《Google软件测试之道》

图书基本信息

书名：《Google软件测试之道》

13位ISBN编号：9787115330246

10位ISBN编号：7115330247

出版时间：2013-10

出版社：人民邮电出版社

作者：James A. Whittaker,Jason Arbon,Jeff Carollo

页数：258

译者：黄利,李中杰,薛明

《Google软件测试之道》

内容概要

每天，google都要测试和发布数百万个源文件、亿万行的代码。数以亿计的构建动作会触发几百万次的自动化测试，并在好几十万个浏览器实例上执行。面对这些看似不可能完成的任务，谷歌是如何测试的呢？

《google软件测试之道》从内部视角告诉你这个世界上知名的互联网公司是如何应对21世纪软件测试的独特挑战的。《google软件测试之道》抓住了google做测试的本质，抓住了google测试这个时代最复杂软件的精华。《google软件测试之道》描述了测试解决方案，揭示了测试架构是如何设计、实现和运行的，介绍了软件测试工程师的角色；讲解了技术测试人员应该具有的技术技能；阐述了测试工程师在产品生命周期中的职责；讲述了测试管理及在google的测试历史或在主要产品上发挥了重要作用的工程师的访谈，这对那些试图建立类似google的测试流程或团队的人受益很大。

最后，《google软件测试之道》还介绍了作者对于google测试如何继续演进的见解、google乃至整个业界的测试方向的一些预言，相信很多读者都会感受到其中的洞察力，甚至感到震惊。本书可以作为任何从事软件测试人员到达目标的指南。

《google软件测试之道》适合开发人员、测试人员、测试管理人员使用，也适合大中专院校相关专业师生的学习用书，以及培训学校的教材。

《Google软件测试之道》

作者简介

james whittaker 是google的工程总监，负责部分google产品的测试，包括chrome、地图、google web apps。在加盟google之前，james在microsoft工作，再之前是一名大学教授。james在全球测试领域闻名遐迩。

jason arbon 是google的一名测试工程师te，曾参与负责google桌面、chrome和chrome os的测试。同时jason也是一系列开源测试工具和个性化实验的开发负责人。在加入google之前，他也曾在microsoft工作过。

jeff carollo 是google的一名测试开发工程师set，曾参与负责google voice、工具框、chrome、chrome os产品 的测试。jeff为许多google内部的开发团队提供咨询服务，帮助提升这些团队初期的代码质量。之后在2010年，jeff转岗为软件开发工程师（se），并领导负责google+ apis的开发。在加入google之前，jeff也曾经在microsoft工作过。

《Google软件测试之道》

书籍目录

《Google软件测试之道》

精彩短评

1、Good!
2、实际帮助不大
3、给一颗星的原因是，翻译真的是差劲到死，看不下去了。
4、读了一半感觉好像还没到读这个的层次..好像很厉害的感觉..
5、感觉能借鉴的东西很少
6、再多一些实践的内容就好了
7、开发人员和测试人员都值得一读的书。讲述了Google内部是如何对自己的产品做测试的，从专职的测试人员关注产品质量，到全团队关注产品质量的过渡；测试人员在Google负责不同的事宜，负责产品测试、将测试自动化、让测试更方便、培养所有角色的质量意识，到最后开发测试人员完全并入开发团队，利用技术手段将测试人员解放出来，开始转型，做更高level的考量，专注多个产品的测试计划和策略的制定、修正以及更多考虑用户体验。无论是敏捷还是持续交付，所有这些热门概念倡导的理念，都能在这本书里看到，并且在Google内部有了很不错的实践，虽然一字未提敏捷和持续交付。
8、好书。回头抽空写评论
9、细读与泛读所看到的区别太大
10、总体来说 一般般
里面的理念我很欣赏，让开发更多的参与测试当中，这样质量才会有显著提高。
里面还提到了测试开发工程师，大概知道做什么的，但具体还是不太清楚，回头还得看一遍。
还有里面提到了测试工具 但很不详细 而且有些已经停止升级了 比如：BITE
总之是把吊足了胃口 然后非常失望
11、介绍了Google的测试，开开眼界。不过薄薄一本书貌似不值这个价格。
12、提出了测试的重要性，都是故事，挺精彩且有启发的故事。
13、前言一大堆的高评价+序言的高调的自述，
却对不住本书切实关于测试精髓承载，里面关于测试的论述像是迷失的帆船永远靠不了岸，永远毫无方向的乱转一气，没有切实到测试关键点上。所以，那些给了极高的评价与序言的同学，请问你们是如何迷乱的海洋上找到了靠岸的方向啊。
14、虽然实用性不强，但google的测试理念还是为测试发展指明了一个方向。
15、不只是测试，还有工程师文化的建设。
16、书不厚，读起来比想象的还要快一些。
前几章介绍了测试相关的人员和内容。其中印象比较深刻的还是那些工具和方法论：ACC，GTA，质量机器人，BITE，RPF等等，可以尝试在项目中引入。不过一看到第四章的标题就不太想往下看了：测试工程经理。果然内容也只是各种访谈。
17、买回来速读了遍，技术要求太高，代码质量类与产品离得有点远了吧
18、对于正践行着敏捷测试的团队来说，这本书再好不过。没有乏味的教条，本书充满了在敏捷思想下为解决实际问题而磨砺出的种种因地制宜循序渐进的测试开发实践。经常的会心一笑证明我们的敏捷测试是走在了正确的道路上。但伟大的公司之所以伟大，也许不在他们的流程比我们先进多少，不在他们的工具比我们先进多少，而在人才的筛选和对技术的尊崇。这是我们难以学到的。五星推荐给团队（both dev and test）。
19、从多个维度分析了Google的测试是如何进行的，及一些问题是如何处理的。
Google内部的测试团队是如何协调工作。
20、看产品吧，毕竟谷歌的软件产品有他自身的特点，适合这种测试方式方法~
21、如果谁在告诉我google没有专职的软件测试人员，我真的会一板砖拍死他的.....Bite、QualituBot、RPF、Google Test Analytics这几个工具有空可以尝试下，感觉很不错的样子。
SET、SWT、ET
测试人员能尽早的能介入到开发流程中去，但不是通过“质量模型”和“测试计划”的方式。

