火龙果•整理
uml.org.cn

第 1 章

探索式测试的定义

本章给出探索式测试的定义，然后介绍语境驱动测试的 7 条原则，最后回答一些有关探索式测试的常见问题。

1.1 什么是探索式测试

探索式测试（Exploratory Testing）是一种自由的软件测试风格，强调测试人员同时展开测试学习、测试设计、测试执行和测试结果评估等活动，以持续优化测试工作。考虑到它所具备的即兴发挥、快速实验、动态调整等特征，其思维方法可以追溯到软件开发的最初岁月。

作为一个技术术语，“探索式测试”是测试专家 Cem Kaner 博士在 1983 年提出的，并受到了语境驱动测试学派（Context Driven Testing School[1]）的支持。Cem Kaner、James Bach 和 Bret Pettichord 合著的《软件测试经验与教训》[Kaner01]对语境驱动测试和探索式测试做了精要且深刻的论述。测试专家 James A. Whittaker 曾是 Cem Kaner 在佛罗里达工学院（Florida Institute of Technology）的同事，后来

担任过微软测试架构师和 Google 测试总监。基于在微软的工作经历，他撰写了《探索式软件测试》一书[2]，进一步扩展了探索式测试的测试方法。

探索式测试有丰富的内涵，Cem Kaner 用如下文字定义了探索式测试的核心。

探索式测试是一种软件测试风格（Style），它强调独立测试人员（Individual Tester）的个人自由和职责（Personal Freedom and Responsibility），为了持续优化其工作的价值（Value），将测试相关学习（Test-related Learning）、测试设计、测试执行和测试结果分析作为相互支持的活动，在整个项目过程中并行地执行[Kaner08]。

不妨将这段定义分成三个部分进行讨论。

首先，探索式测试是一种软件测试风格（Style），而不是一种具体的软件测试技术（如等价类划分、边界值分析等）。作为一种思维方法（Approach），探索式测试强调依据当前语境（Context）选择合适的测试技术（Technique），而不局限于特定的测试技术。测试人员可以在探索式测试中使用任何一种测试技术，也可以将探索式测试应用于任何测试阶段[Kaner09]。

在这种测试风格的指导下，涌现出了一批支持探索式测试的测试技术。例如，James A. Whittaker 在《探索式软件测试》中提出了一套基于系统化错误猜测和测试隐喻的“漫游测试”技术（该测试设计方法将在第 3 章和第 4 章中介绍），丰富了探索式测试的手段。又例如，Jonathan Bach 和 James Bach 发明了基于测程的测试管理（Session-Based Test Management）[3]，显著地提高了探索式测试在测试组织、汇报、交流和度量上的能力（该测试管理方法将在第 8 章中介绍）。再例如，开发工具 Microsoft Visual Studio 2010 开始支持手工测试和探索式缺陷（Exploratory Bug）[4]，虽然相关功能略显单薄，但是它体现了软件行业对探索式测试的认可，

也表明探索式测试辅助工具和自动化将获得更多的重视与发展（第 6 章将介绍 Visual Studio 2010 的相关功能）。

因此，笔者认为术语“探索式测试”[5]有两层含义。

第一，其内涵是一种软件测试风格和思考方法，不拘泥于具体的测试技术；第二，其外延是一批在这种思考方法指导下发展出来的测试技术，包括测试设计方法、测试管理方法、测试辅助工具等。为了避免混淆，本书使用“探索式测试”指代探索式测试风格和思考方法，使用特定的测试方法名指代具体的测试技术。

第二，探索式测试强调独立测试人员的自由和责任。测试人员应该为个人和团队负责，调动所有能量，发挥人的灵活性，在整体上持续优化个人和团队的产出。

测试人员是软件企业的知识工人（Knowledge Worker）。管理大师 Peter Drucker 认为知识工人必须管理自己。一方面，他建议知识工人自己确立工作目标（根据项目情况去做当前最有价值的工作），通过持续创新、持续学习、持续交流来优化其生产效率和产出质量。另一方面，他建议企业信任员工，给予充分的授权，并将他们视为企业的资产加以持续投资[Drucker99]。这两方面可以视为探索式测试对于员工与企业的潜在要求。

第三，探索式测试建议在整个项目过程中，将测试相关学习、测试设计、测试执行和测试结果分析作为相互支持的活动，并行地执行。实际上，人脑难以并行地执行多项任务。探索式测试旨在将测试学习、测试设计、测试执行和测试结果分析作为一个循环快速地迭代，以不断收集反馈、调整测试、优化价值。

5 Exploratory Testing 通常被翻译为“探索式测试”或“探索性测试”。本书的作者们选择“探索式测试”是因为这个术语强调了 Exploratory Testing 是一种测试风格和思考方式，符合该术语的核心思想。此外，“探索式测试”在字面上与现有的测试术语（如功能性测试、安全性测试、兼容性测试等）有更大的差异，表明它是一种可以应用于任何测试活动的测试方式。

- 测试学习：学习任何可以指导测试的知识，可能要学习的内容包括行业背景、领域知识、技术平台、测试技术、产品缺陷、项目风险等。
- 测试设计：安排测试计划，拟定测试策略，开发测试想法，制作测试支持材料。
- 测试执行：执行测试并收集结果。测试可以手工执行，也可以自动执行。
- 测试结果分析：分析并解读从测试中学到的知识，可能的活动包括判定测试是否通过、理解产品实现、发掘风险区域、评估测试方法是否有效等。

