第 2 章

探索式测试设计概论

本章首先介绍探索式测试的思维模型，然后具体讲解测试先知、测试过程和测试覆盖等测试要素，最后以启发式测试策略模型总结全章内容。

2.1 思维模型

上一章介绍了探索式测试的定义。在实际项目的测试执行过程中，读者是否曾遇到如下的几个现象：

- 测试人员按照一个测试用例来执行测试，得到的程序输出与预期输出不一致。
- 测试人员判断程序的行为并不是缺陷，但根据新的输出想到了新的测试思路。
- 测试人员根据新的测试思路采用不同的输入并检查程序输出。
- 测试人员再次根据新的测试结果选择新的输入，反复地探索下去，最终发现了一个程序缺陷。
- 测试人员发现该缺陷的测试思路或测试用例并没有出现在最初的测试设计或测试用例文档中。

相信有很多读者熟悉上述的情景，也许有些人认为这是测试设计的遗漏，但笔者要告诉读者的是，千万不要怀疑你的测试设计能力，因为这是非常正常的现象。由于我们还没有真正深入地了解产品，不可能在测试设计的时候想到所有测试场景，且在需求分析阶段不可能评审到所有的隐含需求，所以最初的测试设计并不能捕获程序的所有缺陷。为了发现尽可能多的缺陷，测试人员需要在测试过程中，根据测试反馈持续地优化测试模型、调整测试设计。这是一个研究、实践和探索的过程。了解探索式测试的思维将有助于测试人员更有效地测试。

根据测试专家 Erik Petersen 对于探索式测试的理解[Erik11]，笔者抽象出探索式测试的思维模型 CPIE（Collation, Prioritization, Investigation, Experimentation），如图 2.1 所示。该测试模型包含迭代的 4 个阶段：整理、排序、调查和实验。

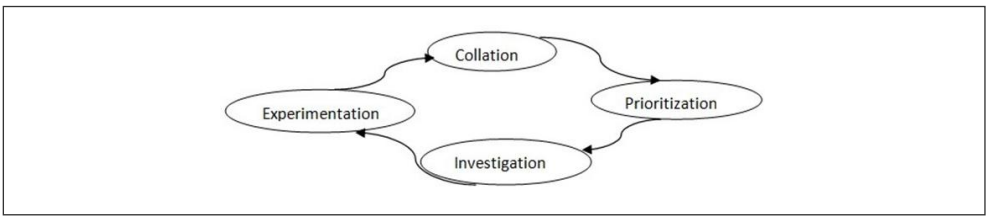

图 2.1　探索式测试的思维模型

- 整理（Collation）：尽最大可能收集关于被测产品的信息，去了解和理解它们。
- 排序（Prioritization）：确定所有测试任务的优先级。
- 调查（Investigation）：对即将执行的测试任务进行仔细的分析并确定测试输入和预期输出。
- 实验（Experimentation）：实际地去测试，验证我们的预测是否正确，检查我们在整理阶段获取到的信息是否正确。根据实验结果，测试人员将收集更多的信息，并调整测试任务的优先级。

对于探索式测试的思维过程，测试专家 James Bach 提出了如图 2.2 所示的思

维模型。该模型包含一组启发式问题，以推动测试人员在知识（Knowledge）、分析（Analysis）、实验（Experiment）和测试故事（Testing Story）上深入探究。

- 知识：掌握产品特性、开发技术、测试技术和领域规则等测试需要的知识。
- 分析：分析产品风险、测试覆盖、测试方法、测试先知[1]和产品缺陷等测试相关因素。
- 实验：配置、操作、观察和评估被测产品。
- 测试故事：用测试计划、测试报告和可工作的产品等组成测试报告，以准确地反映测试状态和产品质量。