测试是应用产品的另外一个功能，而SET就是这个功能的负责人

小型测试——通常指单元测试。外部服务必须通过mock或者fake实现
中型测试——外部指的集成测试，模块应用之间的交互，也类似于接口测试
大型测试——为了验证整个系统作为一个整体是如何工作的
70/20/10原则，70小型，20中型，10大型
22、为测试正身。测试是一件重要且非常有意义的活。
23、Google EP 项目组成员访谈录
24、将自动化做到，力争克服 ”人类智慧的最后一英寸“ 这也是Google的设计理念与目标，也正是正在构建之中的下一代测试工具的努力方向
25、除了测试专家或者测试管理，测试开发适合转开发，测试适合转产品，其实测试的前途和钱途远比开发要宽广，前提是资质差不多的人。现实是很多资质不错的都想做开发，不愿做测试，这是误区！开发岗位适合那些一心走技术路线的人！
26、感叹于google的专业性，测试开发。
27、相当具有启示意义。
28、进行测试开发，重视测试的流程化
29、acc～te面试题的发散思维～web工具介绍～
30、测试通过工具来保证，工具通过企业文化来催生，文化又通过工具来传播。周而复始，终得质量！
31、这本书写的非常好，没有得到8分以上太可惜了。真正在做产品、做测试的人才能明白文章说的很多方法。
32、8gjlo tm6f d6a yg ta eaf jm 96g d7 57 noa noa roa dafb tm 5a nom 96 tm7 noa ym tm6f bm b7 roa j7f ea6f noa 96 bgfb na tgm ta 96a h6 om6f eaf o7 8m ym roa lm eafb bgfb 9mg roa oma r6a l6f jm o7 9a r6g ta b7 96a h6 lm6f 9ma 7j 4m noa ta b7 96f 5omf roa 5o6 5mg 57 noa 7j ta l6 h7f 5o7f ha6f tm d6g noa rmg noa 86fb noa 7j tgm 46g tm6f ra ea roa 9a y7a
33、这本还不错，会让人有些不一样的想法，只是落地应该挺不容易。
34、这本书让我们学会了周五召开TGIF会议，我本人认为分享是一件很好的事情，但是感觉让团队分享成为一种习惯实在是有点力不从心，google的测试主要是交给单元测试，次要是系统自动化测试，全球最前沿的公司告诉你，不会开发的测试不是好程序员，目前中国在测试上举步不前。就行业感受来讲。
35、对测试方法讲的不多
36、了解下大公司的测试工程是怎么做的~~#公司福利#
37、这本书的视角相当高，高到没几家公司学得起，但又是满满的干货。测试成熟度、几种分类法、ACC模型、几个工具的设计思路、测试人员的定位和关注点、不同项目的高层对测试的理念都很棒，值得之后再看一次，仔细归纳整理能落地的点
38、作为编程行业的翘楚，谷歌的软件测试可谓业界工程典范。从无到有搭建测试平台。强烈建议开发工程师也阅读本书，从另一个环节来考量软件工程。毕竟软件质量真的不仅仅是测试工程师的责任。
39、非常好的一本书。对于测试人才，职责，各项测试任务的理解边界划分都很不错，非常值得参考。
40、google是如何进行开发产品的.. 以及一些访谈. 可能是预期太高,读时没达到预期的效果
41、博客文章很久前就看过，这周末终于找到时间把全书快速浏览了一遍。我很想知道的是，如何才能驱动这样一场改变，如何才能成功的进行这个改变，如何才能在资源极有限的情况下让改变走起来？
42、其中的很多理念和做法深有同感，只是对于非谷歌人士来说，以参考为主，切不可鲁莽行事。谷歌可以达到目前这样的状态，也是经过了很多人的努力和争取，而且其中的很多做法也都是跟谷歌业务的特点息息相关的，例如，并不是所有公司的产品都是围绕着Web或者同一种技术进行开发的，那么在测试平台和测试方式的选择上就会有很多的限制。
43、高屋建瓴 先码一下 再分享读后感
44、慢慢读慢慢写读书笔记 看到50页的最大感受就是好想去这样的公司工作，不给工资都行啊。
70/20/10原则很有用，我们现在做的测试不过集中在那10中，也没把代码覆盖率考虑进来，好弱。

45、ACC，attribute，component，capability
46、和旧式测试差异很大的测试方法
47、翻墙不容易的年代有这样一本书全面了解谷歌的测试体系和工作方法，简直不要再好了
干货满满，收获也是很多的，测试必看的书籍
48、Google的软件测试之道，参考意义首先跟之前看的一篇文章一样，互联网并不缺前端，也不缺测试，我们真正缺少的是工程师思维，是创造性的提出和解决问题的想象力，站在用户角度思考的产品的能力
49、这是这个迭代敏捷的测试方法，测试也可以是一个高技术的工作，而非没深度的点击而已。
50、一直印象中测试都挺低级的，看了这个书才对测试有了更深入的认识，当然，依然不是对国内的测试这么看。

《Google软件测试之道》

精彩书评

1、作为软件开发工程师看这本书收获良多，之前没看这本书，对测试产生很多偏见，产品质量由测试负责。ACC方法能清楚描述产品具有什么功能和用户能做什么功能，开发人员也可以利于这种方法对产品理解。最重要一点是访谈内容十分精彩，现在看一遍还有些不明白，有空还会重看一次。
2、这本书大概讲解了一下Google如何做测试。首先介绍了角色划分，然后分别从每个角色的角度进行阐述。每章最后都有一些访谈。这本书让我进一步思考开发和测试的边界在什么地方。目前看起来，开发会负责到单元测试，测试承担了功能测试、验收测试以及其他的职责。但是实际上，开发有多少人写出了有效的测试用例、高效的测试用例？测试又有多少人对产品有着深刻的认识，从而能够保证最重要的功能测试被覆盖到。测试的未来在什么地方？自动化和手工测试如何取舍？开发如提高测试能力？这些问题我并没有从书中得到解答。还需要更多阅读。
3、本书颠覆了我对测试之前的了解本书有两点核心。一是角色（SWE,SET,TE，最终是开发测试极度融合，相互辅助前行）二是ACC理念。（A,C）坐标定义的区域，有两个输出，一个是C(能力，就是test plan)，一个是风险热图（risk heat map）。测试的出发和落脚点是A(attribute)，也是本书提到的，不懂产品的人，根本无法真正的进行测试。技术分享时，整理了ppt和samplehttp://www.kuaipan.cn/file/id_5276661098915144l.htm?source=1
4、作为《微软的软件测试之道》的译者之一，差不多五年以后再来看这本书，是一种很有意思的体验。这本书当然写得很好，但好在哪里，可能未必人人都能说出个道道来——这很像是软件测试行业本身，充满了对这个行业的各种片面的认识，而且这些片面认识的来源往往是“一线的工作经验”。因为很久以来，软件测试的理论和实践的发展一直处于资源和人力相对不足，工作内容的边界也很模糊的状态。市场上的现有书籍的最大问题，在于站的位置不是太高就是太低。以学术口吻写就、工程人员一看就感觉与己无关者有之；以自己的实际工作经验为基础，但是并未形成有效的理论者有之；泛泛而谈的理论家凭借相当的想象写就，但是既没有实例也没有工具者有之——一言以蔽之，光说了一堆测试这件事应该怎么做，但是并没有看到测得什么优秀的产品，也没有在这个过程中发展出工具、理论和文化来，这样的软件测试图书，我以为价值是不高的。但是《Google软件测试之道》——如果允许我小小地、非完全地自夸一下的话——还有《微软的软件测试之道》这样的书，则是极有价值的了。这首先在于，IT巨头已经生产出来的软件产品，其成功是妇孺皆知的。那么，以这些产品的生产过程作为软件开发生命周期中的模范，应该是不仅比较正确，而且也是更有沟通基础的，因而也是更有价值的。软件测试作为软件开发生命周期中接触点最多的一环，看一看这些IT巨头们是怎么做的——我的意思是，它们怎样设计配套的公司组织结构、怎样处理软件测试和其他生命周期环节的关系、基于怎样的思想来实施工程实务、开发了怎样的支撑环境和工程工具……这一系列的问题，看看微软和Google给出的答案，读者就可以知道，在目前的软件和硬件条件下，理想的，或者说是最高水平的软件测试已经达到了一种怎样的程度，从而在规划自己软件测试相关的团队配置和工程技术实务时，有了非常重要的参考。一般来说，能够真正达到这样的水平，是不太可能的，可是榜样和模范的力量就在于此：做一个完整版、旗舰版不可能，但是做一个精简版、定制版就总算有了一个参照可言。可以说，我本人对于测试自动化的全部概念都来自于微软的测试工程，如果不是实地看到成千上万台物理机和虚拟机在极有序、极智能地运行着各种计划任务，来测试一个小小的安全补丁在数种CPU和硬件体系结构、数十种网络条件和共存软件、数百种参数和开关组合下取得种种的功能和性能数据，并自动出具详尽的测试报告和一针见血的分析建议，光是看看教材是绝对不可能带来如此的心灵震撼的。我在之后的工作经历中，也反复地应用了这些宝贵的经验。然而《Google软件测试之道》又让我看到了一些新的东西，这主要是软件交付模式从盒装变为在线带来的，因而时间特性从离散变为持续（服务从主要在客户端运行变为主要在云端运行）、空间特性从单一变为多元（对于操作系统和设备特性的假设更弱化），软件测试为了适应这些变化，必须一方面在概念上变得更灵活，另一方面却要在执行上却要变得更简单有效。这两个相互矛盾的要求，对软件测试的管理和执行人员都提出了极大的挑战。毫不意外地，我看到这本书在基础层面上与传统的盒装软件测试有着同样的测试目标：高可用性、高性能、优化的用户体验，但是在概念和技术上却发生了巨变，我诚挚地请读者们重点关注一下这本书对于测试规模的论述，以及性能测试工具的设计思想，信息量很大。另外，认真研习一下附录的两份测试计划，也会有不小的收获。这本书的问题主要是“太Google的”访谈内容有点儿多，当然这对可读性的提高有一定的贡献，但是至少现在我还没觉得这部分的价值有多大。

