Broadview®
www.broadview.com.cn

测试馒头铺从0到1职业规划丛书

10 年软件测试管理经验，汇成 52 个问题，助你职场顺利进阶

基于问题驱动模式，根据具体应用场景构建解决问题所需的知识

软件测试进阶之路

测试路上你问我答

何飞◎著

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

测试馒头铺从0到1职业规划丛书

软件测试进阶之路

测试路上你问我答

何飞◎著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

如今，众多的互联网产品企业不再仅仅重视用户交互体验和运营推广渠道，也越来越重视产品的质量，所以软件测试岗位的受重视程度自然也就水涨船高，越来越多的同学正在前往或者已经走在软件测试这条路上。

而本书不同于市面上大多数测试理论知识的集合类书籍，它以软件测试的职业发展道路为主线，按不同的工作年限为阶段划分，再围绕各个阶段最常遇见的实际问题，通过问答的形式将解决问题的思路、背景知识、实际应用方法一一道来，读者可以带着具体的问题，也可以根据自己所处的工作阶段来阅读这本书，参考问答的思路去解决自己的实际问题。

图书在版编目（CIP）数据

软件测试进阶之路：测试路上你问我答 / 何飞著. —北京：电子工业出版社，2018.4
（测试馒头铺从0到1职业规划丛书）
ISBN 978-7-121-33850-2

Ⅰ. ①软… Ⅱ. ①何… Ⅲ. ①软件－测试 Ⅳ.①TP311.55

中国版本图书馆CIP数据核字（2018）第048227号

策划编辑：王　静
责任编辑：牛　勇
特约编辑：赵树刚
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：720×1000　1/16　印张：12.25　字数：192千字
版　　次：2018年4月第1版
印　　次：2018年4月第1次印刷
定　　价：49.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序

一年前，我出版了一本教材《软件测试——基于问题驱动模式》，这本书特意加了一个副标题——“ 基于问题驱动模式 ”，这是因为考虑到软件测试所要解决的问题相对复杂，依赖于复杂的情景（上下文）——项目背景、软件研发技术和流程、实际业务的应用场景等。所以，在软件测试工作中，我们不能通过简单地提取已有知识来解决实际问题，而是需要根据具体应用场景来构建解决问题所需的知识。而对应的学习方法就是今天所提倡的“ 建构主义（Constructivism）”，它是学习理论从行为主义发展到认知主义后进一步发展的结果，强调学习者的主动性在建构认知结构过程中的关键作用，认为“ 情境 ”“ 协作 ”“ 会话 ” 是学习环境的关键要素，注重和应用背景的紧密结合，强调在实践（实验）中学习。现在，我要推荐的这本书和之前提到的那本书有相同之处 ：问题驱动——作者先摆出问题，然后逐一回答。之前许多软件测试从业人员基于自己所面对的困境（相当于“ 背景 ”“ 情景 ”）提出问题，作者根据自己十几年积累的实际经验来回答这些问题，以此帮助读者构建软件测试所需的知识。

作者和我共事多年，在工作中不仅认真积极，而且善于思考、总结经验，正如作者在前言中认真地介绍“ 什么是知识 ”、区分知识和信息、进行 3W 分析来帮助读者重新认知学习。3W 分析其实就是回答下面三个问题 ：

- What is the problem?（真正的问题是什么？）
- What is the root cause?（根本原因是什么？）
- What is the solution?（解决方案是什么？）

3W 分析是一种优秀的实践方法，即碰到任何问题，不要急于解决它，不要只

看表面现象，头痛医头、脚痛医脚，而是要明确真正的问题，找出根本原因，这样才能彻底地解决问题。本书基于这样的思路来组织内容，可以更有效地帮助读者解决问题。下面简单地举几个例子来说明本书是如何做的。

第一个例子，本书把问题归为积累，不是按照通常的软件测试知识来进行归类的，如测试计划、测试设计、测试执行，或者单元测试、系统测试和验收测试等，而是基于读者的背景（从事软件测试工作年限、想解决什么问题）来进行归类的。

- 跨进软件测试之门（适合工作年限 ：0 ～ 2 年）。
- 在软件测试之路上越走越好（适合工作年限 ：3 ～ 4 年）。
- 是走 “ 管理 ” 路线还是走 “ 技术 ” 路线（适合工作年限 ：5 ～ 7 年）。
- 成为资深软件测试专家（适合工作年限 ：7 年以上）。

刚开始从事软件测试工作的读者最关心的是如何成为一名专业的软件测试人员，工作几年之后最关心的问题是如何成长为优秀的软件测试人员。之后的成长可能会有困惑，因为再往前，前进的道路就有分叉——是走“ 管理 ”（当“ 官 ”）路线还是走 “ 技术 ”（当 “ 专家 ”）路线，这是每个职场人都需要做出的选择。

第二个例子，针对每个问题，本书会先给出背景，也就是让读者理解问题的上下文，为什么会有这样的问题？在什么情况下会问这样的问题？谁最有可能会问出这样的问题？这样读者就比较容易理解问题及其解决方法，从而将所学的知识和工作中的场景联系起来，让所学的知识得到良好的应用。

第三个例子，有的人问题问得可能不清楚，如 “ 应该先学习软件测试技术方法还是工具 ”，作者首先答复 “ 不能一概而论，得先看看是什么软件测试技术，再看看学习的对象是 ‘ 小白 ’ 还是 ‘ 老兵 ’”，然后就接口测试、性能测试等实际环境来讨论这个问题。这样容易抓住问题的本质，“ 其实他们是被工具的复杂度难倒了，而不是被接口测试本身难倒了 ”。找到了问题，再就问题进行分析解答。在特定环境下，如接口测试，就应该先掌握测试的方法，然后再学习工具。工

具的种类繁多，不同的工具设计是不一样的。但万变不离其宗，方法是基础，掌握方法是首要的。掌握了方法，就更能理解工具，用什么工具也不会觉得困难，掌握工具也会相对快一些。

本书不仅仅讨论软件测试技术，而且还讨论软件测试管理 ；不仅仅讨论传统的软件测试，而是更多地讨论“ 敏捷测试 ”，包括敏捷测试的软着陆、高性价比的敏捷落地方案等。本书也不局限于软件测试，而是深入职场 —— 讨论软件测试人员的职业发展，例如，前面讨论的“ 是走‘ 管理 ’路线还是走‘ 技术 ’路线 ”就是一个职场人普遍会遇到的问题。本书还讨论了其他一些有趣的问题，如“‘ 老兵 ’混职场 ”，并给出打破职业发展瓶颈的终极绝招。

本书覆盖了软件测试人员的“ 一生 ”，适合不同层次的测试人员阅读，虽然不是全部内容，但也值得我们学习和思考。对于软件测试新人来说，有困惑时拿起它，读几个问题，也许它可以陪伴你走完职业之路。

朱少民

一位软件测试老者

前　言

在职场中如何正确学习

学习，是我们将信息转化为知识的一种行为。但在进入职场之后，很多人变得不太会学习了，或者说不知道该怎样正确学习了。

既然我们已经清楚了问题是什么，就一起来看看产生这个问题的根本原因是什么吧。

在此之前，我们先来了解一些基本概念。

信息，指音讯、消息、通信系统传输和处理的对象，泛指人类社会传播的一切内容。

知识，指符合文明方向，人类对物质世界及精神世界探索的结果总和。一条陈述能称得上知识，必须满足三个条件 ：它一定是被验证过的、正确的，而且是被人们相信的。

换一种简单的说法，能够改变你的思维模式和行为习惯，对你以往的输出结果产生了影响的就是知识。

信息只有在被有效地处理后才会变成知识。

3W 分析法

了解了以上概念后，下面再对前面所提出的问题（在职场中如何正确学习）进行 3W 分析。

真正的问题是什么

为什么在进入职场之后，很多人找不到有效的、正确的学习方法？

根本原因是什么

当我们在学校里学习时，因为有对应的问题要去解答，也就是说学了就能立即实践、验证，所以，我们能很容易地将课本中的信息转化成自己的知识。

在进入职场之后，很多人还习惯性地使用学校里的学习方法，对于一本书，不管是不是所有章节都对自己有用，都会要求自己一定得读完，即要求自己必须按照书中完整的结构体系去学习。但是，这种方法恰恰忽略了学习的真正目的 ：解决问题。

也正是因为缺少了“ 问题 ”这个核心，从而导致看似完整的学习实际上却是支离破碎的。因此，我们很难将获取到的信息有效地和自己的认知关联起来，从而形成知识。

解决方案是什么

这也就是我写作本书的初衷和目的。本书不同于市面上大多数的软件测试理论类书籍，并不是按照某个知识领域的脉络体系展开阐述的，而是以软件测试这个行业的职场发展为主线，以不同的工作年限为阶段划分，再围绕各个阶段最常遇见的问题，通过问答的形式将解决问题的思路、背景知识、实际应用方法一一道来。读者可以就问题查问题，根据答案解决自己遇到的实际问题。

如何阅读本书

本书是按照软件测试从业者在职场上所经历的几个阶段来划分的。

Part 1 ：跨进软件测试之门

这部分内容主要面向刚毕业求职的同学和刚刚入职的新人。围绕职业目标定位、软件测试入门前需要了解的知识点，以及在求职过程中与简历、面试和试用期息息相关的一系列实际问题来介绍相关内容。

Part 2 ：在软件测试之路上越走越好

这部分内容主要面向已经进入软件测试行业 3 ～ 4 年的读者。在他们所经历的职场黄金期和倦怠期中可能会面临一些问题，这部分内容针对这些问题帮助读者顺利度过职场倦怠期，并且介绍了相关领域的一些知识。

Part 3 ：是走“管理”路线还是走“技术”路线

这部分内容面向那些已经在这个行业中摸爬滚打了 5 ～ 7 年，面临转型期的读者。他们在转型期会遇到一些问题，而这部分内容从实际出发总结了一些建议，希望能帮助他们在这个时期摆脱迷惘。

Part 4 ：成为资深软件测试专家

度过了转型期，每个人肯定都希望自己成为这个行业的专家。这部分内容针对这类人群，根据笔者的经验给出了相对接地气的建议。

关于这本书的阅读建议，就像笔者关于职场学习的理解和认知，并不建议读者一上来就一定要通读全书，而是应该先看一下自己所处的职场阶段，再将自己在这个阶段中所遇到的问题罗列出来，带着问题翻开这本书，去寻找是否有相应的或者相关的解决思路和方案。

如果本书有你想要的，那么甚好！如果没有，那么你问，我答！

作 者

目　录

PART 2
在软件测试之路上越走越好 适合工作年限：3~4年 / 57

PART 1
跨进软件测试之门

适合工作年限：0～2年

第 1 章　如何找到第一份令自己满意的工作

问答（1）如何快速定位职业目标?

前两天有同学问我：怎样才能加入这家公司呢？怎样才能把它的招聘要求转化成自己奋斗的具体成长要求呢？它的要求都很笼统。

岗位职责：

（1）负责 ×× 应用程序产品的测试工作。

（2）独立执行项目测试，包括需求讨论、设计评审、策略制定、资源分配、设计并执行用例、进行缺陷跟踪和软件质量分析等。

（3）在项目中保持和产品经理、开发工程师、UI 设计师等成员的积极沟通，推动问题解决。

（4）进行测试技术改进，结合手机应用程序的特点设计开发自动测试工具，提升测试的质量和效率。

任职要求：

（1）具有两年以上软件开发或测试工作经验，熟悉互联网产品的研发流程和质量标准。

（2）能快速、深入理解系统内部的工作原理，有对测试需求做透彻分析、对 Bug（缺陷）进行清晰描述及快速且准确定位原因的能力。

（3）有扎实的数据分析能力和数据敏感度。

（4）有测试或项目管理经验，电影爱好者优先。

（5）有责任感、团队精神、良好的沟通能力和推动能力。

如何将招聘要求转化为自己的学习目标？

根据这个岗位的岗位职责和任职要求，稍作分析，不难得出其基本要求如下：

（1）熟知项目流程，从参与需求评审，一直到缺陷的跟踪管理，包括测试策略的制定和测试计划的制订，以及用例的设计和执行，都要熟悉。

（2）具备良好且有效的沟通能力。这是必备能力，因为你对外要跟产品经理、开发工程师沟通，对内要跟项目组的测试成员沟通。

再针对这位同学的能力现状，找出需要有针对性提高的能力，大致如下 ：

（1）依据需求文档和开发的设计文档，绘制产品业务的逻辑图和数据流图，“ 设计 ”测试场景或测试用例，而不是“ 写 ”测试用例，通过这种方法刻意地练习，提高自己的逻辑分析能力。

（2）要熟悉且熟练应用最基础的测试理论，重点是测试计划的制订、用例的设计、缺陷管理和数据分析。

（3）对于服务器端的接口测试和性能测试，需要先了解基础知识，至少知道它们是什么、可以借助什么工具进行测试，后期在工作中边用边学。

（4）在精力和时间允许的前提下，学习一下 Python。

我认为，这种找寻学习目标的方法十分有效。之前我写过一篇文章，题为《“ 老兵混职场 ” 之快速定位职业目标的两个大招》，目的就是引导读者在对职业目标或规划不清晰的时候，快速寻找和定位目标。

（1）打开智联招聘、拉勾网或者 51job。

（2）搜索你所从事的工作的岗位，或你期望的下一阶段的目标职位。

（3）按照职位名称、工作地点和薪资等维度筛选出相对高阶的几家公司岗位。

（4）依次把职位描述里的岗位职责和岗位要求复制到一张表格里。

（5）挑出每家公司都有的常规岗位要求。

（6）挑出薪资较高的公司列出的一些额外岗位要求。

（7）把要求里的技能、知识和素养拆分开，形成一个列表。

（8）跟自己的实际情况逐项进行对比，不具备的就打叉，具备的就根据实际匹配程度打分。

（9）将所有自己不具备的和得分较低的项单独列出来，按照难易程度、与当前工作的契合度等进行优先级排序。

（10）按照优先级制订阶段性的目标和计划。

有些招聘要求的描述看上去比较空泛、模板化，这其实也是很常见的，因为大多数用人单位在让人力资源管理部门招人的时候，一般只提交了招聘岗位，而对于岗位说明，如果没有特殊的技能要求，则一般是通用的。

所以，如果你想了解一些看上去比较笼统的职位说明到底要求具备哪些能力，则可以找几家公司面试一下，通过和面试官的交流去了解你不清楚的职位要求，这也算是变相地对自己当前能力的一种考验吧。

问答（2）在不同的维度划分下到底有多少种测试分类?

背景

我经常被问及测试都有哪些类型，有的是因为在面试中被问到，有的是因为自己在学习中被各种名词所困扰，也有的是因为在与同行交流时所用的叫法不同而迷惑。

我认为，问题本身所想了解的东西其实不重要，比如，对于每个提测版本所进行的快速验证测试，到底是应该叫冒烟测试，还是叫 BVT 测试，重要吗？我们应该关注的或者说应该明白的是在不同的阶段或者针对不同的目的，应该进行哪种类型的测试，知道该类型的测试应该怎么做。

不过，我们仍然需要对测试进行分类，这样才易于沟通和学习。我先按照自己的理解和认知进行了分类，后续再一一展开探究。

你问

在不同的维度划分下到底有多少种测试分类？

我答

1．按软件开发流程的不同阶段划分

（1）需求阶段。

需求测试 :对产品需求进行的测试，包括逻辑错误、需求合理性、需求缺失、需求建议等。

（2）编码阶段。

单元测试 ：对软件的最小组成单元（比如某个函数、方法）进行的测试。

集成测试 ：通常也叫联合测试，指的是在完成单元测试的基础上，将所有模块按业务需求组装起来进行测试。

（3）测试阶段。

系统测试 ：将已经完成集成测试的软件和硬件、网络等其他元素结合在一起，进行系统的各种组装测试和确认测试。系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义。

（4）发布阶段。

验收测试（Acceptance Testing）：是指在产品完成系统测试之后、产品发布上线之前所进行的测试活动。在合同类的项目中，验收测试也被称作交付测试。

2．按测试视角划分

（1）外部视角。

黑盒测试 ：指的是在测试过程中只关注程序的输入和输出，一般用于系统测试阶段。

（2）内部视角。

白盒测试 ：指的是在测试过程中不仅关注程序的输入和输出，还关注程序内部的处理逻辑，一般用于单元测试阶段。

（3）内外兼顾。

灰盒测试 ：顾名思义，就是在测试过程中黑白兼顾的测试方法，一般用于集成测试阶段。

3．按测试对象的属性划分

（1）GUI（图形用户界面）测试 ：指的是对软件的 GUI 进行测试。我认为，

在现如今的 APP 测试里，GUI 测试包含了 UI 测试和适配测试。

（2）功能测试（Function Testing）：这是我们最常见的测试，是为了验证产品是否符合产品需求规格说明书而进行的最基本的测试。

（3）易用性测试（Usability Testing）：对现在的 APP 来说，易用性测试指的是检查系统的交互和界面对用户来说是否友好且易用的测试活动。

（4）接口测试：指的是测试系统组件间接口的测试活动，用于检查模块之间的参数传递是否符合设计文档要求。

（5）性能测试：指的是通过工具模拟多种正常、峰值及异常负载条件来对系统的各项性能指标进行测试，包括负载测试、压力测试、稳定性测试等。

（6）安全测试：指的是验证系统的安全等级和识别潜在的安全性缺陷的过程，目的是找出系统里的安全隐患和漏洞。

（7）兼容性测试：指的是验证软件在不同的硬件、操作系统、浏览器、数据库下是否都能正常运行的测试活动。对 C/S（客户端 / 服务器端）结构的产品来说，还包括服务器端和客户端的兼容性测试。

（8）国际化测试：指的是验证系统是否支持国际化特性的测试活动，包括多字节字符集的支持、区域设置、时区设置、界面定制性及字符串编码等。

（9）本地化测试：指的是验证翻译后的系统，其语言、格式、内容和用户界面是否都正确的测试活动。

4．按测试对象是否处于运行状态划分

（1）静态测试：指在不需要被测试系统运行时所采用的一种测试方法，比如代码静态扫描、代码评审、文档评审等。

（2）动态测试：指在需要被测试系统运行时所采用的单元测试方法或者系统测试方法。

5．按执行测试的角色划分

（1）手工测试 ：指由软件测试工程师手工执行完成的软件测试活动。

（2）自动化测试 ：指由测试工具或测试脚本来执行完成的软件测试活动。

6．按测试的行为划分

（1）计划性测试 ：顾名思义，指的是在测试执行之前，需要进行详尽的测试计划、测试分析、测试用例设计，并按照计划通过执行测试用例来完成的测试活动。

（2）随机测试 ：又叫 Ad-Hoc Testing，顾名思义，就是没有计划的、不按测试用例的、即兴发挥的自由测试。

7．按测试原因划分

（1）新功能测试 ：指的是针对产品新功能的验证所进行的测试。

（2）回归测试 ：指的是因为产品增加了新功能或修复了 Bug（缺陷）而进行的确保原有功能不会受到影响的检查测试。

8．按测试范围划分

（1）冒烟测试 ：我认为也可以等同于 BVT（Build Validation Testing），指的是对测试包所进行的快速可用性验证测试，一般只会覆盖系统核心功能或者所有功能的核心测试点。

（2）全面测试 ：指的是对系统所有功能的所有测试点进行的测试活动。

问答（3）什么书适合软件测试入门者?

背景

一天，一位刚转型做测试的朋友问我：“老师，如果让您就每个领域推荐一本最适合入门的书，您的回答是什么呢？例如，数据库、编程、性能测试、网络协议、Linux 等。”听到这个问题，我当时反问了他一个问题：你为什么需要同时入门这几个领域？因为我认为它们都是独立的知识领域，而且都不是轻量级的，从个人精力上来看，不建议同时去学习它们。但今天还是就书籍推荐的问题来说说吧。

你问

什么书适合软件测试入门者？

我答

这其实是很多人在跨入一个新的领域时最喜欢问的一个问题，但也是很多人都不太想回答的一个问题，一是因为问得太宽泛了，无从答起 ；二是因为背景信息不够全面，很难回答。

我今天也不准备直面这个问题去回答，而是换个角度来回答，希望既能回答这个问题本身，又能告诉你以后该如何有效地提问。

我是如何向别人推荐与测试相关的书籍的呢?

我会先问他几个问题，如下 ：

1．你的目的是什么

你想找这类书来读是为了什么？

你准备进入这个领域，想读这类书做一些了解吗？

你已经入门有一段时日了，想通过读这类书来系统地梳理一下自己零散的知识点吗？

你已经在某个方向积累已久，觉得是时候发展一个新的方向了，但不知道自己该发展哪个方向吗？

你想知道是性能测试、自动化测试好，还是安全测试好吗？

2．你当前的角色是什么

你是刚毕业的学生吗？

你是转行来做测试的吗？

你是经过测试培训的实习生吗？

你是从事测试已经三个月或半年的测试工程师吗？

你是在测试行业至少做了 5 年的测试组长或测试项目经理吗？

3．你当前的舒适领域是什么

换句话说，你当前擅长、熟知的领域是什么？

如果你是学生，那么你所学的专业是什么？

你当前从事的行业是什么？

你当前从事的是黑盒测试、白盒测试，还是自动化测试？

4．你的职业期望是什么

你是想快速找到一份软件测试工作吗？

你是想成为一名合格的软件测试工程师，胜任领导分配的测试任务吗？

你是准备走测试技术路线吗？

你是准备走测试管理路线吗？

下面附上一些推荐书目（非测试技术类，另注：如果可以，则尽量阅读原版的引进书籍）。

1．入门

（1）《全程软件测试（第 2 版）》/ 朱少民 / 电子工业出版社。

（2）《软件测试（原书第 2 版）》/ Rom Patton / 机械工业出版社。

（3）《软件测试过程管理（原书第 2 版）》/ Rex Black / 机械工业出版社。

2．中阶

（1）《Google 软件测试之道》（*How Google Tests Software*）/ James Whittaker 等 / 人民邮电出版社。

（2）《完美测试：软件测试系列最佳实践》/ 朱少民 / 电子工业出版社。

3．高阶

（1）《敏捷软件测试：测试人员与敏捷团队的实践指南》/ Lisa Crispin 等 / 清华大学出版社。

（2）《测试架构师修炼之道：从测试工程师到测试架构师》/ 刘琛梅 / 机械工业出版社。

问答（4）看了我的简历，您想约吗？

背景

有位同学想从手机测试转型做互联网应用程序测试，但因为手机测试经验占比较重而未能获得更多的面试机会或者说面试通过概率较低。于是他把简历发给我，并问了我一个问题：“老师，如果您看到我的简历，会有想法通知我来面试吗？”

我帮他看了一下，结合他想应聘应用程序测试工程师岗位的需求，针对简历里的问题做了一些分析。今天就先说说简历的事儿，后面再说说有关面试的那些事儿。

你问

怎样才能提高简历的命中率？

我答

1．意向岗位：测试经理 / 主管、系统测试、手机测试

定位不明确，意向模糊。针对测试主管和系统测试本身而言，一个是管理岗位，一个是技术岗位，不建议放在同一版本的简历里，建议拆分开，并在“工作经历”栏里有针对性地突出相应的技能和经验，这一点很重要。“技术岗”就重点突出你的专业技能，“管理岗”就重点突出你的项目管理和团队管理能力。

因为从 HR 或用人部门的管理者角度来看这份简历，会认为你首先就没有清晰的定位。假如用人部门其实想招聘一名测试工程师，但看了你的意向之后，会产生这样一种担心：你倾向于做测试主管，那么是否很难再沉下心来去做功能测试呢？或者说你因为写了这个意向，所以会罗列很多这方面的经验和技能，

但这根本就不是用人部门想看的。

2．三家公司的工作经历 + 三个项目的项目经验

工作内容

在该公司主要负责两部分内容 ：

（1）测试管理。

（2）手机终端、定位管理平台网站及应用程序的功能测试，用户体验测试。

具体工作内容包含 ：

（1）测试流程建立，测试团队组建，相关测试文档编写，日常测试管理。

（2）Bug 系统 —— Bugzilla（一种缺陷管理工具）的使用，Bug 工作流程建立，规范制定等。

（3）产品需求评审，测试需求分析，编制测试方案和计划。

（4）测试用例设计和编写及评审。

（5）测试执行，提交有效 Bug，并对 Bug 进行跟踪及验证，同时进行回归测试。

（6）对产品质量负责，全程跟进产品质量 ；定位测试发现的 Bug，以及工厂或客户反馈的 Bug。

（7）使用 Monkey、LoadRunner 等性能测试工具执行压力测试与性能测试。

（8）入网，入库，IP 防护，防爆认证，北斗认证等认证自测与跟进。

（9）组织团队进行测试总结、新知识培训、Bug 解析。

项目职责

（1）参与需求的讨论，利用思维导图工具 XMind 列出测试功能点，进行测试需求的确认，并制订测试计划。

（2）根据需求文档编写测试用例并进行评审和完善。

（3）核验产品需求文档，执行用例测试、用户体验测试，提报 Bug 和跟踪 Bug，进行回归测试。

（4）使用 Emmagee 对应用程序进行 CPU、内存、流量和耗电量等测试。

（5）利用 LoadRunner 录制脚本，进行性能测试。

（6）测试结果统计、总结，风险评估。

这两部分内容存在两个问题 ：

第一，部分内容其实是重复的，例如，使用 LoadRunner 进行性能测试，测试用例设计、编写及评审等。

第二，内容表述相对模式化、大众化或者比较宽泛，没有很突出或者很吸引人的亮点。

个人建议对于“ 工作经历 ”和“ 项目经验 ”这两部分内容的表述应该各有侧重点，且互为补充。

工作经历

（1）描述一下这家公司的重点产品和团队规模。

（2）重点突出你在这个团队里的“ 关键 ”位置、承担的责任，以及由你主导的有关 应用程序类产品的测试工作。

（3）表述出你深度参与的、不会在“ 项目经验 ”里出现的卓越产物和成果。

（4）涉及应用程序测试较少或者无关的工作经历可以简单带过，甚至不写进去也行。

项目经验

（1）数量不宜过多，挑出 1 ～ 2 个应用程序类的测试项目。

（2）使用的测试流程如果没有特殊之处，则可作为概述，重点表述出关于应用程序类产品测试的测试场景设计、业务逻辑分析、交互和适配的用例设计。

（3）清楚地描述你了解的应用程序客户端压力测试工具 Monkey 的基本用法和作用，适当加上你所在的团队对参数所做的一些优化。

（4）对于服务器端性能测试工具 LoadRunner 的使用，不要简单地说一句“ 会用 ”，而要重点表述你在实际的项目中是怎么从性能需求分析、场景设计、脚本设计、数据准备、场景运行，一直做到报告分析的。在这里最好能加上你对当前互联网应用程序的产品，尤其是你去应聘的这家公司的产品的应用场景分析和性能需求建议。

（5）关于手机测试的经验部分，我认为，跟手机操作系统相关的测试可以重点突出，这部分经验在测试互联网类应用程序时其实是适用的。因为如果你对手机系统级别有较深的了解，则会更容易判断在哪些功能场景下应用程序容易出错，比如闪退和死机。所以，这部分经验的表述对你而言应该会起到正向作用。

（6）其余项目可以简单罗列其中，但如果跟应用程序测试无关，则建议省略。

3．自我评价

一般人写自我评价时都会这样表述 ：性格开朗，工作责任心强，有耐心，积极主动性较高，具有一定的抗压能力 ；有较强的学习、沟通、协调能力，做事细心、踏实，具有很强的团队精神和良好的合作意识。

站在面试官的角度，关于“ 自我评价 ”部分，其实看到上述这种通用型的表述，不会引起他太多的注意。建议在“ 自我评价 ”中多关联一些实例或数据来说明自己的某些特质，比如你带动团队做学习型分享，或者你在原有测试团队里会因为处理现网问题而跟多个部门经常有协作关系等，以此来衬托你的学习能力和沟通能力，以及团队精神和合作意识，这比直接用“ 良好的 ”“ 较强的 ”和“ 一定的 ”这种不易量化和衡量的形容词要好得多。

4. 技能专长

一般人会在技能专长部分中罗列出很多内容，例如下面这个案例。

（1）熟练掌握软件测试流程和方法，熟练进行功能测试、压力测试、性能测试、用户体验测试，完整地参与过多个大型项目。

（2）熟练掌握测试计划、测试用例和测试报告的编写。

（3）掌握 Linux 基本命令，如文件管理、进程管理、软件安装 / 卸载等 ；能在 Windows、Linux 系统下搭建环境（SVN、Tomcat、MySQL、DB2、VMware）。

（4）熟悉 SQL 语句，能根据测试需要对数据库进行增、删、改、查等操作。

（5）熟悉 Monkey 压力测试、LoadRunner 性能测试。

（6）对 Bug 具有较高的敏锐性，能够较好地保证产品质量。

（7）熟悉 Bugzilla 和 BugFree 等缺陷管理系统，熟悉 TestLink 用例管理系统。

（8）熟练使用 QXDM、FlashTool、MauiMETA、QPST、Catcher 等工具。

（9）熟悉办公软件 Word、Excel、PPT 的使用。

（10）做事细心、踏实，有较强的自学能力，能快速学会各种应用软件和

熟练掌握新技术。

就我个人的经验来说，技能专长一项罗列得过多，一是没有主次关系，让人力资源主管和面试官很难去芜存菁 ；二是即使你罗列的都是真实写照，也会让人觉得广而不精。

第（1）、（2）、（6）、（7）项应该属于测试工程师的基本能力，所以建议在“ 项目经验 ”中详细表述，会更具说服力。

第（3）、（4）、（5）项，如果你的确熟悉，则可以将相应使用的场景和应用范围写出来，目的是体现你不仅“ 学 ”了，也“ 用 ”了。

第（8）、（9）、（10）项，建议从此处移除，（8）是因为跟应用程序测试没有太大关联 ；（9）属于职场中人的基本技能，对你应聘应用程序测试工程师没有太大帮助 ；（10）属于自我评价项。

最后总结一下，我认为，设计简历有两个注意要点。

1. 因地制宜，量身定制

针对你心仪的公司和岗位要求，有针对性地定制简历，让人力资源主管或用人部门主管看到你的简历时第一反应就是：他就是我要找的人！如果能做到这一点，那么你会获得比其他竞争者更多的面试机会。因为不管你有多优秀、能力有多强，从应聘角度来说，都得先获得面试的机会，才有后续的可能。

2. 事实胜于雄辩

针对你应聘的岗位要求，不论是团队管理、项目管理还是测试技术类的岗位，如果你能附上自己独立梳理或者主导实施的一套可落地的具体方案，则远比模板化的流程和工具使用套路的表述更容易出彩，更容易让面试官动心。

问答（5）面试官，我要怎样您才会给我机会？

背景

还是那位想从手机测试转型做互联网应用程序测试的同学，因为手机测试经验占比较重而不能获得更多的面试机会或者说面试通过概率较低，上一篇说了简历的事儿，这一篇就说说有关面试的那些事儿。

