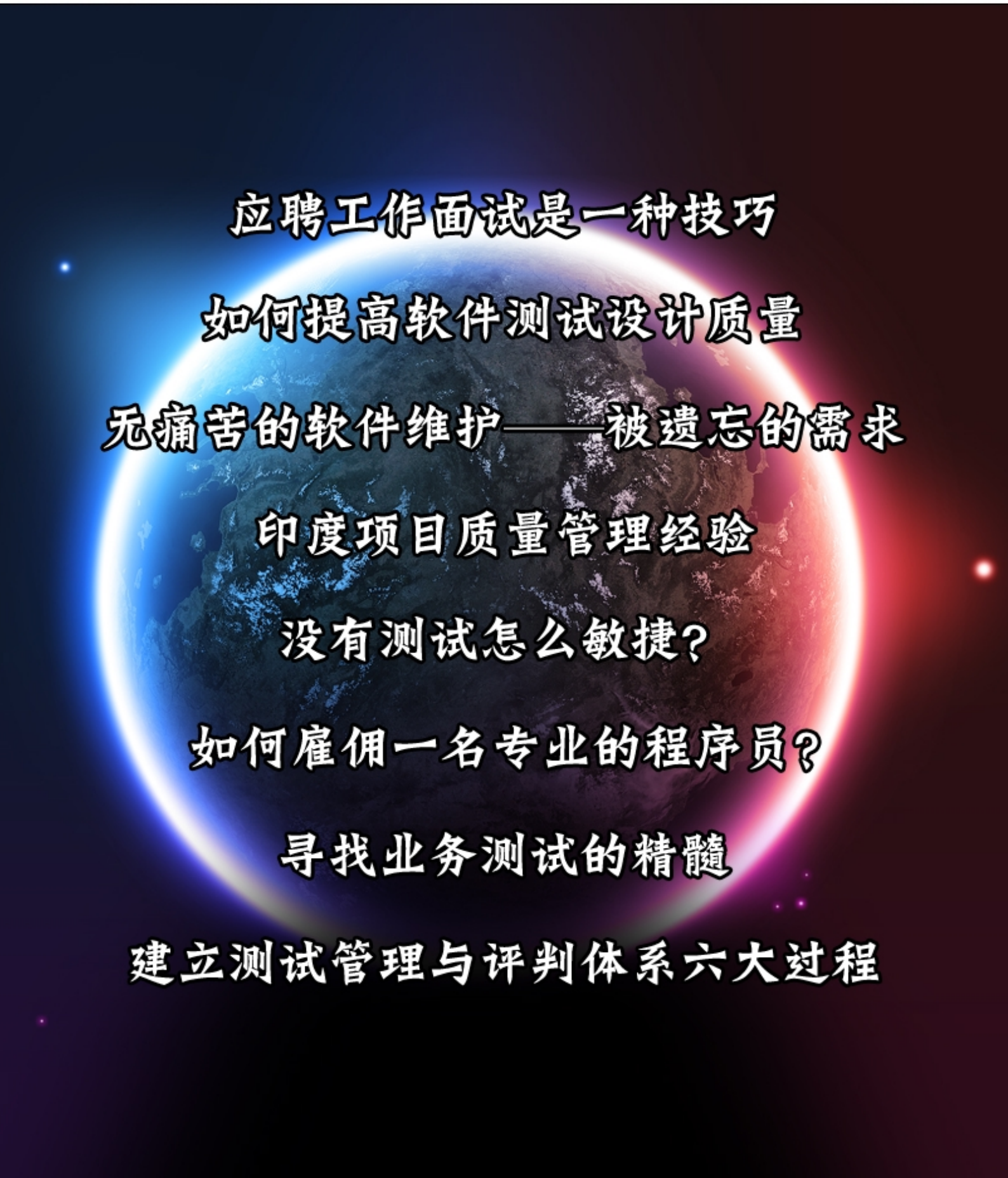
泽众软件测试网
电子期刊

SPASVO

第十七期
2013年5月



应聘工作面试是一种技巧

如何提高软件测试设计质量

无痛苦的软件维护——被遗忘的需求

印度项目质量管理经验

没有测试怎么敏捷？

如何雇佣一名专业的程序员？

寻找业务测试的精髓

建立测试管理与评判体系六大过程

上海泽众软件电子期刊

2013 年 5 月 第十七期

主办单位：上海泽众软件科技有限公司

联系电话：021-61079698 传真：021-61079698 转 8017

意见反馈：fangmh@spasvo.com 投稿：wangmf@spasvo.com

公司地址：上海市普陀区曹杨路 450 号绿地和创大厦 18 楼 1801 室

邮政编码：200063

公司主页：www.spasvo.com 论坛：bbs.spasvo.com

目录

应聘工作面试是一种技巧

面试是一件有趣的事情。要想获得一份编程开发工作，首先面试要面的好。能否被招聘单位选中，通常跟你能掌握的编程技术关系不大，这是程序员心中最典型 的痛处。我们大多数人都觉得这种招聘方式应该改革。而对于我个人而言，我觉得还可以接受。它需要改进，但我要来告诉你如何在这种情况下获得一份工作。

面试是一种技巧

把面试当作一种技巧。你需要练习这种技巧。这种技巧跟编程有很大的区别。通过这些年对程序员的观察发现，缺乏经验的新手通常会认为，他们跟那些熟练的 程序员知道的知识一样多，知道的知识比不知道的多。于是，当你成长为一名熟练的程序员后，你会发现自己是一名最差的程序员，因为你看到了自己不知道的那些 知识。当面试时，这会成为一个很大的麻烦，因为其实你是在把自己售卖给公司。如果你不认为货物是那么的好，怎么可能把东西有效的卖出去?下一步你要考虑的 是如何把自己卖出去。

准备你的推销词

我坚信一条，你一定要为你的面试做好行动计划。熟悉你在简历上写的任何东西。对最常见的可能会被问到的问题准备好答案。把它们用某种方式跟你的过去的 经历拉上关系联系起来，突出你的能力。当面试官没有什么好说的时候，推出你的推销词。我曾有过几次面试，面试官只有几个在开始和结束时有时间问了几个问 题。准备好你的推销词，让他们的问题的答案已经在推销词里体现出来，这十分的有效。

程序样品

在面试时编程是一件很傻的事情。它并不能反映出一个程序员的真实编程水平，我也不相信它能给面试官带来更好的判断。我发现一种最简单的能避免在面试时 进行编程的办法是，事先准备一些能拿给面试官看的程序样品。很重要的一点，你既要让面试官看到这些程序，也要让他们知道这些程序能运行。就我来说，我在业 余时间做了一个 Rails 项目，把它放在了 heroku 主机上，面试官不但看到它，而且能简单的对它进行操作。

所有的面试都是对你的练习

很多程序员在面试时都会很紧张。我用提醒自己“所有的面试都是一种练习”来消除紧张。如果没有应聘上，没什么大不了的。再找一家再面试就行了。就这样，我让我的面试技能得到了非常好的修炼。当然，在每一次面试之前，你一定要从上一次面试中总结经验。

评估你的表现

上次面试我什么地方做错了?怎么样才能改进?什么地方做的很好?在面试之后尽快的检讨自己的表现，这很重要。因为在面试刚刚结束后，你的记忆还很清晰，你不会漏掉什么细节。