5、《Google软件测试之道》总的来说，这本书是我看过的所有软件测试相关书籍中，收益最大的一本。个人觉得，这本书更适合有一些测试或工具开发经验的人看。测试经验较丰富的人，看了收益较大，初学者也能领会到一些基本的东西。这本书主要通过对测试开发工程师（SET）、测试工程师（TE）和测试工程经理三种角色及其各自负责的工作的介绍，将Google测试的整体概况和部分细节（2012年前的情况）介绍给读者。小型测试（fake环境） -> 中型测试（fake或真实环境） -> 大型测试（真实环境）mock：对外面依赖系统的模拟，一般可以动态地设置返回值；fake：一种虚假的实现，只返回固定的结果；stub：和mock意思接近，但它一般不知道是否被call过。我自己觉得这3者差别也是太大，偶尔大家都混用（如全都叫“ mock ”）。个人感觉是，一般是测试方写的mock/stub，提供方在没完成功能前提供fake；另外，mock/stub一般可以工具直接生产，而fake一般是手写的。强调小型、中型测试的自动化覆盖率；在端到端自动化测试上投入过多，常常会和特定功能设计绑定在一起。小型、中型、大型测试的比例：7:2:1测试认证级别，根据测试覆盖率、分层测试情况、继续集成、缺陷和测试用例的关系，将团队分为5个级别。 Level 5 最高，做的做好。SET主要负责mock/stub、API测试、测试工具、CI工具，更偏向于代码开发；2.3节 SET的照片写的不错。TE是面向用户的测试，具备测试代码开发和用户为中心测试的双重能力。TE完成测试整个过程，风险评估、测试计划、测试执行、探索式测试、用户反馈。3.2.4节 bug的生命周期、bug的要素等，也是值得借鉴的。Google feedback 一个极简的用户提交bug的方式。3.2.5节 TE的招聘，SET和TE的区别，面试TE的问题和各种回答的分析，都非常精彩。测试工程经理，把TE和SET联系起来，需要足够的技术能力，需要足够了解产品，也需要知人善用的能力。Google的原则是：ship early and often, fail fast.Google测试的秘方：技能、稀缺、自动化、迭代集成。第5章 关于软件测试的未来的论述也不错，不过里面描述的那个未来估计至少10年之后才会明朗起来，20年后估计才能普及吧。测试工程师和测试经理分散到各个项目团队中去，更少关注测试流程，更多关注产品本身；测试开发工程师成为开发工程师。技术型测试主管，更多地成为资深工程师。

6、0、google强调在互联网时代测试的转身，并且利用测试人员很少的情况下如何做的更好。就是测试团队将能力导入开发，测试写代码逐步向开发转身，一部分测试人员向用户转身更好的测试。1、小型、中型、大型比例约为 7：2：1。如果是基础平台，面向数据的项目，UT比例应该更高。UT带来良好的代码质量，良好的异常处理，优雅的错误报告。大中型测试会带来整体产品质量和数据争取保证。2、测试认证 级别1-5，设定和导入开发团队。1)侧重基本准备。2）提高增量代码覆盖率。3）测试新增代码。4）测试历史遗留代码。5）更换的整体覆盖率，针对每个缺陷都增加测试用例，并要求使用已有可用的静态与动态分析检查工具。3、让每个工程师都重视质量，代码质量从一开始就能更好，早期构建版本的质量会更高，系统测试可以关注与真正面向用户的问题。4、测试的一些老大难问题：开发关注测试，开发和测试组织的分开，测试过于关注测试的产物bug，经过测试的产品发布后总是有问题。5、google的特点，总是让资源很紧缺，负责人想办法提高效率，用到各种的工具，自动化。非常强调自动化。

7、很多时候，人们总会说某些理念或做法太过“ 理想化 ”，其实只是低头走路太久，忘记了抬头看天而已。如果我们心中没有理想，又如何可以创造出伟大的成就呢？而当下在测试领域，谷歌就是坚持理想实现了书中理念的典范。p12~13 小型测试、中型测试、大型测试……mock和fake……小型测试主要尝试解决的问题是“ 这些代码是否按照预期的方式运行 ”……中型测试尝试去解决的问题是，一系列临近的模块互相交互的时候，是否如我们预期的那样工作；这种端到端的使用场景以及在整体产品或服务之上的操作行为，即是大型测试关注的重点；重要的是，在Google测试人员使用统一术语来谈论他们测试的是什么，以及这些测试范围是如何划分的。p15 测试框架（test harnesses）、测试通用测试（test infrastructure）、模拟设施和虚拟设施（mock and fake）p15 对于功能代码而言，思维模式是创建，重点在于考虑用户、使用场景和数据流程上；而对于测试代码来说，主要思路是去破坏，怎样写测试代码用以扰乱分离用户及其数据。p17 工程师团队的交付物就是即将发布的代码。代码的组织形式、开发过程、维护是日常工作重点。p19 最小化对平台的依赖。所有工程师都有一台桌面工作机器，且操作系统都尽可能地与Google生产环境的操作系统保持一致。……所有对平台有依赖的代码，都会强制要求使用公共的底层库。p20 使用统一的运行平台和相同的代码库，持续不断地在构建系统中打包。p21 服务之间的接口需要在项目的早期就确定下来。……这些接口一般都不会真正实现，而只是做一个虚假的实现。p22 在未来可能失败的项目中投入测试资源来构造测试方面基础设施，这是一种资源浪费。p24 所有Google项目都有设计文档。这是一个动态的文档，随着项目的演化也在不断