在现实的软件项目中，穷举测试是不可行的。任何测试都是采样测试，都存在投入测试资源却不能发现缺陷的风险。随着项目的发展，测试的风险也在持续变化。对此，探索式测试人员会在项目过程中，随时收集并判读测试情报，优化测试决策和设计，并将它们立即应用于测试执行，通过分析测试结果来评估开发状态和测试风险。这样的循环有助于最大化测试价值，并降低软件项目的风险。

1.2 语境驱动测试 7 原则

探索式测试的奠基人和积极实践者 Cem Kaner 和 James Bach 都支持语境驱动测试[Kaner12]。语境驱动测试的 7 条基本原则对于正确理解并应用探索式测试具有重要意义，本节将予以简单讨论。

原则 1：任何实践的价值都取决于其语境（Context）。

这条原则几乎是不言自明的。中国人很早之前就有相似的认识，“南橘北枳”[6]指相同的种子在不同的环境中会结出不同的果实。因此古人建议“因地制宜”[7]，即根据当地的具体情况，采用合适的措施。

然而，软件开发者往往会在无意中忘记这条原则。开发团队会照搬以往的经

6 语出《晏子春秋》，其成书于战国，后经西汉刘向整理。
7 语出《吴越春秋》，成书于东汉。

验，却不考虑经验可能已经过时；会不假思索地采用他人建议的开发方法，却不怀疑南橘北枳的可能；会按照高层的指示亦步亦趋，却不思索指令是否合理。更糟糕的是，在感觉到情况不妙后，却将错就错，不思变更。因此，开发团队需要频繁地反思其开发实践是否符合当前的语境，并做出相应调整。

原则 2：在特定语境下存在好的实践，但不存在最佳实践。

这条原则看似有些武断，毕竟软件研发已经沉淀出一批公认的实践方法，它们是现代软件开发必不可少的核心实践。但是，细细一想便会发现这些方法也需要因地制宜。“持续集成”是公认的最佳实践，但是不同的团队往往有不同的集成频率。对于小型项目，一次签入（Check-in）会触发一次完整的构建；对于大型项目，开发团队可能每天做一次完整构建；对于超大型项目，做一次完整的构建可能需要几天甚至更长的时间。不同的构建频率和构建代价自然会导致不同的签入策略和测试方法。虽然都在实施“持续集成”，但是不同的团队会设计出不同的流程和方法。

对于测试工作者，这条原则表示任何一种测试方法（包括探索式测试）都不是无条件的最佳选择。测试人员始终要评估当前情况，寻找适合当前语境的测试风格和技术。

原则 3：人，在一起工作的人，是项目语境中最重要的部分。

这条原则强调了软件开发的社会学因素。软件开发专家 Tom DeMarco 和 Tim Lister 指出：“本质上，我们工作中的主要问题，与其说是技术问题，不如说是社会学问题”[DeMacro99]。而社会学因素的根源是“软件开发是一个创造与沟通的协作游戏（Game[8]）”[Cockburn07]。在创造与沟通的过程中，一定是个体和他们的交互（Individuals and Interactions）起主要作用。

8 Game 也可以翻译为“博弈”。本书将其翻译为“游戏”是为了突出软件开发的乐趣、协作、沟通与互助。软件开发团队的对手不是团队成员，而是亟待解决的问题。

在软件测试中，以下观点反映了人的重要性。

- 软件开发是具有挑战性的智力活动，开发人员（包括测试人员）的责任感、技能、状态将强烈影响软件实现和代码质量。因此，招聘、培训、挽留高水平开发人员是软件企业最重要的工作之一。
- 软件开发是一种创造与沟通的游戏。软件企业应该营造一种开放、协作的工作环境，使得开发人员能够自如地去思考、去发明、去创造。
- 软件测试的核心任务是寻找并传递信息。在寻找信息的过程中，测试人员的能力和相互协作的水平将在很大程度上决定信息的数量和质量。
- 软件测试提供信息服务。服务就意味着有客户，测试人员是否成功，主要取决于是否很好地满足了客户的要求和最佳利益[Kaner01]。也就是说，测试人员的重要任务是理解客户，并与他们展开有效的交流。

以上观点只涉及该庞大主题的一小部分。笔者建议读者去阅读此领域的经典书籍（请参考本书附录中的“推荐阅读”）以获得更多见解。

原则 4：项目的发展往往难以预料。

在一些语境中，项目的发展是可以预料的。图 1.1 摘自 Steve McConnell 所著的《软件估算》，它显示了（对项目范围、代价、功能的）估算值是如何随着项目的进展变得准确的[McConnell06]。随着时间的推移，项目的不确定性逐渐降低，当项目即将结束时，开发团队能够准确预期项目的结局。但是，图 1.1 也提示了在项目的开始阶段，项目的不确定性非常高。在初始概念（Initial Concept）阶段建立的估算值可能是实际值的 4 倍。

该原则并不是一个悲观的见解，相反它体现了一种实事求是的态度和对软件风险的成熟认知。探索式测试有助于快速获得信息，从而降低软件项目的不确定性。成功的测试团队在整个项目过程中会结合广度优先探索和深度优先探索，在特定的时间选择适合的方法，从而更明智地利用测试资源。

