**测试工程师**

测试工程师的工作本质上是一个由表及里的工作，如果你还是每次工作都处在和终端用户使用行为几乎一致的流程上，那么只能说明你还不算一名合格的测试工程师。

**测试工程师的几个方面:**测试基础理论,数据库学习,Linux使用,接口测试,性能测试,自动化测试,其它.

1. 能迅速掌握被测软件的业务功能与内部架构，并在此基础上运用各种测试方法，尽可能多地发现潜在缺陷，并能够在已知缺陷的基础上进一步发现相关的连带缺陷。
2. 需要全面的计算机基础知识，还需了解互联网的基础架构、安全攻击、软件性能、用户体验和常见缺陷等知识.
3. 能够使用常见的测试框架或者工具，需要具有一定的自动化测试脚本的开发能力，这可以把你从大量重复的工作中解放出来，然后你才能有时间去做更有意思的工作.
4. 合格的测试工程师关注的是纯粹的测试，而优秀的测试工程师关注更多的是软件整体的质量，需要根据业务风险以及影响来制定测试策略，有效控制测试的时间和成本，并且能够对测试框架以及工具做出适合项目需求的选型.
5. 理解自动化脚本的分层设计、页面对象模型以及业务流程模型，并且能够把这些设计应用到你的测试框架里.
6. 面对大量测试用例的执行，无论是 GUI 还是 API，都需要一套高效的能够支持高并发的测试执行基础架构；再比如，面对测试过程中的大量差异性数据要求，需要统一的测试数据准备平台；再比如，为了可以更方便地和持续集成与发布系统（CI/CD）以解耦的形式做集成，需要统一发起测试执行的接口, 对一些前沿的测试方法和技术有自己的理解，并能够在恰当的时候、因地制宜地把它们应用到实际项目中.
7. **小到 Linux/Unix/Windows 操作系统的基础知识，Oracle/MySQL 等传统关系型数据库技术，NoSQL 非关系型数据库技术，中间件技术，Shell/Python 脚本开发，版本管理工具与策略，CI/CD 流水线设计，F5 负载均衡技术，Fiddler/Wireshark/Tcpdump 等抓包工具，浏览器 Developer Tool 等；**
8. **大到网站架构设计，容器技术(Docker)，微服务架构，服务网格（Service Mesh），DevOps，云计算，大数据，人工智能和区块链技术等。**

**测试基础理论**

软件开发流程及模型, 软件测试的定义、分类、测试的流程、测试用例编写方法以及缺陷（bug）的管理及生命周期。web端和app的常见测试点和测试方法, 会编写常见的测试文档，如测试计划、软件测试报告等。

**软件测试:** 在规定条件下对软件系统进行审核、运行、评估，检验软件系统是否满足规定需求或者找出预期结果与实际结果之间的差别。为软件产品的质量和评价提供依据。

**软件测试的目的:** 确保软件完成了它所承诺或公布的功能, 确保软件满足性能和效率的要求。确保软件是健壮的、适应用户的环境。提早预防、尽早发现、及时跟踪软件缺陷，满足产品发布需求。

**软件测试如何进行:** 通过手工和自动化方式，利用各种测试工具和管理工具等手段、更早、更快、更多的发现缺陷，并确保这些缺陷得以修复。

**软件测试对象:** 软件程序。与程序匹配的文档。支撑软件运行的配置数据。

**软件测试原则:** 有计划的尽早测试，解决问题的成本越小。测试并不能保证软件100%的没有问题。测试工作的本质都应追溯到用户需求。测试的规模由小而大，从单元测试到系统测试。Good-enough原则，穷举测试是不可能实现的。二八定律：80%的缺陷集中在20%的模块中，经常出错的模块修改后出现错误的概率也会更高。

**软件生命周期:** 从可行性研究到需求分析、软件设计、编码、测试软件发布维护的全部过程。

**软件开发常见模型:** **瀑布模型:** 强调产品定义，各步骤是分离的，前一阶段完成后才能开始后一阶段。缺点：无法回溯，测试在最后运行，惧怕需求变更。

**螺旋模型:**总体思想是一开始不必详细定义所有细节。从小开始，定义重要功能，努力实现这些功能，接受客户反馈，然后进入下一阶段。重复上述过程，直至得到最终产品。缺点：需要经常风险分析。

**敏捷开发:** 敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。

**V模型:**软件开发瀑布模型的变种，把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段，从左到右，描述了基本的开发过程和测试行为。

**W模型/双V模型:** 测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等开发输出的文档同样要测试。有利于尽早地发现问题。

**H模型:** H模型将测试活动从开发流程完全独立出来，使测试流程形成一个完全独立的流程，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以进行了。

**敏捷开发过程:** 产品经理收集需求，形成产品待办列表.项目团队参与迭代(sprint)计划会，形成迭代任务.

开发团队进行具体的开发任务，每日站会。开发团队进行成果演示（迭代评审会），产品经理，市场/高层，项目经理参与.评审通过，产品上线。评审不通过，再对产品做修改。开发团队与项目经理进行迭代回顾会，分析好的地方和需要改进的点。

**软件测试分类: 按测试阶段划分,** **单元测试：**完成对最小的软件设计单元模块的验证工作。对代码风格和规则、程序设计和结构、业务逻辑等进行静态测试。白盒测试技术，开发人员自己执行。

**集成测试:**通过测试的单元模块组装成子系统，然后再进行的测试，主要测试内容是接口。集成测试大部分是接口测试和交互测试。

**系统测试:** 将整个软件系统全部集成好之后作为一个整体进行的测试。主要包括一下测试方向：功能测试： 对产品的各功能进行验证，以检查是否满足需求的要求。 性能测试：通过自动化测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。 安全测试：检查系统对非法入侵的防范能力。 兼容测试： 测试系统在不同的软硬件环境下是否能够正常的运行。

**用户验收/确认测试:** **Alpha测试,**用户在开发环境下进行的测试，Alpha测试是在一个受控的环境中进行的。**Beta测试:**由软件的最终用户在一个或多个用户场所来进行的，开发者通常不在现场，用户记录测试中遇到的问题并报告给开发者。

**对代码的可见划分：黑盒测试：**又称功能测试或数据驱动测试，是针对软件的功能需求/实现进行测试，通过测试来检测每个功能是否符合需求，不考虑程序内部的逻辑结构。**常见的黑盒测试方法：**等价类划分,边界值分析.因果图,错误推测,场景法.

**白盒测试：**白盒测试也称结构测试或逻辑驱动测试，必须知道软件内部工作过程，通过测试来检测软件内部是否按照需求、设计正常运行.白盒测试方法：逻辑覆盖（语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖）、循环覆盖（简单循环、嵌套循环、串接循环）。

**灰盒测试：**灰盒测试介于黑盒测试与白盒测试之间。

**其他测试分类：回归测试：**对软件进行修改之后进行的测试，目的是检验对软件进行的修改是否正确。一是所做的修改达到了预定的目的，也就是确认测试，二是还要保证不影响软件的其他功能的正确性。

**冒烟测试：**在软件中，“冒烟测试”是指测试版本的主要功能，如果能通过测试，才继续进行接下来的其它全功能测试。对一个新版本进行大规模测试之前，先验证一下软件的基本功能是否实现，是否具备可测性。目的是确认软件基本功能正常，可以进行后续的正式测试工作。

**软件测试流程:**

产品需求,需求评审,开发人员写开发计划,测试人员写测试计划与排期,编写测试用例,测试用例评审.开发代码自测,提交测试且代建测试环境,测试人员测试问题反馈开发,开发修复后测试通过,编写测试报告,项目上线.