地保持更新。p26 Google protocol buffer （http://code.google.com/apis/protocolbuffers）p29 提交队列（submit queue）的主要功能是保持“绿色”的构建，这意味着所有测试必须全部通过。这是构建系统和版本控制系统之间的最后一道防线。p48 检验一个项目里小型测试、中型测试和大型测试之间的比率是否健康，一个好办法是使用代码覆盖率。……Harvester……p51 持续集成系统使用构建系统中的构建依赖规则。p63“SET的招聘”对于候选者，最好去考察如何思索问题的解决方案，而不是解决方案本身的实现上体现得多么高雅。……只有一件事情需要去做而我正在做这个事情，这个事情就是写代码。SET不会遵循这样的世界观。我们希望先把问题搞清楚。p70 我认为最艰难和最有趣的挑战总是出现在设计阶段。p98 与其询问他们关于某个模糊概念的看法，不如拿一个明确的结论来引起辩论。p101 Google Test Analytics支持上述基于分类赋值（非常罕见、很少、偶尔、经常）的风险分析。……知道A比B风险大就足够了，不需要过分关心它们的具体风险值。p120 bug分类过程（bug triage process）……聚类算法来自动识别重复记录并确定最频繁的问题。……需要精简到10个左右主要的、共性的问题。p120 修复会产生一个变更列表（CL），CL排队去接受评审，一旦得到批准，就进入构建目标队伍中。p121 TE的招聘尤其困难，因为最好的TE不是那些基础算法、定理实现、功能实现上的牛人。……TE是稀缺个体，是技术人，关注用户，能在系统级别和端到端的视角上理解产品。他们是无情的、伟大的谈判专家，更重要的是，他们富有创造性、善于应对模糊性。p122 混合模型：今天，我们的面试既要考察一般的计算机科学和技术技能，也要考察候选人的测试潜力。编程知识是必需的，但只限于那些完成前述TE工作需要的水平：修改而非创建代码、设计端到端的用户使用场景的能力等。再加上TE工作本身需要的一些特定的能力，如沟通、系统级别的理解以及用户同理心。p124 一开始，我们考察测试资质。……我们寻找的是对于事物结构、对于变量和配置的组合的各种可能性和意义的好奇心。我们寻找的是关于事物应该如何工作的强烈感觉，以及清晰表达的能力。我们还会试图寻找很强的人格魅力。p127 面试时试图考察的另外一个关键特征，是TE所需要具备的处理模糊性、反驳糟糕想法的能力。……TE面试的最后一环是看候选人是否具有“Google味儿”。p128 Google的测试管理更多的是激励，而非强悍的管理；更多的是战略指引，而非频繁的督促检查（每天、每周等）。p131 TE到SET、SET到SWE是最为常见的。p131 OKRs（Objectives and Key Results），也即目标和关键结果。p134 质量机器人（Quality Bot）实验p143 我的反应是很Google式的：表面同意，但私底下一切照旧。p145 这个特性有单元测试用例，但只有bot逮住了这个问题，因为它测试的是真正的网页。p145 BITE代表Browser Integrated Test Environment（浏览器集成测试环境）p151 我们实现了一个称为Record and Playback framework（RPF）的纯Web的解决方案，是用纯JavaScript实现的……在回放的时候，RPF首先查找精确匹配，在找不到的情况下查找近似匹配。……处于安全原因，某些涉及钱的场景无法通过web API自动化。p153 这个做法已经成功地用于众包测试，外包测试人员在安装了BITE的浏览器中执行测试，测试任务的分发通过BITE进行。p165~170“Lindsay Webster的访谈”对于一个新项目，我首先要站在用户的角度了解这个产品。……从头到尾的理解产品。……关注项目的状态，特别是质量状态。……只有熟悉了团队的全貌，才能真正有效的展开工作。……我会了解他们沟通的方式和对测试人员的期望。……询问他们对测试的期望，会帮助发现开发团队没有测试过的内容。……第一件事是把应用分解为合理的功能模块。……排列测试的优先级……再次检查Bug库……按照优先级顺序更加细致地遍历所有模块，创建用户故事……通常会编写测试用例并链接到相应模块的用户故事。……有了测试集合，我接下来会通过再次检查bug和应用来寻找覆盖度上的不足。……有了这些基础材料，我的工作通常只是维护和更新：更新测试用例，增加新特性的文档，更新变化了的模块的截屏或视频。最后，观察哪些bug遗漏到了生产环境，会告诉我们测试覆盖上的不足。……我把自己变成用户，就这么简单。……换句话说，测试要清楚地指出当做之事。……开发经常会低估我的工作，知道我们在一起工作了几个月之后，他们才会改变想法。我在完成了上述工作之后，将邀请整个团队开会，介绍一下我设定的测试流程。……当我坦诚地指出某些组件或领域的测试不应该由我负责，而应该由他们自己负责的时候，开发反而更加看重我的工作了。……这就是为什么要按照一定的优先级处理应用的各种功能和环境支持。……重要的是，要维护团队的这份信任——如果我强烈感到发布时机未到，那么这可能也是他们希望的。……当然，还是有一些不那么尊敬我的工作的SET，但他们就像有类似想法的开发人员一样，不曾与我或其他TE一起工作过。一旦发生了合作，他们的态度通常会迅速的转变。p174“Apple Chow的访谈”遵守70-20-10法则：小型的用来验证单个类或功能的单元测试占70%，中型的用来验证一个或多个应用模块之间集成的测试占20%，大型的高级别的用来验证完整应用的测试占10%。p178 想成为优秀的测试工程经理，第一条建议就是去了解你的产品。……第

二条建议是知人善用。p179 不能仅仅依赖于某位明星测试人员。……必须要沉淀为可用的工具，或者总结成一套方法，这样可以帮助其他人也能走上这条成为明星的道路。p181 在Google，对工程师最好的褒奖就是称赞他的影响力。而对于测试工程经理来说，就是建立一支有影响力的团队。p183 多年来，通过不断地聆听，我发现最有力的问题就是“为什么”。p184 所以经验就是解决掉一些难题来赢得尊重。p187 我们更专注于预防bug而不是检测bug，这为我们带来了巨大收益。p193 目前我还是倾向于使用一种综合的方式，混合使用开发自测、脚本化测试、探索式测试、基于风险的测试、自动化功能测试等多种方法。p198 我们经常由于太难保证后台系统的质量而不能按时发布。后台系统不能出现差错，因为它影响太多产品线了。……在后台系统中如果仅仅使用端到端测试，要是不能发现问题，就会导致连锁反应。p200 一般来说，我首先会让我的团队思考，“对被测系统来说，什么是最为重要的东西？”p201~203“Ashish Kumar的访谈”整个工具集包括：源码工具；开发工具；测试基础架构；本地化工具；度量、可视化和报表。……大规模的持续集成（是我一开始不看好但最后成功的）……从小做起，不断证明其价值，然后当项目体现价值以后扩大规模。……特别重要的一件事，是要关注团队里新来的开发工程师必须使用到的开发环境。要让代码的获取、编辑、测试、运行、调试和部署都非常简单。p214~216“James Whittaker访谈”第一条，是先花一些时间来观察学习。……聆听而不是直接发言，询问而不是直接尝试。……第二条建议，“兄弟，我知道你在来Google之前已富盛名，但是在这个公司里，你还什么成就也没做出来呢。”……先虚心学习，再在一线做出成绩，然后开始寻求创新的方法。……其他公司要想仿效Google的做法，应该从这四个方面做起：技能、稀缺性、自动化和迭代集成。这就是Google测试的“秘方”。p219~221“Google流程中的致命缺陷”第一个致命的缺陷：测试成了开发的拐杖。我们越不让开发考虑测试的问题，把测试变得越简单，开发就越来越不去做测试。……第二个致命缺陷，还是与开发和测试的组织结构分离有关。……第三个致命的缺陷，是测试人员往往崇拜测试产物（test artifact）胜过软件本身。……最后一个致命缺陷也许是最深刻的。产品经过最严格的测试发布以后，用户有多大可能仍然发现测试中遗漏的问题？答案是：几乎必然发现。……是谁在做测试并不重要，关键是进行了测试。……SET的角色越来越像开发，而TE的角色向着相反的方向越来越像用户。……质量需要每一个任的贡献。p222 测试的技能被平均地分散到各个层级的开发工程师身上，而不是集中于测试开发工程师（SET）那里。p223 测试工程会转型成为测试设计。……这个工作需要的是规划、组织和管理近于免费的测试资源。……测试工程师会转变成像安全工程师这样的专家型角色，或者他们会变成测试活动的管理者。p224 技术型的主管，将会更多地转向成为诸如杰出工程师这样的个人角色。……作为思想领袖，为维系……关系而存在……测试活动应该对人们具体工作的产品负责。p224 测试基础设施会最终整体迁移到云端。测试用例库，测试代码的编辑、录制和执行都将在一个网站或通过浏览器插件完成。测试编写、执行和调试需要使用与被测的应用程序本身相同的语言和环境才最为高效。==========徐毅：独立敏捷顾问，经验丰富的国内知名敏捷及精益教练，专注于敏捷软件开发、Scrum、敏捷转型、敏捷测试、测试自动化、robotframework等。

