

测试思维

-启发式测试策略模型

讲师：刘晓光
新浪微博：@skytraveler
邮箱：xgliu@travelsky.com

游戏1

- 尝试在3分钟内记住下列字母

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

游戏2

- 测试新浪微博手机客户端



 **新浪微博** 官方版

★★★★★ 8.5分 (已有13838人评价) 点星星可进行评分

编辑评语: 没微博? 你老好意思出门吗? 新浪官方微博客户端赶紧来个。

来自: N多市场 其他来源: 机锋 木蚂蚁 更多来源 ▾



问题

游戏1你想起了什么？

游戏2你想起了什么？你认为刚才的测试方法好么？

问答

游戏1和游戏2有什么联系？

游戏1里的字母其实给出了做游戏2的思路。
这些字母也是我们今天所要讲的**测试思维**的骨架。

什么是测试思维？

测试思维是一系列**方法论**，它能够帮你**建立**或者**理清**测试的工作思路。

什么是启发式测试策略模型？

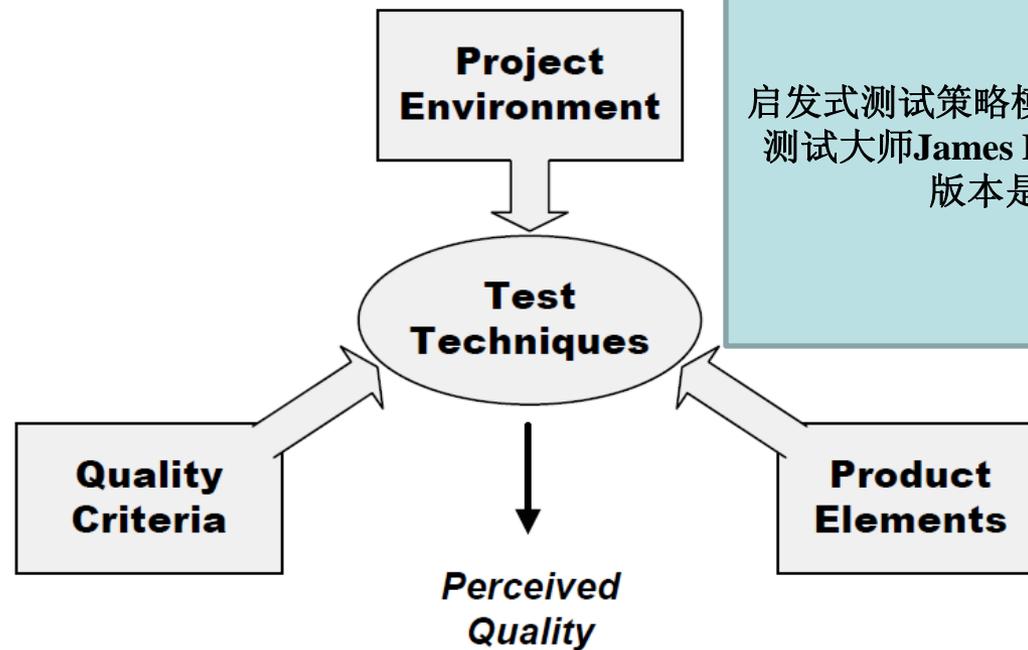
启发式测试策略模型（HTSM）是测试思维的一种具体体现形式。

本课程主要内容:

- 启发式策略模型概述
- 启发式策略模型各个部分
 - 项目环境
 - 项目元素
 - 质量准则
 - 测试技术

启发式测试策略模型

Heuristic Test Strategy Model



启发式测试策略模型（HTSM）由测试大师James Bach建立，最新版本是4.8。

启发式测试策略模型-续

- 启发式测试模型（HTSM）是一组模式（pattern）的集合，我们在制定测试策略的时候可以使用这些模式。
- HTSM的最直接作用就是能够在测试全程提供有益的提示。
- 我们最终要把HTSM具体化，并应用到测试工作中去。

启发式测试策略模型-续

- 项目环境（ **Project Environment** ）包括资源、约束、以及项目中有助于我们测试的其它强力因素，当然也包括阻碍我们做好工作的因素。当面对阻力时，先确认你是否利用了你所有可获得的资源。

