认证测试工程师

基础级扩展大纲
敏捷测试工程师

2014 版
（中文版 20151212）

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

英文版权说明

基础级扩展-敏捷测试工程师工作组成员：Rex Black（主席）, Bertrand Cornanguer（副主席）, Gerry Coleman（学习目标负责人）, Debra Friedenberg（考试负责人）, AlonLinetzki（业务与市场负责人）, TauhidaParveen（编辑）, 以及 Leo van der Aalst（开发负责人）

作者：Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, IstvanForgacs, AlonLinetzki, Tilo Linz, Leo van der Aalst, Marie Walsh,Stephan Weber.

内部评审专家：Mette Bruhn-Pedersen, Christopher Clements, Alessandro Collino, Debra Friedenberg, Kari Kakkonen, BeataKarpinska, Sammy Kolluru, Jennifer Leger, Thomas Mueller, TuulaPääkkönen, MeilePosthuma, Gabor Puhalla, Lloyd Roden, Marko Rytkönen, Monika Stoecklein-Olsen, Robert Treffny, Chris Van Bael, and Erik van Veenendaal; 2013-2014。

中文版权说明

修订历史

版本	日期	备注
大纲 v0.1	2013 年 7 月 26 日	独立章节
大纲 v0.2	2013 年 9 月 16 日	工作组对 v01 版本进行评审
大纲 v0.3	2013 年 10 月 20 日	工作组对 v02 版本进行评审
大纲 v0.7	2013 年 12 月 16 日	v03 版本进行 Alpha 评审
大纲 v0.71	2013 年 12 月 20 日	工作组升级 v07 版本
大纲 v0.9	2014 年 1 月 30 日	Beta 版
大纲 2014	2014 年 5 月 31 日	发布版
大纲 2014	2014 年 9 月 30 日	文稿排版小错误修改

中文版本	日期	备注
中文版 v0.1	2014 年 12 月 12 日	Beta 版
中文版 v0.5	2015 年 08 月 16 日	小组评审版
中文版 v1.0	2015 年 12 月 12 日	正式发布版

目录

致谢

本文由国际软件测试认证委员会基础级工作组编写。

敏捷扩展组感谢评审小组以及各国委员会对他们的建议和输入。

基础级敏捷扩展大纲完成时，敏捷扩展工作组有以下成员：Rex Black（主席）, Bertrand Cornanguer（副主席）, Gerry Coleman （学习目标负责人）, Debra Friedenberg（考试负责人）, AlonLinetzki（业务与市场负责人）, TauhidaParveen （编辑）, and Leo van der Aalst（开发负责人）

作者：Rex Black, Anders Claesson, Gerry Coleman, Bertrand Cornanguer, IstvanForgacs, AlonLinetzki, Tilo Linz, Leo van der Aalst, Marie Walsh, and Stephan Weber.

内部评审专家：Mette Bruhn-Pedersen, Christopher Clements, Alessandro Collino, Debra Friedenberg, Kari Kakkonen, BeataKarpinska, Sammy Kolluru, Jennifer Leger, Thomas Mueller, TuulaPääkkönen, MeilePosthuma, Gabor Puhalla, Lloyd Roden, Marko Rytkönen, Monika Stoecklein-Olsen, Robert Treffny, Chris Van Bael, and Erik van Veenendaal.

工作组同样感谢参与基础级敏捷扩展大纲评审、评论和投票的各国委员会以及敏捷专家社区的人员：DaniAlmog, Richard Berns, Stephen Bird, Monika Bögge, Afeng Chai, Josephine Crawford, TiborCsöndes, Huba Demeter, Arnaud Foucal, Cyril Fumery, KobiHalperin, Inga Hansen, Hanne Hinz, Jidong Hu, Phill Isles, Shirley Itah, Martin Klonk, Kjell Lauren, Igal Levi, RikMarselis, Johan Meivert, Armin Metzger, Peter Morgan, Ninna Morin, Ingvar Nordstrom, Chris O’Dea, Klaus Olsen, IsmoPaukamainen, Nathalie Phung, Helmut Pichler, Salvatore Reale, Stuart Reid, Hans Rombouts, Petri Säilynoja, SoileSainio, Lars-Erik Sandberg, Dakar Shalom, JianShen, Marco Sogliani, LucjanStapp, YaronTsubery, Sabine Uhde, Stephanie Ulrich, TommiVälimäki, Jurian Van de Laar, Marnix Van den Ent, António Vieira Melo, WenyeXu, Ester Zabar, WenqiangZheng, Peter Zimmerer, StevanZivanovic, and Terry Zuo.

本文于 2014 年 5 月 31 日由 ISTQB® 大会正式批准发布。

参加本大纲中文版翻译的 CSTQB 专家有（按姓氏拼音排序）：柴阿峰、丰俊、胡继东、贺炘、李华北、柳毓龙、陆怡颐、马均飞、申健、施彦臣、王宇、于长青、左平、朱少民、翟宏宝（组长）、郑文强。

参加本大纲中文版评审的 CSTQB 专家有（按姓氏拼音排序）：刘晓更、任亮、徐文叶、周震漪

0. 大纲简介

0.1 本文档的目的

本大纲是 ISTQB-FL（国际软件测试工程师认证—基础级）的敏捷测试工程师模块的基础性文档。ISTQB®（国际软件测试认证委员会）提供本大纲用于下列用途：

- 对各国认证委员会，（本大纲可用于）翻译成本地语言和认可培训机构。各国认证委员会可以按照本国语言对大纲做必要的修正，以满足本地出版发行的需要。
- 对各国考试委员会，（本大纲可用于）使用本地语言生成试题，以符合大纲学习目标。
- 对各培训机构，（本大纲可用于）制作课件，并确定适当的教学方法。
- 对认证学员，（本大纲可用于）指导准备考试（作为独立或培训课程的一部分）。
- 对于国际软件和系统工程社区，（本大纲可用于）推进软件和系统测试行业，作为书籍和文章的基础。

ISTQB®可以允许其他机构或个人出于上述之外的其他目的使用本大纲，但是在使用之前需要获得书面许可。

0.2 概览

基础级敏捷测试工程师概览[ISTQB_FA_OVIEW]文档包含如下内容：

- 本大纲的商业价值；
- 本大纲的摘要；
- 不同大纲之间的关系；
- 认知水平的描述（K 级）；
- 附录。

0.3 考试范围内的学习目标

学习目标支持商业价值并用于创建考试以实现测试工程师基础级认证—敏捷测试工程师认证。一般而言，本大纲在 K1 级别的所有内容都是考试范围。也就是说，学员将能识别，记住和回忆本大纲的术语或概念。具体在 K1，K2，和 K3 水平的学习目标会在各个章节的开始部分标识出来。

1. 敏捷软件开发 – 150 分钟

关键词

敏捷宣言（Agile Manifesto），敏捷软件开发（Agile software development），增量开发模型（incremental development model），迭代开发模型（iterative development model），软件生命周期（software lifecycle），测试自动化（test automation），测试依据（test basis），测试驱动开发（test-driven development），测试准则（test oracle），用户故事（user story）

敏捷软件开发的学习目标

1.1 敏捷软件开发基础

FA-1.1.1	（K1）	基于敏捷宣言，复述敏捷软件开发的基本概念。
FA-1.1.2	（K2）	理解“全团队方式”的优点。
FA-1.1.3	（K2）	理解尽早和频繁反馈的好处。

1.2 敏捷方法的特点

FA-1.2.1	（K1）	复述敏捷软件开发方法。
FA-1.2.2	（K3）	与开发人员和用户代表协作编写可测试的用户故事。
FA-1.2.3	（K2）	理解回顾会议作为一种机制，是如何应用于敏捷项目的过程改进的。
FA-1.2.4	（K2）	理解持续集成的目的和用途。。
FA-1.2.5	（K1）	知道迭代计划和发布计划的区别，以及测试人员如何在这些活动中提供价值。

1.1 敏捷软件开发基础

测试人员在敏捷项目中的工作方法和传统项目中是不同的。测试人员必须理解支撑敏捷项目的价值观和原则，以及测试人员如何与开发人员和业务代表一起作为全团队方式的一个组成部分。

敏捷项目成员彼此之间越是在早期和频繁的沟通，越有利于尽早移除缺陷并开发出高质量的产品。

1.1.1 敏捷软件开发和敏捷宣言

2001 年，一批专家在对一系列轻量级软件开发中广泛使用的方法进行了讨论之后，同意将一些具有共性的价值观和原则汇集成敏捷软件开发宣言，或称为敏捷宣言[Agilemanifesto]。敏捷宣言包含如下四条价值观：

- 个体与交互胜过流程与工具
- 可工作的软件胜过详尽的文档
- 与客户合作胜过合同谈判
- 响应变更胜过遵循计划

敏捷宣言认为，尽管右边的项有价值，但是左边的具有更高价值。

个体和交互

敏捷开发是以人为中心的。由人组成的团队构建软件，通过持续的沟通和交互，而不是通过对流程、工具的依赖，来达到团队的高效工作。

可工作的软件

从客户角度来说，可工作的软件比详细的文档更有用和有价值，它提供了向开发团队快速反馈的机会。另外，由于可工作的软件（虽然减少了功能性）可以在开发生命周期的早期可用，敏捷开发可以带来巨大的上市时间优势。因此，敏捷开发特别适用于问题和（或）解决方案不甚清晰的快速变化的商业环境，或在新问题域的业务创新。

客户合作

客户往往很难找到他们需要的特定系统。客户与开发团队的直接合作可以改进理解客户需要的可能。当然，与客户签订合同也是重要的，而与客户定期和紧密的合作可能让项目更容易成功。

响应变化

软件项目中的变化不可避免。业务运营环境、法规、竞争对手的活动、技术进步以及其他因素都会极大影响项目及其目标。开发过程必须适应这些因素。因此，在工作实践中灵活的拥抱变化比简单的遵循计划更重要。

原则

敏捷宣言的核心价值包含以下 12 条原则：

- 我们的最高目标是通过尽早的、持续的交付有价值的软件来满足客户。
- 即使到了开发的后期，也欢迎改变需求，敏捷过程驾驭变化来为客户创造竞争优势。
- 持续地交付可以工作的软件，交付的间隔可以从几个星期到几个月，交付的时间间隔越短越好。
- 在整个项目开发期间，业务人员和开发人员必须天天都在一起工作。

- 围绕被激励起来的个体来构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。
- 在团队内部，最具有效果并且富有效率的传递信息的方法，就是面对面的交流。
- 工作的软件是首要的进度度量标准。
- 敏捷过程提倡可持续的开发。责任人、开发人员和用户应该能够保持一个长期的、稳定的开发进度。
- 不断地关注优秀的技能和好的设计会增强敏捷能力。
- 简单，尽可能简化一切未完成工作的艺术是根本。
- 最好的构架、需求和设计出自于自组织的团队。
- 每隔一定时间，团队会在如何才能更有效地工作方面进行反省，然后相应地对自己的行为进行调整。

不同的敏捷方法论都提供了把这些价值和原则付诸于行动的具体实践。

1.1.2 全团队方式

全团队方式是指把所有具备相关知识和技能的个人都纳入团队，以确保项目成功。全团队方式需要包括客户代表和能够决定产品特性的其他业务干系人。团队应该相对较小，根据观察，一个成功团队的人数在 3-9 人。为促进团队之间交流和互动，一个全团队的工作地点最理想的情况下应该在同一办公区。全团队方式通过每日站会（具体见 2.2.1 节）来实施，每一个团队成员重点沟通自己的工作进展和阻碍工作的困难。全团队方式促进团队动力更加有效和高效。
在产品开发中使用全团队方式是敏捷开发的主要好处之一。它的好处包括：

- 促进团队的沟通和协作；
- 使团队的各种技能得到平衡，以利于整个项目；
- 使团队每个人承担质量职责。

在敏捷项目中，团队中的每个成员都对质量负责。全团队方式的精髓是，在产品开发的每一步中，开发人员、测试人员和业务代表都能一起协同工作。测试人员需要和开发人员与业务代表紧密协作，以保证产品能够达到质量级别。这包括支持和协助业务代表创建适合的验收测试、跟开发人员一起决定测试策略、确定测试自动化方法。通过这样的工作方式，测试人员能够把测试知识传递给团队中的其他成员，从而对产品的开发产生影响。