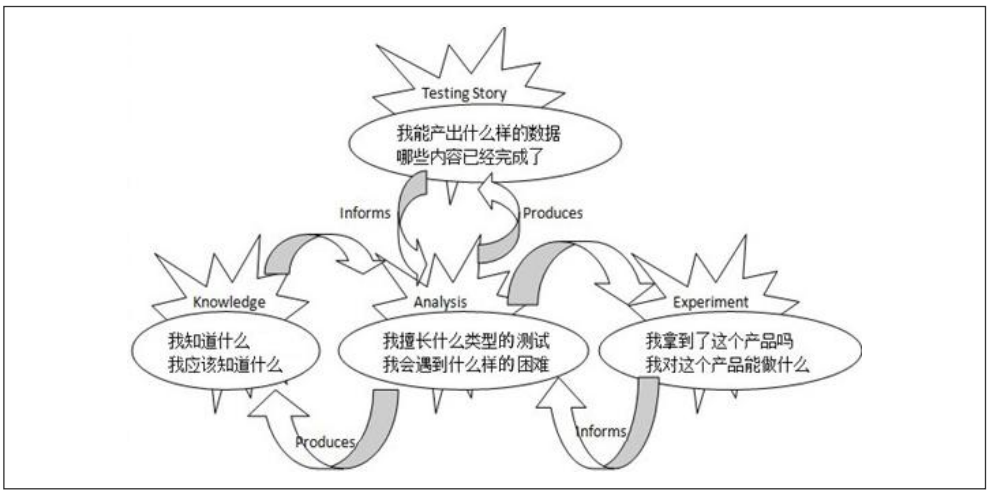

图 2.2　探索式测试的思维过程

从图 2.1 和图 2.2 可以看出 James Bach 和 Erik Petersen 的观点都强调通过实验（Experiment）来持续改进测试设计。他们认为瀑布式的测试设计和用例编写并不会产生优质的测试设计，测试人员还需要在测试执行的时候持续扩展新的测

1 测试先知（Test Oracles）是一组信息，用于确定测试过程中发现的问题是否为软件缺陷，稍后做详细讨论。

试思路，完善测试设计。在探索式测试过程中，测试学习、测试设计、测试执行和测试评估是互相支持和驱动的活动。第 4 章将讲述几个案例来说明探索式测试这种迭代优化的测试风格。

2.2 测试先知和启发式方法

测试人员拿到测试任务时，需要考察两类基本情况。第一类是测试人员的情况：

- 测试人员的测试经验怎么样，丰富还是欠缺？
- 测试人员对被测产品的行业经验怎么样，熟悉还是了解？
- 测试人员对被测产品的需求了解怎么样，熟悉还是了解？

第二类是被测产品的情况：

- 产品开发目前处于什么阶段？
- 产品是否经过了测试，使用了哪些类型的测试，覆盖了哪些功能和属性？
- 产品目前的风险或潜在问题有哪些？

测试人员应该仔细分析和理解这些情况。在时间压力和业务质量压力下，测试人员需要根据正确的信息来驱动测试活动，这样才会取得较好的效果。

首先测试人员需要非常清楚自己的情况，也就是自己所拥有的知识（Knowledge），包括产品知识、行业知识、测试技术、开发技术和计算机基础等。

检查这些知识是为了在测试时能够快速和有效地确定所发现的问题是否是差评缺陷。即测试人员需要综合各种知识以构造一组测试先知，从而高效地识别产品缺陷。一种构造测试先知的可行方法是参考 HICCUPPS（F）启发式规则[Bolton05]，它们通过一致性检查来识别产品缺陷。

- 历史（History）：目前所做的产品的版本是否与过去的版本相一致。
- 愿景（Image）：产品是否与开发组织的愿景相一致。
- 相似产品（Comparable Products）：产品是否与类似的产品相一致。
- 声明（Claims）：产品是否与重要人物的声明相一致。
- 用户期望（User's Expectations）：产品是否与用户期望相一致。
- 产品自身（Product）：产品中可比较的各个功能是否一致。
- 目的（Purpose）：产品是否与其（明确的或隐含的）目的相一致。
- 法规（Statutes）：产品是否与适用的法律相一致。
- 常识（Familiarity）：产品是否与常识相一致。

尽管我们有很多方式去丰富测试先知，但不支付足够的时间成本很难达到非常完美的程度（所有预期输出都在测试之前明确下来是需要很多时间成本投入的），因此我们在探索式测试过程中往往会遇到以下问题：

- 没有测试先知可以使测试人员提前确定所观察到的系统行为必定是正确或错误的。
- 没有一个单独的测试先知可以说明某个功能在任何时间或任何情况下都是正确工作的。
- 有些功能看上去是正常工作的，但在某些情况下会失败，且会影响其他测试先知的正确性和适用性。

可见，测试人员构造测试先知时，肯定会遇到很多困难，那么该如何解决呢？以下是三个可能的思路。

- 忽略这个问题（也许这个信息的价值从成本角度考虑不值得）。
- 简单化这个问题（改善可测试性以获得更多的信息、分析需求、规约和代码，从而获得更简单的检查规则）。
- 转移这个问题（考虑问题的相关性，从类似问题下手）。