你问

怎样才能轻松搞定面试官？

我答

1．面试前

（时间范围 ：始于收到面试通知，终于面试当天站到公司门口）

（1）基本功课要做足，对于应聘公司的背景、经营范围、产品类型和企业文化等都要有所了解，这些内容在公司官网或百度搜索里能够很容易地获取到。

（2）如果你把面试看成一次考试，那么考前要划的重点就是你应聘的岗位要求。我们的简历是因地制宜、量身定制的，那么在面试前就要在这个范围内深入准备。简历中的“ 项目经验 ”和“ 技能专长 ”应该是面试官比较感兴趣且发问的重点区域。

（3）关于项目经验或者具体方案，建议在面试前多进行几次演练，类似于老师在上课前的试讲。通过演练，不仅能找出其中明显的逻辑错误，让整个流程或方案更加真实可靠，而且增加了自己的熟练叙述程度，这样，在面试中也能尽量避免卡壳、结巴、条理混乱。别忘了重要的一点，就是一定要把方案所

带来的收益梳理清楚，尽可能数据化和可视化。

（4）对于自己比较熟悉或精通的技能专长，同样也可以事先演练一下，再挑出几个自己觉得比较满意的亮点或有价值的实践，在面试中如果被问及，则可以很自然地叙述出来，从而吸引面试官的注意力。

（5）对于自己仅限于“ 用过 ”的一些技能专长，可以系统地梳理一下自己以往的使用范围，查找资料或者实际操练一下，把之前理解得比较模糊的部分弄清楚，夯实一下，再从自己的角度找一下可能会被问及的问题，准备一下回答思路，以免在面试中被突然袭击，导致回答得模糊不清。

（6）对自己的入职时间有一个正确的估算，这主要针对在职人员，切记不要用随时或很快就能上岗的表述。个人建议 1 ～ 2 周是比较合适的时长，原因一方面是用人部门基本能接受的到岗时长最多也就 2 周，再长估计就会再去找其他能尽快到岗的人了 ；另一方面是你说随时或很快到岗，用人部门主管可能会感觉你的责任心不是很强，容易忽视工作交接。

（7）针对应聘软件测试工程师而言，可以不需要穿正装，但一定要整洁。不管有没有东西要装，建议携带一款相对正式的背包或挎包，不要空着手去面试，这样会给面试官一种很随意、随便的感觉。

（8）打印一份自己的简历，还可以带上自己主导完成的方案书或相关作品，所谓“ 事实胜于雄辩 ”。但**一定要隐藏之前项目的一些敏感信息**，否则会让面试官觉得你在职业操守上有严重问题。

（9）最后就是要守时。我的个人习惯是，不管是朋友相约还是工作之约，我都会至少提前 15 分钟到达约定地点。按照这个标准去制订时间计划，会避免很多意外情况发生，让自己有充足的时间去调整状态。

2．面试中

（时间范围 ：始于踏入公司大门，终于走出公司大门）

（1）当面试官进入面试区的时候，建议主动站起来示意，做一下自我介绍，不要坐在原地低头玩手机或者沉默不语，非要等面试官发问才说话（这几种风格我都遇见过）。切记！因为从这一刻开始，面试官就在观察你的沟通能力了。

（2）在做自我介绍时，不要照本宣科，因为那些内容简历上都有，面试官也肯定看过了。重点表述你跟应聘岗位匹配度较高的能力亮点，如果你不在这个时候表述出来，那么等进入“他问你答”阶段时，就未必有机会再去表述了。因为你不要指望面试官一定会关注到你简历里的能力亮点。

（3）要尽可能地做到控场。自我介绍其实就是一个最佳切入点，如果在这个阶段通过你的能力亮点吸引了面试官的注意力，那么后面的问题多半会围绕这个方面展开。既然是你自身的能力亮点，那么应该是准备得最充分的，这样就可以避免遇到你所不擅长的领域的问题。

（4）在介绍能力亮点时，尽量先表述你的该项技能或能力曾经有什么成果，再概述做的过程，重点突出成果所产生的效益，比如节省了多少人力、降低了多少成本、提升了多少效率等。这样可以让面试官从数据中得到一种暗示：如果你能把这项成果在这里落地，则也会产生同样的效益。如果你能做到这一点，而这项成果恰巧跟面试官近期的计划方向匹配，那就可以说你的面试成功与否只取决于薪资问题了。

（5）切忌打断面试官的问题。即使你在他问到一半的时候就已经完全理解了他的问题要点，也要听完他的整个问题，停顿几秒后再回答。这种情况在很多准备充分的同学身上反而更容易发生，因为对这个问题十分熟悉，所以就会习惯性地打断对方的提问。

（6）在回答问题时，最忌讳的就是不懂装懂，因为你不懂，不代表面试官也不懂，蒙混过关的概率几乎为零。比如，你的简历里提到了性能测试，但你也许只是基于别人搭建好的体系使用过，此时建议你可以委婉地回答，你们之前有专门的性能测试工程师去做性能测试和搭建体系，所以你只是有所了解，但并不清楚细节。

（7）在面试中后段，按照常规，面试官肯定会问你有什么想问的。建议多问问面试官关于发展空间的问题，比如，他对整个测试部门的规划是什么？他对你所应聘的岗位的定位是什么？根据面谈的感觉是否良好，你还可以问问，如果你有幸加入这个团队，那么团队准备让你负责什么、发展空间有多大，等等。

（8）在你提问的环节，建议不要把时间浪费在问一年有几次加薪、有没有旅游或外训福利这类问题上。你要清楚**面试的目的是什么，是通过面试官的考察检验。**在这个环节其实都还不确定你是否能通过面试，去问那些跟福利及薪资相关的问题没有太大意义，你大可在收到人力资源主管面试通过的结果时再与人力资源主管沟通，明确这些问题。

（9）单独强调一下，**最后的提问环节很重要!! 很重要!! 很重要!!** 千万不要说没什么要了解的。特别是对自己前半段的表现不是很满意的同学，可以充分利用这个环节，通过一些技巧性的提问吸引面试官的注意。即使前面表现不错的同学，也可以通过一些有深度的问题让面试官给你加分。

举例：想转型的这位同学说，面试官提问时直接跟他说“你没有应用程序测试经验啊”，他说“手机软件其实就是一个应用程序集成”，面试官又跟他说“不一样，着重点都不一样”，有的面试官甚至不给他深入阐述的机会。

建议：这时候，你不要跟面试官在这个问题上做过多的争论，你可以在其他你擅长的技能上跟面试官讨论。等到你提问的环节，你可以这样问：能了解一下，你们现在是通过什么工具进行应用程序耗电量测试的呢？面试官的回答要么是还没做过这方面的测试，要么是用了什么工具或方法。这时候，你就可以顺势利用你曾经做手机系统测试的专业技能，分析一下应用程序的哪些行为或功能会耗电，怎么能有效地检测出来等。如果你能有更专业的工具或方案，那就更好了。

这时候，你猜面试官会不会在没有应用程序测试经验这一项上对你的印象有所改观呢？

（10）所谓技巧性的提问，是指通过这个问题既勾起了面试官的兴趣，又

很好地展示了自己之前没有机会展示的能力。而不是给自己挖坑，去问那些的确很高深、但自己都不擅长的问题。

（11）在面试过程中，不要问那种带有挑衅色彩的问题，也不要表述过多带有个人感情色彩的评判性话题，因为你无法知道坐在你对面的面试官的想法是否跟你一样，搞不好，他是站在你的对立立场上的。

3．面试后

（时间范围 ：始于走出公司大门，终于得到确认答复）

如果是自己心仪的公司，那么可以在 1 ～ 2 天之后给人力资源主管打电话询问一下面试结果，表达一下你非常想加入这家公司的愿望，了解一下自己是不是哪里让面试官不太满意，询问面试官能否给出一些改进建议。这种主动型的面试者往往会给人力资源主管留下很好的印象，有时他们也会反馈给用人部门主管，这对你只有好处没有坏处。

按正常公司的效率，这个时间足够人力资源主管和用人部门的面试官沟通好几次了，不过也不排除是撒网式招聘，在几天内收集了很多面试对象的资料，需要全部面试完才能做出甄选。即使这样，也不妨碍你打电话确认一下。即使还需要再等，也相当于让人力资源主管加深了对你的印象。

如果说简历是一块敲门砖，那面试就好比试镜，能不能被导演看中，成为“男一号”或“女一号”，就看这短短半小时的个人秀了。总结几个关键点：

（1）准备要充分，不打无准备之仗。

（2）目的要明确，面试就是为了通过面试官这一关。

（3）心态要平和，控场要讲究技巧，切勿求胜心切，为达目的而不择手段。

问答（6）成熟型企业和初创型企业，第一次选谁比较好？

背景

最近不少同学在向我咨询简历和面试问题时有些疑惑，因为经验不足，简历没办法设计得太丰富，但又想找那种成熟稳定型的公司，可投出去的简历总是石沉大海。所以，下面来说说第一份工作的选择问题。

你问

第一份工作，我该选择什么样的公司？

我答

1．概念

初创型企业：泛指刚刚创立，且没有足够资金及资源的各类企业。这类企业往往都有资金短缺、人才匮乏、只有创始人及为数不多的核心员工、业务开拓吃力等问题。

成熟型企业：泛指成立时间较长，有着充足、稳定的资金，并且在市场上已经占有一定份额的大、中型企业。

2．初创型企业与成熟型企业，选择哪一个

（1）初创型企业因为规模小、**组织结构不完善，通常需要一人身兼多职**。而成熟型企业因为已经具有一定的规模，而且发展时间较长，所以**组织结构相对完整，通常是专人专岗**，而不可能是一人多岗。

从**个人成长空间**来说，在成熟型企业里，会有相对健全的培养体系，如新员工入职培训或者导师制等，而且也有规范的流程体系，做项目或做事都有条

有理、有据可依。但这也会限制你的学习实践空间，因为你只是这部庞大机器上的一颗小螺丝钉，公司只需要你聚焦在你所负责的这个点上，其他的点都有专门的人负责，你只要按标准的流程去做即可。而在初创型企业里，你可能需要身兼多职，而且在**很多领域你就像一张白纸，可以按自己的想法自由发挥、学以致用**，这对于一个人的综合能力会是很好的锻炼机会。

从**个人成长速度**来说，初创型企业不是专人专岗，而且前期属于快速增长期，每个人的任务都会非常重，而且很多时候分配任务并不会太多考虑你是会还是不会、是做过还是没做过，每个人都在边学边做，摸着石头过河。虽然会很累、压力很大，但是你会发现，如果一直保持这个节奏，那么你**每天都在进步**。

（2）初创型企业跟成熟型企业相比，在初期可能因为规模和资金储备方面的差距而无法在薪酬福利体系上体现太多的优势，有时甚至会成为劣势。

因为我们在这里说的是你的第一份工作的选择，所以，如果你没有太多的工作经验，那么在成熟型企业的薪酬体系里**起薪多半不会太高，顶多等于市场的平均价格**，以后就只能按部就班，每年有 1 ～ 2 次的调薪机会，大多数企业最高也就 10% 左右的调薪幅度。但如果你先在初创型企业里磨炼几年，等具有很强的综合实战能力之后再去选择一家成熟型企业，那你的资本就不一样了，对应的筹码也就增加了很多，你的薪资将会**高出市场平均价格**，而且跟你的实际能力绝对是成正比的。

也就是说，站在同样起点的两个人，A 选择了成熟型企业，B 选择了初创型企业，5 年之后，B 也应聘进入 A 所在的那家成熟型企业，其薪资可能比在里面工作了 5 年的 A 还要高。

初创型企业还有一个**高风险高回报**的因子，这也是近些年很多在成熟型企业里工作了很多年的人会接受一家初创型企业邀约的重要因素之一，那就是原始股。如果你所在的初创型企业有上市的计划，而你加入的时间又比较早，你就有相当大的机会通过贡献获得一定数量的原始股 ；当企业启动上市计划后，你就能获得相当可观的市场收益。这跟一些成熟型上市企业的员工激励股权所能获得的收益不是一个数量级的。

初创型企业还会有一些特色福利，比如人文关怀类的家人津贴、学习培训基金等。

3．我的建议

你可以选择以下两种路线。

A. 初创型企业→成熟型企业→初创型企业

B. 成熟型企业→初创型企业

如果这是你的第一份工作，那么，无论是从个人成长、发展空间还是从长期收益来说，我都建议选择 A 路线，优势我就不多说了。最后，我想再简单说一下 B 路线的一些弊端和注意点。

（1）**习惯了在完善和规范的流程体系下做项目**，很难有机会接触全流程，当形成了某种定势思维和行为习惯之后，再去初创型企业，因为没有规范、没有流程，往往就会无所适从，不知该从何下手。这时候一定要谨记，不能将之前公司的成熟体系生搬硬套到初创型企业里，因为工作环境因素不同，公司发展阶段也不同，这些都是直接影响流程体系建立的关键性因素。

（2）**习惯了成熟型企业相对安逸的工作环境和氛围之后**，再进入初创型企业，因为需要快速占领市场、打开局面，很多时候产品的发布和项目的进度不是依据工程师的评估，而是看市场需求的时间节点，所以在一些特定的阶段不可避免地会有高强度的工作量，需要大量加班，传说中的“ 996 ”工作制（早上 9 点上班，晚上 9 点下班，一周工作 6 天）其实在一些初创型互联网企业里真的只算是标配了，没什么大惊小怪的，但这需要你做好心理和生理上的充分准备。

问答（7）被问到是否能接受加班该怎么回答？

背景

我所在的是一家初创型企业，产品正处于不断试错、不断调整的阶段，加班可以说是家常便饭，所以我在面试中经常会问应聘者：“我们现阶段加班特别多，你能否接受？”有说能接受的，也有说能接受一周加班一次的，还有说接受不了加班超过晚上 9 点的，等等。这个问题如果放到日常的聊天中，则看似比较好回答；但如果放在面试中，其实就不那么简单了。

你问

面试时被问到是否能接受加班该怎么回答？

我答

1．回答方式

（1）我能接受加班，但我希望了解一下加班的原因是什么。通过提高自己的工作效率来减少无效的加班时长，可以吗？

（2）加班没问题，但我希望是有计划地安排加班，这样我可以很好地协调工作和生活。

（3）能接受，我之前的公司都是“ 996 ”工作制的，有时候还需要连续加班 48 小时。

2．目的

我真的就是想问你是否能接受加班吗⁇

客观地说一句，不管你怎么回答这个问题，是真接受还是假接受，还是真不能接受，该加班的时候还得加班。

我之所以会问这个问题，其实有两个目的。

第一，让你有一个心理准备，也就是常说的“ 丑话要说在前面 ”，这样不至于等你入职了，才知道加班很多。

第二，我想通过你的回答了解你之前的职场习惯，是被动加班多一些，还是主动加班多一些？如果能听到前两种回答，那最好不过了。如果没听到也没关系，因为在你回答的过程中，我能观察到一些东西。比如，你回答这种开放式问题的条理性、逻辑性和应变能力 ；另外，假如你之前加班也很多，我也能通过你的阐述判断出是因为任务重还是因为你自身工作效率有问题导致的。

站在团队管理者的角度，我认为每家公司都会有独特的加班文化，但再怎么花开百家，正确的加班文化都应该是鼓励主动加班、减少被动加班。

被动加班对于你来说有百害而无一利，你只会觉得累、觉得忙，但却忙得没有价值，而且会产生大量的负面情绪。

而主动加班是你自己要求上进，善于利用公司资源进行自我提升的一种表现，它所产生的就是一种正向的态度或激励情绪。这是我想要的，是我想看到的，也是我期望你能带入新的岗位中的。

友情提醒：最后针对有些喜欢跟面试官讨论存在大量加班是不是正常现象的同学，想提醒一句：别忘了，你是在面试呢！

你面试的目的是什么？通过面试呀！

所以，在面试过程中和面试官讨论加班本身的合理性没什么意义，就算你

发挥出比最佳辩手还要高 120% 的战斗力，说服了面试官，让他认同了被动加班或无效加班是极其不合理的论点，并大发慈悲地让你入职了，但在后续的工作中，需要加班时还是会让你加班的，你也都必须服从，不是吗？

因此，不管你怎么技巧性地回答这个问题，在这之前，请先明确地回答面试官 :“ 没问题，可以接受。” 至于入职以后，你是否能通过提高效率来减少无效加班，那是你自己的能力问题，跟面试无关。

最后，我想说明一下现状。加班在 IT 行业里本身就不是什么稀罕事，在做互联网产品的公司里，因为市场的客观原因，产品发布节奏比较频繁，研发时间总是被随时在变的需求压缩得紧巴巴的，所以大量的加班或者说连续的加班也已经成为行业现状，这还不算每次新版本上线时通宵达旦地工作。

重要声明：想从其他行业转入 IT 行业的同学，请慎重考虑；本来就在 IT 行业，想转入互联网企业的同学，请三思而后行！

问答（8）被问及职业规划，怎么回答才能让面试官满意？

背景

有工作经验的面试者经常会被问及职业规划是什么，这是一个相对开放的问题，也比较难以回答，因为有工作经验并不代表着一定会有职业规划。绝大多数面试者的答案是千篇一律的：先把本职工作做好，不管是手工测试还是自动化测试，等做了几年之后，再转去做管理，希望成为测试组长或管理一支测试团队。

你问

被问及职业规划时，怎么回答才能让面试官满意？

我答

（1）因为这个问题在大多数的二面或终面时都会被问及，你可以针对你应聘的岗位，事先做一些准备。比如你应聘的是初级测试工程师，那你就大致了解一下中级测试工程师的一些要求，然后有针对性地梳理一下。

（2）不建议采用上述那种千篇一律的答案：先技术，再管理。

一方面，站在面试官的角度，其实是想通过这个问题了解你对当前的岗位是否有比较深入的思考和长远的规划。

从我个人的面试经验来看，有时候，仅仅从面试者对于项目经验表述的熟练程度上未必能了解到其真实的能力，因为不排除有些人就是“面霸”，准备得特别充分。但从相应岗位的职业规划这一问题的回答中，我可以获取到更接近其真实水平的一些信息。因为没有真正深入了解和进行思考的同学，想在回答这个问题的过程中展现出自己的能力，还是有一些难度的。

另一方面，面试你的多半是测试管理者，你的职业规划如果是想做测试团队管理，那你是想取代他，还是想管理他呢？开个玩笑，不过并不排除有这么想的管理者。

（3）对于这种开放式的问题，其实不必担心回答的对不对或者面试官是否满意，这其实是一道给自己这次面试加分的机会题。我建议可以通过你的回答让它变得更开放一些，甚至可以引导成一次讨论，从面试角度来说，这绝对是可以获得加分的。

下面就以黑盒测试工程师岗位来举例吧。按照我的思路，我会这么回答 ：

首先，我会在做黑盒测试的过程中学习软件测试的理论知识，比如需求测试方法和测试用例设计，并将其应用在实际的项目任务当中。

等我熟悉了产品的业务逻辑，并且能够胜任具体的测试任务之后，我会开始学习用于提高测试覆盖率的一些工具或方法，来保证我负责的任务能高质量地完成。

同时，我会在项目中去学习测试计划的制订、测试进度的跟踪和测试报告的编制，并争取有机会去实践测试项目管理。

说到这里，稍微说明一下，我刚刚提到的是项目管理，而不是团队管理。

但其实我的内心也许是想在未来的几年里有机会实践团队管理。但我们要清楚，首先要管好项目，才能管好团队，因为很多管理的软技能在项目管理中也是能学习和实践的。

在这个过程中，我肯定会了解到测试技术方面的很多内容，我会做有限的了解，从中、长期来看，我会选择一个方向去学习，但这不是我短期的目标。

这样回答就差不多了，既能让面试官清楚，你在短期内还是以当前岗位为深耕细作的目标，又能让他了解到你也有想学项目管理的意愿。这在短期内就够了，或者说足够他给你制订入职后的培养计划了。

至于中、长期的目标，那只是一个愿景，面试的时候说得太多没有什么意义，因为太遥远了。

问答（9）怎样才能在面试时谈下自己满意的薪资？

背景

当面试接近尾声的时候，HR 或面试官一般会问一个问题：你期望的薪资是多少？不少人在给出一个心理价位之后，还会再报出自己当前的薪资，有的人还会往上虚报一些，期望能将未来的薪资谈得高一些，但这样做对你最终的定薪其实没有太大帮助。有很多人在问：怎样才能谈下自己满意的薪资呢？

你问

怎样才能在面试时谈下自己满意的薪资？

我答

我先从招聘的角度来说一下。你的简历里写的期望薪资，不管是真实的也好，还是虚报了一些也罢，其实都只是一个筛选简历的参考，对你最终的定薪没有什么实质性的帮助。所以我个人不建议虚报太高，因为有时候 HR 在筛选简历时，觉得你各方面的条件都比较匹配，但薪资跟用人部门的期望值相差太多，你可能就会失去面试机会，而实际上你的真实期望值也许是符合要求的。

所以我建议按你真实的期望值去填写，最终的定薪并不取决于这个值。

再从岗位角度来说，它是有行业平均水平的，但是不同的企业也有自己设定的区间范围。除非你在面试中表现得异常出色，或者你拥有应聘企业所迫切需求的、无法拒绝的优势能力，那么你的定薪可能会接近这个区间的中上部；否则，你的定薪只可能处于这个区间的中下部。

如果你在应聘前能通过内部的朋友或同学了解到该岗位在该企业里的薪资范围，那最好不过了。如果不能获取到，那就了解一下行业的真实平均薪资，

或者规模相当的企业的薪资水平，在这个范围内，依据你自身的真实水平，提出一个中等偏上的期望薪资。

关键还是要看你面试前的准备工作是否充分且有针对性，以及你在面试过程中是否把你最有价值的一面展示给了面试官，你是否成功地勾起了他对你的兴趣和期待。

特别要注意的是，在终面的时候，面试官一般是总监或副总经理，他们基本上是有薪资决定权的，在跟他们的沟通中你要把握一点 ：他们其实并不关注你的专业技能细节，因为一面或二面已经有相应的人考察过了，他们更关注的是你能给公司带来什么。所以，你要从这个角度出发，陈述你的优势能力能给公司带来什么样的收益，能直接挂钩成本或效率提升的数据最佳。

其实，一面或二面考察的是你的专业能力，三面或终面考察的是你的非专业能力。如果你能把握好以上几个关键环节，完成一次无懈可击的个人秀，那你再丢下一句话就可以功成身退、敬候佳音了 ：

“ 我相信贵公司应该能够提供与我能力相匹配的薪资待遇，能够很好地激励我为公司创造更大的价值。期待加入这个大家庭！谢谢！ ”

第 2 章　新人如何快速适应职场环境

问答（10）入门学习是方法先行还是工具先行？

今天有同学说，领导给了他一份接口开发文档，让他去做接口测试。他只知道他们用的接口测试工具是 SoapUI，但他之前没有做过接口测试，所以想了解怎么用 SoapUI 去测试接口。跟他沟通了一下，其实根本问题在于他不知道怎么做接口测试，而不是 SoapUI 的使用问题。

应该先学习测试技术方法还是工具？

在测试技术的学习上，应该是方法先行还是工具先行？我认为不能一概而论，需要先看看是什么测试技术，再看看学习的对象是小白一枚还是老兵一个。

1．接口测试的学习：先方法，后工具

提到接口测试，大家总是会提到 SoapUI 和 JMeter，更会习惯性地认为只有用到这些工具才叫作接口测试。所以很多同学会觉得想入门接口测试很难，其实他们是被工具的复杂度难倒了，而不是被接口测试本身难倒了。

对于接口测试，其核心部分并不需要依赖什么工具，而是依赖人的逻辑分析。

关于接口测试的学习，建议如下 ：

（1）了解接口是做什么的。

（2）理解接口的业务处理逻辑。

（3）熟悉接口核心的三组件 ：地址、入参、返回包。

（4）依据参数校验和逻辑校验设计入参组合。

（5）学习 JMeter 或者 SoapUI。

2．性能测试的学习 ：先工具，后方法

在没有深入了解过 LoadRunner 这个工具之前，我总是想当然地认为应该把性能测试和工具分开去学习，工具只是一个辅助性的东西，很简单，而方法才是根本中的根本。有些性能测试文章或书籍的作者也持这种观点。现在想来，这种观点本身没有错，但对于刚接触性能测试的小白来说，我个人认为会产生一定的误导。

如果是一门编程语言，那么你可以说语言本身的学习很重要，在学习期间不要过于关注工具，最好用记事本或 Notepad++。那是因为对于编程而言，开发工具只是一个编辑器，是一个提高编码效率的工具而已。

但对于性能测试这种对工具依赖性很强的事物来说，在初期，如果纯学习方法，那么很多东西就只能停留于表层、流于形式，或者说纸上谈兵，你并不能正确地、有效地理解那些理论上的知识点。

以 LoadRunner 为例，大部分人初次接触性能测试用的就是这个工具。它既然是一款成熟的、商业化的软件，而且用的人很多，那肯定有它的优势或者说有它流行的道理。我们要沉下心来，深入学习工具本身的工作流程、各个组件的使用方法、内部工作原理，然后再同步去实践，遇到问题解决问题，再记录问题、分析原因。

就这样一步一步地坚持去做，有一天我能很负责任地告诉自己，这个工具我已经能够用得很好了，而且能够知其然且知其所以然了。我相信，那时候的我，对于性能测试的理论方法或者性能测试流程不敢说 100% 的熟知，但也至少掌握了 60%，剩下的 40% 就需要再去深入学习了。

3．小白入门

现在很多人在想学习某样东西的时候，喜欢去网上看别人的入门经验帖。的确，这些经验帖有一定的帮助，但你自己也要清楚地认识到，写那篇经验帖的人和你现在所处的阶段是否匹配，或者说他是不是站在你当下的视角去写的。

很多所谓的入门经验帖都是人们在从小白升级到专家之后，复盘自己之前的学习历程，再加上自己在这个过程中的很多思考、总结和经验教训“ 精炼 ”而成的。但这种“ 精炼 ”也许会忘了一个重要的前提，就是“ 设身处地 ”。

如果能站在小白或入门者的角度，设身处地地为他们想一想，怎样更容易入门或学习效率更高，那样才能产出更有实操价值的入门建议帖。

问答（11）工作中没有目标怎么办？

背景

近来有很多同学跟我说，自己不知道接下来的目标是什么，觉得很焦虑。我问他们：当下的每天在做什么、学什么呢？他们说一直在找目标，什么都不想做，因为找不到目标，所以也不知道当下该做什么。

你问

因为没有目标，所以不知道该做什么，怎么办？

我答

一定得有目标才能知道该做什么吗？没有目标就不能前进了吗？这是“病”，得治！

确实，我不否认目标的重要性，不管是在日常生活里，还是在职场中，都应该有清晰的目标。有了清晰的目标，方向才能明确，才知道应该把劲儿用在什么地方。

可是，并不是所有的人都有明确的目标，即使一个有明确目标的人，他也不可能保证在他的人生里程中，每一分每一秒都有明确的目标。所以，人都会有迷惘的时候，都会有看不清方向的时候，都会有没有目标的时候。

因为没有目标，所以不知道该做什么，怎么办？

是像问题里的同学那样，遇到了，就停下脚步，在原地寻找目标，找到之后，再重新上路；否则，就在原地一直找下去吗？

但你别忘了，现在是一个快速发展的时代，从走路到吃饭，速度都比原来

快了很多倍。当你在原地停留的时候，你身边的很多人就会快速地超过你，远远地走在你的前面，那会让你越来越焦虑、心急，越急就越找不到自己的目标，越难看清方向。

我相信，很多人都遭遇过这种时光，迷惘、焦虑，一边在苦苦思考着未来的方向，一边又在为自己每天的“虚度时光”而焦虑，长此以往，陷入了一种恶性循环。

我其实也遇到过很多次，虽然最终都找到了一个新的目标，继续走了下去，但也为此浪费了不少宝贵的时间。不过随着经历的事情越来越多，我也慢慢地找到了根治这种“病”的良方：

看不清方向的时候就去做事，找不到目标的时候就去学习。

与其看不清方向在原地打转，不如选定一个方向开始着手去做。随着进度的推进，方向会越来越明晰。

即使做到一半，发现方向错了，我相信，在这个过程中你所收获到的远大于你停留在原地思考所得的。而且，不断地试错，正是寻找正确方向的不二法则。任何事情，你不去尝试、不去做，你永远不可能知道它是对的还是错的。

与其因为找不到目标而无所事事，不如静下心来去学习。你肯定又会问：我连目标都没有，该学什么呢？

你总是在从事什么职业吧？即使你没有从事任何职业，你也会有自己的兴趣爱好或自己喜欢做的事吧？即使退一万步来说，你什么爱好都没有，那你也可以找一门实用的技能，比如英语、烹饪、手工等。不要给自己定什么学习目标，只是单纯地去学，把自己当作一只空杯子，或者一块干涸的海绵，通过学习来给它不断地加水。当你学到某个阶段的时候，你下一阶段的目标也就慢慢地显现了。

所以，迷失方向也好，缺失目标也罢，如果停滞不前，就都是“病”，是“病”就得治！大家可以试试我上述的方子，这也是经过临床验证的。

问答（12）为什么我的个人计划总是执行不下去?

