试计划也包括识别、收集和跟踪度量项的方法，这些度量项将用于指导项目、确定与计划的符合程度以及评价达成目标的情况。通过在计划阶段确定有用的度量项，可以有效的帮助选择工具、安排培训和建立文件指南。

为测试项目所选择的一个或多个策略有助于确定在计划阶段应进行的任务。例如当采用基于风险的测试策略（见第 2 章）时，使用风险分析可以指导测试计划过程中的风险缓解活动、可以降低识别出的产品风险、可以帮助制定风险的应急措施。如果发现多个可能的并且严重的与安全性相关的潜在缺陷，则应该花费大量精力开发和执行安全性测试。同样，如果发现设计规格说明有严重缺陷，则在测试计划过程中可安排对设计规格说明进行额外的静态测试（评审）。

也可以使用风险信息来确定不同测试活动的优先级。例如，当系统性能的风险较高，可在获得已集成的代码后尽快进行性能测试。类似的，如果采用了应对式测试策略，计划创建测试章程和动态测试技术（如探索性测试）的工具是必需的。

测试经理在测试计划阶段应清楚的定义测试方法，包括要采用的测试级别、每个级别的目标以及每个级别使用的测试技术。例如针对某一航空系统进行基于风险的测试时，风险评估规定了需要哪个级别的代码覆盖，及应采用的测试技术。

在测试依据（例如，特殊需求或风险）、测试条件和覆盖测试条件的测试之间可能存在复杂的关系，在这些工作产品之间也常常存在多对多的关联。需要理解这些关联，才能有效地进行测试计划、监督和控制。对于工具的决策可能也依赖于对工作产品之间关联的理解。

开发组和测试组生成的工作产品之间可能也存在关联。例如，可追溯性矩阵需要跟踪系统设计人员的详细设计规格说明的内容、业务分析员的商务需求以及测试组定义的测试工作产品之间的关系。如果要设计和使用详细测试用例，可在计划阶段规定以下要求：在开始生成测试用例前，开发组的详细设计文档要得到批准。在遵循敏捷的生命周期时，可以在测试开始之前采用非正式的信息传递来交流信息。

测试计划中还可能列出在此计划测试范围内的具体软件特征（适宜时基于风险分析），同时也确切地说明不在其测试范围内的特征。根据形式化程度的高低以及文档与项目的关联度，每个在范围内的特征都能关联到一个对应的测试设计规格说明。

在这个阶段，也可以要求测试经理与项目架构师共同来确定初始的测试环境规格说明、验证所需资源的可获取性，以确保要配置环境的人员有义务做这些工作，并且了解成本 / 交付时间表以及完成和交付测试环境所需的工作。

最后，应该识别所有的外部依赖关系和相关的服务级别协议（Service Level Agreements，SLA），如果需要，应与相关方进行初步接触。所谓的依赖关系包括对外部组织的资源请求、与其它项目的依赖关系（如果是在一个程序内的工作）、与外部供应商或合作开发伙伴的依赖关系、与其它部署团队以及数据库管理员的依赖关系。

1.2.2 测试监督和控制

为了让测试经理进行高效的测试控制，需要建立测试进度计划和监督框架以便对照计划跟踪测试工作产品和资源。此框架应该包括将测试工作产品的状态和活动与计划和战略目标相关联所需的详细的度量项和目标。

对于小型且不太复杂的项目，将测试工作产品和测试活动关联到测试计划和战略目标相对容易，但总体而言，为了达到这些目标，必须对目标进行详细定义，这包括为了满足测试目的和测试依据覆盖的措施和目标。

以可理解的并与项目和业务干系人相关的方式，将测试工作产品的状态和活动与测试依据关联起来是非常重要的。定义目标以及对测试条件和测试条件集进展的测量可以作为一种方法，通过测试条件实现其它测试工作产品与测试依据的关联。正确配置的可追溯性，包括通过可追溯性报告状态的能力，使得开发工作产品、测试依据和测试工作产品间存在的复杂关系更为透明和容易理解。

干系人要求监督的详细度量项和目标有时与系统功能或规格说明不直接相关，特别是没有正式文档或只有少量文档的时候。例如，即使是按系统功能定义了规格说明，业务干系人也许仍对按照运行业务周期建立的覆盖更感兴趣。业务干系人在项目早期的参与有助于确定这些度量项和目标，这不仅可在项目期间帮助提供更好的控制，还可在整个项目过程中帮助推动和影响测试活动。例如，干系人度量项和目标可影响测试设计的架构和测试实施工作产品和 / 或测试执行进度计划，以便按照这些度量项准确地监督测试进展。

测试控制是一个持续的行为，包括实际进度与测试计划之间的比较，在需要时实施纠正措施。测试控制引导测试工作来满足任务、策略和目标，包括在需要时再次进行的测试计划活动。对控制数据的相应反应依赖于详细计划的信息。

测试计划文档和测试控制活动的内容将在第 2 章中阐述。

1.3 测试分析

基础级大纲中对测试分析和设计合在一起进行了描述，而高级大纲虽然认同它们可以作为并行、整合或互动的活动来实现，以推动测试设计工作产品的生成，但在高级大纲中还是分别作为独立的活动来考虑。

测试分析是以测试条件的形式来确定要测试“什么”的活动。可通过分析测试依据、测试目标和产品风险来识别测试条件。它们（测试条件）可被视为详细的度量项和成功目标（如作为出口准则的一部分），应该可追溯到测试依据和规定战略目标，包括测试目标和其它项目或干系人成功的准则。测试条件应该随着测试设计和其它测试工作产品的生成而可以向前追溯。

在某个给定测试级别的测试依据一旦建立就可尽快进行该级别的测试分析。可采用正式的测试技术和其它通用分析技术（如基于风险的分析策略和基于需求的分析策略）来识别测试条件。测试条件可以取决于测试级别、在进行分析时可获得的信息以及所选择的详细程度（即文档编写的粒度）规定也可以不规定值或变量。

在决定描述测试条件的详细程度时，有以下一些因素要考虑：

- 测试级别
- 测试依据的详细程度和质量
- 系统 / 软件复杂度
- 项目和产品风险
- 测试依据间的关系，哪些是要测试的，以及如何进行测试
- 所使用的软件开发生命周期
- 采用的测试管理工具
- 对测试设计和其它测试工作产品进行说明和文档化的程度
- 测试分析人员所掌握的技术和知识
- 测试过程和组织自身的成熟度/成熟性（注意更高的成熟度/成熟性可能要求的详细程度更高，也可能接受低一些的详细程度）
- 能提供咨询的其它项目干系人的可用性

采用详细说明的测试条件可能会造成测试条件的数目较多。例如，你可能有一个概要测试条件，测试一个电子商务应用的“退出登录”。但在一份详细测试条件文档中，这个条件可能被分成多个测试条件，一个针对每一种受支持的付款方式，一个针对每个可能的目的地国家，等等。

详细说明测试条件的优点包括：

- 有助于增加将其它测试工作产品（如测试用例）与测试依据和测试目标相联系的灵活性，以便为测试经理提供更好和更详细的监督和控制
- 如基础级大纲所述，在一个较高级别的测试项目的早期，只要测试依据存在，并在系统架构和详细设计可用之前对测试条件进行详细说明有助于预防缺陷
- 以干系人能够理解的方式向其解释测试工作产品（测试用例和其它测试工作产品对于业务干系人而言常常毫无意义，简单的度量数据，如已执行的测试用例数对于干系人的覆盖要求而言也毫无意义）
- 有助于影响和指导其它测试活动，以及开发活动
- 能够更有效覆盖详细的度量项和目标，使得测试设计、实施和执行以及相应工作产品得到优化
- 为在某一测试级别内更清晰的横向可追溯性提供依据

详细说明测试条件的缺点包括：

- 有可能花费大量时间
- 在变更的环境中可维护性变得困难
- 流于形式并需要整个团队来定义和实施

在以下情况中详细定义测试条件会特别有效：

- 为了适应开发生命周期、成本和 / 或时间限制或其它因素，采用少量测试设计文档方法，例如检查表
- 没有或只有很少的正式需求或其它开发工作产品作为测试依据
- 项目的规模大、复杂且风险高，需要一定的监督和控制，而这样的监督和控制不能简单地通过将测试用例和开发工作产品相关联来实现

当测试依据可以容易且直接与测试设计工作产品相关联时，可用较少的细节来描述测试条件，如下列情况：

- 组件级别测试
- 较不复杂的项目，要测什么和如何测试之间存在简单的层次关系
- 可借助于用例来定义测试的验收测试

1.4 测试设计

测试设计是定义“如何”测试的活动。测试设计包括使用测试策略和 / 或测试计划中确认的测试技术，通过逐步阐述已识别的测试条件或测试依据来生成测试用例。

根据测试监督、控制和追溯的方法，可以将测试用例直接（或通过测试条件间接）与测试依据和规定的目标相关联。这些目标包括战略目标、测试目标和其它项目或干系人的成功准则。

按照规定的方法进行测试设计，在识别出测试条件和有足够信息后通过给定测试级别的测试设计来生成详细或概要测试用例。对于更高级别的测试，测试设计更象是在早期测试分析之后的一个单独活动，而对于低级别的测试，测试分析和设计很可能作为一个整合的活动来进行。

测试实施活动中某些任务也可能用迭代的方法与测试设计过程整合，以构建执行所需的测试，如测试数据的生成。事实上，这个方法在过程中可以优化测试条件的覆盖，生成详细或概要测试用例。

1.5 测试实施

测试实施是由测试分析师组织测试和设置测试优先次序的活动。在之前已经形成正式文档的情况下，测试实施是将测试设计实施为具体的测试用例、测试规程和测试数据的活动。某些遵循 IEEE829 标准的组织在测试用例规格说明中定义输入和相应的预期结果，在测试规程规格说明里定义测试步骤。更常见的情况是，每个测试的输入、预期结果和测试步骤都写在一起。测试实施还包括建立已存储的测试数据（如在平面文件或数据库表格中）。

测试实施还涉及进行最终的检查以确保测试组已准备好执行测试。检查可以包括确保提供所需的测试环境、测试数据和代码（可能运行某个测试环境和 / 或代码验收测试），所有测试用例已写好、经过评审、可以运行。还可能包括对照涉及的级别（见 1.7 章节）检查显性和隐形的入口准则。测试实施可能还涉及到测试环境和测试数据的详细描述。

在测试实施时所完成工作的详细程度和相关复杂性可能受到测试工作产品（如测试用例和测试条件）详细程度的影响。某些情况下，尤其是那些可以长期复用的回归测试，需要详细描述执行测试所需的步骤，以确保无论哪个测试人员都能可靠、一致地执行测试。如果法律法规需要，测试应该提供符合适用标准的证据。（见 2.9 章节）

在测试实施期间，测试执行进度计划内应该包括手动测试和自动化测试的顺序。测试经理应该仔细检查可能要求按特别顺序或在特别设备上运行测试的限制条件，包括风险和优先级，还必须了解和检查对测试环境或测试数据的依赖关系。

早期测试实施可能会有一些缺点。例如，如果使用敏捷生命周期，每次迭代的代码可能发生巨大的变化，造成大量的实施工作被浪费。即使不是敏捷这样易于变更的生命周期，任何迭代或增量生命周期都可能导致迭代之间发生重大变化，使脚本测试不可靠或有更高的维护需求。即使是顺序型生命周期，如果管理不当，需求频繁变更，甚至在项目晚期也出现需求变更，情况也会如此。在进行广泛测试实施工作之前，了解软件开发生命周期和可供测试的软件功能的可预测性是明智的。

早期测试实施可能也有一些优势。比如，具体的测试提供了一些工作例子，用来说明如果按测试依据所编写的内容，软件应该有何种行为。业务领域专家可能会发现，验证具体的测试比验证抽象的业务规则更容易，并可能由此进一步地发现软件规格说明中的不足。这样验证后的测试可能会为软件设计人员和开发人员对软件必须有的行为提供有启发性的说明。

1.6 测试执行

一旦提交了测试对象并且满足了测试执行的入口准则，就可以开始测试执行。测试应该在测试执行之前进行设计或至少进行定义。应该使用工具，特别是用于测试管理、缺陷跟踪和（如果适用）测试自动化执行工具。应该进行测试结果跟踪，包括度量项跟踪，所有组员都应该理解跟踪的数据，提供并公布测试日志和缺陷报告的标准模板。确保在测试执行之前就完成这些事项，就可以高效地开展测试执行活动。

应该按照测试用例执行测试，不过测试经理应该考虑允许测试人员有一定的自由度，以便测试人员在测试执行中能确保覆盖额外有趣的测试场景和观察有趣的行为。当使用的测试策略，至少是部分的使用应对性策略时，应该保留一些时间用于基于经验和基于缺陷的技术进行测试。当然，在这种不使用脚本测试的情况下，发现的任何失效必须描述再现失效所必需的测试用例，而这些新描述的测试用例往往会与已经现有的文档化测试用例有所不同。自动化测试完全是脚本化的测试，会毫无偏离地遵循规定好的指示。

在测试执行期间测试经理的主要职责是按照测试计划监督进展，如有需要，启动和执行控制措施以引导测试的任务、目标和策略朝着成功的方向发展。为了这样，测试经理可以使用从测试结果回到测试条件、测试依据和最终测试目标的追溯性，以及从测试目标向前到测试结果的追溯性。此过程在 2.6 章节中有详细描述。

1.7 评估出口准则和报告

2.6 章节详细讨论了测试进展监督和控制的文档和报告。

从测试过程角度来看，重要的是确保具备有效的过程来提供评价出口准则和报告所需的源信息。

所需信息和收集方法的定义是属于测试计划、监督和控制的一部分。在测试分析、测试设计、测试实施和测试执行的过程中，测试经理应该确保负责这些活动的测试组的成员准确和及时地提供所需的信息，以便推动有效的评价和报告。

报告的详细程度和频繁程度（次数）取决于项目和组织。这应该在测试计划阶段进行协商，包括与相关项目干系人沟通。

1.8 测试结束活动

当确定测试执行完成后，应当收集关键输出成果并且交给相应的人员或归档。这些活动统称为测试结束活动。测试结束活动分为 4 大类别：

1. 检查测试是否完成——确保所有的测试工作已经实际结束。例如，所有计划执行的测试应该已经被执行，或者被有意跳过不执行，所有已知的缺陷应该已经修复并经过确认测试、或被推迟到下一个发布版本、或接受作为系统的永久局限。
2. 交付测试工作产品——向需要软件产品的人员交付有价值的工作产品。例如，对于已经推迟或接受的已知缺陷，应该告知使用者并对他们使用系统进行支持。测试及测试环境应该转交给负责维护测试的人员。回归测试集（无论自动还是手动的）应该形成相应文件并转交给维护团队。
3. 总结经验教训——主持或参与回顾会议，会议中可记录（来自测试项目和整个软件开发生命周期中的）重要经验教训。在这些会议中，建立计划以确保能保持良好实践并避免不良实践，或者如果问题无法解决，则在项目计划中予以适应。待考虑的方面包括：
   a. 参加质量风险分析讨论会中的用户代表的范围是否足够广泛？例如，由于较晚才发现不曾预料的缺陷集群，在未来的项目中可以让更广泛的用户代表来参加项目的质量风险分析会。
   b. 估算是否准确？比如，实际开销与估算可能有严重误差，因而未来的估算活动需要考虑此问题及其根本原因，例如，是测试效率低还是估算结果确实低于应有的开销。
   c. 缺陷的趋势以及缺陷原因和影响分析的结果如何？例如，评估是否在后期与分析和开发的质量相关联的变更请求导致或看似一个不良实践的趋势。又例如跳过某一个测试级别，而这一测试级别本可以更早、以性价比更高、更节约时间的方式发现缺陷。检查缺陷趋势是否与新技术、人员变化和技能不足等因素有关。
   d. 是否有潜在的过程改进机会？
   e. 是否有未预期的与计划之间的偏差应该在今后的计划中予以调整？
4. 在配置管理系统中归档所有的结果、记录、报告和其它文档及工作产品。例如，测试计划和项目计划应该保存到计划的目录下，并明确指明这些文档对应的系统及版本。

以上这些任务非常重要但经常被疏忽。应该在测试计划中明确包括这些任务。

遗漏这些任务中的一项或多项是常见的事，通常是因为过早的重新分配或辞退项目成员、因为下一个项目的资源或进度的压力、因为团队疲劳等等。对于按照合同开展的项目，例如客户定制开发，合同中应该明确这些需要的任务。

2. 测试管理 – 750 分钟.

关键词

级别测试计划（level test plan）、主测试计划（master test plan）、产品风险（product risk）、项目风险（project risk）、质量风险（quality risk）、风险（risk）、风险分析（risk analysis）、风险评估（risk assessment）、风险识别（risk identification）、风险级别（risk level）、风险管理（risk management）、风险缓解（risk mitigation）、基于风险的测试（risk-based testing）、测试方法（test approach）、测试条件（test conditions）、测试控制（test control）、测试总监（test director）、测试估算（test estimation）、测试组长（test leader）、测试级别（test level）、测试管理（test management）、测试监督（test monitoring）、测试计划（test plan）、测试方针（test policy）、测试策略（test strategy）、宽带德尔菲法（Wide Band Delphi）

测试管理的学习目标

2.2 一定条件下的测试管理

TM-2.2.1 (K4) 分析软件项目或程序的干系人、环境、需求，包括软件开发生命周期模型，并识别最佳测试活动

TM-2.2.2 (K2) 理解软件开发生命周期中的活动和工作产品如何影响测试，以及测试如何影响软件开发生命周期中的活动和工作产品

TM-2.2.3 (K2) 解释在基于经验的测试和非功能性测试中对测试管理问题进行处置的方法。

2.3 基于风险的测试及其它测试优先级设定和工作量分配的方法

TM-2.3.1 (K2) 基于风险的测试及其它测试优先级设定和工作量分配的方法

TM-2.3.2 (K2) 举例说明产品风险分析的不同技术

TM-2.3.3 (K4) 分析、识别和评估产品质量风险，从关键项目干系人的角度总结风险及评估的风险级别

TM-2.3.4 (K2) 描述在生命周期和测试过程中，怎样根据评估的风险级别，适当地缓解和管理识别的产品质量风险

TM-2.3.5 (K2) 举例说明测试选择、优先级设定和分工的不同方案

2.4 测试文档和其它工作产品

TM-2.4.1 (K4) 分析给出的测试方针和测试策略，建立主测试计划、级别测试计划和其它与这些文档相补充和相一致的测试工作产品

TM-2.4.2 (K4) 针对给定的项目，分析项目风险并选择适当的风险管理方案（如缓解、应急、转移和/或接受）

TM-2.4.3 (K2) 描述并举例说明测试策略如何影响测试活动

TM-2.4.4 (K3) 制定适合组织、生命周期以及项目需要的测试工作产品的文档规范和模版，适时裁剪标准的主体部分，生成可用模版

2.5 测试估算

TM-2.5.1 (K3) 针对给定的项目，使用所有适当的估算技术创建整个测试过程活动的估算

TM-2.5.2 (K2) 理解可能影响测试估算的因素，并举例说明

2.6 定义及使用测试度量

TM-2.6.1 (K2) 描述并比较典型的测试相关度量

TM-2.6.2 (K2) 比较不同层面的测试进度监督
TM-2.6.3 (K4) 分析和汇报测试结果，主要包括遗留风险、缺陷状态、测试执行状态、测试覆盖状态以及信心来提供见解和建议，帮助项目干系人做出发布决策

2.7 测试的商业价值

TM-2.7.1 (K2) 分别给出决定质量成本四种类别的例子
TM-2.7.2 (K3) 基于质量成本，以及其它定性和定量的考虑，估算测试的价值，并将其告知测试干系人

2.8 分布式、外包及内部测试

TM-2.8.1 (K2) 理解如何成功运用不同人员策略构建团队：是利用分布式团队、外包团队还是内包试团队

2.9 管理行业标准的应用

TM-2.9.1 (K2) 总结软件测试标准的来源及使用

2.1 简介

在高级大纲中，测试从业人员开始划分专业方向。本章主要关注测试职业人员在晋升为测试组长、测试经理及测试总监时所需的知识领域。尽管不同的组织对这些职位和担任这些职位的人员的职责等级有不同界定，但在本大纲中，我们将所有这些专业测试人员统称为测试经理。

2.2 测试管理的背景

作为一名经理，最主要的职责是获取和利用资源（人力，软件，硬件，基础设施等）去开展能带来附加值的工作。对于软件和 IT 经理而言，这个过程通常是项目或程序中旨在（对内部用户或对外部用户）交付软件或系统的部分。对于测试经理而言，这个过程与测试相关，具体而言就是基础级大纲和本大纲第 1 章中描述的基础测试过程。由于测试过程的附加值只能通过整个项目或程序的成功（或通过防止更严重的失效发生）来体现，因此，测试经理必须相应地计划和控制测试过程。换句话说，测试经理必须根据其它干系人的需求、环境、干系人的活动（如进行测试的软件开发生命周期）、干系人的工作产品（如需求规格说明），适当地安排测试过程，包括相关活动及工作产品。

