**软件测试工程师的技能树**

发表于：2019-1-02 11:18  作者：Batkid2018   来源：掘金

[**软件测试**](javascript:;)工程师是一个历史很悠久的职位，可以说从有[**软件开发**](javascript:;)这个行业以来，就开始有了软件测试工程师的角色。随着时代的发展，软件测试工程师的角色和职责也在悄然发生着变化，从一开始单纯的在瀑布式开发流程中担任测试阶段的执行者，到敏捷开发流程中QA（Quality Assurance）角色，为整个团队和产品的质量负责，测试工程师的职责和边界不断的扩大。近年来**[互联网](javascript:;" \t "_self)**行业的很多测试工程师被称为是测试开发工程师，也就是要具备[**自动化测试**](javascript:;)和测试工具开发能力的测试工程师，可以说是对测试工程师的能力要求达到了一个新的高度。

　　相信有过测试工作经验的同学都会深有体会，不管是瀑布式还是agile模式，测试人员的工作总是被压在产品发布的最后阶段，整个团队的压力似乎都压在测试工程师身上，没有人会理会开发过程中产生的延误，因为那已经过去，可以在retro meeting的时候diss，但是目前最重要的问题是完成产品的发布上线。所以在寻找测试工程师需要什么技能之前，测试工程师的核心问题是什么，这是我们要搞清楚的。

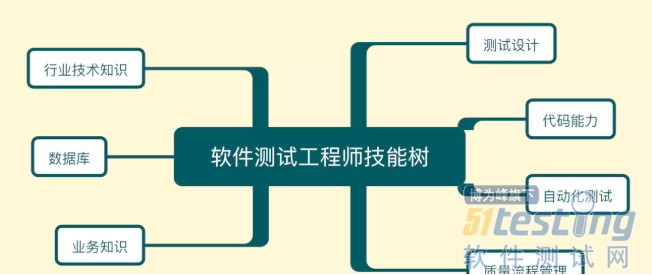
**测试工程师面临的核心问题**

　　如何以最小的投入，最大程度保证产品的质量

　　这个问题相信大家都有所体会，商业社会追求的就是效率，甚至是极致的效率。测试工程师也不能例外，不管是叫测试工程师，QA，或者是听着高大上的测试开发工程师，其实老板们的目标是一致的，就是在尽可能少的投入，最大程度保证产品的质量。说得现实一点，你的薪资水平就取决于你能解决这个核心问题的能力。

　　明确了我们的目标，我们所需要的能力，也是围绕着这一个目标来设定的。

**概述**

[](http://www.51testing.com/batch.download.php?aid=96729)

　　按照笔者的经验和理解，一个软件测试工程师需要具备以下的技能：

　　测试设计能力

　　代码能力

　　自动化测试**[技术](javascript:;" \t "_self)**

　　质量流程管理

　　行业技术知识

[**数据库**](javascript:;)

　　业务知识

**测试设计**

　　作为一名测试工程师，最基础的能力应该就是根据产品来设计[**测试用例**](javascript:;)的能力。最基础的能力往往也是最难做到精通的能力。要设计好的测试用例，需要对产品的特性和业务非常的熟悉，对用户的使用场景有着系统化的思考。除此之外，还有一些科学的测试用例设计方法可以帮助我们设计规范化的用例，而不是仅仅根据经验或者天马行空的想法来设计用例。

　　业界有一些经典的测试用例设计方法需要测试工程师掌握：

　　边界值分析

　　等价类划分

　　因果图

　　判定表

　　正交实验设计

　　上述的这些方法并不是教条，而是帮助我们理清测试用例设计的思路和提高效率的工具。

**代码能力**

　　在传统的思维中，对测试人员的代码能力要求似乎不是很高，在业界确实也是这样的。很多测试工程师基本上不具备代码的能力，更多是测试的执行者。

　　但是在当今这个时代下，要想突破传统[**功能测试**](javascript:;)人员的天花板，代码能力是必须的。

　　具备代码能力的测试工程师有这样两个优势：

**阅读开发代码**

　　如果能够具备阅读开发代码的能力，对于提高测试人员的效率是很有帮助的，它可以帮助我们做到这些一些事情

　　通过开发修改的代码预估影响的范围，即测试的范围

　　参加技术评审，预估测试的风险，难点，重点

　　通过代码的逻辑设计测试用例，强化测试用例的覆盖程度

　　对缺陷进行初步的定位

　　其实可以做到的事情还有很多，体现在测试过程的很多细节当中

**自动化测试的开发**

　　自动化测试是测试发展的方向，也是提高效率的有效方法。具备了代码能力，你可以轻松的驾驭各种流行的自动化测试框架和用例开发。

**自动化测试**

　　接着上面关于自动化测试的讨论。在目前的热门公司的招聘中，自动化能力已经是必备的能力，也是大家很关注的一个领域。

　　目前可以粗略的把自动化测试分为这么几类：

**UI自动化**

　　UI自动化实现的目标是模拟人在产品UI界面上的操作，从而观察结果来完成测试的执行。UI自动化也可以从客户端的形态上分为PC端和移动端的自动化测试，有这样一些著名的自动化工具需要我们掌握：