整个团队需参与产品特性的介绍、分析、评估的磋商和会议。包含开发人员、测试人员和业务代表共同参加所有特性讨论的概念被称为“三驾马车” [Crispin08]。

1.1.3 尽早和频繁的反馈

敏捷项目的迭代周期短，使项目团队能在整个开发生命周期中尽早和持续的接收产品质量方面的反馈。持续集成就是提供快速反馈的方法之一（具体见 1.2.4 节）。

在一般的顺序开发模型中，客户往往只有在项目接近结束的时候才能够看到产品。这时，开发团队再去有效的处理客户的任何问题通常是太晚了。如果敏捷团队在项目进行中可以经常地收到客户的反馈，他们就能在产品开发进程中融入大部分的客户新需求。尽早和频繁的反馈能使敏捷团队重点关注具有最高商业价值或相关风险的特性，这些特性需要首先交付给客户。同时因为团队成员的能力是透明的，这也使得团队管理变的容易，例如，在一个冲刺或迭代周期中我们能够完成的工作是多少？什么能帮助我们进展的更快？什么会阻碍我们继续？

尽早和频繁的反馈能带来以下的好处：

- 避免需求的误解。这些误解可能在开发后期才能发现，而那时再修复问题就要花费更高的成本。
- 澄清客户对特性的要求，并尽早的完成特性的开发以提供给客户使用。这样，产品就能更好的反映客户想要的内容。
- 能尽早发现（通过持续集成）、隔离、解决质量问题。
- 使敏捷团队了解自身的生产率和产品交付能力等信息。
- 为项目注入持续动力。

1.2 敏捷方法的内容

不同的组织会采用不同的敏捷方法，但大多数组织采用的敏捷方法有共同之处。这些共同点包括：创建用户故事、回顾、持续集成、每个迭代以及整个发布的计划。本章节对一些敏捷方法进行了描述。

1.2.1 敏捷软件开发方法

现今使用的敏捷方法有好几种，每一种都用不同的方式实现敏捷宣言里的价值和原则。本文讨论其中三种具有代表性的敏捷方法：极限编程（XP）、Scrum 和看板。

极限编程

极限编程（XP）最早由 Kent Beck 提出[Beck04]，是一种通过某些价值、原则和开发实践来描述软件开发的敏捷方法。

极限编程包含五个价值要素来指导开发：沟通、简单、反馈、勇气和尊重。

作为附加的指导，极限编程描述了一系列原则，包括：人性化、经济性、互惠互利、自相似性（重用现有的解决方案）、持续改进、团队成员的多样性、不断反省、均匀高效的工作流、失败作为机遇、避免冗余、挑战缺陷、重视质量、小步迭代和接受责任。

另外，极限编程还描述了 13 个基本的实践：坐在一起、全团队方式、信息化的工作空间、充满活力的工作、结对编程、用户故事、周循环、季度循环、轻松的工作、十分钟构建、持续集成、测试先于编程和增量设计。

现在的很多敏捷软件开发方法都受到了极限编程的价值和原则的影响，例如 Scrum 敏捷团队就经常采用 XP 的实践。

Scrum

Scrum 是一个敏捷管理框架。它包含以下组成要素和实践[Schwaber01]：

- 冲刺（Sprint）：Scrum 把项目分为若干个固定长度（通常每个迭代周期 2-4 周）的迭代（叫做 Sprints）。
- 产品增量（Prouct Incrementd）：在一个迭代周期中完成一个可发布或可交付的产品（叫做一个增量）。
- 产品待办列表（Product Backlog）：产品负责人管理一个已经划分优先级的产品条目（叫做产品待办列表），该列表在一个迭代周期结束后需要更新（叫做列表细化）。
- 冲刺待办列表（Sprint Backlog）：在一个迭代周期开始时，Scrum 团队需要从产品待办列表中选择一些高优先级的条目放入一个较小的叫做冲刺待办列表内。这里是 Scrum 团队做出选择，而不是产品负责人做出选择。所以，这个选择遵守的原则是“拉”，而不是“推”。

- 完成的定义（Definition of Done）：为了确保每个冲刺结束的时候有一个潜在可发布产品，Scrum 团队讨论并定义冲刺完成的合适准则。团队的讨论可以加深团队对列表项以及产品需求的理解。
- 时间盒（Timeboxing）：只有当 scrum 团队希望把某些任务、需求和特性在某一个冲刺中实现，他们才需要把这些功能放在冲刺待办列表中。如果某些功能在一个冲刺中实现不了，scrum 团队应把它们从本冲刺移回到产品待办列表中。时间盒的适应范围不仅包括任务，对其他的场景（例如，规定会议开始和结束时间点）也同样适用。
- 透明性（Transparency）：开发团队通过每天的例会（叫做每日 Scrum）来汇报和更新冲刺的状态。这使得当前冲刺的内容和进展，包括测试结果，对团队、管理层和其他感兴趣的人员来说都是可见的，例如，开发团队可以在白板上展现冲刺进展。

Scrum 定义了以下三种角色：

- Scrum Master（SM）：确保 Scrum 实践和规则能够实施和遵守，并解决任何可能阻止团队遵守实践和规则的违规、资源问题和其他阻碍因素。SM 不是团队的领导，而是一个教练。
- Product Owner（PO）：代表客户，负责对产品待办列表的内容进行收集、维护和排优先级。产品负责人不是团队领导。
- Development Team ：开发和测试产品。开发团队是一个自组织的团队，团队中没有领导，整个团队共同决策。另外团队具有交叉职能 （具体见 2.3.2 节和 3.1.4 节）。

跟极限编程截然不同，Scrum 没有对具体的软件开发技术做具体的要求与限制（比如测试先于编程），另外，Scrum 也没有对在一个 Scrum 项目中如何完成测试提供指导。

看板

看板[Anderson 13]，是一种有时会用于敏捷项目中的管理方法。它的主要目的是在一个供应链内构建可视化的工作流并优化。看板用了以下三个工具：

- 看板图：需要管理的供应链通过看板图来达到可视化。每一列显示了一个工位，它由一系列相关的活动组成，比如开发或测试。将要产出的项或需要处理的任务用标签加以标示，在看板上从左到右移动的通过各工位。
- 进行中的工作数限制：严格限制并行处理的任务数量。控制每个工位以及/或整个看板图的最大可允许标签数。一旦某个工位有空闲的工作容量，成员可以从前一道工序领取看板标签。
- 交货期：看板通过最小化价值流的平均交货期来优化连续的任务流。

看板和 Scrum 有一些相似性。在这两个框架中，可视化的活动任务（比如放在一个公共的白板上）提供了任务内容和进展的透明性。没有排期的任务停留在待办列表中，一旦有新的空间可用（生产能力），任务就会移动到看板图上。

迭代或冲刺在看板中是可选的。看板处理允许一项一项的发布交付项，而不是作为发布的一部分。时间盒作为同步机制也是可选的，而在 Scrum 中时间盒在一个冲刺中同步所有的任务。

1.2.2 协作用户故事的创建

大部分项目失败的主要原因在于规格说明的缺乏。规格说明的问题可能来自于用户缺乏对他们真实需要的了解，或者是缺乏对系统的全局观，以及在沟通过程中传达了不需要的或不一致的、甚至是错误的特性。在敏捷开发中，用户故事用于从开发人员、测试人员和业务代表的视角来捕获需求。在顺序开发中，

这种特性的共同价值观是在需求编写完成后，通过正式评审来实现；在敏捷开发中，这种特性的共同价值观是在需求编写过程中通过频繁的非正式评审来完成。

用户故事必须处理功能和非功能性特性。每一个用户故事都包含这些特性的验收准则。这些准则通常是在业务代表、开发人员以及测试人员的共同协作下定义出来的。它们提供给开发人员和测试人员对业务代表将要确认的这些特性额外的了解。当一系列的准则满足时，敏捷团队就可以宣称一个任务完成。

通常，测试人员具有独特的视角，通过识别遗漏的细节或非功能性需求帮助改进用户故事。测试人员通过向业务代表提出开放式问题，提出测试用户故事的方法并确认验收准则来帮助改进用户故事。

以作者的身份提供用户故事协作，可以通过采用头脑风暴或思维导图等技术进行。测试人员可以采用 INVEST 技术：

- 独立的 （Independent）
- 可讨论的 （Negotiable）
- 有价值的 （Valuable）
- 可估算的 （Estimable）
- 小的 （Small）
- 可测试 （Testable）

根据[Jeffries00]提出的 3C 概念，一个用户故事是以下 3 个元素的有机联合体：

- 卡片（Card）：卡片是用来描述用户故事的物理介质。卡片描述了需求，紧急性，期望的开发和测试周期，以及故事的验收准则。由于将要用于产品待办列表中，用户故事描述应尽可能地精确。
- 对话（Conversation）：对话解释了软件将会如何使用。对话可以是书面的或口头的。测试人员拥有与开发人员和业务代表不同的观点[ISTQB_FL_SYL]，可以为交换想法、观点和经验带来有价值的输入。对话始于发布计划阶段，并一直持续到用户故事被排入开发进程中。
- 确认（Confirmation）：在对话中讨论的验收准则被用于确认用户故事的完成。这些验收准则可能跨越多个用户故事。正正向和逆向的测试应该被用于覆盖验收准则。在确认过程中，各类参与者扮演测试人员的角色。这可能包括开发人员，也包括关注性能、安全性、互操作性以及其它质量特性的专家。为了确认一个用户故事的完成，已定义的验收准则应该被测试，并显示是被满足的。

敏捷团队有不同的方法来文档化他们的用户故事。不论采用哪种方法来文档化用户故事，文档内容应该是简洁的，充分的以及必要的。

1.2.3 回顾

在敏捷开发中，回顾是指在一个迭代结束后举行的会议，该会议讨论哪些是成功的、哪些需要改进以及如何在未来的迭代里整合这些改进并继续保持成功。回顾会议的主题通常包括：过程、人员、组织、关系和工具。当有适当的后续活动发生，定期举行回顾会议对开发和测试的自组织和持续改进是重要的。

回顾可能会产生关注于测试有效性、测试效率、测试用例的质量和团队满意度等测试相关的改进决定。他们也可能会处理应用程序、用户故事、特性或系统接口的可测试性。缺陷（defect）的根本原因分析可以驱动测试和开发的改进。通常，在每个迭代中团队应只实施少量的改进。这允许以一个持续的步伐来连续的改进。

回顾的时间和组织形式依赖于团队所采用的敏捷方法。业务代表和团队作为参与者参加每一个回顾，而会议组织者则组织并确保会议的开展。在某些情况下，团队可能邀请其他参与人员到会。

在回顾中，测试人员应该发挥重要的角色。测试人员作为团队的一员，在回顾中可以带来独特的观点（见[ISTQB_FL_SYL],1.5 章节）。测试活动在每一个冲刺（Sprint）中进行，对于成功功不可没。所有的团队成员，包括测试和非测试人员，都可以提供测试和非测试活动的输入。

回顾必须在相互信任的专门环境下进行。成功回顾的标志与在基础级大纲[ISTQB_FL_SYL] 3.2 节讨论的其他任何评审一样。

1.2.4 持续集成

在每一个冲刺（Sprint）结束时，产品增量的交付需要提供可靠的、能工作的并已集成的软件。持续集成通过合并构成软件的所有更改和集成所有更改的组件来应对这种挑战，至少每天一次。配置管理、编译、软件构建、部署和测试都打包在一个单独的、自动化的、可重复的过程中。由于开发人员不断地集成他们的工作，不断地构建并不断的测试，代码里的缺陷被更快地发现。

随着开发人员的编码、调试以及提交代码到一个共享的源代码库，一个持续集成过程包括以下自动化的活动：

- 静态代码分析：执行静态代码分析并报告结果；
- 编译：编译并链接代码，生成可执行文件；
- 单元测试：执行单元测试，检查代码覆盖并报告测试结果；
- 部署：安装构建到测试环境中；
- 集成测试：执行集成测试并报告结果；
- 报告（面板 Dashboard）：将这些活动的状态发布到一个公共可见的地方或邮件发送状态给整个团队。