2.2.1 了解测试干系人

测试干系人是指与测试活动、测试工作产品、最终系统或交付物的质量有利益关系的人。干系人的利益可能是直接或间接地介入测试活动、直接或间接地接受测试工作产品，或者直接或间接地受到项目生产的交付物的质量的影响。

测试的干系人取决于项目、产品、组织或其它因素，可能包含以下角色：

- 开发人员、开发组长和开发经理：这些干系人需要部署待测的软件，获取测试结果，通常还必须根据测试结果采取一定行动（如修复报告的缺陷）
- 数据库架构师、系统架构师和设计师：这些干系人设计软件，获取测试结果，通常也必须根据测试结果采取一定行动
- 市场和业务分析师：这些干系人决定软件必须呈现的特性，以及这些特性固有的质量水平。他们通常还参与确定测试覆盖范围，评审测试结果，根据测试结果做出决策
- 高层管理人员、产品经理和项目发起人：这些干系人通常参与确定测试覆盖范围，评审测试结果，并根据测试结果做出决策
- 项目经理：这些干系人负责管理项目，使其取得成功，这就要求在质量、进度、特性和预算之间取得平衡。他们通常负责获取测试活动所需的资源，并协同测试经理进行测试计划和控制
- 技术支持、客户支持和帮助台人员：这些干系人支持用户和客户，同时也是已交付软件的特征与质量的受益人
- 直接和间接用户：这些干系人直接使用软件（即最终用户），或者接受由该软件生产或支持的输出或服务

更多关于测试干系人的信息可参见参考书籍 [Goucher09] 第 2 章。

以上的干系人列表并不全面。测试经理必须为自己的项目或程序识别具体的测试干系人。测试经理还必须理解干系人与测试关系的确切本质，以及测试团队怎样服务于这些干系人的需求。除了识别如上所述的测试干系人，测试经理还应识别其它影响测试和 / 或被测试影响的软件开发生命周期活动与工作产品。否则的话，测试过程可能无法达到最佳的有效性和效率（见 2.2.3 节）。

2.2.2 其它的软件开发生命周期活动及工作产品

软件测试评价的是非测试活动产生的一个或多个工作产品的质量，因此，测试通常存在于一系列软件开发生命周期活动的大背景下。测试经理必须在了解这些其它的活动及其工作产品怎样影响测试（如基础级大纲所述），并了解测试怎样影响这些其它的活动及其工作产品的情况下，计划和指导测试活动。

例如，在使用敏捷开发实践的组织，开发人员执行测试驱动的开发（TDD），生成自动化的单元测试，然后不断将代码（以及针对这些代码的测试）集成到配置管理系统中。测试经理应与开发经理一起确保将测试人员集成到了这些活动中，并与这些活动保持一致。测试人员可以通过评审单元测试，为提高覆盖率和测试活动的有效性给出建议，同时深入了解该软件及其实施情况。测试人员也可以评估将他们自己创建的自动化测试，尤其是功能回归测试集成到配置管理系统中的方式。[Crispin09]

尽管测试活动、其它测试干系人、软件开发生命周期活动和工作产品之间的具体关系取决于项目、选定的软件开发生命周期和许多其它的因素，但测试和以下活动相互关联密不可分：

- 需求工程和管理。在确定测试工作的范围和进行估算时，测试经理需要考虑到需求，并且对需求的变更保持高度的警惕性，在出现变更时，采取测试控制活动以适应这些变更。技术测试分析师和测试分析师应参与需求评审
- 项目管理。在测试分析师和技术测试分析师的协助下，测试经理必须将进度和资源需求提供给项目经理。测试经理必须与项目经理一起去理解项目计划的变更，并采取测试控制活动以适应这些变更
- 配置和变更管理。测试经理必须与测试团队一起确定测试对象交付流程和机制，并将这些记录在测试计划中。测试经理可以要求测试分析师和技术测试分析师生成构建验证测试，并在测试执行过程中确保版本控制
- 软件开发和维护。测试经理应与开发经理一起协调测试对象的交付，包括每次测试发布内容与发布日期，以及参与到缺陷管理中（见第 4 章）
- 技术支持。测试经理应与技术支持经理一起确保在测试结束时测试结果能顺利交付，以便让在产品发布后参与产品支持的人员知道产品有哪些失效以及为这些失效的变通方法，另外，测试经理应该与技术支持经理共同对产生的失效进行分析，以实施测试过程改进
- 编写技术文档。测试经理应与技术文档经理一起确保测试文档能按时交付，且这些文档中体现了对缺陷的管理

除了识别如上所述的测试干系人，测试经理还必须识别其它影响测试和 / 或被测试影响的软件开发生命周期活动与工作产品。否则的话，测试过程可能无法达到最佳的有效性和效率。

2.2.3 测试活动和其它生命周期活动的整合

无论项目使用的是哪个软件开发模型，测试都应是项目的一个主要部分。软件开发模型包括：

- 顺序模型，如瀑布模型、V 模型、W 模型：在顺序模型中，各阶段（如需求、设计、实施、单元测试、集成测试、系统测试和验收测试）的所有工作产品和活动都在下个阶段开始前完成。测试计划、测试分析、测试设计和测试实施与项目计划、业务 / 需求分析、软件和数据库设计、编程重叠进行，具体如何重叠则取决于该测试的级别。测试执行按照基础级大纲和本大纲中讨论的测试级别的顺序进行
- 迭代或增量模型，如快速应用开发（RAD）和统一开发过程（RUP）：在迭代或增量模型中，对待实现的特性进行分组（如根据业务优先级或风险），各项目阶段，包括其工作产品和活动，存在的目的就是实现各组特性。各阶段可能按顺序进行，也可能重叠进行，而迭代本身也可能是有序的或重叠的。项目启动期间，在项目计划和业务 / 需求分析的同时进行粗略的测试计划和测试分析。详细的测试计划、测试分析、测试设计和测试实施发生在每次迭代之初，重叠进行。测试执行通常包括重叠的测试级别。每个测试级别都尽早开始，在后续更高级别的测试开始后也可能会继续

- 敏捷，如 SCRUM 和极限编程（XP）：这些属于迭代生命周期模型，但迭代的周期很短（通常为二到四周）。每次迭代的活动和产品在下次迭代开始之前完成（如迭代是顺序性的）。测试进行的方式与迭代模型类似，但各种测试活动与开发活动之间都有很大程度的重叠，包括测试执行（各级别的）与开发活动之间。在一次迭代中的所有活动，包括测试，都应在下一次迭代开始之前完成。在敏捷项目中，测试经理的角色通常从直接的管理者转变成了技术权威 / 咨询顾问
- 螺旋：在螺旋模型中，项目早期使用原型来确认项目的可行性，试验设计和实施决策，根据业务优先级和技术风险选择原型试验开展的顺序。对这些原型进行测试以确认技术问题的哪些方面仍然没有解决。一旦解决了主要的技术问题，就可以按照顺序模型或迭代模型开展项目

为了能将测试活动适当地整合到生命周期内，测试经理必须对于组织使用的生命周期模型有详细的了解。例如，在 V 模型中，应用于系统测试级别的 ISTQB®基础测试过程可以按照下列方式整合：

- 系统测试计划活动与项目计划同时进行，测试控制一直持续到系统测试执行和测试结束工作完成
- 系统测试分析和设计与需求规格说明、系统和架构（概要）设计说明和组件（详细）设计说明的编写同时进行
- 系统测试实施活动可在系统设计期间开始，但大部分都是与编码和组件测试同时进行。系统测试相关的部署工作通常会延续到系统测试执行开始前的几天
- 当所有的系统测试入口准则被满足（或免除）时开始系统测试执行活动，系统测试入口准则一般是指至少完成了组件测试，通常还包括完成组件集成测试。系统测试会被持续执行直到系统测试出口准则被满足为止
- 整个系统测试执行期间都在评价系统测试出口准则和汇报系统测试结果，一般来说，当项目即将到达最后期限时评价和汇报的频率更高，也更紧迫
- 在满足了系统测试出口准则和正式宣布系统测试执行工作完成之后，进行系统测试结束活动，结束活动有时可以推迟到验收测试结束或所有项目活动完成后进行

在迭代或增量的生命周期模型中，必须开展同样的任务，但任务开展的时间以及程度可以不同。例如，与其在项目初期就部署整个测试环境，不如只部署本次迭代所需的部分可能更有效率。无论是迭代还是增量生命周期模型，计划得越深入，基础测试过程延伸的范围就越广。

除了项目的计划阶段，测试的执行和汇报也可能受到团队使用的生命周期的影响。例如，迭代生命周期中，在下一次迭代开始之前撰写完整的报告和进行迭代后评审会议可能是有效的。通过将每次迭代看作一个小型项目，项目组有机会根据上一次迭代中发生的事件做出纠正和调整。由于迭代周期可能很短，受到时间限制，缩短花在报告和评估上的时间是可以理解的，而任务的开展都应作为跟踪整个测试进展和尽可能快地识别问题区域的一种方式。如果没有采取纠正措施，在某次迭代中发生的问题很容易影响下一次迭代，甚至在下一次迭代中重现。

还有一些与如何将测试和其它生命周期活动整合的信息可能记录在测试策略章节中（见 2.4.2 节）。测试经理应在测试计划和 / 或项目计划期间，对各测试级别和选定的生命周期模型与测试过程的组合进行项目特定的整合。

根据组织、项目和产品的需求，可能需要补充基础级大纲未定义的测试级别，如：

- 硬件-软件集成测试
- 系统集成测试
- 特性交互测试
- 客户产品集成测试

各测试级别都应明确定义以下要素：

- 测试目标，可实现的目的
- 测试范围和测试项
- 测试依据，以及衡量该测试依据覆盖率的方式（如可追溯性）
- 入口和出口准则
- 测试交付物，包括结果汇报
- 适用的测试技术，以及与这些技术相应的确保覆盖度的方式
- 与测试目标、入口和出口准则以及结果汇报相关的度量和度量标准（包括覆盖率的度量）
- 针对具体的测试任务应用的测试工具（适用时）
- 资源（如测试环境）
- 测试团队内外部负责的个人和群体
- 符合组织、法规或其它规范（适用时）

最好的做法是在各测试级别对上述要素进行一致定义，以免不同级别的类似测试之间出现不必要的差距，造成危险和资源浪费。在本章后面部分还会继续讨论。

2.2.4 管理非功能测试

非功能测试计划失败会导致发布后发现严重的，有时是灾难性的系统质量问题。然而，很多非功能测试都是昂贵的，因此，测试经理必须依据风险选择需要执行的非功能测试。另外，非功能测试有许多类型，并非每一种测试类型都适用于给定的应用程序。

测试经理可能并没有充足的专业能力去处理各种计划的相关因素，因此，他们需要将非功能测试活动的部分测试计划责任委派给技术测试分析师（某些情况下是测试分析师）。测试经理应要求分析师考虑到以下一般因素：

- 干系人需求
- 所需的工具
- 测试环境
- 组织因素
- 安全性

更多内容参见高级技术测试分析师大纲第 4 章 [ISTQB® ATTA SYL]。

测试经理需要考虑的另一个重要因素就是怎样将非功能测试集成到软件开发生命周期。常见的错误是一直等到所有的功能测试都完成以后才开始进行非功能测试，这样做会导致太晚发现关键的非功能缺陷。相反，应该优先安排非功能测试，并根据风险对测试进行排序。通常在早期的测试级别，甚至是在开发期间，可以通过多种途径缓解非功能风险。例如，系统设计期间的用户界面原型的易用性评审对于识别重大的易用性缺陷十分有效，如果直到系统测试结束时才发现这些缺陷，则会严重影响项目进度。

在迭代生命周期中，变更和迭代的速度会让人难以关注需要构建复杂框架的非功能测试。那些比单次迭代周期花费的时间更长的测试设计和实施活动应单独安排成迭代之外的工作活动。

2.2.5 管理基于经验的测试

基于经验的测试能高效地找出其它测试技术可能遗漏的缺陷，正因为这个好处，它被用来检查其它测试技术的完整性，但这种技术也为测试管理带来了一些挑战。测试经理在知道基于经验的测试，尤其是探索性测试带来的好处的同时，也应该意识到其带来的挑战。由于一般采用轻量化文档，且通常测试事前准备得很少，因此很难评估这种测试技术达到的覆盖率。管理层需要特别留意测试结果的可重复性，尤其是当有多个测试人员参与测试时。

管理基于经验的测试，尤其是探索性测试的一种方式是将测试工作分解成小到 30-120 分钟的时间段，这些时间段有时被称为测试会话。这种时间段界定该会话中要做的工作，在此期间只关注这些工作，并且提供一定程度的监督和调度。每个测试会话覆盖既定的测试章程，此测试章程由测试经理以书面或口头的方式传达给测试人员。测试章程给出了测试会话中应覆盖的一个或多个测试条件，进一步帮助测试人员集中精力，避免在多人同时开展探索性测试时出现工作的重叠。

管理基于经验的测试另一种的方式是将自主和自发测试集成到更为传统的预先设计好的测试会话中。例如，测试人员获准（或被分配了时间）在他们预定义测试的明确步骤、输入和期待结果以外进行探索。测试人员可能还会被安排在运行一天的预定义测试之前、期间或之后，进行自主测试会话，作为每日测试的一个部分。如果测试会话发现了缺陷或值得进行后续测试的领域，可能会对提前编写的测试进行更新。

在每次探索性会话之初，测试人员确认并执行必要的测试启动任务。在会话期间，测试人员根据采用的测试技术设计和执行测试，查找缺陷，并将测试结果记录在日志中。（如果要求测试必须是可重复的，测试人员应该记录测试输入、操作及事件。）会话结束后，可能会有一个情况汇报，为后续测试会话确立方向。

2.3 基于风险的测试和其它测试优先级设定和工作量分配的方法

测试管理中普遍存在的一个挑战是测试的选择、分配和优先级设定，即测试团队必须在实际上无限的可覆盖的测试条件和测试条件的组合中选择一组有限的测试条件，确定适当的分工以确保各项条件都被测试用例覆盖，然后将生成的测试用例按照优先级排序，从而优化测试工作的有效性和效率。测试经理可以通过识别和分析风险以及其它因素，来帮助解决这个问题，但由于许多限制和其它变量的相互影响，可能导致折衷的解决方案。

2.3.1 基于风险的测试

风险是指负面或不希望发生的后果或事件发生的可能性。当引起客户、用户、参与者或干系人对产品质量或项目成功的信心减弱的问题可能发生时，风险就存在。当潜在问题主要影响的是产品质量时，它们被称为质量风险、产品风险或产品质量风险。而当潜在问题主要影响的是项目成功时，它们则被称为项目风险或计划风险。

在基于风险的测试中，在干系人参与的产品质量风险分析的过程中对质量风险进行识别和评估。测试团队设计、实施和执行测试来缓解质量风险。质量包括影响客户、用户和干系人满意度的全部功能、行为、特征和属性。因此，质量风险是产品可能存在质量问题的情况。系统质量风险的例子包括报告中的错误计算（与准确度相关的功能风险），用户输入响应慢（与效率和响应时间相关的非功能风险），界面和字段难以理解（与易用性和易理解性相关的非功能风险）。当测试找出了缺陷时，通过让他人意识到这些缺陷，并在发布前有机会处理这些缺陷，测试缓解了质量风险。当测试不再发现缺陷时，通过确保在测试条件下，系统可以正常运作，测试也缓解了质量风险。

基于风险的测试使用产品质量风险来选择测试条件，为这些条件分配测试工作，并为生成的测试用例设定优先级。基于风险的测试有各种各样的技术，这些技术在采集的文档的类型和级别，以及运用的形式方面大相径庭。基于风险的测试明确指出的或隐含的目的就是用测试来降低整体的质量风险水平，具体而言是把风险水平降低到可接受的范围。

基于风险的测试包含四个主要活动：

- 风险识别
- 风险评估

- 风险缓解
- 风险管理

这些活动之间部分重叠。后面的几节会讨论到每个活动。

要想取得最佳的效果，风险识别和评估小组应包括所有项目和产品干系人的代表，不过有时项目的实际情况导致某些干系人代理其它干系人。例如，在大众市场的软件开发中，可能抽样一小部分潜在客户，要求他们帮助识别潜在的严重影响他们使用软件的缺陷；在这个例子中，抽样的小部分潜在客户就是作为所有最终客户群的代理人。由于测试人员在产品质量风险和失效方面的专长，他们应该积极参与风险识别和评估过程。

2.3.1.1 风险识别

干系人可通过下列的一项或多项技术来识别风险：

- 专家咨询
- 独立评估
- 使用风险模版
- 项目回顾
- 风险研讨会
- 头脑风暴
- 检查表
- 参考过去的经验

对参与风险识别的干系人的抽样越广泛，风险识别过程就越有可能识别出越多的严重产品质量风险。

风险识别通常会产生一些副产品，例如，识别出并非产品质量风险的问题。例子包括有关产品或项目的一般疑问或问题，或参考的文档，如需求和设计说明中的问题。质量风险识别的另一副产品是识别出项目风险，不过项目风险并不是基于风险的测试关注的焦点。然而，项目风险管理对于包括基于风险测试在内的所有测试方法都是很重要的，这一点在 2.4 节中会进一步讨论。

2.3.1.2 风险评估

识别了风险后，就开始风险评估，即研究识别出的这些风险。具体而言，风险评估包括对每个风险进行分类，确定风险发生的概率和影响。风险评估还可能涉及评价或分配每个风险的其它属性，如风险责任人。

风险的分类就是将每个风险划分到适当的类型中，例如性能、可靠性、功能性等等。有些组织使用 ISO 9126（ISO 9126 标准正在被 ISO 25000 标准所取代）的质量特征进行分类，但大多数组织采用的都是其它的分类方案。风险分类时使用的检查表通常与风险识别使用的相同。有通用的质量风险检查表，许多企业都根据自己的需要定制这些检查表。如果使用检查表作为风险识别的基础，在风险识别时通常就进行了风险的分类。

确定风险的级别通常包括评估每条风险发生的可能性，以及发生之后的影响。发生的可能性是指被测系统存在潜在问题的概率。换句话说，可能性是对技术风险的级别的评估。影响产品和项目风险发生的可能性的因素包括：

- 技术和团队的复杂性
- 业务分析师、设计师和程序员的人员和培训问题
- 团队内部的冲突

- 与供应商的合同问题
- 异地工作的团队队员
- 旧方法与新方法对立
- 工具和技术
- 管理或技术领导不力
- 时间、资源、预算和管理压力
- 缺乏前期的质量保证活动
- 高变更率
- 早期的高缺陷率
- 接口和集成问题

风险发生的影响是指风险发生对用户、客户或其它干系人的影响的严重程度。换句话说，影响来自于业务风险。影响项目和产品风险的影响程度的因素包括：

- 使用受影响的特性的频率
- 该特性对于实现业务目标的关键程度
- 声誉的损害
- 商业损失
- 潜在的财务、生态、社会损失或法律责任
- 民事或刑事法律制裁
- 吊销执照
- 缺乏合理的变通
- 明显的失效导致负面声誉
- 安全性

风险的级别既可定性也可定量地评估。如果确定了可能性和影响的量化数据，可以将这两个值相乘，计算出定量的风险优先级数值。不过通常风险的级别只能通过定性的方式确认。也就是说，我们可以将可能性形容成很高、高、中、低或很低，但无法用精确的百分比表达出来；同样地，我们可以将影响形容成很高、高、中、低或很低，但也不能完整或精确地表达财务上的影响。不应该认为定性的方法比定量的方法差；实际上，如果风险级别的定量评估使用不当的话，会误导干系人对风险程度的理解和管理。风险级别的定性评估通常通过乘法或加法相结合，得到一个风险总分。这个风险总分可能被称为风险优先数，但确切来说，它应是定性的、有序的相对评级。

另外，除非风险分析基于广泛的、有效统计的风险数据，风险分析应该更多基于干系人对于可能性和影响的主观感知。通常，项目经理、编程人员、用户、业务分析师、架构师和测试人员的感知各不相同，因此针对各项风险的风险级别可能有不同的意见。风险分析过程应包括达成共识，或者即使在最糟的情况下，也要有确立都同意的风险级别的方式（如通过管理层下达命令或通过取该风险级别的平均值，中间值或众数值）。此外，还应检查风险级别的分布情况，确保风险的评分对于测试排序、优先级设定和分工能起到实际的指导作用。否则，将无法使用风险级别来指导风险缓解活动。

2.3.1.3 风险缓解

基于风险的测试从质量风险分析（识别和评估产品质量风险）开始。质量风险分析是主测试计划和其它测试计划的基础。正如计划中所述，测试的设计、实施和执行是为了覆盖风险。开发和执行测试的工作量与风险的级别成一定比例，也就是说风险的级别越高，使用的测试技术就越细致（如结对测试），风险的级别越低，使用的测试技术就越不细致（如等价类划分或有时限的探索性测试）。此外，开发和执行测试的优先级也是依据风险级别而设定。有些安全规范（如 FAA DO-178B/ED 12B,IEC 61508）规定了基于风险级别的测试技术和覆盖率。另外，风险的级别还应影响到一些决策，例如使用项目工作产品（包括测试）的评审、独立的级别、测试人员的经验水平、确认测试（再测试）和回归测试的程度。