背景

一天下午，有同学问我，为什么他的个人计划制订出来后，总是虎头蛇尾，执行不下去，觉得按计划执行起来好累，但如果不做计划又觉得每天在不知不觉中时间就被浪费了。我让他把计划发给我看看，想从计划本身找找问题的根源。

拿到手一看，把我吓了一跳。“今天待办事项”清单一屏都显示不下，我从上到下滑动了几屏，从早上 9 点多的项目例会开始一直到晚上 8、9 点，大小不一的各种任务罗列其中，让我印象深刻的有这么几项：

（1）有两个任务都是计划制订类的，间隔时间大概是 40 分钟。

（2）有两个过期未完成任务，计划完成时间有一天前的，还有一周前的。

（3）有一个测试任务，计划开始时间竟然是晚上 7 点。

你问

为什么我的个人计划总是执行不下去？

我答

我先说一下我的看法。

（1）计划类的任务相当于创造性的工作，时间往往不好预估，所以我一般很少在同一天给自己安排两个及两个以上的计划类任务，更别说连续安排在那么紧的时间区间里。

（2）过期未完成的任务，理论上应该是“今天”清单里优先被完成的任务，

可已经到了下午，却还有两个这样的任务。而且还在今天安排了很多其他的任务，本身就不合理。按我的理解，这样最后只会进入恶性循环，昨天的任务未完成，今天继续，然后今天还会有未完成的任务被拖到明天。

（3）至于那个测试任务，我从描述和后面的沟通中了解到，并不是今晚或明天就要完成的，甚至并不是这个月必须完成的，所以说并不是一项紧急且重要的任务，为什么要计划在晚上加班的时间去完成它呢？

再者，我也想说一下我对计划制订持有的一个观点 ：**一个好的计划是需要留白的**。

不论是项目计划、工作计划，还是学习计划，一定要留白。这一点真的很重要，合理的留白能够让整个计划的可执行性提升至少 30%。

留白，其实就是给你的计划预留出合理且足够的机动时间。

回到每天的待办事项清单上，我觉得比较合理的一个体量大概就是 1 项高级任务 +2 项中级任务 +3 项低级任务。当然，这也不是绝对的比例。拆分开来看，基本理念是这样的 ：

（1）首先要能确定每天最重要的任务是什么，也就是那个高级任务，这是需要放在你一天中精力最强的时段去完成的，或者是需要优先完成的。如果从极端的角度来看，它需要花费你 8 小时才能完成，那你的 “ 今天 ” 清单里就应该只有这一项任务。

（2）2 项中级任务，按我的理解，应该就是常规性的工作，比如测试任务、一些常规会议等。理论上每天都应该有这样的任务，毕竟从精力管理的角度来看，连续在一项任务上花费的时间太长会出现疲倦期。换一项不同类型的任务切换一下，有助于重新激活状态。

（3）3 项低级任务其实可以理解成 8 小时内的 “ 留白 ”，它们所预估的时间其实可以用来应付临时紧急任务和对中、高级任务预估不足的情况，从而保证你不至于一天的安排太过紧凑，全是中、高级任务，哪怕加一点点临时任务，

都会导致全线崩溃。

（4）对于那种在计划期就知道一天完成不了的、需要长线作战的任务，可以设置成循环任务。滴答清单里支持的循环模式十分齐全，如每天、每周的工作日、每周的周几、每月的几号、每月的第几个周几、每年的几号（阳历）、每年的几号（农历）和自定义（每天、每周的哪几天、每月的哪几个周几），应该够用了。

这样，每天的任务完成后，系统会自动创建下一次的任务项目，从而不会有那么多未完成的过期任务留在清单里。这对于日事日清的冲击还是很大的，特别是对于强迫症患者来说，总有未完成的任务挂在那里。

所以，留白最美，留白的计划才是合理且容易执行的计划。

问答（13）怎样才能迅速了解一个产品的业务流程？

背景

老师，我想问一下，如何迅速了解产品的业务流程呢？有的根本没有文档可看，有的话也是复杂凌乱的，不知道怎么看，感觉一头雾水，无从下手。

这是很多测试人员进入一家新公司或者接手一个新产品时最常面临的问题，一是因为文档的缺失或更新的不同步；二是因为从管理者的角度来看，时间是稀缺资源，肯定希望你尽快了解产品、熟悉业务，尽早投入工作。

以该同学想迅速了解的产品——企业内部使用的培训管理系统为例，该系统的功能大致包括报名、学习、考试等。

你问

怎样才能迅速了解一个产品的业务流程？

我答

根据我的个人经验，先来说说相对普适性的方法。

（1）找该产品对应的测试负责人、开发人员或产品经理，向他们**了解一下产品的应用场景、用户角色和业务主线**。比如，使用该产品的用户群体是谁？是否会分为系统管理员、老师和学员？该系统的主线流程又有哪些？

（2）按不同的角色去**使用这个系统**。在使用过程中，如果遇到问题，则可以查看相应的文档（不过，在绝大多数公司里，文档都是缺失的），或者询问相应的测试负责人、开发人员或产品经理。问问题的时候要注意，围绕问题多发散一下，特别是在跟开发人员沟通的时候，多问问与代码逻辑相关的内容，也

就是功能背后的东西，这对你理解深层次的业务逻辑会有较大的帮助。

（3）针对不同的角色，在对这个系统的业务场景有了一定的了解之后，我们就可以**开始梳理和学习业务逻辑与功能细节了**。

（4）**梳理主线，把各个角色和主功能串联起来**。比如 ：

“ 课件 ” 主线应该就是老师设计课件→学生报名→学生选课→学生学习。

“ 考试 ” 主线应该就是老师出题→学生考试→老师阅卷→老师评分→学生查成绩。

（5）**跟剥洋葱一样，从外层开始把每个环节拆分成单独的模块，再把模块一层一层地从外到内剥开**。比如 ：

“ 课件 ” 主线能够分为设计课件、报名、选课和学习等模块，而 “ 设计课件 ” 模块还能细分成课件编辑、课件上传、课件管理等子模块。

“ 考试 ” 主线能够分为出题、考试、阅卷、评分和查分等模块，而 “ 考试 ” 模块还能细分成选择考试科目、选择考试日期、答题等子模块，“ 答题 ” 模块还能再继续细分成更小的模块，这主要取决于业务逻辑的复杂程度。

当你按照这个步骤拆解完毕，形成一套思维导图的时候，在业务功能层面，你已经了解了至少 70%，可能欠缺的就是一些没有文档说明的细节问题和异常场景。

如果你想更快速地深入了解产品，那再介绍一种我从其他人那里学来的方法。

（1）从开发人员那里获取到业务日志和 SQL 日志所在服务器的地址、用户名和密码，以及日志路径。

（2）安装 Xshell 之类的工具，连接服务器，打开实时输出的日志文件。

（3）在前端页面进行相应的操作，在后台同步查看输出的日志。通过日志

里显示的接口参数和执行的 SQL 结果，你能清楚地看到前端的某个业务功能，在数据库中会关联到哪几张表，以及表之间的逻辑关系。

问答（14）怎么才能顺利通过试用期?

经过层层面试，杀出重围之后，同学们一方面怀揣着激动的心情走向自己心仪的工作岗位，另一方面看到同期进入了好几名求职者，个个看上去都不是“善茬”，又开始担心自己是否能顺利通过试用期。

怎么才能顺利通过试用期？

1．前言

（1）首先设定一下场景：**三个月的试用期**。因为基于不同的时间长度，有些策略可能需要调整。

（2）再声明一下，我回答这个问题的**角度是双向的**：向上，我有我的“将”；向下，我有我的“兵”。大家可以结合着去解读，会更有参考性。

2．对自己要狠一些

（1）工作效率要高。

领导如果让你周五完成一份报告或者完成对产品业务功能的熟悉，那就做到周三完成。这样做有两点好处：一是体现你的高效率；二是给自己留下充足的修正余地，因为永远不要相信能一次就做到让领导满意。

（2）完成结果要超出领导预期。

如果完成任务算 100 分，那就做到 120 分。就像下象棋一样，永远要比你的对手多想几步，要让领导每次看到你的任务结果时，总会感叹“ 原来还能这样啊 ”。

（3）要有可以拿出来“ 炫耀 ”的“ 数据化 ”产物。

不要只是埋头熟悉产品、流程和完成领导交代的任务，要善于观察当前项目中有什么问题，并从中找寻机会，自己独立解决问题。同时，很重要的一点是，这个结果最好可以通过产物或数据展现出来。比如，通过改进一个方法或形成了某套流程文档，解决了某个问题 ；或者解决了某个问题，提升了 30% 的工作效率，降低了 20% 的人力成本，等等。

数据化结果是为了让别人更直观地看到你的产物的“ 收益 ”。另外，如果你的领导用这些可炫耀的数据化产物去帮你申请提前转正，那是不是胜算很大呢？

3．对你的“‘将’”要主动一些

（1）不要等到 Deadline（最后期限）才提交结果。

永远不要等到领导找你才去汇报进度或反映问题。最佳的做法是将任务拆分成若干个阶段，如规划阶段、执行阶段和收尾阶段，每个阶段都主动跟领导沟通一下，一方面可以让领导随时掌握你的进度，另一方面也可以避免很多无效工作的产生。

比如，你对任务的理解有偏差，最终做出来的东西不是领导想要的。假如你在规划阶段结束时主动发给领导看一下你的规划，领导就能更早地发现你的理解有误，并及时纠正。

（2）向领导反映问题时，不能只抛出问题，而要提供解决建议。

记住，公司聘用你不是来找问题的，而是来解决问题的。但有一点也要注意，

最好提供至少两个解决方案让领导做选择题。因为有些领导未必喜欢你帮他做决定，让他有良好的决策感，也会有助于他更容易接受你的解决方案。

4．其他

（1）在反映问题或执行任务时，切忌越级处理，任何问题都应该由你的直接领导向上反映。不管是多大的领导直接指派给你的任务，都要跟你的直接领导报备一下。

（2）凡事要多换位思考，特别是在对领导的一些安排或举措有不同意见的时候，建议单独跟领导沟通，而不是公开抵触或私下抱怨。也许在沟通后，你会了解到一些站在你的角度和高度所看不到的东西，你会发现，你所看不到的东西恰恰也是你的不足之处，然后就可以窃喜又学到了。

最后想跟大家说的是，试用期其实没必要过分焦虑，也不要对身边其他的竞争者产生敌意，专心做事，适调做人，对不了解的人或事不进行评价，因为你初来乍到，水有多深不是你站在岸上就能看出来的。同时要谨慎言行，远离职场负面情绪和抱怨较多的个人或群体。

坚信一点 ：跟对人，做对事，迅速成为一块发光的金子！

问答（15）学习型圈子能给我带来什么？

背景

最近有同学问我，自己加了好几个关于测试的微信群或 QQ 群，还有小密圈，天天在里面花的时间也不少，但却不知道能给自己带来什么。其实，热播电视剧《人民的名义》里也有圈子这一说法，不论是汉大帮、秘书帮，还是山水集团，都可以看成一个一个的圈子。而那些关于测试、开发还有项目管理的微信群或 QQ 群，其实也就是一个一个的圈子。

你问

学习型圈子能给我带来什么？

我答

如果把视线从电视剧转移到现实生活中，那么在生意场和职场中，圈子也是无处不在的，一个人要么默认就置身于某个圈子中，要么就主动地加入了一个圈子，或者无意识地被贴上了某个圈子的标签。一个圈子就意味着一股人脉，或者说是一个彼此间有着某种利益关系的群体，资源共享，风险共担。如果你身处其中，却又想要完全置身事外，其结果很可能就是被边缘化或者被孤立。

让我们再看看百度百科里关于“ 圈子 ”的定义，这个定义其实更接近我们的日常生活 ：

圈子指具有相同爱好、兴趣或者为了某个特定目的而联系在一起的人群。

有相同爱好的人组成的圈子，如骑行圈、观影圈、越野圈、美食圈、读书圈、集邮圈等，数不胜数。

为了某个特定目的而组成的圈子也有不少，除去前文所说的那种有利益目的的圈子，还有不少出于学习和成长目的的圈子。比如拆书帮和 Toastmaster，还有我们接触较多的技术圈子。这些圈子其实都没有太多的利益关系或目的，就是一群有着同样目标和期望的人，组成这样一个互相学习、互相监督、共同进步的群体。

学习型圈子除了可以互相监督、互相学习，也有着资源共享的优势。在一个圈子里，聚集着对某方面感兴趣、力求上进的同学，其中有小兵也有老兵，有新手也有高手，取长补短，各取所需。通过交流、问答、讨论、分享等多种形式，逐步积累出很多有用的知识和经验。

另外，学习型圈子还有一个良性的作用，就是让新兵在其中能够获得最接地气的指导和解答，能够从很多老兵和高手那里获取到有用的经验，少走一些不必要的弯路，绕过一些大坑、巨坑。相对地，那些老兵和高手也能在圈子里夯实并实践自己的知识技能，提炼自己的经验体系，同时还能收获到满满的成就感，更有动力地去学习和提高自己。

并不是说只有在圈子内才能学得好、才能进步快。我们每个人其实都是从圈子外开始的，包括我自己在内。我接触这种学习型圈子的时间其实不长，却已经深深地感受到这种学习型圈子的正向氛围，以及对自己的帮助了。特别是对于想涉足某个领域、想学习某项技能，但自己的自觉性不是很高、身边资源又不是很多的人来说，加入一个圈子，跟随高手的脚步，学习并实践你的成果，是一个不错的选择。

不一样的圈子，能给你带来不一样的意外收获！

问答（16）在职场中如何学习?

背景

“老师，我怎么在上班以后就不知道该怎么学习了呢？我以前在学校里也算得上‘学霸’了呀！”有同学在制订个人半年度计划时这样问我。

你问

为什么进入职场之后就不会学习了？

我答

信息，指音信、消息、通信系统传输和处理的对象，泛指人类社会传播的一切内容。人通过获得、识别自然界和社会的不同信息来区别不同事物，得以认识和改造世界。

我们正身处一个信息化时代，而且还是大数据时代，所以我们最不匮乏的自然就是**信息**了。比如你们现在看到的这篇文章，你们在平时的工作和学习中读到的书，还有现在各种音频、视频，它们里面都包含了大量的信息。

在这样一个信息化时代，我们最缺乏的东西是什么呢？

我以为，我们最缺乏的应该就是**知识**了。

你们是不是会问，我们天天从各种文章、书、音频和视频里了解到的和学到的难道不是知识吗？

那我们再来看看什么是知识吧。

知识是符合文明方向的，人类对物质世界及精神世界探索的结果总和。对

于知识，至今也没有一个统一而明确的界定。有一个经典的定义来自柏拉图：一条陈述能称得上知识必须满足三个条件，即它一定是被验证过的、正确的，而且是被人们相信的，这也是科学与非科学的区分标准。由此看来，知识属于文化，而文化是感性与知识上的升华，这就是知识与文化之间的关系。

我转译一下：**能够改变你的思维模式和行为习惯，对你以往的输出结果产生了影响的就是知识。**

既然已经了解了什么是知识，那理解学习就比较简单了：**学习，就是将信息转化为知识的一种行为。**

既然已经清楚问题出在哪里了，那就对它做一次 3W 分析。

1．问题是什么

如果你用 RIA 法（一种拆书读书法）去读时间管理类的书籍，则会有明显的收获感。可如果你用同样的读书法去学 LoadRunner 或 Scrum 这类偏技术类的书籍，就会觉得做了很多无用功，有时候即使强迫自己把整本书学完，真正吸收的也不足 20%。

2．根本原因是什么

信息只有被有效地处理后才会变成知识。我在阅读时间管理类书籍的时候，因为有对应的实践经验去比对印证，所以，无论我过往做的是对还是错，都能很容易地将信息转化成知识，而且会相对系统化。

而在学习 Scrum 或 LoadRunner 这类书籍的时候，因为从学校带来的惯性，不管是不是所有的章节对自己有用，都要求自己一定得读完整本书，所以总会先统计一下页数、章节数和天数，再计算出每天的计划学习量，然后就按照书中章节的顺序开始研读了。

这恰恰忽略了自己学习的真正目的：**解决问题**。

也正因为缺少了“ 问题 ”这个核心，从而导致貌似完整的阅读，实际上却是支离破碎的学习。所以，我们很难将获取到的信息和自己的认知有效地关联起来，形成知识。

于是，我们就陷入了一种“ 既然开始了，就要读完，一次只读一本 ”的定势思维，这种学习收获或者学习笔记其实只是在重复地做着信息的复印机或者搬运工。这也是很多从学校进入职场的同学总觉得自己一下子不会学习了、不知道怎么学的原因。

3．解决方案是什么

（1）打破必须读完整本书的思维定势和一次只读一本书的局限。

（2）以解释问题和解决问题为出发点去挑选书籍和资料。

（3）围绕问题去做主题阅读和学习。

（4）以问题为主线，将新、旧知识点连接起来，形成脉络图。

（5）放慢读书的速度，不定量，也不定时，以问题解决与否为标准。

（6）梳理现有的知识关系脉络图，以问题的形式标注出脉络不通的地方。

（7）对应的读书笔记其实应该就是针对某一个或某一类问题的解决方案。

（8）在读书笔记的最后罗列出参考书籍和资料。

问答（17）如何写出一份漂亮的年终总结?

背景

有些同学经常为如何写年终总结而头疼。大多数公司只有在年底的时候才需要忙着写年度总结，不过也有一些公司是半年度，甚至季度和月度都要做。我认为，月度有些短，很多计划未必能完成；而年度又过长，长到足以忘记我今年的“丰功伟绩”。所以，季度或半年度是相对合适的总结长度。

你问

如何写出一份漂亮的年终总结？

我答

写年中（终）总结估计是很多人都比较头疼的事情，先抛开内容不说，不同的公司有着不同的模板，有着不同的格式要求，Word 版的、Excel 版的、PPT 版的……五花八门。而且 HR 部门估计也秉承着迭代改进的精神，坚定不移地保证了每年都有改版，每年都有“惊喜”，每年都会把人虐得死去活来，不花上一个星期的时间，你都搞不定一份“漂亮”的年终总结。

所以我在这里说的漂亮的年终总结仅仅指内容，而与格式无关。

我一直对季度、半年度或年度总结持赞成态度，因为我觉得工作中的定期总结和复盘都是有必要、有好处的。将每次复盘中总结出来的一些收获叠加起来，再有针对性地调整一下，就是一份合格的总结了。

具体分为以下几个步骤。

（1）先把这一年所做的常规性工作数据统计一下，罗列出来，这个数据要

能体现出你的这项工作的“量”和“质”。

（2）检查一下你去年的年度计划，与你的复盘日志对比一下，看看是否都完成了。如果有未完成的，那么最好能找到相关的数据来证明是因为其他重点任务而影响了进度。

（3）从已完成的任务中找出改进较大或产出显著的任务，重点从以下几个维度去总结。

任务的背景，也就是目的和初衷，重点阐述是为了提高某个方面的效率或生产力，或者为了节约某项常规工作的人力成本。

任务的完成过程尽量简洁，突出过程中遇到的有价值的难题和解决方案。如果没有明显亮点，则可一笔带过。

改进结果或产物所带来的效益尽量数据化展现，比如，节省了多少人力、提升了百分之多少的效率等。如果可以，则也可以考虑图形化展示数据。

如果有相关合作部门或者上级领导的感谢邮件或者表扬信等，则也可以附上。虽然当时你的领导肯定知道这件事情，但年中（终）未必还记得，可以适度提醒一下。

（4）计划部分，就是提取复盘中的改进点或产物的优化点，结合自己下一年的定位和学习计划列出。

最重要的一个原则就是：必须得有产物。因为产物是最直观的，是能看得见、摸得着的东西，可方便领导考核你的这项任务。换位思考一下，与人方便也就是与己方便，不是吗？

剩下的工作也就是将这些沉甸甸的内容按照模板要求清晰地展现出来，做到有数据、有图表、有对比，我想就可以收工了。如果全身心投入，那么一天完全能打造出一份高质量、高回报的年终总结。

问答（18）工作计划和个人计划有什么本质区别?

背景

上一篇聊了一下怎么写一份漂亮的年终总结，就有同学问我：工作中的年度计划和自己平时制订的个人年度计划有什么本质上的区别？很明显，这位同学是有困惑的，有困惑就说明他对此是有自己的思考的，而不是糊里糊涂地就把二者混为一谈。

你问

工作计划和个人计划有什么本质区别？

我答

职场中的人可能会比较容易理解，其实工作计划和个人计划从来都是焦不离孟、孟不离焦的。但我们在这里所说的工作计划和个人计划，虽然有百分之六七十的内容可能是重复的，那是因为我们不可能将工作和生活完全分开，比如学习类的计划，因为学习就是为了学以致用，期望在职场上有更多的收获，但两者其实有一个很重要的区别，也可以说是本质上的区别，那就是两种计划的读者不同。

工作计划的读者是你的直属上级，而个人计划的读者就是你自己。

为什么说这是很重要的区别呢？

因为这个区别决定了你在制订职场年度计划时，该从什么视角去筛选你的目标群。

当你明白了你的工作计划是给谁看的之后，你就应该明白，要想写一份“可观”的年度计划，应该具备“三高”：高集中度、高可行性和高回报率。

每个人肯定都有很多想做的事，有很多想学的东西，可是你的精力是有限的，时间也是有限的，你没有办法在一段时间内多头并进，而且还能多点开花，对不对？

首先，罗列出下一年的常规性任务、部门的重点任务规划和自己依据自身的定位或兴趣而想有所提升的方向，这至少会有十几项。

紧接着，结合部门规划或职业规划，做一下筛选及同类项合并，应该还能剩下四五项。

再者就是重头戏了，把这四五项放在一起，按下面的几个维度逐一进行对比。

1．自身的利益

因为不管做什么事情，如果在这个过程中自身得不到任何提升，那就不值得去做。

2．产出的效益

因为这是你的直属上级最关注的，一方面，他可以用这个指标直观地考量你；另一方面，这也是他下一年年度部门绩效的砝码之一。

3．性价比

要综合计划任务的难易程度和产出的效益计算出性价比。这决定了最终加入年度计划的任务项。

计划在精不在多，只有最终能完成的计划才是最成功的计划，在此之前，计划只是计划。计划也不是一成不变的，计划就是为了变化，一份一变就崩塌的计划也不算是成功的计划。

最后再说一点，看似有限的时间其实很长，所以不必过于焦急冒进；看似有限的精力其实也完全够用，所以不必过于贪多求快。只要你迈出一小步，走着走着，回头看去，就会发现，其实你已经迈出了一大步。

PART 2
在软件测试之路上越走越好

适合工作年限：3～4年

第 3 章　怎样才能利用好职场的黄金期

问答（19）如何有效分配每天的 24 小时?

背景

小密圈——“软件测试圈”的圈主老徐有一天问我：你如何分配自己每天的时间？工作、生活、学习、写作的占比如何？就当给圈子内的新人一些建议。我突然感觉一下被问住了，回答这个问题其实可大可小，因为有些习惯已经养成了，突然要我把它给拎出来说说，就觉得毫无章法可循。于是我抽出一点时间来回顾整理一下，顺道看一下是否还有改进之处。

你问

如何有效分配每天的 24 小时？

我答

1. 当下的事

(1) **当下的工作**：一家互联网初创公司的测试团队管理，每天要处理的事基本上差不多，无外乎团队建设、测试项目管理、跟客服和市场人员沟通并跟踪处理各类现网问题等。

(2) **当下的学习**：相关的软件测试专业技能和职场软技能的学习，包括在小密圈和微信上回复一些问题，同时也在收集问题。

((3)) **当下的写作**：在简书上每日更新，也会同步到自己的公众号上，内容基本上围绕当下的学习，相当于学以致用后的回顾、反思和精炼。

((4)) **当下的生活**：大部分精力用在操心孩子的学习和教育问题上，还有一部分精力用于运动、骑行和跑步。

上述那些当下的事情都是从不同维度的不同角色身份出发去设定的，而且我并没有把它们完全割裂开来去看待或者去计划。

【个人观点】工作和生活未必需要有明显的界限，职业和兴趣也未必不能合二为一。

2．时间分配（(从表面上看是在分配时间，其实是在分配精力)）

（1）09:00—18:00

① 完成当下的工作任务。

② 实践当下的学习内容。

③ 记录工作心得，收集值得复盘和分析的问题，获取学习和写作的灵感。

（2）08:00—09:00 + 18:00—19:00

骑行，既不用单独找时间去健身，又能保持较好的精力。

（3）19:00—22:00

① 检查孩子作业 + 无计划区间，用于恢复白天消耗的注意力和精力。

② 慢跑 30 分钟到 1 小时。

（4）22:00—01:00

① 开始进入一天当中精力最好的时间区域。

② 进行主题阅读和学习。

③ 完成日更和相应自媒体平台的更新，比如简书、公众号、小密圈等。

（5）01:00—07:30

熟睡期，睡眠质量较好，一觉无梦到天明，估计跟适量的运动有关。

【个人观点】早起和晚睡没有绝对的对与错，而在于不同的人有不同的精力波峰段。有些人适应早起，效率超高，比如简书一哥彭小六；有些人适应晚睡，早起反而适得其反，比如我，在经历过多次想做一个晨型人的失败实践之后，终于放弃与自己的精力规律相抗衡。

3．每日清单的必要性和策略

（1）做到事事入清单，每件事都有截止日期。对于没有明确要求的任务，自己将截止日期设定为次周的周一。

（2）清空你的大脑，事无巨细，都应该放到清单里，特别是常规的、重复性的、会消耗你的判断力和选择力的东西，比如每月的固定缴费、每周的采购清单等。

（3）在每日待办事项里，优先级最高的任务必须是跟当下工作相关的，必须在第一时间完成它之后，才能开始其他当下的学习和实践类任务。

（4）根据自己在不同时间段所扮演的角色来分析最近一周的时间分布，建议把精力好的大段时间留给那些“重要且不紧急”的任务，把精力中等的大碎片时间留给那些“重要且紧急”的任务，其余碎片化的时间可以用于机动处理

那些“ 不重要 ”的任务。

（5）除非是对时效性有明确要求的任务或者那种迫在眉睫的紧急问题，否则建议先完成自己的既定计划任务，再去响应别人的问题或求助，因为你的高效率体现在你的计划产出比上，而不是充当救火队员的次数。

【个人观点】利用传统的四象限法则，在实际应用中，有时候很难对某项任务进行归类，我建议可以简单粗暴一些：

直接影响你当下主要收入的任务就是重要且紧急的；

会影响你未来主要收入增长的任务就是重要且不紧急的。

对于新入行的同学来说，有一点需要注意，那就是在朝九晚待定的时间段里，自己并不可控，而是更多地依赖领导的任务分配。通常会有很多事先难以预知的临时任务，所以更重要的就是找到并管理好自己的最佳精力时间段，同时利用好工作中的任务，在实战中学习并积累、沉淀。

问答（20）把用户当作“用户”还是“客户”？

背景

有位同学在处理现网工单时遇到一个问题：客服部门和市场部门催得比较急，说用户是当地最大的通信营业厅；但找到开发人员，开发人员却说，我们是做产品的，不能因为某个用户比较大就要立刻解决他的问题，发布上线吧。他觉得这种说法其实没错，但又没办法用这个理由向客服部门和市场部门解释，所以觉得很困惑。

你问

我们应该把用户当作“用户”还是“客户”？

我答

对于企业级产品，其面向的用户就是企业，一般都是签单的，如果是特别大的签单企业，比如我们之前最大的用户是波音公司和美国银行，那么他们遇到的一些问题或者提出的一些需求就是“大”问题或“大”需求，有可能需要你立刻解决、尽快实现，然后就安排上线了。他们不会管你当前的产品开发计划是什么，都会出一个补丁包上线。

对于行业定制化产品，比如 ERP 或电子政务系统，是跟客户签订了合同的，有约束条件，他们遇到的问题或提出的需求也可能有解决时限，并不一定会按照你既定的产品开发周期去安排。

而对于互联网类应用程序产品，大多数人认为产品发布周期只要依据自己的产品规划和开发周期来定就行了，除非遇到很严重的功能障碍或漏洞，否则都不需要因为某些用户的“大”问题或“大”需求而临时发包上线。

我觉得，造成这种观点的原因主要是绝大多数应用程序的使用是免费的，即使付费使用，那也只是“ 用户 ”，而不像上面所说的企业级产品和行业定制化产品的用户实际上都是“ 客户 ”。

所以，在互联网应用程序产品的现网问题处理机制中，我认为，很重要的一点就是我们应该把用户当作“ 用户 ”还是“ 客户 ”。只有从管理层到运维团队的执行人都统一认识，我们才能制定出正确的、适合现网问题的处理流程，才能规定出一些明确的标准 ：

（1）问题在多长时间内要被响应？

（2）缺陷类问题需要在多长时间内被解决？

（3）什么类型和级别的缺陷类问题需要立即发布版本上线？

（4）什么级别的需求类问题需要加入最近的版本发布上线？

如果从上到下都统一了认识，把每个免费用户和收费用户都当成我们的**客户**去对待，就不会有那位同学所面临的困惑了。因为那时候我们会更聚焦问题本身的严重性和级别，而不会因为只是某个用户或少数用户遇到某个问题或提出某个其实还不错的建议而不予重视，或者置之不理。

问答（21）如何设计产品的兼容性测试？

背景

老师，我想请教一下，你们是怎么做兼容性测试的？我现在做兼容性测试，用的是浏览器的不同版本、不同种类的浏览器、不同语言，都要走一遍工作流。我觉得这样做好像很麻烦，而且也搞不清楚这样跟功能测试、业务流程有什么区别？是不是重复了？

这位同学问的其实只是兼容性测试里的一种，就是浏览器兼容性测试，常见于 B/S（浏览器 / 服务器端）结构的产品中。其实兼容性测试有多种类型，我们今天就来看看，对于不同类型的产品，应该怎样设计兼容性测试。

你问

我们应该怎么去设计产品的兼容性测试？

我答

1. 什么是兼容性测试

兼容性测试就是验证开发出来的程序在特定的运行环境中与特定的软件、硬件或数据相组合是否能正常运行、有无异常的测试过程。

2. 兼容性测试包含哪几类

(1) **浏览器兼容性测试**：在指定的浏览器上检查 Web 页面样式和元素的展示效果，以及交互是否正常。

主流浏览器：

Windows：IE 9/10/11，Firefox（最新版本），Chrome（最新版本）。

Mac：Safari，Chrome（最新版本），Firefox（最新版本）。

测试注意事项：

① 浏览器兼容性测试常见于 B/S（浏览器 / 服务器端）结构的产品中。

② 虽然我们能通过官方的一些统计数据去收集主流的浏览器和对应的版本，但最好让产品经理明确定义出支持哪些浏览器和对应的版本，因为这也取决于产品的应用人群和具体的业务场景。

③ 浏览器兼容性测试主要检查 Web 页面样式和元素的展示效果，以及交互是否会有异常，跟具体的业务逻辑无关。

④ 跟前端开发人员多交流，明确哪些样式或元素不是标准的，很有可能会出现兼容性问题，先有针对性地在所有要求支持的浏览器版本上进行验证，再挑选每种浏览器的一个版本去验证所有的标准页面。

⑤ 多记录，多总结，做好统计分析，在后续的测试中，只需针对有改动的、易出现兼容性问题的元素和样式进行测试。

⑥ 留意 IE 大版本的升级，以及 Chrome（谷歌浏览器）和 Firefox（火狐浏览器）的迭代版本更新，阅读更新的版本说明，了解是否有大的改动可能会影响到页面的展示或者交互，有计划地去执行兼容性测试。

(2) **操作系统兼容性测试：**在指定的操作系统上检查产品功能是否正常。

主流操作系统：

Windows 系列、Mac OS X 系列、UNIX/Linux 系列、Android 系列、iOS 系列

测试注意事项：

① 常见于 C/S（客户端 / 服务器端）结构的产品中，互联网时代的应用程序从广义上来说也是 C/S 结构的。

② 基本的注意事项跟上述的浏览器兼容性测试一样，需要关注的是，不同版本的操作系统默认的权限级别会有所不同，从而导致客户端需要访问或调用系统组件或方法时会出错。

③ 同一类操作系统的大版本升级时，需要注意在新的版本或补丁里是否继续兼容老版本的库函数。

(3) **多版本兼容性测试：**是为了验证新版本服务器端是否同时支持新 / 老版本客户端而进行的测试。

测试注意事项：

① 这是很多产品经理在设计需求时容易忽略的地方，也是 C/S 产品和 B/S 产品从兼容性角度来说最大的区别。

② 在产品升级之后，服务器端只会是最新版本，但客户端因为不同的用户场景而可能存在老版本，一种原因是没有强制更新，用户不选择升级；另一种原因是在一些企业级的域环境里，客户端包是否升级取决于域管理员的策略。

③ 针对单客户端的产品而言，测试相对简单一些，只要保证服务器端每次升级都不会因为新需求而修改老接口，基本上不会有太多兼容性问题。

④ 测试相对复杂的是那种既有商家版又有用户版的客户端产品，针对会频繁发生交互的功能，需要重点考虑新、老版本的兼容性测试。

(4) **数据兼容性测试：**因为新功能的需要或者已有功能的升级改造，涉及已有数据的读取和写入而需要进行的验证，以确保数据在新、老版本之间都能正常流转的过程。

测试注意事项：

① 向前兼容（Forward Compability）：新版本的软件要能正常且正确地读取和加载老版本生成的数据。

② 向后兼容（Backward Compability）：当前版本的软件要能支持在后续高版本的平台上正常运行。

③ 常见的 Office 类软件或者多媒体制作或播放类软件，不仅需要考虑新版本客户端是否能正常读取老版本生成的文件，还要考虑新版本生成的文件是否能被老版本客户端正常读取，或者有相应的升级提示信息。

④ 还有一类是常见的订单类数据，要考虑在老版本的服务器端和客户端组合下产生的数据是否能在新版本的服务器端和客户端组合下读取成功，同时业务流程也可以正常进行。

⑤ 对于数据兼容性测试来说，会更多地关联后台历史数据的迁移和转换。这一部分内容也是需要重点关注的，以确保迁移和转换后的数据用户能正常读取。

(5) 分辨率兼容性测试：也被称作适配性测试，是指验证被测网页或产品 UI 在各种分辨率下的显示器或各种分辨率、尺寸屏幕的移动设备上都能正常显示的测试过程。

测试注意事项：

① 需要关注的一种是普通分辨率的屏幕，另一种是高清分辨率的屏幕。

② 需要关注的问题主要包括显示是否完整、图片是否被拉伸、文字和图片位置是否有错位。

问答（22）如何绘制功能模块的数据流图?