如何提高软件测试设计质量

测试设计重要性

设计是测试的灵魂，质量的龙头。

测试设计面临问题

1、测试对象的逻辑路径和测试输入数据的组合几乎是无穷的，而穷尽的测试是不可能的

2、不同利益相关者对软件或者软件产品的质量要求是不同的

3、测试时间和资源有限

4、测试得到的需求和资源不完整

5、测试设计语言规范

穷尽的测试是不可能的

1、如何有效减少测试用例的数目？

2、如何避免测试用例之间的冗余？

3、如何满足测试覆盖率的要求？

如何有效减少测试用例的数目？

1、等价类

（1）有效等价类

（2）无效等价类

2、边界值

如何避免测试用例之间的冗余？

1、规范测试设计

（1）按照一定的设计思路进行测试用例设计

（2）减少热带风暴

2、可复用的测试用例

特性：通用性、有效性、独立性

如何满足测试覆盖率的要求？

测试覆盖率常用的计算公式：

1、 功能覆盖率

至少被执行一次的测试功能点数/ 测试功能点总数（功能点）

2、 需求覆盖率

被验证到的需求数量 /总的需求数量（需求）

3、覆盖率

至少被执行一次的测试用例数/ 应执行的测试用例总数

4、语句覆盖率

至少被执行一次的语句数量/ 有效的程序代码行数

5、判定覆盖率

判定结果被评价的次数 / 判定结果总数

6、条件覆盖率

条件操作数值至少被评价一次的数量 / 条件操作数值的总数

7、判定条件覆盖率

条件操作数值或判定结果至少被评价一次的数量/(条件操作数值总数+判定结果总数)

8、上下文判定覆盖率

上下文内已执行的判定分支数和/(上下文数*上下文内的判定分支总数)

9、基于状态的上下文入口覆盖率

累加每个状态内执行到的方法数/(状态数*类内方法总数)

10、分支条件组合覆盖率

被评测到的分支条件组合数/分支条件组合数

11、路径覆盖率

至少被执行一次的路径数/程序总路径数

目前采用覆盖策略

1、基于需求的测试覆盖评估

依赖于对已执行/运行的测试用例的核实和分析，所以基于需求的测试覆盖评测就转化为评估测试用例覆盖率：测试的目标是确保100%的测试用例全部成功地执行。

2、基于代码的测试覆盖评估

是对被测试的程序代码语句、路径或条件的覆盖率分析。如果应用基于代码的覆盖，则测试策略是根据测试已经执行的源代码的多少来表示的。这种测试覆盖策略类型对于安全至上的系统来说非常重要。

基于代码的测试覆盖评估

1、针对于 java 的项目采用 EMMA 工具进行分析

2、针对于 delphi 项目采用工具还没有定

不同利益相关者对软件质量要求是不同的

1、如何获取利益相关者的不同质量要求？

2、如何设计测试用例以满足不同的质量要求？

3、如何分析和评估最终软件产品的质量？

4、如何提高客户对软件产品的满意度？

测试时间和资源有限

1、如何有效的选择测试重点？

2、如何确定测试优先级？

3、如何有效的配置测试资源？

4、如何分析和评估测试的有效性？

如何有效的选择测试重点

1、软件产品的每个功能模块，对于客户而言并不是是同等重要的；

2、用户实际使用过程中的使用频率的分布

基于风险的测试策略

1、选择测试重点：对测试范围进行优先级的划分，将测试设计和执行放在优先级高的区域。

2、将测试资源分配到高风险的地方，从而更加有效的利用测试资源；

3、确定测试对象在哪些地方容易出现错误，并设计相应的测试用例来发现这些错误；

4、基于风险可以更好的对测试状态和测试结果进行报告，从而更好的对测试过程和测试质量进行监控，并根据实际的测试状态对后续测试进行及时调整。

测试得到的需求和资源不完整

1、如何获取尽量多的显现需求和隐现需求？

2、如何利用已有的经验有效设计测试用例？

3、如何应对需求的不断变更？

如何获取尽量多的需求？

1、 站在用户角度出发

（1）了解用户原始需求

目前公司需求来源于需求人员和项目经理的理解，我们如何得到用户的需要和期望

（2）了解行业隐含需求

（3）需求复用

2、 利用逆向思维方式推测问题

用户当前遇到的问题，是否可以在系统中找到解决方案

如何利用已有的经验有效设计测试用例？

错误推测法

（1）基于代码

（2）基于业务

（3）基于代码的编写者

（4）基于系统框架

如何应对需求的不断变更？

1、变更流程的规范

2、测试流程规范

测试设计语言不清晰导致问题

1、执行者的痛苦

2、测试覆盖率下降

3、测试执行效率下降

4、测试设计的工作成果不被认同

测试用例设计语言规范

1、不要采用一些个性化语言或简称来描述用例

2、用例描述必须含有主谓宾

3、预期结果采用 smart 原则，必须量化、具体

例如：验证数据正确性

4、涉及类似产品，在产品间描述保持一致

5、同一界面的相同实体描述名称必须一致

6、专业术语必须准确一致

无痛苦的软件维护——被遗忘的需求

先说一个小笑话。有一个生产队队长，他对专家说：“现在我们生产队的地越来越多，牛越来越忙不过来了。我想要这么一种牛，他吃的草和普通牛一样多，但 是干的活是普通牛的十倍。”专家说：“这种牛是可以造出来的，现在有基因工程。”队长说：“好吧，你给这造几头这样的牛。”于是专家找到了生物实验室，让 生物实验室的人搞一个基因工程，把牛造出来。于是工程浩大，投资无法保证，合作多半是不愉快的收场。

现实世界里很多人分析需求的过程就类似于这位专家，他们把注意力放在用户提出的功能点上，而对用户的实际需求没有兴趣。有不少软件公司和程序员，其实 都在做类似的基因工程。如果这个专家把注意力放在生产队长的业务需求上，而不是太在乎他提出的功能点，他会说：“我认识一个卖拖拉机的，可以带你去看 看。”

软件的维护为什么这么痛苦，一个很重要的原因在于：需求已经被遗忘了。

需求是对用户具有直接商业价值的活动，而不应该牵涉到任何的功能实现方式。实现同一个需求可以使用多个方案，每个方案有自己的功能方式，在某个方案中至关重要的功能点，也许在另一个方案中根本无关紧要。

瀑布式的开发过程，首先是由一批懂得用户业务的专家去调查用户的需求，分析出这个系统应该具有哪些功能，形成一个非常重要的文件——《xxx 系统需求 规格说明书》。客户认可了这个《说明书》，在上面签字盖章，加入合同附件，到时候项目验收就以此为准。这时候，需求就已经分解成了一个个功能点，从这时候 开始，需求本身就渐渐被人们遗忘。设计人员围绕着这些功能点进行工作，考虑用什么样的技术手段把功能点制造出来。功能点的制作细节形成了另外一份重要的文 件——《xxx 项目设计规格说明书》。这个《设计规格说明书》交给程序员去进行编码。

这样的做法在开发中已经形成了很大的问题。

首先，面对这样的《需求规格说明书》，设计人员已经无法知道最初的需求是什么。假如这个《需求规格说明书》中的功能点没有出现互相矛盾的情况，而他们 串起来却和用户的需求不同，设计人员没办法发现这样的情况，编码人员也无法发现。假如测试也是根据这个《需求规格说明书》来做的，测试人员也发现不了。直 到最后客户看见这个程序展现在他们面前。需求的分析需要在随后的过程中不断得到反馈，传统的过程不是没有反馈，而是反馈的时间太长了。

