泽众软件测试网
电子期刊

SPASVO

第二十一期
2013年9月

上海泽众软件电子期刊

2013 年 9 月 第二十一期

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

泽众软件——专业软件测试门户
2013 年 9 月 第二十一期 电子杂志期刊

软件测试工程师的十年经验之谈

种瓜得瓜，种豆得豆的时代已经过去了，如今种瓜能结豆，种豆也能结瓜，就好比学法律的能当翻译，学英语的能做 IT。可是要在自己的非专业中脱颖而出， 需要付出比专业人更多的努力。下面是微软测试工程师郑洪流，以自己的故事为您讲述如何“在测试中学习测试”：我是作为软件测试工程师（Software Test Engineer）的头衔被招进微软的，工作的主要任务是软件测试。但在进入微软之前，我是既无软件开发，也无软件测试的经历。为应付面谈，突击看了一些 测试的书。但如何去做测试工作，可以说是两眼一抹黑，纯粹是赶着鸭子上架。在这种情况下，微软提供的许多帮助新员工的措施，就对我非常有用了。

首先，技能指导手把手地教给我们如何写测试案例、测试计划、机器设置、软件安装、发现和输入缺陷等。这一过程其实很短，也就是两三天。但是，帮助确实 很大，大大地缩短了我们新人进入工作状态的时间。我印象最深的是我的第一个测试项目。那是一个微软为网络公司提供的网络组合软件 MCIS，该软件包括操作 系统（NT 4.0）、数据库（SQL Server 6.5）、网页服务器（IIS）、电子邮件服务器（Exchange）、电子邮件用户界面（Web Mail）以及专门为管理这些软件开发的管理系统（MCIS）。我的任务是每天新版本一出来，就将整个系统重新安装，设置一次，然后测试诸如发/收电子邮 件；网页设置及访问；会员账户产生，管理等基本功能。这些对于一个刚接触网络的人是很大的挑战。但技能指导将整个过程的上百步过程一一列出，共长达满满 五、六页纸，我们只要按照步骤一步一步地去实现。这样我及其他几个新职工在很快的时间内即投入了日常测试工作。

其次，微软许多网络上的材料，给新职工提供了非常方便的学习条件。这些材料包括各种过去的培训材料、各种工作需要的范本、各种资深职工撰写的理论或体 会文章，以及各项目的网站等。有的组还专门组织有经验的职工为新职工编写新人须知，这些须知常包括如何设置测试机器，如何安装常用诸如缺陷跟踪、测试案例 管理等软件、如何设置软件开发环境，在哪里找到常用的信息等。包括我自己在内，很多员工的第一个缺陷报告、第一份测试计划等，都是从别人的范例、范本或介 绍材料中学来的。在我进入微软不久，即被安排测试一个组合软件的局部化测试。当时主要是测试该组合软件在德语、日语和朝鲜语上的工作性能。我所撰写的测试 计划、测试案例等都是根据公司内部的局部化网站的有关资料完成的。许多诸如哪些是需要重点测试的字串，如何测试快捷键（shortcut），适合局部化的 用户界面的基本要求等，都可以直接从网站上抄录下来。所以，只要自己多注意学习，多注意其他人是如何工作的，你就能利用这些材料一步一步走入正轨。

经理及同事的帮助，是最快和有效的学习途径。员工每周都要和经理至少汇报一次工作，这种汇报一般是书面和面谈相结合。员工将一周的工作进度及下一步的 计划一一列出，逐一和经理讨论汇报。这通常也是员工将自己遇到的问题和经理讨论的时间。一般经理都是经验丰富、能力出众的资深职员，帮助手下员工是他们分 内应尽的义务。所以，这是一个很好的学习机会。员工所要做的即是将自己的问题诚实地提出来，寻求帮助。当然，一旦你得到合理的建议，应立刻付诸实施，并让 经理知道你的进程。如果建议确实帮助你解决了问题，应该及时感谢经理。这样容易建立一个帮助、被帮助的良性循环。经理的帮助，由于他的时间的限制，往往只 是方法、方向性的。更多的帮助是来自许多资深职员，他们的优势是及时，你可以随时和他们商讨；贴切，他们往往处理过和你一样的问题， 因为他们也在做和你一样的工作。我刚开始测试工作时曾经遇到过一个问题：用户界面始终和数据库联不上。自己当时没有经验，不知道是否是测试错误还是软件缺 陷。最后请一位资深测试人员帮助，他很快发现到问题的症结：我使用了一组数字作为机器名，而 SQL Server 6.5在寻找该机器时，自动以为这是 IP 地址，所以不可能找到。