8、Google的软件测试之道，参考意义首先跟之前看的一篇文章一样，互联网并不缺前端，也不缺测试，我们真正缺少的是工程师思维，是创造性的提出和解决问题的想象力，站在用户角度思考的产品的能力。我们学习Google软件测试，只能学其道而没有必要，也不可能去他们的术。如《看板方法》一书所说，对于新组建的团队，尤其初创公司，能力建设才是唯一的问题，业务成功是压倒一切的目标，而且业务本身充满风险和变化，并没有空间去发明太多轮子，何况google这个轮子，直接就是selenium、webdriver这个级别的测试工具，并非一般工程师能做到的。最终软件行业真正寻找的既不是前端，也不是后端，也不是测试，唯有工程师思维，充满好奇心和创造力的头脑，在自由的环境下，发挥创造力这样一个理想状态。回到测试本身，无论测试计划也好，需求分析也好，核心在于谁来看，如果写的时间超过看的时间，而且找不到目标受众，其实也就没有必要。测试金字塔也是一个重要的实践，在自动化everything的理念下，测试又回到了纸带卡片的时代，需要合理的权衡测试的规模和代价。核心其实不在于文档，也不在工具，核心还是思考和创造数据，站在用户的角度思考产品的意义。至于编写测试的工具，发明轮子在做好测试，只是理所当然。至于te和测试总监，其实跟开发一样，只有高质量的有趣的项目，才能吸引更多的测试人员投入，如果测试人员都抛弃的项目，估计被用户抛弃也就不奇怪了。测试总监只需要招聘最合适的人，然后创造条件去激发创新，就可以了

9、1. 自动化测试，说起来容易做起来难，有google能做到不代表所有公司都能做到。况且google自己就做到了么？ 自动化测试占前期测试方案的百分之多少？2. 书中推崇自动化，却缺乏一般性方法，只

举特例，特例又只举成功的，例如某某花了20%时间做了个啥啥，然后大获成功，那没成功的项目花的时间怎么算？描述成功的例子时用大量形容词，几乎从不涉及具体百分比数字3. 看完本书，我觉得google真NB，但是怎么样才能让我们也向google一样NB呢？就不告诉你，因为里面的方法都是针对特定的google产品的。4. 书中提到一个成功的工具或者方法，就说目前有“ 许多 ”小组都在用，“ 许多 ”是多少？占全部小组比例的百分之多少？究竟把百分之多少的测试用例自动化了？5. google的测试人员少，因为项目前期有风险，不测。后期产品成熟了，不测。中期把一大部分交给外包测，确实是好点子。

10、Patrick Copeland谷歌测试和部署技术的架构师我在Google的旅程始于2005年3月。Alberto在前面的序中也介绍了一些当时Google的状况：虽然公司规模还比较小，但已开始感受到成长带来的烦恼。当时适逢快速的技术变革之际，Web世界正在迎接动态内容的到来，而云计算也正在逐渐成为一种新的选择，取代当时还占统治地位的客户机-服务器架构。在加入Google的第一周里，我和其他Nooglers（译注：New Googler，新加入Google的员工）一起，戴着三色的螺旋桨帽，参加了称为TGIF的公司每周例会，听创始人介绍公司战略。我对彼时的工作情形还知之甚少，有些兴奋，也有些害怕。在我之前10年的研发模式经历中，一个典型的交付周期可长达5年，这种经历在Google的速度和规模面前显得毫无价值。更糟糕的是，我觉得自己是所有戴着Noogler帽子的人中唯一的测试人员。当然，其他地方一定还有更多的测试人员！我加入Google的时候，工程团队还不足1000人。测试团队大概有50名全职人员和一些临时工，具体数量我一直没搞清楚。测试团队当时的称谓是"测试服务"，工作重点在UI的验证上，随时响应不同项目的测试需求。可以想象，这并不是Google最闪耀的团队。但这在当时已经足够了。Google当时的主要业务是搜索和广告，规模要比今天小得多，一次彻底的探索式测试足以发现绝大多数的质量问题。然而，世界在变，Web点击量开始史无前例地爆发性增长，文档化的Web正在让位于应用化的Web。你可以感觉到势不可挡的成长和变化，在这种情况下，规模化和快速进入市场的能力变得至关重要和生死攸关。在Google内部，规模和问题的复杂性给测试服务团队带来了巨大的压力。在之前小型的、类同的项目里的一些可行做法，现在却让优秀的测试人员感到筋疲力尽，疲于奔命在多个急需救火的项目之间。更加火上浇油的是，Google在项目快速发布方面的坚持。是时候采取措施了，我面临两个选择，要么沿用这种劳动密集型的流程增加更多的人手，要么改变整个游戏规则。为了适应业界和Google发生的巨变，测试服务团队需要根本性的变革。我也很想说自己是借助于丰富的经验构思出了完美的测试组织模型，但实事求是地讲，我从过去的经历中，学到的只不过是一些过时的做法。我所工作或领导过的每个测试组织都有这样或那样的问题。有问题是常态，代码质量很糟糕，测试用例很差劲，团队也问题多多。我完全清楚那种被技术质量债压得喘不过气来的感受，在那种状态下，一切创新性的想法都会被遏制，以免不小心破坏了脆弱的产品。如果说我在以往的经历中有所收获的话，那就是经历了各种错误的测试实践。那个时候，以我对Google的了解，有一件事情是确定无疑的，那就是Google对于计算机科学和编程能力非常重视。从根本上说，如果测试人员想加入这个俱乐部，就必须具备良好的计算机科学基础和编程能力。变革Google测试的首要问题是重新定位身为测试人员的意义所在。我过去经常在头脑中想象理想团队的模型，想象这样的团队是如何肩负起质量重任的，每次我都会得到相同的结论：一个团队能编写出高质量软件的唯一途径是全体成员共同对质量负责，包括产品经理、开发人员、测试人员等所有人。我认为，达到此目标的最好方式是使测试人员有能力将测试变成代码库的一个实际功能，而测试功能的地位应该与真实客户看到的任何其他功能同等重要。我所需要的能够实现测试功能的技能，也正是开发人员需要具备的技能。招聘具备开发能力的测试人员很难，找到懂测试的开发人员就更难，但是维持现状更要命，我只能往前走。我希望测试人员能为他们的产品做更多的事情，同时，我希望演变测试工作的性质和从属，要求开发团队更大地投入。这种组织结构在当时的业界尚未实现，但我坚信它非常适合Google，我相信在这家公司，时机到了。不幸的是，这种如此深刻、根本性的变革在公司里极度缺乏认同，极少有人能分享我的激情。当我开始推销这种关于软件测试角色的地位平等而作用不同的愿景时，我发现竟然难以找到一个人一起共享午餐！开发工程师们好像被他们将要在测试上发挥更大的作用这个想法吓着了，他们指出"这是测试人员的职责"。而测试人员也不买账，因为很多人已经习惯了当前的角色，维持现状的惯性导致任何变革都变得非常困难。我毫不松懈地继续努力着，主要是出于对Google的研发过程深陷技术和质量债的困境的恐惧，一旦如此，长达5年的开发周期又会成为现实，而我本来已经很高兴地把它们留在客户机-服务器的世界里了。Google是一家由天才组成的公司，以创新为灵魂，这种企业文化与冗长的开发周期是不相容的。这是一场值得打的战斗，我说服自己，一

