第 13 章

企业Web应用中的敏捷测试和瀑布测试

13

Kristan Vingrys，QA咨询师

13.1 简介

同是企业Web应用程序项目，一个用敏捷，一个用瀑布流程，它们的测试策略会有何不同？在二者中，测试的关注点都在于告诉业务客户这个应用程序做了哪些事情，同样也要消除应用程序作为产品交付以后的失败风险。它们的主要区别不是测试本身，而是何时执行测试、由谁执行测试。测试的每个阶段都可以在系统就绪后随时开始，无须等待前一个测试阶段完成。

从未涉足敏捷项目，或是刚启动某个敏捷项目并在寻找指导建议的读者都可以看看这篇文章，它正是为你们而写。文中的信息虽并非笔者新创，但亦是收集整理的结果，希望这些信息能帮助你们朝着更为敏捷的过程迈进。

敏捷项目的测试阶段跟瀑布项目的测试阶段几乎毫无二致。每个阶段都有准出标准，但是在进入某个测试阶段之前，是不需要等待整个应用程序完成的。只要应用程序所完成的部分足以进入下一个测试阶段就行。因为测试的对象是一个已完成的功能，而不是一个发布，所以测试阶段是在持续并行进行的。于是就有了一大堆回归测试。这便也意味着回归测试是测试自动化的基础。对于敏捷项目而言，环境与资源也是要注意的地方，因为对测试环境的需求会来得更早、更加频繁。

“快速失败”是敏捷项目的一句格言，它的含义是尽可能早地判断出应用程序无法满足业务需求。要做到这一点，就需要不断地对解决方案是否满足业务需求进行检测，一旦不满足，就要尽早把问题解决。敏捷团队成员——开发人员、测试人员、架构师、业务分析师以及业务代表等人都关注于尽早交付业务价值。所以，测试需要所有团队成员协力来做，它不仅仅是测试人员的责任。

13.2 测试生命周期

从测试生命周期就可以看出瀑布和敏捷项目之间最大的差异。瀑布项目对每个阶段都有严格的准入和准出标准，而且只有前一个阶段结束，才可以进入下一个阶段。而敏捷项目则会尽可能早的启动测试阶段，并且允许不同的阶段出现重叠。敏捷项目也有一些结构性的内容，如准出标准，但它没有严格的准入标准。

在图13-1中，你一眼就能看出敏捷项目和瀑布项目的测试生命周期差异所在。在敏捷项目中，业务分析师、测试分析师和业务代表等人一起讨论某个想法的行为，它如何适配于整体目标，怎样去验证它是否完成了该做的事情。这些分析构成了功能测试、用户验收测试和性能测试的基础。做完分析之后便开始功能的开发，单元测试、集成测试、探索性测试、非功能性测试（及数据验证——如果有这一项的话）也纷纷开始。不过只有等系统可以作为产品运行时才开始进行产品验证。

测试阶段没有严格的准入标准，这就意味着只要时机合适，随时都能开始。因为每个测试阶段对于确保应用程序的质量都至关重要，所以便应该尽早开始每个阶段的分析工作，这可以帮助人们修正设计，找出问题，为以后节省出大量的时间。下面是敏捷项目的一些准出标准。