各种形式的培训是进修的最佳途径。微软的培训种类包括：公司内部的课堂式的培训、公司外部的课堂式的培训、网上培训、录像或磁带式的培训等。在刚进入 微软时，我重点选择参加了一系列的关于测试的培训。在不到半年的时间里，通过各种培训途径，学习了测试计划、案例的撰写、全局化测试、局部化测试、API 测试、性能测试等。也通过网上材料和公司图书馆的测试书籍，系统地学习了有关测试的一些基本理念。在工作的同时学习的优势是，学到的东西可以直接在实践中 应用。从而大大地增加了对所学内容的理解和记忆。

为什么要进行需求管理？

本文介绍了需求管理的必要性，并介绍了控制需求渐变的一些方法。

软件需求是整个软件项目的最关键的一个输入，和传统的生产企业相比较，软件的需求具有模糊性、不确定性、变化性和主观性的特点，他不像生产汽车、电脑 等硬件的需求，是有形的、客观的、可描述的、可检测的，软件需求是软件项目最难把握的问题，他的复杂性体现在以下方面：

需求的描述问题

笔者曾经被紧急委派主管一个已经进入了编码后期阶段的项目，该项目已经换过2次项目经理了，这是第3次更换项目经理，用户方的 IT 部经理找笔者抱 怨："我已经是第3次来给你们讲补货申请的处理规则了！"。我只能表示抱歉，因为我无法找到原来的需求描述，这是一个变更的需求，前任的项目经理讲他只是 将当时与用户交流的需求记到2页草稿纸上，不幸的是，那2页珍贵的手稿现在已经找不到了！更不幸的是，该 IT 部经理是在转述业务部门的需求，当软件开发完 毕后，业务部门讲"这不是我们最初给 IT 部反映的需求，我们说的不是这样的！"。缺少正式的完整的需求文档浪费了大量的人力物力，但是有了需求文档又出现 了新的问题。曾经有多个项目经理向我抱怨，在用户方进行的需求评审会完全是走形式，因为用户根本不去听他读那上百页的需求文档。不同层次的客户（用户）关 心的问题是不一样的，想要每个客户都成为需求专家是不现实的。

需求的完备程度问题

需求如何做到没有遗漏？如何准确划定系统的范围？这确实是一个两难问题，稍微大一点的系统要想穷举需求几乎是不可能的，每次开需求评审会时，总会冒出 新的需求，以至于系统没有一个准确的范围界定。即使是这样，系统还是要开发，没办法，系统的范围还要硬性的划定一个，从而建立一个基线。

需求开发的工期问题

在需求上花费了大量的时间（而不是人*工时，因为需求阶段人多了也没有作用），客户、软件公司是否能够忍受？为了确保需求的正确性，完备性，项目经理 往往坚持要在需求阶段花费大量的时间，但是客户与公司的高层领导却会为项目迟迟看不到实际可运行的软件担心不已！他们往往会逼迫项目组尽快往前推进，而项 目组的成员往往也会为系统复杂的善变的需求折腾的筋疲力尽，他们也希望尽快结束此阶段。

需求的细致程度问题

需求到底描述到多细，才算可以结束了？仁者见仁，智者见智，并没有定论，如果时间允许，要想细总可以细下去的。但是，需求的周期越长，可能的变化越 多，对设计的限制越严格，对需求的共性提取要求越高，所以只要大家（客户、用户、需求分析人员、设计人员、测试人员）认为描述清楚了，就可以进入设计阶段 了。