背景

小密圈有同学提问：昨天经理给我分配了一项任务——整理需求分析，要用到一种方法——结构化分析方法，但是我搞不明白什么是数据流图。说明一下，就是一个系统里面有一个大的查询订单模块，里面细分了各种订单的查询模块。怎么绘制数据流图，而且还是一个功能模块的数据流图?

你问

如何绘制功能模块的数据流图？

我答

1．基本概念

结构化分析方法（Structured Analysis Method）是强调开发方法的结构合理性及所开发软件的结构合理性的软件开发方法。

结构是指系统内各个组成要素之间相互联系、相互作用的框架。结构化分析方法提出了一组提高软件结构合理性的准则，如分解与抽象、模块独立性、信息隐蔽等。针对软件生命周期各个不同的阶段，又包括结构化分析（SA）和结构化程序设计（SP）等方法。

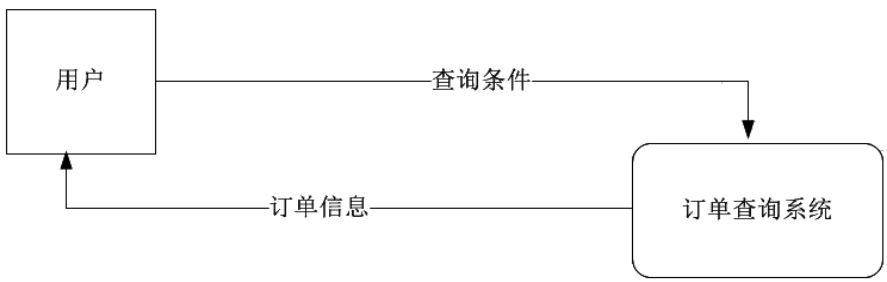
数据流图（Data Flow Diagram，DFD）从数据传递和加工的角度，以图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化分析方法的主要表达工具及用于表示软件模型的一种图示方法。

在结构化分析方法中，数据流图是需求分析阶段的产物。

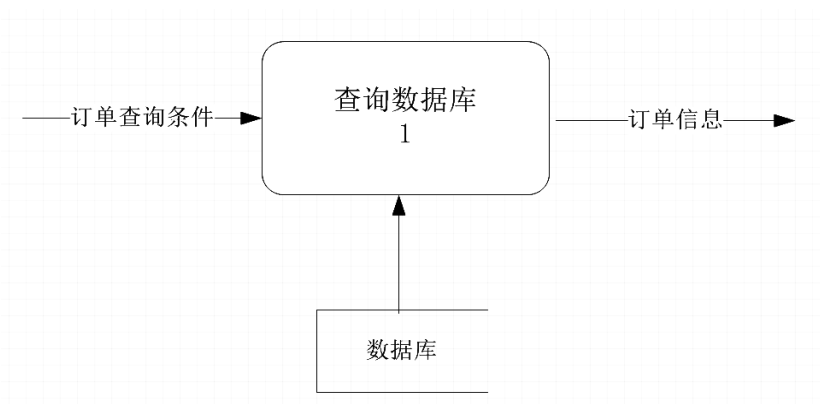
2．结构化分析方法

以订单查询模块为例。

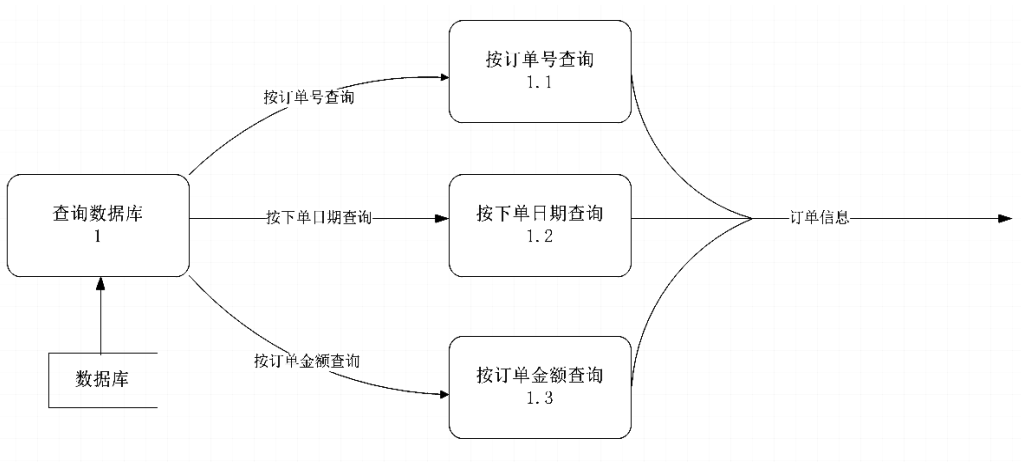
（1）**顶层数据流图**：用户在前端页面输入查询条件，系统依据查询条件搜索到相应的订单数据，返回给前端并展示给用户。



（2）**0 层数据流图**：将前端的查询请求发送到服务器端，服务器端先在缓存里查询，如果查询不到，则再去数据库里查询。



（3）**1 层数据流图**：根据不同的查询条件，如订单号、下单日期、订单金额，从数据库中查询到相应的订单，并将订单信息输出给服务器端。



3．结构化数据流图的说明

（1）**直角方框**：表示数据的源点或终点，是本软件系统外部环境中的实体（包括人员、组织或其他软件系统），统称外部实体，一般只出现在数据流图的顶层图中。

（2）**箭头**：表示数据流，是数据在系统内传递的路径，由一组固定的数据组成。由于数据流是流动的数据，所以必须有流向。除与数据存储之间的数据流不用命名外，其余的数据流都应该用名词命名。

（3）**圆角方框**：表示加工，也就是数据处理，对数据流进行某些操作或转换。每个加工也要有名字，通常是动词短语，简明地描述完成什么加工。在分层的数据流图中，加工还应该有编号。如果 0 层图的编号为 1、2，那么 1 层图的编号就是 1.1、2.1。

（4）**右开口方框**：表示数据存储，指暂时保存的数据，它可以是数据库文件或任何形式的数据组织。

问答（23）软件测试和质量管理是一回事儿吗？

背景

随着很多软件企业的规模扩大化和市场需求的正规化，越来越多企业的招聘岗位里会出现“质量管理”或“质量保证”这样的岗位，这就让从事测试工作的同学觉得有些困惑了：我每天从事的测试工作不就是为了保证产品的质量吗？那“质量管理”又是什么呢？

你问

软件测试和质量管理是一回事吗？

我答

1．基本概念

软件测试（Software Testing）是验证软件的逻辑是否正确、功能是否完整、系统是否安全和质量是否可靠的过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

质量管理（Quality Management）是指确定质量方针、目标和职责，并通过质量体系中的质量策划、质量控制、质量保证和质量改进来使其实现的所有管理职能的全部活动。质量管理是各级管理者的职责，但必须由最高领导者来推动，实施中涉及单位的全体成员。在质量管理活动中，必须考虑经济因素。

2．属性对比

（1）类型。

“ 软件测试 ” 是技术类型的岗位，如软件测试工程、自动化测试工程等 ；而 “ 质量管理 ” 是管理类型的岗位，如 QA（Quality Assurance，质量保证）、QC（Quality Control，质量控制）、QM（Quality Manager，质量经理）、QE（Quality Engineer，质量工程师）。

（2）面向的对象。

“ 软件测试 ” 面向的是产品 ；而 “ 质量管理 ” 面向的是过程。

（3）生命周期。

“ 软件测试 ” 贯穿于整个产品研发生命周期（从狭义上来说，它存在于 “ 编码 ” 之后）；而 “ 质量管理 ” 贯穿于整个公司的流程体系，存在于公司所有部门的各个流程环节，“ 软件测试 ” 只是其中的一个环节而已。

（4）强调。

“ 软件测试 ” 强调的是事后通过相应的技术或工具对产品进行检查，从而保证质量 ；而 “ 质量管理 ” 强调的是在与产品研发有关的所有环节进行流程规范约束和检查，提前预防问题的发生，从而确保质量。

（5）组织结构。

“ 软件测试部门 ” 与 “ 研发部门 ” 和 “ 项目管理部 ” 在整个项目实施过程中形成三足鼎立之势，直接向技术总监汇报 ；而 “ 质量管理部门 ” 一般是公司级的部门，与任何部门都没有隶属关系，直接向公司管理层汇报。

3．个人补充

（1）质量管理体系是一种管理体系流程，也可以说是一种方法论，它采用

的是 PDCA（戴明环）这种核心基础方法，对大多数领域的产品质量管理都是可用且有效的。

（2）我原来在公司做过两年的内部审计，简单说来，不管是 ISO 9000（国际标准化的品质管理）还是 CMMI（软件能力成熟度集成模型），要做的事其实“很简单”：

① 把你正在做事的流程说出来。

② 按你所说的流程去做事。

③ 把你所做的事和流程都记录下来。

④ 检查你所做的事和流程。

⑤ 根据检查出来的问题，持续改进并实践。

（3）软件测试部门在质量管理体系里其实可以扮演一个承上启下的角色，市场 / 客服部门反馈的现网问题经测试统计分析之后，找到可以预防问题再次发生的改进点，再交由质量保证工程师推动实施和检查，形成质量管理体系里的一个小闭环。

问答（24）如何区分测试报告和质量报告?

背景

有位同学问了我一个基础问题：软件测试报告与回归测试报告有何区别？测试细则又是什么呢？简单说来，软件测试报告可以理解成一个统称，而回归测试报告只是其中的一个子项。而测试细则，按我的理解，其实是测试计划里的具体测试策略和步骤，是用于执行参考的，如果把它放到测试报告中，那应该表示测试的执行步骤和细节。而在实际的项目中，大家真正容易混淆的其实是“测试报告”和“质量报告”。

你问

测试报告和质量报告应该如何区分？

我答

开篇先简单补充回答一下背景中的问题。如果从狭义的角度来定义软件测试报告，那它就是每个项目在完成所有测试，准备发布上线时，由测试部门出具的一份测试报告，运维部门以此为上线活动的开始标志。我之前所在的公司就是如此，美国的发布经理不收到测试部门发出的测试发布报告，是不会发出正式的工程发布报告的。

而回归测试报告，其实可以理解成与每个测试包一一对应的回归测试总结。在很多公司里，包括我之前所在的那家公司，测试报告其实就是每日项目报告中的一项内容，并没有出具单独的报告。

言归正传，我想问问你 ：能分得清测试报告和质量报告吗？你知道应该如何去区分它们吗？

你也许会跟我说，我当然能分得清楚，毕竟做了这么多年的项目。

可我如果真的让你去写一份测试报告，你就不会那么清楚了。你可能会详细地罗列出项目中的所有测试环节、步骤、方法，发现的所有缺陷及缺陷产生的原因和分析，在各个环节中遇到的问题，甚至因为开发自测质量不高而导致需求提测时间延迟的问题也会罗列其中。诸如此类的测试数据和问题分析混杂在一起，洋洋洒洒几十页，这是我们常见的一份“ 漂亮 ”且“ 充实 ”的测试报告。

我先不对这种报告做任何正确与否的评判，因为我一直认为，很多东西都没有明确的对错之分，有的只是是否适用于当下的场景。

还是先来说说我将测试报告和质量报告区分开来的依据是什么，然后再回过头来看看上述报告是否合适。

1．报告的阅读对象决定了这两种报告在本质上的区别

测试报告的阅读对象主要是负责产品版本发布的经理。在我之前所在的公司里，这个职位的名称是 Release Manager（发布经理），他阅读这份报告的目的是想从中获取到相应版本的质量是否达到可发布的标准，包括 Pass（通过）、Fail（失败）、Pass with Risk（带着风险通过），我们完成了哪些测试工作，当前遗留的未解决 Bug 有多少，已知问题有哪些等。当然，他也需要从这份报告中了解到这个版本有哪些需求和改动。

质量报告的阅读对象主要是负责产品研发的经理。在一些规模很大的企业里，还有一个岗位的人员会看这份报告，那就是 EPG（过程改进小组），他们阅读这份报告的目的是想从罗列的问题和原因分析中找到所有可以改进的点，有可能是流程上的，也有可能是技术方法上的，当然，还有可能是人的问题。总之，他们就是想通过这份报告里的问题找到可以改进的地方，然后对其进行有计划的、循序渐进的改进。

我们再回到前文提到的那份“ 混合型 ”的报告，你现在应该能告诉我，它是否是一份合适的报告了吧？

对于产品研发经理或者 EPG（过程改进小组）来说，要么对版本需求和改动很熟悉，要么只关注整个研发过程，并不关注产物本身，你在报告里花大量篇幅罗列了需求和改动，没有任何意义。

对于发布经理来说，他只想知道，当前质量是否适合发布，其他的都是研发过程或者研发团队的问题，他并不关心（如果站在产品研发团队的经理角度来说，那些研发过程或研发团队的问题其实都是“ 家丑 ”，那你觉得该“ 外扬 ”吗）。

2．报告的编写人角色决定了这两种报告在直观上的区别

测试报告的编写人是测试经理或测试项目负责人。

质量报告的编写人是 QA。在 QA 由测试兼任的公司里，也由测试经理或测试负责人编写（我之前所在的外企就把测试和 QA 的职责合并在一个岗位里，叫作 QA）。

3．最后列出一些我认为在两种报告中较为关键的条目，但可以因地制宜，不全是必选项

测试报告 ：

（1）版本信息，包括所有发布组件的名称和版本号。

（2） 依赖关系，即所有组件之间的依赖关系，是强依赖，还是弱依赖。如果有依赖关系，则一定要说清楚上线的部署顺序和原因，以免出现问题。

（3）测试环境配置，包括服务器的系统版本、中间件的版本、数据库的版本、客户端的浏览器类型 / 版本、操作系统类型 / 版本等。

（4）已完成的测试范围、类型和方法，包括结果。如果进行了性能测试，则附上性能测试结果 ；如果进行了安全测试，则附上应用程序 Scan（一种安全扫描工具）的扫描结果 ；如果执行了手工用例，则附上用例的执行情况，等等。

（5）测试结论。包括 ：版本发布的标准是什么？依据标准，该版本是否可

发布 如果不可发布，原因是什么？如果是可带着风险发布，那么风险有哪些？

（6）已知问题清单。即该版本截至发布日期，还有哪些是未在当前版本中得到修复的、仍然存在的缺陷。

质量报告 ：

（1）同类型项目的历史质量数据对比。

（2）当前版本的质量期望目标。

（3）经过统计分析的缺陷分布图和缺陷修复曲线图。

（4）针对过程中遇到的问题所做的 3W 分析 ：

① What’s the problem ？（问题是什么？）

② What’s the root cause ？（根本原因是什么？）

③ What’s the solution ？（解决方案是什么？）

（5）针对当前版本的质量问题（数据的和过程的）所做的总结分析和改进建议。

第 4 章　如何度过职场的倦怠期

问答（25）职场中遇到问题时应该怎么办?

背景

每天都有同学在问我问题，各种各样的问题，各种各样的场景，先不看具体的问题，而是看从对问题的描述中折射出的遇到问题时的态度，有客观的、积极的、烦躁的、害怕的，也有无奈的。所以，对于大多数人来说，比解决具体问题更前置的问题是：我遇到问题时该怎么办?

你问

职场中遇到问题时应该怎么办？

我答

在职场中，遇到问题是不可避免的，所以我们也不要抱着侥幸心理去想方设法地降低遇到问题的概率，而应该简单地看待问题，其实就意味着两样东西：

问题 = 麻烦 or 机会？

我们可以先停一停，想一想自己在平时的工作和学习中，如果遇到问题，则会下意识地选择哪个答案呢？

问题如果没有被解决或者没有被处理妥当，那的确会变成麻烦，甚至会变成大麻烦。所以，很多人在工作中会回避问题，或者说很多人在被问题找上门

时会十分焦虑、害怕，甚至会选择逃避。

我们可以换一个角度来看待问题 ：

如果没有问题，那怎么展现你解决问题的能力？

如果没有问题，那怎么找到可以改进提高的地方？

如果没有问题，那怎么体现你在团队中的价值？

如果没有问题，那怎么让你明确下一步的学习方向？

在项目进行中遇到技术上的一个难题，大家都束手无策，而你却在很短的时间内解决了这个问题，你觉得你的老板在加薪时会不会想到你？

在解决问题的过程中，你是不是也积攒了相应的经验，而且可以输出成你的技术笔记，分享给其他同事，在以后的项目里如果遇到同类型的问题，则也可供参考，你觉得你的同事在 360 ° 考评时会不会给你比较高的评价？

在遇到问题的时候，除第一时间要解决问题之外，还需要分析导致这个问题的根本原因，找出这个问题的根源，从而找到对应的流程或工作方法里可以被改进的地方。很多改进点就是在不断地遇到问题、解决问题、总结问题的过程中被发现的。

当其他的合作部门带着问题来找你寻求帮助或支持的时候，不管你能否解决该问题或者该问题是否应该由你解决，你都可以帮忙看一下，然后及时给对方反馈，告诉他问题解决了或者没有解决的原因，最好能提供相应的建议。

同时，你还可以从这些问题中获取到很多有用的信息或者机会。

比如一些总是重复被求助的问题，你可以总结归纳这些问题，先把它们分门别类，再列举出有效的解决方案，做成一个 FAQ 形式的手册，提供给相关的部门。这样做既能减少你的重复解答次数，也能提高对方再次遇到类似问题时的解决效率。

还有一些需要耗费较多人工检查的重复性问题，是否可以有工具或脚本来代替，这不就是技术方向的需求吗？这样的产物是不是能起到双赢的效果？

我们要尽量避免把问题推回去或踢给其他人，因为别人既然找你，肯定是相信你或者倾向于找你，换句话说，别人在遇到问题时总是第一时间想到找你，也证明了你的价值。你如果总是拒绝，慢慢地，问题是少了，但是，别人也不会再相信你，或者说机会也不会再光顾你。最佳的结果是 ：让问题止于你。

如果你在平时的工作和学习中发现自己遇不到太多的问题，或者问不出什么问题，则千万不要以为是自己的能力已经很强了，什么都会了，所以才遇不到什么问题。恰恰相反，那也许说明你已经进入了一个看似零问题的舒适区，没有问题，你就无法知道自己的不足在哪里，也就无法知道自己学习的方向和目标是什么，你会在这种零问题的环境下一直原地踏步、停滞不前。

所以，当我们在职场中遇到问题时 ：

正确的态度——积极、庆幸、开心、认真 ;;

正确的做法——寻找、拥抱、解决、利用。

问答（26）上班偷懒就是占了老板天大的便宜吗？

背景

这段时间在一个微信群里频繁地看到一个朋友准点炫“闲”，而且他总是不停地在说，最近什么都不想干，只想偷会儿懒，做一天和尚撞一天钟。群里还有不少人天天一看他炫“闲”就好羡慕。群里有一个人私下问我：为什么感觉那个人就像占了老板天大的便宜一样，真的是这样吗？

你问

上班偷懒就是占了老板天大的便宜吗？

我答

每个人都会有职场懈怠的时候，每个人都会有偷懒的时候，每个人都会有做一天和尚撞一天钟的想法，这是很正常的现象，是我们每个人在职场上都会经历的时期，而且并不是一次性的经历，往往是间歇性的。

遇见它们并不是坏事，我认为这就好比长跑和骑行进藏，一开始都是劲头十足、精神抖擞，沿途看到的都是美景。可随着体能消耗越来越大，身体和精神上都会越来越疲惫，在某个时期，你看到的再也不是沿途的美景，而是蜿蜒蜒、总是看不到尽头的道路，脑海里总是时不时地冒出来一个个念头：歇歇吧，别再向前了吧，放弃吧……我跑这么远，骑这么远，到底是为了什么呢？

只要我们能正视它，清楚地认识到这是一种必然会进入的状态，也是必须经历的一段时期，那我们就能较容易地调整好自己的心态，采取适合自己的方法，度过这段懈怠期。

最怕的是很多人在进入懈怠期时不自知，当提不起劲工作或者偷懒时，总

认为自己占了老板天大的便宜，觉得老板已经为自己一天的 8 小时买了单，如果自己每天摸鱼打诨 2 小时，那就变相地赚了 2 小时。

其实不然，先不说时间的不可逆性，浪费了就是浪费了，没有什么加班加点补回来一说，每个人每天可用的时间都是一样的。你所理解的赚了 2 小时，并不代表你比我每天就多了 2 小时。你偷懒浪费掉的 2 小时，我相信多半花费在了上网、聊天、刷新闻、看小说等对于你自己没有任何增值作用的行为上。

你浪费的这 2 小时，对于老板来说，只是少了一些产出，或者说在某个项目上的进度会出现一点滞后，其实从他的角度来说，无非就是多增加了 2 小时的人工成本而已。可是对于你自己而言，损失的东西可不是用金钱就能衡量得了的。**你坐在那里摸鱼打诨时，从表面上看浪费的是你宝贵的生命和大好的时光，其实浪费最严重的是你从表面上看不到的，那就是你的激情和机会。**

我想说的是，不管当下的这份工作对于你来说是喜欢还是不喜欢，我们都应该正确地对待它。

如果是你喜欢的工作，这 2 小时就可以用来增加你当下岗位的工作经验，你应该好好利用在办公室里的每一分钟，做好积累、总结、沉淀，为自己的职业发展道路铺下一块块结实的路砖。

即使是你不喜欢的工作，你也可以采取“ 二八 ”策略，花 20% 的时间和精力完成当下岗位的职责所在，剩下的 80% 可以用在自己真正想做的事情上。不过前提是你真的能找到除当下工作之外真正想做的事，否则，还是建议你学会在当下的工作中找寻你的兴趣点，如果你改变不了它，那就学着去适应它。

所以不要再觉得上班偷懒就是占了老板天大的便宜，你眼里那些不会占便宜的人其实早已经在你没有发觉的时候走在了你的前面，现在去追还来得及，我想没有人愿意被别人落下一大截吧？

问答（27）“老兵混职场”之打破职业发展瓶颈的终极绝招是什么？

你已经工作几年了？三年？五年？还是十年？

你是不是对自己接下来的职业发展感到困惑？

你是不是经常会问自己当下到底应该学些什么？

你是不是回过头看自己这三个月、半年，甚至一年，觉得自己没有一点变化或进步？

你是不是越来越难说出你相比于其他人的优势，除了“ 工龄 ”？

如果这些问题你或多或少都已经遇到了，那恭喜你，你摊上事了，摊上大事了 —— 你遭遇了职业发展的瓶颈。

之前，团队里的一位成员给大家做了一次关于用例设计方法的分享，参与的人有刚刚工作三四年的“ 孩子 ”，也有像我这种工作了十几年的“ 老人 ”。

听完整个分享之后，我觉得受益匪浅，不仅从中学到了一些我之前只知其然而不知其所以然的概念，还再次印证了我的观点，也就是我今天想和大家聊的，打破职业发展瓶颈的终极绝招 **：回归理论学习**。

我想很多人应该都有过跟我一样的经历，不管是测试还是项目管理，都是从零开始实践，通过实战摸爬滚打积累一定的认知和经验。因为在我刚入行的时候，国内还没有开始重视软件测试，也没有几家公司有正式的 QA 或测试岗位，所以相关资料很少，大部分都是原版的，更别说什么软件测试培训班了。

所以我都是一边做项目一边积累经验，于是我就默默站到了所谓的“ 实战派 ”阵营里。为什么这么说呢？因为在相当长的一段时间里，我总是瞧不上所谓的“ 学院派 ”，对那些理论概念和方法总是嗤之以鼻，认为那些人都是在纸上谈兵，看那么多理论还不如到项目组里做几个项目来得快。

现在回过头来看看，那时候哪能被称作“ 实战派 ”啊，其实就是“ 野路子 ”，就像打羽毛球、打乒乓球或者下象棋一样，我走的都是“ 野路子 ”，所以都有同样的问题，也是同样的原因。

我们先来看看“ 实战派 ”和“ 学院派 ”相比有哪些不足。

（1）**可复制性差** ：套用学习 PMP 时看到的一段话，PMP 就是众多项目管理人员从丰富实践中提炼出来的一套思想、流程、方法和工具，虽然你不使用这套东西也能得出正确的结果，但你并不能保证自己每次都能得到正确的结果。其实我在这里想表达的就是这个意思，理论知识和方法相对于实战经验来说，对结果正确性的保证更加可靠。

（2）**可分享性差** ：因为都是从实战中摸索出来的路子，没有基于理论的分析和归纳，只能属于个人经验范畴，所以很多东西只可意会，不可言传，于是很难分享给其他人或者指导、培训别人。

（3）**可优化性弱** .：对于实战派来说，优化也是基于更多的实战，所以存在基于错误经验去优化另一个经验的可能性。如果是基于理论知识和方法去优化实战方法，则可靠性更强一些。

说到这里，我们再来看看很多人所谓的职业发展瓶颈，其实就是缺乏理论知识和方法的支撑，导致在实战中成长到一定阶段之后，成长速度就会变得缓慢，甚至会慢到感知不到。所以我个人觉得，这是回归理论学习的最佳时机。

再来说说我自己的经历吧。

（1）在做了大概 10 年软件研发类的项目管理之后，我开始发觉我找不到可以改进的地方了，不论是流程、方法还是工具，觉得都已经很成熟了，而且每个月的项目也都没有任何问题。于是我就开始去学 PMP，其实并不想考什么证书，只是想通过备考过程，跟着老师从头梳理并学习项目管理的理论知识。随着学习的深入，我才发现自己之前做的只是项目管理领域的一小部分，使用的很多方法其实都是有相应的理论方法的，有用的对的，也有用的不对的，突

然发现可以改进的地方数不胜数。

（2）我有一段时间也很迷惘，感觉自己是不是做测试到了一个瓶颈期，于是就想是不是要转型才能解决问题？想了很久，我也没想清楚，于是我就让自己沉下心来去读书，去学习理论，在学习的过程中会结合之前通过实践掌握的经验和知识进行对比和思考，然后再去实践和总结。

同时，我会找各种机会去分享自己的学习所得，通过各种输出手段倒逼自己不断地去总结、分析、改进。一段时间之后，你会发现之前所谓的瓶颈早就不知去向，而你之前所了解的只是这个领域很小的一部分。

回归理论学习，不代表止步于理论学习。最佳实践公式为：实战积累 + 理论学习 + 总结分析 + 实践改进。

问答（28）每天忙忙碌碌，可为什么还是觉得什么都没有学到呢?

背景

很多测试新人在工作了半年或一年之后，经常会觉得没什么成长，跟之前的自己相比，并没有什么进步，于是就会对现状很迷惑，自己明明每天都忙忙碌碌的，也很努力地在工作，可为什么还是觉得自己什么都没有学到呢?