**测试人员介入流程：**

需求评审,分析需求与编写测试用例,搭建环境与配置数据,执行测试用例,发现bug且提交bug,回归测试,bug追踪与管理,测试报告.

**测试计划:** 软件测试工作正式实施以前，对测试资源、测试时间、测试风险、测试策略、测试范围等方面的分析和规划，保证有序有效的实施测试工作。不同公司根据项目内容和管理方式会有自己专有的模板，内容如下:

1.测试范围与目的. 2.测试方法. 3.测试环境与测试辅助工具的描述.4.测试启动准则与结束准则.5.已递交的文档(<<测试计划>>,<<测试用例>><<测试报告>>)与其预计递交时间. 6.测试人员的组织.7.任务表(测试任务,开始时间,结束时间,执行人员),8.可能存在的问题与对策.

**测试方案:** 测试需求的细化，明确测试测试策略，测试用例的设计方法，测试环境的规划，自动化测试框架的设计，测试工具的设计和选择，测试脚本和测试数据的设计。

**测试计划强调“做什么”。测试方案强调“怎么做”。测试计划属于管理文件，而测试计划是技术文档。**

**测试用例（Test Case）:** 对特定软件的具体功能模块的测试描述，由一组测试输入、执行条件以及预期结果等组成，以便测试某个程序路径或核实是否满足某个特定需求。

**测试用例包括：**用例编号、测试标题、重要级别、预置条件、测试输入、操作步骤、预期结果、实际结果与是否通过.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项目名称 | 平台系统 | 用例执行人 |  | | | | | | |
| 功能模块名 | 注册登录 | 程序版本 |  | | | | | | |
| 用例设计人 | 梁渡 | 编制时间 |  | | | | | | |
| 功能特性 | 注册账号登录 | 特殊规程说明 |  | | | | | | |
| 测试目的 | 注册登录正常 | | | | | | | | |
| 参考信息 |  | | | | | | | | |
| 用例编号 | 测试对象 | 检查点 | 预置条件 | 用例说明/操作步骤 | 优先级 | 预期结果 | 测试结果/(通过/不通过) | 缺陷编号 | 备注 |
| 注册页面\_001 |  |  |  |  |  |  |  |  |  |

**测试方法:等价类划分法,边界值分析法, 错误推测法,判定表, ,随机测试法.(日常常用前三种)**

**等价类划分：**把所有可能的输入数据划分成若干子集,然后从每一个子集中选取少数具有代表性的数据作为测试数据，就可以用少量代表性的测试数据。分为:有效等价类和无效等价类有效等价类:是指对于程序的规格说明来说是合理的,有意义的输入数据构成的集合。无效等价类：无意义的，不符合需求规定的集合。

如何用等价类划分设计用例？一般有如下几个步骤：

1、划分等价类和非等价类并编号。2、设计组合方式和可能性。3、根据组合选择数据生成测试用例。

例如注册功能：用户名要求：6到10位字符首字母必须是字母或数字，不能为空和汉字。密码要求：6到10位字符，不能为空和汉字。确认密码：与密码一致。划分等价类和非等价类：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入条件 | 有效等价类 | 编号 | 无效等价类 | 编号 |
| 用户名 | 6-10位 | 1 | 小于6位 | 8 |
| 首字母为字母 | 2 | 大于10位 | 9 |
| 首字母为数字 | 3 | 首字母不是数字或字母 | 10 |
| 密码 | 6-10位 | 5 | 少于6位 | 12 |
| 特殊字符及字符组合 | 6 | 大于12位 | 13 |
| 确认密码 | 与密码相同 | 7 | 与密码字符不同 | 15 |
| 大小写不同 | 16 |

组合方式：用户名正确、密码正确、确认密码正确，用户名无效、密码正确、确认密码正确

用户名正确、密码无效、确认密码正确，用户名正确、密码正确、确认密码无效

设计一个新的测试用例,使其尽可能多地覆盖尚未被覆盖地有效等价类,重复这步.直到所有的有效等价类都被覆盖为止

**边界值分析：**一般大量的错误都是发生在输入或输出范围的边界上,而不是发生在输入输出范围的内部。

应当选取正好等于,刚刚大于或刚刚小于边界的值作为测试数据,而不是选取等价类中的典型值或任意值作为测试数据。一般要取边界点上的上点、内点和离点。上点：边界上的点。内点：区间内的点离点：离边界值最近且与上点不属于同一等价类的点。对于小数不用考虑离点。如：（0，100]，上点：0、100  内点：50 离点：0,101。

**错误推测法：**基于经验和直觉推测程序中所有可能存在的各种错误, 从而有针对性的设计测试用例的方法。错误推测方法的基本思想: 列举出程序中所有可能有的错误和容易发生错误的特殊情况,根据他们选择测试用例。例如，在单元测试时曾列出的许多在模块中常见的错误。以前产品测试中曾经发现的错误等,这些就是经验的总结。还有输入数据和输出数据为0的情况，这些都是容易发生错误的情况。

**判定表：**分析和表述若干输入条件下，被测对象对这些输入作出相应的一种表格。在遇到复杂业务逻辑时可以用该表理清业务逻辑关系。

条件桩：需求规格说明书所定义的被测对象的所有输入。条件项：针对条件桩所有可能的输入数据的真假值。

动作桩：针对条件，被测对象所采取的操作。动作项：针对条件项的各种取值，被测对象响应的动作规则：任何一个条件组合的特定取值及要执行的相应操作。在判定表中贯穿条件项和动作项的一列就是一条规则。

1、确定规则个数，假如有n个条件.每个条件有两个取值（0,1）,故有2^n种规则。2、列出所有的条件桩和动作桩。3、填入条件项。4、填入动作项，等到初始判定表。5、简化，合并相似规则（相同动作）。

**随机测试法**：随意测试，不考虑任何测试用例和需求，完全站在一个用户的角度对产品进行使用。

适用于：所有之前设定的用例已经执行完毕。海量的条件组合没有办法意义遍历的时候。

**测试用例**

**“好的”测试用例一定是一个完备的集合，它能够覆盖 所有等价类以及各种边界值，而跟能否发现缺陷无关。**

**一个“好的”测试用例，必须具备以下三个特征。**

**1. 整体完备性： “好的”测试用例一定是一个完备的整体，是有效测试用例组成的集合， 能够完全覆盖测试需求。 2. 等价类划分的准确性： 指的是对于每个等价类都能保证只要其中一个输入测试通过，其 他输入也一定测试通过。 3. 等价类集合的完备性： 需要保证所有可能的边界值和边界条件都已经正确识别。**