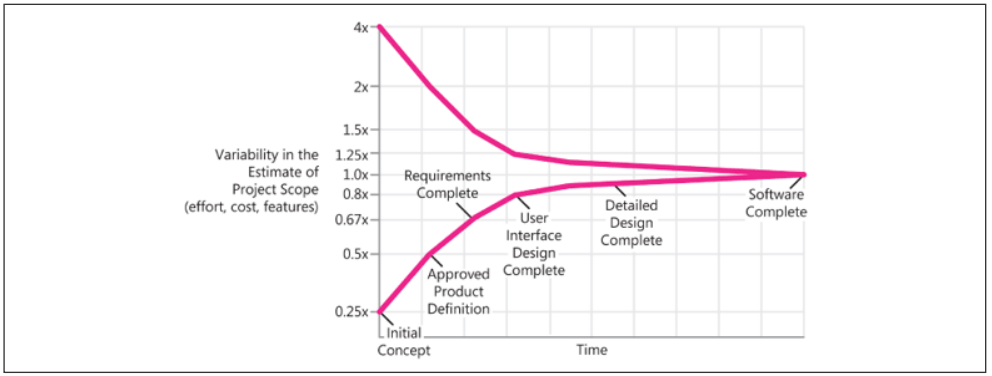

图 1.1　根据日历时间得到的不确定性锥（Cone of Uncertainty）

原则 5：产品是一种解决方案。如果问题没有被解决，它就是无用的（Doesn't Work）。

成功的软件必须帮助用户解决现实世界中的问题，辅助他们获得成功。在极端情况下，一个符合规格说明且没有技术缺陷的软件会遭遇失败，因为它没有解决用户的问题，甚至阻碍用户解决问题。图 1.1 显示，在需求完成（Requirements Complete）时，不确定的范围会大幅缩小。但是，如果需求存在重大缺陷，甚至初始概念就是错误的，那么稳定的开发过程只会“稳定地”产生失败的产品。

这一原则要求测试工作者用软件用户的视角考察整个产品，从显式规格说明（不完整、模糊、包含错误的项目文档）和隐式规格说明（包括竞争对手产品、相关产品、已发布版本、电子邮件讨论、口头讨论、论坛反馈、博客文章、领域专著、测试经验等）中，挖掘、推导、发现需求。这是探索式测试人员需要掌握的探索技能。

原则 6：好的软件测试是一个具有挑战性的智力过程。

这又是一条不言自明的原则，相信本书的读者也会认同该原则。虽然您可能会质疑本书的一些观点，但是您的阅读已经说明软件测试的挑战性和复杂性需要认真研究与思考。

这条原则的推论是枯燥、机械、无成就感的测试过程不是好的软件测试。这样的过程压抑了测试人员的创造性，分散了他们的注意力，降低了他们的智力水平。然而，本书的作者们也必须承认，我们在日常工作中也时常会感到枯燥、单调、缺乏创造力。对此，笔者的基本观点是：第一，测试人员应该以合乎要求的质量完成自己的工作，这是测试人员必须履行的职责；第二，应该将单调的工作视为改进的信号，通过改变工作内容和流程，将更多的时间用于具有挑战性的工作。探索式测试、技术创新和测试自动化是笔者的常用工具。

原则 7：只有通过判断和技能，并在整个项目过程中协同练习（Exercise Cooperatively）它们，我们才能在正确的时间做正确的事，以有效地测试我们的产品。

软件测试领域的一些文献似乎在暗示，只要严格遵循“优秀”的测试过程或模板，就可以使普通的测试人员成为测试专家，并获得理想的测试结果。这种观点是不正确的，因为只有高素质的测试人员才能根据语境设计出合适的流程、计划、策略和用例，而这些才是有效测试的基础。

那么该如何培养高素质的测试人员呢？Cem Kaner 指出：“测试过程的一个重要成果，是更好、更聪明的测试人员。”[Kaner01]优秀的测试人员具备高超的技能，而这种技能只能通过持续的学习和实践才能获得。而且在一个合作与分享的环境中，测试人员可以学得更快、练得更好。

1.3 探索式测试的问与答

本节用对话的形式讨论探索式测试的概念与实践。提问者是本书的一位虚拟读者，回答者是本书的作者们。

问：探索式测试中的“探索”该如何理解？

火龙果•整理
uml.org.cn

答：所谓探索是指有目的的漫游，即带着使命在某个空间中漫游，但没有预先确定的路线 [Kaner01]。探索包括对产品与技术的深入研究和基于研究成果的实践应用。

问：如何实施探索式测试？

答：本书第 3 部分将专门讨论该问题。这里先介绍一种可行的探索式测试实施方法，其灵感来源是基于测程[9]的测试管理（Session-Based Test Management）[Bach2000]。

探索式测试鼓励测试人员依据当前语境选择合适的测试流程与技术。在测试过程中，SMART 原则[10]为测试者提供了很好的指导。

- Specific（具体的）：测试需要一个具体的目标。
- Measurable（可度量的）：有明确的指标可以评估目标是否达成。
- Achievable（可实现的）：目标应该是可实现的。这潜在地要求将一个大目标分解为多个小目标，且每个小目标也是具体的、可度量的、可实现的。而且，追踪小目标的完成情况提供了整体进度的可度量性。
- Relevant（相关的）：目标要切合当前语境，符合团队利益，且不忘企业愿景。
- Time-boxed（有时间限制的）：为每个目标设定一个合理的最后期限。这可以帮助测试人员在固定的时间窗口（Time Window）中排除不相关干扰，专注地工作。

依据 SMART 原则，测试人员可按如下描述展开探索式测试。

9 在 Session-Based Test Management 中，Session 是一段专门用于探索式测试的时间，其常用的中文术语“会话”并不适合该语境。经过慎重考虑，笔者将 Session 翻译为“测程”，以表达它是一段专注于测试的过程。