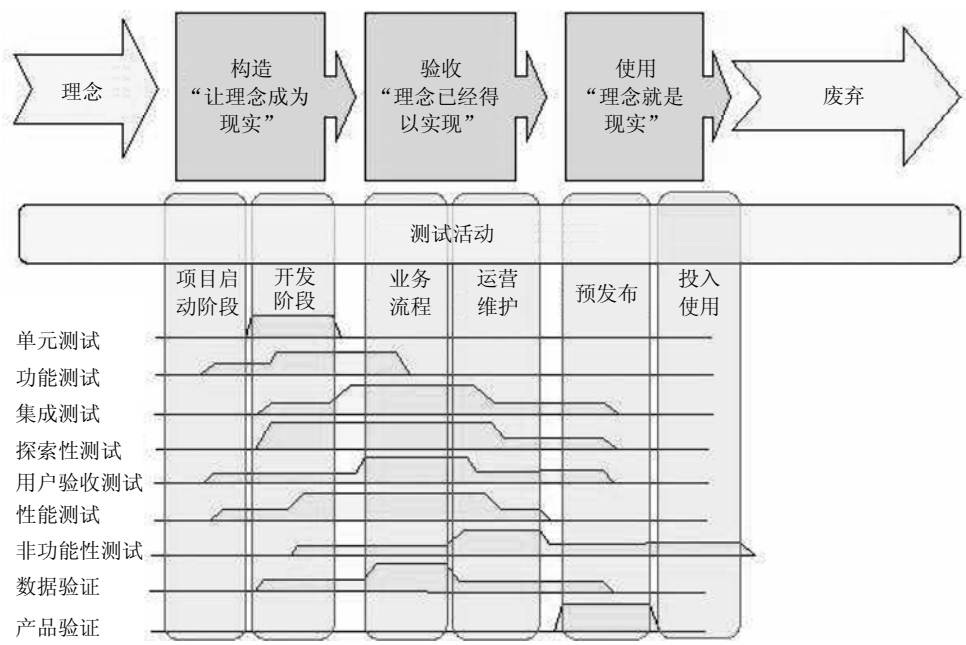

图13-1 敏捷项目和瀑布项目的测试生命周期

单元测试的标准：

- 100%自动化；
- 100%通过；
- 超过90%的代码覆盖率；
- 纳入持续构建。

集成测试的标准：

- 100%自动化；
- 100%通过；
- 纳入持续构建。

功能测试的标准：

- 90%以上自动化；
- 100%通过；
- 所有的自动化测试都纳入持续构建。

探索性测试的标准：

- 测试分析师对应用程序的质量有信心。

用户验收测试的标准：

- 业务代表认可应用程序满足需求；
- 用户认可应用程序的可用性。

性能测试的标准：

- 100%自动化；
- 业务人员认可应用程序满足了业务性能需求；
- 性能测试可以重复执行。

非功能测试的标准：

- 业务人员认可非功能需求得到了满足；
- 操作人员认可非功能需求得到了满足。

数据验证测试的标准：

- 确信数据被正确移植。

产品验证的标准：

- 应用程序在产品环境中正确安装。

瀑布项目的测试生命周期限制某个测试阶段只能在前面的测试阶段走完以后才能开始。从理论上讲这也是说得过去的，因为后面的测试阶段会依赖于前面的执行通过（如果某些功能不正确，就用不着再对它做性能测试了）。但是，要是非得等到所有功能都正确实现以后才开始性能测试，可就一点道理都没有了。敏捷项目会在适当的时候启动每一个测试阶段，这样可以尽早找出问题，让团队有更多的时间解决问题。但是敏捷项目的准出标准跟瀑布项目是一样的。除非功能都验证无误，否则就不能认为性能测试已经完成。

13.3　测试分类

敏捷项目跟瀑布项目的测试分类几乎是一样的。其差别主要在于大部分精力投入的地方和每个测试阶段所执行的时机。敏捷项目倾力关注单元测试和功能测试，从而为稍后执行的测试阶段创造出高质量的代码，于是后期就不会发现本应在早期发现的缺陷，并能专注于所需要测试的领域。而瀑布项目就有一个常见的问题，后期测试的焦点总是放在找出本来应该在前期被发现的缺陷身上。于是修复缺陷的成本提高了，测试工作的投入翻番了，测试的关注点也偏离了。

瀑布项目和敏捷项目另外一个巨大的不同在于测试自动化。敏捷项目力求在所有测试领域内都达到100%自动化。测试跟持续构建系统集成在一起，于是当代码发生变化时，这个变化会被自动检测到，应用程序被构建起来，然后所有测试都会被执行。

测试驱动开发（TDD）在敏捷项目中很常用，用这种方法的时候，测试用例比代码要先一步创建。于是我们就可以越来越多地看到为代码和功能创建的测试用例。用自动化测试来驱动开发，然后消除重复，这种做法可以保证无论复杂度多高，任何开发者都可以写出可靠的、没有bug的代码。TDD常用的地方是单元测试，但也同样可以用于功能测试、集成测试、用户验收测试和性能测试。