基于每天的自动构建和测试过程可以更早、更快地发现集成的错误。持续集成允许敏捷测试人员定期的运行自动测试，在某些情况下，甚至会作为持续集成过程的一部分，并且就代码质量提交快速的反馈给团队。这些测试结果对所有团队成员可见，特别是当自动报告集成到这个过程中的时候。自动回归测试可以在迭代中持续进行。好的自动回归测试能够覆盖尽可能多的功能，包括在前期迭代中交付的用户故事。在自动回归测试中，好的覆盖有助于大型集成系统的构建和测试。当回归测试自动化后，敏捷测试人员可以释放出精力来手工测试新特性、实施的变更以及缺陷修复的确认测试。

除自动测试外，使用持续集成的组织通常使用构建工具（Build Tools）来实现持续的质量控制。除了运行单元测试和集成测试，这些工具可以运行其他静态和动态测试，测量和分析性能，从源代码中提取和格式化文档并促进人工的质量保证过程。这种持续的质量控制应用的目标是为了改进产品的质量，通过替代传统的在所有开发完成后应用质量控制的实践来缩短交付时间。

构建工具可以和自动部署工具（automatic deployment tools）关联起来，它可以从持续集成或构建服务器上获取并部署到一个或多个开发、测试、临时环境甚至生产环境中。这减少了依靠专业人员或程序员在这些环境中手动安装发布的错误和延迟。

持续集成可以提供以下优点：

- 对集成的问题和变更的冲突能更早的识别和发现以及更容易进行根本原因分析；
- 定期向开发团队反馈代码是否能正常工作的信息；

- 目前在测试的版本与正在开发的版本相差不超过一天；
- 基于每个小的变更后的代码快速再测试和回归测试，可以降低开发人员代码重构相关的回归风险；
- 能保障每天的开发工作是基于稳健的基础之上的；
- 能看见是朝着完成一个产品扩展（增量）方向发展，这能鼓励开发人员和测试人员；
- 消除大爆炸式集成相关的进度风险；
- 为测试、演示或培训提供最新版本的贯穿整个冲刺的可执行软件；
- 减少重复的手工测试活动；
- 对测试和质量改善的相关决策提供快速反馈。

但是，持续集成也不是没有风险和挑战：

- 必须引入持续集成工具并加以维护；
- 必须定义并建立持续集成过程；
- 测试自动化需要额外的资源并且建立起来也相对复杂；
- 为了获得自动化测试的优势，必须提供尽可能大的测试覆盖率；
- 团队有时过于依赖单元测试，而执行太少的系统和验收测试。

持续集成需要使用工具，包括测试工具、构建过程的自动化工具、版本控制的工具。

1.2.5 发布和迭代计划

如同在基础级大纲[ISTQB_FL_SYL]中所描述的，计划是一项持续进行的活动，并且针对敏捷项目的生命周期也是同样适用的。在敏捷项目的生命周期中，有两种不同类型的计划活动：发布计划（release planning）和迭代计划（iteration planning）。

发布计划的最终目的是产品的发布，通常在具体项目开始之前的数个月就会开始筹划。发布计划定义甚至是重新定义了产品待办列表，可能还会将大的用户故事细化成一系列小的用户故事。发布计划为所有迭代的测试方法和测试计划提供了基准。发布计划是高层级别的文档。

在发布计划制订阶段，业务代表和项目团队成员合作创建用户故事并设定每一个故事的优先级（参见 1.2.2 节）。根据这些用户故事，团队会识别项目风险和质量风险并且对项目工作量做粗略的估算（参见 3.2 节）。

测试人员会参与到发布计划制订中，尤其在如下活动方面提供具有价值的输入：

- 定义可被测试的用户故事，包括验收准则；
- 参与到项目风险和质量风险的分析活动中；
- 根据关联的用户故事预估测试工作量；
- 定义必要的测试级别；
- 为发布而策划测试活动。

在发布计划创建完成后，指导第一次迭代活动的迭代计划就开始了。迭代计划着眼于一次单独的迭代过程的顺利完成并且主要的关注点在于迭代过程的冲刺待办列表。

在迭代计划过程中，项目团队根据发布过程的冲刺待办列表中优先级的高低，选择用户故事、进行用户故事的评估、对用户故事进行风险分析和对每一个用户故事进行工作估算。如果某个被选中的用户故事过于宽泛并且不能提供进一步的详尽阐述，项目团队有权拒绝把这一用户故事纳入到本次迭代的范围内，

并根据优先级选取下一个用户故事。业务代表必须回答项目团队成员提出的每一个关于用户故事的问题，这样可以帮助团队成员理解如何去实施和测试每一个用户故事。

用户故事选择的数量基于已经建立的团队速率（team velocity）和选择的用户故事估算的规模。在迭代的内容完成后，用户故事被分解成任务，这些任务会由团队成员来执行。

测试人员会积极参与迭代计划过程，并在如下方面提供具有价值的输入：

- 参与用户故事的详尽风险分析；
- 确定用户故事的可测试性；
- 为用户故事创建验收测试；
- 把用户故事分解成工作任务（特别是测试任务）；
- 为所有的测试任务进行测试工作量估算；
- 从功能性和非功能性角度对被测试系统的特性进行识别；
- 支持并参与到多个测试级别的测试自动化过程中。

随着项目的进行，发布计划可能被调整，包括要求修改产品待办列表中的单个用户故事。这些变更可能是由外部或内部的因素导致的。常见的内部因素有项目组交付能力、速率和技术性问题。典型的外部因素有新市场和新机会的发现、新加入的竞争者和对业务的威胁导致发布目的和日期的变化。另外，迭代计划也会在迭代过程中更改。例如，在任务预估时被评估为相对简单的某个用户故事在实际的工作中被证明是远超预期的复杂。

这些变更给测试人员带来了挑战。为了制定测试计划，测试人员必须从更高高度上理解版本发布，同时，如同在基础级大纲[ISTQB_FY_SYL]1.4 节所述的那样，为在迭代中进行测试开发，测试人员必须有足够的测试依据和测试准则。这些必备的信息必须尽早提供给测试人员，而且依照敏捷的原则，变更也必须要包括在内。在这进退两难的局面下，必须审慎的选择测试策略和撰写测试文档。关于更多敏捷测试挑战内容，请参见[Black09]的 12 章。

发布计划和迭代计划要关注的不仅是开发计划，还有测试计划。如下是着重要关注的测试相关问题：

- 测试的范围、以及此范围的延伸范围、测试目标和所做决策的理由；
- 执行各项测试活动的人员；
- 需要的测试环境和测试数据的准备、在什么时间节点需要它们。在项目之前或者进行中是否需要额外增加或者变更测试环境或测试数据；
- 功能性测试和非功能性测试所需的时间、顺序、依赖关系和前置条件等（例如以何种频率执行回归测试、需要依赖于别的功能和测试数据进行的测试），以及测试活动如何与开发活动关联或者有何种依赖；
- 需要关注的项目风险和质量风险（参见 3.2.1 节）。

值得注意的是，较大型的项目团队的工作量估算必须要包含足够完成必要测试活动的时间和工作量。

2. 基本的敏捷测试原则、实践和过程– 105 分钟

关键词

构建验证测试（build verification test），配置项（configuration item），配置管理（configuration management）

基本的敏捷测试原则、实践和过程的学习目标

2.1 传统测试和敏捷测试方法的不同

FA-2.1.1 （K2） 描述在敏捷项目和非敏捷项目中测试活动的不同。

FA-2.1.2 （K2） 描述在敏捷项目中开发和测试活动是如何集成的。

FA-2.1.3 （K2） 描述在敏捷项目中独立测试的角色。

2.2 敏捷项目中的测试状态

FA-2.2.1 （K2） 描述敏捷项目中如何运用各种工具和技术与各参与方沟通测试状态，包括测试进展和产品质量。

FA-2.2.2 （K2） 描述敏捷项目中通过多个迭代演进测试的过程，并解释为什么测试自动化对于管理回归风险是重要的。

2.3 敏捷团队中测试的角色和技能

FA-2.3.1 （K2） 理解敏捷团队中测试人员所需技能（人员技能、领域技能及测试技能）。

FA-2.3.2 （K2） 理解敏捷团队中测试人员的角色。

2.1 传统测试和敏捷测试方法的不同

如同在基础级大纲[ISTQB_FL_SYL]和[Black09]中所描述的，由于测试活动是和开发活动相关联的，所以在不同开发生命周期中的测试活动也是不同的。测试人员必须理解传统的生命周期模型（顺序模型如 V 模型，迭代模型如 RUP）和敏捷生命周期模型之间的区别才能更加有效和高效的开展测试工作。敏捷模型从测试和开发活动集成的方法不同而言，项目的工作产品、命名、各测试级别的入口准则和出口准则、使用的工具、以及如何进行有效的独立测试都是不同的。

测试人员应该记住，组织在其生命周期的实施中有很大的差异。敏捷生命周期（见 1.1 章节）观念的偏离可能表现为明智的定制和对实践的适应。能够适应给定项目环境的能力，包括实际遵循软件开发实践，是测试人员成功的关键因素。

2.1.1 测试和开发活动

传统的和敏捷的测试生命周期中重要的不同点之一就是短迭代，每个迭代的结果是向业务干系人交付有价值的可工作软件的特性。在项目的开始阶段，会有一个发布计划阶段。接着会有一系列的迭代序列。在每一个迭代开始时，会有一个迭代计划阶段。一旦迭代的工作范围被确立，选择的用户故事就会被开发，与系统进行集成并被测试。这些迭代是高度动态的，开发、集成和测试的各项活动会贯穿每个迭代的整个过程，并且有大量的并行和重叠。测试活动贯穿整个迭代，而不是作为最后的活动。

与传统项目的生命周期测试类似，测试人员、开发人员和业务干系人都在迭代过程的测试活动中有相应的角色。开发人员根据用户故事开发特性并且执行单元测试。测试人员随后再测试这些特性。业务干系人会在产品实现后对相应的用户故事进行测试。业务干系人可能会使用书面的测试用例进行测试，但是通常他们会为了尽快向开发团队提供反馈而对产品特性点进行相对简单的尝试和使用。

有些情况下，项目组会进行硬迭代（hardening iteration）或稳定迭代（stabilization iteration），其目的通常是为了解决还没有修改的缺陷，以及其它类型的技术问题，例如，为了快速进入市场而采用了一些临时的技术处理，在质量上有所妥协（技术债）。但是，最佳实践仍然是，直到一个特性点已经被集成到产品中并被测试通过后，我们才能认为这一特性点已经被最终交付了。把上一个迭代中未修复的缺陷添加到下一迭代的待办列表中，在下一迭代的开始阶段尽快修复它们也是一个很不错的最佳实践（通常称为“优先修复缺陷 fix bugs first”）。但同时这一做法也有它的弊端，额外的修复缺陷工作量导致这一迭代中总的工作内容较难被估算和计划，而且也会增加预估余下特性点交付时间的困难程度。尽管在某些情况下，交付发生在每个迭代结束时，但在迭代序列结束时，可能会执行一系列发布活动来确定软件已准备好交付。

当采用基于风险测试的测试策略时，在发布计划阶段，会有一个高级别风险分析，通常是由测试人员来主导分析的。当然，迭代计划通常都会包含识别和评估与该迭代过程相关的具体质量风险的活动。风险分析的结果会影响开发的顺序、测试特性的优先级和深度。并且它还会影响到每特性的测试工作量估算（参见 3.2 节）。

在某些敏捷实践（例如极限编程）中，会采用结对的方法。结对可以由两个测试人员共同测试同一个特性。结对也可能是一个开发人员结对一个测试人员对特性进行开发和测试。如果测试团队的组织形式是分散的，结对测试会相对困难些，不过过程和工具可以帮助解决分散结对的问题。关于分散工作的详细内容，请参见[ISTQB_ALTM_SYL]2.8 章。

测试人员也可以作为团队内部测试和质量的教练，在团队中分享测试知识和支持质量保证工作。这提高了整个集体的产品质量意识。

在许多敏捷团队中都有各个级别的测试的自动化，这就意味着测试人员需要花很多时间去创建，执行，监控和维护自动化测试和结果。由于大量使用测试的自动化，敏捷项目中更高比例的手动测试往往是使用基于经验和基于缺陷的技术，如：软件攻击、探索性测试和错误猜测（见[ISTQB_ALTA_SYL]第 3.3 节和第 3.4 节以及[ISTQB_FL_SYL]第 4.5 节）。开发人员将重点关注创建单元测试，而测试人员则应关注创建自动化集成、系统和系统集成测试。这会导致敏捷团队倾向于拥有很强技术和自动化测试背景的测试工程师。