启发式测试策略模型-续

- 产品元素（Product Elements）：就是你要测试的东西。软件是如此的复杂和不可见，以至于你要特别小心的确保你确实测试了产品中需要测试的所有部分。

启发式测试策略模型-续

- **质量标准(Quality Criteria)**是允许你作为一个测试人员来判断产品是否有问题的准则、价值观和来源。质量标准是多方面的，并且经常是隐藏的或者是自相矛盾的。

启发式测试策略模型-续

- **测试技术(Test Technique)**是创建测试时使用的策略。所有的测试技术都包括对项目环境、产品元素和质量标准的某种分析。

启发式测试策略模型-续

- 看得见的质量(*Perceived quality*)是测试的结果。你**永远不可能知道一个软件产品的“实际”质量**，但是通过对应用的各种各样的测试，你能对其质量做一个比较准确的评估。

常用测试技术

功能测试（Function Testing）

- *测试系统能够完成的功能。*
- 找出产品能够做的事情（功能和子功能）。
- 判断你将怎么才能知道一个功能是否能正常工作。
- 测试每个功能，一次测试一个。
- 看看每个功能是否做了应该做的，而没有做不应该做的。

常用测试技术

域测试 (Domain Testing)

- *数据分治法 (Divide and conquer the data, 与等价类划分+特殊值的方法类似)*
- 分析产品的输入输出数据集。
- 判断测试的特殊值。考虑边界值，典型值，常用值，无效值，以及最具代表性的值。
- 考虑需要在一起测试的数据的组合。

常用测试技术

压力测试（Stress Testing）

- **征服产品。**（在规定的规格条件或者超过规定的规格条件下，测试一个系统，以评价其行为）
- 找出在挑战性的数据或者压倒性的资源面前对超载或者“破坏”敏感的子系统和功能。
- 辨识出与那些子系统和功能相关的数据和资源。
- 选择或生成测试所需的挑战性的数据或约束条件的资源：例如，大量或复杂的数据结构，高负载，长时间运行，大量的测试用例，低速存储器的情况。

常用测试技术

工作流测试（**Flow Testing**）

- *做完一件事再做另一件。*
- 定义测试过程或者高阶测试用例，这些过程或者用例把有交互的单个活动连接起来。
- 在两个测试之间不要重置系统。
- 改变时间和顺序，并且尝试并发的线程。

常用测试技术

场景测试（Scenario Testing）

- *将测试转化为一个有吸引力的故事*
- 首先，思考与产品关联的每一件事。
- 设计把产品中有意义的和复杂的相互作用包括在内的测试。
- 一个好的场景测试是一个关于一些人或事可能对产品进行怎样的操作故事。

常用测试技术

约束测试（Claims Testing）

- **验证每一条约束。**
- 在产品的参考资料中辨识出其包含的约束（隐藏的或直接的）。
- 分析单个的约束，并明确比较含混的约束。
- 验证产品的每条约束是否是正确的。
- 如果你测试的依据是详细的规格说明书，就很值得使用这个技术，并且会把产品做的比较标准。

常用测试技术

风险测试（**Risk Testing**）

- *设想一个问题，然后去找到它。*
- 这个产品会有哪些种类的问题？
- 哪种问题是最要紧的？关注那些问题。
- 如果真有这些问题，你将怎样来侦测？
- 做一个有趣的问题的列表，并特别设计一些测试来发现这些问题。
- 可能需要一些帮助，包括咨询顾问、设计文档、以前的缺陷报告、或者使用风险启发法。

常用测试技术

用户测试（User Testing）

- *涉及了用户的测试。*
- 分清楚用户的类别和角色。
- 判断每类用户会执行什么样的用例，怎样来执行，以及这样做的价值。
- 获得真实的用户数据，或者让真实的用户来测试。
- 否则，就要系统地模拟一个用户了（当心——想像你是一个用户很容易，但是实际上你并不是）。
- 非常有效的用户测试应该要涉及各种各样的用户和用户角色。而不是仅仅一个。

常用测试技术

自动化测试（Automatic Testing）

- 运行大量不同的测试。
- 寻找适合自动的生成大量的测试的机会。
- 开发一套自动化的、高速评估的机制。
- 写程序来生成、执行并评估这些测试。

回想： 游戏1中的字母

注意第一行：

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

F unction Testing	D omain Testing	S tress Testing
F low Testing	S cenario Testing	C laims Testing
U ser Testing	R isk Testing	A utomatic Testing