**做到了以上三点，就可以肯定测试是充分且完备的，即做到了完整的测试需求覆盖。**

**对大多数的软件测试而言，综合使用等价类划分、边界值分析和错误推 测这三大类方法就足够了**

**用户登录的测试用例**

**基于等价类划分和边界值分析方法，测试用例包括：**

输入已注册的用户名和正确的密码，验证是否登录成功；

输入已注册的用户名和不正确的密码，验证是否登录失败，并且提示信息正确；

输入未注册的用户名和任意密码，验证是否登录失败，并且提示信息正确；

用户名和密码两者都为空，验证是否登录失败，并且提示信息正确；

用户名和密码两者之一为空，验证是否登录失败，并且提示信息正确；

如果登录功能启用验证码功能，在用户名和密码正确的前提下，输入正确的验证码，验证是否登录成功；

如果登录功能启用验证码功能，在用户名和密码正确的前提下，输入错误的验证码，验证是否登录失败，提示信息正确。

用户名和密码是否大小写敏感；

页面上的密码框是否加密显示；

后台系统创建的用户第一次登录成功时，是否提示修改密码；

忘记用户名和忘记密码的功能是否可用；

前端页面是否根据设计要求限制用户名和密码长度；

如果登录功能需要验证码，点击验证码图片是否可以更换验证码，更换后的验证码是否可用；

刷新页面是否会刷新验证码；

如果验证码具有时效性，需要分别验证时效内和时效外验证码的有效性；

用户登录成功但是会话超时后，继续操作是否会重定向到用户登录界面；

不同级别的用户，比如管理员用户和普通用户，登录系统后的权限是否正确；

页面默认焦点是否定位在用户名的输入框中；

快捷键 Tab 和 Enter 等，是否可以正常使用。-**-以上用例都是直接针对“用户登录”功能的功能性进行验证和测试的。**

**安全性测试用例包括：**

用户密码后台存储是否加密；

用户密码在网络传输过程中是否加密；

密码是否具有有效期，密码有效期到期后，是否提示需要修改密码；

不登录的情况下，在浏览器中直接输入登录后的 URL 地址，验证是否会重新定向到用户登录界面；

密码输入框是否不支持复制和粘贴；

密码输入框内输入的密码是否都可以在页面源码模式下被查看；

用户名和密码的输入框中分别输入典型的“SQL 注入攻击”字符串，验证系统的返回页面；

用户名和密码的输入框中分别输入典型的“XSS 跨站脚本攻击”字符串，验证系统行为是否被篡改；

连续多次登录失败情况下，系统是否会阻止后续的尝试以应对暴力破解；

同一用户在同一终端的多种浏览器上登录，验证登录功能的互斥性是否符合设计预期；

同一用户先后在多台终端的浏览器上登录，验证登录是否具有互斥性。

**性能压力测试用例包括:**

单用户登录的响应时间是否小于 3 秒；

单用户登录时，后台请求数量是否过多；

高并发场景下用户登录的响应时间是否小于 5 秒；

高并发场景下服务端的监控指标是否符合预期；

高集合点并发场景下，是否存在资源死锁和不合理的资源等待；

长时间大量用户连续登录和登出，服务器端是否存在内存泄漏。

兼容性测试用例包括：

不同浏览器下，验证登录页面的显示以及功能正确性；

相同浏览器的不同版本下，验证登录页面的显示以及功能正确性；

不同移动设备终端的不同浏览器下，验证登录页面的显示以及功能正确性；

不同分辨率的界面下，验证登录页面的显示以及功能正确性。

**在绝大多数的软件工程实践中，测试由于受限于时间成本和经济成本，是不可能去穷尽所有可能的组合的，而是采用基于风险驱动的模式，有所侧重地选择测试范围和设计测试用例，以寻求缺陷风险和研发成本之间的平衡。**

**这些问题：**

**1. 测试用例设计看似简单，却总是不够完备；**

**2. 软件的复杂度和更新迭代速度超乎想象，你需要寻找更高效的学习路径；**

**3. 自动化测试已成主流，但你的团队还停留在手工测试，困惑于怎么借助自动化测试工具提升团队效率；**

**4. 大量的测试框架和工具，让你不知所措，不知如何选择最适合自己项目的框架和工具；**

**5. 由 Google 等公司主导的“去 QE，开发自己测试”的模式，让你不清楚自己未来职业该如何规划；。**

**专栏共五大模块，系统阐述了测试工程师必须具备的核心测试理念和技能。**

**• 软件测试基础篇，将系统地讲解软件测试的基本概念、主流的测试方法和测试理念，为后续技术知识的学习打好基础。**

**• 软件测试技术篇，将包括 GUI 自动化测试、API 自动化测试、代码级测试和性能测试技术，通过理论结合实例的方式，为你讲述这四种测试技术的核心知识、设计思想，以及最佳实践。**

**• 测试架构篇，将讲述测试数据准备与测试基础架构的内容，既有解决测试数据准备痛点的方案，又有大型互联网企业测试基础架构设计的最佳实践。**

**• 测试新技术篇，将选取当下比较热门的探索式测试、测试驱动开发、精准测试、渗透测试技术和基于模型的测试，分别讲述它们解决的问题和具体的实现方法。**

**• 测试人员的互联网架构核心知识篇，将从高性能、高可用、伸缩性和可扩展性四个维度对大型网站架构进行深度剖析，弥补测试工程师相较与开发工程师以及测试架构师之间的鸿沟。**

**o 不仅需要从业务本身出发来对软件进行手工测试验证，还需要掌握完整的自动化测试开发技术来设计自动化测试用例.**

**o 作为测试工程师，你就必须掌握设计开发测试基础架构的关键技术。你必须系统性地思考如何才能将测试数据的准备工具化，服务化，最终实现平台化。**

**初级测试工程师：必须具有快速学习的能力，能迅速掌握被测软件的业务功能与内部架构，并在此基础上运用各种测试方法，尽可能多地发现潜在缺陷，并能够在已知缺陷的基础上进一步发现相关的连带缺陷。从知识体系上看，你需要有比开发人员更全面的计算机基础知识，还需要了解互联网的基础架构、安全攻击、软件性能、用户体验和常见缺陷等知识。从测试技术上看，你需要能够使用常见的测试框架或者工具，需要具有一定的自动化测试脚本的开发能力，这可以把你从大量重复的工作中解放出来，然后你才能有时间去做更有意思的工作。**

**中级：合格的测试工程师关注的是纯粹的测试，而优秀的测试工程师关注更多的是软件整体的质量，需要根据业务风险以及影响来制定测试策略，有效控制测试的时间和成本，并且能够对测试框架以及工具做出适合项目需求的选型. 其次，优秀的测试工程师不仅可以娴熟地运用各类测试工具，还非常清楚这些测试工具背后的实现原理，以及多个同类测试工具各自的优缺点和适用场景在遇到问题时，你还需要能够通过二次开发解决工具和框架层面的问题，对于没有合适可用工具的场景，可以自行设计开发一些小工具来更好地展开测试工作。这个阶段，你很有可能会接触到一些代码级的测试，这就要求你具有一定的开发背景，并能够很好地理解代码级的测试技术, 要求你理解自动化脚本的分层设计、页面对象模型以及业务流程模型，并且能够把这些设计应用到你的测试框架里。,**