需求的变化问题

在软件开发过程中如果只有一条真理的话，那一定是：需求的变化是永恒的，需求不可能是完备的。

软件开发的过程实际上是同变化做斗争的过程，需求的变更不一定是坏事，也有可能是好事，是商业机会，对市场敏感的人可以从需求的变化中发现市场机会。

需求变化的原因很多，如：

一开始没有识别全，需要增加需求；

业务发生了变化，需求必须变化；

需求错误；

需求不清楚；

需求的变化问题是每个开发人员、每个项目经理都遇到的问题，也是最头痛的问题，一旦发生了需求变化，你不得不来修改你的设计、重写你的代码、修改你的 测试用例、调整你的项目计划等等，需求的变化好比是万恶之源，为项目的正常的进展带来不尽的麻烦，怎么办？管理它！使需求在受控的状态下发生变化，而不是 随意变化，需求管理就是要按照标准的流程来控制需求的变化。

难题随之而来，需求中的变化一般不是突发的革命性的变化，最常见的是"项目需求的渐变"（Project Scope Creep）问题，这种渐变很可能是客户与开发方都没有意识到的，当达到一定层度时，双方才蓦然回首，发现已经物是人非，换了一番天地。 控制需求渐变需要注意以下几点：

（1）需求一定要与投入有显示的联系，否则如果需求变更的成本由开发方来承担，则项目需求的变更就成为必然了。人们常说世上没有免费的午餐，同样也不 应该有免费的需求变更。但是，接受需求变更目前却是软件开发商不得不咽下的苦果。所以，在项目的开始无论是开发方还是出资方都要明确这一条：需求变，软件 开发的投入也要变。

（2）需求的变更要经过出资者的认可，需求的变更引起投入的变化，所以要通过出资者的认可，这样才会对需求的变更有成本的概念，能够慎重地对待需求的 变更。笔者曾经经历过一个项目，为了避免项目的风险，我们请了用户代表全程参与了开发过程，结果此用户代表在开发过程提出了大量?quot;小"的需求变 更，当开发人员按此需求变更修改了软件时，在项目进入现场实施阶段时，却有大量的这些变更需要改回去，问题就是出在我们的项目组成员视该用户代表的需求为 圣旨，却忽略了需求是否经过了客户方真正有决策权的人员的认可。

（3）小的需求变更也要经过正规的需求管理流程，否则会积少成多。在实践中，人们往往不愿意为小的需求变更去执行正规的需求管理过程，认为降低了开发效率，浪费了时间。正式由于这种观念才使需求的渐变不可控，最终导致项目的失败。

（4）精确的需求与范围定义并不会阻止需求的变更。并非对需求定义的越细，越能避免需求的渐变，这是2个层面的问题。太细的需求定义对需求渐变没有任何效果。因为需求的变化是永恒的，并非由于需求写细了，它就不会变化了。

注意沟通的技巧。实际情况是用户、开发者都认识了到了上面的几点问题，但是由于需求的变更可能来自客户方、也可能来自开发方，作为客户他们可能不愿意 为需求的变更付出更多的投资，开发方有可能是主动的变更了需求，他们的目的可能是使软件做的更"精致"，于是作为需求管理者、项目经理需要采用各种沟通技 巧来使项目的各方各得其所。

软件需求的复用问题

笔者曾经遇到过一位领域专家，他在有20多年的领域工程经验，积累了大量的领域需求，可是在其每进行一次产品开发时，他总是感到他所理解的需求无法为 与他配合的分析人员、设计人员所接受。当我们一起来讨论这个问题的时候，共同的一个观点就是：没有对需求进行有效的管理，已经形成的需求文档没有很好的复 用。所以需求管理一个很重要的目标应是提高软件需求的复用率。