其次，由于设计人员已经无法知道基本的需求是什么，也就无法对业务进行建模。这样的需求分析是以开发人员的需要为核心的，可是结果恰恰妨碍了开发人员 对需求的理解。如果开发人员对用户的业务过程不甚了解，他们只有一种选择：不要试图去了解需求了，直接按照这些功能点做吧。于是，他们建立的对象模型就不 是以需求为核心的，而是以功能、界面为核心的。我见到过很多这样的系统，开发者确实有很高的抽象思维水平，程序中设计了非常巧妙的控制器和界面，可以很方 便的进行开发和变更。唯独业务层的对象非常简陋，一旦发生实际业务的变更，仍然十分辛苦。

更大的困难发生在维护程序的时候。

假设有一个移动通信公司需要制造一个系统，用来解决手机用户入网的问题。这个需求有下面几个要素：

1：用户付钱，得到一个 SIM 卡和一个号码，把这个 SIM 卡装到手机里就可以通话。
2：营业员收的钱要记录下来，提供给稽核人员，现金和帐目必须是平的。
3：用户付的话费要划入他自己的帐户，可以打印票据。
4：用户要在入网合同上签字，然后营业员把合同归档。

这几个要素都是和通信公司的商业利益直接相关的，没有牵涉到任何系统实现方式。如果不考虑通信公司内部的业务规范，实现方案可以有几十种，下面列举两种：

1：SIM 卡发给营业员，用户入网的时候，选择一个号码，然后付钱。营业员把 SIM 卡号码和电话号码输入系统，在交换网络上进行注册，这个 SIM 卡就可以通话了。然后各种费用记入各人帐户，合同归档。

2：SIM 卡在下发给营业员之前，先在交换网络上和注册，并且已经预先设置了一定的话费。用户选择了这个号码，付钱之后直接 SIM 卡拿走就可以打电话了。营业员事后再输入用户的合同资料，费用计入各人帐户，合同归档。

这两种方案在实现过程上是不同的，因此具有不同的功能点。比如第二种方案中的 SIM 卡在出售之前是可以进行通话的，所以必须对这样的号码的通话费用进 行监控，这个功能在第一种方案中是根本不需要的。并且两种方案在帐目的核对方式上区别也是比较大的。这两种方案是不同的功能点的集合，他们完成的是同一个 业务需求。

系统在开发阶段如果没有保留用户的业务需求情况，而是只留下一个功能点的列表，会给维护人员带来成很大的困难。维护人员无法从这样一堆功能点中发现最 初的需求是什么样子。各位可以试试，假设我们忘记上面的四个需求要素，只看下面的某个实现方案，从这个复杂的实现过程中，我们很难知道用户现在的需求到底 是什么。一旦需求发生了变化，这些功能点就会出错，或者是功能点的时序发生意料不到的错误，也许帐目核对不上了，也许是用户拿走的 SIM 卡不能打电话了。

看不见需求在哪里，不知道手里这段代码会触动需求的哪根神经。维护人员的痛苦大部分来源于此。

“不要紧，客户记得自己的需求。”但是客户通常不懂技术，即使他们懂技术，他们也不知道系统是如何实现的。如果开发人员依靠客户提出新需求的解决方案，结果就是让软件工程变成“生物工程”。到最后是钱基本花光，人基本累死，甲乙双方感情基本破裂。

软件开发必须划分成几个过程，但是各个步骤应该有一个统一的核心——业务需求。

在需求调查阶段要搞清楚用户的业务需求，为了达到这个目的，可以提问回答，可以对用户进行跟踪采访，或者做一个 demo 给用户看看，最终的目的是为了搞清楚用户在做什么事，遇到了什么问题，程序应该去解决什么问题，这就是这一阶段的工作。

然后开始进行设计，设计系统的逻辑结构和物理结构，逻辑结构要符合需求的概念，各个对象互相调用要能够实现需求中的业务过程，同时物理结构划分合理，符合实际的分布状况，可以达到要求的的性能，业务过程的物理运行方式合理高效。这一阶段仍然是以业务需求为核心。

接下来是编码。首先是编写一些基础设施，比如网络通信、数据库、文件的读写、分布式计算，这些基础设施和业务需求没有什么关系，有很多现成的框架，借 助这些框架我们可以尽快度过这个黑暗的阶段。然后编写业务对象，这时候业务需求中的一些概念逐步出现在代码中，比如上面说的那个例子，“用户”、“号 码”、“合同”、“入网”、“SIM 卡资源”这样的业务要素逐渐出现，这些对象所拥有的属性、可以运行的行为也和现实的需求一样。接着这些业务对象串接起 来，实现业务过程，现在业务需求又

回到了人们的视野当中。业务需求是什么，如何实现，在这里一目了然。最后将这些过程在 UI 或者接口中调用，将功能提供给 用户或者别的系统。

测试更是要围绕着业务需求来进行，正常的业务流程应该产生正常的结果，如果缺少某个资源，或者输入了不合适的数据，应该出现业务含义明确的异常。并且系统的业务对象是处在一个独立的层次上，与 UI 和基础设施没有很大的关联，这样可以方便的采用自动化的测试方法。

这样的系统维护起来一定少很多痛苦。

印度项目质量管理经验

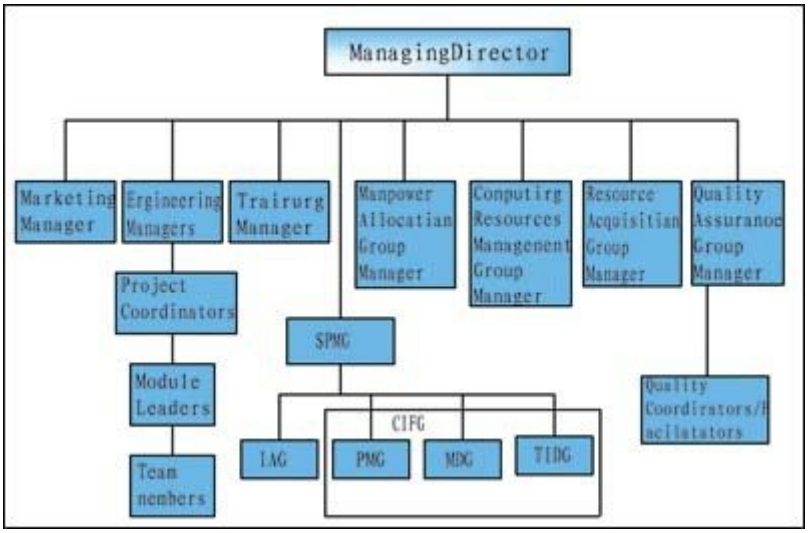
计算机和通信技术的迅速发展，特别是 Internet 技术的发展与普及，为企业内部、企业与外部提供了快速、准确、可靠的信息交流渠道。信息化企业运 作管理系统已成为企事业单位参与全球市场竞争的必备支持系统。正是由于这样的市场需求与技术发展现状，为我国的 IT 行业带来了空前发展的机遇，特别是软件 行业。软件企业能否抓住这样一个难得的发展机会需要多方面的努力，其中软件质量保障在其发展过程中占有重要的位置。 众所周知，印度已成为世界上软件业增长最快的国家，目前每年软件业产值达数十亿美元，并且还在以每年30%～50%的速度增长。比较我国和印度的软件产 业，就不难发现:中国拥有巨大的软件市场和世界公认的软件开发资源，在基础研究和对技术前瞻性的把握上，也有自己的优势，就整体社会经济环境而言也优于印 度。此外，中国的软件开发人员费用比较低廉，仅是世界市场的1/3左右。虽然中国人并不缺乏软件开发的天赋，但是在越来越强调规模化经营的今天，先天不足 的管理痼疾使我们举步维艰，难以摆脱小作坊式的软件开发模式。而印度软件业从一开始就立足于为美国软件企业服务，并遵循其软件开发的管理模式，与国际标准 接轨。