在项目开展期间，测试团队仍应对改变质量风险集和 / 或已知质量风险的风险级别的补充信息保持了解。应该定期调整质量风险分析，至少在项目的主要里程碑到来时（进行这种调整），以此来指导对测试的调整。调整包括识别新风险，重新评估现有风险的级别和评价风险缓解活动的有效性。例如，如果在需求阶段，基于需求规格说明进行了一次风险识别和评估，那么当设计规格说明定案后，就应对已有的需求规格风险重新进行评价。再如，如果在测试中发现某个组件包含的缺陷要远比预期的缺陷数多，则可得出在此区域缺陷的可能性比预期的要高的结论，并将风险的可能性和整体风险级别上调，根据调整结果就应增加该组件测试的工作量。

在测试执行开始之前也可以缓解产品质量风险。例如，如果在风险识别时发现了需求的问题，项目组就可以通过实施全面的需求规格说明评审来缓解这一风险。这样做可以降低风险的级别，可能意味着缓解残余质量风险所需的测试数目减少。

2.3.1.4 生命周期中的风险管理

理想情况下，在整个生命周期期间都在进行风险管理。如果组织有测试方针文档和 / 或测试策略文档，这些文档中应当描述测试中管理产品风险和项目风险的一般过程，以及风险管理怎样集成到测试的各个阶段中，并使其发挥作用。

一个成熟的组织，风险意识应遍及整个项目组，在这样的组织中，风险管理并非仅仅发生在测试中，而是发生在各个方面。重要的风险不仅在特定测试级别的前期就得到处理，而且在早期的测试级别中也得到了处理。例如，如果性能被识别为一个关键质量风险区域，性能测试在系统测试的前期就会开始进行，而且在单元测试和集成测试时也会运行性能测试。成熟的组织不仅识别风险，还识别风险的来源和风险一旦发生将带来的后果。对于那些确实发生的风险，使用根本原因分析来深入理解风险的来源和实施过程改进，防范于未然。风险缓解贯穿于整个软件开发生命周期。风险分析获得了充足的信息，考虑到了相关的工作活动、系统行为分析、基于成本的风险评估、产品风险分析、最终用户风险分析和责任风险分析。当测试团队参与并对整个程序范围的风险分析产生影响的情况下，风险分析超越了测试的范畴。

大多数基于风险的测试方法还包括用风险级别来对测试进行排序和优先级设定的技术，以此确保测试执行时尽早覆盖最多的重要区域，发现最多的重要缺陷。某些情况下，所有的高级别风险的测试都是在较低级别风险测试之前进行的，且测试的运行严格按照风险排序（通常称为“深度优先” ）；其它情况下，测试优先级根据抽样方法来制定。抽样方法就是选择一个测试样本，该测试样本可以覆盖到所有已识别的风险。用风险覆盖率评估选出的测试样本，确保每条风险至少被覆盖到一次（通常称为“广度优先” ）。

无论基于风险的测试是深度优先还是广度优先，分配给测试的时间都有可能不足。基于风险的测试使得测试人员能以剩余风险级别的方式向管理层报告当前测试状态，让管理层决定是否要增加测试时间或将剩余的风险转移给用户 / 客户、服务 / 技术支持人员和 / 或运营人员。

在测试执行期间，最先进的基于风险的测试技术（并不一定是最正式或重量级的技术）让项目参与人、项目和产品经理、高层行政管理人员和项目干系人根据剩余风险级别监督和控制软件开发生命周期，包括做出是否发布的决策。这就要求测试经理在汇报有关风险的测试结果时，要确保所有干系人都能理解。

2.3.2 基于风险的测试技术

基于风险的测试技术有很多，其中有一些很不正式，例如测试人员在探索性测试期间分析质量风险的方法 [Whittaker09]。尽管这种方法能有助于指导测试，但会导致过分关注缺陷的可能性，而不是关注缺陷的影响，也没有引入跨职能干系人的输入。这些方法还很主观，依赖于单个测试人员的能力、经验和偏好。正因如此，这些方法很少可以取得基于风险的测试的全部收益。

为了尝试以最小的成本获得基于风险的测试的收益，许多人采用了轻量级的基于风险的方法。这种方法融合了非正式方法的响应能力和灵活性以及正式方法的力度和一致性。轻量级方法的例子包括：实用风险分析与管理（PRAM）[Black09]、系统的软件测试（SST）[Craig02]和产品风险管理（PRisMa）[vanVeenendaal12]。除了基于风险的测试的常见属性外，这些技术通常还有以下属性：

- 从基于风险的测试行业经验中逐渐演变而来，特别是那些效率至关重要的行业
- 取决于跨职能组的，代表业务类和技术类观点的干系人，在初期风险识别和评估中的广泛参与
- 在一个项目的最初阶段引入缓解质量风险时效果最佳，风险分析的主要工作产品和副产品有助于影响产品的规格说明和实施，从而最大程度地减少风险
- 使用生成的输出结果（风险矩阵或风险分析表）作为测试计划和测试条件，以及所有后续测试管理和分析活动的基础
- 支持将关于剩余风险的测试结果汇报给各级别的测试干系人

这类技术中有些（如 SST）要求将需求规格说明作为风险分析的输入，而且只有在提供了需求规格说明后才能使用。其它技术（如 PRisMa 和 PRAM）鼓励使用将基于风险与基于需求混合的策略，使用需求规格说明和 / 或其它规格说明作为风险分析的输入，但功能可以完全基于干系人的输入。将需求作为输入可以帮助确保对需求的高覆盖率，但测试经理务必要确保不漏测在需求中没有提到的重要风险，尤其是非功能领域的风险。如果得到的输入是好的并设定了优先级的需求，通常我们可以看到风险级别和需求优先级之间的密切关联。

这些技术中有许多都鼓励将风险识别和评估过程作为干系人之间就测试方法达成共识的一种方式。这会带来巨大的好处，但同时也需要占用干系人的时间去参与小组的头脑风暴会议或是一对一的访谈。干系人参与的不足会导致风险分析中的差距。当然，干系人并不一定对每个风险级别意见一致，所以主持质量风险分析工作的人必须积极且有创造性地与干系人一起达到可能的最大程度的一致。一个专门主持评审会议的主持人具有的所有技能都适用于主持质量风险分析的人员。

与正式技术一样，轻量级的技术允许对可能性和影响因素进行加权，以强调业务或技术风险（如取决于测试级别）。与正式技术不一样的轻量级技术：

- 仅使用两个因素，可能性和影响程度；并且
- 使用简单的，定性的判断和尺度

这些方法的轻量性质在诸多领域带来了灵活性和可用性，以及适合整个团队的各种经验和技能水平的人员（甚至是非技术人员和新人）。

在正式的，重量级的那一端，测试经理有许多可用的选项：

- 危害分析，延伸至整个分析过程的上游，尝试识别每个风险背后隐藏的危害
- 一个危害的成本。对每个质量风险，风险评估过程包括三个因素：1）与该风险项相关的失效的可能性（用百分比表示）；2）一旦生产期间发生与该风险项相关的典型失效将造成的失效成本（用金融数量表示）；3）这些失效的测试成本
- 失效模式和影响分析（FMEA）和其变体 [Stamatis03]，识别质量风险，风险的潜在原因和可能造成的影响，然后确定严重程度，分配优先级和检测等级

- 质量功能展开（QFD），一种影响测试的质量风险管理技术，具体而言，关注的是对客户或用户需求的理解不当或不充分引起的质量风险
- 故障树分析（FTA），对各种实际发现的失效（从测试或生产中）或潜在失效（质量风险）进行根本原因分析，首先分析会导致失效的缺陷，其次是会导致这些缺陷的（人为）错误或（其它）缺陷，一直继续直到找出根本原因

基于风险的测试采用的具体技术和怎样采用该技术取决于项目、过程和产品。例如，类似 Whittaker 的探索性技术的非正式方法可能适用于补丁或是快速修复。在敏捷项目中，质量风险分析被完全集成到每个冲刺（Sprint）的初期，风险的文档也融入了用户故事跟踪。综合系统要求不仅对单个系统，对整个综合系统也有风险分析。安全关键系统和关键任务项目要求的正式程度和文档的级别相对较高。

基于风险的测试的输入、过程和输出取决于选定的测试技术。常见的输入包括干系人见解、规格说明和历史数据。常见的过程包括风险识别、风险评估和风险控制。常见的输出包括质量风险列表（附带各风险的风险级别和建议的测试工作量分配）、规格说明等输入文档中发现的缺陷、关系到风险项的疑问或问题，以及影响测试或项目整体的项目风险。

通常，影响基于风险的测试成功与否的最关键因素是有正确的干系人参与风险识别和评估。干系人对于产品质量由什么构成都有各自的理解，而且他们都有着自己的一套风险的优先级和关注点。但他们还是倾向于分成两大类：业务干系人和技术干系人。

业务干系人包括客户、用户、运营人员、帮助台和技术支持人员。这些干系人了解客户和用户，因此可以从业务的角度识别风险和评估影响。

技术干系人包括开发人员、架构师、数据库管理员和网络管理员。这些干系人知道潜在的软件失效方式，因此可以从技术层面识别风险和评估可能性。

某些干系人既代表业务角度的观点也代表技术角度的观点。比如说，担任测试分析或业务分析角色的行业专家，由于其在技术和业务领域的专业知识，一般会有更广阔的视角。

识别风险项的过程就是生成一张充实的风险列表的过程。干系人没有必要争论风险项的内容，只要有一个干系人认为是系统质量的风险，那么它就是风险项。重要的是干系人必须对风险级别的评定达成共识。例如，在使用可能性和影响程度作为评定因素的轻量级方法中，过程必须包括找到对于可能性和影响程度的通用的分级方案。包括测试组在内的所有干系人必须使用同样的尺度，且必须能覆盖每个质量风险项的可能性和影响。

另外，如果要长期使用基于风险的测试，那么测试经理必须要成功地向干系人倡导基于风险的测试。要想持续使用这个技术，跨职能组必须要看到风险分析的价值。这就要求测试经理理解干系人的需要、期望和参与过程的可用时间。

干系人适当参与质量风险分析过程能让测试经理获益匪浅。对于需求缺失或描述不明确的项目，干系人仍然可以在合适的检查表的指引下识别风险。在实施了基于风险的测试后，当测试团队的缺陷检测有效性提高时，我们就可以看到干系人参与所带来的收益。之所以这样，是由于使用了更全面的测试依据，此处是使用了质量风险项列表。

在结束基于风险的测试时，测试团队应衡量他们的效益。许多情况下，这关系到回答以下需要通过度量和与团队咨询的部分或全部 4 个问题：

- 测试团队发现的重要缺陷是否比次要缺陷要多？
- 大部分的重要缺陷是否是测试团队在测试执行的初期发现的？

- 测试团队能否就风险向干系人解释测试结果？
- 测试团队跳过的测试（如果有的话）是否比执行的测试的相关风险级别要低？

大多数情况下，基于风险的测试只要成功，以上四个问题答案都是肯定的。长期来看，测试经理应该为这些度量标准设定过程改进目标，并积极提升质量风险分析过程的效率。当然，其它的度量标准和成功标准也可以用于基于风险的测试，测试经理应仔细考虑这些度量标准之间的关系，测试团队奋斗的战略目标，以及管理这些特定度量标准和成功标准集的使用需要的行为。

2.3.3 选择其它测试技术

在许多测试经理使用基于风险的测试作为他们测试策略的一个要素的同时，也有许多测试经理还选择了包括其它技术。

其中一种最流行的开发测试条件并设定优先级的替代技术就是基于需求的测试。基于需求的测试可以使用多种技术，如歧义评审、测试条件分析和因果图。歧义评审识别和消除需求文档（作为测试依据）中的歧义，通常是通过使用常见需求缺陷的检查表（参见 [Weigers03]）。

如 [Craig02] 中所述，分析测试条件包括仔细阅读需求规格说明，以识别需要覆盖的测试条件。如果事先已经对需求分配了优先级，这些需求的优先级就可以用来分配工作和设定测试用例优先级。如果没有，又不把基于需求的测试融入基于风险的测试的话，就难以确定适当的工作分配和测试的先后顺序。

在高级测试分析师模块中，谈到覆盖测试条件的组合作为测试分析的一部分时，讲到了因果图。因果图其实有更广泛的用途，它可以将极大的测试问题归纳为可控数目的测试用例，并且仍然能保证测试依据的功能覆盖达到 100%。因果图还可以在设计测试用例时识别出测试依据中的分歧，如果在需求草案阶段就开始设计测试，就可以在软件开发生命周期的早期发现缺陷。采用因果图的一大难题是画图的复杂度。因果图太过复杂，很难人工完成，现在有许多工具可以代劳。

基于需求的测试的一个普遍障碍是需求规格说明模糊不清、可测试性差、不完整或根本不存在。并非所有的组织都会积极解决这些问题，如果面临这种情况，测试人员必须另选测试策略。

有时另一种方法也会用到现有需求，那就是通过基于模型的方法，创建使用概况或运行概况，综合使用测试用例、用户（有时称作人物角色）、输入和输出，以准确地展现该系统在现实中的应用。这么做不仅测试了功能性，同时还测试了易用性、互操作性、可靠性、安全性和性能。

在测试分析与计划期间，测试团队识别使用概况并尝试用测试用例覆盖这些概况。使用概况是一种基于可用信息，对软件现实应用的估计。也就是说，和基于风险的测试一样，使用概况可能无法完全展现最终的真实情况。不过，如果有充足的信息和干系人输入的话，那么展现的情况就是恰当的（参见 [Musa04]）。

有些测试经理还会采用检查表等有条理的方法来决定要测试什么，测试多少，以怎样的顺序测试。在产品非常稳定的情况下，一张列出了所有待测的主要功能和非功能区域的检查表，加上现有的测试用例积累便足够了。检查表通常以发生变更的类型和数量为基础，对测试分工和测试排序有启发作用。但这种方法只在测试小变更时比较有效。

最后，另一种常用的方法就是应对型方法。在应对型方法中，测试执行之前做的测试工作很少。测试团队关注的是应对实际交付的产品。当发现了缺陷群时，缺陷群就成了后续测试的焦点。测试优先级设定

和分工不断变化。应对型方法可以作为其它方法的补充，但如果单独使用的话，它往往会漏掉程序中的一些至关重要的区域，仅仅因为（在这些区域中）没有发现大量的缺陷。

2.3.4 测试过程中的测试优先级设定和工作量分配

无论测试经理采用的是哪种技术——或者，更好的做法是混合使用一些技术——测试经理都必须将该技术融入项目及测试过程中。例如，在顺序生命周期（如 V 模型）中测试团队在需求阶段选择测试、分配测试工作量，并开始设定测试的优先级，然后再进行定期调整，但迭代或敏捷生命周期则要求逐次迭代的方法。测试计划和控制必须考虑到风险、需求、使用概况、以及做出相应反应的程度。

在测试分析、设计和实施期间，必须采用在测试计划阶段做的测试分工和设定的优先级。仅仅因为没有使用该信息来指导后续的测试过程，会造成测试过程中的细致分析和 / 或建模的普遍障碍。这个障碍通常发生在设计和实施阶段。

在执行测试时，还是应该使用在计划阶段设定的优先级，不过有一点很重要，在计划制定后要根据获取的信息定期更新优先级。在评估和汇报测试结果和出口准则状态时，测试经理也必须从风险、需求、使用概况、检查表和其它用于测试选择和优先级设定的指导这些方面评估和汇报。必要时，需根据测试优先级方案对测试进行分类。

作为结果汇报和出口准则评估的一部分，测试经理可以衡量测试完成的程度，应包括将测试用例和发现的缺陷追溯到相关的测试依据。例如，在基于风险的测试中，随着测试运行发现缺陷，测试人员可以检测残留的、剩余风险级别。这么做可以在使用基于风险的测试时知道发布的正确时机。测试汇报应提到覆盖到的，仍未关闭的风险，以及已获得或还未获得的收益。基于风险覆盖汇报测试结果的例子，参见 [Black03]。

最后，在测试结束期间，测试经理应评估与所有测试干系人，包括客户和用户的质量方面的需求和期待相关的度量标准和成功准则。只有在测试满足了这些需求和期待时，测试团队才可以称得上是真正有效的团队。

2.4 测试文档和其它工作产品

文档编制通常是测试管理活动的一部分。在不同的组织和项目中尽管测试管理文档的具体名称和每个文档的范围往往不一致，但组织和项目中常用的测试管理文档有以下类型：

- 测试方针：描述组织的测试目标和目的
- 测试策略：描述组织中通用的、不按项目而变化的测试方法
- 主测试计划（或项目测试计划）：描述一个特定项目中测试策略的实施情况
- 级别测试计划（或阶段测试计划）：描述在每个测试级别中所要进行的特定活动

这些类型的文档的物理存在形式在不同的环境下可能不同。在某些组织和项目中可能合并成一个文档，有时可能分成多个文档，而有时则可能宣称这些文档中的某些内容是凭直觉去撰写的、某些内容可能被忽略、或者将未修改的传统测试方法写入文档。较大、较正式的组织和项目倾向于撰写上述所有类型的文档，并将其作为书面的工作产品予以保存，而较小、不太规范的组织和项目通常只会撰写其中的一部分。虽然使用何种类型的文档取决于组织和项目，但是本大纲将分别描述每一个文档类型。

2.4.1 测试方针

测试方针描述了组织为什么要进行测试。它定义了组织期望达到的总体测试目标。测试方针应由组织中的高级测试管理人员，在测试干系人组的高层经理的协助下制定。

在某些情况下，测试方针是更广泛的质量方针的一部分或补充。质量方针描述质量管理的整体价值和目标。

测试方针可能是一份简短的、概要的文档：

- 总结组织驱动测试的价值
- 定义测试目标，如建立对软件的信心，检测软件的缺陷，降低质量风险的级别（参见 2.3.1 节）
- 描述如何评估测试满足这些目标的有效性和效率
- 列出典型的测试过程，可能用 ISTQB®基础测试过程作为基础
- 说明组织将如何改进其测试过程（参见第 5 章）

测试方针应可针对新开发项目的测试活动，同时也针对维护项目的测试活动。测试方针也可以指定整个组织要使用的测试术语和测试工作产品的内部和 / 或外部参考规范。

2.4.2 测试策略

测试策略文档描述组织的通用测试方法，包括用测试管理产品和项目风险、测试级别的划分和与测试相关的概要活动的方式。（同一个组织在不同情况下可能有不同的策略，例如不同的软件开发生命周期，不同的风险级别，或不同的规定要求。）测试策略及其描述的测试过程和测试活动应与测试方针一致。它应该为组织或一个或多个项目提供通用测试入口和出口准则。

如上所述，测试策略描述一般测试方法，通常包括：

- 分析型策略，如基于风险的测试。在这种情况下，测试团队分析测试依据，以识别要覆盖的测试条件。例如，在基于需求的测试中，通过测试分析从需求中导出测试条件，再设计和实施测试来覆盖这些条件。然后执行测试，通常用每个测试覆盖的需求的优先级来决定测试运行的顺序。从需求状态方面汇报测试结果，如通过了测试的需求，测试了需求但未通过，还未完全测试需求，需求测试受阻，等等
- 基于模型的策略，如基于运行概况测试。在这种情况下，测试团队开发一个模型，该模型展现了系统存在的环境（基于实际或期望的情况），系统应该接受的输入和系统运行的条件，以及系统应如何运行。例如，在针对高速发展的移动设备应用程序的基于模型的性能测试中，基于当前的使用情况和项目随着时间推移的发展情况，可能会开发出展现往来的网络通信量、活跃和不活跃的用户以及造成的处理负载的模型。另外，也可能开发出当前生产环境的硬件、软件、数据容量、网络和基础设施的模型。还可能开发出理想的、预期的和最小的吞吐率、响应时间和资源分配的模型
- 系统化的策略，如基于质量特征的测试。在这种情况下，测试团队使用既定的一套测试条件，如测试规范（如 ISO 9126）（ISO 9126 标准正在被 ISO 25000 标准所取代），有关特定领域、应用或测试类型（如安全性测试）的一般化的逻辑测试条件的检查表或集合，每个迭代或每次发布期间使用这一套测试条件。例如，在对一个简单的，稳定的电子商务网站进行维护测试时，测试人员可能会用检查表来识别每一页的关键功能、属性和环节。测试人员会在每次修改该网站时都会覆盖检查表上的相关要素
- 符合过程或标准的策略，如受制于美国食品药品管理局（US FDA）标准的医疗系统的测试。在这种情况下，测试团队遵循一系列由标准委员会或其它专家组制定的过程，这些过程使用文档