在测试设计过程中，测试人员还可以使用启发式方法（Heuristics）来产生更

多更好的测试思路，例如[Bach11]：

- 引导词启发法（Guideword Heuristics）：一些词语或标签能引导测试人员发掘自身的知识和经验，产生新的测试思路。。
- 触发器启发法（Trigger Heuristics）：一些存在于事件或条件中的想法，能提醒测试人员采用另外一种方式来进行试验，就像思维的闹钟一样。
- 副标题启发法（Subtitle Heuristics）：能帮助测试人员重构测试想法并获得更多的选择点。
- 启发式模型（Heuristics Model）：一组系统性的想法能帮助测试人员控制、管理和挖掘更多的想法。

实际上，人们在日常生活中也经常使用启发式方法，例如，我们常用如下简单的规则处理复杂的现实问题：

- 酒后驾驶非常危险。
- 一鸟在手要强于二鸟在林。
- 不入虎穴，焉得虎子。
- 人们有时在计算机旁边隐藏自己的密码，尝试从那里寻找。
- 假期商店一般较晚开门。
- 如果你的计算机出现了莫名其妙的行为，重启。如果还是有问题，重装操作系统。
- 如果这是个真正重要的任务，你的老板会跟踪它，否则，你可以忽略它。

由这些例子可知，启发式方法是一种经验方法，它针对复杂的问题提出了一种简单的、较可能成功的解决思路。使用启发式方法，人们可以快速地采取行动，在实践中去探索答案，从而避免陷于无止境的问题分析，毫无进展。然而，启发式方法只是一种“捷径”，它不保证提供了“最佳答案”，也不能确保总是提供“正确答案”。因此，明智的测试人员不会依赖特定的、单一的启发式方法，他会综合运用多个启发式方法，并根据实践结果调整测试方法和测试先知。

2.3 测试过程

测试专家 James Bach 曾经指出测试是测试人员尝试去操作被测系统并且查看被测系统是否正常工作的过程，图 2.3 简述了测试过程需要考虑的因素。

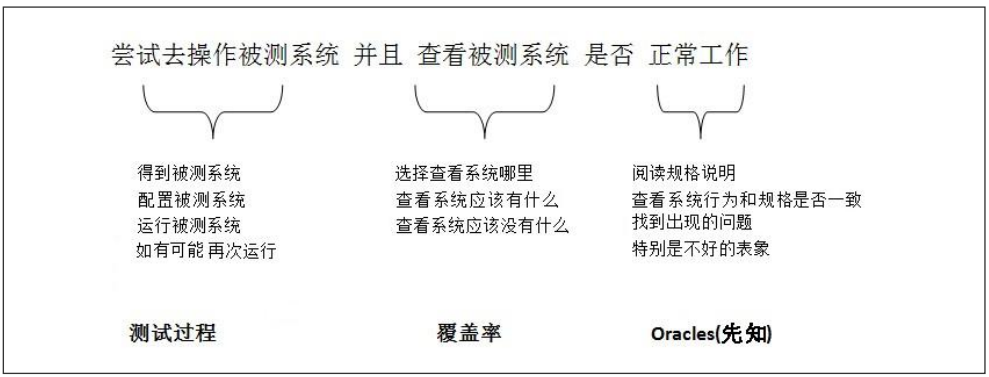

图 2.3 测试过程需要考虑的因素

James Bach 还曾提到过，探索式测试就像对一个人进行面试。测试人员需要向面试对象提出高质量的问题（一个问题类似于一个测试想法），并根据面试对象的回答提出更好的问题。为了提高发问的水平，测试人员需要着重培养如下的能力和技巧：

- 提出有用的问题（目的驱动问题）。
- 观察什么事情正在发生。
- 描述自己能够感觉到或看到的东西。
- 对于自己的所知进行批判性的思考。
- 组织和管理业务上的规则。
- 能够提出假设和进行试验验证。
- 尽管已经了解，但仍然进行深入思考。
- 分析其他人的思考方式。

- 根据因果关系进行推导。

不妨将测试团队的工作比喻成足球团队的传球配合，测试人员拿到了一个测试任务，就类似于足球传到他的脚下，他需要针对该测试任务问自己一些问题：

- 我是否能够接住这个传球？
- 我能否确认已经接到了这个球？
- 我有几种方式处理这个球吗？
- 我所承担的角色和任务是什么吗？
- 我的装备是否已经准备好了？
- 我的队友在哪里？
- 当前球场上的形势如何？
- 能否让队友来帮助我？
- 我是否意识到情况的严重性？
- 我是否做好了准备去行动，现在？