**测试架构师：配置一套高效的能够支持高并发的测试执行基础架构，准备统一的测试数据准备平台，统一发起测试执行的接口，必须对一些前沿的测试方法和技术有自己的理解，并能够在恰当的时候、因地制宜地把它们应用到实际项目中。**

**“好的”测试用例一定是一个完备的集合，它能够覆盖所有等价类以及各种边界值，而跟能否发现缺陷无关**

**单元测试**

**单元测试**是指，对软件中的最小可测试单元在与程序其他部分相隔离的情况下进行检查和验证的工作，这里的最小可测试单元通常是指函数或者类。

单元测试通常由开发工程师完成，一般会伴随开发代码一起递交至代码库。单元测试属于最严格的软件测试手段，是最接近代码底层实现的验证手段，可以在软件开发的早期以最小的成本保证局部代码的质量。另外，单元测试都是以自动化的方式执行，所以在大量回归测试的场景下更能带来高收益.

**如何做好单元测试？**

要做好单元测试，你首先必须弄清楚单元测试的对象是代码，以及代码的基本特征和产生错误的原因，然后你必须掌握单元测试的基本方法和主要技术手段，比如什么是驱动代码、桩代码和 Mock 代码等。

驱动代码，桩代码和 Mock 代码，是单元测试中最常出现的三个名词。驱动代码是用来调用被测函数的，而桩代码和 Mock 代码是用来代替被测函数调用的真实代码的。必须严格根据代码的功能逻辑来设定，而不能通过阅读代码来推算预期输出，否则就是“掩耳盗铃”.

**移动应用测试方法与思路**

**移动端应用分为三大类：Web App、Native App 和 Hybrid App。**

**Web App 指的是移动端的 Web 浏览器，**其实和 PC 端的 Web 浏览器没有任何区别，只不过 Web 浏览器所依附的操作系统不再是 Windows 和 Linux ，而是 iOS 和 Android 。Web App 采用的技术主要是，传统的 HTML、JavaScript、CSS 等 Web 技术栈，加上 HTML5。另外，Web App 所访问的页面内容都是放在服务器端的，本质上就是 Web 网页，所以天生就是跨平台的。

**Native App 指的是移动端的原生应用，** 对于 Android 是 apk，对于 iOS 就是 ipa。Native App 是一种基于手机操作系统（iOS 和 Android），并使用原生程序编写运行的第三方应用程序。Android 使用的语言通常是 Java，iOS 使用的语言是 Objective-C。通常来说，Native App 可以提供比较好的用户体验以及性能，而且可以方便地操作手机本地资源。

**Hybrid App（俗称：混血应用），**是介于 Web App 和 Native App 两者之间的一种 App 形式。Hybrid App 利用了 Web App 和 Native App 的优点，通过一个原生实现的 Native Container 展示 HTML5 的页面。更通俗的讲法可以归结为，在原生移动应用中嵌入了 Webview，然后通过该 Webview 来访问网页。Hybrid App 具有维护更新简单，用户体验优异以及较好的跨平台特性，是目前主流的移动应用开发模式。

**从业务功能测试的角度看，移动应用的测试用例设计和传统 PC 端的 GUI 自动化测试策略比较类似，只是测试框架不同，数据驱动、页面对象模型和业务流程封装依旧适用；**

**对于移动应用，顺利完成全部业务功能测试往往是不够的。如果你的关注点只是业务功能测试，那么，当你的移动应用被大量用户安装和使用时，就会暴露出很多之前完全没有预料到的问题。于是我们从交叉事件测试、兼容性测试、流量测试、耗电量测试、弱网络测试、边界测试这 6 个最主要的专项测试来展开。**

**交叉事件测试也叫中断测试，**是指 App 执行过程中，有其他事件或者应用中断当前应用执行的测试。比如，App 在前台运行过程中，突然有电话打进来，注意，**此类测试目前基本还都是采用手工测试的方式，并且都是在真机上进行，不会使用模拟器。**

**交叉事件测试，需要覆盖的场景主要包括：**

多个 App 同时在后台运行，并交替切换至前台是否影响正常功能；

要求相同系统资源的多个 App 前后台交替切换是否影响正常功能，比如两个 App 都需要播放音乐，那么两者在交替切换的过程中，播放音乐功能是否正常；

App 运行时接听电话，App 运行时接收信息，App 运行时提示系统升级；，App 运行时发生系统闹钟事件；,App 运行时进入低电量模式,App 运行时第三方安全软件弹出告警；,App 运行时发生网络切换，比如，由 Wifi 切换到移动 4G 网络，或者从 4G 网络切换到 3G 网等；其实你可以发现，这些需要覆盖的场景，也是我们今后测试的测试用例集，每一场景都是一个测试用例的集合。

**兼容性测试**顾名思义就是，要确保 App 在各种终端设备、各种操作系统版本、各种屏幕分辨率、各种网络环境下，功能的正确性。常见的 App 兼容性测试往往需要覆盖以下的测试场景.

不同操作系统的兼容性，包括主流的 Andoird 和 iOS 版本；主流的设备分辨率下的兼容性；主流移动终端机型的兼容性；同一操作系统中，不同语言设置时的兼容性；同网络连接下的兼容性，比如 Wifi、GPRS、EDGE、CDMA200 等；在单一设备上，与主流热门 App 的兼容性，比如微信、抖音、淘宝等；

**兼容性测试，通常都需要在各种真机上执行相同或者类似的测试用例，所以往往采用自动化测试的手段。** 同时，由于需要覆盖大量的真实设备，除了大公司会基于 Appium + Selenium Grid + OpenSTF 去搭建自己的移动设备私有云平台外，其他公司一般都会使用第三方的移动设备云测平台完成兼容性测试。第三方的移动设备云测平台，国外最知名的是 SauceLab，国内主流的是 Testin。

**流量测试:**  App 经常需要在移动互联网环境下运行，而移动互联网通常按照实际使用流量计费，通常包含以下几个方面的内容：App 执行业务操作引起的流量；App 在后台运行时的消耗流量；App 安装完成后首次启动耗费的流量；App 安装包本身的大小；App 内购买或者升级需要的流量。

流量测试，往往借助于 Android 和 iOS 自带的工具进行流量统计，也可以利用 tcpdump、Wireshark 和 Fiddler 等网络分析工具。

对于 Android 系统，网络流量信息通常存储在 /proc/net/dev 目录下，也可以直接利用 ADB 工具获取实时的流量信息。另外推荐一款 Android 的轻量级性能监控小工具 **Emmagee**，类似于 Windows 系统性能监视器，能够实时显示 App 运行过程中 CPU、内存和流量等信息。

对于 iOS 系统，可以使用 Xcode 自带的性能分析工具集中的 Network Activity，分析具体的流量使用情况。

但是，流量测试的最终目的，并不是得到 App 的流量数据，而是要想办法减少 App 产生的流量。虽然，减少 App 消耗的流量不是测试工程师的工作，但了解一些常用的方法，也将有助于你的测试日常工作：