[**Selenium**](javascript:;)

　　Selenium是一个很经典的WEB端产品的UI自动化工具，针对不同的开发语言都有很好的支持。它的原理简单来说就是通过WebDriver把脚本产生的操作指令传递到**[浏览器](javascript:;" \t "_self)**，执行我们需要的操作并且获取相应的反馈，在脚本中完成校验。

**Appium**

　　从这个名字就可以看出这个工具和Selenium的相似之处。其实Appium可以理解为就是移动端的Selenium。同样也是在移动端模拟人的操作来实现执行测试用例的目的。

　　随着[**移动互联网**](javascript:;)时代的到来，更多的业务已经从PC的WEB端转移到了移动端，移动端的自动化测试越来越重要。

　　其实UI的自动化实现的原理都是很类似的，基本的逻辑都是：

　　定位元素

　　操作元素

　　获取反馈

　　最后通过某种测试用例框架来管理测试用例，例如**[python](javascript:;" \t "_self)**的unittest，JAVA的**[TestNG](javascript:;" \t "_self)**，**[Ruby](javascript:;" \t "_self)**的respec等等。

　　所以说了解了某一种UI自动化的框架和工具，很容易的就能触类旁通的学习新的框架和工具。

**接口自动化**

　　在目前SaaS成为主流的情况下，API，即接口，成为了支撑业务的核心部分。前端页面和App里面的业务数据都是通过各种API与服务器进行通信，从而实现业务功能。

　　目前大多数的接口都是基于HTTP协议的，其中Restful的接口又占大多数。而很多语言，例如Python和Ruby都有很好的库来支持HTTP协议的请求，这就为我们设计接口自动化提供了很好的基础。

　　回到我们的核心问题，投入产出比的衡量。UI的自动化无论是从实现的成本还是维护的成本来说都是巨大的，所以业界越来越把重心放到了接口层的自动化实现上。

　　接口的自动化具备这样的优势：

　　运行效率高

　　开发成本低

　　维护成本低

　　可以与开发代码同步开发

　　接口自动化的实现思路也是简单明了的，那就是模拟浏览器，发送HTTP请求来实现对接口的调用，然后比较返回与期望值，达到验证结果的目的。

　　当然，要设计一套真正高效的接口自动化框架也是不容易的。这里面涉及到如何提高用例的开发效率，降低开发维护成本等关键问题。同时还可以把**[接口测试](javascript:;" \t "_self)**与[**性能测试**](javascript:;)结合起来，丰富接口自动化测试的内涵。

[**质量管理**](javascript:;)**流程**

　　在敏捷开发的流程中，测试工程师有了一个新的定义：Quality Assurance Engineer。而测试的执行仅仅是职责中的一部分，更为重要的是要为整个团队的产品质量负责。

　　从整个sprint的周期来看，QA工程师都要始终如一的贯彻质量保证的意识，与开发的关系也从早期的发现bug，转变为如何帮助开发团队一起提高产品的质量。同时还要和产品团队密切的合作，在需求的分析阶段就介入，分析质量保证工作如何规划和设计，而不是在产品发布前的测试执行阶段才介入。

　　这个里面还包含很多Soft skill的要求，包括如何与团队合作，沟通等等，这也是敏捷开发模式的关键之一。

**行业技术知识**

　　这一部分内容其实涵盖的内容是非常丰富的，就以互联网行业举例吧。

　　对于一个互联网产品，测试工程师需要了解的甚至是精通的知识是很多的，从前端页面的技术栈，API的设计，后端服务器的设计，后面会专门提到的数据库，还有整个服务的架构等等，测试工程师都需要有所了解。

　　针对这个问题，其实有一个非常好的问题可以帮助大家去梳理涉及到的知识，这就是：

　　从在浏览器的输入框输入一个网址，到看到网页的内容，这个过程中发生了什么？

　　回答这个问题的深度和广度，基本就能反映一个测试工程师对于互联网产品技术的掌握情况。

　　在这里呢，我简单的罗列一些涉及到的技术和概念，这些内容对于我们测试产品，都是非常有帮助的。

　　DNS

　　TCP/IP

　　HTTP

　　SSL

　　Restful

　　HTML

　　DOM

　　CSS

　　Render

　　Xpath

　　服务器

　　nginx

[**SQL**](javascript:;)