基于上述的问题，必须对需求进行管理，使需求能够真正成为软件工程和管理的基线，使软件计划、活动和工作产品同软件需求保持一致，使需求可以复用。

给刚参加工作大学生的十点建议

引言：牛在哪里？大门口

有大学毕业的朋友下乡做村干部，几乎每天都有村民到办公室来求助，总结一下出现最频繁的问题是：牛在哪里？意思是他家的牛找不到，能否帮忙找牛？为人 民服务是干部的重要职责，朋友自然尽心竭力为民服务，但久而久之，一有村民出现在门口，朋友心里就开始盘算，今天不是找牛就是找猪啊羊啊鸡啊诸如此类的， 不免烦躁与空虚，并扪心自问：难道我一个堂堂大学本科生，来这就是为了找牛吗？

确确实实，今年的大学就业相当严峻，中国社会调查所最近在上海、北京、深圳、石家庄、郑州、成都、沈阳、武汉、厦门、哈尔滨、西安等地抽取一千位应届 大学生进行调查显示，到目前为止09年大学生就业率整体为35.6%。这个数据是相当恐怖的，原因很多，我不作分析，但结果是让人担心的，中国的大学很 多，学生很多，普遍收费很高，但最后却很多人找不到工作，这种打击是很大的，特别对于出身贫困的学生，原以为可以跳出农门摆脱穷困，但现实是残酷无情的， 有的人不得不面对毕业即失业的困境，可能是因为他们不够优秀，也可能是社会所能提供的职位太少了……

十点建议

对于找到工作的大学毕业生，是值得高兴与珍惜的，接下来就要好好地努力，争取尽快能更上一层楼，FasterSoft 今年来了7个应届毕业生，他们基 本上都在公司实习半年后留下来的，另外公司也招聘了五六个新员工，对于应届毕业生或新进员工，我总会对他们抱有希望，并鼓励他们认真做人努力做事，尽快适 应环境，尽快做出贡献，总结一下，总共有十条建议：

1) 企业不是学校

这是思想上的问题，应届毕业生必须在最短的时间内认识到企业与学校的不同，比如以前是你给学校钱，学校提供环境并指导你的学业，现在是企业给你钱，你 为企业工作；企业的目标永远是追求尽可能多的利润，因此企业需要的是你具体的工作业绩与工作质量，我们每一个月都会对你进行考核，评估你的贡献，发现你的 问题，如果我们的收益低于所付出的成本，那么你处于不合格、危险的处境；企业比学校面临更多更为激烈的竞争，我们提倡“优胜劣汰”的原则，如果你不努力， 或者你努力了但没有效果，那么你将被淘汰，同时强调这里是以追求利润为目的的企业，你的上司或同事只能启发你去学习去提高，但没有时间给你上课或手把手教 你，你需要有较强的学习能力、理解能力、接受能力、实践能力。

2) 忘记过去，重新开始

很多应届毕业生或新员工喜欢沉迷在过去的光环里，甚至心高气傲，眼高手低，一直觉得自己在学校或原企业里是多么的优秀，得过这么多的赞美与褒奖，在这 里也一样会发热发光，首先这种自信是值得鼓励的，但这种想法是单纯与幼稚的，你接下来预计会面对连绵不绝的挫折与失败，因此建议你要先做好这种心理准备，忘记那些乱七八糟的优秀重新开始吧，什么优秀学生优秀干部优秀党员优秀 XX，这些可能对政府机关或事业单位有用，但对于以追求利润为目的的企业来说，它们 只不过是一个参考，如果连一个简单逻辑的算法代码都写不出来，即使你有十几个优秀，我们也不稀罕。

3) 笨鸟先飞，勤能补拙

尽管企业原则上不会招聘笨鸟，但在众多的鸟里总会有比较，一有比较就会有优劣，如果你不幸是属于非优产品，那么你要做的就是比优质产品更加地努力与勤 奋，阿蒙认为毕业后的前三年是最佳的学习周期，如果三年过后你还是感觉不到自已在技术与能力上有质的提高，那么后面的日子估计比较难过，因为三年过后理应 是你上升或发挥骨干作用的时候，而你还在学习与提高，这是多么尴尬的事情啊，即使是你换个新的环境，也一样尴尬。