项目环境

- 创建和执行测试是测试项目的核心所在。然而，在你决定要创建什么样特定的测试时，项目环境中有很多因素都是关键性的。对于下面的每个类别中的因素，都要考虑它们可能会怎样帮助或阻碍你的测试设计进程。试着利用好每一个资源。

项目环境

客户(Customers)

这个测试项目中作为客户的任何人。

- 你知道谁是你的客户么？谁的意见要紧？谁从你做的的工作中受益或者利益收到损害？
- 你同你的客户联系和交流过了么？可能他们对你的测试有帮助。
- 可能你的客户对你要创建和运行的测试有很强烈的想法。
- 可能客户之间对测试有相冲突的期望。你可能不得不帮着分析清楚并解决这些冲突。

项目环境

信息(Information)

关于需要被测试的产品或项目的信息。

- 有任何的可获得的工程文档么？用户手册？网上的资料？
- 这个产品有历史渊源么？有已经被修复或者遗留的老问题么？客户有经常抱怨什么？
- 在你知道怎么测试该产品之前，你需要更熟悉该产品么？
- 你的信息是最新的么？你怎样得知新的或者修改的信息？
- 看起来信息好像异乎寻常少的产品中有任何复杂或者富有挑战性的部分么？

项目环境

与开发人员的关系(Developer Relations)

你怎样同程序员相处。

- 傲慢型：开发团队看起来对于产品的任何方面都过于自信？
- 防卫型：产品中存在开发人员似乎很奇怪地反对进行测试的部分？
- 融洽：你同开发人员发展出了一种伙伴合作关系？
- 反馈环路：你能同程序员根据需要进行快速交流么？
- 反馈：开发人员对你的测试策略怎么想？

项目环境

测试团队(**T**est **T**eam)

任何会执行或支持测试工作的人。

- 你知道谁会来测试这个项目？
- 有不在“测试团队”中，但可能会有帮助的人么？有人以前测试过类似的产品，并可能提供一些建议么？作者？用户？还是程序员？
- 你有足够的有正确的技能来执行一个合理的测试策略的人么？
- 这个团队有没有基于一些特殊技能或者动机地需要，来刻意执行使用一些测试技术？
- 需要任何的培训么？可以获得一些什么培训？

项目环境

设备和工具(Equipment & Tools)。

管理测试所需要的硬件、软件或者文档。

- 硬件：我们有执行测试所需要的所有的设备么？是否已经安装好，并准备运行了？
- 自动化：需要使用任何的自动化测试工具么？工具是否都准备好了？
- 探测器：在测试产品时，需要任何辅助性的工具来帮助我们观察么？
- 矩阵和一览表：有需要跟踪或者记录测试的进程的任何文档么？

项目环境

时间表(Schedule)

项目事件的顺序、持续时间和同步。

- 测试设计：我们有多少时间？这些测试晚一点创建是否会好一点？
- 测试执行：测试应该什么时候执行？是否有些测试要被重复的执行，比如说，在回归测试中？
- 开发：版本什么时候可以来测试，增加了功能，代码冻结，等等？
- 文档：用户文档什么时候可以评审？

项目环境

测试要素(Test Items)

被测试的产品对象。

- 范围：在你的测试职责范围中，包含产品中的哪一部分功能，不包含哪些？
- 可用性：你有需要被测试的产品么？
- 易变性：产品是否在不断的变化？再次测试时需要些什么？
- 新元素：最近，产品中新增或者修改了些什么？
- 可测试性：产品的功能和可靠性是否足够让你能有效的来进行测试？
- 将来的版本：你测试中哪部分，如果有的话，必须被设计来应用到产品的将来的版本中？

项目环境

可提交物(Deliverables)

测试工程的可以看得见的提交物。

- 内容：你要提交哪种报告？你会分享你的工作记录，还是只有结果？
- 目的：你的提交物是不是作为产品的一部分？有别的其他人要运行你的测试么？
- 标准：你有要遵循的一些特别的测试文档标准么？
- 媒体：你要怎样记录和并与其它人交流你的报告？

回想： 游戏1中的字母

注意第二行：

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

Customers **I**nformation **D**eveloper Ralations

Test Team **E**quipments&Tools **S**chedule

Test Items **D**eliverables

产品元素

- 产品元素
- 一个产品，最终是要提供给客户一种体验或者是一个解决方案。产品是有很多尺度的。因此，为了获得好的测试结果，我们必须检验这些尺度。下面所列的每一类，都代表着一个产品的重要的并且是独一无二的的一个方面。测试人员如果对这些方面关注得很少的话，很可能会错过很重要的Bug.