单元测试

单元测试又称白盒测试，它要测试所开发出来的每一个模块。瀑布项目不关注这个测试阶段，

而且大多数时候即便有的话也是随意为之。敏捷项目强调单元测试，而且会把所有单元测试都自动化。自动化的单元测试是敏捷项目的基础，对持续集成和重构起辅助作用。

单元测试应当考虑以下几点：

- 用stub和mock来消除对外部接口的依赖；
- 由创建代码的开发人员编写单元测试；
- 单元测试要能自动化执行，而且要被包含在持续开发构建中；
- 单元测试之间不能有依赖，让每一个单元测试都能独立执行；
- 任何开发人员都要能在他自己的机器上执行单元测试；
- 用代码覆盖率来判断哪些部分的代码没有被单元测试覆盖；
- 在检入代码的修改之前，要保证单元测试100%通过。

任何测试失败都表示构建失败。

功能测试

功能测试常常跟系统测试相关联，它的重点是测试应用程序的功能（包括负面测试和边界条件）。在瀑布项目中，测试团队一般是在这个阶段开始测试工作。测试团队的成员会一直等下去，直到开发者完成了所有的功能，并通过所有单元测试之后才进入功能测试阶段。敏捷项目把功能拆分成故事，每次迭代开发一定数量的故事。每个故事都有一些验收标准，这些标准一般都是由业务分析师和测试分析师制定的，它们也可以看作跟测试条件类似。测试分析师会根据验收标准创建测试用例，用它们来检测代码行为的完成程度。故事一旦编码完成，而且单元测试运行通过以后，就可以运行功能测试来判断是否满足验收标准了。这就意味着在敏捷项目中，只要第一块功能已经完成编码，功能测试就可以启动，而且会贯穿整个项目的生命周期。

功能测试应当考虑以下几点。

- 要能自动化执行，并且进入持续构建（如果测试运行时间很长，也可以只在开发持续构建中包含一小部分精挑细选的功能测试，而在系统集成持续构建中包含全部功能测试）。
- 在编码之前写下测试意图，代码完成后对测试进行实现。
- 把通过所有功能测试作为故事完成的条件。
- 在应用程序安装到另外一个环境（如试机环境或产品环境）上的时候需要执行功能测试。

任何测试失败都表示构建失败。

探索性测试

探索性测试又称随机测试。瀑布项目在它们的测试策略中没有这种测试类型，不过大多数测试人员都会多少做一些探索性测试。在敏捷项目中这是个很关键的测试阶段，因为它可以用来检测自动化测试的覆盖率，并收集对于应用程序质量的反馈。它为测试人员和业务分析师提供了一种结构化的方式去使用、去探索系统，从而找到缺陷。如果探索性测试在某个功能区域内找到了大量缺陷，那这部分的自动化测试用例就要被审核。

探索性测试应当考虑以下几点：

- 在系统集成环境中执行；
- 在较高的层次（如在wiki中）上监测探索性测试活动；
- 用自动化的环境设置方式来减少搭建环境的时间；
- 将破坏性测试作为探索性测试的一部分。

集成测试

应用程序在作为产品部署的时候，需要各个部分的协作；集成测试就是把这各个独立的部分集成起来进行测试。瀑布项目不但会把应用程序的各个独立模块进行集成，还会把那些虽然不属于项目的一部分，但是要开发当前应用程序就必须用到的一些应用程序也做集成。在敏捷项目中，独立模块间的集成是被持续构建所覆盖的，所以集成测试的关注点就是那些不属于当前项目的外部接口。

集成测试应当考虑以下几点。

- 在执行集成测试的时候，需要考虑到当前迭代还没有开发的功能；
- 写一些集成测试来定位特殊的集成点，以协助代码调试，即便功能测试会调用到这些集成点也无妨；
- 将集成测试自动化，放到系统集成持续构建中。