一旦测试人员对自己的知识和自己的技能有所了解，接下来就是确定自己的任务，并根据以下因素对产品、需求和模块进行测试分析：

- 产品和项目风险。
- 测试覆盖。
- 测试先知。
- 资源和限制。
- 价值和成本。
- 已存在的缺陷。

在初始分析结束后，测试人员可以开始探索式测试，通过“提问↔回答”的循环不断地对产品进行试验。通常，一次试验包含配置、操作、观察和评估四个步骤。

那么测试人员是怎么来进行试验的呢？具体包括如下几个步骤。

（1）配置：安装产品，准备测试数据和工具，驱动产品进入特定的状态以待测试，根据自己的测试任务准备测试先知和启发式问题。

（2）操作：通过试探性的输入向产品提问。

（3）观察：收集关于这个产品是如何工作（利用正确和错误的输入）的信息来评价产品是否正常工作。

（4）评估：应用测试先知发现和确认产品缺陷。

在实验过程中，测试人员可以利用如下思路实施探索。

- 充分利用自己现有的资源，包括用户、文档信息、开发人员关系、同事、设备和工具和计划等。
- 综合使用多种测试技术，包括功能测试、域测试、压力测试、场景测试、用户测试、风险测试、声明测试（针对重要客户的声明内容进行测试）和自动化测试等。
- 利用自己实际能够做的东西，包括自己擅长的技术、最容易暴露问题的数据问题和可利用的硬件资源等。

  - 尽可能发散地（De-Focus）测试：
  - 从一个不同的产品状态开始测试。
  - 优先执行复杂且有约束条件的操作。
  - 从系统模型的变化中产生测试思路。
  - 质疑我们的试验流程和工具。
  - 试着去观察每个产品细节，从中获取测试灵感。

第 1 章提到了探索式测试的定义：测试相关学习、测试设计、测试执行和测试评估是并行实施的。所以，测试执行的过程也是学习的过程，即通过测试来学

习（Testing to Learn）。其中，测试学习通常包含如下几个步骤。

（1）形成一个关于产品功能的模型。

（2）了解这个产品旨在完成什么任务。

（3）列出需要测试的产品元素。

（4）尝试构建多个测试先知，例如参考 HICCUPPS（F）启发式规则开发一批针对当前产品的测试先知。

（5）产生测试数据。

（6）考虑被测产品的可测试性和尝试不同的测试工具。

（7）尝试用多种不同的方法去试验。

（8）报告所发现的产品缺陷。

在测试与学习的过程中存在分支是正常的。测试人员需要不停地产生新的分支（测试思路），而不是一条直线走到底。因为通常难以预料走另外一个分支后会发生什么事情（也许会发现重要的新信息或隐藏的缺陷）。但凡事都有个度，测试人员需要定期去检查现在测试的模块是否与分配到的测试任务相一致，防止将过多的时间花在非测试任务上，否则就得不偿失了。

在试验的过程中，测试人员可能遇到令人困惑的问题。这时，他可以考虑使用以下方法来解除疑惑。

（1）简化自己的测试思路。

（2）保留测试现场并寻求帮助。

（3）不断重复执行自己的操作。

（4）返回到一个已确认过的状态。

（5）一次只考虑一个因素，一次只修改一个因素。

（6）做出精确的观察。

在探索式测试过程中，测试人员的下一个测试思路往往是不可预知的。在很大程度上，下一个思路取决于本次测试用例的执行情况。有时，测试人员需要自问：是沿着当前思路继续挖掘，还是更换思路另辟蹊径？这时，可以参考“陷入与退出”（Plunge in and Quit）策略[Kaner01]：面对复杂的产品，先深入研究 30 分钟或 1 小时，然后停下来做些别的工作。经过几轮的陷入与退出，就会开始理解产品的模式和轮廓，并在头脑中形成更系统、更具体的测试和研究策略。

2.4 测试覆盖

本节将详细介绍测试覆盖。测试覆盖是测试的范围，也就是需要测试什么，不测试什么，以及从什么角度来进行测试。例如，在第三方软件兼容性测试[2]中，测试人员可以采用如下策略覆盖产品的功能：

在时间允许的情况下合理地测试所有的主要功能。让测试经理知道有哪些贡献性的功能[3]没有时间测试。测试人员也可以对一个或多个有趣的贡献性的功能进行测试，并在探索和测试主要功能时接触更多的贡献性的功能。