描述了如何适当的识别和使用测试依据和测试基准，以及组建测试团队。例如，在遵循敏捷管理技术的项目中，每次迭代期间测试人员分析描述特定特性的用户故事，作为该迭代的计划过程的一部分，估算测试每个特性所需的工作量，识别每个用户故事的测试条件（通常称为“验收准则”），执行覆盖这些条件的测试。执行测试时，将每个用户故事的状态（未测试、测试失败或测试通过）汇报给团队组长（通常称为“Scrum Master”）

- 应对型策略，如缺陷攻击测试。在这种情况下，测试团队在收到了软件后才开始设计和实施测试，对待测的实际系统作出反应。例如，在对基于菜单的个人电脑应用程序进行探索性测试时，可能会开发出一套与特性、菜单选择和屏幕对应的测试章程。每个测试人员都分到了一套测试章程，然后使用这些章程来组织探索性测试的会话。测试人员定期向测试经理汇报测试会话的结果，测试经理再根据发现的结果修改测试章程
- 咨询式策略，如用户导向的测试。在这种情况下，测试团队根据一个或多个关键干系人的输入决定要覆盖的测试条件。例如，在对基于网站的应用程序进行外包兼容测试时，公司可能会提供外包的测试服务供应商一份浏览器版本、反病毒软件、操作系统、连接类型和其它配置项的优先级清单，他们想要根据这份清单对照自己的应用软件进行评估。然后测试服务供应商就可以用诸如结对测试（针对高优先级项）的测试技术和等价类划分（针对低优先级项）来生成测试
- 面向回归测试策略，如广泛采用自动化来进行测试。在这种情况下，测试团队使用各种技术来管理回归风险，尤其是一个或多级别的功能和／或非功能回归测试自动化。例如，如果对一个基于网站的应用程序进行回归测试，测试人员可以使用基于 GUI 的测试自动化工具来自动化该应用的一般和特殊用例。然后在每次修改该应用程序时执行这些测试

不同的测试策略可以相互组合。所选的测试策略应符合组织的需要和方法，并可根据其特定的业务或项目进行修改。

测试策略可描述要执行的测试级别。在这种情况下，应给出制定每个测试级别的入口准则和出口准则的概要指导以及这些测试级别之间的关系（例如，不同的测试级别对应不同的测试覆盖目标）。

测试策略也可包括如下方面的内容：

- 集成方法
- 测试规约说明技术
- 测试独立性（可根据不同的测试级别而有所变化）
- 必需的和可选的应遵守的标准
- 测试环境
- 测试自动化
- 测试工具
- 软件工作产品和测试工作产品的可重用性。
- 确认测试（再测试）和回归测试
- 测试控制和测试汇报
- 测试度量和度量标准
- 缺陷管理
- 测试件的配置管理方法
- 角色和职责

长期和短期的测试策略可能有必要区分开来。不同的测试策略适合不同的组织和项目。例如，对于安全关键的应用，应使用比其它应用更精确详细的测试策略。此外，测试策略还因开发模型不同而异。

2.4.3 主测试计划

主测试计划描述如何在特定的项目中所有待进行的测试工作，包括要执行哪几个测试级别，以及这些级别之间的关系和测试级别与对应的开发活动之间的关系。主测试计划应说明测试人员如何在这个项目中实施测试策略（即如何选择测试方法）。主测试计划应与测试方针和测试策略相一致，若有不一致，则应解释为何会出现偏差和例外，以及该偏差可能导致的潜在影响。例如，如果组织的测试策略是在一个不变的系统即将发布之前进行一次完全通过的回归测试，但当前的项目没有回归测试，那么它的测试计划就应说明这么做的原因和将采取哪些行动缓解由与常用策略的偏差带来的风险。测试计划还应说明预期的由该偏差导致的影响。例如，跳过回归测试可能要求在项目初次发布一个月后安排一次维护发布。主测试计划描述了较大项目或业务中的测试工作，所以它应作为项目计划或业务指南的补充。

主测试计划的内容和结构根据组织、组织的文档标准、项目形式的不同而变化，通常都会有以下部分：

- 测试项和不测试项
- 测试的质量属性和不测试的质量属性
- 测试进度安排和预算（应根据项目预算或组织运营预算调整）
- 测试执行周期及其与软件发布计划的关系
- 测试团队与其它人或部门之间的关系和交付内容
- 对于每个测试级别定义要测试项和不测试项
- 每个测试级别的明确的入口准则、延续（暂停 / 继续）准则和出口准则，以及这些测试级别之间的关系
- 测试项目的风险
- 测试工作的整体治理
- 执行每个测试级别的职责
- 每个测试级别的输入和输出

在一些较小的项目或业务中，若只设定一个正式的测试级别，那么可将主测试计划和该级别的测试计划合并成一个文档。例如，如果系统测试作为唯一的正式测试级别，非正式的组件测试和集成测试由开发人员执行，非正式的验收测试将作为 Beta 测试的一部分由客户执行。在这种情况下，该系统测试计划可包含本节中提到的内容。

另外，测试通常依赖于项目中的其它活动。若这些活动在其文档中未充分定义，尤其是它们与测试的关系和对测试的影响，那么在主测试计划（或相应的级别测试计划）中应包含这些内容。例如，若没有定义配置管理过程，则在测试计划中应说明如何将测试对象交付给测试团队。

2.4.4 级别测试计划

级别测试计划描述在每个测试级别中所要进行的特定活动或测试类型。级别测试计划拓展了主测试计划中的特定级别或类型。它包含了进度安排、任务和里程碑等在主测试计划中不一定包含的细节。另外，它还应包括该级别的测试规格说明应遵守的标准和使用的模板。

在不那么正式的项目或业务中，通常只有一份级别测试计划作为测试管理文档。在这种情况下，该级别测试计划可包含本节前面所提到的要素。

对于敏捷项目，Sprint 测试计划或迭代测试计划可以替代级别测试计划。

2.4.5 项目风险管理

适当的项目计划的一个重要部分就是处理风险。项目风险的识别可以使用 2.3 节中所述的类似的过程。当识别出风险后，就应该告知项目经理，再由项目经理制定对策。这种风险的缓解并不总是测试组织力所能及的。然而，有些风险是可以而且应该由测试经理成功缓解的，如：

- 测试环境和工具的就绪
- 测试人员的可用性和能力
- 缺乏测试工作的标准、规则和技术

管理项目风险的方法包括尽早准备测试件、预测试的测试环境，预测试产品的早期版本，使用更严格的测试入口准则，加强需求的可测性、参与项目早期工作产品的评审、参与对变更的管理、以及监督项目的进展和质量。

在识别和分析了项目风险后，管理风险有以下四种方案：

1. 缓解风险，通过预防性措施降低风险的可能性和 / 或影响
2. 制定应急计划，降低风险成为事实后的影响
3. 将风险转移给其它方来处理
4. 忽略或接受风险

最佳方案的选择取决于各方案带来的利益和机会，需花费的成本，以及和它们相关的潜在的任何额外风险。如果为某个项目风险确定了应急计划，最佳实践是同时确定应急计划的触发时间（以此确定触发应急计划的时机和方式）和所有者（开展应急计划的人）。

2.4.6 其它的测试工作产品

测试会生成许多其它的工作产品，如缺陷报告、测试用例规格说明和测试日志。其中大部分都是由测试分析师和技术测试分析师编制。在测试分析师大纲中，讨论了生成工作产品和编制相关文档的考虑因素。测试经理应该确保这些工作产品在下列活动中的一致性与质量：

- 建立并度量，以监督这些工作产品的质量，如被拒绝的缺陷报告的比例
- 与测试分析师、技术测试分析师一起选择和（根据需要）定制工作产品的适当模版
- 与测试分析师、技术测试分析师一起建立工作产品的标准，如测试、日志和报告需要详细到什么程度
- 选择适当的参与人和干系人并使用适当的技术对工作产品进行评审

测试文档的范围、类型和特性主要取决于选定的软件开发生命周期、可用的标准和法规和开发的特定系统的产品质量和项目风险。

测试工作产品的模版有许多来源，如 IEEE 829。测试经理要记住 IEEE 829 标准适用于任何行业。因此，该模版中包括了很多非常详细的内容，其中有一些可能并不适用于特定组织。最佳实践是对 IEEE 829 进行裁剪，为特定的组织创建自己的标准模版。模版的一致应用会减少培训需求并且帮助组织内部统一过程。

测试还会生成测试结果报告，这些报告通常由测试经理编制，本章稍后将介绍测试结果报告。

2.5 测试估算

作为一种管理活动，估算是为了得到特定业务或项目中各种活动的成本和完成日期的近似目标。最好的估算：

- 代表有经验的实际工作者的集体智慧，并能得到相关参与人员的支持
- 提供具体详尽的，关于资金、资源、任务和相关人员的清单
- 指出每个被估算活动的最可能的成本、工作量和周期

尽管为估算建立了项目管理最佳实践，但软件工程和系统工程的估算依旧充满了困难，包括技术上的和政治上的困难。估算就是将最佳实践应用到与项目或业务有关的测试活动中。

测试估算应包含第一章所描述的测试过程中的所有活动。由于测试执行通常在项目的关键路径上，所以管理人员会对测试执行的成本、工作量、特别是测试执行持续时间的估算值非常感兴趣。然而，当软件的整体质量很差或未知时，对测试执行进行估算往往会比较困难，并且估算的结果也不可靠。此外，对系统的熟悉程度和经验水平也很可能影响估算的质量。常用的做法是估算需要执行的测试用例数，但这仅仅适用于当可以假定待测软件中的缺陷较少时。在估算过程中所作的假定应记录在估算的文档中。

测试估算应考虑影响测试活动的成本、工作量和持续时间等所有因素。这些因素包括（但不仅限于此）：

- 系统所要求的质量级别
- 待测系统的规模
- 以前测试项目的历史数据，也可包括行业数据或其它组织的基准数据
- 过程因素，包括测试策略、开发或维护生命周期和过程成熟度/成熟性，以及项目估算的准确度
- 物质因素，包括测试自动化和工具、测试环境、测试数据、开发环境、项目文档（如需求文档、设计文档等）、以及可复用的测试工作产品
- 人员因素，包括经理和技术组长、高级行政管理人员的承诺和期望、项目团队的技能、经验和态度、项目团队的稳定性、团队成员之间的关系、测试和调试环境的支持、是否有熟练的承包商和顾问、以及相关领域的知识
- 过程、技术和组织的复杂度、测试干系人的人数、子团队的组成和位置
- 项目启动、培训和定向需求
- 新工具、技术、定制硬件、大量测试件的熟悉和开发
- 非常详细的测试规约说明的要求，特别是要符合不熟悉的文档标准
- 难以确定的组件交付时间，特别是对于集成测试和测试开发而言
- 敏感的测试数据（如对时间敏感的数据）

交付的待测软件的质量也是测试经理在估算时应该考虑的一个主要方面。例如，如果开发人员已经采用了最佳实践，如自动化的单元测试和持续集成，那么在将代码交付给测试团队之前，多达 50%的缺陷就已经被清除了（关于这些实践的缺陷清除有效性的更多内容，参见 [Jones11] ）。一些报道称使用敏捷方法，包括测试驱动开发，交付的待测件质量较高。

估算可以自底向上进行，也可以自顶向下进行。在测试估算中可以单独或组合使用的技术有下列几种：

- 直觉、猜测和以往经验
- 工作分解结构（WBS）
- 团队估算会议（如宽带德尔菲法）
- 公司标准和规范
- 测试占整个项目工作量的百分比或人员编制（例如，测试人员与开发人员的比例）

- 组织的历史和度量，包括基于度量的模型，该模型用于估算缺陷数、测试周期数、测试用例数、每个测试的平均工作量以及回归周期的数目
- 工业平均值和预估模型，例如功能点、代码行数、估计的开发工作量或其它项目参数（如参见 [Pol01] ）

大多数情况下，得出估算值后，需要提交给管理人员，并说明理由（请参见 3.7 节）。通常，随后进行协商，协商的结果常常是需要重新进行估算。理想情况下，最终的测试估算代表了组织和项目目标在质量、进度安排、预算以及特性方面的最佳平衡。

要记住任何估算结果都是基于进行估算时可用的信息。项目早期信息可能比较有限，而且在项目早期已经可用的信息后面也可能发生变化。为了保持准确性，应该更新估算结果来反映新的信息和变更过后的信息。

2.6 定义和使用测试度量

管理上有这样一句名言，进行度量的工作才会得到有效的执行。反之，因为很容易忽略那些不进行度量的工作，所以不进行度量的工作通常不会得到有效的执行。因此，对于包括测试在内的任何活动，建立适当的度量都是很重要的。

测试度量可以划分到以下的一种或多种类型中：

- 项目度量，对照既定的项目出口准则，如测试用例执行率、通过率和失败率，度量项目进展
- 产品度量，度量产品的某些属性，如测试程度和缺陷密度
- 过程度量，度量测试或开发过程的能力，如通过测试发现的缺陷百分比
- 人员度量，度量个人或小组的能力，如在给定的时间内测试用例的实施情况

任何给定的度量都属于以上的两种、三种、甚至四种类型。例如，体现每日缺陷发现率的趋势图可以与以下内容都相关：出口准则（连续一周都未发现新缺陷）、产品质量（测试无法再找出产品的缺陷）、和测试过程的能力（在执行测试的前期发现了大量缺陷）。

人员度量尤为敏感。测试经理有时会把过程度量误认为是人员度量，导致大家为了让该度量对他们更有利而采取一些行动，产生灾难性的后果。本大纲第 7 章和专家级大纲“测试管理”模块中讨论了对测试人员的适当激励和评估。

在高级大纲中，我们主要关注的是使用度量来衡量测试工作的进展，如项目度量。某些用于衡量测试进展的项目度量也关系到产品和过程。在专家级大纲“测试管理”模块中描述了更多关于管理产品和过程度量的使用的内容。在专家级大纲“改进测试过程”模块中描述了更多关于使用过程度量的内容。

度量的使用可以让测试人员在汇报结果时保持一致，而且可以连贯地跟踪测试进展。测试经理通常被要求在各种会议上展示度量数据，这些会议的参与人可能包括从技术人员到执行管理层的各级别的干系人。因为有时确定一个项目整体成功与否也会用到这些度量数据，在决定需要跟踪什么内容、汇报的频率和呈现这些信息的方式时都需要特别留意。需要强调的是，测试经理必须考虑以下内容：

- 度量的定义：应定义一组有限的有用度量。度量的定义依据项目、过程和 / 或产品的具体目标。定义度量时应考虑到平衡，因为单个的度量可能会误导对状态或趋势的印象。对这些定义的度量的解读必须得到所有干系人的认可，避免讨论这些度量时产生混乱。经常发生定义了过多的度量而没有关注那些最相关的度量的情况

- 度量的追踪：应该尽可能自动化度量报告和汇总，以缩短采集和处理度量数据的时间。随时间的推移，特定的度量数据可能会反映出和度量定义阶段约定的解读不同的信息。测试经理应做好准备，仔细分析这些度量数据和期待可能出现哪些偏差，以及造成偏差的原因
- 度量的汇报：目的是帮助管理层迅速理解所获得的信息。应呈现对某段时间度量的“快照”或度量随时间推移的变化，这样才能进行趋势分析
- 度量的有效性：测试经理还必须验证汇报的信息。为某个度量收集的数据可能无法反应项目的真实情况或可能传达了过于乐观或过于悲观的趋势。在呈现数据前，测试经理必须就数据的准确度和可能传达的信息两方面对数据进行评审

监督测试进展主要就以下五个方面：

- 产品（质量）风险
- 缺陷
- 测试
- 覆盖率
- 信心

在项目和业务中，产品风险、缺陷、测试和覆盖率可以，且通常以特定的方式进行度量和汇报。如果这些度量数据和测试计划中定义的出口准则相关，他们可以作为判断测试工作是否完成的客观标准。信心的度量可以通过调查或使用覆盖率作为替代度量，不过通常还会以主观的方式汇报信心。

与产品风险相关的度量包括：

- 完全覆盖的风险百分比（所有的测试都通过（Pass））
- 部分覆盖的风险的百分比（有些测试或很多测试都没有通过）
- 还未完全测试的风险的百分比（有些测试还没有测试完）
- 按风险类别划分的覆盖的风险百分比
- 在初次质量风险分析后识别的风险的百分比

与缺陷相关的度量包括：

- 已报告（发现）的缺陷总数对比已解决（修复）的缺陷总数
- 失效的平均时间间隔和失效出现率
- 按下列分类统计的缺陷数或百分比
  - 特定的测试项或组件
  - 根本原因
  - 缺陷来源（如需求规格说明、新功能、回归等）
  - 测试发布
  - 引入、发现和移除缺陷的阶段
  - 优先级 / 严重程度
  - 拒绝或重复的缺陷报告
- 从报告缺陷到修复缺陷所花的时间趋势
- 引入了新缺陷（有时也称子缺陷）的缺陷修复数

和测试相关的度量包括：

- 已计划的、已详细说明（已实施）的、已运行、通过的、失败的、无法执行的和跳过不执行的测试总数
- 回归测试和确认测试的状态，包括趋势和未通过的回归测试总数及未通过的确认测试总数
- 计划的每日测试时长对比实际的每日测试时长
- 测试环境的可用性（准备测试团队可用的测试环境占计划测试时长的百分比）

和测试覆盖率相关的度量包括：
- 需求和设计要素的覆盖率
- 风险覆盖率
- 环境 / 配置覆盖率
- 代码覆盖率

重要的是测试经理要知道怎样去解读和使用覆盖率的度量，以便理解和报告测试状态。对于级别较高的测试，如系统测试、系统集成测试和验收测试，主要的测试依据通常是需求规格说明、设计规格说明、用例、用户故事、产品风险、支持环境和支持配置等工作产品。结构化的代码覆盖率度量更适用于级别较低的测试，如单元测试（如语句和分支覆盖）和组件集成测试（如接口覆盖）。测试经理可能使用代码覆盖的度量数据来衡量测试覆盖待测系统的程度，但在汇报较高级别的测试结果时，通常不会提到代码覆盖的度量。此外，测试经理应该知道即便单元测试和组件集成测试达到了结构覆盖目标的 100%，缺陷和质量风险仍有待较高级别的测试来处理。

度量也可以连系到基本测试过程中的活动（在基础级大纲和本大纲中有描述）。这样一来，在整个测试过程中，就可以对照项目目标和测试过程本身，使用度量数据来监督测试过程本身以及达成项目目标的进展。

和监督测试计划和控制活动相关的度量包括：
- 风险、需求和其它测试依据要素的覆盖率
- 缺陷发现情况
- 计划开发测试件和执行测试用例的时长对比实际的时长

和监督测试分析活动相关的度量包括：
- 识别的测试条件数
- 测试分析中发现的缺陷数（如通过使用测试依据识别风险或其它测试条件）

和监督测试设计活动相关的度量包括：
- 测试用例覆盖的测试条件百分比
- 测试设计中发现的缺陷数（如通过对照测试依据开发测试）

和监督测试实施活动相关的度量包括：
- 测试环境配置的百分比
- 测试数据记录加载的百分比
- 测试用例自动化的百分比

和监督测试执行活动相关的度量包括：
- 执行、通过和失败的测试占已计划的测试的百分比
- 执行（和 / 或通过）的测试用例覆盖的测试条件的百分比
- 计划与实际的报告 / 解决的缺陷对比
- 计划与实际达到的覆盖率的对比

监督测试进展和测试完成活动的度量包括里程碑、入口准则和出口准则（测试计划中定义和批准的）的映射，其中可能包括以下内容：

- 计划的测试条件、测试用例或测试规约说明的数目和按测试是否通过分别统计的执行的测试条件、测试用例或测试规约说明的数目
- 总缺陷数，通常按严重程度、优先级、目前状态、受影响的子系统或其它分类统计（见第 4 章）
- 要求的、接受的、开展的和测试过的变更数
- 计划成本对比实际成本
- 计划工期对比实际工期
- 测试里程碑的计划日期对比实际日期
- 有关测试的项目里程碑（如代码冻结）的计划日期对比实际日期
- 产品（质量）风险状态、通常按已缓解与未缓解的风险，主要的风险区域、测试分析后发现的新风险等分类统计
- 由于阻塞事件或计划的变更导致的测试工作量、成本或时间损失的百分比
- 确认和回归测试状态

和监督测试结束活动相关的度量包括：

- 测试执行期间已执行的、通过的、失败的、无法执行的和跳过不执行的测试用例的百分比
- 纳入可复用的测试用例库的测试用例的百分比
- 自动化的测试用例的百分比或计划的与实际的自动化的测试用例百分比对比
- 并入回归测试的测试用例的百分比
- 已解决 / 未解决的显著缺陷报告的百分比
- 识别和归档的测试工作产品的百分比

另外，标准的项目管理技术，如工作分解结构，通常被用来监督测试过程。在敏捷团队中，测试是燃尽图上用户故事进展的一部分。使用精益管理技术时，测试进展以一系列故事为基础，通常通过用户故事卡在看板图上移动的状态来监督。