你问

每天忙忙碌碌，可为什么还是觉得自己什么都没有学到呢？

我答

GTD 就是 Getting Things Done 的缩写，直译过来就是“ 把事情做完 ”。GTD 的核心理念概括起来就是必须记录下要做的事，然后整理安排并使自己依次去执行。GTD 的 5 个核心原则是 ：收集、整理、组织、回顾、执行。

为什么先说 GTD 呢？因为在 GTD 的理论里有一个核心原则叫作“ **回顾** ”，它的本意其实就是让你至少在每周结束的时候回顾一下自己本周的待办事项，并根据最新的情况进行总结、分析和调整。

理论解释如下 ：

如果你不至少每天或者只要有时间就回顾检查，那么你的行动和提醒的列表将会变得毫无用处。以你当时拥有的精力、资源和时间，决定什么对你来说是最重要的事情，然后去做。如果你倾向于拖延，那么你可能会总是做最容易的事情，而避免那些困难的。为了解决这个问题，你可以一个接一个地做列表上的事情，按照它们的顺序，就像你处理你的收件箱一样。

至少以星期为周期，GTD 要求你回顾所有比较主要的“ 行动 ”“ 项目 ”和“ 等待 ” 的事项，确保所有的新任务或者即将到来的事件都进入你的系统，而且所有的事情都更新到符合最新的情况。

能对自己的现状产生迷惑，觉得自己在过去的一年中什么都没有学到的人，我想多半是一个会做定期回顾的人。就算没有做定期回顾，那至少也会在一年当中的某几个特定时刻对自己做一次总结回顾，比如月度总结、季度总结或者年终总结。

所以，很多人在工作了三个月、一年、三年，甚至五年之后，经常觉得自己没什么成长，跟几年前相比也没什么进步，于是他们就会心生迷惑，就会感到焦虑，然后就会开始分析 ：

是不是我选择的方向不对，那我是不是应该换一个方向呢？

还是 GTD 里的“ 回顾 ” 原则其实没用，那我是不是应该换一种时间管理方法呢？

……

在绝大多数时候，这种问题都跟方向无关，也不是方法的错，方法本身其实就是根据大量的实践检验而总结提炼出来的一种行之有效的手段。

我们可以先假设“ 回顾 ” 原则对我们的成长肯定是有帮助的，那问题肯定就出在我们自己身上，不是理解偏差的问题，那就是执行的问题。

我相信很多人跟我一样，不管是不是采用 GTD 这种时间管理方法在管理自己的工作和生活，刚开始的时候，我每天都会进行回顾，但慢慢地就会变成隔天回顾一次，然后就是一周回顾一次，再往后就不再回顾了。

原因是我那时候只是对所有待办事项的完成情况做一次回顾，已按时完成的直接忽略，未按时完成的就调整计划，提醒自己在什么时候要赶紧完成，仅此而已。所以我每天都觉得只是在重复做 To-do-list 的调整，并没有什么收获，

于是就放弃了。

既然找到了问题的原因，于是我就思考，是否可以改进一下“回顾”的方法呢？因为我认为那种逐条逐项的检查产生不了收益，所以我就想，是否可以**在每天结束的时候，从当天的已完成工作中挑出一项自己满意或者不满意的进行复盘**。

在对这件已完成事项进行复盘的过程中，我们可以把事情的起因、背景、采用的方法、执行的步骤，以及最后的结果都写下来，然后从头开始审视每一个要素和环节，进行客观的衡量和分析，看哪些做得好、哪些做得不好，假如换一种思路或者方法，结果是否会更好一些？对于每一个环节，多做几次这种假设，最终我们可以归纳出一套能取得最优结果的方法或思路，并记录下来，在以后的同类型任务中可以去实践和改进。

复盘的作用我们已经说过了，再来说说为什么一定要“**每日**”复盘。这只是一个建议频率，因为我个人认为，现如今是一个快节奏的时代，很多时候不仅仅要靠“质”取胜，也要靠“快”取胜，所以才需要小步快跑，将复盘的频率提高到极致。

另外，现在我们每天处理的信息量很大，如果将复盘的频率设为每周或每个月，那么你真正能做到复盘的还是你做复盘那天所做的事情。如果你不信，则可以回想一下，你上周完成了哪些事情？你是否能回想起每件事情的起因、背景、采用的方法和执行的步骤？我猜，你大概能记住的也只是在 Done-List 里记录的完成结果吧。

综上所述，你之所以每天忙忙碌碌的，但还是觉得自己什么都没有学到，是因为你没有做到**每日复盘**。我想跟你说的是，在职场中，虽然知识的学习、理论的实践和经验的积累都很重要，但它们并不是拉开你和其他人的差距的关键性因素，而**每日复盘是真正能让你和其他人拉开差距的决定性因素**。

问答（29）需不需要“死磕”自己的“短板”？

背景

前段时间给一位同学做职场发展方向的咨询，他一方面很清楚自己的优势和长处，但另一方面又一直纠结自己在逻辑思维方面的短板。在后几次跟我的沟通里，他也曾有意识地问我，是不是太过纠结于自己的不足之处了？

你问

需不需要死磕自己的“短板”？

我答

“木桶定律”讲的是一只木桶能盛多少水取决于它最短的那块木板。一只木桶想盛满水，必须每块木板都一样平齐且无破损。如果这只木桶的木板中有一块不齐或者某块木板下面有破洞，这只木桶就无法盛满水。一只木桶能盛多少水，并不取决于最长的那块木板，而是取决于最短的那块木板。这种现象也被称为“短板效应”。

我们在学生时代就被灌输了这种理论，老师会说：“×××，你的语文、数学成绩都不错，但英语是你的短板，你要好好补一补，不然你的总分排名就上不去。”于是 ××× 就开始通过早起、晚睡或培训班等各种方法来提高自己的英语成绩，想补上这块短板，从而让自己的总分提高一些。

在职场中，有那种每块板都很长的人，虽然人数并不多，但他们确实很优秀。还有一部分人没有特别短的板，也没有特别长的板，落差不大，这部分人数不少，也很常见，你既说不出他太多的缺点，又找不到他太多的优点。

还有一类人，我觉得也是相当常见的，他们会有一项特长，但或多或少有

一些短板，可他们需要补上他们的短板吗？我认为是不需要的，因为他们有非常突出的长板，这就是他们的优势，这个优势足以弥补他们其他的所有短板。因为他们够专，在某项能力或某个领域中具有绝对的长板优势，所以，他们的短板就显得不那么重要了。

而且，正所谓“ 人无完人 ”，作为一名普通的职场人，我不会苛求自己去成为那种全是长板的高手，但我也不会放任自己成为那种让别人既找不到优点又挑不出缺点的人。我会坚定地走在挖掘自己的优势长板且深耕延展的道路上。

不过短板理论如果落在任何一家企业中，或者初创公司的管理者身上，就不能像个人那样去处理了，因为短板部分往往决定了整家企业的水平及其能发展到哪个层次。所以，创始人和企业本身都需要正视自己的短板，但不代表一定要通过提高自身的短板来让它变为长板。锤子科技创始人罗永浩的一个观念我十分认同，他认为，你自身的短板不要总想着自己去补上，因为你可以一辈子都补不好，与其浪费精力去做这种事，不如通过引进相应短板处的人才，通过互补机制，以团队这个整体来解决企业或者创始人的短板问题。

不管你是一名普普通通的职场人，还是一位初创公司的管理者，都应该清醒地找出自己的短板并正视它，不要刻意地去回避或者视而不见，只有知道自己的短板在哪里，才能充分利用身边的资源去弥补，或者扬长避短，将短板对自己的影响降到最低。但千万不要死磕短板，而要结合你当前的岗位，或者说即将上任的岗位，具体短板具体分析，看看这个你自认为的短板在相应的岗位要求里是否占比很大，再决定是需要自己提升它，还是通过其他方面的加强来弱化这个短板。

问答（30）如何完成“重”任?

背景

我今天回到家才得知孩子连晚饭都没吃，还在书房里跟试卷发脾气呢，因为考试成绩没达标，老师罚抄他们一遍试卷，还有一份英语试卷要做、一个单元的语文生字要听写，还有数学复习等作业。从量上来看，的确有些多，而他从放学回家开始抄试卷，已经抄了 2 小时，才抄了 2/3，一边哭，一边说有太多字要抄，好烦；还有好几样作业都没写，好急。问我，能不能不抄试卷了？

这让我想起今天有位同学跟我说，明天他们就要开需求评审会了，可他还有 5 份英文的需求文档要看，而且还有好多看不懂，不仅仅是英文，而是需求内容本身。所以他很焦虑、很着急，心里也很虚，问我怎么才能快速理解那么多英文的需求文档。

两个不同场景下的问题，其实深究起来却很相似，都是在“重”任和“死”线的双重重压下不堪负荷，产生了焦虑和急躁的情绪。

你问

哭泣和焦虑能帮我完成“重”任吗？

我答

先说一下我是怎么解决孩子的问题的。我让他先不要抄试卷了，去喝口水，然后写较为容易的英语试卷，完成后再去抄试卷。当英语试卷写到一半的时候，他的情绪明显好了很多。等他写完英语试卷，再次开始抄试卷时，一点情绪都没有了，很快就抄完了。

再来说一下我是怎么建议那位同学的。我让他先不要纠结英文的理解，准

备好有道词典，遇到不认识的单词就查一下。先看需求文档的目的和概述，再找出该需求的主线流程，做到理解这个需求的主要背景和功能概要，然后按照主线流程上的一个个节点去了解逻辑细节。

一个是小朋友，一个是大朋友，所面临的问题其实就是我们很常见的 ：同时承接了好几个任务，其中还有一个是“ 大青蛙 ”，这里的“ 大青蛙 ”指的就是那种在主观意愿上觉得很难、很麻烦、一直不愿意开始去做的任务。再加上任务通常都会有 Deadline（最后期限），这样一来，任务就会给我们带来双重压力。

你们好好想一想，自己在日常的工作中是不是都遇到过相同的场景？那你们又是怎么解决的呢？

我先声明一下自己的观点 ：哭泣和焦虑是解决不了这些问题的。

再来说说我常用的解决方案，也许并不适合所有人，但也会对一部分人有所帮助。

（1）建议集中优势精力和时间段先吃掉“ 大青蛙 ”，因为它在体积和分量上都会给你造成相当大的压力和紧迫感，如果先把它吃掉，你的心理压力就会减轻很多，这对你完成剩余的几个小体量任务会有很大的好处。

（2）如果是那种机械式、容易产生疲惫感和厌烦感的“ 大青蛙 ”，则不需要一气呵成，在任务内没有太多紧密关联的可以拆分成若干部分，完成一部分，就切换去完成一个小体量的任务，再切换回来继续，以避免长时间作战，从而产生厌烦、抵触情绪。

（3）如果是那种有相互关联、逻辑性较强的任务，比如前文中提到的需求文档阅读，则可以先粗后细，从主体框架和流程开始熟悉、理解，在有了整体上的认识之后，再逐一细化了解，各个击破。这样，在熟悉了主体框架和流程之后，就能做到心中有数了，一方面可以更加从容地、逐字逐句地阅读细节 ；另一方面，即使细节没有读完，去参加需求评审或需求讨论，也不会心里发虚，两眼一抹黑了。

（4）最后，个人建议，最好还是在承接任务时就做好计划，管理好自己的时间和精力，尽量不要并行执行太多的任务。因为从注意力角度来说，我们可以同时“ 做 ”几件事，但是我们肯定不可能同时“ 专注 ”几件事，无论是从结果质量还是从过程效率来看，这都是有很大差别的。

问答（31）如何快速缩短职场倦怠期?

背景

老师，我一直努力地工作和学习，可是最近每天感觉都特别疲惫，不想看需求文档，不想写测试用例，不想写自动化脚本，每天从上班开始就盼着下班，总觉得时间过得特别慢，下班回到家就瘫在沙发上，不再像前段时间那么认真地看书学习了。您说我是不是变懒了？是不是不求上进了？我该怎么办呢？

其实，你一直都很努力，也很上进，你现在只是遇到了一个特殊的时期，这其实是我们每个人都会遇到的，那就是职场倦怠期。

你问

如何快速缩短职场倦怠期？

我答

“职场倦怠”又称“职场枯竭”，它是一种由工作引发的心理枯竭。职业倦怠现象是上班族在工作的重压之下所体验到的身心俱疲、能量被耗尽的感觉，这和肉体的疲倦劳累是不一样的，而是缘自心理的疲乏。

这是一种很常见的职场病症，每个人在自己的职业生涯中多少都会经历几次这种时期。常见的症状有如下几种 ：

（1）突然对工作失去热情，产生抵触情绪，觉得前途渺茫，对周围的人和事都漠不关心。

（2）工作态度消极，对服务对象缺乏耐心，比如开发人员厌倦写代码、测试人员不想测试等。

（3）感觉不到自己工作的意义和价值，到办公室的时间开始推迟，或者请假变多，打算跳槽甚至转行。

当你发现自己有了上述症状中的一项或多项时，说明你已步入职场倦怠期。产生这些症状的原因通常有以下几种 ：

（1）因为长时间的工作压力或长期高强度的工作节奏而产生了强烈的疲惫感。

（2）因为长期的机械式、重复性工作而产生了明显的枯燥感。

（3）对工作环境和流程规范有较多的不满。

（4）因为胜任不了当前所承担的工作，每天既焦虑又没有成就感。

（5）因为工作量、报酬、考评不公而导致的感情衰竭。

从现象和起因上来看，职场倦怠肯定是不好的，但我认为，我们不用去刻意地回避它或者预防它，因为它是一个我们必然会经历的时期，我们要做的，也就是今天我想教给大家的，就是怎样才能大大地缩短职场倦怠期。

（1）多与领导和周边的同事沟通，将心里觉得不公平的事情说出来，孰对孰错，说开了比闷在心里要好。

（2）凡事要量力而行，不要承接挑战难度超出自己可承受范围的任务，这样一方面很难完成，另一方面会产生强烈的挫败感。

（3）调整工作计划，把一些不是很重要、很紧急的任务暂时搁置一下，集中精力在少数几项重要的任务上，将自己的工作强度适当降低一些、更聚焦一些。

（4）减少被动加班时间，在日常的机械式、重复性工作中找寻是否可以由工具来代替，以此为目的，学习或尝试一些新的技术和工具，找出一些有趣的或者以前没做过的事去做。

（5）下班后，找几本自己感兴趣的书，与自己从事的职业有所关联最好，

静下心来阅读，不要抱着太强烈的学习目的，暂时只是为了读书而读书，而不是为了输出而读书。

（6）最重要的还是心态。当倦怠期来临的时候，要坦然接受它，而不是因此而焦虑，也不要回避它，仍然保持当前的工作状态，假装什么都没发生，通过调整和应对来缩短倦怠期的长度。

问答（32）什么时间是做职业规划的最佳时机？

绝大多数人在迈入软件测试这扇大门之后，时常会问自己这样一个问题 ：等我做几年测试之后，是走技术路线还是走管理路线呢？

因为在国内软件企业的管理方法上，总会给人这样一种引导 ：技术不可能做一辈子，最终还是要走管理路线的。

很多人喜欢拿年龄说事，说到了多少岁之后，你的技术更新就比不过年轻人了，都一把年纪了还是一个兵，你这资历不做管理真是浪费了，等等。特别是在一些企业当中，你会发现，当一位工程师做得比较出色或者优秀时，领导就会提拔他去做管理，反正不是测试组长就是项目经理之类。

注意观察一下这样的人，他们基本上只会有两种结局 ：一是发现自己的确更适合做管理，慢慢地就喜欢上了管人，不再钻研技术了 ；二是不喜欢管人，只喜欢单纯地研究技术，但又不得不身兼两职，精力分配不过来，自己既痛苦，事情又没做好。

我所在的第一家公司里有一位经常打交道的美国同事，50 多岁了还在写代码，天天乐呵呵的。其实在外企里，工程师的等级最高能达到 13 级，相当于管理路线中的 VP 等级，但就是一个普通工程师的角色，并没有人会跟你说，你技术好就一定要做管理，真正做管理的经理都是那些喜欢管人且擅长沟通的，而技术只是辅助他纵观全局的一种工具或手段。

我在给别人做职业规划或者软件测试职业咨询的时候，不是很喜欢按照“ 技术 ”和“ 管理 ”去绘制路线，而是喜欢按“**“深耕细作”**”和“**“广度发掘”**”去区分。其实不管是什么岗位，在这两个维度上都有较大的发展空间。

要么你在这个岗位的某个领域或某些相关联的领域里深耕细作下去，成为某个领域的专家和权威 ；要么你就在这个岗位上先精通一个领域，再延伸拓展，去发展你的广度。

而且你也不用急于求广，因为在你深耕的路上不可能完全不涉及其他，但

由深耕而触及的其他会是你真正需要的，而不是刻意或者赶潮流赶出来的。

我们可以把职业发展图谱想象成一棵大树，不管你是想长成一棵白桦树，还是想长成一棵大榕树，你首先都要有一副坚实的主干，**只有主干足够结实，你才有足够的支撑去广度发掘**。

最后，附上我绘制的软件测试工程师职业发展图谱。

1. 深耕细作

1.1 黑盒

（1）业务功能测试专家。

（2）用例设计专家。

（3）需求测试负责人。

1.2 灰盒

接口测试专家。

1.3 白盒

单元测试专家。

1.4 性能测试专家

1.5 安全测试专家

1.6 自动化测试

（1）Web 端自动化测试专家。

（2）应用程序端自动化测试专家。

（3）持续集成专家。

（4）Lab Admin（实验室管理员）。

1.7 云测试

2. 广度发掘

2.1 项目管理

（1）项目经理（PMP）。

（2）敏捷教练（Agile）。

2.2 团队管理

（1）测试组长。

（2）测试经理。

2.3 质量管理

（1）SQA（质量保证）。

（2）SCM（配置管理）。

（3）EPG（过程改进）。

3. 全栈

3.1 软件测试架构师

3.2 软件测试咨询师

第 5 章　除软件测试工作外，还需要了解的相关领域知识

问答（33）软件工程师也应该具备产品化思维吗?

背景

有位同学跟我讨论自动化测试工具的构思，我给他的建议都是将这个工具产品化的思路，所以，在整个讨论过程中，他总是不断地问我：“软件工程师也应该具备产品化思维吗？”

你问

软件工程师也应该具备产品化思维吗？

我答

我们先来看看产品化思维和软件工程师思维具体有什么不同。

比如，**一套好的性能测试方案应该包含什么**？

软件工程师思维：有测试计划、测试脚本、测试环境、测试报告，能支撑项目组完成项目当中的性能测试任务即可。

产品化思维：具有指导性的作业文件、测试计划模板、独立的测试数据和测试脚本、分布式测试环境搭建脚本或手册、测试报告和相应的分析模板，能支撑一套完整的性能测试框架迅速落地，快速适应不同的项目，并且能让测试

工程师以最小的学习代价完成性能测试任务。

特别是性能测试，从表层来看，既复杂，又庞大，但如果有一份很好的指导性文件，可以让测试人员对照着标准和步骤去分析需求和执行结果，也许性能测试对于一般的测试人员来说就没有那么“高不可攀”了。

又如，**一个供运营人员使用的数据扫描工具应该是什么样的⁇**

软件工程师思维：一个脚本文件和一个日志记录文本，用户运行脚本文件，执行完毕后，打开日志检查是否有错误结果。

产品化思维：一个网站，一个执行按钮，用户无须在本地管理一个不知道是否为最新版本的工具，只要访问这个网站，选择需要使用的扫描工具，单击执行按钮即可。等待执行完毕后，查看收到的扫描报告。

所以，学会产品化思维很重要。产品化思维不仅可以帮助我们找到有高回报的功能亮点，还可以让我们设计出来的工具或方案更具有可推广性和可落地性，而且可复制性也比较高，便于用户使用，或者说使用的门槛极低。从公司运营角度来说，会带来人力成本的降低 —— 站在公司管理层的角度，你的产物有这一项益处就够了。

我们在平日的工作中可以多锻炼自己的这种思维方式。不管是在找寻目标，还是在制订计划时，都可以把要设计的工具或者方案看成一个产品，不仅要考虑这个产品本身的实现技术或者工作量，还要多考虑一下作为一个产品，其易用性如何？是否能快速实施？是否具有一定的可适配性？用户使用或学习它的成本有多高？后期升级和维护成本大概是多少？

产品的价值预估也很重要，换句话说，就是这个工具或这套方案能够帮助客户 / 用户节省多少人力成本，它可以帮助你在面临多个选择时做出最正确的选择。

看到这里，也许你会觉得，怎么软件工程师设计一个工具或者一套方案要考虑这么多东西，什么值不值得、回报收益、人力成本等。你是不是觉得这么

考虑太功利了？

那你再想想，自己在平日的工作中是不是有绩效考核的约束，绩效考核是不是跟你的工资、奖金、升职挂钩？

如果对上述问题的回答是 Yes，那么我们在设计工具或方案时，把它当作一个产品去考虑思量，除仔细制订研发计划之外，还要更多地考虑如何以最低的成本去研发、如何更好地推广和运营它，以此来获取丰厚的回报，是不是也就无可厚非了？

问答（34）什么是接口测试？为什么要做接口测试？

背景

“老师，我想问一下，什么情况下需要用到接口自动化测试？为什么要做接口自动化测试？是不是两个系统之间交互就要用到接口自动化测试？”这是一位同学曾问我的问题，跟他沟通后，我才明白，其实他想了解的并不是什么接口自动化测试，而就是接口测试本身。现在很多有关接口测试工具或自动化测试工具的文章和书会让新入行的同学把接口测试和接口测试工具搞混淆，于是我今天就想说说我理解中的接口测试。

你问

接口测试的是什么和为什么？

我答

这个问题其实很好回答，我们先来搞清楚相关的基本概念。

接口，按我的理解描述它，其实就是由一段具备逻辑处理功能的程序代码组成的，可被其他方法、服务或应用所使用。

对于调用接口的那一方，可以把接口看作一只黑匣子，只需要负责按约定传入参数，再接收返回的数据，而不需要知道黑匣子里的逻辑。

接口的作用如下：

（1）系统与系统之间的调用。比如银联会提供支付接口给负责第三方支付的应用程序调用，应用程序在用户发起支付请求时，将相关的必要参数值通过支付接口传给银联服务器，银联服务器处理完成之后会调用应用程序方的回调接口，返回支付处理结果。

（2）前端应用对后端服务的调用。比如应用程序调用服务器端的接口，服务器端调用 DAO（Data Access Object，数据访问对象）的接口。以查询航班的应用程序来说，应用程序本身其实主要包含两部分：一是交互；二是数据展示。应用程序通过服务器端提供的接口将需要查询的航班名称传给服务器端，服务器端调用数据访问对象的接口从数据库中获取到相应的数据，服务器端接口再将数据做相应的处理并最终返回给应用程序，应用程序将其展示出来。

（3）服务与服务之间的调用。比如，在注册用户时，会先调用查询用户信息的服务，目的是检查该用户是否已经注册。如果返回的结果是该用户已存在，则负责注册用户的接口就会把该结果返回给前端页面。

接口测试是测试系统组件间接口的一种测试，主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。测试的重点是检查数据的交换、传递和控制管理过程，以及系统间的相互逻辑依赖关系等。

为什么要做接口测试？

（1）现如今，互联网应用程序产品的系统复杂度不断上升，只靠手工的前端测试很难确保很高的覆盖度。通过接口测试，我们能模拟出各种类型的入参，包括一些在前端模拟不出来的入参，还能根据接口文档的定义，设计出相对完善的入参值，力争在接口层先保证质量，剩余的绝大多数问题就是应用程序自身的交互和数据展示问题。

（2）接口测试相对于交互界面测试和功能测试来说，更容易实现自动化，执行起来比较稳定，维护成本也比较低。

（3）接口自动化适用于现网巡检、回归测试等，可以减少人工回归测试的人力成本。

（4）现在很多系统前、后端是分离的，从安全层面来说，只依赖前端进行 Input Validation（输入校验）已经不能满足系统的安全要求，因为绕过前端相对容易，所以就需要后端同样进行输入校验，这就只能依赖接口测试去验证了。

问答（35）收到现网问题，除解决外，还能做什么？

背景

现网问题，也就是在产品发布后，用户或客户的问题反馈。大多数公司由测试人员对接，经过筛查或复现，再反馈给客服或交由开发人员修复。有些即将承担或者已经承担该项工作的同学就问我，遇到这种问题，真正能解决的人最终还是开发人员，那测试人员在整个过程中除协助解决和验证外，还能做什么呢？

你问

收到现网问题，除解决外，还能做什么？

我答

遇到问题，第一时间去解决，的确没错，这是正确的响应行为。但我认为，既然问题已经发生了，解决问题只是一个理所应当的动作，但仅仅解决完，还不到松一口气的时候。真正重要的事情其实是在问题被解决之后才刚刚开始的，而这些恰恰是最容易被忽略的。

首先，我们要对现网问题进行分类统计，才能根据不同类别的问题进行后续的深入动作。我个人理解，主要包含两类问题。

1．缺陷

也就是 Bug，对这类问题需要做 3W 分析。其实在接到现网问题时就会分析一次，事后需要再分析一次，因为在处理问题时的分析也许相对个体化或特例化，事后可以集中一些同类别的现网 Bug 做更垂直深入和具有拓展性的分析。

What’s the problem?（问题是什么？）

对问题做出清晰且专业的描述，因为现网反馈的问题往往只是表象上的描述或者非专业的、破碎化的叙述，我们需要对它们进行“ 转译 ”，便于后面会说到的再利用。

What’s the root cause?（根本原因是什么？）

找出产生此类问题的根本原因。我所遇到过的有这样几类 ：

（1）测试用例没有覆盖到。

（2）测试策略没有考虑到。

（3）测试执行的遗漏。

（4）系统没有做相应的操作保护，用户操作导致的异常数据。

What’s the solution?（解决方案是什么？）

针对原因 1 和 2，要从测试用例的设计方法和测试计划环节着手去提高和改进。

针对原因 3，要从测试过程的监控和测试产物质量的验收环节着手去提高和改进。

针对原因 4，要从自身系统的易用性和健壮性着手去提高和改进。

2．操作问题

从这类问题中我们一般会汇总产出两样东西。

（1）常见问题解答（FAQ）。

这个常见问题解答除了可用于产品帮助文档的补充，或客服的问题参考文

件，还有一个比较重要的功能，就是通过整理这个常见问题解答的过程，可以分析出一些我们没有想到的用户行为习惯，这往往可以帮助我们找到一些潜在的系统问题。

（2）产品交互设计的改进。

这个产物可以帮助我们的产品交互设计师持续提高产品的用户体验满意度。

在现网问题的处理过程中，还有一个重要的、可持续更新的产物，那就是“ 现网问题信息确认清单 ”。这份清单主要包含某一类多发性问题，在用户反馈到客服处时，客服需要跟用户沟通收集哪些必要信息，一方面对问题可以做一次初步筛查，另一方面也为后续技术部门检查原因提供了更为完整的信息。

只有做完上述这些事情，我们才能松一口气。这也是我始终坚持的一个观点，那就是现网问题的处理过程一定要和产品研发过程（至少是产品质量检测环节）形成一个闭环，即从现网问题的处理流程中获取到的产物一定要作用于产品研发或质量检测流程，改进也好，补充也好，杜绝也罢，都不能落在空处，否则现网问题的处理就仅仅停留在解决问题的表层，而无法真正做到避免同类问题再次发生。

我们可以允许自己第一次犯错，但绝不能允许自己再犯同一个错误。

问答（36）测试计划很难制订吗?

背景

测试计划是不是很难制订啊?

我怎么可能提前把后面一个月的事情都计划好呢?

我连自己的个人事项都计划不好，又怎么能计划整个测试项目呢?

经常有人为了这些问题而焦虑。

你问

测试计划很难制订吗？

我答

测试计划是描述了要进行的测试活动的范围、方法、资源和进度的文档。它主要包括测试项、被测特性、测试任务、谁执行任务、各种可能的风险等。测试计划可以有效预防计划的风险，保障计划的顺利实施。

很多人在开始制订计划时不清楚为什么要制订计划，只知道这是流程的要求，于是拿到模板就开始逐条逐项地填充。

而在不知道为什么要写测试计划的前提下去写，很多东西其实就是为了写而写，写完之后，怎么看都会觉得可读性很差，更别说是否有很高的可行性了。

长此以往，很多人就会把制订测试计划看作一个填充模板的行为，测试计划制订就会变成一种固化的流程形式，失去了它本身应起的作用。

所以，我们在制订测试计划前，首先应该明确一件事情 :为什么要制订计划？

制订计划就是为了让项目管理者和项目团队对接下来的前进路线达成一致的认知。

我理解的项目计划制订其实也不需要模板化，更不需要多么“ 完美 ”，甚至可以把制订计划当成关卡游戏那样去玩。

（1）识别测试范围，如下图一样，画一条可以从起点（测试开始）抵达终点（测试结束）的路线。

（2）站在起点的地理位置和时间位置上，根据已知的信息，在能看得比较清晰的距离上标注一个里程碑（实心圆圈）。

（3）再假设自己已经站在第一个里程碑的地理位置和时间位置上，往前标注出下一个里程碑的位置。但这个时候，因为你所掌握的信息不足，很多东西都是基于假设的条件，没关系，标注一个大概的位置即可。以此类推，标注出

剩余的所有里程碑位置。

（4）识别风险，并把可能会遇到风险的位置标识出来（问号）。

（5）为第一段路线制订详细的计划，因为行走这段路线所需的信息比较充足和真实可靠。

（6）行走路上的每一天都可以根据实时掌握的信息动态调整前进的方向和步伐。

（7）当走到第一个里程碑的时候，看下一个里程碑会比站在起点的时候要清晰得多。根据掌握的最新信息和相关环境因素的变化，再适时地制订前往下一个里程碑的详细计划。

（8）以此类推，边走边调整计划，遇到风险就去应对，直到通关。

从上述“游戏通关”式的步骤来看，这种制订计划和执行计划的方式比较易于操作。从我的经验角度来看，这种方式可以解决几个传统项目计划中的弊端。

（1）需要花费大量的时间和精力去制订一套近乎“完美”的项目计划。

（2）项目经理在执行时常常会为了遵循既定的计划而牺牲一部分灵活应变的优势。

（3）可能会基于不够充分和不够实时的信息制订部分无用的计划，甚至错误的计划。当项目进行到某一个里程碑时，因为项目环境因素的变化而导致既定的后续计划需要大幅修改甚至推翻重来，造成前期工作的浪费。

问答（37）在软件测试项目里需要做风险管理吗？

背景

今天有同学问我，他原来在跟项目经理打交道时，经常听说风险管理，所以想了解一下，在内部研发的测试项目管理里，是不是也需要单独做风险管理？

你问

在软件测试项目里需要做风险管理吗？

我答

风险管理是指如何在项目或者企业这样一个肯定有风险的环境里把风险可能造成的不良影响降至最低的管理过程。风险管理的内容包括风险管理计划、风险识别、定性风险分析、定量风险分析、风险应对计划、风险监督与控制。

在执行过程中，项目风险管理可以简化为风险识别、风险度量、制定应对措施和风险监控 4 个过程。

项目的风险管理是一个动态的工作过程。

项目风险识别是项目风险管理的重要环节。

在任何环境下开发项目，风险都是无处不在的，也是防不胜防的。项目风险其实就是一种不确定的事件或条件，如果发生，就会对项目造成积极或消极的影响，如范围、进度、成本和质量。

所以，我们在制订测试计划时，最重要的一项工作就是识别风险和制定风险应对方案。

上面我们了解了风险和风险管理的概念，风险管理其实就是在识别出项目可能有的风险之后，事先就可以计划的一系列应对手段，最终目标是降低风险发生的概率，或者当风险发生时，如何快速、有效地应对和解决风险所带来的后果。

风险管理主要包括如下几个步骤 ：

（1）描述清楚风险是什么，会带来什么样的严重后果，包括发生概率有多大。

（2）分析风险产生的可能原因有哪些，可能发生在哪个阶段或时间点。

（3）针对每个原因再制定应对方案。

（4）每个风险应对方案要有明确的责任人和时间计划节点。

在实际项目当中，常常有人把风险产生的原因当作风险本身，问题虽然不大，但会导致计划里的风险计划清单可能比较长，给人一种“ 这个项目的风险也太大了吧，还能正常完成吗 ”的错觉。所以，今天就以软件测试项目中的常见风险为例，来制订一份风险管理计划。

1．风险

测试执行时间被压缩，产品未经充分测试就发布上线，导致上线后用户投诉增多（有 70% 的概率）。

2．原因

（1）产品需求给出得晚，而且也不完善、清晰，多次反复评审的时间挤占了研发时间。

（2）开发自测不充分，导致提测的版本质量不高，影响正常的测试计划进度。

（3）Bug 修复速度慢，阻碍了完整的新功能测试和最后的回归测试。

（4）中、后期频繁变更需求，或者新增需求，导致测试资源被分散。

3．应对方案

（1）优化测试用例的设计，减少冗余的和无效的测试用例，同时提高自动化脚本的覆盖率，节省相应的人力用于新功能的验证测试。

（2）制定测试包的可接收标准，编写 ATC（Acceptant Test Case，可接受的测试用例），没达到可提测标准的一律退回。采取这种倒逼的方式，促使开发人员提高自测质量。

（3）和开发负责人明确缺陷修复的要求，每天几点之前要日清当天新的缺陷。同时，测试任务包尽可能拆解得小一些、独立一些。在某些模块里出现阻碍性缺陷时，可以略过，先测试其他模块。

（4）在进行测试任务量评估时，预留 10% ～ 15% 的缓冲量，用于应对需求变更和临时新增需求。

在实际的项目中，风险管理其实也是需要渐进递推的。在项目计划阶段，对于很多风险只是预估和猜测，它们可能会发生，也可能不会发生。因为当时所处的时间位置离项目结束还很远，有效的项目信息也比较少，很难看清风险的虚实。

只有当项目开始后，随着时间和进度的推移，离风险发生的时间位置越来越近，同时项目信息也越来越丰满，才能更加清楚地看到相应风险是否会如期发生。到那时候，当初计划好的应对方案大多数需要做一些适时的优化和调整。

所以，我认为，风险是随着项目的推进而不断变化的，我们需要及时地识别和适时地出手应对。这是风险管理的难点所在，当然，也是乐趣所在。

问答（38）人人都在说的敏捷到底是什么？

背景

最近又有人开始讨论敏捷了，一些同学在找工作时，经常看到招聘要求上说“熟悉了解瀑布研发模型和敏捷开发”；也有的同学在进入公司，特别是互联网产品企业之后，经常听人说，××公司在用敏捷，我们用的也是敏捷，等等。所以，他们十分好奇到底什么是敏捷，因为这个概念在学校里或者培训班里是很少被提及的。