产品元素

结构(Structure)

构成产品本身的任何东西。

- 代码：构成产品的代码结构，从可执行到独特的例程（from executables to individual routines）
- 接口：子系统间连接和通讯的点。
- 硬件：对产品必不可少的任何硬件组件。
- 非可执行的文件：与多媒体和程序不同的任何文件，例如文本文件，样品数据，或者帮助文件。
- 附属品：产品中除了软件和硬件之外的任何其它部分，例如纸质文档，Web链接和内容，包装，许可声明，等等...

产品元素

功能(Functions)

产品能做的任何事情。

- 用户界面：用来给用户交互数据的任何功能（例如，浏览器，显示界面，数据输入窗口）。
- 系统接口：用来给不同于用户的其它事物交互数据的任何功能，例如，其它程序，硬件，网络，打印机，等等。
- 应用：用来定义或区分产品或者实现核心需求的任何功能。
- 算法：任何计算的功能或者算法的操作嵌入在其它功能里面的。
- 与时间有关的：暂停时间设置；每日或每月报告；每晚的批处理作业；时区；商业假日；利息算法；条款和担保期；以及要求精确的功能。

产品元素

功能(Functions)-续

- 改变：修改或改变某些东西的功能（例如，设置字体，插入剪贴画，从账户中取钱）
- Startup/Shutdown：启用和初始化或者退出产品时需要用到的任何方法与接口。
- 多媒体：声音，图片、视频、或者嵌入在产品中任何图形化的显示。
- 出错处理：侦测错误并从错误中恢复的任何功能，包括所有的错误信息。
- 集成交互：产品的功能之间的任何交互或接口。
- 可测性：提供的可以用来帮助测试的任何功能，例如诊断(diagnostics)，日志文件，断言，测试的菜单，等等。

产品元素

数据(Data)

产品处理的所有东西。

- 输入：产品要处理的任何数据。
- 输出：产品处理过的结果数据。
- 预设值：作为产品的一部分提供的任何数据，或者直接
在产品中创建的数据，例如初始化数据库，系统默认值，
等等。
- 配置项：任何内置的并且会在多个操作持续使用数据。
包括产品的状态或样式，例如参数设置，视图模式，文
档目录，等等。

产品元素

数据(Data) -续

- 数据排序：数据的任何顺序或排列，例如文字的排序，分类或未分类的数据，测试顺序。
- 数据规模(Big and little)：数据的大小和聚集的变化量。
- 异常数据：以不受控制或不正确的方式产生的无效的、被破坏掉的、或者产生的任何数据或者状态。
- 生命周期：在数据实体的整个生命周期中的变化，包括创建、访问、修改和删除。

产品元素

平台（**Platform**）

产品所依赖的所有事物（并且也是你项目的外部环境）

- 外部硬件：并不是要交付的产品的一部分，但是是为了保证产品运行的必须的硬件组件和配置（或者是可选的）：CPU，内存，键盘，外设，等等。
- 外部软件：：并不是要交付的产品的一部分，但是是为了保证产品运行的必须的软件组件和配置（或者是可选的）：操作系统，同时执行的应用程序，驱动，字体，等等。
- 内部组件：被嵌入在你的产品中，但是不是由你的项目生产的库或者其它组件。既然你不能控制它们，你必须决定如果它们失效了该做些什么来解决问题。

产品元素

操作（Operations）

产品将会被怎样的使用。

- 用户：各种不同的用户的属性。
- 环境：产品操作所在的物理环境，包括例如噪音、灯光、和分心的事物这样的元素。
- 正常操作(Common Use)：产品输入的模式和顺序可能会与通常的使用习惯冲突。这根据用户的不同而不同。
- 异常操作(Disfavored Use)：由不了解、误解、不小心或者恶意的使用产生的输入模式。
- 极端操作(Extreme Use)：挑战与产品期望的使用方法一致的输入模式和顺序。

产品元素

时间（Time）

产品与时间之间的任何关系。

- 输入/输出：什么时候提供输入，什么时候创建输出，以及他们之间的时间联系（延迟、间隔，等等）。
- 快/慢：用“快速”输入或者“慢速”输入来测试；最快和最慢；快和慢结合起来测试。
- 变化率：加速和减慢（峰值、突然爆发、中止、瓶颈、中断）。
- 并发：一次发生超过一件事情（多用户，分时，线程、信号、和共享数据）。