在给定了一组度量标准后，度量数据可以通过口头陈述、在表格中以数值的形式，或用图形来进行汇报。度量数据可以有很多用途，包括：

- 分析，找出可从测试结果中观察到的趋势和原因
- 汇报，将测试结果告知感兴趣的项目参与人和项目干系人
- 控制，改变整个测试或项目的进程和监督进程纠正的结果

收集、分析和报告这些测试度量数据的适当方式取决于具体的信息需要、目标和使用这些度量数据的个人能力。另外，测试报告的具体内容也应该根据不同的读者而变化。

为了测试控制的需要，非常重要的一点是度量数据必须能够提供给测试经理有关整个测试过程（测试计划完成后）的信息，并能指导测试经理成功完成测试任务、实施测试策略和实现测试目标。因此在计划时一定要考虑到信息需要，监督时一定要包括收集任何需要的工作产品度量。需要的信息量和采集信息需要的工作量取决于各种项目因素，包括项目规模、复杂度和风险。

测试控制一定要回应测试产生的信息和项目或活动存在的不断变化的环境。例如，如果动态测试在某些认为不可能有很多缺陷的区域发现了缺陷群，又如由于测试开始时间延迟导致测试执行周期缩短，则必

须对风险分析和计划作出修改。这么做可能需要对测试优先级重新设定和对剩余的测试执行工作重新进行分配。

如果通过测试进展报告发现与测试计划出现了偏差，则应执行测试控制。测试控制的目的是为项目和 / 或测试重新定向到更可能取得成功的方向。当项目的控制工作取决于测试结果或受测试结果影响时，需要考虑以下内容：

- 修改质量风险分析、测试优先级和 / 或测试计划
- 增加资源或增加项目或测试工作量
- 推迟发布日期
- 放松或加强测试出口准则
- 改变项目的范围（功能或非功能）

实施这些内容通常需要项目或业务干系人之间达成共识，并且取得项目或业务经理的同意。

测试报告中发布的信息应该大部分取决于目标读者，如项目管理人员或业务管理人员的信息需要。项目经理最可能感兴趣的是关于缺陷的详细信息，而业务经理最关注的可能是产品风险的状态。

2.7 测试的商业价值

测试经理必须努力优化测试，以确保发挥好的商业价值。过量测试无法发挥好的商业价值，因为测试强加的不合理的延时和成本要比通过测试节约的时间和成本更高。测试不足也无法发挥好的商业价值，因为交付给用户太多的缺陷。最佳的方案介于两者之间。测试经理一定要帮助测试干系人理解这个最佳方案和测试发挥的价值。

虽然大多数组织认为测试在某种意义上是有价值的，但很少有管理人员，包括测试经理，可以量化、描述或明确界定测试的价值。另外，许多测试经理、测试组长和测试人员关注测试战术上的细节（测试任务或测试级别方面），却忽略了与测试相关的更大的战略上（更高级别）的问题，而这些问题恰恰是其它项目参与者，尤其是管理人员关注的部分。

测试为组织、项目和 / 或业务带来了定量和定性的价值：

- 定量的价值，包括发现缺陷并在产品发布前预防或修复这些缺陷、发现缺陷并了解在产品发布前依旧存在的缺陷（未修复但有相关文档记录，且可能采取了变通方法）、通过运行测试减少风险并发布了有关项目、过程和产品状态的信息
- 定性的价值，包括提高产品质量的声誉、使发布更顺利和更可预测、增加对产品的信心、降低产品功能失效甚至造成人员伤亡的可能性，避免承担法律责任

测试经理应知道哪些价值对他们的组织、项目和 / 业务有意义，并能在沟通测试情况时将测试与这些价值联系起来。

一种用于度量测试的定量价值和效率的成熟方法称为质量成本（或有时称为不良质量成本）。质量成本方法将项目和运行的成本分成与产品缺陷成本相关的四个类别：

- 缺陷预防成本，例如培训开发人员，提高他们编写的代码的可维护性和安全性
- 缺陷检测成本，例如编写测试用例，配置测试环境和评审需求
- 内部失效成本，例如在发布之前，测试或评审期间，修复发现的缺陷
- 外部失效成本，例如将有缺陷的软件发布给客户导致的支持成本

部分测试预算是用于支付缺陷检测成本（即便在测试人员没有找到缺陷的情况下也会花费的成本，比如开发测试花费的钱），而剩余的用于支付内部失效成本（与找到的缺陷相关的实际成本）。缺陷检测成本和内部失效成本的总和通常会比外部失效成本低很多，而这正是测试的价值所在。测试经理可通过列出这四类成本来作一个有说服力的商业用例，以说明测试的价值。

更多关于测试的商业价值，包括质量成本的内容参见 [Black03]。

2.8 分布式测试、外包测试以及内包测试

很多时候，有些，甚至可能所有的测试工作都是由身处异地的人员完成的，这些人为不同的公司工作，或完全独立于项目团队之外。如果测试工作分散在不同地点，则被称为分布式测试。如果承担测试工作的人员与开发团队不同属一个公司，同时测试工作又是在非开发地点进行的，则被称为外包测试。若测试工作由其它与项目团队在同一地点的非同公司成员承担，则被称为内包测试。

对于上述所有类型的测试，需要明确团队之间的沟通方式，并明确定义关于使命、任务和交付内容的期待。项目团队必须减少对非正式交流渠道的依赖，如在走廊上的对话或同事之间的私人交流。定义应有的沟通方式是很重要的，定义的范围应该包括处理问题升级等内容，通过交流要传达的信息类型和沟通的方式。团队各方干系人都必须清楚知道自己以及他人的角色和职责，避免产生误解或不切实际的期待。地点、时区、文化和语言的差异使得这些工作变得更加重要。

对于上述所有类型的测试，另一个共同点是各个团队要使用一致的方法。尽管在任何项目中都可能发生方法不一致的情况，但在分布式和 / 或由其它公司执行测试时，这种情况发生的可能性更高。如果测试小组之间或测试小组、开发小组、管理人员之间使用各自不同的方法，那么可能造成严重的问题，尤其是在测试执行期间。 例如，如果客户使用敏捷开发，而测试服务供应商预先定义的测试方法是假设使用顺序生命周期，那么测试件交付给供应商的时间以及性质就会成为双方争论的焦点。

对于分布式测试，不同工作地点之间的测试分工必须要明确、合理。如果不这么做的话，可能导致把测试工作分配给不具备相应能力的测试小组。另外，如果每个小组不知道自己负责哪部分工作的话，那他们可能就不会做他们本该做的事。对于各小组的期待必须要传达清楚。如果缺乏细致的管理，整个测试工作可能有遗漏（增加产品发布的残余质量风险）和重叠（导致降低效率）的困扰。

最后，对于各种类型的测试，整个项目团队建立并维持对测试团队的信任非常重要。要相信各个测试团队虽然在组织、文化、语言和地理位置上有所不同，但他们依旧能很好地履行他们的职责。缺少信任会导致低效和延迟。缺少信任会引起验证彼此的活动、相互追究问题的责任以及上演组织政治斗争这类问题，进而造成低效和延期。

2.9 管理行业标准的使用

在基础级大纲和高级大纲中，都参考了许多标准。这些参考标准涉及到软件开发生命周期、软件测试、软件质量特征、评审和缺陷管理。测试经理应该知道这些标准，所在组织对使用这些标准的方针，以及他们是否必须或有必要使用这些标准，或使用标准是否有帮助。

标准可以有不同的来源，例如：

- 国际标准或具有国际性目标的标准
- 国家标准，如国际标准在某个国家的国内版
- 专业领域特定的标准，如对国际标准或国家标准进行裁剪，以应用到特定的领域，或是为某个专门的领域开发的国际或国家标准

国际标准机构包括 ISO 和 IEEE。ISO 是国际标准组织，又称 IOS，国际标准化组织。国际标准化组织是由各国标准化团体（ISO 成员团体）组成的世界性的联合会，它们为软件测试人员提供了许多有用的标准，例如 ISO 9126（将被 ISO 25000 取代）、ISO 12207 和 ISO 15504。

IEEE 是电气与电子工程师协会，它是一家被一百多个国家承认的美国专业组织。该组织同样为软件测试人员提供了许多有用的标准，例如 IEEE 829 和 IEEE 1028。

许多国家都有自己的国家标准，其中的某些标准可用于软件测试。例如英国的 BS-7925-2:1998，该标准提供了很多关于在高级测试分析师和高级技术测试分析师大纲中描述的测试设计技术的信息。

有些标准是专门用于特定行业领域的，其中的某些标准可用于软件测试、软件质量和软件开发。例如，在航空电子领域，美国联邦航空管理局的标准 DO-178B（等同于欧盟的 ED 12B）适用于民用飞机中使用的所有软件。该标准规定了根据待测软件的关键程度的不同，相应级别的结构化覆盖标准。

另一个领域特定标准的例子是医疗体系中的美国食品与药品管理局的第 820 章 21 条的内容。该标准引荐了一系列结构化测试技术和功能测试技术，同时还引荐了一系列与 ISTQB®大纲相一致的测试策略和原则。

有些时候，标准和方法并不专注于测试领域，但它们对于测试所处的软件开发上下文影响却十分重大。这里有个例子，CMMI® 软件过程改进框架，其中包含了两个重要的过程域，验证和确认。对验证和确认的解读通常是他们指的是测试的不同级别（如系统测试和验收测试）。验证和确认还涉及到测试策略，通常被解读成要求在测试策略中纳入分析型基于需求的测试。

还有几个重要的例子，PMI 的 PMBOK、PRINCE2® 和 ITIL®。PMI 和 PRINCE 2 是北美和欧洲使用较广泛的项目管理框架。ITIL® 是用于确保 IT 团队能向其所在的组织交付有价值的服务的一个框架。在这些框架中提到的术语和活动与 ISTQB®大纲和术语表中提到的有很大区别。如果所在的组织使用 PMI 的 PMBOK、PRINCE2®和 / 或 ITIL，测试经理必须充分了解选定的框架、框架的实施和相关术语，以便在该组织中能有效地工作。

无论采用的是什么标准或方法，记住这些方法和标准都是由一组专业人员制定的。标准体现了这些专业人员的集体的经验和智慧，但也体现了他们的不足。测试经理应该了解他们所处的环境和背景中采用的标准，无论是正式标准（国际标准、国家标准或行业特定标准）或是组织自身的标准和推荐实践。

在考虑使用多个标准时，记住某些标准可能和其它标准不一致，或甚至出现定义上的冲突。测试经理应该决定不同标准在测试所处的上下文中的用处。某个标准中阐述的信息可能会对项目有正向作用，也可能会起到反作用。然而，通过这些标准，可以参考已证实的最佳实践，并且提供组织测试过程的基础。有些时候，符合标准是强制性的，而且也涉及到测试。测试经理必须知道与标准符合情况相关的所有要求，确保维持适当的符合度。

3. 评审 – 180 分钟.

关键词

审计（audit）、非正式评审（informal review）、审查（inspection）、管理评审（management review）、主持人（moderator）、评审（review）、评审计划（review plan）、评审员（reviewer）、技术评审（technical review）、走查（walkthrough）

评审学习目标

3.2 管理评审和审计

TM-3.2.1 (K2) 了解管理评审和审计的关键特性

3.3 对评审的管理

TM-3.3.1 (K4) 分析项目，选择适宜的评审类型并定义开展评审的计划，确保评审得到适当的实施、跟进，和确定责任

TM-3.3.2 (K2) 了解参与评审需要的因素，技能和时间

3.4 评审的度量

TM-3.4.1 (K3) 定义评审使用的过程和产品度量

3.5 管理正式评审

TM-3.5.1 (K2) 举例说明正式评审的特点

3.1 简介

ISTQB®基础级大纲把评审作为对产品的静态测试活动引入。审计和管理评审更关注测试过程而不是软件工作产品。

评审作为一种静态测试，测试经理可能要对其成功实施负责，尤其是在涉及测试件产品的时候。然而，从更广的软件项目的角度来说，这项责任应该属于组织方针。由于不管在有软件项目之前还是软件项目当中，正式评审都可能在各个领域广泛应用，责任方可能是测试经理、质量保证经理或经过培训的评审协调员。在本大纲中，责任方（无论是谁）被称为评审组长。

评审组长应该确保现实条件与 ISTQB®基础级大纲里定义的、保证测试成功实施的因素相符合。另外，评审组长还应编制度量计划以确保评审带来成效。

因为测试人员对软件系统的操作行为和软件系统必须具备的特性有深入的了解，所以他们的参与对评审过程是非常重要的。

评审的参与人员应该接受评审的培训，了解在评审过程中各自的角色。所有的评审参与人都必须尽力实现一个好的评审应带来的效益。

评审如果进行得当，对整体交付的质量来说，是性价比最高的质量保证方法。因此，评审组长能够在其项目中实施有效的评审并证明这些评审的收益尤为重要。

项目可能进行的评审有：

- 合同评审，项目开始启动和项目主要里程碑点
- 需求评审，在需求可以被评审时发起，最好同时覆盖功能和非功能需求
- 概要设计评审，在整体架构设计可被评审时发起
- 详细设计评审，在详细设计可被评审时发起，一般在编码工作开始之前
- 代码评审，在单个软件模块生成时进行，可能包括单元测试及其结果以及代码本身
- 测试工作产品评审，可能覆盖测试计划、测试条件、质量风险分析结果、测试、测试数据、测试环境和测试结果
- 每个测试级别的测试入口（测试准备性）评审和测试出口评审，分别在测试执行开始前检查测试入口准则，和结束测试前检查测试出口准则
- 验收评审，用于获取客户或干系人对系统的批准

在对产品实施多种评审的基础上，评审组长需谨记虽然通过评审可以识别缺陷，但也要在实施评审的同时，结合其它的静态测试（如静态分析）和代码的动态测试。通过结合这些测试技术能提高测试覆盖率，并将识别出更多的缺陷。

不同的技术有不同的侧重点，例如，评审可以在编码前就消除需求的问题，避免将该问题引入到代码中。静态分析可以协助编码规范的执行，并识别出通过检查工作产品难以识别的问题。审查不仅能发现和移除缺陷，还能训练作者如何避免在其工作产品中生产缺陷。

ISTQB®基础大纲介绍了以下几种评审：

- 非正式评审
- 走查
- 技术评审
- 审查

除上述评审之外，测试经理也可能参与：
- 管理评审
- 审计

3.2 管理评审和审计

管理评审用于监督进度，评估状态和对未来活动做决策。此类评审支持项目未来的决策，如调整资源等级，实施纠正措施或改变项目范围。

管理评审的关键特性如下：
- 由或为直接负责项目或系统的经理组织
- 由或为某个干系人或决策者组织，例如更高级别的经理或主管
- 检查计划之间的一致性与偏差
- 检查管理规程的充分性
- 评估项目风险
- 评估行动的影响和衡量影响的方法
- 提供行动，待解决问题和决策的清单

过程的管理评审，如项目回顾（例如，经验教训），是过程改进活动的组成部分。

测试经理应该参与针对测试进度以及与此有关的管理评审。

审计一般用来展现对既定准则，如适用的标准，规章限制，或者合同义务的符合性。也就是说，审计的目的是独立评估过程、规章、标准的符合性。审计的关键特性如下：
- 由一位审计组长组织和主持
- 通过访谈、见证和检查文档收集符合性的证据
- 记录的结果包括观察结果、建议、纠正措施和通过 / 不通过的评价

3.3 对评审进行管理

评审应该安排在软件项目的自然转换点或里程碑。一般来说，应该在定义了需求和设计后举行相关的评审。评审以商业目标为起点，到设计的不可再分的级别，自上而下地进行。管理评审应该在项目的主要里程碑点进行，一般作为测试执行和其它主要项目阶段之前、过程中、之后的验证活动的一部分。评审策略必须与测试方针和总体测试策略一致。

在制定项目的总体评审计划前，评审组长（有可能是测试经理）应考虑：
- 评审的对象（产品和过程）
- 评审的参与人
- 需要覆盖的相关风险因素

在项目计划阶段初期，评审组长应该识别评审的对象，选择适当的评审类型（非正式评审、走查、技术评审或审查，或两、三种类型的混合），以及正式程度。在此，建议提供额外的评审培训。在确定上述内容后，测试经理可以分配评审过程的预算（时间和资源）。决定预算时应考虑到风险的评估和投资回报的计算。

评审的投资回报是指进行评审的成本与这些在评审中发现的相同缺陷在后期处理（假如它们在评审中没有被发现，或者它们被完全忽略了）的成本之间的差异。这里可以使用 2.7 节所述的质量成本计算方法来确定这个数值。

执行评审的最佳时机的确定取决于以下方面：
- 待评审项是否已经处于可评审的状态（如最终版本）
- 是否有合格的评审人员参与
- 最终版的评审对象应该到位的时间
- 特定评审对象的评审过程所需时间

评审组长应在计划测试时充分定义用以评估评审的度量。如果使用审查，应该在文档组成部分完成时（如单个需求或章节）按作者的要求进行简要审查。

在计划测试期间，必须定义评审过程的目标，包括有效和高效地开展评审，以及就评审反馈达成共识。项目评审多数针对整个系统进行，但对于子系统甚至单个软件要素也可能是必要的。应该根据项目规模和复杂度以及产品风险决定评审的次数、评审的类型、评审组织以及涉及的人员。

为了保证效率，评审参与人必须有足够的技术和规程方面的知识，除了这些，为了让评审有效，还需要参与人有其它的一些技能，如严谨和关注细节。好的评审意见必须足够清晰且按照恰当的优先级排列。对规程知识的需要可能意味着有必要进行一些培训，确保评审人员了解他们在评审过程中的角色和职责。

评审计划应该考虑到与进行评审时的技术、组织和人员相关的评审风险。是否有足够技术知识的评审人员的参与对评审成功与否是非常关键的。项目中所有的团队都应该参与评审计划的制定，以确保所有团队都对评审过程的成功做出承诺。计划时必须确保每个组织为评审人员分配了足够的准备和参与时间，他们能按照项目进度安排的评审节点参与评审。同时，也应该为评审人员安排接受所需的技术和过程培训的时间。测试经理还必须识别关键评审人员的后备人选，以防关键评审人员因为个人原因或工作原因无法参与评审。

在实际执行正式评审时，评审组长必须确保：
- 参与人提供的度量数据足以评估评审效率
- 编制了检查表，检查表也得到了维护以改进以后的评审
- 针对评审发现的缺陷定义了缺陷严重程度和优先级，用于缺陷管理（见第4章）

每次评审后，评审组长应该：
- 收集评审度量和确保识别的问题得到了充分的解决，符合评审确定的测试目标
- 使用评审度量作为确定最终的评审投资回报率（ROI）的输入
- 向相关干系人提供反馈信息
- 向评审参与人提供反馈

为了评价评审的有效性，测试经理可以把后续测试（即评审之后）的实际结果与评审报告上的结果进行比较。对于经过评审批准的工作产品，如果后面发现缺陷较多，评审组长应该考虑评审过程中可能会让缺陷遗漏的地方。可能的原因包括评审过程的问题（如入口 / 出口准则不当）、评审组组成不当、评审工具（检查表等）不充分、评审员培训和经验不足以及评审准备和会议时间太短。

遗漏缺陷（尤其是严重缺陷）的模式在若干项目重复出现，说明评审的开展有严重问题。这种情况下，评审组长需要检查过程并采取适当的措施。也有可能因为各种原因，经过了一段时间后，评审不再有

效。这种情况在项目回顾中会得到揭示，当评审发现缺陷的有效性降低时。这时评审组长也必须调查原因并予以解决。无论在什么情况下，评审度量数据都不应该用于惩罚或者奖励单个评审员或者文档作者，而应该关注于评审过程本身。

3.4 评审度量

评审组长（前面说过可能是测试经理）必须确保有可用的度量来：

- 评估评审对象的质量水平
- 评估进行评审的成本
- 评估进行评审后的下游收益

评审组长可以使用度量数据来确定评审的投资回报和效率。这些度量也可以用于报告和过程改进活动。

对于每个评审过的工作产品，都可以测量和报告下列度量，用于评估产品：

- 工作产品规模（文档页数，代码行等）
- 准备时间（评审前）
- 举行评审的时间
- 修复缺陷的返工时间
- 评审过程的持续时间
- 发现的缺陷数和缺陷严重程度
- 识别工作产品中的缺陷集群（即缺陷密度较高的领域）
- 评审类型（非正式评审、走查、技术评审或审查）
- 平均缺陷密度（如每页或每千行代码的缺陷）
- 估计的遗留缺陷数（或遗留缺陷密度）

每个评审都可以测量和报告下列度量，用于评估过程：

- 缺陷发现率（包括在生命周期的更晚期发现的缺陷）
- 评审过程工作量和时间的改进
- 计划评审的工作产品的覆盖率百分比
- 发现的缺陷类型和严重程度
- 参与人对评审过程有效性和效率的反馈
- 评审缺陷和动态测试缺陷以及产品缺陷的质量成本度量
- 评审有效性的相互关系（评审类型与缺陷发现有效性）
- 评审员的数目
- 每花一个小时发现的缺陷数
- 估计节省的项目时间
- 缺陷评估工作量（即总的发现和修复时间除以缺陷数量）

此外，上面提到的用于评估产品的度量也能用于过程评估。

3.5 管理正式评审

ISTQB®基础级大纲描述了正式评审的不同阶段：计划、启动、个人准备、评审会议、返工和跟进。要想正确实施一个正式评审，评审组长需要确保遵循了评审过程的所有步骤。

正式评审有许多特性，例如：

- 定义了入口和出口准则
- 评审人员使用检查表
- 交付物包括报告、评价表或其它评审总结表等
- 用于报告评审有效性、效率和进展的度量

在发起正式评审前，应该由评审组长确认满足了评审的先决条件（定义在规程或者入口准则清单上），并得到评审组长的同意。

如果没有满足评审的先决条件，评审组长可以向评审决策者提议以下任一项：

- 修改目标重新定义评审
- 为了使评审继续，进行必要的修正活动
- 推迟评审

最后，作为控制正式评审的一部分，这些评审在整体（更高级）项目的环境下得到监督，并与项目质量保证活动相关。正式评审的控制包括根据产品和过程度量提出反馈信息。

4. 缺陷管理 – 150 分钟.