（1）测试人员制订测试计划。分析被测试应用，确立若干个具体的测试使命（Mission），每个使命针对一个可能的产品风险。

（2）测试人员将测试使命分解为一系列测试任务（Charter），每个任务都有明确的退出条件和时间限制。

（3）在短暂的测试计划之后，测试人员根据优先级选择一个任务，在一个固定的时间窗口中执行探索式测试（窗口的长度是 60~120 分钟，以 90 分钟为宜）。这样一个时间窗口被称为一个测程（Session）。在该测程中，测试人员设计测试、执行测试、评估测试结果。他会根据获得的知识和发现的疑问再设计测试用例，以拓展测试的广度和深度。

（4）在测程结束后，测试人员适当休息，放松思维。

（5）随后，他会反思当前的测试进展，并优化测试计划。也许他会为当前任务追加一个测程；也许他会再增加一个新的任务以弥补先前测试计划的不足；也许他会删除一些任务以反映他对测试对象的最新认知。

（6）这时，他会更有自信地开始新一轮探索式测试。

以上只是一种可能的探索式测试实施方法。负责任的测试人员一定会选择他自己的方式展开测试，因为只有作为领域专家的他，才能做出最符合语境的决策。此外，集合整个团队的力量，进行同行评审、头脑风暴、结对测试等活动，有助于产生更好的测试结果。

问：探索式测试与即兴测试（Ad-hoc Testing）有何区别？

答：探索式测试与即兴测试都强调“即兴发挥”，即利用直觉和经验，快速地测试软件，并不停地调整测试策略。软件专家 Andrew Hunt 指出，直觉是非显性知识的代名词，是大脑富（Rich）模式的杰出能力。如果人类只使用大脑的线性模式（包括语言可表达的显性知识、抽象能力、逻辑能力等），而漠视富模式的能

量，我们将浪费自身的巨大潜力[Hunt08]。

然而人是不完美的，某些直觉可能是认知偏见或错误。这就引出了探索式测试与即兴测试的关键不同：探索式测试是带着“反思”的测试。在探索式测试中，测试人员不断地提出假设，用测试去检验假设，并分析测试结果来证实或推翻假设。在此过程中，测试人员持续完善头脑中的产品模型和测试模型，然后利用模型、技能、经验去驱动进一步的测试。通过将测试学习、设计、执行和结果分析作为相互支持的活动并行展开，探索式测试总是在不停地优化测试模型、测试设计和测试价值。因为测试设计和测试执行的切换速度很快，许多人误以为探索式测试没有计划和设计。实际上，这些活动被划分到细微的时间片中，被反复执行。

即兴测试往往利用错误猜测、典型风险和常见攻击来快速地试探软件，可以在短时间内发现许多软件错误。但是即兴测试不强调测试的系统性和完整性，测试遗漏的风险很高，也难以发现一些需要深入研究才能发现缺陷。探索性测试通过测试来透彻地理解被测试产品，从而拓展测试的广度与深度，以持续优化测试的价值。

问：如果探索式测试是硬币的正面，那么硬币的反面是什么？

答：探索式测试的对立面是脚本测试（Scripted Testing）。脚本测试要求预先编写好测试脚本（Script），脚本规定了如何配置被测试软件、如何输入、如何判断软件输出了正确的结果。编写详细的脚本往往需要大量的测试资源。

如果运用得当，脚本测试可能有如下收益[Kaner08]：

- 测试人员可以仔细地思考被测试软件。
- 测试脚本可以被项目关系人（Stakeholder）评审。
- 测试脚本可以被复用。
- 测试团队可以评估测试脚本集的完备性。
- 测试团队可以度量测试脚本的执行情况，以评估测试进度。

火龙果•整理
uml.org.cn

问：本书为什么反对脚本测试？

答：笔者不反对任何测试思想或方法，但是反对不分语境地滥用某一种测试思想或方法。例如，苛刻地要求编写详细的测试脚本可能会导致如下测试风险。

- 在测试执行前，大量的测试资源被用于测试设计。但是，产品的发展往往难以预料，早期的预先设计不能有效地处理动态变化的情况。有可能花费了大量的时间，却获得了一批充满缺点的测试脚本。
- 过于详细的测试脚本压抑了测试执行的灵活性，使得测试执行变成单调的过程。这可能导致测试人员对一些明显的错误视而不见，因为它们不在测试脚本的检查范围之内。此外，运行测试是观察软件行为、获得测试灵感、设计新测试用例的极佳时刻，这要求测试人员在测试执行时全神贯注、头脑灵活、反应敏捷。但是，枯燥的测试执行将使这些目标难以实现。
- 大量的详细测试脚本导致了沉重的维护代价。在进度压力下，测试人员可能没有时间更新测试脚本，这使得测试脚本不能随需求和产品一同演化。随着时间的推移，大量的测试脚本成为过时的、无人问津的“摆设”，原先投入大量资源所获得的测试资产几乎消耗殆尽。
- 要求测试人员编写求全责备的测试脚本，很可能带来不良的心理暗示：测试人员在不知不觉中将编写脚本看做测试的目的，而不是辅助测试的工具。他们会盲目追求脚本的数量，而很少关注产品和项目的风险，用看似“完备”的脚本集提供虚假的安全感。更糟糕的是，醉心于脚本数量的测试领导很可能会鼓励这种以“文案工作”为核心的测试流程，使测试的价值进一步流失。

相比“硬币”的比喻，笔者更喜欢 Cem Kaner 的观点：“纯粹的探索式测试和纯粹的脚本测试好似区间的两个端点。在实践中，大多数测试者位于这两者之间。然而，大多数好的测试非常接近探索式测试的一端。”[Kaner09]