管理上的问题不能得到彻底的解决，软件的质量保障就无从谈起。笔者最近在与印度一家通过了CMM4级评估的软件公司（以下简称 A 公司）进行合作的过程 中，较为详细地了解了他们有关项目管理的一些详细情况，更深刻地感受到了项目管理的规范化与企业软件质量保障之间的密切关系。下面想着重从软件企业的构 架，软件项目计划、项目管理、项目经理的职责等方面对印度软件的项目管理及我国软件质量保障应注意的问题进行一些经验总结，供业内人士参考。

1.软件企业的组织结构

(1)A 公司结构

下图是 A 公司的组织结构图，同国内公司差异较大的部门有 QA、SSG 和人力资源部门。



* A 公司中，QA(Quality Assure)部门与研发部门独立，负责监督流程的执行。QA 同时负责领导与研发部门组成的联合工作组，制定公司流程。

* SSG(System Support Group)类似我们的 IT 部门，负责公司所有计算机软件和硬件资源的分配和管理。所有的办公环境和开发/实验室环境由 SSG 负责安装和维护，计算机资源 属于 SSG，由各个项目向 SSG 提出需求，项目结束后，设备需要交还给 SSG。个人和项目组没有固定的软件和硬件资源。SSG 是与研发平行的部门。

* 人力资源部门负责公司的人力资源管理，并维护员工的技能数据库。项目开始时，项目组向人力资源申请人力，向 SSG 申请计算机硬件和软件。项目结束时需要释 放计算机资源给 SSG，释放人力资源到人力资源池，并同时更新员工的技能数据库。研发部门的人力资源由研发总负责人和其助手分配(类似我国各公司的人力资 源部)。

(2)项目组结构

1) A 公司对项目组进行独立核算，项目具体负责人为 PC(Project Coordinator)，负责项目计划和执行，对项目具体成员进行分工。在每个阶段的结束会议上(如概要设计结束)，PC 要接受 QC(Quality Coordinator)的审查。除了 PC 与 QC 的接口外，所有其他外部接口都由 EM(Engineer Manager)完成，EM 负责与客户打交道，向 SSG、人力资源要求资源，与其他项目组协调进度。

2) 汇报关系为:

Team Member->Team Leader->PC->EM->研发总负责人。

3) 印度工程师分为7级，半年一次考评，即半年有一次升级机会。

1级:Software Engineer，刚毕业的本科生和研究生。

2级:Senior Software Engineer。

3级:Project Leader。

4级:Project Manager。

5级:Senior Project Manager。

3级可以成为 PC，4级可以成为 EM。刚开始平均2年升一级，越往后升职越慢。

A 公司规定，一人最多可以同时兼任两个项目的 PC，EM 管理的项目没有限制。

A 公司通常的项目组为4到5人，最多不超过10人。

以上是 A 公司（同时也是印度大多数规范化的软件公司）的组织结构和项目组结构。可以看出，A 公司的组织结构非常清晰，各个部门分类非常细，任务明确， 软件生产的每一个步骤都有专门的部门、专门的人员负责，从最基础的开发人员到负责统领全局的总经理，层层管理，沟通渠道畅通。而在我国，管理的不规范往往 首先体现在公司的组织结构上，集中表现为部门的缺失和管理的交叉上。我国的软件公司，大部分规模较小，开发人员超过100人的公司很少。在印度，软件公司 无论大小，都是“麻雀虽小，五脏俱全”，绝不会因为公司的规模大小而改变合理的组织结构。因此笔者认为，国内的软件企业要想有效地保障产品质量，首先就要 在构架合理的组织结构上下功夫，这就如同盖高楼首先要打好地基一样，地基不打牢，结构不合理，其他方面再下功夫也是徒劳。有人说，因为国内软件企业规模 小，所以造成结构设置的欠缺，但笔者认为恰恰是因为没有建立一个规范化的组织结构，才会使软件产品质量不保，进而严重影响了企业的发展扩大。

2.项目计划

凡事预则立，不预则废。这里的“预”就是指计划。对于软件企业，计划的重要性是不言而喻的。让我们先看看 A 公司的项目计划是如何制定的:在 A 公司，项 目开始之前必须先估计项目的规模（以代码行数来衡量）;然后制定项目计划。通常时间为2～3周，已知的最长有5周。EM 负责制定项目 EWP(Engineer Work Paper)，其中定义了项目需要的人力和计算机资源，由相关部门同意，并报研发总负责人批准后才能开始项目。

项目的正式开始时间由项目组的 Kickoff Meeting 算起，Closeout Meeting 结束。

大概很多人都听过这样一句话：“计划赶不上变化”。这种“变化”对某些行业而言也许并不会产生太大的影响，但对于软件企业而言，却会给软件产品的质量 保证带来严重的负面影响。为什么会造成这种“计划赶不上变化”的现象？究其原因，笔者认为主要是因为对计划的重视程度不够，计划过于笼统、粗糙导致可执行 性太差，再加上一些人为因素的影响，必然会产生这样的后果。

如果我们的软件企业都能像 A 公司这样，在作计划时能考虑到每一个细节，不是仓促做出决定，而是由所有的相关部门共同对产品计划进行反复研究、制定、讨 论、修改，最终形成一套系统、严密、具有很强的可执行性的计划。计划一旦形成，就严格按照计划去执行，而不受某个人、某件事的影响，那么就不仅能够减少大 量资源的浪费，产品的质量也得到了保障。

因此，对计划的高度重视、周密制定、严格执行是企业有效保障产品质量的一个重要环节。

3.项目管理

当企业构架了合理的组织结构并制定了缜密的计划后，就进入了产品的开发阶段。在这个阶段中，项目管理起了重要作用，它所涉及的环节相当具体复杂，下面先介绍一下 A 公司在项目管理上的具体细节：

(1)开发阶段和项目周期

开发阶段比较明显，注重各阶段应完成的功能，对本阶段应完成的工作不能留到下一阶段。

(2)流程

* A 公司对流程比对项目更重视。

* 软件开发流程非常规范和系统化，其流程的可执行性很高，并且能在实践过程中不断改进。A 公司的流程已覆盖到了一个项目研发的所有方面，包括从最开始的意向到最后软件的版本发布(release)，都有相应的流程规定，基本上已形成一种工业化的软件开发。

* 人和流程是保证项目成功的两个最关键因素。由好的人按好的流程进行项目开发，才能最大限度地保证项目的成功。一个好的流程可以保证差的人做出来的东西不至于太差，但不能确保做出精品。通过流程可以实现一种规范化、流水线化、工业化的软件开发。

(3)计划

1) 计划详细、周到。

2) 流程中明确定义开发阶段。

3) 每个阶段都列出了该阶段的各项活动，并详细描述每项活动的属性:

* 进入条件，输入;

* 验证方法;