2 常见的第三方软件兼容性测试是反病毒软件兼容性测试。测试人员会安装主流的反病毒软件，然后检查产品在此环境中能否正常工作。

3 贡献性的功能是产品中非主要的功能，是锦上添花的功能。

测试人员选择一些潜在的不稳定因素进行测试。一般情况下，选择 5～10 个这样的功能。测试经理会决定这些功能在功能性和稳定性上的测试需要多长时间，一般花 80%的时间在主要功能上，花 10%的时间在贡献性的功能上，花 10%的时间在不稳定的方面。

具体而言，测试人员可以从六个方面考虑测试覆盖[Bach11]。

第一，结构（Structure）覆盖。测试人员要分析产品是如何构成的，然后用测试去覆盖产品的重要组成部分。例如，在测试打印功能时，测试人员可考虑覆盖如下内容：

- 打印需要用到的文件。
- 实现打印功能的代码模块。
- 在这个模块里面的代码语句。
- 在这个模块里面的代码分支。

可见，测试人员关注的是产品的内部结构。

第二，功能（Function）覆盖。测试人员要分析产品包含哪些功能，然后用测试去覆盖功能点。同样以打印测试为例，看看功能覆盖要考虑什么：

- 页面设置，打印预览。
- 双面打印，打印份数，缩放。
- 打印所有页，打印当前页，打印指定的范围。

可见，测试人员关注的是产品的功能或特性。

第三，数据（Data）覆盖。测试人员要分析产品会处理什么数据、传输什么数据、输出什么数据，并用测试覆盖这些内容。同样以打印测试为例，看看数据覆盖要考虑什么：

- 打印文档的类型。
- 文档里面的元素，文档的大小，文档的结构。
- 打印的控制参数，如缩放因子、打印份数等。

可见，测试人员关注的是产品使用过程中的数据处理。

第四，平台（Platform）覆盖。测试人员要分析产品依赖于什么平台及平台特性，并用测试覆盖这些平台和特性。同样以打印测试为例，看看平台覆盖要考虑什么：

- 不同类型的打印机、缓冲池、网络连接。
- 不同厂家的计算机。
- 不同厂家的操作系统。
- 不同厂家的打印机驱动程序。

可见，测试人员关注的是产品所依赖的环境和第三方软硬件产品。

第五，操作（Operation）覆盖。测试人员要分析用户是如何使用产品的，并用测试覆盖使用流程和操作。同样以打印测试为例，看看操作覆盖要考虑什么：

- 默认配置下使用。
- 真实环境下使用。
- 真实的场景下使用。
- 复杂的流程下使用。

可见，测试人员关注的是产品的使用场景，考察稳定性、可用性、安全性、可扩展性、性能、可安装性、兼容性、可测性、可维护性和本地化等因素。

第六，时间（Time）覆盖。测试人员要分析产品受什么时间因素的影响，并用测试去覆盖有风险的因素。同样以打印测试为例，看看时间覆盖要考虑什么：

- 尝试在不同的网络或端口速度下打印。

- 一个文档打印完，紧接着打印另一个文档，或隔很长时间再打印。
- 尝试与时间相关的限制，如使用打印缓冲区、触发超时等。
- 尝试整点、凌晨（天与天的交界）、月底或年底打印文档。
- 尝试从不同的两个工作站同时打印。

可见，测试人员关注的是产品使用是否受时间影响。

在真实项目中，测试人员需要对于某一个类型的覆盖做全面的分析，而且还需要组合使用所有的覆盖类型。关于组合策略，根据实际项目的具体情境会有所不同，本节不做详述。有时候，测试人员确实很难覆盖到一些特殊的部分（如特定的异常处理代码），这时他需要考虑改进产品的可测试性。

可测试性主要包括如下几个方面。

- 可操作性：易于测试执行，允许并行的开发和测试。
- 可控制性：可通过灵活的方式控制产品的行为，如脚本化的接口。
- 可观察性：有明确的可观察的输出，如日志文件。
- 简单性：从用户的角度看功能简易，从开发团队的角度看代码简洁、结构简单。
- 稳定性：功能较稳定。
- 易理解性：便于判断缺陷，具有良好的技术文档。

可测试性越好的项目，测试的效率越高，测试效果会越好。

2.5 启发式测试策略模型 HTSM

本章已经介绍了测试先知、测试过程、测试覆盖等内容，而测试人员还需要一个模型将这些内容系统地组织起来。为此，本节介绍由 James Bach 提出的启发式测试策略模型（Heuristic Test Strategy Model，简称 HTSM）[Bach11]。