任何测试失败都表示构建失败。

数据验证

如果项目需要移植既有数据的话，就要进行数据验证。它可以保证现有的数据被正确地移植到新的Schema中，新的数据被添加进来，旧的数据被移除。这种类型的测试在瀑布项目和敏捷项

目中是被同等对待的，除了敏捷项目会尽力把它自动化以外。

数据验证应该考虑以下几点：

- 在系统集成环境、试机环境和产品环境中都要执行；
- 尽可能地自动化；
- 要把测试放到系统集成持续构建中。

任何测试失败都表示构建失败。

用户验收测试（UAT）

UAT关注的是完整的业务过程，它用来确保应用程序能按照业务的处理方式工作并能满足业务需求。它同样还要从客户、消费者、管理员等各种用户的角度出发考虑应用程序的可用性。在瀑布项目中，这个阶段通常就被用来找出应该在早期阶段就被发现的bug。业务人员也会在这个阶段验证开发团队交付的应用程序的质量。敏捷项目用UAT来确保应用程序满足业务需求，因为等到进入这个测试阶段的时候，代码质量已经较高了。在敏捷项目中，业务人员从早期的测试阶段就开始参与，所以他们对交付的东西有更多的信心。

UAT应该考虑以下几点：

- 首先进行手工测试，等它验证了系统行为以后再把它自动化；
- 把自动化测试放到系统集成持续构建中；
- 让应用程序的最终用户亲自将整个程序运行一遍，不过项目的测试人员要在旁边协助；
- 在试机环境下执行UAT，用作验收；
- 只要完成了一个业务过程或者一个主要的UI组件，就要执行UAT。

任何测试失败都表示构建失败。

性能测试

性能测试涵盖的范围比较大，不过一般可以分成以下三类。

- 容量测试：独立测试核心功能组件的容量。例如，可以支持多少用户并发搜索？一秒钟能做多少次搜索？等等。容量测试被用来精确度量系统的极限，还可以对容量规划和系统的扩展性起辅助作用。
- 负载测试：侧重于系统在负载下的表现。负载应该要体现出用户所期望系统可以经受得

住的流量。

- 压力测试：关注系统在压力下的表现。比较常用的技术是浸泡测试（soak test）；在浸泡测试中，系统会在一定的负载下持续运行一段时间，用来找出长期问题，如内存泄漏、资源泄漏等。压力测试还会覆盖到故障转移和恢复，例如让正在工作的集群中的一台服务器出现问题，检查是否可以做到故障转移和恢复。

瀑布项目直到项目接近尾声的时候才做性能测试，这个时候应用程序已经“完成了”开发，通过了单元测试和功能测试。而敏捷项目则会尽快启动性能测试。

性能测试应该考虑以下几点。

- 在功能测试中设置一些性能度量，例如一个测试第一次运行要花多长时间，接下来每次又要花多久，把这些时间所占的百分比做一下比较（上升表示有问题，下降表示良好）。
- 把一些性能测试放到系统集成持续构建中去做。
- 一旦一个业务过程，或是某个规模比较大的功能或接口完工，就要做性能测试。
- 只有在试机环境中运行的时候才签收性能测试。

任何测试失败都表示构建失败。

非功能性测试

非功能性测试所涵盖的范围很广，性能测试通常也属于这个话题。但是因为性能测试是企业解决方案中很重要的一部分，而且需要不同的资源和技能集，所以就被划出来单独成为一个测试阶段。非功能性测试一般都包含有这些内容：可操作性（包括监控、日志、审计/历史记录）、可靠性（包括故障转移，单个组件故障，完整故障，接口故障）以及安全性。无论瀑布项目还是敏捷项目，在这个测试阶段都会遇到重重阻碍，二者的差异不太突出。

非功能性测试应该考虑以下几点：

- 非功能性需求一般都是很难捕获的，而且即便被捕获，也很难进行度量（例如，99.9%的无故障时间）；
- 尽可能让所有的非功能测试都自动化执行，把它们也都放到系统集成测试环境中；
- 在定义测试用例的时候，让真正要监控产品环境并对其提供支持的人也参与进来；
- 在应用程序成为产品以后，非功能测试或监控还要继续。