此外，根据语境驱动测试的基本原则，测试人员需要经常反思：根据当前语境，测试策略是应该偏向探索式测试，还是脚本测试？如何综合它们的优势，以

获得更好的测试结果？

问：探索式测试编写测试文档吗？

答：探索式测试者创建任何有助于实现其目标的文档 [Kaner08]，他会撰写并维护符合语境的测试文档。这里提出一种可能的测试计划编写方法，供读者参考。

在不同的开发组织，测试计划有不同的名称：测试计划、测试规格说明和测试设计文档等。但是，大多数测试计划会涵盖产品概述、测试范围、测试风险、测试策略、部分详细测试用例、资源分配和日程安排。

与脚本测试在测试执行前编写大量文档不同，探索式测试在整个测试过程中持续编写、修改、优化测试计划。测试计划的内容、格式、详略是由项目语境决定的。

在项目计划阶段，测试计划可以包含产品概述、测试范围、测试风险、测试策略、资源分配和进度安排。编写文档的目的是建立对产品的整体认知，根据风险提出相应的测试策略，并安排测试任务和日程。测试计划不必面面俱到，用精简的格式记录必要的内容即可。此时项目的不确定性较大，未知因素较多，很可能不适合做出详细的测试决策。

在项目计划阶段，应该邀请项目关系人对测试计划进行评审。评审的形式可以多种多样，会议评审、桌面走查、邮件评审、在线文档批注、头脑风暴皆可。评审的目的是发现“大图景”上的缺陷，并丰富测试策略。眼界开阔的项目经理、负责任的开发人员、经验丰富的测试同事能够提供很好的建议。

在测试阶段，测试人员需要根据测试进展持续更新测试计划，内容可能包括产品模型、检查列表、测试策略、覆盖大纲和风险列表等。测试计划应该是“活”的文档，能够反映测试人员对产品和项目的最新认识。对测试范围、测试风险、测试策略和进度安排的重大变化应该写入测试计划，重要的测试用例（如发现严

重缺陷的测试用例）也应该及时融入测试策略。测试计划应该尽可能地精简，这有助于文档的持续更新，而冗长的文档将不利于阅读、增删和修订。

当测试人员认为测试计划相对完整之后，他还应该邀请项目关系人对测试计划进行评审。评审的目的是发现测试遗漏，补充测试策略，提高测试覆盖率。

以上方法有三个重要的特点。

- 测试计划的编写与改进贯穿整个测试过程。在测试执行之前，对测试计划不必求全责备。在测试执行中，测试人员根据测试反馈，动态地调整测试范围、项目风险和测试策略，生成新的测试用例，并对测试计划做必要的更新。
- 测试人员持续地收集对测试计划的意见，并将改进意见纳入测试实践。正式和非正式的测试计划评审可以识别潜在风险，丰富测试策略。在一些测试流程中，测试计划评审只发生在测试开始之前。但是，笔者建议在测试流程的中期再次评审测试计划，目的是进一步丰富测试策略的多样性，并发掘可能的测试遗漏。这时，测试团队对产品与风险拥有更深刻的理解，能够提出更具针对性和多样性的测试策略，也可以帮助测试人员发现测试盲点，从而提高测试覆盖率。
- 测试计划侧重测试规划（一组指导测试过程的想法）和测试策略（一组指导测试设计的想法）[Kaner01]，而不是脚本化（Scripted）的测试用例，即测试计划用精简的格式表达测试人员的测试想法，而不必详细记录测试用例的设置、步骤和预期结果。测试人员可以用文字、列表、表格、思维导图等任何合适的方式表达想法，目的是激发测试灵感，并促进测试思想的交流。此外，他会利用发现缺陷的测试用例来完善测试策略，而不是让过去的测试用例控制未来的测试。

关于其他形式和内容的测试文档，测试专家 Michael Bolton 的文章 What Exploratory Testing Is Not: Undocumented Testing[11]提出了很好的建议。

火龙果•整理
uml.org.cn

问：探索式测试的核心优势是什么？

答：探索式测试的核心优势是有助于“学习”。此处的学习是指学（获取知识）与习（应用知识）的持续过程。

对于测试人员，软件测试是一个持续学习并实践的过程。他的学习范围如下。

- 行业知识：为什么需要这个软件？软件如何帮助使用它的人和团体去获得成功？
- 用户角色：目标用户是谁？他们有什么特点，有什么期望？软件如何帮助他们去获得个人成就？
- 软件产品：产品是一种解决方案，它解决了行业和用户所面临的问题吗？
- 计算平台：只有深刻理解软件所依赖的计算平台（如操作系统、中间件和网络协议等），才能更好地进行测试。
- 开发技能：理解项目所使用的具体技术，知晓典型的技术缺陷，具备测试开发的能力。
- 测试技术：针对当前项目，选择合适的测试技术，并能够熟练地应用。
- 程序缺陷：研究已知的软件缺陷，提炼错误模式，制订缓解或预防方案。
- 开发团队：语境决定策略和实践。在一起工作的人，是所有项目语境中最重要的组成部分。

测试人员不需要在项目之初就掌握所有知识，他可以通过每天的工作去逐步理解用户、项目、技术和团队。更重要的是，他需要在每天的工作中实践所学的内容：规划测试方案、创建并执行测试用例、分析测试结果和编写测试报告。实践是练习，是“学”的自然延伸。知行合一才构成“学习”的完整内核。

学习的一个重要成果是成为更优秀的测试人员。他们可以根据项目语境，选择合适的流程、技术和工具来高效地测试，以推动软件质量的提高。

既然学习非常重要，那么如何才能高效地学习呢？软件专家 Andrew Hunt 指出："一种高效的学习环境应该允许你安全地做三件事情：探索、创造和应用。"[Hunt08] Andrew 的解释如下：