你问

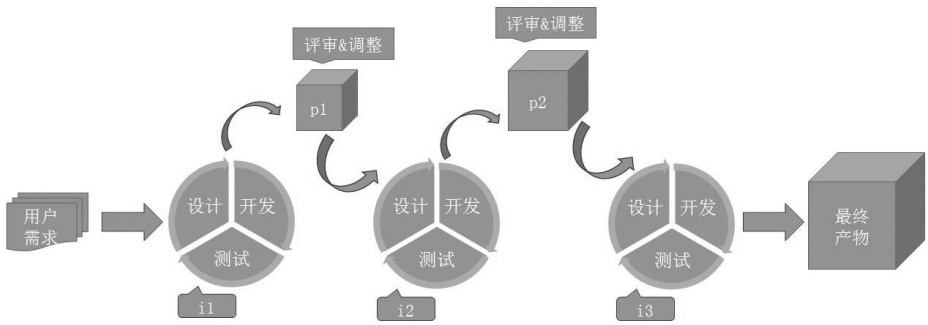
人人都在说的敏捷到底是什么？

我答

今天我们就先来简单地聊聊什么是敏捷。这个东西已经似火非火好多年了，特别是在进入移动互联时代之后，感觉一说到研发模式，要不是敏捷模式，都不好意思说出口。

我们先通过两张图来看一下传统的瀑布研发模式和敏捷研发模式的区别在哪里。

瀑布研发模式



敏捷研发模式

在采用**瀑布研发模式**时，用户直到产品发布才能看到是不是自己想要的，最终成品与需求的**偏差率相对较大**。

在采用**敏捷研发模式**时，用户能持续看到阶段性可用的产品，并能及时提出改进反馈，最终成品与需求的**偏差率相对较小**。

我们再来看看平时大家常说的敏捷软件开发、Scrum 或者 XP 到底是什么关系。

（1）敏捷软件开发是一种以用户的需求进化为核心、迭代、循序渐进的开发**方法**。

（2）Scrum 是一种迭代式增量软件开发**模型**，通常用于敏捷软件开发。

（3）XP，又叫极限编程，也是一种敏捷软件开发的**模型**，它以代码为核心，且由 4 部分组成 ：交流、简化、反馈、勇气，我们常说的结对编程就是其中一种模式。

在国内，我们听到最多的是 Scrum，那是因为经过很多项目和团队的实践，证明了它**有效、简单、持续交付的能力**，所以很多初创型企业或小规模团队都比较青睐它。

我们通过一段对话来了解一下 Scrum 模型中的角色投入程度。

鸡跟猪说 :“ 我们一起开一家餐馆吧。”

猪问鸡道 :“ 餐馆起什么名字呢？ ”

鸡跟猪说 :“ 就叫 ‘ 火腿与鸡蛋 ’ 吧。”

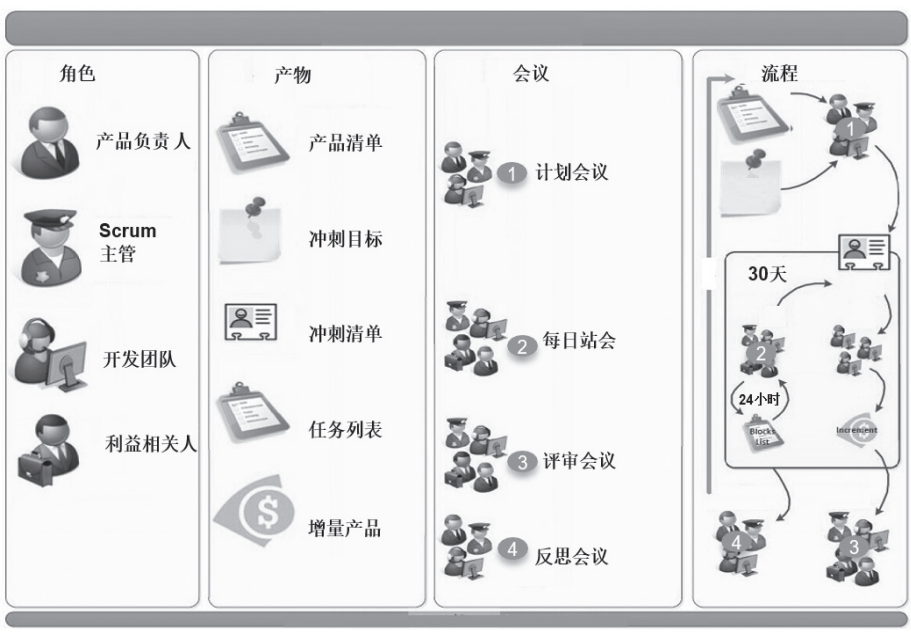
猪跟鸡说 :“ 谢谢！还是算了吧。我是全身心投入的，而你只是部分参与而已。”

从上面“ 猪 ”和“ 鸡 ”的对话中我们不难看出，在 Scrum 中有以下两种角色：

（1）内部角色（全身心投入的成员），如产品负责人、敏捷教练、研发团队。

（2）外部角色（部分投入的成员），如用户、管理层、其他利益相关者。

最后，我们通过一张图来快速了解一下 Scrum 中主要的角色、产物、会议和流程。



4 个角色：

Product Owner：PO，产品负责人，团队的掌舵人，引领团队朝着正确的方向前进。

Scrum Master：SM，敏捷教练（这是我个人最喜欢的，也是认为最贴切的中文翻译）。

Team：由开发和测试等相关人员组成的团队，每个迭代周期内所有任务的完成者。

利益相关人：在敏捷里，利益相关人指的是被项目过程和最终产物所影响到的角色，比如管理者、客户和用户。

5 个产物：

Product Backlog ：产品清单。

Sprint Goal ：迭代目标，也叫冲刺目标。

Sprint Backlog ：迭代清单，也叫冲刺清单。

Task List ：任务列表。

Increment Product ：增量产品。

4 个会议：

Planning Meeting ：计划会议。

Daily Scrum Meeting ：每日站会，传说中的 15 分钟，“ 站 ” 只是多种形式中效率最高的一种。

Review Meeting ：评审会议，也叫验收会议。

Retrospective Meeting ：反思会议。我认为，这是所有会议里最重要的一个，因为敏捷思想的核心就是持续改进，而持续改进来源于持续总结和反思。

流程：

（1）团队在产品负责人的主持下，通过计划会议、产品清单和冲刺目标产出冲刺清单。

（2）团队在敏捷教练的指导下，通过冲刺清单、任务列表和每日站会，在一个迭代周期（图中是 30 天）内产出增量产品。

（3）产品负责人和团队通过评审会议，验收每个迭代周期产出的增量产品。

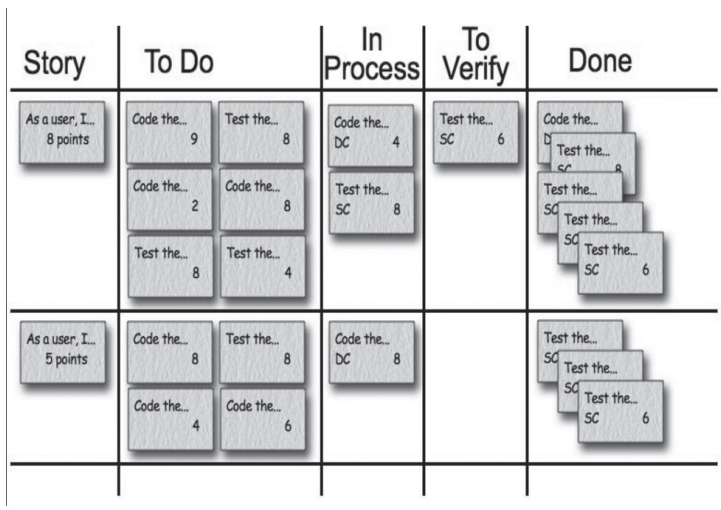
（4）敏捷教练和团队通过反思会议，反思每个迭代周期里做得好和做得不好的地方，持续总结和改进，以此来提高每个迭代周期的战斗力。

最后附上一些相关的概念。

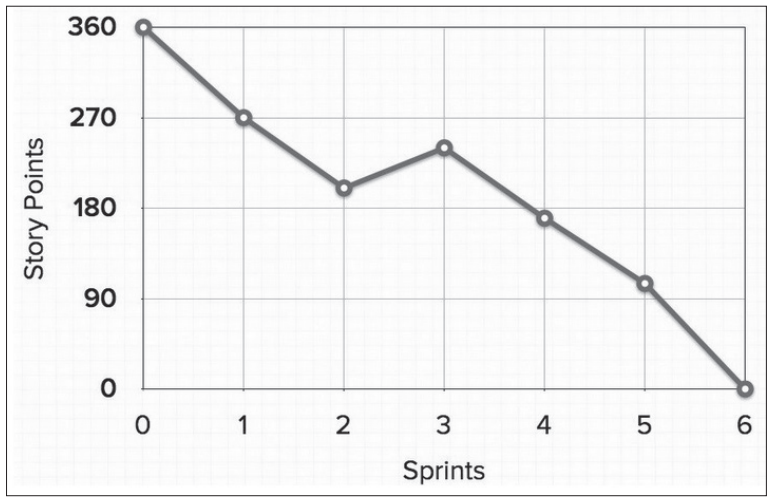
User Story：用户故事，从用户的角度来描述用户渴望得到的功能。

Task Board：任务墙，将 Scrum 过程中的各项事务放大并进行可视化展示的各种类型的载体。

Burn Down Chart：燃尽图，在迭代周期内用于跟踪任务进度的可视化图形。



任务墙



燃尽图

问答（39）为什么一定要引入敏捷呢?

背景

之前因为很多同学想知道什么是敏捷，所以我简单地说了说。又有同学问我，为什么一定要用敏捷，瀑布不能用吗？我觉得，在面对敏捷的时候，能对为什么要用它而产生疑问的同学很不错，至少他们不会盲目跟风、人云亦云。

你问

为什么一定要引入敏捷呢？

我答

我先从亲身经历说起。我之前所在的公司做的是企业级的网络会议系统，也就是一种 SaaS（Software as a Service）服务。从面向的客户群体来说，长期以来都没有对新功能的交付速度有很高的要求，一直以服务稳定为主。

但突然有一天，老大说，我们要准备转敏捷了。随后，在很短的时间内，敏捷培训、项目转型等蜂拥而至，当时的确有些措手不及。在相当长的一段时间里，很多人其实是有些不理解的，当然，也包括那时候的我在内。

不过，在跳出当时所处的环境之后，再回过头去看，其实也明白了一些东西，现在写出来，有着不同经验的人应该都能从中看到一些自己想了解的东西吧。

1．引入敏捷的初衷

早些年，我们的产品的确占有很大的份额优势，大概占整个欧美在线会议市场的 60%。后来几年，在原来一些专攻轻量级在线会议系统的公司中开始出现几家做得比较优秀的，也是因为很多中小型公司的在线会议需求逐渐涌现，

其中具有代表性的是 Ctrix 公司。它的整个团队也就四五十人，功能更新速度非常快，对于用户反馈的响应时间也非常短，所以我们的在线会议市场份额被不断地蚕食。

而我们当时的主版本从开始研发到全球上线部署，最快也需要六个月到一年的时间，等我们的若干个大功能上线了，别人新鲜热辣的功能早已上线半年多了。更别提移动互联时代的产品更新速度了，六个月到一年的时间，有时候，一款产品从生到死也不过这么长吧。

虽说我们以产品的质量和服务的稳定为重中之重，不过看看别人家的质量，也没有差到哪里去，服务也挺稳定的，虽然没达到 4 个“ 9 ”，但也有 3 个“ 9 ”。而且别人家的团队小而精，人力成本相对少得多，产出比就把我们甩开了好几个街区。

所以，市场决定了我们必须不断地改进现有功能和快速上线更新，这样才能在保持现有市场份额的同时继续提高市场占有率，而不是一味地吃着质量第一、稳定第一的老本，而选择性地忽略以速度制胜这一关键性因素。

2．引入敏捷的期望

（1）加快产品新功能的上线速度，能够跟上操作系统和浏览器越来越快的迭代速度，能够快速、及时地响应客户的反馈，解决客户问题，从而增加产品的竞争力，提高市场占有率。

（2）大大减少产品需求和最终实现之间的偏差率，降低后期返工所增加的成本和减少项目延期的风险。在此之前就出现过耗费了几百个人日（IT 任务工作量评估的一种单位）开发出来的新功能，产品经理说跟他预期的出入太大，必须在下一个大版本里推翻重来，甚至会一而再、再而三地推翻重来。

（3）提升团队自组织、自管理的能力和技术水平，让各个团队始终处于一种相对工作量饱和并且积极向上的工作状态。

（4）加强产品经理、工程师、开发、测试和运维团队、技术支持团队的合

作和沟通，提升从设计、开发、测试到上线的流程效率，缩短新功能上线的周期。

（5）通过敏捷提高各个团队的人力资源使用效率，减少因任务分配不均而导致的资源闲置状况。

（6）提升用户满意度。随着生活节奏的加速，用户对于问题从提出到解决所能等待的时间也在逐步缩短，但同时又不喜欢频繁地、无规律地收到升级通知，这对于版本发布的速度和节奏就有了更高的要求。

最后，我还想强调一点，敏捷研发模式的引入一定要基于现有流程存在的问题，不能为了敏捷而敏捷，要具体问题具体分析，如果只有引入敏捷才能解决，那再引入 ；否则，最佳办法还是就事论事，就问题解决问题。

问答（40）为什么敏捷不是万能的?

背景

上一篇最后强调了一点，敏捷研发模式的引入一定要基于现有流程存在的问题，不能为了敏捷而敏捷，要具体问题具体分析，如果只有引入敏捷才能解决，那再引入；否则，最佳办法还是就事论事，就问题解决问题。所以，这一篇再针对一些同学的疑问说一说为什么敏捷不是万能的。

你问

为什么敏捷不是万能的？

我答

在实施敏捷的过程中，我们遇到过一些问题，并找到了解决方案，但还有一些问题我们没有找到解决方案，今天就列举出来，从侧面说明一点：敏捷不是万能的。

问题 1：开发工程师对自动化测试的认知和理解比较粗浅，所以在做架构设计和代码实现时并没有考虑如何有效地支持自动化测试框架。

分析 1：高效的敏捷团队离不开自动化测试框架的支撑：如持续集成的自动构建环境、自动执行的版本校验脚本、用于版本回归的自动化脚本等，这些都是敏捷流程中不可或缺的一部分。如果在前期设计阶段没有将这些问题考虑进去，那么，因为代码结构的复杂度，在开发过程中很难对架构进行调整，所以只能在自动化方案的设计阶段考虑能否按模块去做一些相对简单的改动，比如针对一些非标准控件去增加控件编号属性，从而便于自动化脚本识别到它们。

问题 2：团队中的测试工程师，其黑盒测试经验丰富，但技术是短板，在短期内很难基于代码去进行单元测试或白盒测试。

分析 2：这是由于在产品开发初期对测试工程师的要求相对单一导致的，如今只能逐步引导部分优秀的黑盒测试工程师转型，或者补充新鲜的技术类测试工程师资源。但无论采取哪种方案，都不能立竿见影，学习、提升、熟悉、磨合等都是需要时间的。

问题 3：产品的功能模块之间耦合度和依赖性比较大，在实现一个规模较大的需求时，通常需要数据库、服务器端、前端页面和客户端同时支持，所以在用户故事的细化、任务的拆分以及不同的 Scrum 团队之间的协作上有很大难度。

分析 3：首先，不可能为了敏捷而去大幅度调整产品代码架构，只能在合适的时期逐次分批地进行调整。其次，在用户故事的细化和任务的拆分上尽量降低耦合性。如果数据库、服务器端、前端页面和客户端都是由不同的 Scrum 团队在承担任务，那就需要通过 Scrum of Scrum 的形式去管理项目，从而保证所有模块在每次迭代中的步调是一致的。

问题 4：虽然主线发布版本能做到在一个发布周期内的每次迭代都提交可发布版本，但运维团队的部署节奏跟不上。所以，产品研发团队即使在一个季度里提交了三个可发布的迭代版本，但运维团队只能发布最后一个版本。

分析 4：产品的复杂度决定了部署的步骤繁多，而且不同组件之间还有 Hard Dependency（强依赖关系）和 Feature Dependency（功能依赖关系）。同时，线上部署还没有做到完全自动化，上线成本较高，这也是在敏捷实施过程中一个暂时不可调和的矛盾。

问题 5：线上产品的小版本并存和定制版本太多，而且自动化测试覆盖率不高，导致在安全问题修复和操作系统 / 浏览器版本更新时，敏捷团队可能需要在一个迭代周期内同时在很多 Branch（代码分支）上进行代码合并和大量的手工回归测试，敏捷教练也很难保证团队在一个迭代周期内只聚焦在当前冲刺的计划任务上。

分析 5：提高持续集成测试和自动化测试覆盖率，同时要尽可能地说服客户尽快升级到最新版本，逐步减少多版本并存的现象。

PART 3
是走“管理”路线还是走“技术”路线

适合工作年限：5～7年

第6章　管理和技术，各有千秋

第 6 章　管理和技术，各有千秋

问答（41）让下属心悦诚服的领导长什么样?

背景

圈子里有朋友问，有些领导对下属要求很多，会让下属反感，作为管理者，是否可以做到在严谨细致的同时让下属心悦诚服？严谨细致是一种工作态度，是一种责任心的表现，它是一种可以让别人学习的东西，但并不意味着靠它就能让下属心悦诚服。

你问

让下属心悦诚服的领导长什么样？

我答

1．原则

（1）会因地制宜，帮助团队里的每个人找到适合自己的方向，跟每个人一起量身定制发展路线。

（2）能让每个人从进公司到再回到市场上的时候，在能力上都有实质性的提升，可选择的面更广，身价也有大幅度的增长。

2．方法

（1）愿意“教”。

不管你懂再多的东西、会再多的技能，如果不知道怎么教给别人，其实就意味着自己并没有真正吃透并掌握它们。

教人的过程是一个自我反哺的过程、一个剔除杂质的锤炼过程。

（2）善于“分享”。

分享会给自己带来很大的成就感，特别是当自己分享出去的东西被人所接受，看着自己的经验和心血被其他人采纳并应用时。

分享也会反作用于自己，激励自己不断地学习，不断地锤炼自己的知识体系。

（3）允许“练”。

掌握一项技能或方法的最有效的途径，没有之一，就是练。“**学以致用**”这 4 个字精准地阐述了这层意思。如果你想要真正地掌握一项技能或方法，则只要花 20% 的时间和精力快速学习一下相关的基础理论知识，剩余 80% 的时间和精力都要去应用。初期可以是边学边练，从模仿到跟练，慢慢地就能以战带练、实“弹”训练了。

管理者需要做到的就是允许犯错，不能因为怕出错就不让下属去学以致用。

（4）抓大放小。

规划期会站在公司大环境、部门目标和团队整体的规划角度，把握计划产物的商业价值和方向性。

不纠结过程细节，把注意力放在关键性的进度节点和最终的产物质量上。

3．角色

Scrum Master 常被翻译成“**敏捷教练**”，这是我最喜欢也是最认可的一种翻译。它很准确地定义了 Scrum Master 这个角色在敏捷团队中的作用：先教会大家如何玩转 Scrum 这种研发流程，再带领大家一边实践，一边持续改进。

作为团队的管理者，也应该具备教练的心态，带领队伍按照正确的方法训练，并让队伍在各种项目中作战，积累实战经验，并及时带领大家复盘，分析并找出改进点，然后加以改进。

问答（42）如何制订能有效落地的管理类学习计划？

背景

之前有些同学在向我咨询项目管理学习的时候说道，虽然目标有了，但在制订计划时却感觉无从下手，因为这种偏管理的学习计划不像执行类任务或工具类任务的计划，有着明确的完成指标或者清晰易衡量的标准，所以他们希望知道如何才能制订符合 SMART 原则的管理类学习计划。

你问

如何制订能有效落地的管理类学习计划？

我答

不管是团队管理还是项目管理，都有对应的方法论，肯定都是学习为先。不过，在职场的工作计划里，学习计划的占比通常很小。这并不是说学习不重要，而是因为多数公司都不会为你的学习和充电过程买单。

所以，我建议大家最好有一个提前量的学习，至少在你开始制订该类目标计划之前，对该领域的基础知识有一个框架上的认知。最简单的方法就是找一本该领域的经典入门书籍，快速地浏览一遍，不要求多，至少得像目录一样，知道这个领域大概有哪些内容。

学以致用才是目标计划里的大头，因为前面也说了，公司会为你买单的只有结果。所以，你的学习最终都得落到实处，比如，用在了什么地方、带来了多大的改进、产生了多少效益的提升等。

这类目标结果的检验标准通常写的都是我管理过多少项目、我管理的项目是否按时按质完成交付、有没有出现延期或成本超支等相对泛化的成果，较难

体现出你和别人的差异化，也很难体现出你学习的效果和成果。

所以，建议大家在罗列这种项目管理类的计划目标时，最好以产物为导向。你们肯定会问，这种管理类的计划能有什么产物，都是一些泛泛的东西。好吧，那我们接着往下看。

此类目标一般会有两类产物 ：一类就是在你的管理实践中形成的流程文档、作业文件、模板、工具等 ；另一类就是你在学以致用的过程中自己总结的知识或方法。

第一类产物，其实就是能体现你跟其他管理者差异性的重要项。大家可以试想一下，两名同样的项目管理者，完成了同样的项目数量，都按时按质完成了交付，数据也都正常。如果要在他们两人中间评优，这类产物是不是就可以成为取胜的关键？

两人完成了同等难度、同等量的工作，其中一人边做边总结，得出一些可以让其他人使用并保证项目成功的流程或工具，差异立现。

这里建议大家在描述此类产物的期望目标时，多用一些简单、直观、数据化的关键词，比如，提高 30% 的效率、减少 10 个人日的成本、节约 100 000 元的开支等。

第二类产物，也是现在越来越被看重的一个方面，就是你在整个过程中学到的很多知识和方法、积累的丰富经验。之前很多人认为这只是自己的学习成果，怎么能作为工作计划的成果呢？别忘了，你可以将自己的学习成果分享给其他人，在团队内部或团队外部都可以进行分享，这种分享包括课程和课件，都可以成为计划里锦上添花的产物。

上面说的是通用层面的，就是不管什么领域的管理，基本上都可以借鉴或套用。接下来我们就以项目管理为例来细说一下。

项目管理共分为五大过程域、十大知识领域。如果制订的是年度学习目标，那怎么去制订计划呢？

如果你总是想着，我怎么才能做出一份可执行的项目管理学习计划，那你就离实现目标又远了一步。因为项目管理从理论角度来说已经分为十大知识领域，如果落到具体公司的项目管理流程上，则又能再细分成不同的过程阶段，如需求评审、代码设计、测试执行和上线发布等。一想到这么多过程、这么多领域，是不是头都大了？是不是更觉得无从下手了？

这里我们应该先了解项目管理全过程大致应该分为几个阶段，每个阶段又对应哪些知识领域 ；再结合你所在公司的实际项目流程，把整个过程分割成 *N* 块，先挑出自己能力不足的，或者还不能游刃有余地管理的那几块，再从这几块中挑出今年从公司层面或者个人角度来看最重要的几块，个人建议不要超过 3 块。

就这最重要的几块，先逐个分析目前的不足、可提高的地方，再针对可提高的地方，列举一下大致的改进思路和想法。举个例子，假如你觉得在当前项目管理过程中，测试人员因为对需求的理解偏差或错误会导致一些问题在开发阶段就被发现，甚至在产品发布后，你想针对这一点去改进，那该怎么计划呢？

在需求设计阶段，需求文档是产物，那么，在进行需求评审时，测试负责人发现的问题数是可以记录下来的，就类似于缺陷 ；当需求评审阶段结束后，发现有些问题是在需求评审阶段遗漏的，也需要记录 ；最后看遗漏的需求 Bug 占需求 Bug 总数的比例，类似于需求发布后的缺陷率。

如果你试着用这种方法改进计划，最后提高了需求评审质量，那么这种方法是不是可以总结提炼成流程文档和作业文件，对于质量的提升是不是也可以量化？例如，采取了这种流程的项目，相比之前的项目，在需求评审阶段发现的问题数增加了多少，需求发布后出现的该类问题减少了多少，节约了多少个人日，以及后期沟通和补救成本，等等。

说了这么多，大家应该可以自己去试着制订一份可以落地的管理类学习计划了吧？

问答(43)为什么老板眼里的公司跟我眼里的公司不一样呢?

背景

有人问我，为什么他觉得有些事公司应该这么做才对，而且是显而易见的，老板却不这么认为呢？他一直以为员工眼里的公司和老板眼里的公司是一样的。

你问

为什么老板眼里的公司跟我眼里的公司不一样呢？

我答

员工眼里的公司和老板眼里的公司会是一样的吗？

有人说，肯定是一样的，大家都是为了公司更美好的明天在努力着。

我说，怎么可能一样呢？

不同的角色，不同的身份，所处的位置不同，看待事物的角度不同，关注的核心点也不同。

老板关注的是公司的发展战略、盈利状况，或者说公司就是老板的理想、梦想、奋斗的目标。

员工关注的是自己的职业发展、薪资福利，或者说公司就是员工的土壤、大树、生存的工具。

很多老板在年初或年底的会议上喜欢大谈特谈自己的理想，叙述自己奋斗的艰辛，期望大家能够和他同甘共苦、共渡难关。这属于感性的老板，只需要

配合一下，专心地倾听。但有时候它也像一碗鸡汤，喝一点，也能获得一定的能量。

这样的老板其实是想获得员工的理解，引发员工的共鸣，从而起到激励作用。通常效果怎么样呢？不能说很差，也不能说很好，保守估计，得在他的期望值上打个对折吧（我是感性偏理想化的）。

我认为老板的那些理想、艰辛对于员工来说太遥远了，或者说跟员工有什么关系呢？员工之所以加入公司，是因为公司提供了适合他们的岗位，提供了令他们满意的薪酬。而那些所谓的理想、艰辛、难关都是老板的，他们并不关心，也可以说他们并不关心这一层面的东西。

我觉得马云总结的那句话十分到位 ：员工离开公司只有两个原因，要么是钱没给到位，要么是做得不开心。我个人认为还有一个很重要的原因，就是没有发展空间和学习的土壤。

只要这几个因素都没问题，老板根本不用谈理想、诉艰辛，员工就能跟你齐心协力，把事情做得很好，什么难关扛不过去？

如果工作环境人性化，做得开心，有让自己发挥的空间，能学到新东西，到手的薪资能保证后顾无忧，那么，就算忙点、累点，员工也不会怨声载道，沟通执行的效率绝对会很高。

如果不从这几个因素出发去做员工的管理和规划，只是一味地喊口号，或者出台各种政策，则只会适得其反，跟老板的期望背道而驰。

所以，优秀的公司管理者或者管理团队应该站在员工的角度，从员工最关注的东西出发，考虑、规划、制定各种政策，去发掘怎么做才能最大化地激发员工的潜能和积极性，而不应该站在老板或者合伙人的角度，通过约束、奖惩、强制等手段来让员工被动地接受任务。

我心目中理想的团队是这样的 ：

（1）没有 KPI。

这是第一个要摒弃的东西，不仅劳民伤财，还总是适得其反。其实只需要一个 360 ° 互评，团队里谁做得好、谁做得不好，一目了然，客观公正。

（2）有奖，没有罚。

每个人的目标产物和成长速度直接与奖金、加薪、股权挂钩，目标没完成或做得不好，不加以惩罚，但也没有这三样奖励。我想没有人会因为没有惩罚就不反思、不努力，不想要这三项奖励。

（3）目标自助。

每个人都围绕团队目标选择自己的短期或中长期目标，不限制你的目标方向，但团队成员肯定都会围绕团队目标来进行计划和选择。

问答（44）我应该成为什么样的领导才会受欢迎？

背景

这大概是一位同学在跟现在的领导打交道时，因为什么事而引发了他对于自己以后如果做了领导，该成为什么类型的领导才会受欢迎的思考，于是就有了这个问题。

你问

我应该成为什么样的领导才会受欢迎？

我答

在职场中，每个人都会遇到形形色色、各式各样的领导，让我们先来归归类吧。

命令式 ：任务下达之后，只做必要的解释，只关注最终的结果。

保姆式 ：任务下达之后，先逐条地解释，然后手把手地教。

教练式 ：任务下达之后，不做非必要的解释，过程中会针对方法或结果做相应指导，并持续关注进度。

萝卜青菜，各有所爱。所以，从主观情感上也难以评判这三类领导孰好孰坏，我们再客观地从以下几个维度来比较一下。

1. 任务的下达率 ：命令式 ⩾ 教练式 ⩾ 保姆式

这比较好理解，令行禁止，就像部队作战一样，所以说命令式在这种场景下效率较高。

2．任务的完成效率

（1）新任务 ：保姆式 > 教练式 > 命令式。

不管是教练式还是命令式，任务承接人都需要自己了解任务，找寻可行的方法，然后才能开始执行，所以时间相对较长，效率相对保姆式肯定要低。

而保姆式，因为全程手把手地教，教完了，其实也就相当于做完了，所以完成的效率相对较高。

（2）老任务 ：教练式 > 命令式 > 保姆式。

如果是教练式的，那么任务承接人已经掌握了完成这类任务的方法，所以很快就能上手，完成效率当然不会低。

这时候如果还是保姆式的指导，则反而会影响完成的效率。因为任务承接人原本已经会做了，你却还要手把手地教，就会适得其反。

3．员工自身能力的提高和成长 ：教练式 ⩾ 命令式 ⩾ 保姆式

其实差异很明显。

教练式，顾名思义，边教边练，正所谓 “ 授人以鱼，不如授人以渔 ”。

命令式也很锻炼人，因为需要任务承接人独立完成任务，不过十分考验人的心志。

保姆式最不可取，因为会让任务承接人产生依赖性，从而丧失成长和锻炼的机会。

我喜欢命令式的领导，但我自己喜欢做教练式的领导。

我在承担执行任务的时候，比较喜欢按照自己的计划和节奏进行，而且比较喜欢去完成那种有挑战性的任务，简单来说就是自己没有接过的任务，需要重新研究方法或方案。当完成这类任务时，我会比较有成就感 ；同时，在这种

任务的执行过程中，学到的东西也相对较多。

而那种重复性比较高的、有历史解决方案或经验可借鉴的任务，不是说学不到东西，而是可以用于打磨已掌握的技能和技巧，不断地锤炼，持续地改进。只是这种任务让人比较容易产生疲劳感，容易陷入一种自我打造的舒适区。

我自己在管理团队或者项目时，其实是属于教练式的，具体的策略请参见问答（41）。

问答（45）管理体系审核为何还有内、外审之分？

背景

有些同学问我，自己公司也在做或者通过了 ISO 9001 质量管理体系认证，在这个过程中，经常听到外审和内审这两个词，自己身边还有同事兼职做内审员，所以想了解这种管理体系审核为什么还有内、外审之分。

你问

管理体系审核为何还有内、外审之分？

我答

ISO 内审员是指经过 ISO 标准要求培训，并考核合格，取得了内部质量管理体系审核资格的人员。他们只负责企业的内部审核，从事这个岗位的人员需要熟悉企业运转流程及管理职责，负责对内部管理流程进行评审、审查、监督，以及提出整改方案。