4) 简单任务都做不好的人，永远没有机会做复杂以及重大的任务

如果上司分配任务给你，你个人认为很容易，不情愿去做，或者做的效果不好，那么在我们这里，你就不可能有机会去做复杂或重要的任务。这不是技术问题， 也不是能力问题，更多的是态度问题，企业不会总是有复杂或重要的任务，更多的时间是由一系列琐碎繁杂的任务所充斥，我们喜欢那些能将简单任务做到超出期望 的员工.

5) 培养优秀的职业道德和良好的工作习惯

人的一生有可能在很多的企业里呆过，特别是 IT 公司，人员的流动性是非常之高，但无论我们走到哪里，我们都应该遵循一定的职业道德，比如尊重您的上一 家公司，尊重您曾经的同事，不要为了达到目的而失去道德底线…….无论在哪里，做到什么时候，都要认真对待每一件事情与每一个人，如果明天就要离开，那么 也要将今天的事情做完做好，这是基本的道德观念。在企业里，要养成良好的工作习惯，比如遵守各种规章制度与技术保密工作，比如每天上 OA 或其他企业管理系 统，时刻关心公司的各种事情，比如养成每天上下班之时收发电子邮件，我们不想看到这种情况发生：有一天我问你“哪件事情进展得怎样了？”，你很惊讶地回答 到：“什么事啊，我忘记收邮件了”。

6）没有完美的公司，也没有完美的个人

无论是你所在的企业，还是你的老板或上司，都不可能是完美无缺的，他们总会有这样那样的不足之处，大企业可能死板并且上升空间小，小企业可能不规范并 且似乎都是老板一个人说了算，不大不小的企业可能有太多的勾心斗角并且内耗严重，市场出身的老板可能夸夸其谈但值得信任的会是百分之多少，技术出身的老板 可能过于苛刻并且缺乏幽默感与号召力……包括你自已也不可能完美，因此，你需要有心理准备，需要面对各种可能的情况，如果因为一点点不顺眼或打击，就要换 工作，那么你的一辈子可能都在换工作。

7）有技术地发表你的意见与建议

我们提倡每个人自由发表建议，但要讲究技术细节，比如如何提出才能让大家比较容易接受，在什么场合下提出才是合理的，我们尊重你的每一个建议，但如果 你的建议没有说服力没有被采纳，那么请你服从上司的安排与管理工作，而不能一意孤行固执己见，这是企业，通常情况下是你来适应这里的环境，而不是企业改变 自身的环境来适应你。

8）细节决定竞争力

我们不喜欢这样的员工：写完代码后没有经过自已的测试就提交，写完文档后没有仔细地检查就提交，在网上搜索材料然后没有经过大脑思考就胡乱拼凑成工作 结果，做事马马虎虎粗枝大叶，讲话随随便便敷衍了事……这些人在细节上做得很差，可能技术很强，但我们并不喜欢，我们喜欢精益求精，阿蒙认为国内软件企业 之间的竞争并不在于技术，而在于管理上的理念与细节上的把握，多少年来我们都喜欢制造口号，然后高呼口号，最后并没有按口号去做或去努力，什么要做中国的 微软，中国的

比尔盖茨等等，实际上他们连一个小的 MIS 都做得粗糙不堪，甚是可笑。

9) 喜欢有特点与有目标的员工

你的特点会给企业留下深刻的印象，我承认很多企业或管理者喜欢全能型的人才，但这种人毕竟是少数的，人的精力与时间都是有限的，孙悟空型的人才我倒是 很少见到，我是不喜欢什么都普通的员工，因为他太容易替代了，你需要在某一方面突出，需要与众不同，这样你才容易被注意与被关注。人之所以一直在努力着， 很多时候是目标在驱动，一个没有目标的人是毫无野心可言的，他什么都无所谓，没有思想，没有灵魂，他的人生是黯淡的，没有色彩的，这种人会给企业带来消极 的一面。