启用数据压缩，尤其是图片；使用优化的数据格式，比如同样信息量的 JSON 文件就要比 XML 文件小；遇到既需要加密又需要压缩的场景，一定是先压缩再加密；减少单次 GUI 操作触发的后台调用数量；每次回传数据尽可能只包括必要的数据；启用客户端的缓存机制；

**耗电量测试:** 耗电量测试通常从三个方面来考量：App 运行但没有执行业务操作时的耗电量；App 运行且密集执行业务操作时的耗电量；App 后台运行的耗电量。

耗电量检测既有基于硬件的方法，也有基于软件的方法。我所经历过的项目都是采用软件的方法，Android 和 iOS 都有各自自己的方法：Android 通过 adb 命令“adb shell dumpsys battery”来获取应用的耗电量信息；

iOS 通过 Apple 的官方工具 Sysdiagnose 来收集耗电量信息，然后，可以进一步通过 Instrument 工具链中的 Energy Diagnostics 进行耗电量分析。耗电测试中，Google推出的history batterian工具很好分析耗电情况.

**弱网络测试 :** 移动应用的测试需要保证在复杂网络环境下的质量。具体的做法就是：在测试阶段，模拟这些网络环境，在 App 发布前尽可能多地发现并修复问题。

**推荐一款非常棒的开源移动网络测试工具：Facebook 的 Augmented Traffic Control（ATC）。ATC 最好用的地方在于，它能够在移动终端设备上通过 Web 界面随时切换不同的网络环境，同时多个移动终端设备可以连接到同一个 Wifi，各自模拟不同的网络环境，相互之间不会有任何影响。也就是说，只要搭建一套 ATC 就能满足你所有的网络模拟需求**

**边界测试:**移动 App 在一些临界状态下的行为功能的验证测试，基本思路是需要找出各种潜在的临界场景，并对每一类临界场景做验证和测试。

主要的场景有：系统内存占用大于 90% 的场景；系统存储占用大于 95% 的场景；飞行模式来回切换的场景；App 不具有某些系统访问权限的场景，比如 App 由于隐私设置不能访问相册或者通讯录等；长时间使用 App，系统资源是否有异常，比如内存泄漏、过多的链接数等；出现 ANR 的场景；