旦这些天才理解了这种旨在打造一个生产线式的、可重复的"技术工厂"的开发和测试实践，他们就会改变看法。他们就会理解我们不再是一个初创公司，快速成长的用户群、不断累积的bug和糟糕结构的代码形成的技术债将会导致开发过程的崩溃。我逐个接触各产品团队，寻找优秀的案例，试图为我的立论找到比较容易的切入点。在开发人员面前，我描绘了一个持续构建、快速部署的蓝图，一个行动敏捷、省下更多时间用于创新的开发过程；在测试人员面前，我激发他们对于成为同等技能、同等贡献和同等薪酬的完全的工程合作伙伴的渴望。开发人员的态度是，如果我们招聘到有能力做功能开发的人，那么，我们应当让他们做功能开发。其中一些人对我的想法非常反感，甚至发信给我的主管，非常直率地建议如何来处理我的疯狂之举，这些信塞满了我的主管的邮箱。幸运的是，我的主管并没有采纳那些建议。令我吃惊的是，测试人员的反应竟然与开发人员类似。他们沉湎于老的做事方式，抱怨自己在开发面前的地位，但又不想去改变。我的主管对这些抱怨只有一句话："这里是Google，如果你有想法，尽管去做就是。"于是我开始付诸行动。我召集了一批志同道合的骨干分子，组成了一个面试团队，开始招聘。事情进行得比较艰难，我们寻找的人要兼具开发人员的技能和测试人员的思维，他们必须会编程，能实现工具、平台和测试自动化。我们必须对招聘和面试的标准与流程做出一些调整，并向已经习惯了既有模式的招聘委员会做出合理解释。最初的几个季度进行得异常艰难。好的候选人经常在面试过程中失利，也许是因为他们没能很快地解决一些奇怪的编程问题，或是在某些人认为很重要的方面表现得不够好（然而这些方面其实与测试技能毫不相干）。我预料到了招聘过程的困难，每周都要抽出大量时间写辩词。这些辩词最终会到达Google联合创始人Larry Page手里（他一直是招聘的最终批准者）。他批准了足够多的候选人，我的团队开始稳步增长。直到现在，我猜每次Larry听到我的名字时想到的一定是："招聘测试的！"当然，到这个时候，我已经做了大量的宣传和鼓动工作，来说服大家这是唯一的选择。整个公司都在看着我们，一旦失败，后果将是灾难性的。对于一个混合了很多不断变化的外包人员和临时人员的小测试团队而言，期望显得如此之高。然而，即使是在我们艰难的招聘进行中同时减少了临时人员的数量时，我已经注意到了变化在发生。测试资源越稀缺，给开发人员留下的测试工作就越多。很多团队都勇敢地接受了挑战。我感觉，如果技术保持不变的话，这个时候的状态已经在接近我们的目标了。然而，技术不是静止不动的，开发和测试实践处于飞速的变化之中。静态Web应用的时代已经成为过去，浏览器还在努力追赶之中，围绕浏览器的自动化技术比已经迟缓的浏览器还要落后一年。开发人员正面临着巨大的技术变革，在这个时候，把测试交给开发人员，这看上去是徒劳的。我们甚至还不太会手工测试这些应用，更不用提自动化测试了。开发团队身上的压力也同样巨大。当时Google 开始收购拥有富含动态Web应用的公司。YouTube、Google Docs等后继产品的融入，延展了我们内部的基础设施。开发团队在编写功能代码的过程中，要面临很多问题，与我们测试人员在测试过程中要面临的问题一样，令人生畏！测试人员面对的测试问题无法孤立地解决。把测试和开发割裂开来，看成两个单独的环节，甚至是两类截然不同的问题，这种做法是错误的，沿着这条路走下去意味着什么问题也解决不了。解决测试团队的问题，只是我们前进路上的其中一步而已。进展在继续。雇佣优秀的人是一件很有意思的事情，他们会推动进展的发生！到了2007年，测试团队有了更好的定位。我们能够很好地处理发布周期的最后环节。开发团队已经视我们为顺利上线的可靠合作伙伴。不过我们仍然是在发布过程的后期才介入的支持团队，局限于传统QA模型。尽管有了优秀的执行能力，我们还没达到我设想的目标。我解决了招聘方面的问题，测试也向着正确的方向发展，但是我们还是在整个流程中介入太晚。我们在一个被称作"测试认证"（本书后面的章节会详细介绍）的事情上取得了不少进展。我们向开发团队提供咨询，帮助他们改善代码质量并尽早进行单元测试。我们开发工具并指导团队进行持续集成，使产品一直保持可测试的状态。我们进行了无数的改进和调整，从而消除了之前的很多质疑，本书详细介绍了其中的很多方法。但是，在那个时候，还是感觉缺乏整体感，开发依旧是开发，测试依旧是测试。虽然很多文化变革的因素已经存在，但是，我们还需要一个催化剂把它们聚合成一体。自从根据我的想法开始招聘担当测试角色的开发人员以来，测试组织在不断壮大。基于对这个团队的思考，我意识到测试仅仅是我们所负责的工作的一部分。我们的工具团队开发了从源代码库到编译框架，再到缺陷数据库的各种工具。我们是测试工程师、发布工程师、工具开发工程师和咨询师。触动我的是，我们所做的非测试的工作对生产力的提升产生了巨大的影响力。我们的名称是测试服务，但是我们的职责已经远大于此。因此，我决定正式把团队名称改为工程生产力（Engineering Productivity）团队。伴随着称谓的改变，随之而来的是文化的革新。人们开始更多地谈论生产力而不是测试和质量。生产力是我们的工作，测试和质量是开发过程里每个人都要承担的工作

。这意味着开发人员负责测试，开发人员负责质量。生产力团队负责帮助开发团队搞定这两项任务。开始的时候，这个观点还只是一种梦想和志向，我们提出的"给Google加速"的口号听起来也很空洞，但是，随着时间的推移和我们的努力，我们实现了这些诺言。我们的工具让开发的动作更快，我们帮助开发人员扫清了一个又一个障碍，消除了一个又一个瓶颈。我们的工具还使开发人员能够编写测试用例，并在每次构建时看到这些测试的结果反馈。测试用例不再只是隔离地运行在某些测试人员的机器上。测试结果会在仪表盘上显示，并把成功的版本积累下来，作为应用发布健康性的公开数据。我们并不是仅仅要求开发人员对测试和质量负责，我们还提供帮助让他们可以轻松地达到这些要求。生产力和测试的区别最终变成了现实--Google的创新能够更为顺畅，技术债也不会累积了。最终结果如何呢？我可不愿这么早就交了底，因为这本书就是要详细讲述这个问题的。作者们花费了巨大精力，根据自身和其他Googler的经历，把我们的秘诀浓缩成了一套核心实践。但其实，我们的成功有很多方面，从将构建次数以数量级式地降低，到"跑完即忘"式的测试自动化，再到开源一些非常新颖的测试工具。在我写这篇序的时候，生产力团队已经拥有1200名工程师，这个数量比我在2005年加入Google时整个工程部门的工程师的数量还要多。生产力品牌的影响力已经相当大，我们加速Google的使命已经作为工程文化的一部分，被广泛接受。从我困惑、迷茫地坐在TGIF会议上的第一天到现在，这个团队已经走过了漫长的征途。这期间唯一没变的是我那顶三色螺旋桨帽，我把它放在我的桌上，作为我们一路走来的见证。Patrick Copeland是Google工程生产力部门的高级总监，处于Google整个测试链的最顶端。公司里所有的测试人员都最终汇报给Patrick（而他恰好跨级汇报给Larry Page，Google的联合创始人和CEO）。Patrick加入Google之前是微软的测试总监，并在那里工作了近10年。他经常公开演讲，在Google内部被公认为Google软件快速开发、测试和部署技术的架构师。via http://book.51cto.com/art/201312/420195.htm