10) 人的成功70%靠个人的努力，另外的30%包括运气、天赋、环境、人脉等等。

这可能会引起很多人的不同意见，但对于我个人来说是这样的，尽管我离成功还很远，但目前到这个地步是遵循以上的比例的。不过，中国这个地方可能很奇 怪，人们成功主要靠的东西都是乱七八糟的，很多时候个人的努力并不能保证你的成功，有时似乎还与风水命理相关，不知是不是真的？

面试时，你可以问面试官哪些问题？

明天又要去参加一次面试。每次面试的时候，面试官都会在最后给面试者一些时间，来问问题。这是个非常好的机会，能按照自己的思路，来了解职位、技术、 企业文化、福利待遇、企业状况和前景等情况，以弥补前面面试过程中没有了解到的情况。但较早以前面试准备不太充分，虽然也能地问上一些问题，但挂一漏万， 每次回来后，总觉得对企业、对职位没有完全了解清楚，不能依此作出很理性的决断。去？还是不去？往往还是靠大概其的感觉而定。

后来，我总结出了一张问题表，等让我问问题的时候，可以有针对性地提出来，来进一步了解职位和企业。今天又把它拿出来，复习复习，同时也跟大家分享一下，看看大家还有那些问题会问，也取取经。

1. 关于职位

a) 职责和任职要求。这大概是最核心的问题，如果你的职位有清晰详细的职位描述（JD），可能这就不是个问题了。但我曾经任职的一个职位，因为没有清晰详细的 职位描述，我在任职前也忽略了这一点，致使我在任职期间，感觉很别扭，总感觉这不是我当初想象中职位，是个不适合我的职位。

b) 前任去向和离职原因。问这个问题的目的是看看这个职位是不是个棘手的职位，如果你的前任因为这个职位很难做，遇到了这样那样的问题而无法解决，而离开，你 就要谨慎了，想想你自己是不是有能力超越你的前任。但是，如果真是这种情况，面试官往往在此问题上含糊其辞，说什么这是个新职位，或者说前任自己去创业开 公司了等等。你的对策就是“刨根问底”，寄希望从细枝末节上嗅出一点味道。

c) 上司情况（汇报对象）。我想谁也不想找个不容易伺候的老板。如果，面试官就是你将来的老板，你可以从他的言谈举止、行为方式上了解他的脾气、秉性，喜欢什么、看中什么、擅长什么，想想适合不适合你。别将来每天头上都顶个雷工作。

d) 下属情况。如果是个管理的职位，你还得了解了解你的下属（团队）的情况，比如：人员构成（研发、测试、构架、UED 等等）、学历、年龄、薪资水平等等。员 工的工资也许面试官不会直接告诉你，但你可以这样问：贵公司5年工作经验的高软平均工资大概多少？或类似的问题，以此，大概推断出此公司工资水平在行业内 的位置。如果有可能，要求跟1、2位将来可能成为你团队中的成员的人面谈一下。我曾就这样做过，这样做还可以深入了解下面我要说的技术方面的问题。

2. 关于技术

a) 开发语言和技术。你可能在自己的简历里写你会十几种语言，精通上百种技术（尽管有些你可能只会写“Hello World”），但你肯定还是倾向于运用自己熟悉的语言和技术来工作。所以最好还是问清楚。

b) 开发平台。我本人熟悉并且喜欢 Windows 平台开发，如果做其他的，我会感到不舒服。所以也会问清楚。

c) 开发工具（包括：IDE、SC、DTS、QA、Process 等工具）。这个并不是个主要的问题，但是如果你有所了解，碰巧你后面还有多轮面试，你可以事先做些功课，准备准备，不至于被人问住。

d) 系统构架。了解系统构架，一方面可以事先了解你将来要工作在什么样的构架下，另一方面，可以向面试官展现一下你更关注宏观、全局的问题，这也是作为管理者的软素质的表现。