- 探索就是在陌生的环境中玩（Play）。你需要自由地探索才能学习。我们不仅仅接受信息，而是亲自探索和构建思维模型。玩引入了一种新奇的感觉，也就是乐趣。用一种好玩的方式学习新资料或者解决问题，可以让这个过程变得更让人享受，也让学习变得更容易。为了更好地学习，请更好地玩。
- 你需要自由地创造——不介意自己的创造没有成功。通过构造来学习，而不是通过学习来构造。这是"原型"、"曳光弹"[12]、持续集成等方法获得成功的原因之一。
- 你需要在日常实践中应用到你学到的东西。你持续使用和实践的技能会在脑皮层竞争中占据统治地位，大脑会为它们提供更多方便。
- 这种探索应该相对没有风险，因为你肯定不想因担心害怕而止住探索的脚步。安全有助于更好地利用反馈，让失败也变得有意义。

虽然 Andrew 讨论的是广义的学习与探索，但是他的话解释了探索式测试的成功之道：探索式测试在本质上鼓励测试人员去自由地探索、创造和应用。

- 探索是带着使命在某个空间中有目的的漫游，但没有预先确定的路线。探索包括不断学习和实践[Kaner01]。
- 探索式测试者持续应用他所学到的知识。所谓"学而时习之，不亦说乎"，探索式测试将学习与应用作为相互支持的活动逐步展开，为测试者的知识提升提供了平滑的学习曲线。
- 探索式测试有助于测试人员在合适的时间学习合适的内容，并立即应用，

12 曳光弹是 Andrew Hunt 和 David Thomas 在《程序员修炼之道》（http://book.douban.com/subject/1152111/）中提出的开发方法，通过快速实现系统的部分功能以获得即时反馈。曳光弹与原型制作都可以快速获得反馈，其主要差别是，原型制作生成实验性且最终被抛弃的代码，曳光弹生成简约却完整的代码，并且构成了最终系统的一部分。

火龙果•整理
uml.org.cn

以获得真实反馈。这提高了学习效率和效果，并降低了浪费测试资源的风险。

- 在探索式测试中，测试者创造出一切有助于学习和实践的数据和工具。它们包括测试输入数据、结果检查脚本、数据分析报告和业务流程图表等。
- 探索式测试是一项富有智力挑战的活动，充满了乐趣。

问：“学习”有助于独立测试人员更好地工作与发展。从测试团队和开发组织的角度看，探索式测试能够提供什么帮助？

答：探索式测试有助于“机动性”，该特性在现在比以往任何时候都更重要。

随着互联网与 Web 应用席卷全球，软件竞争比以往更加激烈。开发组织不仅要减少缺陷，还要跟踪不断变化的用户需求和市场需求，在紧迫的时间约束下交付赢得用户的产品。这要求软件企业在业务、研发、营销等方面均保持高机动性。其中，探索式测试可以在以下方面为产品研发提供帮助。

首先，探索式测试将许多测试决策置于更合理的时机，从而避免了浪费。在 Web 应用等高竞争性领域中，开发组织面临模糊且持续变更的用户需求。更重要的是，他们必须在一切明朗之前着手行动，因为交付的时机将在很大程度上决定市场的反应，此外真实的用户反馈将有助于下一次更准确地交付。面对高变动性的开发过程，测试人员需要将测试设计合理地分配到整个开发周期，根据当前的开发进度和真实的测试反馈，做出恰当的测试决策。这有助于避免在信息不充分的情况下做出错误的决定，从而导致严重的浪费和返工。

其次，探索式测试不停地根据实际情况，调整测试计划，优化测试策略，因此能够更有针对性地测试，从而更快速地稳定（Stabilize）产品。探索式测试和语境驱动测试建议，测试人员总是自问：“我现在可以测试什么？应该如何测试？”这有助于测试人员更动态地审视被测试产品和测试策略，并运用多样化的手段去探索产品。随着对业务领域的认识不断深入，随着逐渐了解产品的设计和弱点，随着对测试技术和工具的更加熟悉，测试人员会不断调整测试策略，使得在整个

产品开发过程中，重要错误的发现率都保持在比较高的水平[Kaner01]。而及时地发现重要错误能够帮助开发组织降低风险、快速推进。

测试需要探索，而探索要要大量的思考[Kaner01]。积极思考的探索式测试是具有挑战性的智力过程，常常需要以不确定的顺序反复实施钻研、尝试、迂回、调整、评估等活动。好的探索式测试不会使测试更简单，但是它会使测试更有效，从而在测试速度和缺陷发现上获得更多的回报。

问：探索式测试是否只适用于功能性测试？

答：作为一种测试风格，探索式测试可应用于各种类型的测试。

以安全测试为例，请想象一下黑客是如何攻破软件产品的。他们的基本方法是“错误猜测”：通过分析已知的安全缺陷，抽象出可利用的缺陷模型，然后开发出相应的缺陷挖掘工具。这些工具可以是黑盒工具，通过持续地输入攻击数据来发现缺陷；也可以是白盒工具，通过扫描（开源）源代码来识别漏洞。他们运行工具，分析输出中的蛛丝马迹，一旦发现疑似缺陷，便深入探索。整个攻击过程常常需要广泛扫描、深入挖掘、迂回前进、反复尝试。从安全测试的角度看，黑客都是探索式测试的绝顶高手。

相比安全测试只需“断其一指”，性能测试需要建立被测试产品的性能模型，并考察它在不同负载下的性能指标，因此需要更多的预先设计。然而性能测试的关键内容：用户的行为模式、充足的高质量测试数据和完整的性能监测指标（特别是面向业务的性能指标），是难以仅通过一次测试设计便可以获得的。性能测试工程师需要与开发团队紧密协作，通过阅读、研究、实验等手段来探索性能测试模型和技术，并逐步挖掘用户行为、制造测试数据及建立性能指标。