操作系统时间早于或者晚于标准时间的场景；时区切换的场景；

**APP测试要点：**功能测试、兼容测试、流量测试、耗电量测试、性能测试、安全测试、网络测试、稳定性测试等。

**APP功能测试：主**要是依据需求规格和产品说明来验证各项功能，需要关注软件在正常和异常场景下的运行情况。

**UI测试:** (1）界面（菜单、结构、窗口、按钮）等是否满足需求，文字，图片，是否美观统一。(2）程序界面和操作是否友好、易用、易理解。

**安装卸载:** 验证App是否能正确安装、运行、卸载以及操作过程和操作前后对系统资源的使用情况。

**安装:**

1）软件安装后是否能够正常运行，安装目录和文件是否正常建立。

2）在不同系统版本和手机品牌下安装。

3）安装向导UI及功能是否正常。

4）安装过程中取消，下次安装是否正常。

5）安装过程来电，短信，通知，结束后是否继续安装。

6）是否支持覆盖安装。

7）安装空间不足时是否有相应提示。

8）安装后没有生成多余的目录结构和文件。

9）软件安装过程中关机重启，断电，断网的处理机制是否符合需求。

**卸载:**

1）直接卸载app是否有提示。

2）卸载后是否删除相应的安装目录。

3）卸载是否支持取消功能，单击取消后，是否正常可用。

4）卸载过程中死机，断电，重启等，手机恢复后能否正常卸载。

**登录运行:**

**登录:**

1）用户名和密码错误、漏填时，界面有提示信息。

2）自动登录时间失效后，启动app进入登录界面。

3）密码更改后，登录是否正常。

4）用户主动退出登录后，下次启动APP时，应该进入登录界面。

5）切换账号登录，检验登录的信息是否做到及时更新。

6）对于多个端都进行操作时，确保数据库操作无误，且每个端可以及时看到数据的更新。

7）登录界面切换到后台，或其他界面，或者锁屏是否影响登录过程。

8）ios与android设备登录同一个账号，用户数据是否同步。

9）已经登录的账号，再次登录是否强制退出。

10）登录超时的处理是否符合需求。

**运行:**

1)APP安装完成后，是否可以正常打开，是否有加载图示等。

2）APP的运行速度正常，切换是否流畅。

3）用户登录状态太久，sessionId会过期，会出现“虽然是登录状态，系统会提示用户没有登录。

**切换测试:**切换场景包括：app切换到后台、多个app之间切换。

1）app切换到或其他app或者系统界面，再回到app，是否停留在上一次操作的界面。

2）app切换到后台或其他app或者系统界面，再回到app，app是否正常使用。

3）当app使用过程中有电话进来中断后再切换到app，功能状态是否正常。

**消息推送:**设置开关打开状态下，消息推送是否可正常接收（应用启用中和应用关闭时都应该可以收到）

1）推送默认状态，一般默认开关应该是打开状态。

2）推送设置开关，存在“打开”，“关闭”选项。

3）开关打开时，可以收到消息推送，且点击可查看。

4）设置开关关闭时，客户端接收不到消息推送。

5）用户设置了免打扰的时间内，用户接收不到推送。在非免打扰时间段内，用户能正常收到推送。

6）检查推送消息内容与用户账号是否符合。

**升级更新:**

1）当app有更新版本时，手机端有更新提示。

2）当app版本为非强制升级版时，可以取消更新，旧版本能正常使用。用户在下次启动app时，仍出现更新提示。

3）当app版本为强制升级版时，给出强制更新后用户取消更新时，退出客户端。下次启动app时，仍出现强制升级提示。

4）当app有新版本时，直接更新检查是否能正常更新。

5）更新后，检查更app功能是否是新版本。

**中断测试**:app使用过程中突然来电、短信弹出、闹钟、QQ聊天信息、微信、低电量等提示时能否正常使用。

1）当app使用过程中有电话进来中断后再切换到app，功能状态是否正常。

2）当杀掉app进城后，再开启app，app能否正常启动。

3）出现必须处理的提示框后，切换到后台，再切换回来，检查提示框是否还存在，有时候会出现应用自动跳过提示框的缺陷。

**网络测试:**目前手机手机接入的网络主要分为2G、3G、4G、wifi。

1) 无网络时，有切换网络的操作或者提示。

2）网络间切换app断网有相应提示，重新联网后正常使用。

3) 在网络信号不好时，检查功能状态是否正常，确保不因提交数据失败而造成crash。

4) 在网络信号不好时，检查数据是否会一直处于提交中的状态，有无超时限制。如遇数据交换失败时要给予提示。

5) 弱网络下操作是否有提示。

**兼容性测试:** 1）操作系统版本的兼容性（Android各个版本，ios各个版本）2）不同手机品牌的兼容性。

3）手机分辨率兼容性4）网络的兼容性：2G\3G\4G\WIFI,弱网下、断网时5）app跨版本的兼容性。

6）与其他app的兼容性。

**权限测试:** 当权限没有开启时，或友好提示是否允许设置，当允许开启时，跳转到设置界面。

1）有限制允许接入网络提示或选项。

2）有限制允许读写通讯录、用户数据提示或选项。

3）有限制允许相机提示或选项。

4）有限制允许录音功能提示或选项。

5）有限制允许定位功能提示或选项。

**其他手机端特性测试:**1）关机、待机后app能否正常使用。2）手机解锁屏幕后进入进入app是否正常。

3）app在清空数据或强制退后还能正常运行。4） 长时间开机app开启情况下是否会出现异常情况。

5）app运行时关机重启。6）app运行时充电。

**Web端**

**WEB功能测试：** UI及易用性测试、表单测试、cookies测试、链接测试、兼容性测试。

**UI及易用性测试：**

　1）各个页面的样式风格是否美观统一，如图片大小、颜色是否统一，页面、文字、图片是否居中等。

　2）各个页面的标题和描述是否正确，有无错别字，字体大小、颜色是否正确统一，文字描述准确，无歧义。

　3）页面布局统一，美观，间距合理。

　4）操作是否符合人们的常规习惯（有没有把相似的功能的控件放在一起，方便操作）。

　5）提示信息是否正确，鼠标停留到上面是否正常显示提示。

　6）调整分辨率验证页面格式是否错位现象。

　7）窗口的最大化、最小化是否能正确切换。

8）执行风险操作时，有确认、删除等提示。

9）快捷键检查：是否支持常用快捷键，如Ctrl+C Ctrl+V Backspace等。

**cookies测试：**cookies是否正常工作。刷新操作是否影响cookies。cookies是否按预定时间保存。

**表单测试：**表单测试主要是验证对数据的增删改查修改是否正常实现，以及验证码功是否可用。

（1）、注册、登陆、输入信息提交等操作是否正常。

（2）、用户填写的信息是否合理，是否在需求规定的范围内，对于一起日期时间地点等选择是否合理；

（3）、检验默认值的正确性；

（4）、如表单只能接受指定的某些值，测试时跳过这些字符，看系统是否会报错。

（5）、短信验证码、邮箱验证码、字符验证码、图片验证码功能是否正常。

**链接测试：**

（1）、测试所有链接是否按指示的那样确实链接到了该链接的页面；

（2）、测试所链接的页面是否存在；

（3）、保证Web应用系统上没有孤立的页面(所谓孤立页面是指没有链接指向该页面，只有知道正确的URL地址才能访问)。

**兼容性测试：**

1）平台兼容性：不同硬件平台（PC、手机、平板等），不同操作系统（linux、windows、macOS、android、ios等）。

2）浏览器兼容性（IE、360、搜狗、chrome、火狐等）。

**软件缺陷报告**

**缺陷报告本身的质量将直接关系到缺陷被修复的速度以及开发工程师的效率，同时还会影响测试工程师的信用、测试与开发人员协作的有效性.**

**一份好的缺陷报告需要包括哪些具体内容呢？**

**测试缺陷统计，bug趋势分析，bug收敛情况，bug模块分布，bug发现阶段统计等等，都属于面向管理层和质量流程保障团队的，这类报告通常不是人为去产生的，而是在现在缺陷报告的基础上通过统计得到的，往往都是直接利用缺陷管理系统的自带自带功能来生成这类报告。比如ALM，tapd ,Jira, Bugzilla、BugFree 和 Mantis都支持这类报告的产生。当使用这类系统递交缺陷时，会自动生成模板，你只要按照其中的必填字段提供缺陷的详细信息就可以了.**

一份高效的缺陷报告主要由**缺陷标题,** **缺陷概述,** **缺陷影响,** **环境配置,** **前置条件,** **缺陷重现步骤,** **期望结果和实际结果,** **优先级（Priority）和严重程度（Severity）,** **变通方案（Workaround）,** **根原因分析（Root Cause Analysis）**

**,** **附件（Attachment）,** **总结.**

**缺陷标题:**是对缺陷的概括性描述，通常采用“在什么情况下发生了什么问题”的模式。对“什么问题”的描述要简洁与具体,楚地表述发生问题时的上下文，也就是问题出现的场景。其次，标题应该尽可能描述问题本质，而避免只停留在问题的表面。最后，缺陷标题不易过长，对缺陷更详细的描述应该放在“缺陷概述”里。

**缺陷概述:**通常会提供更多概括性的缺陷本质与现象的描述，是缺陷标题的细化。用清晰简短的语句将问题的本质描述清楚是关键。

**缺陷影响:** 缺陷引起的问题对用户或者对业务的影响范围以及严重程度. 缺陷影响决定了缺陷的优先级（Priority）和严重程度（Severity），开发经理会以此为依据来决定修复该缺陷的优先级；而产品经理会以此为依据来衡量缺陷的严重程度，并决定是否要等该缺陷被修复后才能发布产品。

**环境配置:** 用以详细描述测试环境的配置细节，为缺陷的重现提供必要的环境信息.

**前置条件:** 测试步骤开始前系统应该处在的状态，其目的是减少缺陷重现步骤的描述。合理地使用前置条件可以在描述缺陷重现步骤时排除不必要的干扰，使其更有针对性. 比如，某个业务操作需要先完成用户登录，你在缺陷重现步骤里就没有必要描述登录操作的步骤细节，可以直接使用 “前置条件：用户已完成登录”的描述方式；

**缺陷重现步骤:** 缺陷报告中最核心的内容，其目的在于用简洁的语言向开发工程师展示缺陷重现的具体操作步骤。采用步骤列表的表现形式.

**期望结果和实际结果:** 期望结果和实际结果通常和缺陷重现步骤绑定在一起，在描述重现步骤的过程中，需要明确说明期待结果和实际结果。期待结果来自于对需求的理解，而实际结果来自于测试执行的结果。

**优先级（Priority）和严重程度（Severity）:** 严重程度是缺陷本身的属性，通常确定后就不再变化，而优先级是缺陷的工程属性，会随着项目进度、解决缺陷的成本等因素而变动.

**变通方案（Workaround）:** 提供一种临时绕开当前缺陷而不影响产品功能的方式，通常由测试工程师或者开发工程师完成，或者他们一同决定。

**根原因分析（Root Cause Analysis）:** 根原因分析就是我们平时常说的 RCA，如果你能在发现缺陷的同时，定位出问题的根本原因，清楚地描述缺陷产生的原因并反馈给开发工程师，那么开发工程师修复缺陷的效率就会大幅提升，而且你的技术影响力也会被开发认可。

**附件（Attachment）:** 通常是为缺陷的存在提供必要的证据支持，常见的附件有界面截图、测试用例日志、服务器端日志、GUI 测试的执行视频等。

**测试计划**

**一份成功的测试计划应该包含哪些内容，以及如何才能做好测试计划。**

**如果没有测试计划，会带来哪些问题呢？**

很难确切地知道具体的测试范围，以及应该采取的具体测试策略；很难预估具体的工作量和所需要的测试工程师数量，同时还会造成各个测试工程师的分工不明确，引发某些测试工作被重复执行而有些测试则被遗漏的问题；测试的整体进度完全不可控，甚至很难确切知道目前测试的完成情况，对于测试完成时间就更难预估准确的时间节点了；整个项目对潜在风险的抵抗能力很弱，很难应对需求的变更以及其他突发事件。