3. 关于企业文化

a) 加班。有的人没结婚、没孩子，可能喜欢下班了还留在公司，如果再有加班费，那就更好了。但我问这个问题，主要目的是想看看这个企业是不是把加班当作一种常 态，是不是已经成为企业文化的一部分了。把加班当理所应当的事，这样的企业恐怕我待不了。大多数面试官在此问题上，不会承认自己公司有太多加班，但他们一 般还会说，有些情况下是要加班的，比如：XXXX。这个时候是你真正了解他们对加班看法的最佳时机。

b) 出差。我本人上有老、下有小，不愿经常出差。不过我知道很多人喜欢出差。所以无论喜不喜欢出差，问一问还是有好处的。这条算做企业文化，有些勉强，但也没别的更好的地方归类。暂且放这

吧。

c) 考勤。工作十几年，绝大部分时间没打卡上过班。所以事先问清楚考勤制度，心理上好做个准备。这条跟企业文化还是有很大关系的。本人还是喜欢弹性工作制。

d) 培训。企业是不是给员工提供不同形式、内容的培训能看出这个企业是不是以人为本，给员工提供职业发展的路径和机会，而不是只会一味地使唤人。

e) 团队建设。我原来的公司会经常搞一些团队建设活动。比如：Team Lunch、郊游、拓展、年会等等。这也是企业文化的一种表现。只知道让我埋头苦干的企业，我想不是我理想中的企业。

f) 同事关系。此问题一般得到的都是正面的回答。所以问不问都行。但是至少在试用期里要关注这件事情。在团结和融洽的团队内工作，心情会很不错。

4. 关于福利待遇

a) 薪水组成。薪水包括哪些内容，除了工资，是不是有双薪、奖金、各种补贴、股权股份。不要直接问月薪给多少这样的问题，这一般有 HR 的人会主动问你的。但你要了解薪水的组成，因为有时候，奖金、股权等其它收益可能会很大程度地弥补薪水的不足。

b) N 险一金。这在北京好像不是个问题，一般公司都比较正规。但不排除有阴损的企业。问一句不会有损失的。

c) 休假制度。你如果不是个工作狂，最好问问这个。如果他们比国家规定的休假多不少，你可以适当降低其它方面的要求。对我至少是这样。

d) 合同。问问工作合同跟谁签，主要目的是看看这个公司是不是外包公司。我不愿意做外包，所以每次都问问。

5. 关于企业

a) 年收入。以前面试过一家国际著名的软件公司在北京的研发中心研发经理的职位，经过7轮3个Manager5个 Developer 的面试，最终得到了研发中 心老大的面试机会。他问我，你现在所在公司的年收入多少？我说不知道。又问我再前一个公司的年收入是多少？这个我幸好知道，我说07年7千多万美元，后来 金融危机，到09年降到1千多万美元，他问我降这么多，你为什么没有离开那个公司？我说我对公司是很忠诚的。但显然没有使他信服。最后他对我的结论是：对 自己的职业发展没有明确的目标。结果倒在终点线前。所以我现在面试也问企业年收入、年利润等等，如果人均企业年收入比较低，我就要好好考虑考虑它的发展前 景。如果面试官是公司高管，我会再问下面这个问题。

b) 企业的愿景和目标。如果面试官给了你一个非常好的愿景和目标，至少说明这家企业是个有思想、有抱负的企业。如果某个公司老板自己都说不好，5年后，他的企业会在哪里、会变成什么样？我看我还是不要进到这样的企业里了。

这些问题问得时候也得引人而异、因地制宜。如果真是双向选择，多问问没什么坏处。如果只是人家在挑我，我无论如何都想进入这家企业，那你可悠着点，别把面试官问毛了。如果真是这样，后果自负噢。