一个核心的敏捷原则是变更会在项目的整个生命周期中发生。因此，轻量级的工作产品文档在敏捷项目中是受欢迎的。现有特性的变更是对测试有影响的，特别是对回归测试的影响。测试的自动化是管理测试变更工作量的一个方法。然而，重要的是，变更率不能超过项目团队处理与这些变更相关的风险的能力。

2.1.2 项目工作产品

与敏捷测试人员直接利益相关的项目工作产品通常可分为以下三类：

1. 面向业务的工作产品，其描述什么是必要的（例如：需求规格说明）以及如何使用它（例如：用户文档）；
2. 面向开发的工作产品，其描述了如何构建系统（例如：数据库实体关系图），实际实现的系统（例如：代码），或评估单个代码片段（例如：单元测试自动化）；
3. 面向测试的工作产品，其描述了如何测试系统（例如：测试策略和计划），实际测试系统（例如：人工和自动测试），或当前的测试结果（例如：在第 2.2.1 节讨论的测试面板）。

在一个典型的敏捷项目中，一个常规实践是避免产生大量的文档。相反，更多关注实现可工作的软件，并结合自动化测试以证明其符合要求。鼓励减少不必要的文档，即减少不能为客户带来价值的文档。一个成功的敏捷项目，应该在减少文档以提高效率和提供足够文档以支持业务、测试、开发、和维护活动两者之间取得一个平衡。在发布计划期间，团队必须决定，哪些工作产品是必需的和什么详细程度的工作产品文档是必需的。

在敏捷项目中，典型业务导向的工作产品是用户故事和它的验收准则。用户故事是需求规格说明的敏捷表达形式，其应阐明系统针对单一和可理解的特性或功能应该如何运作和反应。用户故事应该定义一个足够小，可以在单次迭代内完成的特性。由相互关联特性组成更大的组合，或由多个子特性组成一个单一复杂特性，被称为“史诗（Epic）”。史诗可能包括不同开发团队的用户故事。史诗示例：一个用户故事可能描述在 API 层级（中间件）的需求，而另一个用户故事则描述在 UI 层级（应用）的需求。这些集合可能在一系列的冲刺中被开发出来，每个史诗和它的用户故事应该有相关的验收准则。

在敏捷项目中，开发人员的典型工作产品包括代码。敏捷开发人员还经常创建自动化的单元测试。这些测试可能是在代码编写完后就创建。然而，在某些情况下，开发人员会在编写某部分实际代码之前增量的创建测试，这样做可以使得这部分代码编写完成后即可马上验证它是否能如预期地运行。虽然这种方法被称为测试优先或测试驱动开发，而在现实中，与其说是测试，但它更多的是可执行的详细设计的一种形式 [beck02]

敏捷项目中测试人员的典型工作产品包括自动测试脚本和一些文档如：测试计划、质量风险清单、手工测试、缺陷报告和测试结果日志。像当下流行的那样，这些文件应尽可能编写得更轻量，其实，在传统开发生命周期中也是需要这样的轻量化文档。测试人员也会从缺陷报告和测试结果日志中产出测试度量，同样，也要注重轻量级方法的使用。

在一些敏捷项目实施中，特别是在规范的、安全关键的、分布式的或高度复杂的项目和产品中，更进一步的规范化其工作产品是需要的。例如，一些团队把用户故事和验收准则转换成更正式的需求规格说明。同时可能需要准备纵向和横向跟踪报告，以满足审计、法规和其他需求。

2.1.3 测试级别

测试级别是逻辑性相关的一系列测试活动，通常是由测试项的成熟度和完整性方面决定的。

在顺序生命周期模型中，测试级别通常定义一个级别的出口准则是下一个级别的入口准则的一部分。在一些迭代模型中，这个规则并不适用。测试级别会重叠，需求规格说明、设计规格说明和开发活动都可能会与测试级别重叠。

在一些敏捷项目的生命周期中，重叠的发生是因为需求、设计和代码的变更可能在一个迭代的任何点出现。而在 Scrum 中，从理论上讲，在迭代计划后，是不允许更改用户故事的，但在实践中，这样的变更有时会发生。在一个迭代中，所有给定的用户故事通常会执行如下一系列测试活动：

- 单元测试，通常由开发人员完成；
- 特性（Feature）验收测试，有时候分成以下两个活动：
  - 特性验证测试，这往往是自动化的，可以由开发人员或测试人员完成，并涉及对用户故事的验收准则测试；
  - 特性确认测试，这通常是手动测试，并可通过开发人员、测试人员和业务干系人一起协同工作来确定特性是否适用，以及取得对进展的清晰认知，并从业务干系人（一般是用户或用户代表——译者）获得真实的反馈；

此外，在整个迭代过程中，经常会发生回归测试的并行处理。这个涉及到从当前迭代和上个迭代中重新运行自动单元测试和特性验证测试，这往往是通过持续集成框架来实现的。

在一些敏捷项目中，还可能会有系统测试级别，它是从第一个用户故事准备好测试后就开始了。这可能包括执行功能测试，也包括了如性能、可靠性、易用性以及其他相关测试类型的非功能性测试。

敏捷团队可以采用各种形式的验收测试（使用初级大纲[ISTQB_FL_SYL]里解释的术语），可以是内部 alpha 测试和外部 beta 测试，可能发生在每次迭代结束时、每次迭代完成后（集成后）、或在一系列的迭代之后。用户验收测试、运行验收测试、法规符合性验收测试和合同符合性验收测试也可能会出现，也同样会是在每次迭代结束时、每次迭代完成后，或一系列迭代后进行。

2.1.4 测试和配置管理

敏捷项目往往采用大量自动化工具来开发、测试和管理软件开发。开发人员使用工具来进行静态分析，单元测试和代码覆盖。开发人员使用自动化构建和测试框架持续地把代码和单元测试入库到配置管理系统中。这些框架允许新软件与系统的持续集成，当新软件入库时，静态分析和单元测试会重复的运行[Kubaczkowski]。

这些自动化测试还可能包括在集成和系统级别上的功能测试。这些自动化的功能测试可以通过功能测试框架，开源的用户界面功能测试工具，或商业工具来创建，它还可以与自动化测试运行集成而作为持续集成框架的一部分。在某些情况下，由于功能自动化测试的持续时间长，该部分就会从单元测试分离出来，并降低运行的频率。例如，单元测试可能在每次代码提交（check in）时都运行一次，而需要较长时间的功能测试只是每隔几天才运行一次。

自动化测试的目标之一是要确认该构建是有效的、可安装部署的。如果任何自动化测试失败，团队应及时地修复潜在缺陷，保证下一次代码提交（check in）可以完成。这就需要在实时（real-time）测试报告上投入更多以提供良好的测试结果可视性。这种方法有助于减少在许多传统项目中发生的“构建-安装-失败-重新构建-重新安装”的费用和低效循环，因为这种破坏构建或造成软件安装失败的变更会被快速地检测到。

自动化测试和构建工具可以帮助管理在敏捷项目中频繁变更导致的回归风险。然而，过度的只依赖于自动化单元测试来管理这些风险可能也是一个问题，因为单元测试往往对缺陷检测只有有限的效果[Jones11]。所以，集成和系统级别的自动化测试也是需要的。

2.1.5 组织对独立测试的选择

如在基础级大纲[ISTQB_FL_SYL]讨论的一样，独立的测试人员往往在发现缺陷方面更为有效。在一些敏捷团队中，开发人员以自动化测试的形式建立了许多的测试。一个或多个测试人员可能安排在团队内部中执行着许多测试任务。然而，使这些测试人员成为研发团队的一部分，可能会导致失去测试独立性和客观性的风险。

有些敏捷团队则保持其完全独立的、分离的测试团队，只在每次冲刺的最后几天指派需要的测试员。这可以保持独立性，并且测试人员可以给软件提供客观的、公正的评价。然而，时间压力，缺乏对产品新功能的了解，和与业务干系人和开发人员的关系的问题往往会导致这种方法的执行遇到各种问题。

第三个选择是维持一个独立和分离的测试团队，但测试人员在项目开始时就被长期的分配到敏捷团队中，这样既允许他们保持独立性，同时也使他们获得了对产品很好的理解和良好的团队关系。此外，独立的测试团队可以由敏捷团队以外的专门测试人员参与长期和/或迭代无关的活动，如开发自动化测试工具、进行非功能性测试、创建并支持测试环境和数据以及执行不适用于单个冲刺的测试级别（例如，系统集成测试）。

2.2 敏捷项目中的测试状态

在敏捷项目中，变更快速的发生。这些变更意味着测试状态、测试进展和产品质量不断的演变，测试人员必须设法将这些信息传递给整个团队，以使他们能够在正确的轨道上做出决定，确保每一个迭代能成功的完成。此外，变更可能影响之前迭代已存在的特性。因此，手工和自动化测试必须要及时更新以有效地处理回归风险。

2.2.1 沟通测试状态、进展和产品质量

敏捷团队的进展是通过在每个迭代结束时产出可工作的软件。要确定团队何时才产出可工作的软件，他们需要在迭代和发布时监控所有的工作产品。在敏捷团队的测试人员利用各种方法来记录测试进度和状态，包括监控测试自动化结果、监控敏捷任务板里测试任务和故事的进展、以及监控燃尽图以了解团队的进展。这些信息可以通过类似于维基仪表盘和面板的邮件等媒介，或简单的口头例会的形式来传达给其他项目成员。敏捷团队也可以使用基于测试结果和任务进度而自动生成状态报告的工具，从而更新维基风格的面板和电子邮件。这种沟通方式也可以从测试过程中收集度量指标，这些度量指标同时可以在过程改进中使用。通过自动的方式来沟通测试状态可以把测试人员的时间释放出来，以使其拥有更多的时间去关注测试设计和执行更多测试用例。

团队可以在整个版本发布和每次迭代中使用燃尽图来跟踪进度。一个燃尽图[Crispin08]可以反映当前版本或迭代中剩余工作量与剩余时间的对比。[Crispin08]

为了给整个团队提供目前状态（包括测试状态）实时和细致的可视化展示，团队可以使用敏捷任务板。故事卡片、开发任务、测试任务和其他在迭代计划期间建立的（参见 1.2.5 节）的任务都可以在任务板上展示，通常是使用彩色协调卡来确定任务类型。在迭代中，是通过在任务板中移动任务到“计划中”、“进行中”、“已验证”、“完成”这四列当中去从而来管理项目的进展。敏捷团队可以使用工具来维护他们的故事卡片和敏捷任务板，它能自动化进行面板和状态的更新。

例如，在任务板上的测试任务与用户故事的验收准则相关联。当与测试任务相关的测试自动化脚本，手工测试和探索性测试通过后，这个任务就可以移动到任务板的“完成”列中。通常在每日站会中，整个团队会评审任务板的状况，从而确保任务正以可接受的速度在任务版中移动。如果任务（包括测试任务）没有向前移动，或者移动进度过于缓慢，团队则要评审并解决任何有可能阻碍这些任务前进的问题。

每日站会人员包括敏捷团队的所有成员，也包括测试人员。在会议上，所有人传达他们目前的进展状况。每个成员的议程是[敏捷联盟指南]：

- 自上次会议后，你完成了什么？
- 在下次会议前，你计划完成什么？
- 什么阻碍了你？

任何可能会阻碍测试进度的问题都会在每日例会里沟通，所以整个团队都意识到问题所在，并可以相应的解决它。

为了提高产品的整体质量，许多敏捷团队进行客户满意度调查，以获得产品是否满足客户期望的反馈。为了提高产品的质量，团队可以使用类似于在传统开发中的度量指标，如测试通过/失败率、缺陷发现率、确认和回归测试结果、缺陷密度、发现和修复的缺陷、需求覆盖率、风险覆盖率、代码覆盖率和代码改动率。正如在任何生命周期里一样，捕获到和上报的指标应该是决策相关和辅助决策的。度量指标不应该被使用来奖励、惩罚、或者孤立任何的团队成员。