* 结束条件，输出。

4)每个阶段结束都要召开阶段结束会议。前一个阶段结束才能进入下一阶段。

5)计划中每个活动都比较具体，每个活动的时间以天（半天）为单位。计划包括了开展质量控制活动的时间。

(4)Review

按印度公司流程，一般把 Review 和测试作为保证软件质量两个主要手段。测试的重要性就不需说明了，而 Review 则是一个非常简单有效并能尽早发 现软件中错误的方法，可以说，任何交付物都要经 Review 后才能进行基线化。目前 A 公司有很详细全面、可执行性很高的 Review 流程和各种交付物的 Review Checklist。

在印度软件企业，现有这么一句口号：凡事有计划，凡事必 review。

(5)QA

QC（质量经理）作为质量保证部门（SQA）的代表，监督和保证项目的进展遵循 QMS 各项流程和模板，并且收集项目中发现的一些问题和解决方法以优化流程。

(6)度量数据

CMM 中比较强调用数据说话，对项目过程中基本上所有的数据都会有记录，最后把收集的数据提交质量保证部门进行分析，以改进流程。A 公司的项目经理和 质量经理很重视项目中的数据收集，包括各种 Review 数据、测试数据以及项目组员每天的活动数据等。项目经理也要维护一个项目档案，在这个项目档案中可 以说包含了项目开发过程中所有的产出、开发活动、管理活动等的记录。可以这么说，有了这个项目档案，你就可以完全了解这个项目的开发过程。

(7)团队精神

印度公司都比较强调团队精神、合作精神，应该说，其流程本质上就要求员工之间的互相协调和理解。相对而言，印度员工的合作精神和协调精神都比我国员工要好得多。

(8)培训

印度公司都比较强调培训，一般有专门的培训部门进行协调。在新员工进入公司后都会有公司流程和其他一些公司普遍章程的培训，以保证员工对流程的理解和 执行。对于具体项目，项目经理在制定项目计划时就会在项目计划中提出所有的培训需求，包括技术上的培训和其他所需的培训。

(9)配置管理

在项目正式开展前，项目经理就要制定配置管理计划，并且指定配置管理员建立起配置管理库，按配置流程严格进行配置管理。在配置流程中也详细提供了对更改的控制，没有经过批准的更改请求是

绝对不能进行的。

(10)记录

记录及时、充分、比较准确。这些记录包括:重要的邮件、会议纪要、审核记录、缺陷报告、测试报告。

1)与客户和其他项目组的所有往来必须邮件记录。

2)对所有的活动都有一个跟踪落实的过程，比如对所有的 Review 记录和更改请求都会有一个状态标识，标识其当前状态，通过跟踪其状态来监督其落实。

3)对所有的活动，包括对文档和代码的更改都会有一个历史记录。

4)记录比较准确、比较客观。

5)许多记录都是通过定量的数值记录，强调以数据说话（CMM4级的重点就是量化管理）。

以上是 A 公司在项目管理中所涉及到的一些主要环节，很值得国内的软件企业在制定项目管理规划时借鉴。除此之外，我国的软件企业在产品开发管理的过程中，还易出现以下几个方面的问题:

1)需求说明差─需求不清楚、不完整、太概括、或者不可测试，都会造成问题。

2)不切实际的时间表─如果在很短的时间里要求做许多事，出现错误是不可避免的。

3)测试不充分─只能根据客户意见或系统崩溃来判断系统的质量。

4)不断增加功能─在开发正在进行过程中要求增加许多新的功能。这是常见的问题。

5)交流问题─如果开发人员对客户的要求不了解，或者客户由不恰当的期望，必然会导致错误。

这些问题的出现，将会对软件质量的保证产生不良影响，针对上述问题并结合 A 公司在项目管理方面的经验，笔者提出一些相应的解决方法，以供参考：

1)可靠的需求─应当有一个经各方一致同意的、清楚的、完整的、详细的、整体的、可实现的、可测试的需求。为帮助确定需求，可使用模型 (prototypes)。

2）合理的时间表———为计划、设计、测试、改错、再测试、变更、以及编制文档留出足够的时间。不应使用突击的办法来完成项目。

3)适当测试─尽早开始测试；每次改错或变更后，都应重新测试。项目计划中要为测试和改错留出足够时间。

4)尽可能坚持最初的需求─一旦开发工作开始，要准备防止修改需求和新增功能，要说明这样做的后果。如果必须进行变更，必须在时间表上有相应的反映。 如果可能，在设计阶段使用快速的模型，以便使客户了解将会得到的东西。这将会使他们对他们的需求有较高的信心，减少以后的变更。

5)沟通———在适当时机进行预排和检查;充分利用团组通信工具—电子邮件、群件(groupware)、网络故障跟踪工具、变更管理工具、以及因特网 的功能。要确保文件是可用的和最新的。优选电子版文档，避免纸介质文档:进行远距离联合作业及协作;尽早使用模型，使客户的预想表达清楚。

4.PC（项目经理）

项目经理是项目成败的关键人物，其对项目的成败负主要责任。因此在这里将项目经理的有关内容单独提出，以 A 公司为例详细说明 PC 在整个产品研发过程中所扮演的角色，希望能对国内软件企业的项目经理有所启示。

（1)在 A 公司，按流程在一个项目正式开展之前，项目经理需要完成：

* 项目计划(Project Plan):在此描述整个项目所应完成的交付物、项目时间表、培训需求、资源需求、质量保证计划以及过程和交付物的定量质量目标等。

* 项目配置管理计划(Project Configuration Plan):在此指定配置管理员，描述项目配置项列表、配置管理库、版本管理计划等等。

*项目过程手册(Process Handbook):在此描述本项目所采取的裁剪后的生命周期模型和流程。

（2）在项目开发过程中，项目经理需非常了解项目进度，进行工作任务细化、具体计划和安排项目成员工作任务等工作。对突发事件项目经理需能及时合理地进行协调。

（3)总的说来，PC 安排工作有这么几个特点:

a.PC 对软件开发具有丰富的经验，了解软件开发的普遍流程，了解各个阶段所需完成的工作，这是安排好项目组成员工作的前提，在 A 公司对 PC 的整体素质要求非常高。

b.在项目正式开展前，PC 准备项目计划文档，在项目计划中包含了项目进度时间表，但此时间表比较粗，只能给出各个阶段和各个子阶段的起始结束日期。 对各个阶段和各个子阶段的详细工作安排和各项工作责任人只能在项目开展工程中根据项目实际情况进行安排，一般是在每周项目组例会上进行本周详细工作安排。

c.PC 对工作安排往往精确到天，有时甚至精确到小时，要做到这一点，需要:

* PC 对本项目进展非常了解。了解渠道通常是每周组员的状态报告和直接与组员接触了解，这也需项目组成员能如实汇报工作。

* 对现阶段或本周所需完成的工作非常了解。知道现在该做什么，并且能把各项工作进行合理细致地划分，因为各个分解的工作比较细致，因此能相对精确地评估出这些工作完成所需的时间。