外审员，也称国家注册审核员，指的是从事 ISO 9001 质量管理体系、ISO 14001 环境管理体系、OHSMS 18001 职业健康安全管理体系、ISO 50001 能源管理体系等认证审核活动的人员。

他们通常是以认证公司的名义从事第三方审核的人员，可以直接到各家公司里进行审核。外审员实行国家注册制，需要先参加外审员培训合格并取得外审员培训合格证，才能参加国家每个季度末的统考，考试通过才能申请注册成为审核员。

看完上述两种角色或者岗位的介绍，我再结合自己参与过的体系认证活动，说说为什么要有内、外审之分。

从实际的体系认证角度来说，企业通常有两种做法 ：一是务虚 ；二是务实。

务虚，顾名思义，就是通过咨询公司入场咨询和协助，顺利且成功地通过相应的体系认证审核，拿到相应的证书，这可用于一些招投标项目或作为公司的某种资质使用。

这里的审核指的就是具有合法和合规审核资格的外部审核专家进场对企业所做的审核活动，也就是通常所说的外审。

至于为了拿到证书而准备的一系列体系文件，基本上是不需要真正实施落地的，最多就是在每年复审或抽审的时候，再由相应的咨询公司协助更新或整理，以应对复审。

务实，就是不仅仅为了拿证，也为了将相应的质量管理体系实施并落地。这时候，当咨询公司入场进行咨询、体系流程梳理和实施改进的时候，企业内部除有专门的质量管理体系项目负责人和他们对接之外，还需要企业提供内审员全程协助整套体系流程的梳理和实施。

大多数咨询公司一般都具备内审员的培训、考试和发证资质。在咨询公司的协助下，顺利通过体系认证之后，内审员就需要配合质量管理体系相关人员，每年在企业内部做 1 ～ 2 次内部审核，审核的方法和流程基本等同于外审活动。

内审的作用有两个 ：

第一，确保该质量管理体系在企业内部真正落地且持续改进，始终有合理合规的项目及完备的相关流程体系文档。

第二，确保企业每年都能顺利地通过可能来临的外审。

所以，管理体系审核的内、外审之分也可以理解成学生在整个学习过程中需要不断地做模拟试卷（内审），最终目的是通过期末考试（外审）。

问答（46）性能测试是不是很难做？

背景

有些同学因为工作需要，看了我之前的学习笔记，然后跑来问我，说领导让他负责产品的性能测试，但他买了几本书，也安装了相关的工具，看来看去觉得太复杂了，无从下手。

你问

性能测试是不是很难做？

我答

我从零距离接触性能测试到今天，也才一年多的时间，在这上面走过的路崎岖蜿蜒，个中滋味只可意会，不可言传。虽然我已经从入门到“放弃”了，但是我一直在思考和寻找，怎样才能让性能测试不再看上去那么难，不再那么看着“高不可攀”。

性能测试其实就是测试的一种类别，那么相应地，它也是有一套标准流程的，无外乎就是需求分析、测试计划制订、测试执行、结果分析等几个环节。

所以，针对性能测试流程里的几个环节，我把自己换位到当初的小白，去思考自己当时最希望得到什么样的支持和帮助，再结合产品化的思维，思考出下面这样一个性能测试框架或指导性体系。

1. 是什么

学习性能测试里的常用基本概念、测试方法、标准流程的定义和解释。

2．做什么

性能测试需求的分析方法，可以采用检查清单的问题形式来帮助使用者得出对应需求所需要采用的性能测试种类，是压力测试、稳定性测试，还是健壮性测试等。

3．怎么做

（1）对应上述第 2 步得出的具体测试种类，每一种都有相应的测试方法说明，包括需要准备什么样的数据、步骤和如何选取相应的脚本进行修改或组装。

（2）有一套对应的样例库，包含脚本（.usr）、参数化文件（.dat）、场景（.lrs）。虽说不可能百分之百通用或者套用，但至少在同类产品的性能测试中都能套用，它们都是相对独立、结构清晰的一个一个的数据包，便于更新和管理。

4．怎么样

性能测试完成后，系统都会生成一份报告。针对常用的单分析图和组合分析图，有样图与实际图形进行对比，并告知这些数据图分别代表性能的哪些指标，这些指标的值又分别代表性能是好还是坏。

5．怎么办

对于常见的性能问题，罗列出通用的解决方案，比如，是应该检查并优化 SQL，还是应该修改服务器端 Tomcat 的连接数大小，等等。

如果能有这样一套产品化的性能测试框架，那么性能测试这座大山对于大多数测试工程师来说也就不那么“高不可攀”了吧：具有指导性的作业文件、测试计划模板、独立的测试数据和测试脚本、分布式测试环境搭建脚本或手册、测试报告和相应的分析模板，能支撑一套完整的性能测试框架迅速落地，快速适应不同的项目，并且能让测试工程师以最小的学习代价完成性能测试任务。

不过，这样一套框架不是一朝一夕就能建立起来的，必须在性能测试工程

师对理论有了深入的理解，并通过多个项目的实战，从中总结、归纳而形成一套方法论体系，再辅以相对独立的数据和脚本、计划模板、分析步骤和模板等相关工具，不断地打磨、优化和改进后，才能形成一套不论是入门级的小白还是进行中的老鸟都可以利用它轻松登上高峰的产品。

PART 4
成为资深软件测试专家

适合工作年限：7 年以上

第 7 章　培训师、咨询师和教练

第 7 章　培训师、咨询师和教练

问答（47）学以致用之后还可以做什么？

背景

我的系列问答基本上遵循“学以致用”的原则，平日在回答一些同学提问的时候也都在宣传这种思想，所以也有同学会多问一些，那就是：我做到学以致用之后，还可以做什么呢？

你问

学以致用之后还可以做什么？

我答

学以致用的最终目的是把理论知识和实际应用衔接起来，由浅入深、熟能生巧。

2016 年，我对自己和身边的朋友都提出了“学以致用”的目标要求，不管你是学自动化测试、项目管理、性能测试也好，还是学 SQL 语法也罢，也许各有各的出发点，于公于私都没有问题，但我只提出了一项要求：学以致用。无论你学的是什么，你都得在实际的工作当中利用起来，这是一个很实际的制定目标或规划的方法，再加上“能有带来利益的产物”的要求，就变成了很现实的方法。

而在 2017 年伊始，我给自己定下了回归理论学习的目标，直到今天，我

都在做一件事，就是把自己多年积累的有关时间管理、软件测试、敏捷方法的知识与经验进行梳理和记录，并输出为一篇一篇的文章或读书笔记。这个过程应该快结束了，我很快就能清空自己了，接下来就要开始重新输入的过程了。

我为什么会在自认为做得还不错的 2016 年结束之时做出这个决定呢？虽然 2016 年的“ 学以致用 ”做得很好，对我之前从未涉足的两个领域有了较多的认知和理解，但我一直觉得还欠缺了什么，总是感觉这些都不是我最终想要的东西，而且这种感觉早已存在。

先说说我决定回归理论学习的初衷吧。有不少人会给自己或别人贴上这样两个标签 ：学院派和实战派。在进入软件测试这个领域的头几年，我也很坚定地站在所谓的“ 实战派 ”阵营里，对所谓“ 学院派 ”阵营里的很多理论概念都嗤之以鼻。

举个简单的例子。那时候的我喜欢对学院派的人说 ：“ 我没有你懂的理论方法多，我不知道什么是等价类方法，也不清楚什么样的算场景法，更分不清因果图和判定表设计法，但是，我每年都能成为 Bug King（发现 Bug 最多的是人），而你可以吗？ ”那人说 ：“ 既然你每年都能成为 Bug King，那你跟大家分享一下你的方法和经验吧。”我就只好闭嘴了。为什么呢？因为我根本就没什么可分享的，我只有实战经验。也就是说，虽然我做得好，但说不好，更别提分享了。

我相信很多人都会有这样的问题，就是你在某一方面做得很好，你是一个技术牛人或者管理高手，可是在让你分享成为牛人或高手的方法，或者指导别人怎么成为牛人或高手时，你多半会像我一样，磕磕巴巴地说不出几句就江郎才尽了，对不对？

古人云 ：只可意会，不可言传。这句话用在这里虽然并不是很恰当，但我觉得很形象。

说了这么多，我想不少人应该已经明白了，我一直觉得缺少的东西和一直在找寻的东西就是方法论。

方法论是什么？我们先来看看百度百科中的解释 ：

方法论是一种以解决问题为目标的体系或系统，通常涉及对问题阶段、任务、工具、方法技巧的论述。方法论会对一系列具体的方法进行分析研究、系统总结并最终提出较为一般性的原则。

我再通过一个例子来阐述一下我的理解。

从 A 地到 B 地，我们通过百度地图能够查询到 3 条路线，于是我们会按照这 3 条路线从 A 地到 B 地。而我在从 A 地到 B 地实地往返了多次之后，发现了第 4 条路线，比这 3 条路线都要近。然后，你让我带着你再按第 4 条路线从 A 地到 B 地走一次，我却找不到路了。

在只有实践经验时，我只能凭经验让自己更快地从 A 地到 B 地，但是无法保证每次都能成功，原因就在于没有记录、分析沿途的标志物和判断方向的方法，没有总结出一条可以保证每个人每次都能从 A 地到 B 地的路线。

这个例子中的“ 路线 ”其实就是我所理解的方法论，也就是我在学以致用之后回归理论学习想要找寻的东西。

问答（48）如何做软件测试咨询?

诉求都是双向的，即使产生诉求的时候是单向的，但最终都会形成供需双方。

现在随着用户体验越来越重要，很多中小型软件企业对产品质量的重视程度也越来越高，但是因为前期组建测试团队和建立研发流程时的问题比较多，而且很多问题早已被固化或者形成了定势思维，从而导致由内而外的自愈可能性变得很小或者很难。

所以，引入外部软件测试咨询的诉求也就产生了。相应地，想利用自身的软件测试知识和经验为他人做咨询和解决方案的供方其实也早已慢慢滋生了。

今天我们就一起来看看，为他人做一套软件测试的咨询大致是一个怎样的流程。

1．项目前期

（1）职责。

① 明确客户的需求。

② 了解客户的产品架构。

③ 了解客户的产品在测试中都有哪些问题。

④ 制订客户的产品测试执行计划书。

（2）流程。

① 客户提供与产品相关的文件资料，主要包括如下内容。

必选 ：产品核心架构图、产品功能流程图、产品涉及的主要技术和功能需求文档。

可选 ：已存在的测试流程和方法、当前产品的质量缺陷分析报告、期望测

试所要达到的成熟度。

② 回顾客户提供的相关信息，如有需要，则进行一段时间的产品试用体验，目的在于 ：

· 了解产品功能，划分产品模块，评估产品规模。

· 确定产品的当前测试水平，评估产品质量等级。

· 根据产品规模和质量等级，初步评估测试成本。

③ 与客户进行第一次面对面沟通，主要解决以下问题 ：

· 对客户所提供的信息中不清楚和不正确的地方做一次确认。

· 清楚地向客户陈述产品架构及功能，以确保对产品没有理解错误或偏差。

· 客户如有疑问或具体要求，那么双方需商讨并达成共识。

④ 制订产品测试执行计划书。

基本内容包括如下几项 ：

· 测试执行流程，每个测试阶段的重点及达标标准。

· 划分自动化测试与手工测试的覆盖范畴，并选定自动化测试工具。

· 测试用例结构列表，包括核心功能点、全用户场景测试点、异常场景等。

· 测试报告及质量评估。

附加内容（并不包含在本次咨询范围内，但可作为后续扩展计划）包括如下几项 ：

· 是否提供详细的测试用例设计方法。

·是否需要提供自动化测试的需求分析和测试框架设计及部署，以及相关培训。

·是否需要提供性能测试的需求分析和测试框架设计及部署，以及相关培训。

·是否需要提供安全测试的需求分析和测试框架设计及部署，以及相关培训。

⑤ 与客户回顾产品测试执行计划书。主要目的如下 ：

· 详细解释产品测试执行计划书中的各项内容，取得客户的认可和确认。

· 指导客户根据自己的需求选择最合适的测试点项目。

· 确定最终的产品测试执行计划书及测试执行负责人。

（3）成果。

① 产品质量评估报告。

② 产品测试执行计划书。

2．项目中期

（1）职责。

① 指导测试执行负责人按产品测试执行计划书完成整个测试过程，通过过程监督，确保测试计划的顺利执行并达到最终标准。

② 对执行过程中出现的异常情况及客户提出的计划范围内的疑难问题提供咨询。

③ 对客户在测试执行过程中提出的新需求制定补充方案。

④ 持续优化测试方案，并实施改进。

（2）流程。

① 该阶段严格按产品测试执行计划书执行，实施过程监控。

② 在每个计划里程碑到达的时候，验收阶段性成果，并完成验收报告。

③ 在测试结束后，进行产品和过程质量评估。

（3）成果。

① 测试执行监督日志 ：问题答复记录，达标测试报告。

② 最终质量评估报告。

3．项目后期

（1）职责。

① 总结和分析本次测试计划的执行过程有哪些良好的实践和存在的问题，并进行改进。

② 预测潜在的质量风险，制定 Known issue（已知问题）列表。

③ 收集客户的反馈意见，完善测试咨询流程本身。

（2）流程。

① 根据测试过程中的缺陷数据，制作测试计划执行的过程分析报告。

② 与客户面对面沟通 ：总结测试执行过程，分析产品潜在质量风险，收集客户反馈信息。

（3）成果。

① 测试计划执行过程的分析报告。

② 产品潜在质量风险分析报告。

③ 客户意见反馈表。

问答（49）什么是高性价比的敏捷落地方案?

背景

我跟一些同学聊了不少关于敏捷实施的问题，有些人也会问有没有可以照搬套用的敏捷实施方案。其实是没有的，因为不同的公司有着不同的项目事业环境因素和企业文化、项目特点、人员组成结构等，再加上敏捷本身就是一种思想，实际落地后都是各式各样的模式，所以不大可能有什么拿来就能套用的流程。

你问

有高性价比的敏捷落地方案吗？

我答

Scrum 什么时候才能成熟到广泛应用⁇

这是我在 2010 年左右开始接触 Scrum 时问自己的一个问题，我当时的判断是 ：

或许这个时间点始终不会到来，也许 Scrum 也不会成为主流的研发模型。因为企业的行业领域、价值观、现状和工作环境因素的本质差异决定了不会有一个像 Scrum 这样被 “ 精确定义 ” 的方法能普遍适用。相反，极有可能出现类似过去 XP 的情况 ：Scrum 及其他方法中的各种实践会被打散，不是简单地重新编排一下顺序，而是被企业零散地选择性采纳，最终形成 “ 四不像 ” 的研发过程。

在这个过程中，熟悉企业现状、能够深入理解及灵活应用方法论的高级管理人才将发挥重要的作用。

当初公司引入 Scrum 之后，我认为最大的变化并不是每个版本的发布速度提升了多少、发布的东西多了多少，而是**团队整体对质量的信心和工作效率**提升了很多，产品、开发和测试人员的**合作更紧密**了，**有效沟通时间**比之前翻了好几番。

基于这个最大的变化，下面所遇到的一些问题就都不是不可克服的问题了。

1. 我们很迷惘，也很害怕改变

（1）原有的项目开发周期也是一个月，其实版本迭代已经很频繁了，为什么还要引入敏捷呢？

（2）虽然开发过程比较快了，但原有瀑布流程里的沟通不多，整个团队还是按照职能一个里程碑一个里程碑地线性工作。应用敏捷之后，大家觉得沟通多了，对质量有了更大的信心。

2. Scrum 和现有流程该怎么并行和融合

（1）产品需求文档、功能规格说明书和测试计划等相关文档还需要保留，主要是为了保持产品的延续性，但这会增加团队成员的工作量，需要在冲刺计划里把工作量计算进去。同时，Scrum 是从一个不完善的或者不确定的需求开始的，这种持续改进的特性要求这些文档也应该是协同持续更新的。

（2）一些原有的里程碑因为管理层的需要而被保留下来，这其实冲突不大，只要在前期计划和过程监控中进行有效的控制，就不会产生太大的影响，再将整个冲刺期里的某些时间节点对应到原有的里程碑上即可。

（3）让大家看到工具的使用会有助于他们在冲刺中很容易地掌握自己的进度、计划自己的时间，执行完成后可以很容易地统计自己的工作量、检查自己的效率和能力提升。

（4）在工具使用的学习阶段，可以把这部分工作量算在计划当中，因为在敏捷团队的容量计算中是可以包含学习量的。

3. 团队成员能力参差不齐怎么办

（1）自主学习，由近到远地按规划做技术储备。

（2）结对开发，以老带新，思维互补，查漏补缺。

（3）代码审查，黑白结合。

4. 管理层从人力资源利用效率角度要求 Scrum 团队不能固定怎么办

（1）铁打的营盘流水的兵和稳定的佣兵团，孰强孰弱？

（2）团队是需要时间磨合的，一个默契度、熟悉度和相互认可度都很高的团队有着非常高的工作效率和很强的战斗力，而且随着时间的推移，这种战斗力会呈几何级增长。

5. 项目范围总是不可避免地发生变化怎么应对

（1）在制订冲刺计划的时候，预留 20% 的缓冲量，用于应对项目进行中的需求变化。

（2）每个用户故事都需要有相对准确的故事点评估和清晰的优先级，在制订计划时，不要只选同一优先级的用户故事，而要选择不同优先级的用户故事。这样一来，当发生范围变更风险的时候，就可以用低优先级的用户故事去冲抵。

在上面这些问题的解决方案里，不管是预留的学习时间、评估里的缓冲量，还是佣兵团的培养，都是不容忽视的潜在成本，甚至有的就是沉没成本。所以，对于一家公司来说，不论是初创的还是成熟的，都不能只从单方面去研究敏捷是否适用于它，而应该从项目类型、人员结构、成本效益及管理层的决心等多方面入手，这就不是简简单单地找一位老师来给大家集体培训一下什么是敏捷，或者找一家咨询公司入场咨询一两个月就能着手开始的事情了。

时至今日，我依然坚持我的观念 ：是否需要引入敏捷研发应该由问题驱动（Problem Driven）。

（1）要知道为什么当前的研发模式不能满足现有需求，找出真正的原因。如果组织或团队中不存在需要改进的问题，就没有必要非得引入敏捷。

（2）针对已知问题，一个一个地找到解决方案，再结合敏捷的思想、具体模型、方法和工具，对现有流程做出性价比较高的补充或改进。

（3）管理层要对自己的公司 / 企业 / 组织 / 团队的现状和存在的问题有一个清醒的认识。

（4）下面一些问题可以用于自测，仅供参考 ：

我们企业现在存在哪些问题？

Scrum 是否可以解决这些问题？

Scrum 是否与我们企业的管理制度和企业文化相匹配？

如果不匹配，则需要我们做出多大程度的改变，是否在我们的可承受范围之内？

如果 Scrum 不是适合我们企业的最佳实践，那么是否可以采用混合的敏捷实践？单个方法不行，复合方法的行不行？西式的做法不行，中式的行不行？

问答（50）如何让敏捷软着陆?

背景

这一篇来和大家聊聊怎么才能让敏捷软着陆。它和其他规范或者工具引入不太一样，因为它包含了一整套研发体系流程，而且现在要引入敏捷的公司大多是想从瀑布模型转型而来的，这可是一套根深蒂固的传统流程，如果选择硬着陆，则很容易折翼。

你问

怎么才能让敏捷软着陆？

我答

我也不敢说软着陆的成功率就是 100%，但这至少是我在实际的敏捷实施过程中通过不断总结和反思得出的一些经验，有接地气的，也有偏个人理想化的。

1．问题驱动

我之前说过，敏捷并不是万能的，所以，不要只听到其他公司说用了敏捷之后，效率提高了百分之多少、成本降低了百分之几等，就去盲目地追赶这股“敏捷风”。

还是应该由问题驱动，先收集当前的研发模式有哪些问题，再分析一下，到底是人的问题、技术的问题，还是流程的问题。如果真的是流程问题，就再看看，是需要全盘替换，还是仅仅吸取敏捷里的若干特色工具或方法即可。

这里举一个真实的案例。曾经有一个项目组想全面实施 Scrum 研发流程，所以找到我去给他们做了一次分享。分享结束后，我跟项目组的人沟通了一下

他们想采用敏捷的初衷和当前到底有哪些问题，最后发现根本原因是项目组的规模比原来扩大了，从而导致项目经理在对项目进度的把控和组内沟通上出现了问题。所以，我提出了两套方案。

方案 A：引入 Scrum 的全套流程，前期可能需要消耗不少时间在学习和磨合上，所以同期的产出量肯定会下降。

方案 B：针对目前的问题，只引入 Scrum 里的“任务墙”和“每日站会”，让整个项目的进度变得直观透明，同时让沟通更及时快捷，其他流程保持不动。这种方案对于项目组成员来说，学习成本最低，基本不会影响产出量。

项目经理最终选择了方案 B，而且在实施了一段时间后，的确解决了他们的实际问题。所以，**敏捷的引入一定要切合实际，而不要为了敏捷而敏捷**。

2. 自上而下的推动

敏捷的引入及推行一定是自上而下的。什么意思呢？也就是说，公司高层首先要从思想上认可敏捷，同时要给予最大限度的支持。这样一来，敏捷在落地的过程中会减少很多障碍。千万不能自下而上地推行敏捷，那样基本上很难成功。

举个例子，我们的敏捷教练和团队开了一整天的计划会议，制订了团队都认可的冲刺计划，当大家正准备按照这个计划开工的时候，职能经理跑过来质问团队：代码哪天完成提交测试？代码哪天冻结？哪天提交运维发布？在你们的计划里怎么都看不到这些里程碑了呢？

敏捷教练跟他说，现在都拆分到用户故事颗粒度了，每个冲刺结束的产物都可以发布上线。结果职能经理来了一句：我不管什么颗粒度，也不管什么冲刺，我只想看到那几个主要的里程碑，最终上线不会延期，就可以了。

结果，敏捷教练又熬了一晚，从冲刺计划里提炼出几个主要的里程碑，按线性模式列出并提交给了职能经理。最后，大家在制订计划时，又回归传统的模式了。

3．专职的敏捷教练

很多公司在实施敏捷的时候，对于敏捷教练这个角色总是不够重视，特别是那种内部研发类的项目，因为之前没有专门的项目经理角色，所以在转为敏捷时，也经常会让开发组长或测试组长兼任项目经理。但从实践角度出发，我一直强烈建议这个角色要专人专职，特别是在初期。

在前期整个团队开始采用敏捷时，敏捷教练起到了很重要的作用，他需要花很多精力和时间在学习、指导、总结、分析和改进等工作上，还要负责相关会议的组织、项目数据的统计、项目问题的跟踪、进度的汇报等常规工作。如果不是由一个专职的人去做，则势必会影响到他的本职工作，也会消耗他在教练这个角色上的很多精力，最终导致两个角色都做得不好。

另外，如果由某个角色的人去兼任，那么，他不管是在做教练还是在做保镖时，都会下意识地站在自己的本职角色去考虑和看待问题，在做一些决策和判断时就会带有偏向性。

4．核心成员，以老带新

敏捷团队强调的是自组织、自管理，对每个人的要求都相对较高，但从实际角度出发，不可能每个敏捷团队的人员配比都是高阶人力，所以只能建议在初期组建时，至少保证每个角色都配备一名高阶人力，再按实际需求配备中、低阶人力，以老带新，通过若干次迭代的打磨和成长，让团队成员都能达到自组织和自管理的“理想”状态。

5．从“‘循规蹈矩’”到“‘因地制宜’”

很多 Scrum 团队在刚开始的时候就对本身标准的流程、方法和工具进行优化和改进。我一直在问他们：你们所谓的“优化”和“改进”是基于什么的呢？我听到得最多的回答是：“我们原来的流程就是这么玩的呀！”那你们就按原来的玩法玩呗，为什么要用敏捷呢？

这些团队最终的成果就是打造出了 *N* 个不同模样的、奇形怪状的、似是而

非的研发流程，不仅效率没有得到提升，团队成员在执行起来也显得蹩手蹩脚的。

所以，我一直建议的实施步骤如下。

初始阶段：在我们刚刚起步，什么都不懂的学习阶段，就让我们严格按照 Scrum 教科书，采用标准的方法、流程和工具，打好理论基础，学以致用，通过几次迭代的“生搬硬套”，真正理解并掌握 Scrum 的精髓。

优化阶段：有了几次迭代的真实数据和经验，我们可以开始尝试对 Scrum 流程中做得不好的地方进行总结分析和反思，看看到底是流程水土不服的问题，还是我们自身的问题，再针对性地进行优化。

推广阶段：我们已经形成了团队自己的一套模式，而且是通过实践证明且整个团队都认可的，这个时候，我们就可以开始结合公司实际的项目环境，总结流程，将其固化下来，作为“本地化”的 Scrum+ 流程，开始向其他项目组推广运行。

自此，敏捷研发流程才算真正地软着陆了。不过，不要忘记 Scrum 的核心理念：**持续改进**!!

问答（51）敏捷落地是不是很难？

背景

前几篇从什么是敏捷聊到了为什么要引入敏捷，又声明了敏捷并不是万能的，在这个过程中，不少同学产生了一种困难重重感，所以有人就在问我，敏捷落地是不是真的很难？

你问

敏捷落地是不是真的很难？

我答

今天我还是说说自己在敏捷落地过程中感受到的那些痛吧，看看是否能给不同的同学带去不一样的感同身受。

1．第一步：全员培训

（1）面向管理层、未来的产品负责人和敏捷教练的敏捷实施培训。

（2）面向公司所有员工的敏捷思想和基本概念的培训。

问题 1：在实施过程中，每个产品负责人和敏捷教练都按照自己的理解去灌输敏捷流程，并指导团队如何去执行，导致同一类型的项目，不同的敏捷团队却有着不同的流程。

分析 1：先严格按照 Scrum 的标准流程去执行，让团队通过实践，在理解并熟悉了 Scrum 的流程和方法之后，再结合原有的流程进行融合改造。同时，让各个 Scrum 团队里的产品负责人和敏捷教练定期开展工作坊形式的活动，交流分享各自的经验心得。

2. 第二步：在新产品的研发过程中要求使用 Scrum

问题 2：产品虽然是新的，但是团队成员都是从老的项目组中抽调出来的，所以大家不论是思想还是骨子里都还是传统的瀑布式流程，而且在开工之前并没有人提供相关的模板或工具，导致团队成员无从下手，最后只能又做成每个冲刺里包含着一个个小瀑布。

分析 2：产品负责人和敏捷教练通过理论结合实际，预先定义一些算法、公式和模板，包括但不限于用户故事的颗粒度、故事点大小的估算、团队容量的算法、相关会议流程和注意事项等。在前几次冲刺计划会议和反思回顾会议上做持续性的宣传贯彻和讨论。

问题 3：公司的产品团队在美国，因为时差和地域的限制，他们和本地团队的工作节奏很难同步，是否能由开发团队的开发人员或测试人员兼任 PO（Product Owner，产品负责人）？

分析 3：我们通过在本地设立 PPO（Proxy Product Owner，产品负责人代理）来解决这个问题，他们相当于美国产品负责人在本地的角色投影，他们从美国的 PO 那里获取产品的需求和相关信息，和本地的敏捷团队紧密合作。

问题 4：有些团队成员在学习敏捷时以为敏捷是轻文档的，就简单地认为文档能省则省，甚至包括功能规格说明书和测试计划等。

分析 4：很多敏捷的团队在初始阶段都有这样的误解。敏捷思想的意思是**有价值且可工作的软件胜于详尽的文档**，但是必要的需求文档、设计文档、测试计划、测试用例等在项目实施过程中都是必需的，否则无法做到良好的知识积累和传承。

在敏捷里只是不需要像传统的瀑布式流程那样，把文档作为每个阶段的一个输入条件，比如产品需求文档是开发设计阶段的输入内容、开发的功能规格说明书是测试用例设计阶段的输入内容等。在敏捷里，文档应该是一个输出产物，是产品经理、开发人员、测试人员在完成一项冲刺任务的同时，协同产出

的有价值并持续更新的一些知识和资料。

3．第三步：在本身就是一个月发布一次的项目中推行

问题 5：项目组成员在最初推行 Scrum 时会有较大的困惑，因为项目本身已经是一个月发布一个版本了，为什么还要转为敏捷模式？而且迭代周期也是一个月，那又有什么区别呢？

分析 5：产品负责人和敏捷教练引导团队从几个方面去感受传统的瀑布式流程和敏捷流程的区别，如对需求理解的一致性、对代码改动的回归测试范围的确定性、对项目整体进度和自身能力的可视化等。虽然从项目周期上看没有区别，但是从开发人员到测试人员在对质量的信心和减少需求实现偏差上都有了很大的提高和改进。

4．第四步：在公司的所有项目中开始运行

问题 6：管理层认为敏捷团队是一个自组织、自管理的团队，可以是铁打的营盘流水的兵，因此会随意调动人员去应对一些 EP（Emergency Patch，紧急补丁），或者在每个项目开始之前打散原有团队重新自由组合。

分析 6：敏捷教练需要通过对比执行过程中的问题分析和燃尽图，让管理层明白，一支团队的合作熟练度是需要时间去磨合的，当团队成员之间的默契度达到一个较高指数时，其战斗力将会呈几何级增长，任何任务都可以很好地完成。所以，我一直认为敏捷团队应该是铁打的兵团流水的营盘。另外，团队在制订冲刺计划时，应该预留 10% ～ 20% 的缓冲量来应对紧急补丁和一些临时任务。

问题 7：在引入 Scrum 的初期，因为美国的发布经理和运行维护部署团队并没有同步跟上转型的节奏，他们仍然按照瀑布模型里代码完成、代码冻结、工程师发布等里程碑去跟本地团队讨论发布和部署计划，让本地团队觉得同时在两种流程标准下开展项目很痛苦。

分析 7：产品负责人和敏捷教练在跟团队制订冲刺计划的时候，会把团队

承接的有代码改动的用户故事工作量控制在代码冻结之前完成，从代码冻结到工程师发布之间主要承接的是版本的回归测试、升 / 降级测试、发布文档的编写等用户故事。同时敏捷教练也会尽可能地去跟发布经理和运行维护部署团队统一沟通，保证团队不受太多的干扰。

问题 8：在前期引入 Rally（适用于 Scrum 的一种管理工具），增加了团队额外的工作量，从表面上看，工作效率反而下降了，团队成员产生了一些抵触情绪。

分析 8：敏捷教练在跟团队制订冲刺计划的时候就要把工具的学习量也算在内，并跟管理层和团队做好沟通，在敏捷团队的容量算法里要加上学习量的计算。因为团队是在不断总结、不断学习中成长的，不管是工具，还是新的项目、新的技术领域，都需要花时间去学习，这些都需要在每个冲刺开始制订计划的时候考虑进去。

问题 9：有的团队觉得过多的会议耗费了他们的精力，而且在一个冲刺结束之后就要立即投入下一个冲刺的工作当中，所以取消了项目结束之后的反思会议，还自以为对流程效率进行了优化。

分析 9：敏捷教练通过对几个冲刺的容量数据进行统计对比，让团队认识到这个值已经连续几个冲刺没有变化了，再引入一些成功案例进行对比，在团队里做几次分享和头脑风暴，并采用各种方法引导团队成员慢慢说出自己眼中的不足，持续总结、持续改进，让团队成员切身体会到这种反思会议正是敏捷中持续改进的重要基础，这样团队自然就不会觉得召开反思会议是在浪费时间了。另外，要切记，这种会不能开成抱怨大会或者批斗大会。

问答（52）如何建立自己的人脉？

背景

现在很多同学加入了各式各样的群体之后，就认为自己的人脉很广了，比如线上的微信群、QQ 群、小密圈，线下的读书会、俱乐部之类，然后就开始咨询如何利用这些“人脉”来扩大自己的影响力。针对这样的焦急心态，我先不说如何扩大影响力，而是要说说在现在这种新形态的群体里如何建立自己的人脉。

你问

我该怎么建立自己的人脉？

我答

在人们追求事业成功和幸福快乐生活的过程中，同样存在一个类似血脉的系统，称为人脉。如果说血脉是人的生理生命保障系统，那么人脉则是人的社会生命保障系统。常言道，“一个篱笆三个桩，一个好汉三个帮”，“一人成木，二人成林，三人成森林”，都是说，要想做成大事，必定有做成大事的人脉网络和人脉支持系统。

上述是传统人脉的名词解释，我们今天要说的是近几年越来越火的一些人脉形式，常见的有微信群、QQ 群、论坛，新兴的有线下社群、读书会、小密圈等。

我因为机缘巧合加入了一个微信群和一个小密圈的圈子，所以才对这些新型的人脉形式有了一些直观上的认识，它们和我们所说的传统人脉有着本质上的区别。

传统人脉一般都是通过熟人介绍、聚会等形式来维系原有的人脉和拓展新的人脉，其实就是采用各种手段拉拢一些可能用得上、也可能用不上的资源关系。

而这些新型的人脉形式，在你还没有任何积累的时候，所谓的人脉其实并不存在。或许你会说，怎么会不存在呢？我现在就是一个新手，就是一个小白啊，可我已经混迹在 N 个微信群、QQ 群或小密圈了。可是你想过吗？你所在的那些群或圈子真的算你的人脉吗？你可以试着在那些群里发布一条任务消息，或者发起一个讨论，再看看响应的人数，你就能认清现实了。

上述观点有点简单粗暴，可是我觉得话糙理不糙，但仅代表个人观点。

假设你加入了某个技术类的圈子，而且一直扮演着寻求帮助的角色，你会经常在里面问一些问题，而且多数还是小白型的问题，比如，×× 怎么入门啊？我想学 ××，我该读哪些书啊？我今天遇到了 ×× 问题，有谁能告诉我该怎么解决吗？

而在同一个圈子里同时有一类人，他们经常解答别人的问题，经常帮助别人解决问题，而且自己又在相关的领域有着丰富的经验和深厚的积累。

你觉得是那类人更容易被关注，还是你更容易被关注呢？

“ 物以类聚，人以群分 ”，能类聚、可群分的肯定是同一类人，代入我们刚才所说的，是不是就可以理解成人脉或圈子呢？

所以我认为，建立新型人脉最有效的方法是靠 “ 吸引 ”，而不是靠 “ 拉拢 ”。

吸引，顾名思义，就是把别人的目光和关注点引向自己。这要靠什么呢？靠的就是自我修炼。当你变得越来越强的时候，你会发现，关注你的人也会越来越多，也就是你吸引了更多人的目光和注意力，这就意味着你的人脉圈子也越来越大。

当你没有一点内功，或刚刚入门某个领域的时候，你眼里看到的是遥不可及的大神，身边都是跟你一样的小白。但当你的内功修炼到一个新的等级，上了一个或几个台阶时，你会发现你自己已经站在另一个层面上了，原来遥不可及的大神也近在咫尺，在不知不觉中，你的身边也聚集了一群和你一样厉害的人，这些人才是可以和你守望相助、互惠互利的，也就是所谓的人脉。

附录：关于敏捷研发模式相关知识点的个人阐述

Scrum 七剑（1）【Product Owner】

The product owner is typically a project ' s key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product.