2.2.2 使用演进的手工和自动化测试用例来管理回归风险

在敏捷项目中，随着每个迭代的完成，产品也不断的壮大。因此，测试的范围也随之扩大。除了测试当前迭代中代码变更以外，测试人员还需要验证之前迭代已经开发和测试但还没有回归的功能。由于大量的代码改动（从一个版本到另一个版本的代码行增加、修改或删除），在敏捷开发中引入回归的风险是很高的。由于应对变更是敏捷的一个关键原则，为满足业务的需求，变更也可以在之前已经交付的功能中发生。为了保持速度，同时不产生大量的技术债，团队应尽早在所有测试级别中投入测试自动化是至关重要的。同样重要的是，所有的测试资产，如测试自动化、手工测试用例、测试数据和其他的测试件，都在每个迭代中保持最新状态。在此强烈推荐使用配置管理工具来维护所有的测试资产，以加强版本控制，确保团队成员可以轻松访问，并支持由于功能变更所导致的更改，同时仍保留着测试资产的历史信息。

因为要进行完全回归测试几乎是不可能的，尤其是在时间很紧的敏捷项目中，所以测试人员需要在每次迭代中分配时间来评审之前和当前迭代中的手工和自动化测试用例，从而选择适合作为回归测试套件的测试用例，并淘汰掉不相关的测试用例。在早期迭代中用于验证特定特性的测试用例可能在后面的迭代中价值不大，因为特性的变更或新特性的加入会改变这些早期特性的运行方式。

在评审测试用例中，测试人员应该考虑这些用例是否适用自动化。团队应从之前和当前的迭代中尽可能多的实现测试自动化。使用自动化回归测试可以降低回归风险，且与手工回归测试相比，需要的工作量可能较少。减少回归测试工作量，释放测试人员，使他们可以更彻底地测试当前迭代的新特性和新功能。

然而，至关重要的是，测试人员应拥有能力可以从之前的迭代和/或版本发布中快速的识别和更新受到当前迭代变更影响的测试用例。定义团队如何设计、编写、和储存测试用例应该是在版本发布计划阶段进行。测试设计和测试实施的好实践应尽早采用并始终如一的执行。因为不足的测试时间和每次迭代中的不断的变更会扩大不良测试设计和实施带来的负面影响。

在所有的测试级别中使用测试自动化，可以使得敏捷团队针对产品质量快速的提供反馈。编写很好的自动化测试为系统功能性提供了活生生的文档[Crispin08]。通过把与其产品构建版本一致的自动化测试和相应的测试结果入库到配置管理系统中，敏捷团队可以在任何时间点查看任何构建的已测试功能和其测试结果。

自动化单元测试会在源代码检入到配置管理系统的主干（mainline）之前运行，以确保代码的更改不会破坏软件构建。构建被破坏会拖慢整个团队的进度，为了减少构建破坏的出现，代码应当在所有自动化单元测试都通过了才能入库。自动化单元测试结果提供对代码和构建质量的即时反馈，但不是对产品质量的反馈。

自动化的验收测试作为持续集成整个系统构建的一部分而定期的运行。这些测试是针对一个完整的系统构建，至少每天运行一次，但一般不会在每次代码入库后都运行，因为它们比自动化单元测试耗时更长，并可能减慢代码入库的速度。自动化验收测试可以对上次构建后产品在回归方面的质量风险提供反馈，但它们不反映产品整体质量状况。

针对整个系统的自动化测试可以持续的执行。那些用于覆盖关键系统功能和集成点的自动化测试，应该在新构建部署到测试环境后马上创建。这些测试通常被称为构建验证测试。构建验证测试的结果将提供对部署后软件的即时反馈，因此团队不再需要浪费时间去测试一个不稳定的构建。

作为回归测试集的一部分，自动化回归测试通常是作为日常主构建的一部分，在持续集成环境中运行，当新构建部署到测试环境时，这些自动化脚本需要再次被运行。一旦有一个自动化回归测试失败，团队会停止当前工作以调查失败原因。这些测试失败可能是由当前迭代中合理的功能变更引起的，在这种情况下，测试和/或用户故事可能需要更新以反映新的验收准则。此外，如果建立了覆盖这些变更的新测试，则需要撤掉原来的旧测试。但是，如果这些测试是因为有缺陷而没有通过，良好的实践是，团队应优先修复这些缺陷，然后再继续开发新特性。

除了测试自动化之外，下面的测试任务也是可以自动化的：

- 测试数据生成；
- 加载测试数据到系统中；
- 部署构建到测试环境中；
- 恢复测试环境（例如，数据库或网站的数据文件）到基线；
- 数据输出的对比。

这些任务的自动化降低了开销，并允许团队花更多的时间去开发和测试新的功能。

2.3 敏捷团队中测试人员的角色与技能

在敏捷团队中，测试人员必须与所有其他团队成员和业务干系人紧密合作。这意味着敏捷团队的测试人员必须掌握一些技能和参与敏捷团队的活动。

2.3.1 敏捷测试人员的技能

敏捷测试人员应该具备基础级考试大纲 [ISTQB_FL_SYL] 中提到的所有技能。除了这些技能，敏捷团队的测试人员还要胜任测试自动化、测试驱动开发、验收测试驱动开发、白盒、黑盒和基于经验的测试。

由于敏捷方法在很大程度上依赖团队成员之间和团队外干系人之间的协作、沟通和互动，所以敏捷团队的测试人员应该具备良好的人际沟通技巧。敏捷团队中的测试人员应该：

- 与团队成员和干系人保持积极的、以解决方案导向的态度；
- 针对产品，具备严肃的、质量导向的和怀疑精神的思维；
- 积极地从干系人获取信息（而不是完全依靠书面规格说明）；
- 准确地评估和报告测试结果、测试进度和产品质量；
- 与客户代表和干系人高效的工作，并定义出可测试的用户故事，特别是验收准则；
- 合作精神，可以与程序员和其他团队成员结对工作；
- 迅速地应对变化，包括更改、增加，或改善测试用例；
- 计划和组织好自己的工作；

持续的能力提升，包括人际交往能力的提升，是测试人员最核心的技能，对敏捷团队的测试人员来说也不例外。

2.3.2 敏捷团队中测试人员的角色

敏捷团队的测试人员，其职责不仅包括提供对测试状态、过程以及产品质量的反馈信息，还包括对整个过程的质量反馈。执行包括但不限于下面罗列的各种活动：

- 理解、实施和更新测试策略；
- 通过所有可能的覆盖维度度量和报告测试覆盖情况；
- 确保对测试工具的合理使用；
- 配置、使用和管理测试环境与测试数据；
- 报告缺陷，并与团队合作以解决这些缺陷；
- 给其它团队成员提供与测试有关的培训；
- 确保在发布和迭代计划中，加入了测试任务并安排了合理的时间进度；
- 积极地与开发人员和业务干系人协作，对需求进行细化，尤其是对需求的可测试性、一致性和完整性进行细化；
- 积极地参加团队的回顾会议，建议并实施改进措施。

在敏捷团队中，全员都需要对产品质量负责，并承担与测试有关的角色和任务。

同时，在敏捷团队中，也会遇到一系列与测试有关的组织风险：

- 测试人员与开发人员过于密切的工作导致他们失去了测试人员思维；
- 测试人员对团队内低效率、无效果以及低质量的实践保持容忍或沉默；
- 测试人员在有时间限制的迭代中不能跟上变更的步骤。

为了缓和这些风险，组织内部需要考虑保护测试独立性（在 2.1.5 节可以看到更多相关讨论）。

3. 敏捷测试方法、技术和工具 – 480 分钟

关键词

验收准则（acceptance criteria），ET 探索性测试（exploratory testing），性能测试（performance testing），产品风险（product risk），质量风险（quality risk），回归测试（regression testing），测试方法（test approach），测试章程（test charter），测试估算（test estimation），测试执行自动化（test execution automation），测试策略（test strategy），测试驱动开发 TDD（test-driven development），单元测试框架（unit test framework）

敏捷测试方法、技术和工具学习目标

3.1 敏捷测试方法

FA-3.1.1	（K1）	复述测试驱动开发，验收测试驱动开发，行为驱动开发的概念。
FA-3.1.2	（K1）	复述测试金字塔的概念。
FA-3.1.3	（K2）	总结测试四象限法以及各个象限与各种测试级别和测试类型的关系。
FA-3.1.4	（K3）	对于给定的敏捷项目，测试人员在 SCRUM 团队中的角色实践。

3.2 评价质量风险和估算测试工作量

FA-3.2.1	（K3）	在敏捷项目中评价质量风险。
FA-3.2.2	（K3）	基于迭代内容和质量风险，估算测试工作量。

3.3 敏捷项目中的技术

FA-3.3.1	（K3）	理解为支持测试活动相关的信息。
FA-3.3.2	（K2）	能给业务干系人解释如何定义可测试的验收准则。
FA-3.3.3	（K3）	给定一个用户故事，编写验收测试驱动开发（ATDD）的测试用例。
FA-3.3.4	（K3）	基于给定的用户故事，使用黑盒测试设计技术，编写功能和非功能性测试用例。
FA-3.3.5	（K3）	在敏捷项目中使用探索性测试。

3.4 敏捷项目中的工具

FA-3.4.1	（K1）	复述在敏捷项目中，根据目的和活动，测试人员可用的工具。

3.1 敏捷测试方法

事实上，许多测试实践都可以应用于各种开发项目（无论它们是否是敏捷的）中，以开发出高质量的产品。这些实践包括：提前设计测试以体现正确的用户行为，专注于早期的缺陷预防、发现和消除，确保在合适的时间、合适的测试级别上执行合适的测试类型。敏捷实践者们早期努力将这些最佳测试实践引入到项目中。而敏捷项目中的测试人员在指导人们在整个软件研发生命周期中应用这些最佳实践上扮演着重要的角色。

3.1.1 测试驱动开发、验收测试驱动开发和行为驱动开发

测试驱动开发、验收测试驱动开发和行为驱动开发是敏捷团队中常用的三种互补的方法，可以在不同的测试级别中使用。其每种方法都认为是软件测试的基本原则，测试和 QA 工作应尽早介入，都提倡测试应该在代码编写之前进行。

测试驱动开发

测试驱动开发（TDD）使用自动化测试用例来指导和验证开发代码。TDD 过程包含下列活动：

- 新增一个测试（成生测试用例并自动化），用于描述在代码内的某一小片段所期望的功能（程序设计的思路）；
- 此时因为相应的业务代码还没有编写，测试无法通过；
- 编写业务代码并运行测试直到测试通过；
- 在测试通过后，如果代码又有变更，例如对业务代码进行重构，则需再次运行测试以确保变更后的代码仍然可以通过测试；
- 对代码内的下一小片段继续执行上述过程，需要运行之前的测试（回归测试）以及新加入的测试。

尽管前文所述的测试，主要是指单元级别的测试且以代码为中心，但也可以应用在集成测试和系统测试等级别上。测试驱动开发是在极限编程实践中开始流行[Beck02]，但是测试驱动开发也应用在其他敏捷实践中，有时在传统顺序开发生命周期中也有运用。测试驱动开发帮助开发人员关注在明确定义的期望结果。而测试驱动开发中的测试需要实现自动化并应用到持续集成中。

验收测试驱动开发

验收测试驱动开发（ATDD） [Adzic09]，是一种在用户故事（用户故事的定义见 1.2.2 节）的创建阶段就设定验收准则和测试的技术。ATDD 强调协作，各个干系人（开发、测试、业务代表等）不仅要理解软件组件应该有何种行为，而且要理解如何做才能确保可以通过验收。这将在 3.3.2 节中详述 ATDD 的过程。

ATDD 为回归测试创建了可重用的测试。ATDD 需要使用一系列工具以支持测试创建和执行，而持续集成工具是其中最常用的。这些工具可以在数据和应用服务层之间搭建桥梁，从而允许在系统和验收级别上执行测试。ATDD 允许对缺陷进行快速修正及对修正的特性进行验证。可以帮助确认特性实现是否符合验收准则。

BDD 行为驱动开发

行为驱动开发（BDD）[Chelimsky10]使开发人员聚焦于测试软件行为是否符合预期。由于测试是以软件行为的形式展示出来的，测试对于团队成员和其他干系人来说更容易。