* PC 对项目组员的能力比较了解，安排工作时能做到有的放矢。当安排的员工对工作不熟悉时，会指定相应的组员进行协助。

* PC 对组员的工作安排都比较细致饱满。一般不会出现有些员工有事干，有些员工没事干的情况，当出现这种情况或员工提前完成工作时，PC 就会进行相应的协调。

d.PC 在项目组例会上的工作安排一般只限于本周或甚至是过后的二、三天，一般不会太长，对长时间工作的安排容易失去精确并且不易控制。相对而言，短 时间的工作安排就比较精确而且容易控制，并且能不断根据完成的工作进行调整。当然，这就要求 PC 能根据项目计划中的项目时间表进行整体进度的把握。

e.项目组例会一般一周一次（时间不能太长），但必要时（如组员工作已完成或其他事情），也可在中途召开项目会议进行工作安排，一般时间都比较短（十几分钟左右，一般不超过半小时，以免浪费时间），总之，当 PC 觉得需要时，就会召开项目会议。

f.当项目组出现意外事件或影响项目团结的事件时，PC 能及时合理协调，解决项目组内的不和谐气氛。

g.PC 善于鼓励手下，发挥员工的潜能，PC 往往会赞扬很好地完成了工作的组员。

从上面可以看出，对 PC 的能力（包括技术和管理能力）要求是非常高的，我国的软件企业往往只重视 PC 的技术能力，但事实上，一个只精通技术的人往往不 能成为一个合格的领导者， 笔者认为对 PC 而言，首先要求他能够比他的下属看得更远一步，顺利时不盲目乐观，遇到挫折时不茫然失措，使整个团队始终保持高昂的士气。

总结

以上结合印度软件项目管理的经验总结了一些我国软件质量保障应注意的问题。曾有人提出:这样一味地学习模仿，民族软件工业没有多大希望。但笔者认为， 在这个问题上不妨采取“拿来主义”的办法，对于好的，事实证明是成功的经验，首先是“占有”，然后才是“挑选”和“创新”。如果能把印度的管理经验真正领 会并付诸实践，相信对我们的民族软件工业一定会起到积极的推动作用。

没有测试怎么敏捷？

Bug 是什么？

警告：如果你有挑剔的眼光，并且你已经发现了写软件的唯一正确方法，你可能会感到被挑战。

新来的开发人员提交了代码，但是有 bug。他们搞砸了代码，一个链接显示为明亮的、刺眼的红色，而不是公司要求的柔和的蓝色。你找到她，坐在一起，开始教导她测试的重要性，这时接到市场部的祝贺——因为最近一次发布，销售额上涨了了50%。

你知道唯一的修改是链接的颜色。

代码真的有 bug？你还要诚实的把它回退？

更重要的是，这也是许多人搞错的问题：你从这件事得到什么教训？

为什么要测试

我们使用敏捷开发是希望改善开发过程，更好的分享信息，特别是更加方便频繁的获得反馈（这是一个反复发生的流程）。有趣的是所有的敏捷开发理论都要求 使用者调整敏捷开发理论来满足个性的需求。我之前说过并且现在重复一下：你不能敏捷开发除非你自身是敏捷的。所以一些敏捷团队没有固定的迭代过程。其余的 （大多数？）不搞结对编程。代码审查成了 tricky bits（译注：不了解的词汇）和初级程序员。然而，尽管方法不同，这些公司往往做的很好。

但是他们都没有想要不做测试，他们没有想过。

在我们的“红色链接”例子中，如果进行了测试，那么就不会有50%的销售额增长。

我们大多数人很早就接触测试并且认为它不错。然后我们了解了 TDD 并且意识到这就是测试的涅磐。 FIT testing 被那些 FIT 测试咨询机构过分的夸大了。现在 BDD 的情况也是如此：一些人兴奋的不得了，另一些人不以为意。我想这不过是另一次炒作，不是么？

但是测试是为了什么？从技术的角度来看，我们也许会争论说测试就是确保软件按我们希望的方式运行。但是这是最重要的事吗？请铭记一点就是：在一咕哝行代码里，所有软件都有 bugs，所有的！

相反的，我认为更好的说法是所有软件都有无法预料的 behavior.我们写测试是因为我们希望软件会按我们期望的那样运行。但是反过来，如果用户按 我们希望他们做的方式去操作不是更好吗？你可以建立一个更好的捕鼠器，但是不能确保它们会钻进去。如果有些是从 Digg 或者一些技术灾害学习来的，那就会 是这样：客户将会按他们该死的很乐意的方式去操作，无视你的软件是否是“正常工作”、你咨询的专家们和你举行了多少次讨论组。

与其说软件测试是客户行为的一种代理，还不如我们静下来为软件的客户好好想一会儿。

意外列表

考虑上面的“闪亮红色链接”的例子，再问一次：什么是 bug？在那个例子中（不是随机选取的），可以简单说是一个软件 bug，仅仅因为按意料外的行为。在这个例子中，是50%的销售增长。

现在，bug 不再总是坏事情！我们可以用“意料外的行为”来思考——有时好，有时坏。

那怎么知道到底是好还是坏呢？

做一个意外列表。网站的500错误是坏的，而302呢？难说。也许你希望 RAM 使用率低于一定水平，或不像看到明显的销售额下降。也可能希望响应时间永远不超过$x 毫秒。

列出所有明确不合预期的事件（例如：Facebook 没有“like”按钮不属此列）并对所有这些行为进行监控。每当你修改或添加一些技术，请再次仔细检查你的不合预期列表。它们是不是最新的？你是否需要修改什么？

一些不合预期事情是可逆的（销售下降）且替代方法是好的。因此，也要监控这些。或许当一个版本提升了10%的响应速度时你想得到通知。

那然后呢？

嗯，这是伟大的但绝对不可能代替测试。你提交了两周的版本，内存消耗猛增，你疯狂地交换以致现在需要检查300行的文件差异。你很难在最佳时间找到内 存泄露问题并且正常的测试经常遗漏它们（除了检查 Test::LeakTrace），但现在你要回滚一个巨大的修改然后检查3000行代码来找出你的问 题。

因此你不必那么多。相反，你可以转换为不间断部署。在这个模型下，在准备好的前提下，可以把代码推送到产品里面。当然，假如你把它推送到盒子里面的话 也很好，观察它，推送它到集群里面，观察它，然后把它推送到所有的服务器。通过多方面的监控，不寻常的情况常常出现的很快。并且你的内存泄露是30行的 diff 而不是3000行的 diff.

你想处理那种情况呢？

（理所当然，我用内存泄露作为例子，但是这是一件经常花很长时间才显露出来的，而且我也懒得去改变这个例子。假装我过去写了5%的增加在404.）

在我的这个经验中，顾客对一些小异常简直就是很宽容，像大部分意料之外的行为都是一些诸如“图片不能显示”或者“这些查询结果排序是不正确的。”这些 并不会趋向于导致灾难性。事实上，许多次这种异常的行为是被忽视的。就不良的事情而言，大部分时间意料之外的的行为将转变为中性或者不良。但是有时候他们 也会转变成为好的意想不到的行为。假如你不试一试的话是永远不会知道的。

正如你所预料的，这个技术在“A/B testing”这本书中运用得很好。如果你有勇气查找非预期的行为而不是 bugs 的话，这本书将是下一个合符逻辑的计划。

注意：以上并没有排除编写测试（用例）。我看过“监控不合预期的事件”战略工作得非常好也坚信它可以结合软件测试进行工作。然而，测试软件是不同的，它更依赖客户行为而不是严格规范。因此，这篇博文的标题实际上是有点不对，它只是一个看待测试的不同想法。

而这正是这篇博文真正最有趣的想法：客户的行为比你应用的行为更重要。

原文链接：http://www.oschina.net/translate/how-to-be-agile-without-testing

如何雇佣一名专业的程序员?