2.5.1 为什么需要 HTSM

根据产品的风险设计测试是一种常见的测试设计思路。在复杂的现实世界，产品面临的风险多种多样，只有全面考虑、周密测试才能避免风险暴露导致的严重后果。因此，测试人员需要一个相对完整、可以定制、容易扩展的风险列表或参考模型，来帮助他们发现产品风险。HTSM 就是一个结构化的、可定制的参考模型，它由一组层次化的指导词（Guide Word）组成，从测试技术、产品元素、项目过程和质量标准等多个角度启发测试设计。

HTSM 能够在测试全程提供有益的提示。在制定测试计划初稿时，它可以帮助测试人员完整地思考产品的方方面面，从而产生系统性的测试计划。在测试过程中，它可以帮助测试人员组合测试想法、深入探索产品，以开发出强有力的测试策略。在回归测试中，它可以帮助测试人员确定测试范围，制定测试方案。

2.5.2 HTSM 的内容

图 2.4 是 HTSM 的概要描述，测试人员利用质量标准（Quality Criteria）、项目环境（Project Environment）、产品元素（Product Elements）、指导测试技术（Test Techniques）的选择与应用，产生观察到的质量（Perceived Quality）。

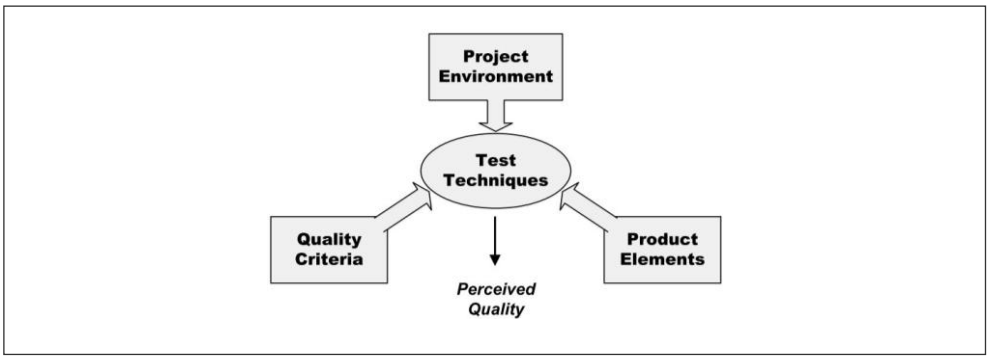

图 2.4 HTSM 测试策略模型

HTSM 具有层次结构，其顶层元素（质量标准、项目环境、产品元素、测试技术）可以分解为次层元素，而次层元素可进一步分解为第三层元素。本文只概要介绍次层元素，更多的细节请参考 James Bach 的文档[4]。

测试技术：生成测试的策略。有效地选择和实施测试技术，需要综合分析项目环境、产品元素和质量标准。可选的测试技术如下。

- 功能测试（Function Testing）。
- 域测试（Domain Testing）。
- 压力测试（Stress Testing）。
- 流测试（Flow Testing）。
- 场景测试（Scenario Testing）。
- 声明测试（Claims Testing）。
- 用户测试（User Testing）。
- 风险测试（Risk Testing）。
- 自动测试（Automatic Testing）。

项目环境：资源、约束和其他影响测试的项目元素。测试总是受到项目环境的约束。在某个团队运转良好的策略不一定适合另一个相似的团队，以往富有成效的方法未必适应当前的项目。有经验的测试人员会根据当前语境（Context），在约束条件下充分运用资源，以高效地测试。项目环境包括如下方面。

- 用户（Customers）：理解产品的用户。
- 信息（Information）：发现测试所需的信息。
- 开发者关系（Developer Relations）：与开发者协作加速开发。
- 测试团队（Test Team）：利用团队的力量支持测试。

4 http://www.satisfice.com/rst-appendices.pdf

- 设备与工具（Equipment & Tools）：可利用的硬件、软件和文档等。
- 进度（Schedule）：项目实施的流程。
- 测试条目（Test Items）：测试范围和重点。
- 交付品（Deliverables）：测试的产出。

产品元素：需要测试的对象，包括如下方面。

- 结构（Structure）：产品的物理（physical）元素，如代码、接口、配置文件、可执行文件等。
- 功能（Functions）：产品的功能。
- 数据（Data）：产品所操作的数据。
- 平台（Platform）：产品所依赖的外部元素。
- 操作（Operations）：产品将被如何使用。
- 时间（Time）：影响产品的时间因素。