典型的 BDD 框架将验收准则用 given/when/then 的格式表示：

*Given（给定）某些*初始上下文，
*When（当）*一个事件发生，
*Then（则）*确保某些输出。

基于这样的需求，BDD 框架可以帮助开发人员生成测试代码。BDD 框架帮助开发人员与其他干系人（包括测试人员）有效协作，从而能够针对业务需求定义精确的单元测试。

3.1.2 测试金字塔

一个软件系统可以在不同的测试级别上进行测试。典型的测试级别结构是金字塔型的，从金字塔的最底层直到塔尖包括了单元、集成、系统和验收测试（此部分内容见 ISTQB 基础级大纲，2.2 节）。测试金字塔强调在底层——即低级别中进行大量的测试，随着开发过程的进行，测试向高层进行，测试的数量逐渐减少。通常单元和集成级别的测试是借助基于 API 测试工具实现自动化。而系统和验收测试级别的自动化测试则是由基于 GUI 测试工具创建的。测试金字塔是基于“尽早进行测试和质量保证”（即尽可能在生命周期的早期解决缺陷）这一理念创建的。

3.1.3 测试象限，测试级别和测试类型

测试象限，是由 Brian Marick 定义的，描述在敏捷开发中如何将合适的测试类型和测试级别整合起来。[Crispin08]测试象限法（以及该方法的一系列变体）有助于软件开发生命周期中确实包含了重要的测试类型和测试级别。这个方法可以用于向干系人（开发人员、测试人员和商业代表）讲解和区分不同类型的测试。

在测试象限中，测试被分为面向业务（用户）的或者面向技术（开发人员）的。一些测试是用于支持敏捷团队的工作，用于判断软件的行为（是否符合预期）。而另外一些测试则是（从用户角度）对产品进行验证。测试可以是完全手工的，或者完全自动化的，或者半手工半自动化的，或者有工具支持的手工测试。四个不同的测试象限表述如下：

- 第一象限 Q1，即单元级别，是面向技术的，目的是支持开发人员。该象限包含单元测试。这些测试应该能够完全自动化并能整合在持续集成中。
- 第二象限 Q2，即系统级别，是面向业务的，主要目的是对产品的行为进行确认。该象限内包含功能测试，例如，对用户故事的测试、用户体验原型和仿真。这些对验收准则的检验可以是手工的，也可以是自动化的。这种测试常常在用户故事的构建阶段就被创建，因而有助于（发现用户故事编写中的问题）提高用户故事本身的质量。他们对于创建自动化回归测试套件也是有帮助的。
- 第三象限 Q3，是系统或用户验收级别，是面向业务的，该象限内的测试使用真实的用户数据和场景，包含对产品进行确认。在该象限中经常使用到探索性测试、场景测试、业务流程测试、易用性测试、用户验收测试、α测试和β测试。这些测试通常是手工的、面向用户的测试。
- 第四象限 Q4，是系统或者运维验收级别，面向技术的，同时包含对产品的确认。该象限经常使用性能、负载、压力测试、可伸缩性测试、安全性测试、可维护性测试、内存管理、合规性测试、数据迁移测试、基础设施测试以及可恢复性测试。这些测试也经常是自动化进行的。

在任一迭代中，可能都需要某一象限或者所有象限的测试。测试象限法更适用于动态测试而非静态测试。

3.1.4 测试人员的角色

整篇大纲覆盖了敏捷测试中常用的方法和技术，以及测试人员在众多敏捷测试生命周期中的作用。这个部分将着重讨论测试人员在 Scrum 项目中的角色[Aalst13] 。

团队合作

团队合作是敏捷开发的基本原则。敏捷一词强调开发人员、测试人员、业务代表协作的全团队工作方法。下面列出了 Scrum 团队在组织上和行为上的最佳实践要点：

- 跨职能团队：每个团队成员都会带来不同的技能。整个团队一起协作开发测试策略、计划、测试规格说明、执行测试、进行测试评估以及整理测试结果报告。
- 自组织：整个团队可能仅由开发人员组成，但是如 2.1.5 节所说，比较理想的情况是，团队有测试人员参与。
- 相同工作地点：测试人员要和开发人员、产品负责人（product owner）坐在一起。
- 协作：测试人员和自己所在团队成员、其他团队成员、干系人、产品负责人以及 Scrum Master 共同协作。
- 授权：关于设计和测试的技术层面决定由整个团队（开发人员、测试人员、Scrum Master）做出，必要时可与产品负责人和其他团队合作。
- 承诺：测试人员致力于发现问题，同时根据客户和用户的期望和需求评估产品性能和特点。
- 透明：可以借助敏捷任务板（或任务墙）展示开发和测试进程（参考 2.2.1 节）。
- 可信：测试人员必须确保测试从策略到实现、执行的可信性，否则干系人不会相信测试结果。通常需要向干系人提供测试过程的有关信息。
- 开放：对反馈的开放态度是任何项目成功的重要因素之一，特别是对于敏捷项目。回顾可以让团队从成功和失败中吸取经验教训。
- 适应：测试必须像敏捷项目中的其他活动一样，及时响应变化。

通过以上这些最佳实践，使 Scrum 项目中进行成功测试的可能性得以最大化。

零冲刺（Sprint Zero）

零冲刺是项目开展各种准备工作所进行的第一次迭代（参见 1.2.5 章节）。在这次迭代中测试人员和整个团队合作完成如下任务：

- 确定项目的范围（例如产品待办列表）；
- 创建最初的系统构架以及高层次的原型；
- 规划、获取并安装所需的工具（例如测试管理、缺陷管理、测试自动化、持续集成所需的工具）；
- 制定最初的针对所有级别测试的测试策略，包括测试范围、技术风险、测试类型（参考 3.1.3 节）、覆盖目标方面；
- 进行最初的质量风险分析（参考 3.2.1 节）；
- 制定针对测试过程、项目中测试进展、产品质量的度量；
- 明确“完成”的定义；
- 创建任务板（参考 2.2.1 节）；
- 确定在将系统提供给用户之前何时继续或停止测试。

零冲刺指明了测试所需达到的目标以及如何通过迭代实现这个目标。

集成

敏捷项目的目标在于持续向客户交付价值（最好在每个冲刺中）。为此，集成策略应将设计和测试一同考虑进来。对于交付的功能和特性，为了达到一个可持续测试的策略，识别出潜在的功能和特性之间的依赖是重要的。

测试计划

由于测试是完全集成在敏捷团队中的，测试计划应该开始于发布计划会议，并在每个冲刺中持续的更新。发布和每个冲刺的测试计划应处理的相关问题，在 1.2.5 章节进行了讨论。

冲刺计划将一系列的任务加入到任务板上，在这个任务板中每个任务应该是 1-2 天的工作。此外，为了保持一个稳定的测试流程，任何测试问题都应该被跟踪。

敏捷测试实践

在 Scrum 团队中，有许多对测试人员来说有效的最佳实践：
结对：两个组员（例如：一个测试人员和一个开发人员、两个测试人员、或者一个测试人员一个产品负责人）在同一工作站坐在一起工作，工作内容可能是测试，也可能是其他迭代中的工作。
增量测试设计：根据用户故事和其他的测试基础逐渐建立测试用例和测试章程，从简单的测试开始，逐步推动进行更复杂的测试。

思维导图：非常有效的测试工具[Crispin08]。例如，测试人员可以通过思维导图来识别执行哪个测试会话、来展示测试策略以及来描述测试数据。

这篇大纲中讨论的这些实践要点是对 ISTQB 基础级大纲[ISTQB_FL_SYL]第四章中所述测试实践的一个补充。

3.2 评估质量风险和估算测试工作量

在所有项目中（不管是敏捷还是传统），典型的测试目标都是减少产品质量问题的风险，从而在发布前将质量问题的风险缓解到可接受的水平。在敏捷项目中，测试人员可以利用与传统项目中同样类型的技术来识别质量风险（或者产品风险）、评估相关风险等级、预测为有效降低风险所需要的工作量，以及通过测试设计、实施和执行活动缓解上述风险。但是，敏捷项目的短迭代和变更频度使得上述技术需做适当调整。

3.2.1 评估敏捷项目中的质量风险

测试所面临的众多挑战之一是对测试条件的选择、分配以及优先级排序。这包括：为了在测试中覆盖每个条件，要分配合适的工作量；为优化测试工作的有效性和效率，对测试工作进行排序。在敏捷团队中，测试人员可以运用风险识别、分析和风险缓解策略来确定需执行测试用例的数量，尽管这过程中有许多相互作用的制约和变数需要我们做出折衷。

风险是一个负面的、意料之外的结果/事件出现的可能性。风险等级可以通过评估其出现的可能性和严重程度来确立。当潜在问题的主要影响集中于产品质量时，潜在的问题则为质量风险或者产品风险。当潜在问题的主要影响为项目的成功与否，潜在问题则为项目风险，或计划风险[Black07] [vanVeenendaal12]。

在敏捷项目中，需要在下列两个时间点做质量风险分析：

- 在制定发布计划时：业务代表知道就发布的特性提供风险的高级别概述，整个团队，包括测试人员可以帮助识别和评估风险。
- 在制定迭代计划时：整个团队一起识别和评估质量风险。

下面是典型的系统质量风险例子：

- 输出报告中的计算错误（与准确性相关的功能性风险）
- 对用户输入的反应太慢（与效率和响应时间相关的非功能性风险）
- 难以理解的屏幕和字段（与易用性和易理解性相关的非功能性风险）

如前所述，迭代始于迭代计划会议，重点在于对任务板上的任务进行估算。这些任务可以基于相关的质量风险进行优先级排序。高风险任务应尽早启动，并分配更多的测试工作量。低风险任务可稍迟启动，相应的测试工作量也可以减少。

下面的例子展示了在敏捷项目里的迭代计划中，执行质量风险分析的基本步骤：

1. 敏捷团队成员（包括测试人员）集合在一起；
2. 列出当前迭代的所有待办列表（例如，在任务板上）；
3. 识别每个条目的相关质量风险，考虑所有相关的质量特性；
4. 评估每个识别出的风险，包括两个行动：对风险进行分类，然后基于风险的严重程度和风险出现的概率来决定风险等级；
5. 根据风险等级决定测试的规模的比例；
6. 基于风险、风险等级和相关质量特性，选择合适的测试技术来缓解风险。

测试人员随后设计、实施和执行测试来缓解风险。这包括所有特性、行为、质量特性以及影响客户、用户和干系人满意度的属性。

在整个项目过程中，团队应该了解那些会改变或影响风险或已知质量风险等级的额外信息。要定期地分析质量风险并进行调整，进而进行测试的调整。调整包括识别新的风险，重新评估现有风险等级，和评估风险缓解措施的有效性。

质量风险也可以在测试执行之前得到缓解。例如，如果在风险识别过程中就发现了用户故事的问题，作为一种缓解策略，项目团队可以彻底地重新评审用户故事。

3.2.2 基于内容和风险估算测试工作量

在发布计划中，敏捷团队估算完成发布所需的工作量。这个估算也包括测试工作量。敏捷项目中所用的常见估算技术是计划扑克，它是一种基于共识的技术。PO（产品负责人）或客户向估算者读出用户故事，上面带有类似斐波那契序列（fibonacci sequence）的数值（即 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...），或其他渐进式的选择（例如 T 恤尺码，从 XS 到 XXL）。这些值代表了故事点数，人天数，或团队估算的其他单位。这里，推荐斐波那契序列，因为数字序列反映了不确定性随着故事的变大而增长。估算值高通常意味着故事没能很好地理解或应该分解为多个更小的故事。

估算人员讨论特性，并根据需要向产品负责人提问。包括开发和测试工作量、故事复杂度，测试范围也对估算值有影响。因此在开始计划扑克会议之前，除了产品负责人给出的优先级，将待办列表的风险等级也考虑进去才是明智的。当功能得到充分讨论后，每个估算人员私下选择一张卡片代表他自己的估算。然后同时亮出所有的卡片，如果所有估算人员选择了相同的数字，那么该数字就成为估算值。否则，估算人员随后就讨论估算之间的差异，重复这个过程直到取得一致，要么是达成了共识，要么通过采取某种规则（比如使用中位数，或使用最大数）来限制出牌的回合数。 这些讨论能确保给出完成产品待办列表所需的可靠估算，满足产品负责人的要求，同时有助于改善要实现功能的集体知识 [Cohn04]。

3.3 敏捷项目中的技术

许多运用到传统项目中的测试技术和测试级别也可以应用到敏捷项目中。然而，对于敏捷项目来说，在测试技术、术语和文档方面还有一些特殊的注意事项和变化要考虑。

3.3.1 验收准则、充分覆盖和测试的其他信息

敏捷项目在开始阶段会用待办列表中的用户故事来勾勒初始需求。初始需求是短小的，通常遵循某种预定义格式（见 1.2.2 节）。非功能性需求，比如易用性和性能也是重要的，可以指定为单独的用户故事或连接到其他的功能性用户故事。非功能性需求可以遵循某种预定义格式或标准，比如 [ISO25000]，或行业的具体标准。