回想：游戏1中的字母

注意第三行：

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

Structure **F**unctions **D**ata

Platform **O**perations **T**ime

质量标准

- **质量标准类别**

- 质量标准是一些定义产品应该是什么样的需求。通过对不同类型的标准的观察和思考，你会能够比较好的制定一个快速发现重要问题的测试计划。下面每一个列表中的条目都可以被认为是一个潜在的风险区域。对于下面的每一个条目，判断对于你的项目是否是重要的，然后思考你怎样得出产品是否运行得很好还是很差的结论。

质量标准

- 能力（**Capability**）是否能完成需要的功能？
- 可靠性（**Reliability**）是否运行得很好并且在所有需要的情况下都不会失效？
 - 出错处理:产品在出错的时候抵御失效，并很快恢复，这在失效时是非常棒的。
 - 数据完整性：系统中的数据避免丢失或被破坏。
 - 安全：产品失效也不会对生命和财产造成威胁和伤害。

质量标准

- 操作标准（Operational Criteria）

质量标准

- 可用性(**Usability**) 对于一个真实用户来使用这个产品到底有多容易?
 - 可学习：产品的预期用户能够很快的掌握其操作。
 - 可操作：产品操作起来不会很费劲，也不会很麻烦。
 - 可接近：产品接近相关的标准，并且与操作系统的相关标准比较接近。

质量标准

- **安全(Security)**。在保护产品不受到非法的使用或入侵上做得有多好？
 - 证明：系统验证用户的确是她说的那个人的方式。
 - 授权：授予被证明的用户不同程度的权限级别的权利。
 - 隐私：客户或雇员数据避免受到未授权的人的危害的方式。
 - 安全漏洞：系统不能保证安全的方式（例如社会工程的脆弱性）

质量标准

- 可伸缩性/可度量性(**S**calability)。产品部署时是否能够很好的进行扩展（scale up）或者收缩（scale down）？
- 性能（**P**erformance）。系统有多快的响应速度？

质量标准

- **可安装(Scalability)**。产品安装到目标平台上有多容易？
 - 系统需求：如果一些必须的组件缺失或者无效，产品是否能够识别？
 - 配置：系统的哪一部分会受到安装的影响？这些文件和资源都保存在什么地方？
 - 卸载：当产品被卸载时，是否可以干净的移除？
 - 升级：新模块或者版本是否能很容易的增加？它们是否不改变已有的配置。

质量标准

- 兼容性（**C**ompatibility）。同外部组件和配置一起运行得有多么好？
 - 应用兼容性：产品和其它软件产品一起运行。
 - 操作系统兼容性：产品运行在特定得操作系统上。
 - 硬件兼容性：产品运行在特定的硬件组件和配置上。
 - 向后兼容性：产品同自身早期的版本一起运行。
 - 资源利用：产品没有使用不必要的大量内存、存储空间，或者其它系统资源。

回想： 游戏1中的字母

注意第四行：

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

Capability

Reliability

Usability

Security

Scalability

Performance

Installability

Compatibility.

质量标准

- 开发标准（Development Criteria）

质量标准

- 可支持性（**S**upportability）。是否可以很经济的为产品的用户提供支持？
- 可测试性（**T**estability）。产品能够被如何有效的测试？
- 可维持性（**M**aintainability）。是否可以很经济的对产品进行打包、修复或者增强？

质量标准

- 可移植性（**P**ortability）。是否可以很经济在其它地方移植或重用该技术？
- 本地化能力（**L**ocalizability）。产品是否可以很经济的适应其它地方？
 - 规则：是否有超越州或国家边界的不同的规则或报告的需求？
 - 语言：产品能容易适应更长信息,从右到左或者表意字的字体么？
 - 货币：产品能够支持多币种么？币种汇率呢？
 - 社会或文化差异：顾客可以发现文化参考资料中有令人费解的或者侮辱的吗？