**逆向思维推导出，一份好的测试计划要包括：测试范围、测试策略、测试资源、测试进度和测试风险预估，这五大方面，并且每一部分都要给出应对可能出现问题的解决办法。**

**测试范围:** 测试范围描述的是被测对象以及主要的测试内容。比如，对于用户登录模块，功能测试既需要测试浏览器端又需要测试移动端，同时还考虑登录的安全和并发性能相关的非功能性需求的测试等。**测试范围中需要明确“测什么”和“不测什么”。**

**测试策略:** 需要明确“先测什么后测什么”和“如何来测”这两个问题。采用什么样的测试类型和测试方法。 这里需要注意的是，不仅要给出为什么要选用这个测试类型，还要详细说明具体的实施方法。

**对于功能测试，**根据测试需求分析的思维导图来设计测试用例。主线业务的功能测试由于经常需要执行回归测试，所以你需要考虑实施自动化测试，并且根据项目技术栈和测试团队成员的习惯与能力来选择合适的自动化测试框架。通常应该先实现主干业务流程的测试自动化。

实际际操作时，你通常需要先列出主要的功能测试点，并决定哪些测试点适合采用自动化测试，并且决定具体使用什么样的框架和技术。对于需要手工测试的测试点，你要决定采用什么类型的测试用例设计方法，以及如何准备相关的测试数据。

另外，你还要评估被测软件的可测试性，如果有可测试性的问题，需要提前考虑切实可行的变通方案，甚至要求开发人员提供可测试性的接口。

**对于兼容性测试来说，**Web 测试需要确定覆盖的浏览器类型和版本，移动设备测试需要确定覆盖的设备类型和具体 iOS/Android 的版本等。兼容性测试用例的选取，往往来自于已经实现的自动化测试用例。道理很简单，因为兼容性测试往往要覆盖最常用的业务场景，而这些最常用的业务场景通常也是首批实现自动化测试的目标。

**对于性能测试，**需要在明确了性能需求（并发用户数、响应时间、事务吞吐量等）的前提下，结合被测系统的特点，设计性能测试场景并确定性能测试框架。比如，是直接在 API 级别发起压力测试，还是必须模拟终端用户行为进行基于协议的压力测试。再比如，是基于模块进行压力测试，还是发起全链路压测。**最后，无论采用哪种方式，都需要明确待开发的单用户脚本的数量，以便后续能够顺利组装压测测试场景。性能测试的实施首先需要根据你想要解决的问题，确定性能测试的类型；然后，根据具体的性能测试类型开展测试。**

1. 性能测试的实施，往往先要根据业务场景来决定需要开发哪些单用户脚本，脚本的开发会涉及到很多性能测试脚本特有的概念，比如思考时间、集合点、动态关联等等。
2. 脚本开发完成后，你还要以脚本为单位组织测试场景（Scenario），场景定义简单来说就是百分之多少的用户在做登录、百分之多少的用户在做查询、每个用户的操作步骤之间需要等待多少时间、并发用户的增速是 5 秒一个，还是 5 秒 2 个等等。
3. 最后，才是具体的测试场景执行。和自动化功能测试不同，性能测试执行完成后性能测试报告的解读，是整个测试过程中最关键的点。

还有很多测试类型（比如，接口测试、集成测试、安全测试、容量验证、安装测试、故障恢复测试等）。这些测试类型，都有各自的应用场景，也相应有独特的测试方法，就不再一一展开了

**测试资源:** 测试资源通常包括测试人员和测试环境，这两类资源都是有限的。测试计划的目的就是，保证在有限资源下的产出最大化。所以，测试资源就是需要明确“谁来测”和“在哪里测”这两个问题。

**测试进度:** 描述各类测试的开始时间，所需工作量，预计完成时间，并以此为依据来建议最终产品的上线发布时间。

**比如，版本接受测试（Build Acceptance Test）的工作量，冒烟测试（Smoke Test）的工作量，自动化脚本开发的工作量，缺陷修复的验证工作量，需要几轮回归测试、每一轮回归测试的工作量等等。**

**测试风险预估:** 通常需求变更、开发延期、发现重大缺陷和人员变动是引入项目测试风险的主要原因。对于需求变更，比如增加需求、删减需求、修改需求等，一定要重新进行测试需求分析，确定变更后的测试范围和资源评估，并与项目经理和产品经理及时沟通因此引起的测试进度变化。测试经理 / 测试负责人切忌不能有自己咬牙扛过去的想法，否则无论是对测试团队还是对产品本身都不会有任何好处**。**

**测试覆盖率**

**测试覆盖率通常被用来衡量测试的充分性和完整性，测试覆盖率分为两大类，一类是面向项目的需求覆盖率，另一类是更偏向技术的代码覆盖率。**

**需求覆盖率:**指测试对需求的覆盖程度，通常的做法是将每一条分解后的软件需求和对应的测试建立一对多的映射关系，最终目标是保证测试可以覆盖每个需求，以保证软件产品的质量。通常采用 ALM，Doors 和 TestLink 等需求管理工具来建立需求和测试的对应关系，并以此计算测试覆盖率。

互联网测试项目中很少直接基于需求来衡量测试覆盖率，而是将软件需求转换成测试需求，然后基于测试需求再来设计测试点。因此，现在人们口中的测试覆盖率，通常默认指代码覆盖率，而不是需求覆盖率。

代码覆盖率的局限性主要是不能发现有需求但是代码没有实现的场景，但是对于已有代码还是具有很高参考价值的，至少知道你那些分支和语句覆盖到了

**代码覆盖率:** 至少被执行了一次的条目数占整个条目数的百分比.

**常见的三种代码覆盖率指标:**

行覆盖率又称为语句覆盖率，指已经被执行到的语句占总可执行语句（不包含类似 C++ 的头文件声明、代码注释、空行等等）的百分比。这是最常用也是要求最低的覆盖率指标。实际项目中通常会结合判定覆盖率或者条件覆盖率一起使用。

判定覆盖又称分支覆盖，用以度量程序中每一个判定的分支是否都被测试到了，即代码中每个判断的取真分支和取假分支是否各被覆盖至少各一次。比如，对于 if(a>0 && b>0)，就要求覆盖“a>0 && b>0”为 TURE 和 FALSE 各一次。

条件覆盖是指，判定中的每个条件的可能取值至少满足一次，度量判定中的每个条件的结果 TRUE 和 FALSE 是否都被测试到了。比如，对于 if(a>0 && b>0)，就要求“a>0”取 TRUE 和 FALSE 各一次，同时要求“b>0”取 TRUE 和 FALSE 各一次。

**统计代码覆盖率的根本目的是找出潜在的遗漏测试用例，并有针对性的进行补充，同时还可以识别出代码中那些由于需求变更等原因造成的不可达的废弃代码。高的代码覆盖率不一定能保证软件的质量，但是低的代码覆盖率一定不能能保证软件的质量。**

**代码覆盖率工具:**

**JaCoCo** 是一款 Java 代码的主流开源覆盖率工具，可以很方便地嵌入到 Ant、Maven 中，并且和很多主流的持续集成工具以及代码静态检查工具，比如 Jekins 和 Sonar 等，都有很好的集成。

 C/C++，JavaScript 等语言的代码覆盖率工具，比如 GCC Coverage、JSCoverage 和 Istanbul 等.