关键词

异常（anomaly）、缺陷（defect）、缺陷分类委员会（defect triage committee）、失效（failure）、假阴性结果（注：漏报）（false-negative result）、假阳性结果（注：误报）（false-positive result）、阶段遏制（phase containment）、优先级（priority ）、根本原因（root cause）、严重程度（severity）

缺陷管理学习目标

4.2 缺陷生命周期和软件开发生命周期

TM-4.2.1 (K3) 为测试组织开发缺陷管理过程，包括缺陷报告流程，用于在测试生命周期中监督项目缺陷

TM-4.2.2 (K2) 说明有效的缺陷管理必需的过程和参与者

4.3 缺陷报告信息

TM-4.3.1 (K3) 定义在缺陷管理过程中应该收集的数据和分类信息

4.4 使用缺陷报告信息评估过程能力

TM-4.4.1 (K2) 解释缺陷报告信息怎样用于评估测试和软件开发过程的过程能力

4.1 简介

组织的缺陷管理过程和工具对于测试组和参与软件开发的各组来说都是非常关键的。有效的缺陷管理收集的信息可以帮助测试经理和其它项目干系人深入了解项目在整个开发生命周期中的状态，通过持续收集和分析数据，还可以帮助识别测试和开发过程的改进机会。

除了了解整个缺陷生命周期和如何使用它来监督测试和软件开发过程外，测试经理还必须熟悉哪些数据是亟需收集的，而且必须提倡过程和缺陷管理工具的适当用法。

4.2 缺陷生命周期和软件开发生命周期

在基础级大纲里提到，工作产品在产生过程中由于人为错误引入了缺陷。这些工作产品可能是需求规格说明、用户故事、技术文档、测试用例、程序代码或者其它任何在软件开发或维护过程中产生的工作产品。

在软件开发生命周期中任何时间点和相关工作产品中都可能引入缺陷。因此，软件开发周期的每个阶段都要包括发现和移除潜在缺陷的活动。例如，在设计规格说明，需求规格说明，代码交付给下一阶段的工作前可以使用静态测试技术（即评审和静态分析）。越早识别并移除缺陷，系统整体的质量成本越低；在缺陷引入阶段解决该缺陷，是质量成本最低的做法（如当软件过程有效实现阶段抑制时）。另外，在基础大纲里提过，静态分析直接找到缺陷而不是发现失效，因此，由于不需要通过调试活动将缺陷隔离，移除缺陷的成本得以降低。

在单元测试、集成测试和系统测试等动态测试过程中，缺陷是通过其引起的失效来揭示，即实际测试结果与预期结果不符（即异常）。在某些情况下，测试人员会因为没有观察到异常而发生假阴性结果（漏报）。如果测试人员观察到了异常，就意味着需要做进一步的调查。以填写缺陷报告作为调查的开始。

在测试驱动开发时，自动单元测试被用作为一种可执行的设计规格说明。产品代码在开发后立即执行这些测试。只要部分或全部测试没被通过，此单元的开发就不能算完成。因此，这类测试的失效不构成缺陷（报告），而且一般不会被跟踪。

4.2.1 缺陷工作流程和状态

大部分的测试组织在缺陷生命周期中使用工具管理缺陷报告。缺陷报告通常在工作流中得到处理，在整个缺陷生命周期会历经一系列状态。在大多数状态下，缺陷生命周期的唯一一个参与者拥有该缺陷报告，他负责执行某个任务，该任务完成时会让缺陷进入下一个状态（分配给下一个责任人）。在最终状态下，如当缺陷报告被关闭（通常意味着该报告指出的缺陷已被修复，修复的结果已经通过确认测试验证），被取消（通常意味着该缺陷报告无效），无法重现（通常意味着再也无法观察到该异常），或延期时（通常意味着该异常与一个实际存在的缺陷相关，但该缺陷在项目中无法得到修复），缺陷报告将不再有所有者，因为将不再需要采取进一步行动。

对于测试人员在测试中发现的缺陷，有三种状态需要测试团队采取相应行动：

- 初始状态
  - 在这个状态下，一名或者多名测试人员为负责解决缺陷的人员收集必要的信息，用以重现异常（详见下面缺陷报告的内容）
  - 还可能被称为 “打开”或“新”的状态
- 退回状态

- 在这个状态下，报告接收人拒绝接收报告或者向测试人员询问进一步的信息。这个状态表明初始状态的信息收集过程或者测试本身存在不足，测试经理应该监督以免出现过高的退回率。在这状态测试人员必须提供补充信息，或确认该报告应该被拒绝
- 还可能被称为“拒绝”或“澄清”状态

- 确认测试状态
  - 在这个状态下，测试人员会运行确认测试（一般按照缺陷报告中重现失效的步骤），判断修复是否解决了问题。如果确认测试表明缺陷已被修复，测试人员应关闭缺陷报告。如果确认测试显示缺陷没有被修复，测试人员应该再次打开缺陷报告，让之前安排的缺陷负责人继续修复缺陷
  - 还可能被称为 “解决”或“验证”状态

4.2.2 管理无效和重复缺陷

某些情况下，异常的出现不是因为缺陷，而是因为测试环境、测试数据、测试件的其它要素，或者测试人员自己的误解。如果测试人员建立缺陷报告后发现其内容和测试的工作产品缺陷无关的话，则被称为假阳性结果（误报）。这类报告一般会被取消或者作为无效缺陷报告被关闭。另外，在某些情况下，一个缺陷会表现出在测试人员看来完全无关的不同症状。如果事后发现两个或以上缺陷报告是由相同的根本原因引起的，通常只保留其中一个缺陷报告，其它的将视为重复缺陷报告被关闭。

尽管无效缺陷报告和重复缺陷报告代表了一定程度的低效率，但一定数量的此类报告是不可避免的，而且应该被测试经理所接受。当测试经理试图消除所有的无效和重复缺陷报告时，假阴性结果（漏报）的数量往往会增加，这是因为测试人员担心出错而不敢填写缺陷报告，这降低了测试组织的缺陷发现有效性，而缺陷发现有效性往往和测试组织的关键目标密切相关。

4.2.3 跨职能缺陷管理

虽然测试组织和测试经理是整个缺陷管理过程和缺陷管理工具的所有者，但是一般会有跨职能的小组负责管理整个项目所报告的缺陷。除了测试经理、缺陷管理委员会（或缺陷分类委员会）的成员外一般还可包括开发、项目管理、产品管理和其它软件开发的利益干系人。

当发现缺陷并记录在缺陷管理工具中后，缺陷管理委员会需要判断缺陷的有效性和是否要修复或延期。在做决策时要求缺陷管理委员会考虑修复缺陷与否的效益、风险和成本。如果要修复缺陷，小组要把此任务和其它项目任务做比较确定修复该缺陷的优先级。测试经理和测试团队可能会被问到某个缺陷的重要性并需要提供其它有效的客观信息。

缺陷跟踪工具不应替代沟通交流，缺陷管理委员会的会议也不应替代缺陷跟踪工具。沟通交流、充分的工具支持、定义明确的缺陷生命周期和缺陷管理委员会的充分参与对有效和高效的缺陷管理来说，都是不可或缺的。

4.3 缺陷报告信息

当识别出一个缺陷（通过静态测试）或者观察到一个失效（通过动态测试）时，相关人员应该收集相应数据并记录在缺陷报告中。

这些信息需要满足三个目的：

- 在整个缺陷生命周期中管理报告

- 对项目状态的评估，特别是就产品质量和测试进度而言
- 对过程能力的评估（见 4.4 节）

用于缺陷报告管理和项目状态的数据可以根据缺陷识别阶段而调整，通常越早发现缺陷，需要的信息越少（例如需求评审和单元测试）。然而，为了能在整个项目周期或者不同项目之间对过程缺陷数据进行有意义的比较，在整个生命周期和所有项目（理想情况下）中收集的关键信息应是一致的。

收集缺陷数据有助于监督测试进度和评估出口准则。例如，缺陷信息应该支持缺陷密度分析，缺陷发现和解决的趋势分析，从发现缺陷到缺陷解决的平均时间和失效强度（如**平均失效时间间隔**分析 MTBF）。

收集的缺陷数据可能包括：

- 缺陷发现人的名字
- 缺陷发现人的角色（如最终用户，业务分析师，开发人员，技术支持人员）
- 测试类型（如易用性测试，性能测试，回归测试）
- 问题总结
- 问题详细描述
- 重现失效（由某个缺陷引起的）的步骤，实际结果和预期结果（强调异常），包括截图、数据库转储，如果需要也可以提供日志
- 缺陷引入，发现和消除的生命周期阶段，包括测试级别（适时）
- 被引入缺陷的工作产品
- 对系统和 / 或产品干系人的影响严重程度（一般由系统的技术行为决定）
- 缺陷修复的优先级（一般由系统失效带来的业务影响决定）
- 存在缺陷的子系统或者组件（用于缺陷集群分析）
- 发现缺陷时项目正在进行的活动
- 识别问题的方法（例如，评审，静态分析，动态测试，产品使用）
- 缺陷的类型（通常与使用的缺陷分类法一致）
- 缺陷影响的质量特性
- 观察到缺陷的测试环境（动态测试）
- 出现问题的项目和产品
- 目前的缺陷所有者，如目前被安排处理该问题的人员，前提是该报告还没到最终状态
- 报告目前的状态（一般作为生命周期的一部分，使用缺陷跟踪工具进行管理）
- 被观察出问题的工作产品（如测试对象和他们的版本号），和问题修复后的工作产品
- 对项目和产品干系人利益的影响
- 解决 / 不解决问题的结论、建议和批准
- 修复 / 不修复问题的风险、成本、机会和效益
- 各缺陷生命周期状态转移发生的日期，每个转移后的报告所有者，以及项目组成员隔离，修复和验证缺陷修复的活动
- 关于缺陷最终怎样修复的描述和对修复结果进行测试的建议（如果缺陷是通过软件变更修复）
- 其它参考信息，如识别缺陷的测试，风险、需求及其它和缺陷相关的测试基本要素（动态测试）。

各类标准和文档，如 ISO 9126 [ISO9126]（将被 ISO 25000 替代），IEEE 829 [IEEE829]，IEEE1044 [IEEE1044]，和正交缺陷分类，有助于测试经理决定报告缺陷时需要收集哪些信息。

无论最终决定收集哪些信息，测试人员在输入信息时，必须确保信息的完整、简明、准确、客观、相关和及时，这一点很关键。即便通过人工干预或者面对面的交流能克服个别不符合要求的缺陷报告数据带来的问题，但这些数据会对项目状态、测试进度和过程能力的正确评估带来无法克服的障碍。

4.4 使用缺陷报告信息评估过程能力

在第二章已经讨论过，缺陷报告对项目状态的监督和汇报是非常有用的。尽管度量对过程的影响主要在专家级测试管理大纲 [ISTQB® ETM SYL] 中讨论，在高级大纲里，测试经理应该意识到缺陷报告对于评估测试能力和软件开发过程能力的意义。

除了在第二章和 4.3 节提过的测试进展监督信息，缺陷信息应该支持过程改进活动。例如：

- 根据缺陷引入的阶段、识别的阶段和消除的阶段的信息，或者按阶段来划分，评估阶段缺陷密度，并提出改善各阶段缺陷发现有效性的方案
- 根据缺陷引入阶段的信息，对引入缺陷最多的阶段进行帕累托图（Pareto）分析，使改进更具针对性，减少缺陷总数
- 根据缺陷根本原因信息确定缺陷引入的潜在原因，以减少缺陷总数，支持过程改进
- 根据缺陷引入、识别和消除的阶段的信息进行质量成本分析，尽量降低与缺陷相关的成本
- 根据缺陷组件信息进行缺陷聚集分析，以便更好地理解技术风险（对基于风险的测试来说）和重新设计问题较多的组件

在专家级测试管理大纲 [ISTQB® ETM SYL] 里讨论关于使用度量数据评估测试过程的有效性和效率的内容。

某些情况下，小组可能选择不跟踪部分或者整个软件开发周期里发现的缺陷。这么做通常是为了加快效率和减少过程费用，但事实上这种做法极大地降低了测试过程和软件开发过程能力的可见度。这也导致由于缺乏可靠的数据难以实施上述的改进。

5. 改进测试过程 – 135 分钟.

关键词

能力成熟度模型集成（Capability Maturity Integration（CMMI））、关键测试过程（Critical Testing Process（CTP））、系统性测试和评价过程（Systematic Test and Evaluation Process（STEP））、测试成熟度模型集成（Test Maturity Model integration（TMMi））、TPI Next

测试过程改进的学习目标

5.2 测试改进过程

TM-5.2.1 (K2) 举例说明为什么测试过程改进是重要的

5.3 改进测试过程

TM-5.3.1 (K3) 使用 IDEAL 模型定义测试过程改进计划

5.4 使用 TMMi 改进测试过程

TM-5.4.1 (K2) 总结 TMMi 测试过程改进模型的背景，范围和目标

5.5 使用 TPI-Next 改进测试过程

TM-5.5.1 (K2) 总结 TPI-Next 测试过程改进模型的背景，范围和目标

5.6 使用 CTP 改进测试过程

TM-5.6.1 (K2) 总结 CTP 测试过程改进模型的背景，范围和目标

5.7 使用 STEP 改进测试过程

TM-5.7.1 (K2) 总结 STEP 测试过程改进模型的背景，范围和目标

5.1 简介

在建立了组织的测试过程之后，需要对过程进行持续改进。本章我们先讨论通用的过程改进问题，然后介绍可用于测试过程改进的模型。测试经理应该假设他们是测试过程变更和改进的推动力量，因此必须熟悉本章讨论的行业认可的技术。更多的测试过程改进内容见专家级测试过程改进大纲。

5.2 测试改进过程

正如组织使用测试改进软件，过程改进技术可以被选择用于改进软件开发过程和产生的软件交付物。过程改进也适用于测试过程。改进对软件和包含软件的系统的测试有很多方法。这些方法旨在通过提供指引和改进的区域改进过程和从而改进交付物。

测试通常占了项目总成本的主要部分。然而，在诸多软件过程改进模型中对测试过程的关注都很有限，如 CMMI®（详见下文）。

测试成熟度模型集成（TMMi®），系统性测试和评价过程（STEP），关键测试过程（CTP）和 TPI Next®等测试改进模型的开发解决了大多数软件过程改进模型对测试关注不够的问题。这些模型使用得当的话，可以提高一定程度的跨组织度量，用于基准比较。

本章列出的模型并不作为推荐使用，而是作为代表介绍此类模型的工作原理和内容。

5.2.1 过程改进的介绍

过程改进与软件开发过程和测试过程有着密切的联系。吸取错误的教训，有助于组织改进用于开发和测试软件的的过程。戴明改进循环（Deming improvement cycle）：计划、实施、检查、行动（Plan、Do、Check、Act --PDCA），已经使用了多年，直到现在，测试人员在过程改进中仍在使用。

过程改进的一个重要前提就是要相信，系统质量在很大程度上由开发该软件的过程质量决定。软件工业方面的质量提升，减少了软件维护对资源的需求，由此提供了更多的时间在未来创造更多更好的解决方案。过程模型可作为改进的起点，通过使用模型对组织的过程成熟度/成熟性进行测量。模型还基于评估的结果为组织提供改进的框架。

过程评估明确了过程能力，并推动过程改进。过程改进的效果可通过后续的过程评估进行度量。

5.2.2 过程改进的类型

使用评估模型是常见的做法之一，该做法的标准方法是确保利用尝试过并被证明有效的实践来改进测试过程。

过程改进模型可以分成两种：

1. 过程参考模型，包括成熟度/成熟性度量的评估，以模型作为基准在框架内评估组织的能力，并提供过程改进的路线图。
2. 内容参考模型，对组织的改进机会进行以业务为驱动的评价。某些情况下，通过将客观的度量数据对比业界的平均水平基准识别改进机会。这种评价可以用于建立改进路线图。

即便不使用上述模型，也可以通过分析法和回顾会议等方式实现测试过程改进。

5.3 改进测试过程

为了达到更高的成熟度/成熟性和更专业的水平，IT 业常常参照测试过程改进模型开展工作。行业标准模型有助于开发跨组织的度量，用于比较。针对测试行业过程改进的需要，几套受推荐的过程得以实现，它们分别是 STEP、TMMi、TPI-Next 和 CTP。阶段式模型，如 TMMi 和 CMMI ，不同的公司和组织提供了比较基准。持续性模型，如，CTP、STEP 和 TPI Next，允许公司在过程改进实施时拥有更高的自由度，优先处理优先级最高的问题。下面会更详细地讨论各个模型。

上述模型可以帮助组织在测试过程方面更准确地定位。在评估过后，TMMi 和 TPI-Next 会为测试过程的改进提供建议的路线图。STEP 和 CTP 则是向组织提供方法以判定哪些过程改进投资能带来最大的回报，并让组织自行选择合适的路径。

决定过程需要评审和改进后，要遵循的过程改进实施步骤可以像在 IDEAL[SM] 模型 [IDEAL 96] 里定义的：

- 启动改进过程（**I**nitiating）
- 诊断目前的情况（**D**iagnosing）
- 制定测试过程改进计划（**E**stablishing）
- 实施改进（**A**cting）
- 改进经验教训总结（**L**earning）

启动改进过程
在过程改进活动开始前，需要取得干系人对改进目标、目的、范围和覆盖的共识，并选择过程改进模型。可以选择业界公认的模型（如 CTP、STEP、TMMi 和 TPI-Next）或内部开发的模型。另外，还要定义在整个改进活动开展期间判断改进成功与否的标准和定义度量方法。

诊断目前的情况
使用与干系人已经达成一致的评估方法，编写测试评估报告，内容包括对目前测试实践的评估和改进机会清单。

制定测试过程改进计划
过程改进机会清单上应设定了改进机会的优先级。优先级设定可以参考投资回报率、风险、与组织战略的符合度和 / 或可度量的定量或定性的效益。设定好优先级后，编写实施这些改进的计划。

实施改进
实施测试过程改进计划，以实现改进。过程改进实施包括需要的任何培训、辅导、过程试运行和最终的全面部署。

改进经验教训总结
在充分部署了过程改进后，需要验证改进带来的收益（包括之前既定的收益和潜在的意料之外的收益）。另外，检查满足了哪些过程改进活动的成功标准也很重要。

根据不同的过程模型，本阶段是开始监督更高成熟度/成熟性等级的节点和决定重新开始改进还是结束改进的决策点。

5.4 使用 TMMi 改进测试过程

测试成熟度模型集成（TMMi）包含 5 个成熟度等级，旨在补充 CMMI 模型。每一个等级都包括已定义的过程域，组织必须通过实现某个过程域的特殊和通用目标，达到 85%的过程域满足度，才能升级到更高等级。

TMMi 的成熟度等级分别是：

- 1 级：初始级
  初始级表示测试过程没有正式的记录，而且杂乱无章。测试在编码后以特别的方式执行，与调试没有区别。测试的目的被理解为证明软件正常工作。
- 2 级：管理级
  2 级要求测试过程和调试完全分开。可以通过以下办法达成：设定测试方针和目标，引入基本测试过程中步骤（如测试计划）和实施基本测试技术和方法。
- 3 级：定义级
  3 级要求将测试过程集成到软件开发生命周期，并在正式标准、规程和方法中予以记录。组织开展了评审，而且有单独的软件测试职能，并对其进行了监督和控制。
- 4 级：度量级
  4 级要求组织能够有效地对测试过程进行度量、管理，有利于项目开展。
- 5 级：优化级
  最后级表示组织的测试过程能力度达到如下状态：测试过程的数据可以用于预防缺陷，而且注意力集中在优化已建立的过程。

更多 TMMIi 的内容，见 [vanVeenendaal11] 和 [www.tmmi.org].