用户故事是一种重要的测试依据，其他可能的测试依据包括：

- 来自之前项目的经验；
- 现有系统的功能、特性和质量特性；
- 代码、架构和设计；
- 用户配置（user profiles）（上下文、系统配置和用户行为）；
- 来自现有和之前项目的缺陷信息；
- 按照某种方式的缺陷分类；
- 适用的标准（如，航空软件的 [DO-178B]）；
- 质量风险（见 3.2.1 节）。

在每个迭代中，开发人员编写代码来实现用户故事中描述的功能和特性，满足相应的质量特征，使代码通过验收测试来得到验证和确认。要做到可测试，验收准则应该解决下列的有关问题 [Wiegers13]：

- 功能行为：在某种配置下，用户活动作为输入操作的外部可观察行为。
- 质量特性：系统如何执行指定行为。这些特性也称为质量属性或非功能性需求。常见的质量特征包括性能、可靠性、易用性等。
- 场景（用例）：为了完成特定目标或业务任务，外部行动者（通常是用户）与系统之间的一系列行动。
- 业务规则：由外部规程或约束定义的，在某些条件下只能在系统中执行的行为。（例如，保险公司用来处理保险索赔的流程）。
- 外部接口：描述待开发系统与外部世界之间的联系。外部接口可以分为不同的类型（用户接口、与其他系统的接口等）。
- 约束：任何设计和实现的约束都会限制开发人员的选择。嵌入式软件设备必须遵循物理约束，比如尺寸、重量和接口连接。
- 数据定义：对于一个复杂的业务数据结构，客户会针对数据项来描述格式、数据类型、允许的值和默认值（例如，美国邮政地址的 ZIP 编码）。

除了用户故事及其关联的验收准则，其他与测试人员相关的信息包括：

- 系统应该如何工作和被使用；
- 用于测试该系统的可访问的系统接口；
- 当前工具的支持是否足够；
- 测试人员是否具备足够的知识和技能来执行必要的测试。

在迭代中，测试人员经常发现自己需要额外信息（比如，代码覆盖），就应该与敏捷团队的其他成员进行协作，来获取该信息。相关信息有助于决定某个特殊行为是否可以标记为完成。“完成的定义”的概念在敏捷项目中非常关键，正如后续章节中所讨论的，它以各种不同方式得到运用。

测试级别

每个测试级别都具有自己的完成定义。下列清单针对不同测试级别给出了例子。

- 单元测试
  - 尽可能达到 100%判定覆盖率，仔细地评审不可达路径；
  - 对所有代码执行静态分析；
  - 没有未解决的严重缺陷（基于优先级和严重度来分级）；
  - 在设计和代码中没有已知的不可接受的技术债 [Jones11]；
  - 所有代码、单元测试和单元测试结果都得到评审；
  - 所有单元测试都是自动化的；
  - 重要特性满足约定的限制（比如，性能）。
- 集成测试
  - 所有功能需求都经过测试，包括正向和逆向测试，测试的数量基于规模、复杂度和风险；
  - 所有单元之间的接口都经过测试；
  - 根据约定的测试程度，所有质量风险都已覆盖；
  - 没有未解决的严重缺陷（根据风险和重要性来排序）；
  - 所有发现的缺陷都已报告；
  - 在可能的情况下，所有回归测试都是自动化的，所有自动化测试都存放在一个公共库中。
- 系统测试
  - 用户故事、特性和功能的端到端测试；
  - 覆盖了所有用户角色；
  - 覆盖了系统最重要的质量特性（例如，性能、健壮性、可靠性）；
  - 在类似生产环境中完成测试，在一定程度上包括了所有软硬件和所支持的配置；
  - 根据测试约定的测试程度覆盖了所有质量风险；
  - 在可能的情况下，所有回归测试都是自动化的，所有自动化测试都存放在一个公共库中；
  - 所有发现的缺陷都已报告并被修复；
  - 没有未解决的严重缺陷（根据风险和重要性来排序）。

用户故事

用户故事的完成定义可以由以下标准来决定：

- 迭代所选择的用户故事全部完成，能被测试团队理解，并且有详细的、可测试的验收准则；
- 用户故事中的内容都已经被详细设计并通过了评审，与之匹配的验收测试也已经完成；
- 为实施和测试用户故事所需完成的任务已经被识别，并被团队估算。

特性

特性的完成定义可以跨越多个用户故事或者史诗（Epics）：

- 所有用户故事的组成要素，包括验收准则，都已经定义并通过了客户的验收；
- 设计已完成，并且没有已知的技术债；
- 编码已完成，并且没有已知的技术债或者未完成的重构；
- 单元测试已经被执行，并且达到所定义的覆盖等级；
- 对特性的集成测试和系统测试已经根据所定义的覆盖标准被执行过；
- 没有遗留的未修复严重缺陷；
- 特性文档已完成，可以包括发布记录，用户手册，和在线帮助功能等。

迭代

迭代的完成定义可以包括：

- 迭代包含的所有特性都已就绪，并且根据特性等级标准单独被测试过；
- 所有（由于迭代资源限制）不能被修复的非关键缺陷都被加入产品待办列表，并且被排定优先级；
- 迭代内所有特性的集成都已完成并且经过测试；
- 文档撰写完毕，并且经过审核和批准。

在这点上，由于迭代已经成功的完成，软件是潜在可发布的，但并不是所有的迭代都会成为一次发布。

发布

一次发布可以横跨多个迭代，发布的完成定义可以包括以下内容：

- 覆盖率：所有与发布相关的内容的测试基本要素都已经被测试覆盖。覆盖的充分程度由新增加的内容或者改动，以及它们的复杂程度、规模，以及失败相关的风险所决定。
- 质量：缺陷强度（如每天或每个事务有多少缺陷），缺陷密度（如找到的缺陷个数比上用户故事的个数，工作量，和质量属性），剩余缺陷数目在可接受范围之内，未解决的和剩余的缺陷会导致的结果（如严重度和优先级）已被理解并且接受，每个已识别的质量风险所对应的残余风险等级都被理解并且接受。
- 时间：如果已经到了预先决定的交付日期，则需要考虑与发布决策（是否发布版本）相关的业务因素。
- 成本：估算得到的生命周期成本应该用来计算已交付系统的投资回报（也即，计算出的开发和维护成本应大大低于产品所期望的总售价）。生命周期成本中的大部分来自于产品发布之后，逃逸到生产环境的缺陷所导致的维护成本。

3.3.2 应用验收测试驱动开发

验收测试驱动开发是一种测试优先的方法。测试用例在实施用户故事之前就已经创建。测试用例由包括开发人员、测试人员和业务代表在内的敏捷团队创建，可以是手工测试用例也可以是自动化测试用例。第一步是开展一个由开发人员、测试人员和业务代表一起参与分析、讨论和撰写的规格说明研讨会。任何在用户故事中不完整、不清楚或者错误的地方都会在这个过程中得到修复。
下一步是创建测试。这一活动可以由团队一起完成或者由测试人员单独完成。在任何情况下，测试都应该由一个独立的个人，如业务代表来确认。这里的测试是描述用户故事中特性的例子。这些例子会帮助团队正确地实施用户故事。由于例子和测试是相同的，二者经常可以互换使用。这项工作从基本的例子和开放的问题开始。

一般而言第一次测试都是正向测试，在没有意外或错误条件下确认正确的行为，比较活动执行的顺序是否如期望的那样进行。在正向路径测试完成后，团队应该撰写逆向测试并且覆盖非功能属性（如性能，易用性）。测试须以干系人都能理解的方式表述，包含以自然语言表达的语句，这些语句应该包括必要的前提条件，和任何可能的输入和相关输出。

例子必须覆盖所有用户故事的特性，并且不应该添加新内容到用户故事中。这意味，不应存在那些描述没有被写在的用户故事中某个方面的例子。另外，不应该有两个例子描述用户故事中的相同特性。

3.3.3 功能和非功能黑盒测试设计

在敏捷测试中，很多测试是由测试人员在开发人员写代码的同时创建的。因为开发人员编写的程序是基于用户故事和验收准则的，所以测试人员也要基于用户故事和验收准则创建测试。（有一些测试，如探

索性测试和一些其他的基于经验的测试，是后来在测试执行阶段创建的，见 3.3.4 节解释）测试人员可以应用传统的黑盒测试设计技术，如等价类划分、边界值分析、决策表和状态转换测试来创建这些测试。例如，当一个客户被若干可能选择购买的东西所限制的时候，边界值分析可以用来选择测试值。

在很多情形下，非功能测试需求可以被写到用户故事里。黑盒测试设计技术（如边界值分析）也可以被用来创建对非功能质量特性的测试。用户故事可能包含性能或者可靠性方面的需求。例如，一次给定的功能执行不能超过限定的响应时间，或者在给定操作中产生的失效次数必须小于特定数量。

黑盒测试设计技术的使用详见 ISTQB 基础级别大纲[ISTQB_FL_SYL]和 ISTQB 高级级别测试分析师大纲[ISTQB_ALTA_SYL]。

3.3.4 探索性测试和敏捷测试

探索性测试在敏捷项目中很重要，主要是因为测试分析只有有限时间可用，并且用户故事只有有限的详细资料。为了达到最佳结果，探索性测试应该与其他基于经验的技术结合，作为一个应对式测试策略的一部分，与其他测试策略混合使用，如基于风险的测试分析，基于需求的测试分析，基于模型的测试和面向可重用的测试。测试策略和测试策略混合在 ISTQB 基础级别大纲[ISTQB_FL_SYL]有详细的讨论。

在探索性测试中，测试设计和测试执行是在同时进行的，并且由一个准备就绪的测试章程所引导。测试章程提供相应的测试条件来覆盖一段固定时间的测试。在探索性测试期间，最新的测试结果会引导下一次的测试。在测试设计时，探索性测试可以同样使用前面所提的白盒和黑盒技术。

一个测试章程可以包含以下信息：

- 行动者（Actor）：系统的预期用户；
- 目的（Purpose）：章程的主题包括行动者想要达到的特殊目标，也就是测试条件；
- 设置（Setup）：为了开始测试执行所需要准备的事项；
- 优先级（Priority）：此章程的相关重要性，基于相应的用户故事的优先级或者风险等级；
- 参考（Reference）：规格说明（如用户故事），风险，或者其他信息源；
- 数据（Data）：任何用来执行此章程所需数据；
- 活动（Activities）：一个包含行动者想要与系统进行的交互动作（如“作为超级用户登录到系统上”）和感兴趣测试（包括正向和逆向测试）的清单（想法清单）；
- 准则（Oracle notes）：如何评估产品，从而来确定正确的结果（如，来获取屏幕上所发生的事件并且与用户手册的内容比较）；
- 变化（Variations）：可供选择的行动和评估来补充在活动部分中所描述的想法。

为了管理探索性测试，我们可以用一种被称为基于测试会话（session based）的测试管理方法。测试会话被定义为一个可以持续 60 分钟到 120 分钟而不被打断的测试周期。测试会话包括以下过程：

- 调研会话（学习和了解系统是如何运作的）；
- 分析会话（对功能或者特征的评估）；
- 深度覆盖（边界用例，场景，相互作用）

测试质量取决于测试人员对测试什么东西的相关问题的提问能力。提问包括以下例子：

- 哪些与系统有关的信息是最重要的，需要被发现的？
- 在什么情况下系统会失败？
- 如果……会发生什么？
- 当……应该发生什么？
- 是不是满足了客户的需要、需求和期望？

- 系统是否能在所有支持的升级路线内安装（或者必要时进行卸装）？

在测试期间，测试人员使用创造力、直觉、认知力和技能来找到产品可能的问题。测试人员也需要对所测试的软件、业务领域、软件如何使用和系统失效时如何决断有很好的知识和理解。

有一系列的启发式方法可以被应用在测试中。一个启发式方法可以引导测试人员如何执行测试和如何评估结果[Hendrickson]。例子如下：

- 边界
- CRUD（创建 **C**reate、读取 **R**ead、更新 **U**pdate、删除 **D**elete）
- 配置变动
- 中断（如，注销、关机、或者重启）