如果你会问这个问题，那你肯定遇到过这种情况：全公司的人互相问来问去，解决硬件、软件还有打印机联网之类的问题，极其浪费时间;或者是，在你的操作系统崩溃时，负责设置系统的 IT 外包员工却在度假。

无论是哪种情况，总有一天你会发现，拥有一个全职的 IT 人员不再是“奢侈品”，而是“必需品”。根据拥有全球用户的小型企业 IT 管理平台公司 Spiceworks 的调查，对高科技公司来说，通常在员工发展到20名时就需要一个专业的 IT 人员;非高科技公司则通常可以等到员工数超过70名时再雇 佣专业 IT 人员。

那么，如何判断这位技术部的应聘者是否能胜任这份工作呢?我们咨询了马萨诸塞州沃尔瑟姆一家提供护理服务的公司 Care.com 的 IT 和生产运营部门高级主管 Nathan Brown，他介绍了详细的招聘流程。

应该给 IT 岗位应聘者什么样的测试?

我会在面试结束时，要求应聘者为公司配置一台电脑，并按照公司的标准安装和配置软件。这个测试可以帮助我评估应聘者是否有长远眼光和全局观，我希望我 的 IT 员工除了技术能力外，还能专注于支持维护、技术许可等公司未来可能出现的需求。如果他们对我们的现状提出改进、更新等建议，我会认为值得对他们做进 一步考察。

在雇佣第一个 IT 员工时，应该注意哪些细节?

我不会过分相信资质证明，我认为实践经验更重要。具体来说，要找一个了解服务器、交换机、路由器、防火墙和虚拟化技术(比如一台电脑同时支持两个完全 独立的用户、系统或程序等等)，并有相关工作经验的人。我会询问应聘者他们在家里用什么电脑，如果他们只说“戴尔”而不谈处理器、内存和操作系统，他们就 不是你想要找的人。

最后，寻找一个有桌面支持、网络和系统管理等全方面经验的人，了解 IT 基础设施才能更好地开展工作。我也很看重应聘者是否有各种不同的操作系统(如 Windows、Mac、Linux)和硬件方面的知识，这表明他知识面广，心态开放，并热爱技术。

非技术方面如何考核呢?

IT 人员很多时候都需要进行客户服务和关系维护。你雇佣的第一个 IT 员工既要是个技术极客，也要能与 C??EO 交流顺畅，能向他解释复杂的技术问题，帮他做出正确决策。如果你不确定应聘者是否能与所有性格类型的人相处，那么最好不要聘用他。

寻找业务测试的精髓

最近，我们经常听到测试转型，减少人力密集型的业务测试的比例，一线业务测试缺少发展、无法晋升等等问题。似乎，随着互联网的不断发展，业务测试成为 一个看不到前途的职业，那么，是否真的如此呢？我作为一名在一线业务做测试管理有些年头的过来人，并不这么认为，恰恰相反，我认为业务测试很重要，它是一 个不可能消失的行业。

一、业务测试为什么存在？我们按照现在对业务测试的理解，画一张图：

这张图示意我们，测试工具是最能解决测试效率的手段，专项技术是测试具有含金量的发展技术，例如性能、安全、自动化等等，而业务测试属于劳动密集型的 岗位，可以被淘汰。金字塔结构是大多数人的理解，但事实上，很多一线业务测试的管理者，都有一个真实的感受：工具再强大、转向技术再牛 b，在业务层面，真 正出力的还是一线的业务测试。那么，这个图示错误了吗？也不是！现在的业务测试，确实是劳动密集型岗位。

如果我们转换下思维，将上图做些改变：

根据现实情况，我们综合工具、技能在实际工作中发挥的作用，就会发现，其实工具、专项技术其实是相互驱动的往前发展，这很像太极中的阴阳平衡，而业务 测试，是工具、专项技术进行有效实践的载体，这个载体中发挥最大作用的便是业务测试。而业务各有不同，即便是业务中的产品生命周期不

同，所对应的测试方 法、测试投入都不相同，所以，这些不同，就形成了太极外围的八卦，各有想通，却各不相同。

从这幅图看，测试工具、技术要发展，必须基于业务，而业务测试要发展，除了要工具、技术的支持外，更重要的是要适应于业务的需要。业务测试和业务的关系，即是水与舟的关系，大海中行小舟，小河中放大船，都是不合场景，无法适应的。

从这个角度说，取合适的资源，用合适的方法，为业务保证质量，要做这样的业务测试，不简单。如果还说业务测试没发展，那没发展的不是业务测试，而是这个在做业务测试的同学的脑袋。

二、用什么来衡量业务测试？

在阿里，测试工程师的发展有自己的诠释：

P4	处于学习成长阶段，需要对目标进行随时的沟通和指导
P5	对相应的流程政策有基本了解，能独立完成项目，遇有较复杂的情况能提出问题，并找到帮助
P6	对流程政策理解深刻，能够和经理一起讨论本岗位的产出和任务，并对经理具备一定的影响力，对于复杂问题有自己的见解，对于问题的识别能进行优先级设置，善于找到相关资源，对于问题有创新的点，可在一定的指导下独立完成项目，能指导新员工
P7	对自己从事的职业具备一定的前瞻性的了解，在某个方面有独道的优势和能力，在此方面对公司有影响力，对于复杂问题的解决有自己的见解，对优先级尤其有影响力，善于寻找资源解决问题，因为对于工作有创新的想法，可独立领导跨部门的项目，能培训和指导新员工
P8	对于公司内外及业界的相关资源和水平比较了解，开始参与公司相关战略，对部门管理层在某个领域的判断能力产生影响，对事物和复杂问题更有影响力
P9	在公司内部被认为是某一方面的专家，对公司某一方面的规划和未来产生影响，对本领域的思想和研究在公司具备一定的影响力

目前，业务测试的同学普遍层级覆盖在 P5、P6，是什么阻碍了业务测试的发展？在我看来，当业务测试的同学对如何在业务中进行测试这件事驾轻就熟的时候，就到了该考虑自身发展的时候。

所谓驾轻就熟，并不是作为业务测试的价值，这恰恰只能说明，你到了一个发展的起点，而非终点。驾轻就熟，从一定意义上来说，是在照搬别人的做法，达到了作为一名业务测试最基本的工作能力。如果用阿里的层级表示，那就是 P5、P6的水平。

业务测试要做专，有两点必须把握：为产品质量负责的能力、为产品体验负责的能力。

对产品质量负责的能力：这是一项能够根据产品发展阶段、项目组成员能力结构来定制合适质量保证手段的能力。基本的测试流程、基本的测试方法可以照搬其 他团队的经验，但如何确保测试工作真正有效，和产品所处的发展阶段、项目组成员的能力非常有关，一个刚起步的创业产品和一个已经处于稳定状态的维护产品， 两者采取同样的项目流程、同样的测试方法，你能确保给予最好的质量保证结果吗？依据以往的经验，我的答复会是，有办法做到，但付出的代价很大。更重要的 是，业务测试习惯性的照做思维，导致他们没有思考为什么要加班，为什么如此努力仍然要在上线时心惊胆战？