5.5 使用 TPI Next 改进测试过程

TPI-Next 模型定义了 16 个关键领域，每个领域覆盖测试过程的一个特定部分，例如测试策略、度量、测试工具和测试环境。

该模型定义了四个成熟度等级：

- 初始级
- 控制级
- 高效级
- 优化级

对每个成熟度等级的关键领域的评估都设定了特殊检查点。检查结果汇总到成熟度矩阵中展示出来，成熟度矩阵覆盖了所有关键领域。根据测试组织的需要和能力，可以对改进目标的定义和实施方式进行裁减。

通用方法使 TPI Next 模型独立于其它软件过程改进模型。它覆盖了测试工程和支持管理决策的内容 [deVries09]。

更多关于 TPI Next 的内容，见 [www.tpinext.com]

5.6 使用 CTP 改进测试过程

使用关键测试过程（CTP）评估模型的基本前提是某些测试过程是关键的。这些关键测试过程，如果实施得当，将会造就测试团队的成功。相反，如果这些活动实施不当，即便测试人员和测试经理本身再有能力也不会获得成功。该模型识别十二个关键的测试过程。CTP 是一个内容参考模型。

CTP 模型是一个依赖背景的方法，它允许裁剪，包括：

- 识别特殊挑战
- 认识良好过程的属性
- 选择过程改进实施的顺序和重要性

CTP 模型可用于所有软件开发生命周期模型。
除了参与人访谈，CTP 模型还包括使用度量，将组织实际情况与行业平均值和最佳实践进行比较。

更多 CTP 的内容，见 [Black03].

5.7 使用 STEP 改进测试过程

STEP（系统化测试和评估过程），与 CTP 比较类似，而与 TMMi 和 TPI-Next 不同，并不要求遵循特定的顺序来进行改进。

STEP 主要是一个内容参考模型，它基于这样的设想：测试是一个生命周期活动，从明确需求期间开始直到系统退役。STEP 方法强调“先测试后编码”，而这种方针的实现途径是使用基于需求的测试策略以保证在设计和编码之前，已经设计了测试用例以验证需求规格说明。

STEP 方法的基本前提包括：

- 基于需求的测试策略
- 在生命周期初期开始测试
- 测试用来作为需求并使用模型
- 由测试件的设计引导软件设计（测试驱动开发）
- 较早发现缺陷或在整体上预防缺陷
- 对缺陷进行系统地分析
- 测试人员和开发人员一起工作

在某些情况下会结合 STEP 评估模型和 TPI-Next 成熟度模型。

更多关于的 STEP 内容，见 [Craig02].

6. 测试工具及自动化 – 135 分钟

关键词
开源工具（open-source tool）、定制工具（custom tool）

测试工具及自动化的学习目标

6.2 工具的选择

TM-6.2.1 (K2) 描述选择开源工具时需要考虑的管理问题
TM-6.2.2 (K2) 描述决定定制工具时需要考虑的管理问题
TM-6.2.3 (K4) 对一个给定的情况作出评估，来制定选择工具的计划，包括风险、成本和收益

6.3 工具生命周期

TM-6.3.1 (K2) 阐述一个工具的生命周期的不同阶段

6.4 工具度量

TM-6.4.1 (K2) 阐述如何通过使用工具来改进度量的收集和评估

6.1 简介

此章节在基础级大纲的基础上进行了拓展，阐述了测试经理在工具和自动化方面必须考虑到的一些一般概念。

6.2 选择工具

在选择测试工具时，测试经理必须考虑许多不同的问题。

从历史经验来看，最常见的选择是从商业供应商处购买工具。某些情况下，这也许是唯一可行的选择。但是，也可能有其它可行的选择，例如开源工具和定制工具。

不管是哪种类型的工具，测试经理都必须通过成本收益分析来仔细调查该工具在预期使用年限内的所有成本。在下面的投资回报率（ROI）这一章节会谈到这一点。

6.2.1 开源工具

在测试过程中几乎所有的方面都有开源工具，仅举几例：从测试用例管理到缺陷追踪，再到测试用例自动化等。开源工具有一个很重要的特质，即工具本身的初始采购成本通常不高，同时也可能没有任何正式可用的支持。然而，还是有很多开源工具确实为用户提供了免费的后续跟踪来进行非传统或非正式的支持。

另外，很多开源工具最初是为解决某个特定问题或针对单个问题而创建的，因此，该工具也许不能实现一个与之类似的商业工具的所有功能。正因如此，在选择开源工具之前，需要对测试团队的实际需求进行周密的分析。

使用开源工具的一个好处是，用户通常可以对工具进行修改或扩展。如果该组织有核心能力，可以修改工具，使其可以与其它工具一起工作，或改变该工具以适应测试团队的需要，可以组合多个工具来解决供应商软件不能解决的问题。当然，使用的工具越多和修改的内容越多，会导致复杂度和花费越高。测试经理必须确保测试团队不仅仅是为了尝试工具而去使用开源工具。和其它工具一样，使用工具的目的永远是为了获得投资正回报率。

测试经理必须理解所选工具的许可协议组合。很多开源工具的协议属于 GNU 通用公共许可的变化版本，规定软件分发的条件必须与收到软件的条件相同。当测试组为了更好地支持测试而改变测试工具时，这些改动可能需要提供给在该工具许可下的所有外部用户。测试经理应该了解组织再次发布该软件的法律后果。

开发安全关键或业务关键软件的组织，或是需要遵循一定规章的组织，在使用开源工具时可能会遇到问题。尽管很多的开源工具质量很高，但其准确性往往不能被证实。而商业工具通常可以证明其准确性和对特定任务（如 DO-178b）的适用性。尽管开源工具可能不比现货软件逊色，但对工具的认证可能需要使用该工具的人群去做，这样一来就会造成额外的开销。

6.2.2 定制工具

测试组织也许会发现，有一些特定的需求是商业和开源工具都不能满足的。其原因可能是独家的硬件平台、定制的环境或被一种独特的方式修改过的工作流程。在这种情况下，如果测试团队有核心能力，测试经理不妨考虑开发一个定制工具。

开发定制工具的好处是，该工具可以满足团队的确切需要，并能在团队所要求的环境下高效地运行。工具可以被编写成在使用时能与其它工具交互的形式，并生成团队所需要的特定格式的数据。另外，自定义的工具还可能用于组织的其它项目中。但是，在计划对其它项目发布工具前，先评审使用工具的目的、目标、收益和潜在缺点是很重要的。

测试经理在考虑开发定制工具时，还必须考虑可能产生的负面问题。定制工具往往依赖于开发它们的人员。因此，定制工具必须有充分的文档以便其它人员能对其进行维护。否则，在开发工具的人离开项目后，工具可能会无人问津，然后被弃用。随着时间的推移，定制工具的应用范围可能会扩展超出最初创建的动机，而这也许会导致工具的质量问题，例如误报的缺陷报告或创建了不准确的数据。测试经理必须牢记，定制工具同样也是一款软件产品，正因如此，它也会出现其它软件产品类似的开发问题。定制工具也需要进行设计和测试，以保证它们按照预期运行。

6.2.3 投资回报（ROI）

测试经理有责任确保测试组织引入的所有工具对团队工作都是有附加值的，且为组织带来投资正回报。为了确保工具将实现真正的和持久的收益，在购买和开发工具前应该进行成本-收益分析。在分析中，投资回报需要考虑重复成本和临时成本，其中一些是费用成本，另一些是资源或时间成本，以及可能降低工具价值的风险。

临时成本包括下列内容：

- 定义对工具的需求以达到目标和目的
- 进行评价和选择正确工具和工具的供应商
- 采购、调整或开发工具
- 进行初始的工具培训
- 将该工具与其它工具集成
- 采购支持工具所需的硬件 / 软件

重复成本包括下列内容：

- 持有工具
  - 许可和支持费用
  - 工具本身的维护费用
  - 由工具创建的工件的维护
  - 持续的培训和辅导
- 将工具移植到不同的环境
- 调整工具以适应未来的需求
- 改进质量和过程确保已选工具的最佳使用

测试经理同时应考虑任何工具固有的机会成本。花在采购、管理、培训以及使用该工具的时间，其实可以花在实际的测试任务上，因此，直到工具上线前，可能需要更多的测试资源。

在使用工具时会有很多风险，并非所有的工具都是收益大于风险的。基础级大纲中已经讨论过工具风险。测试经理在分析投资回报时，还应当考虑下列风险：

- 组织的不成熟（并未准备好使用该工具）
- 该工具创建的工件可能难以维护，需要根据待测软件的改变持续更新
- 减少测试分析师在测试任务中的介入可能降低测试的价值（例如，只运行自动化脚本，可能会降低缺陷发现有效性）

最后，测试经理必须着眼于使用工具可能带来的收益。引入和使用工具可能带来下列收益：

- 减少重复性工作
- 缩短测试周期（如通过自动化回归测试）
- 降低测试执行成本
- 增加某些类型的测试（如回归测试）
- 减少不同测试阶段的人为错误，例如：
  - 使用数据生成工具生成更有效的测试数据
  - 使用比较工具对测试结果进行更准确的比较
  - 使用脚本工具输入更正确的测试数据项
- 获取测试信息的工作量减少，例如：
  - 工具生成的报告和度量数据
  - 复用测试资源，如测试用例、测试脚本及测试数据
- 增加没有工具就无法完成的测试（如性能测试，负载测试）
- 测试人员通过实现测试自动化和组织测试，证明自身对复杂工具的理解和应用，从而提升了测试人员的地位

总的来说，很少有测试团队只用一种工具，团队所获得的总的投资回报通常是使用所有工具的作用。工具需要共享信息，共同协作。建议建立一个长期、全面的测试工具策略。

6.2.4 选择流程

测试工具是一个长期的投资，可能用于一个项目中的多次迭代，和 / 或适用于多个项目。测试经理必须从几个不同角度考虑一个预期的工具。

- 对于业务来说，投资正回报率是必需的。为了能在投资中获得高价值，组织应当确保这些工具的互操作性——可能包括测试工具和非测试工具的交互操作。在某些情况下，要实现互操作性，必须改善流程和工具使用的衔接能力，这可能需要一段时间来实现
- 对于项目而言，工具必须是有效的（例如，在进行人工测试时避免错误，如避免数据输入时的录入错误）。工具可能需要很长时间才能获取投资正回报率。在很多情况下，投资回报可能在第二次版本发布时或维护阶段才出现，而不是在最初实施自动化的项目中。测试经理应当从应用的整个生命周期来考虑投资回报
- 对于使用工具的人来说，该工具必须支持项目组成员，让他们更有效、更高效地完成自己的任务。必须考虑学习曲线，确保使用者能够在最小的压力下快速地学会使用工具。在初次引入某个测试工具时，需要对使用者进行培训和辅导

为了确保所有观点都有考虑到，为测试工具的引入创建一个规划图是很重要的。

在基础级大纲中已经讨论过选择测试工具的流程，如下所示：

- 评估组织的成熟度/成熟性
- 识别工具的需求
- 评估工具
- 评估供应商或服务支持商（开源工具、定制工具）
- 识别为使用工具进行训练和辅导的内部需求
- 考虑目前测试团队的测试自动化技能，评估培训需求
- 估算成本-收益（如在 6.2.3 章节关于投资回报率的讨论）

对于每个类型的工具，不管将在哪个测试阶段使用，测试经理应当考虑下列功能：

- 分析
  - 这个工具是否能“理解”给定的输入？
  - 工具对于目的是否适用？

- 设计
  - 此工具是否能根据现有信息帮助设计测试件（例如，根据需求生成测试用例的测试设计工具）？
  - 能否自动生成设计？
  - 能否以可维护和可使用的格式生成或部分生成实际的测试件代码？
  - 能否自动生成必要的测试数据（例如，基于代码分析生成数据）？
- 数据及测试选择
  - 工具如何选择所需数据（例如，使用哪组数据执行哪个测试用例）？
  - 工具能否接受人工或自动输入的选择准则？
  - 工具能否根据已选的输入来决定如何过滤产品数据？
  - 工具能否根据覆盖准则来决定需要执行哪些测试（例如，提供一组需求后，工具能否顺着可追溯性来决定需要执行哪些测试用例）？
- 执行
  - 工具能否自动运行，还是需要人工介入？
  - 工具如何停止和重启？
  - 工具能否“领会”关联事件（例如，当缺陷报告中的测试用例已关闭，测试管理软件是否能自动更新测试用例状态）？
- 评估
  - 工具如何判断它是否收到了合适的结果（例如，工具将使用测试准则来决定响应的正确性）？
  - 工具具备什么样的错误恢复能力？
  - 工具能否提供充分的记录和报告？

6.3 工具生命周期

一个工具的生命周期分为以下四个不同的阶段，测试经理必须对这四个阶段进行管理：

1. 获得：工具的获得必须如上文所述（见 6.2 章节 选择工具）。在决定引入某个工具后，测试经理应当指定一人（通常是测试分析师或技术测试分析师）来管理工具。此人应当决定工具使用的时机和方式，创建的工件如何储存，命名规则等。应当事前决定好这些内容，而不是等到要用工具了再临时去决定，做决定的时间不同，最终的投资回报率会有显著的差异。通常需要对工具的使用者进行培训。
2. 支持与维护：需要对工具进行持续的支持和维护。维护工具的责任也许会落在工具管理员身上，或是被分配到一组特定的工具团队上。如果工具需要与其它工具一起运行，那么应该考虑数据交换和合作进程。还需要考虑工具的备份、恢复和输出。
3. 演变：必须考虑到转换。随着时间的推移，环境、业务需求或供应商问题都可能要求对工具本身或其用途进行大的改动。例如，工具供应商也许会要求对工具进行更新，但这可能导致与其它工具合作的问题。由于业务原因导致必要的环境变动也可能会导致工具的问题。工具的运行环境越复杂，变更越会破坏其使用。在这一点上，根据工具在测试中扮演的角色，测试经理也许需要确保组织在出现突发事故时能保证服务的连续性。
4. 退役：工具总有它的使用寿命。也就是说，工具也需要优雅地退场。工具所支持的功能需要被替代，数据需要进行保存和归档。当一个工具步入了生命周期的尽头时，或简单来讲，转换到新工具的机会和收益已经超越了成本和风险时，它（旧工具）将会退役。

在工具的整个生命周期中，测试经理负责确保工具能为测试团队顺利运作并提供持续的服务。

6.4 工具度量

测试经理可以设计和收集客观的度量，而这些度量来自于技术测试分析师和测试分析师使用的工具。不同工具可以捕捉宝贵的实时数据，又能减少数据收集的工作量。测试经理可以使用这些数据来管理整体的测试工作。

不同的工具侧重于不同类型数据的收集。例如：

- 测试管理工具可以提供多种度量。从需求到测试用例及自动化脚本的可追溯性允许进行覆盖范围的度量。任何时间都可获取目前可用的测试、计划的测试以及当前执行的状态（通过、失败、跳过、中断、排队中）的快照
- 缺陷管理工具可提供关于缺陷的丰富信息：目前状态、严重度和优先级、在系统中的分布等。其它富有启发性的数据如缺陷的引入和发现阶段、缺陷逃脱率等，都帮助测试经理推动过程改进
- 静态分析工具可为发现和报告可维护性问题提供帮助
- 性能工具可提供在系统可扩展性方面有价值的信息
- 覆盖工具可以帮助测试经理了解系统实际进行过测试的比例

应该在工具的选择过程中就定义工具的报告的需求。这些需求必须在工具配制过程中恰当的实现来确保通过工具跟踪的信息能以干系人易理解的方式进行汇报。

7. 人员技能 – 团队构成 – 210 分钟.

关键词
测试独立性（Independence of testing）

人员技能——团队构成的学习目标

7.2 个人技能

TM-7.2.1 (K4) 使用技能评估数据表，分析团队成员在使用软件系统、领域及业务知识、系统开发领域、软件测试及人际交往技能方面的强项和弱项

TM-7.2.2 (K4) 分析一次给定的团队技能评估，制定一份培训及技能培养计划

7.3 测试团队动力

TM-7.3.1 (K2) 针对某一给定的情况，讨论领导测试团队所必须具备的硬技能和软技能

7.4 使测试适合组织

TM-7.4.1 (K2) 说明独立测试的可选做法

7.5 激励

TM-7.5.1 (K2) 列举能对测试人员产生激励和消极影响的因素

7.6 沟通

TM-7.6.1 (K2) 说明哪些因素会影响测试团队内的沟通，哪些因素会影响测试团队与干系人之间的沟通

7.1 简介

一个成功的测试经理所招聘、雇佣及维护的团队，其成员各有所长，能力互补。技能的需求可能随着时间而改变，所以，除了在最初雇佣合适的人员外，为了让测试团队保持着最好的水平，提供适当的培训和成长机会也是很重要的。除了团队所需技能外，测试经理还须具备一系列技能，使其在高压力、快节奏的环境能有效地履行职责。

本章节着眼于如何评估技能，如何填补差距以创造一个内部凝聚力强大并在组织中有效的协作团队、如何激励团队以及如何有效沟通。

7.2 个人技能

一个人在软件测试方面的能力，可以从经验、教育或培训中得来。下列每项都是测试员知识的来源：

- 软件系统的使用
- 领域或业务知识
- 参与软件开发过程活动的多个阶段，包括分析、开发和技术支持
- 参与软件测试活动

软件系统的最终用户对系统如何运作，哪个部分一旦出现失效造成的影响最为严重，以及系统在多种情况下应作出何种反应都非常了解。拥有专业领域知识的用户知道哪些领域对于业务来说是最重要的，以及这些领域如何影响业务满足需求的能力。这些知识可用于帮助对测试活动进行优先级排序，创建实际的测试数据和测试用例，以及验证或创建用例。

软件开发过程（需求分析、架构、设计和编码）方面的知识能帮助洞察错误如何引入、何处可被发现以及如何防止引入错误。在技术支持方面的经验能帮助了解用户经验、期望及易用性要求。软件开发经验对于使用测试工具是很重要的，使用测试工具需要有编程及设计的专业知识，参与过静态代码分析、代码评审、单元测试以及集成测试的技术层面。

特定的软件测试技能在基础级大纲、高级测试分析师以及高级技术测试分析师大纲中都讨论过，它包括分析规格说明的能力，参与风险分析、设计测试用例，尽职运行测试并记录结果。

拥有项目管理方面的知识、技能和经验对于测试经理来说尤为重要，因为测试管理包括了很多项目管理活动，例如制定计划、追踪进展并向干系人报告等。在没有项目经理的情况下，测试经理可能承担起测试经理及项目经理两个角色，尤其是在项目的后期。这些技能是超出基础级大纲及本大纲中讨论的内容。

除了技术能力外，人际交往能力，如给予和听取建设性批评、影响力以及协商能力对于测试角色而言都很重要。一个技术能力强的测试员，如果不具备必要的软实力，往往也会失败。除了有效地与他人合作之外，成功的专业测试人员必须要有条理、注意细节并具备良好的书面和口头沟通能力。

理想的测试团队会组合不同技能和经验水平，团队成员应该乐意并能够相互学习。在某些环境下，一些技能比其它的更为重要或更受到尊重。例如，在一个需要 API 测试和编程技能的技术测试中，专业技能可能会比领域知识更有价值。而在黑盒测试中，领域专业知识也许是最有价值的。切记环境和项目并不是一成不变的。

在创建技能评估数据表时，测试经理应当列出所有工作和岗位的重要技能。在列出这些技能后，按照给定的打分机制（例如，从 1 分到 5 分，5 分为该领域中最高的能力等级），对团队中的个人能力进行评

估。根据评估结果来判断团队中各人的强弱项，并基于这些信息，制定个人或团队培训计划。测试经理也许会创建一些个人在特定领域提升能力的绩效目标，并应确定用于评估个人技能的具体标准。

雇佣人员应该出于长远的考虑，而不仅是为了某个项目。当测试经理在测试人员身上投入资源，并建立起一个持续学习的环境时，团队人员会受到激励，并去提高他们的技能和知识，为下次机会的到来做好准备。

测试经理不太可能招聘到完美的测试团队成员。并且，即使团队成员对目前项目来说是完美的，对下个项目来说也许就不是完美的了。测试经理应当聘请聪明、好奇心强、适应能力强、乐于工作、能作为团队的一部分有效工作、愿意并能够学习的人员。即使可能无法招到这样完美的个人，也可以通过平衡团队中个人的强项和弱项来建立一个强大的团队。

根据技能评估数据表，测试经理可以找出其团队的强弱项，并以此作为培训及技能培养计划的基础。从对团队的有效性和效率影响最大的弱项着手，测试经理需要决定如何解决这些领域。一种途径是培训，例如安排人员参加外部培训课程、进行内部培训、开发定制培训、使用在线培训课程等。另一种途径是自学，例如通过书籍、网络讨论、互联网资源等。还有一种方法是交叉培训，例如安排一位想要学习某种技能的人员，去和一位已具备此技能的人员一起工作，且该工作会使用到此技能，或让当地专家提供关于擅长的领域专业知识的演讲等。（辅导是类似的一种方式，让新担任此角色的人与此职位的资深人士结对，由资深人士持续地提供建议和协助。）除了解决弱项外，测试经理还应当谨记可以借助技能评估中发现的强项来帮助进行培训和技能发展计划。更多的测试团队发展计划，请参阅 [McKay07]。