回归测试

在瀑布项目中，回归测试无论从时间上还是费用上来看，都算得上是成本最高的测试阶段了。如果在项目晚期（如UAT阶段）发现了缺陷，再构建应用程序的时候，就得把所有单元测试、功能测试和用户验收测试重新运行一遍。因为大多数瀑布项目都没有自动化测试，所以回归测试的开销就很大。敏捷项目以持续构建和自动化测试来应对回归测试，使每次构建都可以进行回归测试。

回归测试应该考虑这一点：

- 每次迭代结束时都做一下手工测试（如果规模很大的话，就进行拆分，做到每三四次迭代就能执行完一次），收集早期反馈。

产品校验

产品校验会在把产品交付给用户使用之前，审查产品环境中的应用程序，看看所有内容是否都已经正确安装，系统的操作性如何。而有些测试还是只能到了产品环境中才能完整运行的，最好尽快将其完成。无论是瀑布项目还是敏捷项目，产品校验的方式并无二致。

产品校验应该考虑以下几点：

- 让最终用户来做产品校验测试；
- 趁着产品系统还没有正式上线，从这个测试阶段的早期就要尽可能多地执行自动化回归测试。

瀑布项目和敏捷项目的测试阶段还是很相似的，主要差异就是每个测试阶段所关注的重点和执行时机。敏捷项目中有大量的自动化测试，用持续集成来减少回归测试对项目带来的影响。在瀑布项目中，如果发现应用程序的质量低下，那么在晚期再去执行前期的测试就是很常见的事情（如在UAT的时候作功能测试）。敏捷项目可以减少测试中的浪费，在早期发现问题，让团队在交付应用时增强信心。

13.4 环境

在开发过程的各个阶段都要用到测试环境，从而确保应用程序的正常运行。越到后期，测试环境与预期的产品环境就会越相似。测试环境一般都会包括一个开发环境，让开发者集成代码并运行一些测试。系统集成环境跟开发环境有些类似，不过它会集成更多的第三方应用程序，也许还有大批量的数据。试机环境几乎是产品环境的镜像，也是应用程序变成产品之前的最后

一个阶段。

敏捷项目和瀑布项目所需的环境没太大区别，其中一个不同之处在于，敏捷项目从项目伊始直至项目结束，都要用到所有的环境。在敏捷项目中，保证所有的环境都一直正常工作也是很重要的。无论因为什么原因让某个环境出现故障，都要立刻让它重新工作起来。在这个话题上，敏捷和瀑布还有另外一点差异，那就是环境的计划和资源分配对它们的影响不同，尤其是当各种环境都被项目之外的团队进行管理的时候，其差异尤为显著。

开发集成环境

开发者在开发环境中集成代码，开发应用程序。瀑布项目对开发环境的重要性不会考虑太多：开发环境中的代码一直都不能工作，到了开发者需要彼此集成代码的时候才想起来要用，而这时项目已经临近尾声。在敏捷项目中，开发环境是整个开发工作中不可分割的一部分，在开始编码之前就必须准备就绪。这个环境会被用来持续地集成代码和运行测试套件。无论因为什么原因造成环境故障，都要立刻修复。

开发环境应该考虑以下几点。

- 集成代码、构建和测试的时间加起来不要超过15分钟。
- 每个开发人员所用的环境跟开发环境要保持一致（硬件环境可以不一样，但是软件环境一定要一样）。
- 所用的数据要尽可能跟产品数据保持一致。如果产品数据过于庞大，难以载入，也可以只截取一部分。在每个发布周期的开始阶段，这些数据要从产品数据中重新更新。
- 管理这个环境的责任应该落在项目团队身上。
- 向这个环境中部署的频率大约是以小时计算。
- 自动部署。

系统集成环境

系统集成环境用来将所开发的应用程序和其他应用程序进行集成。在瀑布项目中，这个环境（如果有的话）只会在项目临近尾声的时候才会用到，而且倾向于多个项目共用一个集成环境。在敏捷项目中，一旦开始编码，这个环境就要准备就绪。应用程序会被频繁部署到这个环境中，继而开始执行功能测试、集成测试、可用性测试和探索性测试等等。应用程序的演示就是在这个环境中进行。