问：在功能实现之前，探索式测试是不是无用武之地？

答：对于探索式测试有一个误解，那就是探索式测试只通过运行测试以获得知识。实际上，探索式测试者能够也必须通过其他手段探索被测试软件。

语境驱动测试认为：产品是一种解决方案，它必须解决现实世界中的问题。因此，探索式测试者必须从项目关系人（包括软件用户、项目投资人和开发团队等）的视角考察整个产品，研究显式规格说明和隐式规格说明，从而发现软件在概念、需求、设计、实现等层面上的缺陷。值得强调的是，测试人员应该主动探究隐式规格说明，从而拓宽探索的范围。以下是一些常见的隐式规格说明。

- 竞争产品。竞争产品不一定是软件，例如笔记软件的竞争者就包括纸质笔记本和铅笔。
- 相关产品。软件套装（如 Microsoft Office）中的软件往往相互补充、相互约束。
- 同一软件的已发布版本。向前兼容可能是非常重要的约束。
- 口头讨论、采访、闲聊。
- 电子邮件、会议记录、采访记录。
- 用户反馈：电话支持记录、论坛讨论、Beta 测试反馈。
- 技术反馈：软件提交的崩溃信息、错误消息。
- 第三方评论：杂志文章、博客文章、社交网络反馈。
- 技术标准。所有软件都建立在一系列技术标准之上，某些标准会对测试产生重要影响。
- 法律法规。法律可能对软件有强制性约束。
- 领域专著。测试财会软件需要学习相关著作，以掌握财会知识。
- 测试人员经验。

Cem Kaner 拥有法律学位，对语言文字非常敏感。他认为积极阅读（Active Reading）是探索式测试者需要具备的技能 [Kaner08]。积极阅读是一个内涵丰富的主题，广受好评的经典书籍包括《如何阅读一本书》[13]、《探索需求》[14]和《你

的灯亮着吗？》[15]。

此外，在功能实现之前，可以通过小组讨论、头脑风暴等方式发掘测试策略和测试思路。针对一个开发中的功能，测试人员可以邀请几个同事，组织一个测试研讨会。在会议上，大家自由发言，提出自己的测试策略，通过脑力震荡相互启发，从而获得一批观点各异、视角不同的测试策略。会后，测试负责人对这些相对粗糙的测试思路进行整理，将完善后的结果写入测试计划。

问：如何评估探索式测试的测试效果？

答：评估探索式测试结果的前提是测试记录。

探索式测试者常常在一个固定的时间窗口内（60~120 分钟），根据预设的测试目标，对软件进行 Z 探索式测试。这样的测试活动被称为测程（Session）[Bach00]。

测程类似于科学实验。一次科学实验大致可以分为以下三个阶段。

（1）实验设计：确定实验目标。科学实验的常见目的是假设检验，那么此次的假设是什么？

（2）实验记录：执行实验步骤，并记录实验所发现的点点滴滴。

（3）实验分析：分析实验数据，总结实验结果，为下一次实验提供目标和假设。

相应地，一次测程包含如下三个阶段。

（1）测试计划：明确测试目标。测试是获得信息的过程，那么此次测试要获

得什么信息？

（2）测试执行：设计并执行测试用例，记录测试所发现的点点滴滴。

（3）测试分析：分析并总结测试所发现的信息，为下一次测试提供目标。

详细的实验记录是科学实验的基本要求之一。同理，详略适当的测试记录也是测试执行的自然结果，是测试评估的基本要求。通常，测试记录可以包含如下内容[Wikipedia11a]。

- 测试目标：本次测试要提供什么信息？
- 测试范围：本次测试覆盖了哪些功能、模块、用户情景？
- 测试策略：本次测试使用了何种测试方法？
- 缺陷列表。
- 在测试过程中发现的疑问，它们值得进一步探索。
- 可以复用的测试资源：被测试软件配置、测试数据和测试脚本等。
- 测程的耗时。
- 测程的时间分配：在测试设计与执行、缺陷调查与报告、测程的启动与结束和非测试活动上各花费了多少时间。

测试记录可以转化为测试备忘录，供今后的测程参考。测试记录也可以提炼为测试报告，反映当前项目的进展。更重要的是，测试记录是测试评审的素材。基于测试记录，测试团队可以开发出符合项目语境的评估方法，对测程进行专家评审和定量度量。这有助于度量探索式测试结果，并提出改进方案。

问：探索式测试只适合测试专家，不适合测试新手？

答：“探索式测试不适合测试新手”是一种似是而非的说法。第一，所有高效的测试都依赖于测试人员的测试技能和行业知识。测试专家能够准确地选择测试策略、有效地运用测试方法，因此测试效果更佳。第二，测试新手采用任何测试方法，都需要指导和帮助。这有助于他们充分利用方法的优点，并避免方法的潜

在陷阱。可见，更有意义的问题是：如何帮助测试新手尽快地掌握测试方法，尽快地成长为测试专家？

从个人发展的角度看，探索式测试有助于测试新手快速学习。探索式测试将学习与应用作为相互支持的活动逐步展开，为测试人员的技能提升提供了平滑的学习曲线。此外，并行地进行测试学习、测试设计、测试执行和测试评估为测试人员的成长提供了持续、及时、有效的反馈，这有助于他主动学习和快速调整。

从企业发展的角度看，测试团队应该积极帮助测试新手成长。可以采用的方法包括：为他安排工作导师、评审其测试文档、评审其测试记录、在测程中安排测试专家与他结对测试、定期进行一对一的会谈等。这些活动会消耗测试团队的人力资源，但是它们是帮助新员工成长最快速、最有效、最廉价的方法。