质量标准之操作性标准（Operational Criteria）：面向用户和运营团队，包括如下方面。

- 能力（Capability）。
- 可靠性（Reliability）。
- 可用性（Usability）。
- 安全性（Security）。
- 可伸缩性（Scalability）。
- 性能（Performance）。
- 可安装性（Installability）。
- 兼容性（Compatibility）。

质量标准之开发标准（Development Criteria）：面向开发团队。

- 可支持性（Supportability）。

- 可测试性（Testability）。
- 可维护性（Maintainability）。
- 可移植性（Portability）。
- 本地化能力（Localizability）。

由以上介绍可知，HTSM 由一组指导性词语组成，它们构成一个层次结构，让测试人员从高层抽象到底层细节对产品和测试进行思考。这些指导性词汇是测试的指南，其作用不是教导如何具体测试，而是启发测试人员的思维，发掘测试对象和测试策略。

图 2.5 摘录自 James Bach 的测试培训教材 [Bach11]，体现了 HTSM 对于测试设计的意义。

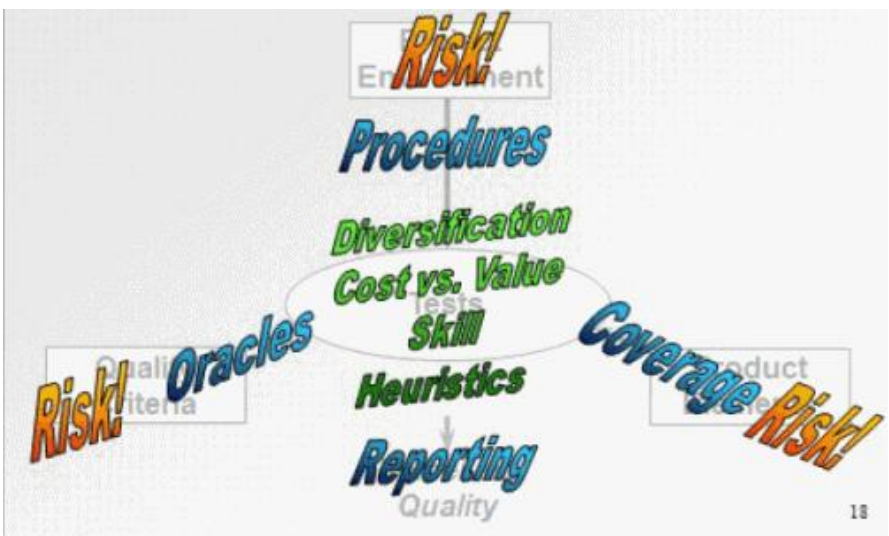

图 2.5　HTSM 中的测试先知、测试过程、测试覆盖和测试报告

- 测试设计以风险驱动。测试人员分析质量标准、项目环境和产品元素中的风险，设计有针对性的测试策略。

- 在测试设计时，质量标准启发测试先知（Oracles），项目环境启发测试过程（Procedures），产品元素启发测试覆盖（Coverage），观察到的质量启发测试报告（Reporting）。
- 对于测试，HTSM 强调测试策略的多样性（Diversification），平衡代价和收益（Cost vs. Value），利用启发式方法（Heuristics）充分发挥测试人员的技能（Skill）。

2.5.3 定制 HTSM

在定制化之前，HTSM 对测试人员的帮助很小，因为此时的 HTSM 是“James Bach 的模型”，而不是符合当前语境的模型。HTSM 是通用的模型，虽然能够普遍使用，但是不能快速、高效地指导具体的测试工作。测试人员需要将其“本地化”，才能发挥其威力。

Cem Kaner 教授提出利用思维导图（Mind Map）定制 HTSM [Kaner11]。如图 2.6 所示，他将 HTSM 作为图的中心，将 Quality Criteria、Project Environment、Product Elements 和 Test Techniques 作为主干。

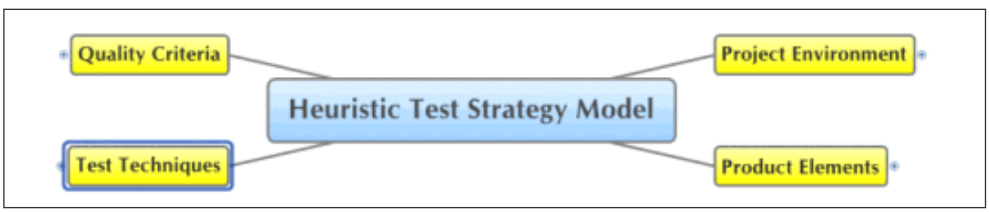

图 2.6 思维导图形式的 HTSM

在分支上，Cem Kaner 添加了他觉得重要的节点。例如，在图 2.7 中，他在 Product Elements 下增加了 Benefits 节点和 Time 节点，使 HTSM 符合他的工作需要。