系统集成环境应该考虑以下几点。

- 集成点应该被真正的外部应用程序所代替。外部应用程序应该处于测试状态，而非真正的生产版本。
- 把产品环境的架构复制过来。
- 在这个环境中所使用的数据应该是产品环境数据的副本，每个发布周期的开始阶段，都要从产品数据中重新更新。
- 建立运行这个环境中所有测试的系统集成持续构建。
- 管理这个环境的责任应该落在项目团队身上。
- 向这个环境中部署构建的频率大约是以天计算。
- 自动部署应用程序。

试机环境

试机环境用来验证应用程序可以部署为产品，而且工作正常。为了达到这个目的，试机环境应该完全复制产品环境，包括网络配置、路由器、交换机以及计算机性能等等。在瀑布项目中，这个环境往往需要“预订”，也要有一个计划，计划在这个环境中进行多少次部署以及何时进行部署。敏捷项目对这个环境不像对开发环境和集成环境那样依赖；不过在项目的整个生命周期中，还是需要常常进行部署。

试机环境应该考虑以下几点。

- 这里的数据应该是产品数据的完整副本，每次应用程序部署前都要更新。
- 用它来验收UAT，性能测试和非功能测试（稳定性、可靠性等等）。
- 向这个环境中部署构建的频率大约是以迭代计算，如每两周一次。
- 管理这个环境的人应该就是管理产品环境的人，让他提前接触应用程序并进行知识传递。
- 自动部署应用程序。

产品环境

产品环境是应用程序正式上线时的环境。产品校验测试就在这个环境中执行，同时会做一些度量以检验测试成果。瀑布项目和敏捷项目在这里的区别不大。在敏捷项目中，发布过程会尽力做到自动化，为向产品环境中频繁发布提供条件。

产品环境应该考虑以下几点。

- 在应用程序正式上线之前（或者刚刚上线之后），在产品环境中做回归测试。
- 要做度量，以检验测试成果，例如在前三个月到六个月之前，用户报告了多少问题，问

题的严重性如何。

无论是哪种项目，要保证时间期限，就都得做到在需要用到环境的时候就有一个环境可供使用。瀑布项目会设法遵守严格的计划，便于对环境做安排。敏捷项目的灵活性更强。也许有了足够的功能就可以进入试机环境了，或者业务人员会决定产品提前上线。为了做到向试机环境快速部署，然后进入产品环境，就要有系统集成环境以及相关过程的辅助。

13.5　问题管理

问题包括缺陷（bug）和变更请求。瀑布项目有着严格的缺陷和变更请求管理，而敏捷项目饱含变化，所以就没有那么严格的变更管理。如果需要有变更，就创建一个（或是一些）故事，放到待处理事项里面。高优先级的故事会放到下一次迭代中完成。

缺陷管理在敏捷项目中依然适用。如果有人发现一个正在开发故事有缺陷，大家就会进行非正式的沟通，发现缺陷的人把它告诉开发人员，缺陷会立刻被修复。如果某个缺陷并不属于当前迭代中开发的故事，或者属于当前故事，但并不严重，那就用缺陷跟踪工具记录下来。这个缺陷会被当做故事处理；也就是说，会创建一张故事卡，让客户排优先级，放到待处理故事里面。团队需要进行权衡：要找到问题所在，为大家理解这个问题并安排优先级提供足够的信息，但又不能在一个对客户而言优先级并不是很高的缺陷上面花太多时间。

瀑布项目和敏捷项目的缺陷内容（描述、组件、严重程度等）是一样的，除了一个字段以外：敏捷项目多了一个字段——业务价值，如果可能的话就用币值描述。这个字段表示如果缺陷被解决的话可以带来多少业务价值。将业务价值跟缺陷关联以后，客户就更好地判断这个缺陷跟新功能相比是不是更有价值，是不是应该有更高的优先级。