为什么先引用 Product Owner 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的角色之一，在下文中就用常见的简称 PO 来表示了。

今天，我这个老兵就通过理论结合实践来说说 PO 在 Scrum 里到底是做什么的，后续会结合更深入的学习实践继续改进更新。

（1）PO 是敏捷团队里的一员，和敏捷教练及开发团队都保持着紧密的合作和及时的沟通。

（2）PO 一定是团队里最熟悉产品和用户的那个人。

（3）如果我们把 Scrum 看作一艘龙舟，PO 就是这艘龙舟上的 “ 舵手 ”，他掌控着整艘龙舟前进的方向，在整个航程中，确保整个团队将力气用在了正确的方向上。这也意味着，整个团队必须完全听从 PO 的决定，而且只需要听从 PO 的指引，其他任何利益相关人都无权越过 PO 去告诉团队该做什么。

（4）PO 很清楚自己想要什么，他有一个清晰而明确的愿景，并且会将这

个愿景传达给整个团队。我认为，这是评价一位 PO 优秀与否的关键点。一位优秀的 PO 很清楚自己要的是什么，或者说客户 / 用户想要的是什么，从而可以很好地保证整个团队总是优先在做最有价值的事情。而一位平庸的 PO 会让整个团队乱成一团，今天开发一项功能，明天又推翻重做，使团队成员完全丧失成就感。

（5）PO 需要维护 Scrum 里很重要的一个工件 —— PB（Product Backlog，产品清单）。在这份清单里，PO 会根据需求背景、用户期望、故事点大小等多方面因素对产品需求进行排序，并持续维护。PB 里的需求可能来自客户、用户、团队、管理层或者产品经理团队，但唯一有权限对其进行增、删、改的只有 PO。

（6）PO 在整个冲刺过程中必须参加的三次会议如下。

计划会议 ：PO 需要清楚地告诉团队要做什么，并要求整个团队对所有需求的理解达成一致。

评审会议 :PO 需要依据自己制订的 DoD（Definition of Done，完成标准）对完成的用户故事进行评审验收。

反思会议 ：PO 需要提出在这个冲刺里做得好的和做得不好的实践点，并从自身出发提出改进建议。

（7）PO 对产品负责，在每个冲刺里，他都应该全身心投入。对于一个大型项目来说，PO 也许会同时服务于多个敏捷团队，因为每个敏捷团队的规模不会超过 9 人。

（8）PO 一定是一个人，但他可能会得到一个产品团队的支撑。

（9）在有的团队里会有一种角色叫 PPO（Proxy Product Owner，产品负责人代理），这是由于公司架构的关系，产品经理团队和开发团队不在一座城市，甚至不在一个国家，出于沟通的及时性和效率考虑而设置的一种角色。

Scrum 七剑（2）【Scrum Master】

The scrum master is responsible for making sure a Scrum team lives by the values and practices of Scrum. The scrum master is often considered a coach for the team, helping the team do the best work it possibly can. The scrum master can also be thought of as a process owner for the team, creating a balance with the project ' s key stakeholder, who is referred to as the product owner.

为什么先引用 Scrum Master 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的角色之一，在下文中就用常见的简称 SM 来表示了。

今天，我这个老兵就通过理论结合实践来说说 SM 在 Scrum 里到底是做什么的，后续会结合更深入的学习实践继续改进更新。

（1）SM 是敏捷团队里的一员，和 PO 及开发团队都保持着紧密的合作和及时的沟通。

（2）如果我们把 Scrum 看作一艘龙舟，SM 就是这艘龙舟上的“ 鼓手 ”，他通过鼓点控制着所有“ 划手 ”的划桨节奏，在整个航程中，确保整个团队按照正确的方法和合理的节奏在使劲，以确保在既定的时间内，以最快的速度到达“ 舵手 ”指定的方向。

（3）SM 是一个具有双重身份的角色 ：教练 + 保镖。

（4）当 SM 作为教练时，他需要教会团队怎么玩转 Scrum 流程、方法和工具，让团队理解并接受敏捷思想，引导大家按照正确的方法去实践，并帮助团队一起找出每个冲刺中的改进点，让整个团队的战斗力越来越强。

（5）当 SM 作为保镖时，他需要保护团队在整个冲刺期内尽可能地不受外界干扰，比如临时任务、人员调动，甚至团队成员个人的一些问题，只要是会影响整个冲刺进度的因素，他都有责任去协调和解决。但一定要注意，SM 是“ 保

镖 ”,而不是“ 保姆 ”,很多 SM 在团队成员抱怨事情太多时会主动承担一些责任，比如更新任务墙、更新 Scrum 电子工具里的任务状态等。请一定要避免这种情况的发生，因为这是让团队成员通过自我操作来对自己的进度有一个直观感觉的机会，所以请一定不要越俎代庖。

（6）SM 在整个冲刺过程中需要组织和参加所有的常规会议。

计划会议 ：SM 需要跟 PO 和团队一起完成冲刺计划。

评审会议 ：SM 需要组织该会议，并给需求任务的完成者创造更多更好的展示时间。

反思会议 ：SM 需要组织并主导这次会议，引导大家积极地总结和反思刚刚结束的这个冲刺，收集到尽可能多的实践，再组织大家分析出其中的改进点，然后挑出高优先级的放入下一个冲刺里去尝试改进。同时，SM 还需要确保这次会议不会开成问题吐槽会或批斗会，这一点很重要。

每日站会 ：SM 需要组织这次会议，并控制会议的节奏和时长。通过这次会议，SM 需要了解到当前的任务进度和问题，并迅速协调资源，帮助问题的提出者找到解决方法。

（7）SM 对流程和团队负责，在每个冲刺里，他都需要全身心投入。对于一个大型项目来说，PO 也许会同时服务于多个敏捷团队，但 SM 建议专人专职，特别是在敏捷实施的早期阶段，SM 需要花费很多精力在学习、引领、指导、组织、跟踪和协调等多维度的工作上。

（8）什么才是优秀的 SM ？我最喜欢的一个回答就是 ：做到极致就应该是团队感觉不到你的存在，或者你在或不在时，团队都能正常运作。简单来说，就是一个优秀的 SM 做到极致就是让自己 “ 失业 ” 了。

（9）有一种组织叫 SoS（Scrum of Scrum），从我个人的实践经验出发，建议采取如下两种形态的 SoS。

Scrum of Scrum ：当一个项目是由多个子项目组成时，不可能由一个 SM 同时工作于所有子项目里的所有敏捷团队，这时就需要一个跨团队和跨子项目的横向沟通管理组织。

Scrum of Scrum Assistant ：当一个项目是由多个敏捷团队协作时，也许不需要每个敏捷团队都配备一名专职的 SM，但至少有一个团队成员兼职敏捷助理，协助 SM 开展一些工作。

Scrum 七剑（3）【Scrum Team】

A scrum team in a Scrum environment does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Because of this, Scrum teams develop a deep form of camaraderie and a feeling that “ we ’ re all in this together ” .

为什么先引用 Scrum Team 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的角色之一。

今天，我这个老兵就通过理论结合实践来说说 Scrum Team 在 Scrum 里到底是做什么的，后续会结合更深入的学习实践继续改进更新。

（1）Scrum Team 指的就是 Scrum 流程里的开发团队，他们会按照 Scrum 实践方法来实现每个冲刺里的用户故事，在每个冲刺结束的时候提交增量的产物。

（2）如果我们把 Scrum 看作一艘龙舟，Scrum Team 就是这艘龙舟上的“ 划手 ”，他们按照鼓点的节奏和正确的方法使劲，以最快的速度到达 “ 舵手 ” 指定的方向。

（3）Scrum Team 是开发团队，但它在理论上强调包含的不是传统研发团队里的那些角色，比如开发人员、测试人员、架构师、设计师等。

（4）我个人对于 Scrum Team（敏捷团队）中角色弱化思想的理解是 ：它提倡的并不是简单的有没有某种角色设定，而是为了避免在传统研发模式下存在角色之间的依赖关系，提倡团队的无缝合作，共同快速、高效地完成每个冲刺任务。

（5）有些敏捷团队在组建时，生搬硬套一些理论目标，希望开发和测试人

员能够互补，也就是开发可以去做测试的活，测试能去帮开发写代码，从而淡化职能角色。但我个人认为，这种做法的可行性不高，因为实际上很多公司在最初招聘开发、测试或设计师的时候，都是专岗专职，不会要求跨域技能。所以，要想让现有成员转型，是一项长期持续性的工作，也就意味着需要增加每个迭代的学习成本。

（6）Scrum Team 在整个冲刺过程中需要参加所有的常规会议。

计划会议 ：Scrum Team 需要跟 PO 和 SM 一起完成冲刺计划。

评审会议 ：Scrum Team 需要参加该会议，将完成的冲刺产物演示给 PO 和相关用户，这是他们的展示时间。

反思会议 ：Scrum Team 需要参加该会议，总结和反思刚刚结束的这个冲刺里好的和不好的实践，将好的实践形成自己的团队规则，将不好的实践放入下一个冲刺里进行改进。

每日站会 ：Scrum Team 需要参加该会议，依次回答三个问题 ：

· 我今天完成了什么？

· 我明天会完成什么？

· 我当前遇到了什么问题？

（7）一个标准的 Scrum Team 由 5 ～ 9 人组成，在一个冲刺里不允许有人员方面的变动。从个人经验出发，我一直认为 Scrum Team 最好是 “ 铁打的兵，流水的营盘 ”，而不是很多管理层在人力利用效率上所推崇的 “ 铁打的营盘，流水的兵 ”。因为我觉得从团队层面考虑，磨合是需要时间的，只有经过若干次迭代的持续磨合，一支团队的战斗力才能真正发挥出来。

（8）什么才是优秀的 Scrum Team ？我个人喜欢把 Scrum Team 比作佣兵团队，队伍里的每个人在某个领域都能独当一面，相互之间又有很高的配合度，彼此都很熟悉，熟悉其他人的习惯、风格和行事方法。

Scrum 七剑（4）【Product Backlog】

The agile product backlog in Scrum is a prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it ' s not necessary to start a project with a lengthy, upfront effort to document all requirements. Typically, a Scrum team and its product owner begin by writing down everything they can think of for agile backlog prioritization. This agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers.

为什么先引用 Product Backlog 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的工件之一，在下文中就用常见的简称 PB 来表示了。

今天，我这个老兵就通过理论结合实践来说说 PB 在 Scrum 里到底有什么用，后续会结合更深入的学习实践继续改进更新。

（1）PB 指的是 Scrum 流程里的产品清单，PO 和团队会基于这份清单得出冲刺清单和冲刺计划，所以你可以把 PB 理解成一个产品需求池和规划路线，PO 对它负责，只有 PO 才拥有创建和维护权限，它也是持续增长的一份清单。

（2）PB 通常包含如下几项内容：Bugs、Technical work、Knowledge acquisition、Features。下面会分开描述。

（3）Bugs：缺陷，通常来自用户反馈和之前冲刺遗留下来的已知问题。这类任务需求比较清晰单一，责任人也相对明确。

（4）Technical work：技术类工作，通常来自开发团队内部，如代码优化、模块代码重构、自动化脚本更新和性能优化等。这类任务需求比较清晰单一，责任人也相对明确。

（5）Knowledge acquisition：知识获取，通常来自产品交接及新项目所需

的知识学习、技术预研等。根据不同阶段的冲刺，这类需求的优先级会有所不同。

（6）Features ：功能或特性的需求，一般会用简短的文字进行描述，通常有固定的语法格式，在 Scrum 里称之为用户故事，顾名思义就是从用户的角度来描述用户渴望得到的功能。下面我们来重点说一下用户故事，你可以简单地理解为 PB 就是由多个用户故事组成的集合。

（7）用户故事的标准格式：As a role, I want to do, so that some reasons。即：作为“某种角色”，我想要“做什么”，所以“需要什么”。

（8）什么是优秀的用户故事？一个好的用户故事应该包括以下三个要素和六大特性。

三个要素：角色、活动、商业价值。角色和活动要描述清晰，商业价值的作用在于让团队能够清楚这个用户故事的价值，同时也能很好地理解真实的用户场景。

六大特性：独立性、可协商性、有价值、可估算性、短小、可测试性。

（9）产品负责人在编写用户故事的时候，需要定义好验收标准，也就是 Scrum 里常听到的 DoD（Definition of Done，完成标准）。因为 PO 在做冲刺验收的时候，需要一个清晰且唯一的标准，否则会导致 PO 和开发团队在验收问题上的扯皮。

Sprint 验收的 DoD：代码完成、单元测试完成、功能测试完成、帮助文档完成。

项目发布的 DoD：性能测试完成、安全测试完成、灾难恢复测试完成。

（10）用户故事之间尽可能地减少依赖关系，同时根据用户故事的标准尺寸 Story Point（SP，故事点）去评估任务大小。这里的故事点只是一个相对值，而不是绝对值。通常选定一个合适规模的用户故事作为 1 SP，再将其他用户故事与它进行比较，估算出相应的 SP 值。

Scrum 七剑（5）【Task Board】

When practicing Scrum, we can make the sprint backlog visible by putting it on a Scrum task board. Team members update the task board continuously throughout the sprint; if someone thinks of a new task（" Test the snark code on Windows 8.1 "）, she writes a new card and puts it on the wall.

为什么先引用 Task Board 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的工件之一，在下文中就用常见的简称 TB 来表示了。

今天，我这个老兵就通过理论结合实践来说说 TB 在 Scrum 里到底有什么用，后续会结合更深入的学习实践继续改进更新。

（1）TB 是指通过各种类型的载体，将 Scrum 过程中的各项事务放大并进行可视化展示。可以是物理载体，如白板 ；也可以是电子实体，如 Rally、Tower 等在线系统。

（2）如下图所示，TB 通常包含以下几个状态列。

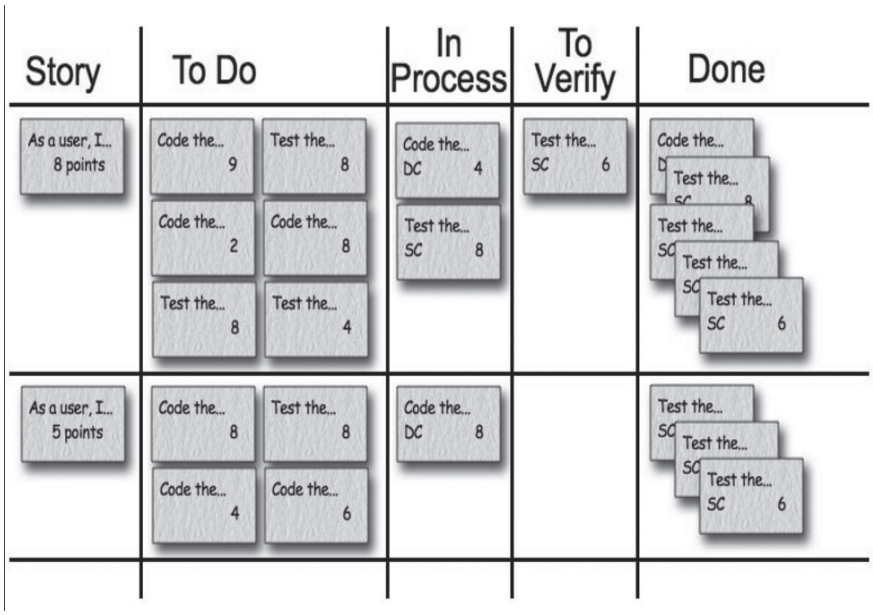
Story ：当前 Sprint 需要完成的用户故事。

To Do ：已经计划并认领过的任务，但还处于未开始状态。

In Process ：已经开始，正在进行中的任务。

To Verify ：已经完成开发工作，等待 PO 验收的任务。

Done ：PO 已经完成验收，处于已完成状态的任务。



团队成员会将自己的任务便签贴在对应的状态列里，这也是在每日站会前大家会同步做的一件事。

（3）我建议更新任务状态这件事一定要由每个人自己去做，而不要由 SM 代劳。因为在做更新这个动作的时候，每个人对自己的进度会有一个直观感受，是领先于别人，还是落后于别人，这种成就感或紧迫感如果不是亲身感受，效果就会大打折扣。

（4）有些团队觉得使用 TB 很麻烦，就会省略掉。我个人认为使用 TB 有以下几点好处，特别是使用物理白板。

透明性 ：每个人的任务量和进度对整个团队都是透明的，而不会仅仅局限在部分特定的团队成员中。因为不管是什么角色，如果想要了解这个冲刺的状态，则只需看一下 TB 就可以了。

鞭策性 ：因为任务和进度的透明化，无形中会给每个人增加压力。不仅有来自内部的驱动力，当你看到自己的进度落后于他人时，你会自我反省，查找原因，然后尝试改进自己的做事方法或者策略，力争将进度赶上去 ；也有外部

压力，你也不想你的领导或其他同事每次在路过你们的 TB 时，都看到你大部分的任务都在 To Do 和 In Process 列吧。所以，物理白板能鞭策每个人提高自管理能力，做到持续改进。

（5）也有一些敏捷团队的 TB 会跟本文中提及的状态列有所不同，如下。

Story Test-Driven Development ：故事测试驱动开发，指的是在开始写用户故事的代码之前，先完成测试用例的设计，每部分代码只有在通过故事测试之后才算完成。

Acceptance Test-Driven Development ：验收测试驱动开发，指的是在开始写用户故事的代码之前，先完成验收测试点的设计，每部分代码只有在通过验收测试之后才算完成。

Scrum 七剑（6）【Sprint Burndown Chart】

Progress on a Scrum project can be tracked by means of a sprint burndown chart. The scrum master should update the release burndown chart at the end of each day.

The horizontal axis of the sprint burndown chart shows the sprints; the vertical axis shows the amount of work remaining at the start of each day. Work remaining can be shown in whatever unit the team prefers - - story points, ideal days, team days and so on.

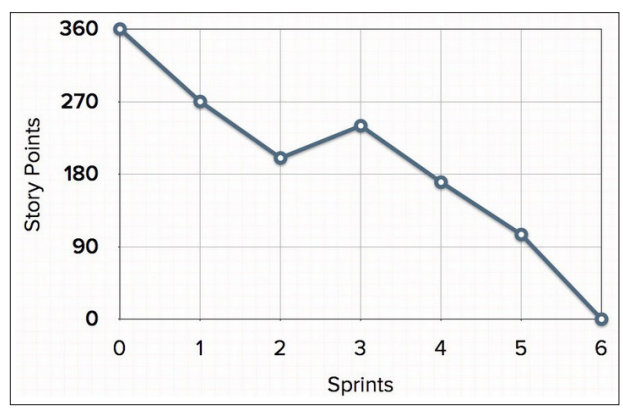
为什么先引用 Sprint Burndown Chart 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的工件之一。

今天，我这个老兵就通过理论结合实践来说说 Spring Burndown Chart 在 Scrum 里到底有什么用，后续会结合更深入的学习实践继续改进更新。

（1）Spring Burndown Chart 就是敏捷教练跟团队和管理者沟通每个冲刺进度的工具。

（2）右图中的 *Y* 轴显示的是剩余故事点数，*X* 轴显示的是天数（该图显示的是冲刺数，因为这是一张燃尽图）。

（3）燃尽图由 SM 每天更新，体现所有未完成任务的剩余工作量。燃尽图不记录实际耗费的工作量。



燃尽图

（4）燃尽图的 *X* 轴可以是天数，也可以是冲刺数。当 *X* 轴为天数的时候，代表这是 Sprint Burndown Chart（冲刺燃尽图）；当 *X* 轴为冲刺数的时候，代表这是 Release Burndown Chart（发布燃尽图）。

（5）燃尽图可以作为一个主要指标来预测工作完成时间。在任意一个时间点，我们都能根据历史数据趋势线，对比看出如果当前的节奏和范围不变，预计完成时间的三种状态。

按时：当趋势线和横轴的相交点接近于冲刺持续期末尾时。

提前：当趋势线和横轴的相交点偏左时。

延期：当趋势线和横轴的相交点偏右时。

（6）既然说到了燃尽图，我们也顺带看一下燃烧图（Burnup Chart）。与燃尽图相反，燃烧图表示的是在达成发布和冲刺目标过程中所完成的工作量。

（7）燃烧图的 *Y* 轴与燃尽图一样，可以用小时数，也可以用故事点，但建议用故事点，因为在冲刺结束的时候，我们需要统计敏捷团队在这个冲刺中所完成的有价值的用户故事，这些都是用故事点估算的。

（8）由燃尽图在 Scrum 中的作用不难看出，冲刺的执行并不是由一个完备的事先计划指导的；相反，冲刺的执行更接近见机行事，SM 充分利用团队的技能、对已完成任务的状态和进度的反馈，根据燃尽图来实时调整工作节奏和工作量，从而让整个冲刺或发布能够如期完成。

Scrum 七剑（7）【Sprint Retrospective Meeting】

No matter how good a Scrum team is, there is always opportunity to improve. Although a good Scrum team will be constantly looking for improvement opportunities, the team should set aside a brief, dedicated period at the end of each sprint to deliberately reflect on how they are doing and to find ways to improve. This occurs during the sprint retrospective.

The sprint retrospective is usually the last thing done in a sprint. Many teams will do it immediately after the sprint review. The entire team, including both the scrum master and the product owner should participate. You can schedule a scrum retrospective for up to an hour, which is usually quite sufficient. However, occasionally a hot topic will arise or a team conflict will escalate and the retrospective could take significantly longer.

为什么先引用 Sprint Retrospective Meeting 的原文解释呢？因为很多名词的释义在被翻译成中文之后，要么比较晦涩，要么有所偏差，通过原文能更精准地理解这个 Scrum 里重要的工件之一。

今天，我这个老兵就通过理论结合实践来说说 Sprint Retrospective Meeting 在 Scrum 里到底有什么用，后续会结合更深入的学习实践继续改进更新。

（1）Spring Retrospective Meeting 是 Scrum 三大会议中最重要的一个，它通常是在每个冲刺结束的时候，由 SM 组织、全员参与的一场反思会议。

（2）这个会议经常会被团队在实施敏捷的时候省略，因为他们往往忽视了总结、分析、持续改进对于团队战斗力的提升价值，认为在每个冲刺结束时都要开一次反思会议太浪费时间了。其实套用中国的一句古话，“ 磨刀不误砍柴工 ”，这个会议其实就是一个磨刀的过程。

（3）在这个会议上，SM 需要组织团队成员把刚刚结束的冲刺里做得不好

的地方和做得好的地方都说出来，好的实践要总结、提炼，形成自己团队的规则保持下去 ；不好的实践就需要进行 3W 分析，找出根本原因和解决方案，列入需要改进的清单里，最终从待改进清单里挑出 2 ～ 3 个合适的、优先级高的改进点放入下一个冲刺里去优化实践。

（4）主持反思会议需要注意以下几点，也可以理解为营造会议氛围的注意事项。

就事论事，对事不对人，不能把反思会开成批斗会或吐槽会。

人们一般很难当面指出自己或其他人的问题，不能把反思会开成 SM 一个人的分析汇报会。前期可以采用匿名提交便笺纸的形式，参会者每人在纸上写上 3 件好的实践和 3 件不好的实践，然后交给 SM，再列到白板上，挑出票数最多的 2 ～ 3 件实践，大家集中讨论，依次击破。

养成每个人都要发言、都要积极参与的习惯。

（5）反思会议的参会人员必须由 SM 和团队共同决定，PO 是否参与取决于 PO 和开发团队之间是否信任和有安全感。管理者不建议被邀请，因为那会使整个反思会趋于“ 安静 ”或“ 正常 ”。

（6）反思会这种形式其实在我们的日常工作和生活里也经常被使用，比如职场中的月度总结和年度总结，家庭每个月或者每个季度的财务盘点，个人按周或按月的阅读计划和写作计划的盘点，都是不同场景下的反思会模式。

到这里，Scrum 的 7 把剑都介绍完了。这 7 把剑其实不仅能应用于 Scrum，在工作和生活的方方面面都可以灵活应用它们。我认为，这种所谓的方法论本身就来源于生活，它们都是跨界而来的，是敏捷的先行者从制造行业等其他百年行业中汲取改造而来的，所以它们能被跨界应用到其他领域也就顺其自然了。

写好用户故事

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

什么叫用户故事？就是从**用户的角度**来描述**用户渴望**得到的功能，英文为User Story。

一个完整的用户故事应该包括三个要素 ：角色、活动、商业价值。

一个好的用户故事应该遵循 INVEST 原则，包括以下六大特性 ：

INVEST = Independent, Negotiable, Valuable, Estimable, Small, Testable

独立性 ((Independent)) .:要尽可能地让一个用户故事独立于其他的用户故事。用户故事之间的依赖关系会使得制订计划、确定优先级和估算工作量变得很困难。通常，我们可以通过组合用户故事和分解用户故事来减少依赖性。

可协商性 ((Negotiable)) .:一个用户故事的内容应该是可以协商的。用户故事不是合同。一张用户故事卡片上只是对用户故事的一个简短描述，不包括太多的细节。具体的细节在沟通阶段产出。如果一张用户故事卡带有太多的细节，则实际上限制了和用户的沟通。

有价值 ((Valuable)) .:每个故事必须对客户具有价值（无论是用户还是购买方）。一个让用户故事有价值的好方法是让客户来写下它们。一旦客户意识到这

是一个用户故事，而不是一份契约，而且可以进行协商的时候，他们将非常乐意写下用户故事。

可估算性（(Estimable)）：开发团队需要估计一个用户故事以便确定优先级、工作量，安排计划。但是让开发者难以估计故事的问题来自对领域知识的缺乏（在这种情况下需要更多的沟通），或者故事太大了（这时需要把故事切分成一些小故事）。

短小（(Small)）：一个好的用户故事在工作量上要尽量短小，最好不要超过 10 个人日的工作量，至少要确保在一次迭代或冲刺中能够完成。用户故事越大，在安排计划、估算工作量等方面的风险就会越大。

可测试性（(Testable)）：一个用户故事应该是可被测试的，以便确认它是可以完成的。如果一个用户故事不能被测试，就无法知道它什么时候可以完成。所以，一个好的用户故事还需要定义 DoD（Definition of Done）。

其实不仅功能需求才能被转化为用户故事，并加入产品清单里，很多相关的需求都可以转化为用户故事，比如代码重构、架构优化、性能需求、安全需求、上线需求等。

很多 PO 在写用户故事或者维护产品清单的时候，经常会混淆用户故事和任务；或者说敏捷教练在和团队沟通时，也分不清用户故事和任务到底有什么区别。

下面基于我的经验和理解来阐述一下用户故事和任务的区别。

1. 概念

用户故事的定义是用户想要得到什么样的功能。

任务的定义是描述功能怎么被实现。

2．承接对象

用户故事因为包含的是一个完整的功能，所以承接对象一般来说就是开发和测试，也有可能会分为接口开发、后端开发、前端开发、UI 设计师和数据库设计师等。

任务因为指的就是一个单一的任务项，所以承接对象就是一个人。

3．颗粒度

用户故事是通过若干项任务共同实现的，所以说它其实就是一个多任务的集合。

4．生命周期

用户故事源于用户，载体就是产品清单，贯穿于整个发布的始终。

任务源于用户故事，载体就是冲刺清单，贯穿于每个冲刺里用户故事从开始到结束。

从这几个维度是不是能够较为清楚地理解用户故事和任务的区别了？

反侵权盗版声明