恰如 Cem Kaner 所说“大多数严肃对待此模型的人会定制它以符合自己的需

要”（Most people who work seriously with this model customize it to meet their needs），测试人员可以也应该修改 HTSM，以获得符合项目语境的模型。

- 增加节点：增加与当前项目相关的测试技术、测试想法、测试对象和任何测试人员认为有价值的元素。
- 删减节点：忽略一些与项目或任务无关的元素。
- 增加标记、注释、链接等图元：标记可以突显重要的元素，注释可以增加更多的细节，链接可以指向更详细的信息源。

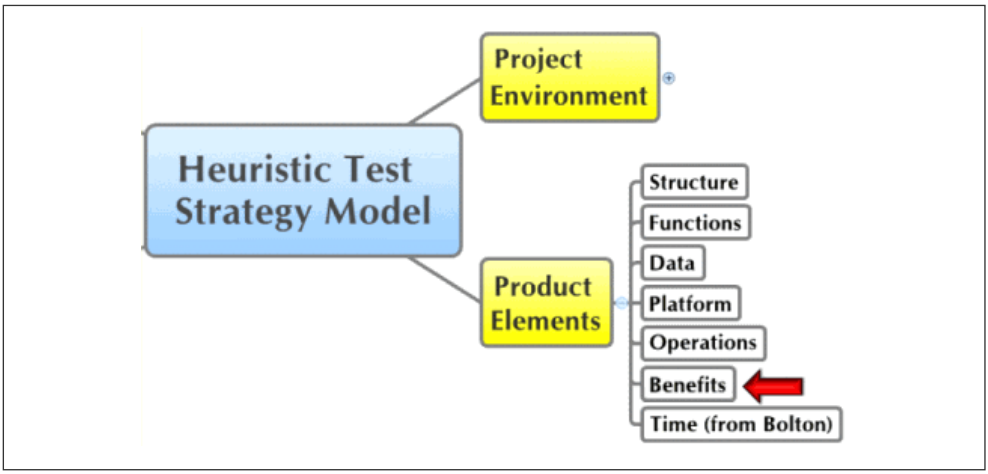

图 2.7　通过增加节点来定制 HTSM

定制 HTSM 是理解并掌握 HTSM 的过程。与大多数方法一样，测试人员需要修改它，加入自己的风格和元素，才能真正掌握它。

测试专家 Michael Larsen 在 XMind.net 提供了他制作的 HTSM 思维导图[5]，为测试人员制作自己的 HTSM 提供了很好的基础。

5　http://www.xmind.net/share/mkltesthead/

2.5.4 应用 HTSM

定制 HTSM 就是应用 HTSM 的过程。测试人员遵循 HTSM 的结构化指南，深入地思考产品、项目与测试，添加自己的想法、评论、标记和启发式问题。这本身就是极好的测试学习过程。作为学习的结果，定制化的 HTSM 为进一步进行测试设计提供了坚实的基础。在测试过程中，测试人员会接触新信息，学习新知识。他应该持续地将新知补充到 HTSM 中，以迭代地优化测试模型。从这个角度，HTSM 既是测试想法的源头，也是测试过程的产出。

在测试设计时，测试人员可以逐个检查 HTSM 中的每个元素（指导性词语），阅读相关标记、注释和链接，以启发测试思路。他可以自问：

- 该元素与当前测试任务相关吗？
- 针对该元素，产品有什么风险？可能会有什么缺陷？
- 通过什么测试可以发现这些缺陷？
- 依据当前的进度和资源，如何实施这些测试？

另一种更有威力的方法是综合 HTSM 中的多个元素，开发测试策略。当开发人员用单元测试检查了组件，测试人员需要在系统层面检查产品。此时，产品的缺陷往往存在于组件的交互和复杂的流程。综合产品的多个方面，开发多样化的测试，以更深入地测试产品，才能够更好地体现测试人员的价值。一些有帮助的启发式问题包括：

- 该元素与哪些元素相关？
- 元素的组合有没有揭示出新的风险？
- 如何设计测试，以同时测试这些元素？
- 能否让来自元素 A 的信息帮助元素 B 的测试？