Peter Drucker 指出：知识工人的创造性（Productivity）要求他们被视为企业的资产（Asset）而不是开销（Cost）[Drucker99]。培养高水平测试人员是测试团队和测试领导不可回避的职责。

问：有什么工具可以支持探索式测试？

答：本书第 5 章将讨论探索式测试的工具。这里强调两个基本观点。

第一，作为一种测试风格，探索式测试可以使用任何开发和测试工具。探索式测试者应该根据语境选择合适的工具，去完成自己的使命。

第二，软件测试存在大量的创新空间，测试人员应该勇于开发自己的探索式测试工具。

测试专家 James A. Whittaker 提出过一种测试工具构建方法[Whittaker01]，值得测试人员参考。

（1）寻找缺陷：发现或收集软件的缺陷。

（2）提炼模式：分析出缺陷的根本原因，编写一个模式，用它捕获相似的缺陷。一个模式是一个结构化的攻击手段，它包含如下内容。

- 何时实施该攻击？
- 该攻击会捕获何种错误（Fault）？
- 利用该攻击如何识别软件失败（Failures）？
- 如何实施攻击？
- 样例和分析。

（3）开发自动化工具：识别出攻击过程中机械的部分，编写一个工具去自动化模式的应用。此处的测试自动化不是自动地执行测试用例，而是提供计算机辅助功能，其目的是让计算机完成高负荷运算，让人专注于富有智力挑战的任务。

按照 James 的方法实施软件测试，测试团队可以积累一批有益的模式和测试辅助工具。这可以帮助团队更有效地测试现在和未来的项目。

问：探索式测试能用于测试自动化吗？

答：本书第 6 章将讨论探索式测试与测试自动化。这里简单陈述一下笔者的观点。

- 测试自动化可以大致分为测试用例自动执行（Automated Test Execution）和计算机辅助测试（Computer-Assisted Testing）。
- 对于测试用例自动执行，探索式测试可以提供一批适合自动执行的测试用例。
- 对于计算机辅助测试，探索式测试要求人尽其才（自由发挥测试者的智能）和物尽其用（充分利用计算机的性能），利用计算机强大的计算能力去帮助测试人员完成测试使命。
- 许多复杂的测试自动化应该以探索式的风格来构造。对于困难的测试，应该先构建简单的测试代码，将其投入测试，利用测试反馈来改进测试代码。然后，将改进后的测试代码投入测试实践，分析测试反馈，再优化测试代

码。所谓探索式测试自动化，就是将学习、设计、实现、评估纳入迭代开发，以逐步提高测试自动化和产品的质量。

问：探索式测试与敏捷测试有何关系？

答：探索式测试在本质上是敏捷的，且可以很好地应用于敏捷项目。

2001 年，17 位软件专家在美国犹他州雪鸟（Snowbird）城集会，缔结敏捷联盟，并发表敏捷宣言。与会者之一 Brian Marick 具有深厚的测试背景，因此软件测试社区对敏捷宣言亦有贡献。

敏捷软件开发宣言[Agile01]

我们一直在实践中探寻更好的软件开发方法，身体力行，同时也帮助他人。由此我们建立了如下价值观:

个体和互动高于流程和工具

工作的软件高于详尽的文档

客户合作高于合同谈判

响应变化高于遵循计划

也就是说，尽管右项有其价值，我们更重视左项的价值。

语境驱动测试的基本原则“任何实践的价值都取决于其语境”和“人，在一起工作的人，是项目语境中最重要的部分”，与敏捷宣言的首条价值观“个体和互动高于流程和工具”不谋而合。高效的探索式测试不但需要优秀的测试人员，也要求测试人员、开发人员、客户和项目关系人紧密协作、频繁互动。

在思想层面，探索式测试要求测试人员不断地研究产品，通过应对、激励、

拥抱变化来驱动对问题空间的积极探索。这不但符合敏捷价值观，也可以应用于其他类型的测试项目。敏捷测试专家 Lisa Crispin 和 Janet Gregory 指出："敏捷测试可以发生在敏捷项目之外。例如探索式测试，无论它是否应用于敏捷项目，其本质是敏捷的。通过测试逐渐学习产品，并让所学信息指导测试实践，这无疑符合敏捷价值观：工作的软件和响应变化。"[Crispin09]

在实践层面，探索式测试是评价产品的面向业务测试的主要手段[Crispin09]。在用户故事和测试策略的指导下，测试人员会模拟真实用户去测试系统。当一部分代码被签入，一部分用户故事被实现后，测试人员就会探索新的区域，并逐步完善测试模型和测试策略。随着测试人员对产品的了解，探索式测试不但可以弥补自动化测试的不足，还可以揭示出更有效的自动化测试区域，为自动化测试设计添砖加瓦。此外，探索式测试能够发掘新情景（Scenario），而这些情景往往会演变成新的用户故事，从而在需求层面提高产品质量。

从术语的历史看，"探索式测试"（Exploratory Testing，Cem Kaner 提出于 1983 年）的历史比"敏捷软件开发"（Agile Software Development，敏捷宣言缔结于 2001 年）更悠久。它们都是在描述已经长期存在但是没有得到合适命名的思想及实践：Cem Kaner 用"探索式测试"来描述一种已经长期存在的测试思维，敏捷宣言的缔造者们用"敏捷"来描述他们对软件开发的共识。虽然这些思想来自不同的头脑、实践和社区，但是它们拥有相似的核心，并可以相互借鉴与补充。