13.6　工具

所有项目都要用到工具，只是程度不同。瀑布项目用工具来强化过程以及提高效率，这有时会造成冲突。敏捷项目用工具辅助提升效率，与过程无关。在敏捷项目中，所有的测试都应该可以在任何团队成员的个人环境中运行，也就是说，所有人都可以使用那些自动化测试用例的工具。所以敏捷项目中会经常用到开源产品，这又意味着使用这些工具需要不同的技能。开源工具不像商业工具那样有齐备的文档和完善的支持，用这些工具的人要有很强的编码能力。如果有人编程能力偏弱，就可以通过跟人结对来提升个人技能。在敏捷项目中也可以使用商业工具，但是大多数商业工具在开发的时候都没有考虑敏捷过程，所以跟敏捷项目匹配起来就不太容易。而且要让

商业工具跟持续集成配合，可能要写很多代码才行。

项目中应该考虑为下面一些测试任务使用工具：

- 持续集成（如CruiseControl，Tinderbox）；
- 单元测试（如JUnit，NUnit）；
- 代码覆盖率（如Clover，PureCoverage）；
- 功能测试（如 HttpUnit，Selenium，Quick Test Professional）；
- 用户验收测试（如Fitness，Quick Test Professional）；
- 性能测试（如JMeter，LoadRunner）；
- 问题跟踪（如BugZilla，JIRA）；
- 测试管理（如Quality Center）。

13.7 报表与度量

度量数据是用来衡量软件质量和测试成果的。在瀑布项目中，有些测试度量指标需要在测试之前就把所有测试用例都写好，而且仅在应用程序开发完毕时进行一次测试。这种指标包括：每个测试用例执行的时候发现多少缺陷，每天执行的测试用例会发现多少缺陷。这些度量数据收集起来以后，便用来判断应用程序是否已经就绪并可以发布。在敏捷项目中，功能完成的时候测试用例就已经写好且运行完毕，这就意味着用来度量瀑布项目的一些指标是无法应用在这上面的。

回到收集度量数据的原因上来——衡量软件质量和测试成果，你可以看看下面这些概念。

- 用代码覆盖率度量测试效果；这对于单元测试尤其有效。
- 在探索性测试阶段发现的缺陷数量可以说明单元测试和功能测试的效果。
- 在UAT阶段发现的缺陷表示先期的测试并不像UAT一样充分，我们应该关注业务过程，而不是软件的bug。如果UAT发现了很多功能性问题，而不是软件的bug，这就表示团队对故事或是变化的需求理解不足。
- 故事完成以后所发现的缺陷数量能够作为衡量软件质量的好手段。这些缺陷包括在集成测试、非功能测试、性能测试和UAT测试中发现的缺陷。
- 缺陷重现率。如果缺陷常常重现，软件质量就很低。

13.8　测试角色

测试角色并不是跟单个资源一一对应的。一个资源可以担任多个测试角色，一个测试角色也可以由多个资源负责。下面列出的这些角色是确保项目测试效果所必需的。一个优秀的测试人员应该具备所有这些角色的特征。敏捷项目和瀑布项目都有这些角色，只是扮演这些角色的人不同。在敏捷项目中，所有团队成员都会扮演一些测试角色，在图13-2中展示了一个例子，你可以看到在敏捷项目中，每个团队成员都是怎样扮演各个角色的。这并不是强制性的规定；每个团队各有差异，不过这种做法也算得上是不错的组合。