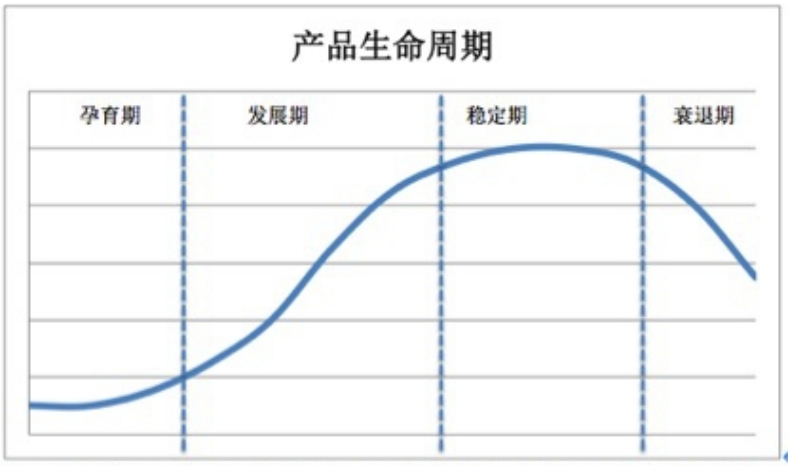
为产品体验负责的能力：这是一项作为业务测试最重要的能力。对业务精专，对产品的发展具有一定影响力，对产品的问题有自己独到的见解和解决能力，这是 业务测试不同于其他转向测试的能力。为什么我们的业务测试总停滞在测试阶段？我们好像对产品很重要，但似乎又不那么重要？我们会对产品设计的体验问题提出 自己的见解，但我们总不对体验问题负责。因为，看上去，体验与质量无关。但是否真的无关呢？如果你认可体验在产品生命中占用重要位置，那么作为业务测试， 就应该对产品体验负责。

如果业务测试先改变自己，那么我们完全可以从三角的最低端资源密集型产业，变化为太极八卦图的特质八卦，即一套适合业务的测试架构，一套能提升产品质量和产品体验的测试体系。

三、业务测试与开发的黄金配比

在业务测试团队，我们常听到的测试效率，是用开发测试比来衡量。对这项简单的衡量标准，在产品发展的稳定阶段，非常适用，但不完全精确。举个简单的例 子，当稳定团队的开发测试比达到10：1的时候，我们为这个测试团队的效率欢呼。但这10：1的背后，是真实的质量提升？还是测试工作的转移？当测试工作 用开发自测来代替的时候，这10：1仅仅是人员机构的重组，而非业务测试效率的提升。当然，在良好的自测工具支持，完善的质量数据衡量前提下，开发自测效 率更高，这是毋庸置疑的。只是，这个业绩的主角是开发，而非业务测试。

什么才是业务测试与开发的黄金配比呢？没有标准答案。我们对照产品生命周期来看测试：



测试要点	如何保证上线质量？	如何做到快速响应？	如何减少资源投入？	如何为产品升级做好技术沉淀？
测试重心	需求分析 测试建模	自动化回归 性能测试	测试统计分析，减少人力投入后的质量监控	核心功能继承

以上针对产品生命周期和测试对比的分析，是我个人的观点，可能有不足，但从整体上，可以说明，一个产品在发展的不同阶段，测试的投入和重点是不同的， 所对应的业务测试团队组成、测试工程师的技能需求也不相同，所以，每个业务团队要得到统一的测试开发黄金比，结果可能是牺牲团队效率，或者是个人成长空 间。

综上所属，业务测试的精髓到底是什么？我的回答是，分析测试对象，用测试所具备的技能，寻找合适的质量保证方案，给予有效、实时的质量反馈。概念仍然 没有变，业务测试要发展，最大的区别就是要做到经验的继承而不是复制，技术的精用而不是泛用，个性化的测试服务就是一个业务测试的发展之道。

建立测试管理与评判体系六大过程

软件测试过程模型或软件测试生命周期模型为我们提供了软件测试的流程和方法，为测试过程管理提供了依据。由于测试过程管理牵涉的范围非常广泛，包括过程定义、人力资源管理、风险管理等，我们仅从前面介绍的软件测试过程模型来介绍软件测试过程管理的思想。

现代软件测试过程管理不是仅锁定在测试阶段，软件测试过程管理在各个阶段的具体内容是不同的，但在每个阶段，测试任务的最终完成都要经过从计划、设 计、执行到结果分析、总结等一系列相同步骤，这构成软件测试的一个基本过程。通过软件测试过程管理我们要尽量达到测试成本最小化、测试流程和测试内容完备 化、测试手段可行化和测试结果实用化的理想目标。

软件测试是软件工程中的一个子过程，为使软件测试工作系统化、工程化，必须合理地进行测试过程管理，包括签订第三方独立测试合同、制订测试计划、组织 项目人员、建立项目环境、监控项目进展等等。软件测试过程管理主要集中在软件测试项目启动、测试计划制定、测试用例设计、测试执行、测试结果审查和分析， 以及如何开发或使用测试过程管理工具。概括起来包括如下基本内容：

(1) 测试项目启动

首先要确定项目组长，只有把项目组长确定下来，就可以组建整个测试小组，并可以和开发等部门开展工作。接着参加有关项目计划、分析和设计的会议，获得必要的需求分析、系统设计文档，以及相关产品/技术知识的培训和转移。

(2) 制定测试计划

确定测试范围、测试策略和测试方法，以及对风险、日程表、资源等进行分析和估计。

(3) 测试设计和测试开发

制订测试的技术方案、设计测试用例、选择测试工具、写测试脚本等。测试用例设计要事先做好各项准备，才开始进行，最后还要让其他部门审查测试用例。

(4) 测试实施和执行

建立或设置相关的测试环境，准备测试数据，执行测试用例，对发现的软件缺陷进行报告、分析、跟踪等。测试执行没有很高的技术性，但是测试的基础，直接关系到测试的可靠性、客观性和准确性。

(5) 测试结果的审查和分析

当测试执行结束后，对测试结果要进行整体或综合分析，以确定软件产品质量的当前状态，为产品的改进或发布提供数据和依据。从管理来讲，要做好测试结果的审查和分析会议，以及做好测试报告或质量报告写作、审查。

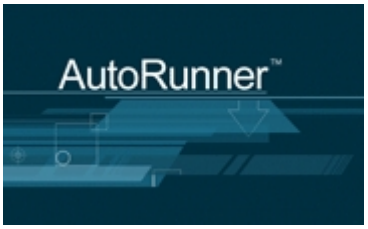
泽众软件工具使用技术支持

电话：021-61079698

Email：sales@spasvo.com

QQ：1404189128

MSN：spasvo_support@hotmail.com

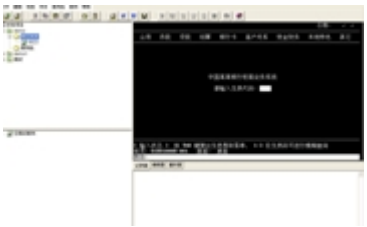
	产品租用		
	下载	在线申请	详细
	AutoRunner 是一款自动化测试工具。AutoRunner 可以用来执行重复的手工测试。主要用于：功能测试、回归测试的自动化。它采用数据驱动和参数化的理念，通过录制用户对被测系统的操作，生成自动化脚本，然后让计算机执行自动化脚本，达到提高测试效率，降低人工测试成本。		

	在线体验		产品租用	
	企业版	免费版	在线申请	详情
	TestCenter 是一款功能强大的测试管理工具，它实现了：测试需求管理、测试用例管理、测试业务组件管理、测试计划管理、测试执行、测试结果日志察看、测试结果分析、缺陷管理，并且支持测试需求和测试用例之间的关联关系，可以通过测试需求索引测试用例。			

其他测试工具

Precise Project Management

Terminal AutoRunner



PerformanceRunner

有关培训、产品购买及试用授权方法等事宜

电话：021-61079698

Email： sales@spasvo.com

QQ：1404189128

MSN： jennyding0829@hotmail.com

SPASVO