《Google软件测试之道》

章节试读

1、《Google软件测试之道》的笔记-第2页

Patrick Copeland（Google测试和部署技术架构师）说：

* 一个团队能编写出高质量软件的唯一途径是全体成员共同对质量负责，包括产品经理、设计师、开发人员、测试人员等所有人。
* 达到此目标的最好方式是使测试人员有能力将测试变成代码库的一个实际功能，而测试功能的地位应该与真实客户看到的任何其他功能同等重要。

2、《Google软件测试之道》的笔记-测试工程师的工作

p123
你可能是一个TE
本部分讲述确认自己是否合适TE的若干场景，很有用

3、《Google软件测试之道》的笔记-第23页

开发团队在寻求测试帮助的时候，有义务让测试人员相信他们的产品是令人兴奋且并充满希望的。在Chrome OS的开发总监给我们介绍他们项目、进度和发布计划时，我们也要求提供当前已有的测试状态、期望的单元测试覆盖率水平、以及明确在发布过程中各自承担的责任。
“ 让测试人员相信他们的产品是令人兴奋且并充满希望的 ”觉得这个很重要啊……

4、《Google软件测试之道》的笔记-第143页

HGTS：Flash占据了YouTube内容和UI的一大部分，它怎样测试的呢？你们是否有某种通过Selenium测试Flash的秘籍？
Apple：不幸的是，没有。有的只是大量的艰苦劳动。Selenium在某些方面有帮助，因为我们的JavaScript API是暴露的，可以利用Selenium来进行调用测试。我们使用了一个图像比较工具pdiff来测试缩略图、最后一屏（end of screen）的渲染。我们还使用了大量的HTTP流代理来监听流量，这样就可以了解页面变化的更多信息。我们使用As3Unit和FlexUnit来加载播放器来播放不同的视频，以及触发播放器事件。关于验证，我们可以使用这些框架来验证软件的各种状态、完成图像对比。我想说这就象变戏法一样，但实际上是大量代码铺就的。

5、《Google软件测试之道》的笔记-第8页

SET写代码的目的是可以让SWE测试自己的性能

6、《Google软件测试之道》的笔记-第36页

" Quality cannot be tested in". Quality is not equal to test. Quality is achieved by putting development and testing into a blender and mixing them until one is indistinguishable from the other.

Google Test Rule NO.1: to merge development and testing so that you cannot do one without the other. build a little and then test it. Build some more and test some more. The key here is who is doing the testing. Quality is more an act of prevention than it is detection. Quality is a development issue, not a testing issue. Testing must be an

unavoidable aspect of development, and the marriage of development and testing is where quality is achieved.

在产品开发过程中，开发和测试缺一不可。

Roles we need to figure out: the software engineer (SWE) and the software engineer in test (SET) as well as test engineer (TE). SETs are partners in the SWE codebase, but are more concerned with increasing quality and test coverage than adding new features or increasing performance. SETs write code that allows SWEs to test their features. Test engineers is related to the SET role. but it is a different focus. It is a role that puts testing on behalf of the user first and developer second. TEs spend a good deal of their time writing code in the form of automation scripts and code that drives usage scenarios and even mimics the user. They also organize the testing work of SWEs and SETs, interpret test results, and drive test execution, particularly in the late stages of a project as the push toward release intensifies. TEs are product experts, quality advisers, and analyzers of risk. Many of them write a lot of code; many of them write only a little.

在这里，开发和测试不是根据“是否需要写代码”而区分。而是根据工作性质“开发产品和测试产品”区分。测试开发工程师，不单只是测试功能，还需要研究开发测试工具，完成测试脚本。而测试工程师，着重于站在用户的角度去测试一件产品。

Unfortunately, we have never actually seen that developers and testers exist as part of the same product team.

7、《Google软件测试之道》的笔记-第10页

测试发布过程：
金丝雀版本：每日构建的版本，用于排除一些明显不合适的版本，象煤矿井里的金丝雀
开发版本：开发人员日常使用的版本，一般每周发布一个，通过了一系列的测试
测试版本：通过了持续测试的版本。通常是近一个月内最稳当的版本，可以被挑选为内部尝鲜（dog food）。（也可以给一些特殊的外部合作伙伴使用测试）
beta或发布版本：非常稳当的测试版本演变而来，并经历了内部使用和通过了所有质量考核的一个版本，也是对外发布的一个版本。

测试类型：
小型：覆盖单一的代码段，一般运行在完全虚假实现的环境里。mock等方式。一般都是自动化实现。————可以理解为单元测试、代码测试
中型：覆盖多个模块且重点关注模块之间的交互上，一般运行在虚假实现的环境或真实环境。尽可能的也自动化。----可以理解为模块测试、集成测试
大型：覆盖任意多的模块，一般运行在真实的环境中，并使用真正的用户数据和资源，或是通过自动化，或是探索性测试，着重用户测。------可以理解为系统测试、发布测试

小型中型--set介入多
中型大型--te介入多

8、《Google软件测试之道》的笔记-第179页

工程师只要在当前的项目工作了18个月以上就可以自由离开，而不需要获得当前经理和未来经理的许可负面因素是有经验的员工也可能转到其他团队。这对测试工程经理提出的要求是不能过于依赖于某些成员。不能仅仅依赖于某位明星测试人员，那些促成这位测试人员成为明星的东西，必须要沉淀成可用的工具，或者总结成一套方式，这样就可以帮助其他人也能走上这条成为明星的道路。

对于非明星软件公司，这也是可借鉴的相当聪明和有魄力的管理方式。既然人员变化总是存在可能的，不如积极面对，以免风险发生时，除了加薪许诺外就不知所措。

项目被个别人绑架，不但增加了人员流失带来的风险，项目本身也容易僵化，而个人被项目绑架后，发展也越发有限，从而又增加了流失的可能性。

9、《Google软件测试之道》的笔记-第76页

一种面向用户的测试角色，用户开发者
职责基本描述：
1、测试计划（ACC特质 组件 能力）和风险分析（两个因素：失败频率【罕见、少见、偶尔、常见】影响【最小、一些、最大】GTA： google test analytics）
2、评审需求、设计、代码和测试
3、探索式测试
4、用户场景
5、编写测试用例（GTCM google测试用例管家）
6、执行测试用例
7、众包crowdsourcing
8、使用统计
9、用户反馈

google的bug管理流程特殊之处：
1）数据库完全开放
2）所有人都提交bug
3）不存在正式的自顶向下的确定bug优先级的流程

google feedback（百度倾听？？）

TE
能够在已有的代码中寻找错误，迅速理解可能的软件失效模式，但并不关心从头编写这块代码或者做修改
更愿意到slashdot和news.com去阅读其他人的代码
会阅读一份未完成的产品规格说明书，添加剩余部分，完成这份文档
梦想参与的产品带给人们的生活带来巨大的影响
惊骇于某个网站的用户界面，怀疑它怎么可能会有用户
为数据的可视化感到幸福不已
发现自己狠乐意在现实世界里和人交流
心中的领导力：辅助其他工程师的创意
当被询问产品是否可以上线时，你可能会说：我觉得可以了