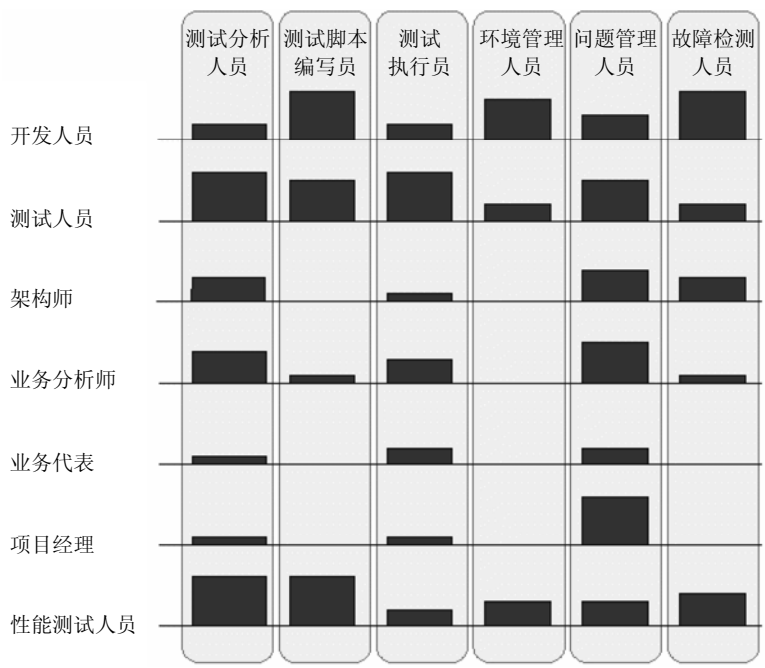

图13-2　不同团队成员的测试角色

测试分析人员

测试分析人员要了解需求、架构、代码等各个产物，从而判断哪些需要做测试，哪些是测试要重点关注的地方。

在瀑布项目中一般是有一个（或多个）资深的资源来担任这个角色。他们检查相关文档（需求、设计、架构），编写测试计划，编写高层的测试用例描述，然后把所有的东西都交给一个

初级员工，让他填补详细的测试脚本。敏捷项目鼓励所有团队成员一起担任这个角色。开发人员的关注点是通过分析代码和设计来编写单元测试，但是他们也会协助业务分析师或者测试人员编写功能测试，还会参与非功能测试和性能测试的分析。业务分析师和测试人员紧密协作，编写功能测试和用户验收测试，并执行探索性测试。客户/最终用户会被邀请参与用户验收测试。

测试脚本编写员

该角色就是编写详细测试脚本的人。这些脚本可能供手工执行，也可能被自动化。瀑布项目中的脚本编写员就是个初级员工，他根据测试计划和测试用例描述来编写手册，每一步都描述的很详尽。自动化脚本编写员就得是更资深的人了，开发人员也会参与单元测试用例的编写。敏捷项目会大量使用开发人员来编写测试脚本，主要是因为测试脚本是自动化执行的。

测试执行员

不管是手工测试还是自动化测试都有这个角色，不过在自动化的时候，这个角色的扮演者就是一台电脑。测试执行员会执行详细测试脚本，判断哪些测试失败，哪些测试通过。瀑布项目一般都用测试人员来做这件事情，而敏捷项目则鼓励所有人都来参与，尤其是测试人员、业务分析师和客户。

环境管理人员

这个角色管理测试环境，包括应用程序运行的环境以及支持自动化测试的基础架构。他们还会关注外部接口和用作测试的数据。这个角色在瀑布项目和敏捷项目中很相似。

问题管理人员

问题出现以后就要解决。这个角色可以帮助筛查问题，确保它们被正确地创建，有各种属性，如严重程度、优先级、组件等等。这个角色还要管理问题的生命周期，并提供工具支持。这个角色在瀑布项目和敏捷项目中很相似。

故障检测人员

这个角色当问题出现的时候就要去做故障检测工作，判断是不是软件缺陷。如果是软件缺

陷，他们就要去找出问题根源、可能的解决方案和变通措施。这个角色在瀑布项目和敏捷项目中很相似。

敏捷团队所注重的是让各个角色得到充分发挥，而比较少关心谁在做什么事情、谁对哪些事情负责。测试人员和其他团队成员之间没有界限，他们共同的目标是生产出更高质量的软件，每个成员都要尽一切可能帮助达成这个目标。在敏捷团队中，测试人员可以从所有人那里得到帮助，而他们又可以帮助其他人提高测试技能。这种方式能够确保团队中的每个人都在为交付高质量应用程序而付出。

13.9　参考文献

Test-Driven Development：By Example by Kent Beck（Addison-Wesley Professional，2002）

“Exploratory Testing Explained v.1.3 4/16/03，”copyright 2002–2003 by James Bach，http://www.James@satisfice.com