7.3 测试团队动力

要尽可能地建立最好的团队，对于组织的管理人员来说，挑选人员就是最重要的工作之一。除了工作所必需的个人技能之外，还需要考虑很多方面。当挑选人员加入团队时，必须考虑团队动力。此人的技能和个性是否能与团队中已存的技能和个性互补？要考虑测试团队拥有多种个性和技术能力组合的优势。一个强大的测试团队能够处理各种复杂程度的项目，同时能成功地做好与其它项目团队成员的人际互动。

测试活动往往压力很大。软件开发进度计划往往会被压缩，甚至压缩到不切合实际。对测试团队的期望值通常很高，有时甚至是不合理的。测试经理必须雇佣能应对高压环境的人员，能够转移挫折，并且即便在看似不可能的进度安排下也能专心工作。尽管是由测试经理来处理进度安排和期望值等问题，但测试经理也必须明白，团队成员同样也承受了这些压力。当测试经理在为团队招聘人员时，考虑工作环境以及个性能否适应工作环境也是很重要的。在时间不够，工作很多的情况下，测试经理要寻找应该是完成任务后能主动询问接下来做什么的人。

当个人感到自己是有价值的并被需要时，他们将会工作的更努力，花更多心思。个人必须明白，他们每个人对团队来说都是重要的一员，他的努力对团队的成功是至关重要的。当有了这种动力时，在测试团队内会发生非正式的知识传播、测试团队成员也会自发地平衡各自不同的工作负荷，测试经理则有了更多的时间来处理外部问题。

新的团队成员必须迅速融入团队，并对其进行足够的管理和支持。每个人在团队中都应有定义的角色。角色的定义可根据个人的评估过程。目标是让每个人都能为团队整体的成功作出贡献。这大部分取决于将个性与团队角色匹配，在个人已具备的技能的基础上，增加他们的技能组合。

在确定哪些员工将被录用，或谁该进入测试团队时，客观的评估其技能会很有帮助，这可以通过面试、对候选人的测试、对工作样本的评估和通过对推荐和证明的验证来完成。需要评估的技能包括：

技术技能（硬技能）可由下列活动体现：

- 根据需求文档生成测试用例
- 评审技术文档（需求、代码等）
- 清晰、易于理解并客观地提出评审意见
- 根据给定的场景从多种测试技术中选择适用的测试技术（例如，使用决策表来测试一系列业务规则）
- 评估失效并准确地记录
- 阐述对缺陷分类信息的理解
- 阐述对缺陷根本原因的理解
- 使用工具来测试一个给定的 API，包括正向测试和逆向测试
- 使用 SQL 发现并修改数据库信息以测试一个给定的场景
- 设计一个可以执行多种测试并收集测试结果的测试自动化工具
- 执行自动化测试并排除失效
- 编写测试计划 / 规格说明
- 编写包含了测试结果评估的测试总结报告

人际交往能力（软技能）可由下列活动体现：

- 演示一个落后进度的测试项目的信息
- 向一位坚信无缺陷的开发人员解释一份缺陷报告
- 培训一位同事
- 向管理层演示一个无效过程的问题
- 评审同事编写的测试用例，并向其表达自己的意见
- 面试一位同事
- 表扬一位开发人员

这张列表并不完整，所需的技能会随着环境和组织的变化而变化，上述的技能是比较常见的需求。通过建立有效的面试问题，并通过技能演示，测试经理可以评估候选人的技能，并确定他的强弱项。在建立起一套良好的评估机制后，还应用于所有现有的人员，以确定发展和培训的领域。

除了团队成员所需要的技能之外，测试经理还必须具备良好的沟通和外交技巧。测试经理必须能够缓和纷争，知道必要沟通应使用的正确方式，以及专注于建立和维护与组织的关系。

7.4 使测试适合组织

组织有许多方式使测试能适配组织结构。尽管在整个软件开发生命周期，质量是每个人的责任，一个独立的测试团队能为高品质产品作出显著的贡献。测试职能的独立性在实践中有着很多方式，如下列清单所示，测试的独立性由从低到高排序：

- 没有独立测试人员
  - 在这种情况下没有独立性可言，开发人员测试自己写的代码
  - 开发人员在时间允许的情况下，会确定代码是否按照预期运行，这也许会或不会符合实际需求
  - 开发人员可以快速修复任何发现的缺陷
- 测试由编写这部分代码以外的开发人员完成
  - 在开发人员和测试人员之间的独立性很低
  - 开发人员对其它开发人员的代码进行测试时，也许会不愿报告缺陷
  - 开发人员对测试的思维定势通常集中于正面的测试用例
- 由属于开发团队的测试员（或测试团队）进行测试
  - 测试员（或测试团队）或向项目经理报告，或向开发经理报告

  - 测试员的思维定势更多地关注去验证需求是否得到实现
  - 由于测试员是开发团队的一员，该测试员除了测试工作外也许还有开发工作
  - 测试员的经理也许更关心满足进度安排而不是达到质量目标
  - 在团队中，测试工作的地位可能比开发低
  - 测试员可能无权去影响质量目标或是否遵守这些目标
- 测试由来自业务组织、用户社区或其它非开发的技术组织的专业测试人员进行
  - 测试结果信息会被客观地报告给干系人
  - 团队的首要关注点是质量
  - 能力发展和培训都会针对测试
  - 测试被视为一条职业道路
  - 关注质量的测试团队能获得管理投入
- 外部测试专家执行特定类型的测试
  - 在一般性测试不足的特定领域中应用专门技术
  - 在需要专门技术的测试领域，如易用性、安全性、性能或其它专业化测试
  - 质量应是这些人的焦点，但他们的视角受限于其专业领域。性能专家未必能察觉一个性能良好的产品未能满足其功能需求
- 测试由公司外部的组织进行
  - 这种模式实现了测试人员和开发人员之间最大的独立性
  - 知识传播可能不足以让测试人员有能力进行测试
  - 需要清晰的需求和定义良好的沟通架构
  - 必须定期对外部组织的质量进行审计

此清单基于个体的典型关注点，但对于特定组织来说可能并不适用。职位和头衔不一定决定一个人的关注点。开发经理可以关注质量，因此成为优秀的测试经理。独立测试经理也许会向关注进度的管理层报告，继而倾向关心进度而不是质量。测试经理必须了解组织的目标才能确定测试团队在组织中的最佳位置。

开发和测试组织之间的独立性有很多不同的级别。重要的是要知道可能会有这样的折衷，独立性越强，越会被孤立，知识转移得越少。较低的独立水平也许会增加知识，但也可能引入有冲突的目标。独立的水平还可能由所使用的软件开发模型决定，例如，敏捷开发中测试员通常属于开发团队。

上述任一项都可能在某个组织中出现。测试工作可能既由开发组织也由独立的测试组织完成，最终可能由外部组织进行认可。重要的是理解测试每个阶段的职责和期望，并同时在保持进度和预算的限制下设定要求，以最大限度地提高成品的质量。

7.5 激励

可以有很多方法去激励从事测试工作的人员，这些方法包括：

- 对已完成的工作的肯定
- 通过管理层的审批
- 在同事和项目团队中受到尊重
- 委派更多责任和增加自主权
- 完成工作的适当回报（包括薪水、增加责任、认可）

有些项目影响可能会阻碍这些激励方式的实施。例如，在一个有着不可能期限的项目中，测试人员可能会非常辛勤地工作。他会尽可能地让团队专注于质量，花费额外的时间和工作量。但在外部影响下，产品却有可能在完成这些工作之前就需要交付了。尽管测试人员尽了最大的努力，最后的产品质量却很低

劣。当测试人员的贡献不能得到理解和衡量时，无论最终产品成功与否，都很容易对测试人员动力产生负面影响。

测试团队必须确保对适当的度量进行了跟踪，这些度量可以用来证实很好地完成了测试、缓解了风险、准确记录了结果的工作。如果不收集和发布这些数据，测试团队很可能因为没有收到对其工作的正面认可而变得消极。

认可不仅仅是无形的尊重和批准，还可通过提供晋升机会，增加责任和绩效奖励等形式。不过如果测试团队没有获得尊重，也不太可能获得这些机会。

当测试人员对于项目的增值作出了明显贡献时，就会获得认可和尊重。在一个项目中，最好的方式是在项目初期就让测试人员参与进来，并持续参与在整个生命周期中。随着时间的推移，测试人员将会以其对项目的积极贡献赢得其它项目成员的尊重。同时也应对这些贡献在降低质量成本和风险缓解方面进行量化。

测试经理在激励测试团队成员，并在组织中支持测试团队的工作上扮演着重要角色。测试经理应该意识到每个测试人员所取得的成就，且必须公正和诚实地评估错误。一个公正且一视同仁的测试经理将会成为激励团队成员的榜样。

7.6 沟通

测试团队主要通过以下方式进行沟通：

- 测试产品的文档——测试策略、测试计划、测试用例、测试总结报告、缺陷报告等
- 就已评审的文档提供反馈——需求、功能规格说明、用例、组件测试文档等
- 信息收集和传播——与开发人员、其它测试组员、管理层的互动等

测试人员和其它干系人之间的沟通必须是专业、客观且有效的，只有这样才能建立并维护对测试团队的尊重。在向他人提供工作产品反馈，尤其是建设性反馈时，必须要保持客观和老练。除此之外，沟通的重点应在于实现测试目标，以及提高产品和软件系统生产过程的质量。

测试经理有着广泛的沟通对象，包括用户、项目组成员、管理层、外部测试团队以及客户。和目标对象的沟通必须是有效的。例如，为开发团队而做的报告，包括了缺陷的数量和严重性及其趋势，而面对董事会的管理报告来说未免过于详细且不合适。在任何沟通，尤其在演示中，理解发布的信息，信息接受的方式，需要通过哪些解释来让信息可被接受是很重要的。由于测试经理通常需要演示项目状态信息，他需要把握这些信息适当的细致程度（例如，管理者通常更希望看到缺陷趋势，而不是单个缺陷），并以表述明确、易于理解（例如简表和多色图表）的方式演示。有效的沟通能抓住听众的注意力，同时传达正确的信息。每份演示文档都需要经过测试经理的评审，以帮助提升质量和质量过程。

测试经理不仅仅与部门外部人员沟通（对外沟通），还有一部分重要的工作是在测试团队内进行有效沟通（对内沟通），传递正常测试过程中的最新消息、指示、优先级变动及其它标准的信息。测试经理还有可能对组织管理链中上游（对上沟通）或下游（对下沟通）的特定人员进行沟通。无论沟通的方向如何，规则都是一样。应该按照听众的不同进行相应的沟通，沟通应有效地传递信息，并确认听众是否理解。

测试经理必须掌握多种沟通手段。很多信息是通过电子邮件、口头互动、正式或非正式会议、正式或非正式报告，甚至通过测试管理工具如缺陷管理工具来进行沟通。所有的沟通都应当专业和客观。即使在最紧急的沟通中，对质量和内容的校对都是必须的。书面的沟通往往会保存得比实际项目时间更长久。因此，重要的是测试经理能编写高质量的专业文档，而此文档适应公司对质量的要求。

8. 参考资料

8.1 标准

- [BS7925-2] BS 7925-2, Software Component Testing Standard.
  第二章
- [FDA21] FDA Title 21 CFR Part 820, Food and Drug Administration, USA
  第二章
- [IEEE829] IEEE Standard for Software and System Test Documentation
  第二章和第四章
- [IEEE1028] IEEE Standard for Software Reviews and Audits
  第二章
- [IEEE1044] IEEE Standard Classification for Software Anomalies
  第四章
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  第二章和第四章
- [ISO12207] ISO 12207, Systems and Software Engineering, Software Life Cycle Processes
  第二章
- [ISO15504] ISO/IEC 15504, Information Technology - Process Assessment
  第二章
- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)
  第二章和第四章
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  第二章

8.2 ISTQB® 文档

- [ISTQB®_AL_OVIEW] ISTQB® 高级大纲概述，版本 1.0
- [ISTQB®_ALTM_SYL] ISTQB® 高级大纲 - 测试经理，版本 1.0
- [ISTQB®_ALTTA_SYL] ISTQB® 高级大纲 - 技术测试分析，版本 1.0
- [ISTQB® ETM SYL] ISTQB® 专家级大纲 – 测试管理, 版本 1.0
- [ISTQB®_FL_SYL] ISTQB® 基础级大纲，版本 2011
- [ISTQB®_GLOSSARY] 软件测试标准术语表，版本 2.2, 2012
- [ISTQB® ITP SYL] ISTQB® 专家级大纲 – 测试过程改进, 版本 1.0

8.3 商标

本文档中使用了下列注册商标和服务标志：

- CMMI®, IDEAL℠, ISTQB®, ITIL®, PRINCE2®, TMMi®, TPI Next®
- CMMI 已由卡耐基梅隆大学在美国专利商标局注册
- IDEAL 是卡耐基梅隆大学软件工程研究所的服务标志
- ISTQB® 是国际软件测试认证委员会的注册商标
- ITIL 是政府商务办公室的注册商标和注册社区商标，并已在美国专利商标局注册
- PRINCE2 是内阁办公室的注册商标
- TMMi 是 TMMi Foundation 的注册商标
- TPI Next 是 Sogeti Nederland B.V.的注册商标

8.4 书籍

[Black03]: Rex Black, “Critical Testing Processes,” Addison-Wesley, 2003, ISBN 0-201-74868-1

[Black09]: Rex Black, “Managing the Testing Process, third edition,” John Wiley & Sons, 2009, ISBN 0-471-22398-0

[Black11]: Rex Black, Erik van Veenendaal, Dorothy Graham, “Foundations of Software Testing,” Thomson Learning, 2011, ISBN 978-1-84480-355-2

[Craig02]: Rick Craig, Stefan Jaskiel, “Systematic Software Testing,” Artech House, 2002, ISBN 1-580-53508-9

[Crispin09]: Lisa Crispin, Janet Gregory, “Agile Testing: A Practical Guide for Testers and Agile Teams”, Addison-Wesley, 2009, ISBN 0-321-53446-8

[de Vries09]: Gerrit de Vries, et al., “TPI Next”, UTN Publishers, 2009, ISBN 90-72194-97-7

[Goucher09]: Adam Goucher, Tim Riley (editors), “Beautiful Testing,” O’Reilly, 2009, ISBN 978-0596159818

[IDEAL96]: Bob McFeeley, “IDEAL: A User's Guide for Software Process Improvement,” Software Engineering Institute (SEI), 1996, CMU/SEI-96-HB-001

[Jones07]: Capers Jones, “Estimating Software Costs, second edition,” McGraw-Hill, 2007, ISBN 978-0071483001

[Jones11]: Capers Jones and Olivier Bonsignour, “Economics of Software Quality,” Pearson, 2011, ISBN 978-0132582209

[McKay07]: Judy McKay, “Managing the Test People,” Rocky Nook, 2007, ISBN 978-1933952123

[Musa04]: John Musa, “Software Reliability Engineering, second edition,” Author House, 2004, ISBN 978-1418493882

[Stamatis03]: D.H. Stamatis, “Failure Mode and Effect Analysis,” ASQC Quality Press, 2003, ISBN 0-873-89300

[vanVeenendaal11] Erik van Veenendaal, “The Little TMMi,” UTN Publishers, 2011, ISBN 9-490-986038

[vanVeenendaal12] Erik van Veenendaal, “Practical Risk-based Testing,” UTN Publishers, 2012, ISBN 978-9490986070

[Whittaker09]: James Whittaker, “Exploratory Testing,” Addison-Wesley, 2009, ISBN 978-0321636416

[Wiegers03]: Karl Wiegers, “Software Requirements 2,” Microsoft Press, 2003, ISBN 978-0735618794

8.5 其它引用

下列参考地址指向英特网上可用的信息。当本高级大纲出版时已检查过这些引用。

http://www.istqb.org
http://www.cstqb.cn
http://www.sei.cmu.edu/cmmi/
http://www.tmmi.org/
http://www.tpinext.com/

9. 索引

:DO-178B 标准, 24
CMMI, 40, 54
CTP, 53, 54, 56
ED-12B 标准, 24
GNU 通用公共许可, 58
IDEAL, 54
IEC 61508 标准, 24
IEEE, 40
IEEE 1028, 40
IEEE 829, 40
ISO, 40
ISO 25000, 22
ISO 9126, 22
ITIL, 40
PMI, 40
PRINCE2, 40
STEP, 53, 54, 56
TMMi, 53, 54
TPI Next, 52, 53, 54, 55
V 模型, 18
暴露的成本, 25
标准
  BS-7925-2, 40
  CMMI, 52
  DO-178B, 40
  ED 12B, 40
  TMMI, 52
测试策略, 29
测试策略, 9, 15
测试策略, 30
测试成熟度模型集成(TMMi), 55
测试度量, 34
测试方法, 15, 31
测试方针, 29
测试方针, 15
测试分析, 11
测试改进过程, 53
测试估算, 15, 33
测试管理, 15, 17
测试规程, 8
测试会话, 21
测试级别, 15, 31
测试计划, 8, 9, 15, 18, 23
测试计划、监督和控制, 9
测试监督, 15
测试脚本, 8
测试结束, 8
测试结束活动, 14
测试控制, 8, 15
测试日志, 8
测试设计, 8, 12
测试实施, 12
测试实现, 8
测试条件, 8, 15
测试条件分析, 27
测试团队动力, 65
测试用例, 8
测试章程, 21
测试执行, 8
测试主管, 15
测试总结报告, 8
测试组长, 15
产品度量, 34
产品风险, 9, 15, 21
产品质量风险, 21
出口准则, 8
戴明改进循环, 53
到岸外包测试, 39
迭代, 18
定量的价值, 38
定性的价值, 38
度量, 35, 68
度量项, 9
对内沟通, 68
对评审进行管理, 43
对上沟通, 68
对外沟通, 68
对下沟通, 68
反回归测试策略, 30
非正式评审, 41
分布式测试, 39
分析型策略, 29
风险, 15, 21
风险分析, 15, 23
风险管理, 15, 24
风险缓解, 15, 23
风险级别, 15, 22, 23
风险评估, 15, 22
风险识别, 15, 22

风险优先级数值, 23
风险总分, 23
服务级别协议 (SLAs), 10
符合过程或标准的策略, 29
负面影响, 68
改进测试过程, 54
概率, 22
个人技能, 64
根本原因, 47, 51
工具度量, 62
工具生命周期, 61
工作分解结构(WBS), 33
沟通, 68
故障树分析 (FTA), 26
关键测试过程 CTP, 56
关键测试过程(CTP), 52
管理评审, 41
管理正式评审, 45
广度优先, 24
过程度量, 34
过程改进的类型, 53
回归测试, 24
基本的测试过程, 9
基于方法的策略, 29
基于风险的测试, 9, 15, 21, 22, 24, 25, 26, 27, 51
基于经验的测试, 20
基于模型的策略, 29
基于需求的测试, 27
激励, 67
级别测试计划, 31
级别测试计划, 15
计划风险, 21
技能评估, 64
技术风险, 22
技术干系人, 26
技术评审, 41
假阴性结果, 47, 48
假阳性结果, 47
检测成本, 38
阶段遏制, 47
阶段抑制, 48
开源工具, 58
客户产品集成测试, 20
宽带德尔菲法, 15, 33
离岸外包测试, 39
敏捷, 10, 13, 18, 26, 31, 67
内部失效成本, 38
评审, 41, 42
评审度量, 45
评审计划, 41
评审员, 41
瀑布模型, 18
歧义评审, 27
缺陷, 47, 48
缺陷报告内容, 51
缺陷分类委员会, 47, 49
缺陷管理委员会, 49
缺陷生命周期, 48
确认测试, 24
人际交往能力, 64
人为错误, 48
人员度量, 34
人员技能, 63
软技能, 66
深度优先, 24
审查, 41
审计, 41
失效, 47, 48
失效模式和影响分析 (FMEA), 25
使测试适合组织, 66
使用 STEP 改进测试过程, 56
使用 CTP 改进测试过程, 56
使用 TMMi 改进测试过程, 55
使用 TPI Next 改进测试过程, 55
使用概况, 27
顺序模型, 18
探索性测试, 20
特性交互测试, 19
投资回报 (ROI), 59
团队构成, 63
外部测试专家, 67
外部失效成本, 38
危害分析, 25
系统集成测试, 19
系统性测试和评价过程 (STEP), 52
项目度量, 34
项目风险, 15, 21
严重程度, 47, 50
业务干系人, 26
异常, 47, 48
因果图, 27
影响, 22
应对型策略, 30
应对型方法, 27
硬技能, 66

硬件-软件集成测试, 19
优先级, 47, 50
预防成本, 38
运行概况, 29
质量风险, 15, 21
质量风险分析, 21
质量功能展开 (QFD), 26
主测试计划, 31
主测试计划, 15
主持人, 41
咨询式策略, 30
走查, 41