10、《Google软件测试之道》的笔记-第9页

看了序和前言，很幽默，很幽默啊！

11、《Google软件测试之道》的笔记-第178页

测试经理
作为独立贡献者的一个技术岗位，负责所有支持团队（开发、产品管理、产品发布、文档）之间的联络。需要同时具备SET\TE的技能。

第一 去了解你的产品 第二 知人善用

google员工通常每隔18个月就可以自由选择一个不同的项目，工程经理可以发布空缺职位

确保测试团队有影响力是测试工程经理的责任

每个工程师的个人目标都应该是建立影响力。年度评审和晋升中，影响力是一个非常重要的因素。

创造价值，并持续创造价值

google流程中的致命缺陷：
1）测试成了开发的拐杖
2）开发和测试的组织结构分离，测试人员更关注自己的角色，而不是他们的产品
3）测试人员往往崇拜测试产物胜过软件本身
4）产品经过 最严格的的测试发布以后，用户多大可能能仍然发现测试中遗漏的问题，答案是：几乎必然发现

SET的未来————开发
TE的未来——安全工程师，测试活动的管理者、测试设计师
我们熟知和喜爱的测试方式即将终结。敏捷开发、持续集成、早期用户介入、众包测试、在线软件交付

12、《Google软件测试之道》的笔记-第128页

google的管理核心：领导力和洞察力、协商、外部沟通、技术水平、战略规划、招聘和面试、完成团队绩效考核

13、《Google软件测试之道》的笔记-第16页

提升代码可测试性，提供自动化测试建议，构建持续测试环境

测试认证 类似敏捷CI成熟度模型 分为5级

14、《Google软件测试之道》的笔记-第141页

在测试上难以自动化的软件，很难成为好的软件。
除此之外，安排好优先级，寻找小成本大回报的自动化项目。一定要记住自动化并不能解决所有问题，尤其是前端项目和设备测试。

15、《Google软件测试之道》的笔记-第134页

质量机器人：Quality Bot
chromebot：利用数据中心的空闲机器周期，在成千上万的虚拟机上启动大量的chrome浏览器，去爬取成百上千万的URL，初期，用于发现崩溃问题非常好使，后期数量减少，启动另外一个DOM。
BITE：Browser Intergrated Tes Enviroment 浏览器集成测试环境，把尽可能多的测试活动、测试工具、测试数据集中到浏览器和云里，并在上下文中呈现相关信息，从而减少分散操作的麻烦，使得测试工作

更高效。

chrome OS测试计划：
1 基于风险
2 自动化硬件测试组合
3 支持快速迭代
4 开放测试用例和工具
5 chrome os的主要浏览器平台
6 测试提供数据
7 可测试性和乘数效应

测试团队推动风险分析，达成以下目标：
1 保证产品的质量风险被周知
2 保持测试团队始终仅关注ROI最高的任务
3 保证存在一个质量和数据评估框架————————质量标准？？

chrome的漫游测试（？？还不是很理解）
购物漫游
学生漫游
国际长途电话漫游
地标漫游
通宵漫游
公务漫游
危险地带漫游
个性化漫游

16、《Google软件测试之道》的笔记-第13页

如果能够自动化，并不需要人脑的睿智与直觉来判断，那就应该以自动化的方式实现

17、《Google软件测试之道》的笔记-第5页

质量不是被测试出来的，但未经测试也不可能开发出有质量的软件
测试是开发过程中比不可少的一部分，当开发过程和测试一起携手联姻时，即是质量达成之时（持保留态度）

三种google中的角色：swe 软件开发工程师 set软件测试开发工程师 te测试工程师————————工程生产力团队 EP

18、《Google软件测试之道》的笔记-第158页

我首先会让我的团队思考，“对被测系统来说，什么是最为重要的东西？”对搜索来说是性能，对新闻来说是时效性，对地图来说是综合性和完整性。每个应用都有其最重要的属性。类似的，对系统基础架构来说，数据完整性对存储最为重要，可扩展性对网络系统最为重要，利用率对任务管理系统最为关键。当你分清了你要测试的特定产品的关键因素以后，就要把你的大部分精力集中在检验系

统的核心能力是不是能够满足这些关键属性要求上。
当这些重要的事情搞定以后，再去关心那些简单的事情（用户界面这些锦上添花的东西）。还要关注那些核心的不容易改动的方面（如性能设计），而不对那些很容易修改的方面花费太多精力。如果你过早报告关于字体的bug，我就会担心你是不是没有搞清楚事情的优先次序。

19、《Google软件测试之道》的笔记-第25页

审阅设计文档时应该有一定的目的性
point：完整性、正确性、一致性、设计、接口与协议、测试

20、《Google软件测试之道》的笔记-第12页

mock对象是指对外面依赖系统的模拟，在运行时刻可以根据假设的需求提供期望的结果。fake对象是一种虚假的实现，内部使用了固定的数据或逻辑，只能返回特定的结果。
对这两个不是很理解……

21、《Google软件测试之道》的笔记-第6页

质量不等于测试。当你把开发过程和测试放到一起，就像在搅拌机里混合搅拌那样，直到不能区分彼此的时候，你就得到了质量

22、《Google软件测试之道》的笔记-第127页

TE面试的最后一环是看候选人是否具有“ Google味儿 ”。拿自己面试中常问的问题和Google味儿匹配一下:
1. 有好奇心，充满热情的工程师，不满足于简单完成分派的工作。
问题：有没有什么工作或任务，当时因为时间或者其他原因，很遗憾没有做好的？如果现在有机会，你打算如何做？

2. 与现实世界和计算机科学团体紧密联系的人，如参与开源项目，或者通用化自己的工作以提高复用性的人。
问题：如何获取业界资讯？有没有订阅或者follow的博客，微博，微信或者常去的社区？你开发必备的工具是什么？

3. 与他人和睦相处，合作愉快的人。
问题：你理想的团队什么样？

4. 愿意持续学习和成长的人。
问题：如果现在你有一段空闲时间，你打算学习什么或者读什么书？

5. 带来新鲜思想和经验的人。
问题：针对高级人员，如果让你负责一个项目的技术或者团队，你打算如何做？

23、《Google软件测试之道》的笔记-第9页

质量不等于测试，没有哪个软件质量是测出来的，如果开始就是错的，怎么测试也不能保证质量是正确的。但是，没有测试过的软件质量也是无法保证的，测试时保证软件的重要方法和实践，但不

是决定性的以及根本性的。

软件开发的三个角色：
SWE（Software engineer）：软件开发工程师，其关注对象是需求，功能，如何实现它。
SET（Software engineer in Test）：软件测试开发工程师，其关注的对象是SWE以及其生产出的功能代码，单元测试，测试文档等，主要工作是提供测试框架，使SWE的工作更高效。
TE（Test engineer）：测试工程师，其关注对象是最终的系统，从用户的角度去测量软件质量，并验证SWE以及SET的测试。

资深管理者一般都来自产品经理或开发经理，而不是来自测试团队。

测试团队独立存在的部门，是与其他专注领域团队平行的一个组织，其成员以租借的形式进入产品团队。其会根据产品的优先级，复杂度，并与其他产品实际比较之后，来决定分配测试人员。

在Google有一个广泛接受的做法：对于一个测试人员，如果在某个产品中工作满18个月，就可以无理由地自愿转岗到其他产品，当然，这个转岗不是强制性的。

24、《Google软件测试之道》的笔记-第14页

通过使用定位点击的验证方式、录制技术等可以把一些手动测试转变成自动化测试，这些自动化测试在每次建立之后都会重复地回归运行，而手动测试更倾向于关注于新功能。
现在所接触到的测试基本都是手动测试，想多学习一点关于自动测试的方法，这里提到的定位点击的验证方式、录制技术，之后找时间多了解一点

25、《Google软件测试之道》的笔记-第148页

我喜欢由快速迭代和高质量带来的挑战。这两者相互矛盾但又都很重要。我喜欢由快速迭代和高质量带来的挑战。这两者相互矛盾但又都很重要。我喜欢由快速迭代和高质量带来的挑战。这两者相互矛盾但又都很重要。

《Google软件测试之道》

版权说明