　　Cookie&Session

　　XSS，CSRF

　　这里仅仅是涉及到一部分内容，具体的内容可以根据工作中遇到的场景去深入学习和了解。

**数据库**

　　之所以把数据库单独列出来，是因为数据库的知识对于当今的很多产品都是非常核心的内容。

　　不管是在手动测试还是自动化测试中，都有需要到数据库进行数据校验的时候。

　　目前主要使用的数据库可以分为两类：

　　关系型数据库

　　非关系型数据库

**关系型数据库**

　　关系型数据库是最常见的数据库类型，这类数据库通过RDBMS数据库程序来进行管理和使用，常见的有SQL Server, [**MySQL**](javascript:;)等等。

　　关系型数据库中强调一个事务（Transaction）的概念。所谓事务是用户定义的一个数据库操作系列，这些操作要么全部执行，要么全部不执行，是一个不可分割的工作单位。例如在关系数据库中，一个事务可以是一条SQL语句、一组SQL语句或整个程序。

　　事务应该具有4个属性：原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

　　原子性（Atomicity）：事务作为一个整体被执行，包含在其中的对数据库的操作要么全部被执行，要么都不执行。

　　一致性（Consistency）：事务应确保数据库的状态从一个一致状态转变为另一个一致状态。一致状态的含义是数据库中的数据应满足完整性约束。

　　隔离性（Isolation）：多个事务并发执行时，一个事务的执行不应影响其他事务的执行。

　　持久性（Durability）：一个事务一旦提交，他对数据库的修改应该永久保存在数据库中。

　　对于实际的应用来说，SQL语言是必须要掌握的。能够通过SQL语句在数据库中找到需要的数据，是测试工程师必备的技能。SQL语句的语法大体上比较类似，在一些细节上不同的RDBMS会有些许的差别。

　　对于自动化实现来说，在自动化测试中通过访问数据库来获得期望值也是很常见的场景。不同的语言都有访问数据库的库，整体来说应用也很简单。

**非关系型数据库**

　　随着互联网中大量的非结构化数据的产生，例如社交网络等等应用，用户的个人信息，社交网络，地理位置，用户生成的数据和用户操作日志已经正在以几何级数的速率增加，同时还面临大量的数据挖掘工作，传统的关系型数据库已经无法满足。所以NoSQL渐渐的发展了起来。

　　NoSQL最突出的特点就是数据的非结构化，通俗的讲，就是数据不再是以列和行这样的形式存储的。

　　NoSQL存储数据的方式很多：值对存储，列存储，文档存储。

　　例如比较常见的MongoDB就是将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。

**RDBMS vs NoSQL**

**RDBMS**

　　高度组织化结构化数据

　　结构化查询语言（SQL） (SQL)

　　数据和关系都存储在单独的表中。

　　数据操纵语言，数据定义语言

　　严格的一致性

　　基础事务

**NoSQL**

　　代表着不仅仅是SQL

　　没有声明性查询语言

　　没有预定义的模式：键 - 值对存储，列存储，文档存储，图形数据库

　　最终一致性，而非ACID属性

　　非结构化和不可预知的数据

　　CAP定理

　　高性能，高可用性和可伸缩性

**业务知识**

　　对于测试工程师来说，所测试产品的业务知识也是非常重要的。

　　一个测试工程师可能已经具备了上述的所有技能，但是怎么把这些技能用来解决我们最先提到的软件测试的核心问题呢？这个里面的关键，或者说中心点，就是你所测试的产品的业务。

　　测试的方法，规划，实施方法都是多种多样的，如果在这些方法中进行选择，所依赖的正是对产品的业务的深刻理解。

　　这里的产品业务不仅仅指产品的特性，同时还包括了产品的用户特征，用户的使用习惯，以及由此带来的对产品的流量趋势。也可以说，测试人员必须要站在用户的角度来分析产品，而不是产品开发人员的角度。

　　测试人员还需要找到产品的核心功能和核心业务，通过这样的分析来进行测试优先级的划分，以及缺陷的定级。同时对于自动化测试的规划和架构也有着重要的影响。例如在自动化测试中要首先覆盖那些核心的业务和功能，同时根据业务的特性，用自动化的方法去模拟用户的使用场景，把有限的自动化资源投入到最关键的部分。

　　这一块技能听起来可能很虚，好像没有什么具体的知识点，但是在不断的工作和总结中，优秀的测试工程师是能够总结出一套符合某一类产品的测试方法的，甚至还可以提炼出一些更具备通用性的best practice，用到不同的产品中。

**说在最后**

　　或者这样一篇短短的[**文章**](javascript:;)无法涵盖软件测试的内涵，但是笔者也只是想抛砖引玉，让读者能够通过这样一种不能算全面的梳理，结合自己的工作经验，对自己所从事的软件测试工作有一个更深的理解。

　　笔者计划根据这篇文章所列出的技能树，分别写文章进行更加细致的梳理和总结，希望能够和各位同行一起学习，一起进步，同时非常欢迎大家指正我的错误和不足。