回想： 游戏1中的字母

注意第五行：

FDSFSCURA

CIDTESTD

SFDPOT

CRUSSPIC

STMPL

Supportability

Testability

Maintainability

Portability

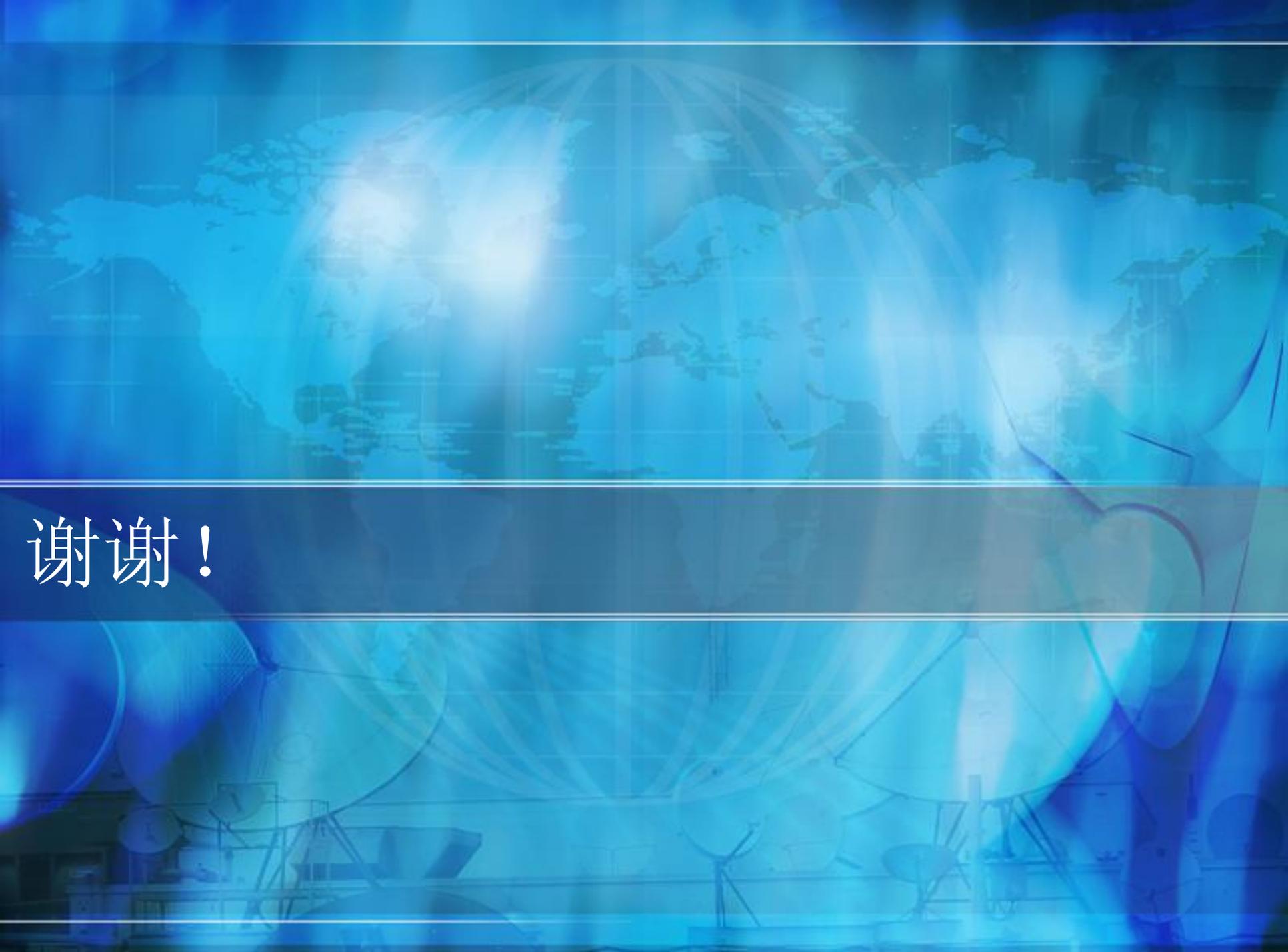
Localizability

常见问题

- 问：这些条目在一项测试中全部需要用到么？
- 答：当然不是，要根据实际情况进行裁剪和扩充。

- 问：有些条目有交叉，到底该归属到哪一类？
- 答：不必强制归类，重要的是启发式测试策略模型帮你**想到了**这些。

- 问：如何把启发式测试策略模型很好的应用到实践中？
- 答：通过学习-练习-改进的方式。首先记住这些条目，然后尝试比对你现在的测试工作，找到需要扩充的地方（总会有的），尝试根据实际情况进行扩充，改进你的测试。不断往复几次，并尝试多种项目，你就会发现它已经植根到你的脑子里了。



谢谢！