**代码覆盖率工具的实现原理:**

实现代码覆盖率的统计，最基本的方法就是注入（Instrumentation）。简单地说，注入就是在被测代码中自动插入用于覆盖率统计的探针（Probe）代码，并保证插入的探针代码不会给原代码带来任何影响。对于 Java 代码来讲，根据注入目标的不同，可以分为源代码（Source Code）注入和字节码（Byte Code）注入两大类。基于 JVM 本身特性以及执行效率的原因，目前主流的工具基本都是使用字节码注入，注入的具体实现采用 ASM 技术。根据注入发生的时间点，字节码注入又可以分为两大模式：On-The-Fly 注入模式和 Offline 注入模式。

**测试报告**

**项目概述：**XXXX平台是一款为政府企事业单位提供音视频业务管理平台，XXXXXXXXX,为政府企事业单位应急处突、活动安保、抗险救灾等提供有力保障。

**测试目标：**

**测试背景：**本次测试需求来源为，目的为了实现XXXXX，验证同时在线1000人在线下。

**接口性能指标：**每秒事务数（TPS）（寻找系统支持最大TPS） 响应时间小于2s 服务器CPU：平均利用率小于70%，瞬时峰值不高于80% 服务器内存：平均利用率小于70% 磁盘IO：平均利用率小于70%

并发用户数 事务处理成功率>=99%，若有任何错误，需给出错误具体原因； 带宽利用率：根据业务需求给出测试结果，无具体指标要求；

**测试范围：**接口范围，业务场景

**测试环境：**硬件配置与软件配置，测试拓扑

**测试策略：**测试策略，负载测试，稳定性测试

**测试数据：**性能指标数据，测试图表

**测试结论及风险分析：**结果分析，风险分析及建议，测试结论

**测试工程师的岗位要求与招聘**

“作为测试人员，必须要深入理解业务，但是业务知识不能等同于测试能力。”

“测试开发岗位的核心其实是“测试”，“开发”的目的是更好地服务于测试，我们看重的是对测试的理解，以及在此基础上设计、开发帮助测试人员提高效率并解决实际问题的工具，而不是一个按部就班、纯粹意义上的开发人员。”

**传统测试工程师师应该具备的核心竞争力**

**测试策略设计能力、测试用例设计能力、快速学习能力、探索性测试思维、缺陷分析能力、自动化测试技术和良好的沟通能力。**

**测试策略设计能力是指，**对于各种不同的被测软件，能够快速准确地理解需求，并在有限的时间和资源下，明确测试重点以及最适合的测试方法的能力。具备出色的测试策略设计能力，你可以非常明确地回答出测试过程中遇到的这些关键问题：

1.测试要具体执行到什么程度；2.测试需要借助于什么工具；3.如何运用自动化测试以及自动化测试框架，以及如何选型；4.测试人员资源如何合理分配；5.测试进度如何安排；5.测试风险如何应对。

**测试用例设计能力,** 无论对于什么类型的测试，都能设计出高效地发现缺陷，保证产品质量的优秀测试用例。需要深入理解被测软件的业务需求和目标用户的使用习惯，还要熟悉软件的具体设计和运行环境，包括技术架构、缓存机制、中间件技术、第三方服务集成等平时就要多积累，对常见的缺陷模式、典型的错误类型以及遇到过的缺陷，要不断地总结、归纳，才能逐渐形成体系化的用例设计思维。

**快速学习能力,** 对不同业务需求和功能的快速学习与理解能力, 对于测试新技术和新方法的学习与应用能力.

**快速学习能力，乍一看是比较难培养的，但其实也有一些小窍门。比如，当你学习一个新的开源工具时，建议你直接看官方文档：一来，这里的内容是最新而且是最权威的；二来，可以避免网上信息质量的参差不齐。知识输入源头是单一，而且权威的话，你的学习曲线也必然会比较平滑。另外，当学习新内容时，你一定要做到理解其原理，而不是只停留在表面的、简单的操作和使用，长期保持这种学习状态，可以在很大程度上提高逻辑思维和理解能力。这样，当你再面对其他新鲜事物时候，也会更容易理解，形成良性循环。**

**探索性测试思维:**测试工程师在执行测试的过程中不断学习被测系统，同时结合基于自己经验的错误猜测和逻辑推理，整理和分析出更多的有针对性的测试关注点。精准测试最通俗的理解可以概括为针对开发代码的变更，目标明确并且有针对性地对变更点以及变更关联点做测试，这也是目前敏捷测试主推的测试实践之一。

**缺陷分析能力:** **缺陷分析能力，通常包含三个层面的含义：**

对于已经发现的缺陷，结合发生错误的上下文以及后台日志，可以预测或者定位缺陷的发生原因，甚至可以明确指出具体出错的代码行，由此可以大幅缩短缺陷的修复周期，并提高开发工程师对于测试工程师的认可以及信任度；

根据已经发现的缺陷，结合探索性测试思维，推断同类缺陷存在的可能性，并由此找出所有相关的潜在缺陷；

可以对一段时间内所发生的缺陷类型和趋势进行合理分析，由点到面预估整体质量的健康状态，并能够对高频缺陷类型提供系统性的发现和预防措施，并以此来调整后续的测试策略。

**自动化测试技术:** 一定的写代码的能力.

**良好的沟通能力:**沟通好团队.

**测试开发工程师的核心竞争力:**

除了代码开发能力，测试开发工程师更要具备测试系统需求分析的能力。你要能够站在测试架构师的高度，识别出测试基础架构的需求和提高效率的应用场景。从这个角度说，你更像个产品经理，只不过你这个产品是为了软件测试服务的。

测试开发工程师需要具备非常宽广的知识体系，你不仅需要和传统的测试开发工程师打交道，因为他们是你构建的测试工具或者平台的用户；而且还要和 CI/CD、和运维工程师们有紧密的联系，因为你构建的测试工具或者平台，需要接入到 CI/CD 的流水线以及运维的监控系统中去。

了解更高级别的测试架构部署和生产架构部署、你还必须对开发采用的各种技术非常熟悉。

**自我介绍的的核心逻辑：**

1. 说清楚我是谁。第二，我做过什么，有什么成就。第三，我能给公司带来什么，跟招聘岗位有什么契合点。

您好，我叫XXX，毕业于XXX，在XXX、XXX两家公司工作过，目前担任XXX工程师，负载XXX、XXX产品的开发/测试工作，擅长性能测试、接口测试……，希望能够加入贵公司，担任XXX方向的工作。