对于测试人员来说，尽可能的用文档记录下这个过程是很重要的。否则，就很难追溯系统内问题是如何被发现的。下列清单中提供了在文档中应记录的信息：

- 测试覆盖：哪些输入已经在测试中被使用，还有多少输入尚未被测试。
- 评估记录：在测试中的发现，例如，被测系统或者功能特性是否稳定，是否有缺陷被发现，根据当前状况计划下一步应该进行的工作，以及是否还有任何其他想法？
- 风险/策略清单：哪些风险已经被覆盖，最重要的风险条目中还有哪些依然存在，最初制定的风险应对策略是否依然有效，是否需要对策略做必要的修正。
- 不足、问题和异常：任何非预期的行为，任何方法的有效性问题，任何关于想法/测试尝试、测试环境、测试数据的考虑，对功能、测试脚本，或被测系统的错误理解。
- 实际行为：需要被保存的实际系统行为（例如，录像、屏幕截图、输出数据文件等）。

这些被记录的信息应被存放在工具中（例如，测试管理工具、任务管理工具、任务板），这些工具应支持对信息的状态管理，通过对信息的汇集和整理，方便干系人及时了解进行中的各种测试的状态。

3.4 敏捷项目中的工具

在 ISTQB-FL 基础级大纲中，测试工具部分描述的信息对敏捷团队依然有效。当然，相对于传统项目，在敏捷项目中并不是所有的工具都以相同方式使用，有些工具在敏捷项目中有着更现实的意义。例如，虽然敏捷团队也可以使用测试管理工具、需求管理工具、事件管理工具（缺陷跟踪工具），但敏捷团队更倾向于使用“一站式”解决方案（例如，应用生命周期管理或者任务管理），这些解决方案可以更好的支持敏捷开发，例如任务板、燃尽图或者用户故事。对于敏捷团队来说，配置管理工具是非常重要的，这是因为敏捷团队大量的使用覆盖多个层次的自动化测试，配置管理工具需要存储、管理、集成这些自动化测试工件。

作为 ISTQB-FL 基础级大纲测试工具部分内容的扩充，在敏捷项目中，测试人员还需要掌握下面小节中工具的使用。这些工具是整个团队都在使用的，目的是支持信息共享和团队协作，而沟通和协作是关键的敏捷实践。

3.4.1 任务管理和追踪工具

某些敏捷团队使用物理介质的故事/任务板（例如，白板，公告板）来管理和追踪用户故事、测试，以及其他迭代中的任务的信息。有些团队则会使用应用生命周期管理和任务管理软件，例如，电子任务板。这些工具的主要目的如下：

- 记录用户故事以及与之相关的开发和测试任务，以确保在迭代过程中没有遗漏任务。

- 记录团队成员对任务的估算，并自动计算开发一个用户故事所需要的资源，进而可以支持进行高效的迭代计划会议。
- 在用户故事中整合开发任务和测试任务，以获得全局视角，可以包含开发一个用户故事所需的资源。
- 整合开发人员和测试人员的任务状态更新信息，自动整理和提供当前用户故事、当前迭代以及整个发布等不同层级的状态汇总报告。
- 对每个用户故事、迭代和发布的当前状态提供一个直观的描述（通过度量、图和面板），使得所有的干系人，包括分布在不同地点的所有人，能够快速的获得状态信息。
- 和配置管理工具集成，对代码的检入和每个任务对应的版本进行自动的记录，有时也会自动的更新任务的状态。

3.4.2 沟通和信息共享工具

除了 e-mail 电子邮件、文档和口头的交流以外，敏捷团队还经常使用另外 3 种类型的工具来进行沟通和信息共享：wiki、即时通讯和桌面共享。

Wiki 允许团队针对项目的各个方面创建和共享一个在线的知识库，包括：

- 产品特性图、特性讨论、原型图、白板讨论照片和其他信息
- 团队中其他成员发现的对开发和测试有用的工具和/或技术
- 关于产品状态的度量、图和面板，当 wiki 和其他工具集成在一起的时候，例如构建服务器和任务管理系统，由于工具能够自动的更新产品状态，此时 wiki 尤其有用
- 团队成员之间的对话，与即时通讯和 email 类似，但是这种方式可以和团队中所有人共享

即时通讯、电话会议和视频聊天工具可以提供以下好处：

- 允许团队成员实时直接沟通，尤其是对分布式团队
- 让分布式团队参与站立会议
- 通过 IP 语音技术减少电话费用，避免造成分布式团队成员之间沟通的成本约束

桌面共享和抓图工具可以提供以下好处：

- 分布式团队可以进行产品演示、代码评审，甚至结对
- 在每个迭代的最后，抓取产品的示例发布到团队的 wiki 上

在敏捷团队中，这些工具可以用来补充和扩展，而不是代替面对面的交流。

3.4.3 软件构建和分发工具

本大纲的前面讨论过，在敏捷团队中一个关键实践就是软件的每日构建和部署。这需要使用持续集成工具和构建发布工具。1.2.4 节描述了这些工具的作用、好处和风险。

3.4.4 配置管理工具

在敏捷团队中，配置管理工具不仅可以用于存储源代码，还可以用于自动化测试，但是手工测试和其他的测试工作产品常常和产品源代码保存在相同的资源库中。这样可以提供被测软件的版本和具体的测试版本之间的可跟踪性。版本控制系统的主要类型，包括集中的源代码控制系统和分布式的版本控制系统。具体敏捷项目中选择哪种版本控制系统，取决于团队的规模、架构、地点和与其他工具集成的需求等。

3.4.5 测试设计、实施和执行工具

某些工具对于敏捷测试人员在软件测试过程中某些点上是有用的。虽然大部分工具都不是新的或者只针对敏捷的，他们对敏捷项目的快速变更提供了重要的能力。

- 测试设计工具：在为一个新的功能快速设计和定义测试时，使用像思维导图这样的工具变得越来越流行。
- 测试用例管理工具：敏捷中测试用例管理工具可能是整个团队的生命周期管理或者任务管理工具的一部分。
- 测试数据准备和生成工具：当测试应用程序需要大量的数据和数据组合时，能够在应用程序的数据库中生成数据的工具是非常有用的。随着敏捷项目中产品的变化，这些工具可以帮助重新定义数据结构，重构生成数据的脚本。这样能够在发生变化时快速更新测试数据。有些测试数据准备工具使用原始生产数据作为原始输入，但是使用脚本将其中的敏感数据删掉或者匿名化。还有一些测试数据准备工具可以帮助验证大量的数据输入或者输出。
- 测试数据加载工具：用于测试数据生成之后，加载到应用程序中。手工的数据录入常常需要花费大量的时间，而且容易出错，但是数据加载工具可以使得这个过程可靠并有效率。事实上，很多数据生成工具包括集成数据加载组件。有时，也可能会使用数据库管理工具来加载大量的数据。
- 自动的测试执行工具：有一些测试执行工具更适合敏捷测试。这些工具既有商用也有开源的用来支持测试优先的方法，例如：行为驱动开发、测试驱动开发和验收测试驱动开发。这些工具允许测试人员和业务人员通过表格或者采用关键字的自然语言来描述期望的系统行为。
- 探索性测试工具：在执行探索性测试会话时这些工具能够捕获和记录在应用程序上执行的活动，记录采取的措施，它们对测试人员和开发人员是有益的。由于捕获了在失效发生前的行为并且可以用于向开发人员报告缺陷，这在发现缺陷时是有帮助的。如果测试最终包括自动化回归测试套件，在一个探索性测试会话中记录执行的日志步骤可以证明是有益的。

3.4.6 云计算和虚拟化工具

虚拟化允许一个单独的物理资源（服务器）被当作许多分开的、小的资源来操作。当使用虚拟机或者云实例的时候，团队有很多可用于开发和测试的服务器。这有助于避免因为等待物理服务器而导致的延期。通过使用很多虚拟化工具中集成的快照功能，配置或者恢复一台服务器将更加高效。现在有些测试管理工具在检测到失效时使用虚拟化技术对服务器进行快照，允许测试人员将快照分享给开发人员，以对失效进行调查。

4. 参考文献

4.1 标准

[DO-178B] RTCA/FAA DO-178B, 机载系统和设备认证中的软件考量, 1992.
[ISO25000] ISO/IEC 25000:2005,软件工程 – 软件产品质量需求和评估 （SQuaRE）, 2005.

4.2 ISTQB 文档

[ISTQB_ALTA_SYL] ISTQB 高级测试分析师大纲, 2012 版
[ISTQB_ALTM_SYL] ISTQB 高级测试经理大纲, 2012 版
[ISTQB_FA_OVIEW] ISTQB 基础级敏捷测试概述, 1.0 版
[ISTQB_FL_SYL] ISTQB 基础级大纲, 2011 版

4.3 书籍

[Aalst13]Leo van der Aalst and Cecile Davis, “TMap NEXT® in Scrum,” ICT-Books.com, 2013.
[Adzic09] GojkoAdzic, “Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing,” Neuri Limited, 2009.
[Anderson13]David Anderson, “Kanban: Successful Evolutionary Change for Your Technology Business,” Blue Hole Press, 2010.
[Beck02] Kent Beck, “Test-driven Development: By Example,” Addison-Wesley Professional, 2002.
[Beck04]Kent Beck and Cynthia Andres, “Extreme Programming Explained: Embrace Change, 2e” Addison-Wesley Professional, 2004.
[Black07] Rex Black, “Pragmatic Software Testing,” John Wiley and Sons, 2007.
[Black09] Rex Black, “Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, 3e,” Wiley, 2009.
[Chelimsky10] David Chelimsky et al, “The RSpec Book: Behavior Driven Development with Rspec, Cucumber, and Friends,” Pragmatic Bookshelf, 2010.
[Cohn04] Mike Cohn, “User Stories Applied: For Agile Software Development,” Addison-Wesley Professional, 2004.
[Crispin08] Lisa Crispin and Janet Gregory, “Agile Testing: APracticalGuide for Testers and Agile Teams,” Addison-Wesley Professional, 2008.
[Goucher09]Adam Goucher and Tim Reilly, editors, “Beautiful Testing: Leading Professionals Reveal How They Improve Software,” O'Reilly Media, 2009.
[Jeffries00] Ron Jeffries, Ann Anderson, and Chet Hendrickson, “Extreme Programming Installed,” Addison-Wesley Professional, 2000.
[Jones11] Capers Jones and Olivier Bonsignour, “The Economics of Software Quality,” Addison-Wesley Professional, 2011.
[Linz14]Tilo Linz, “Testing in Scrum: A Guide for Software Quality Assurance in the Agile World,” RockyNook, 2014.
[Schwaber01] Ken Schwaber and Mike Beedle, “Agile Software Development with Scrum,” Prentice Hall, 2001.

[vanVeenendaal12]Erik van Veenendaal, “The PRISMA approach”, UitgeverijTuteinNolthenius, 2012.
[Wiegers13] Karl Wiegers and Joy Beatty, “Software Requirements, 3e,” Microsoft Press, 2013.

4.4 敏捷术语

每一章的开头包括了可以从 ISTQB® 术语表中找到的关键字。对于通用的敏捷术语，我们依赖于以下广泛认可的提供定义的网络资源。

http://guide.Agilealliance.org/
http://whatis.techtargetcom/glossary
http://www.scrumalliance.org/

如果在本文档中发现不熟悉的敏捷相关的术语，我们鼓励读者阅读这些网站。这些链接在本文档发布的时候是可以访问的。

4.5 其他参考文献

以下参考文献指向网络或者其他地方可以获得的信息。虽然在本大纲发布的时候已经检查过这些参考文献的可用性，但是 ISTQB® 无法保证这些参考资料永远可用。

- [Agile Alliance Guide] Various contributors, http://guide.Agilealliance.org/.
- [Agilemanifesto] Various contributors, www.agilemanifesto.org.
- [Hendrickson]: Elisabeth Hendrickson, “Acceptance Test-driven Development,” testobsessed.com/2008/12/acceptance-test-driven-development-atdd-an-overview.
- [INVEST] Bill Wake, “INVEST in Good Stories, and SMART Tasks,” xp123.com/articles/invest-in-good-stories-and-smart-tasks.
- [Kubaczkowski] Greg Kubaczkowski and Rex Black, “Mission Made Possible,” www.rbcs-us.com/images/documents/Mission-Made-Possible.pdf.

5. 索引