单元测试设计思路

单元测试任务包括：1模块接口测试；2模块局部数据结构测试；3模块边界条件测试；4模块中所有独立执行通路测试；5模块的各条错误处理通路测试。

模块接口测试是单元测试的基础。只有在数据能正确流入、流出模块的前提下，其他测试才有意义。测试接口正确与否应该考虑下列因素：

1、输入的实际参数与形式参数的个数是否相同；

2、输入的实际参数与形式参数的属性是否匹配；

3、输入的实际参数与形式参数的量纲是否一致；

4、调用其他模块时所给实际参数的个数是否与被调模块的形参个数相同；

5、调用其他模块时所给实际参数的属性是否与被调模块的形参属性匹配；

6、调用其他模块时所给实际参数的量纲是否与被调模块的形参量纲一致；

7、调用预定义函数时所用参数的个数、属性和次序是否正确；

8、是否存在与当前入口点无关的参数引用；

9、是否修改了只读型参数；

10、对全程变量的定义各模块是否一致；

11、是否把某些约束作为参数传递。

如果模块内包括外部输入输出，还应该考虑下列因素：

1、文件属性是否正确；

2、OPEN/CLOSE 语句是否正确；

3、格式说明与输入输出语句是否匹配；

4、缓冲区大小与记录长度是否匹配；

5、文件使用前是否已经打开；

6、是否处理了文件尾；

7、是否处理了输入/输出错误；

8、输出信息中是否有文字性错误；

检查局部数据结构是为了保证临时存储在模块内的数据在程序执行过程中完整、正确。局部数据结构往往是错误的根源，应仔细设计测试用例，力求发现下面几类错误：

1、不合适或不相容的类型说明；

2、变量无初值；

3、变量初始化或省缺值有错；

4、不正确的变量名（拼错或不正确地截断）；

5出现上溢、下溢和地址异常。

除了局部数据结构外，如果可能，单元测试时还应该查清全局数据（例如 FORTRAN 的公用区）对模块的影响。

在模块中应对每一条独立执行路径进行测试，单元测试的基本任务是保证模块中每条语句至少执行一次。此时设计测试用例是为了发现因错误计算、不正确的比较和不适当的控制流造成的错误。此时基本路径测试和循环测试是最常用且最有效的测试技术。计算中常见的错误包括：

1、误解或用错了算符优先级；

2、混合类型运算；

3、变量初值错；

4、精度不够；

5、表达式符号错。

比较判断与控制流常常紧密相关，测试用例还应致力于发现下列错误：

1、不同数据类型的对象之间进行比较；

2、错误地使用逻辑运算符或优先级；

3、因计算机表示的局限性，期望理论上相等而实际上不相等的两个量相等；

4、比较运算或变量出错；

5、循环终止条件或不可能出现；

6、迭代发散时不能退出；

7、错误地修改了循环变量。

一个好的设计应能预见各种出错条件，并预设各种出错处理通路，出错处理通路同样需要认真测试，测试应着重检查下列问题：

1、输出的出错信息难以理解；

2、记录的错误与实际遇到的错误不相符；

3、在程序自定义的出错处理段运行之前，系统已介入；

4、异常处理不当；

5、错误陈述中未能提供足够的定位出错信息。

边界条件测试是单元测试中最后，也是最重要的一项任务。众的周知，软件经常在边界上失效，采用边界值分析技术，针对边界值及其左、右设计测试用例，很有可能发现新的错误。

2.单元测试用例及用例设计

了解了单元测试的任务，下面介绍测试用例，测试用例也是单元测试的核心，决定你的单元测试是否成功。测试用例的核心是输入数据。预期输出是依据输入数 据和程序功能来确定的，也就是说，对于某一程序，输入数据确定了，预期输出也就可以确定了，至于生成/销毁被测试对象和运行测试的语句，是所有测试用例都 大同小异的。

单元测试测试用例一般采用逻辑覆盖法和基本路径法进行设计。

2.1.逻辑覆盖法

逻辑覆盖是以程序内部的逻辑结构为基础的测试用例设计技术，这一方法要求测试人员对程序的逻辑结构有清楚的了解。逻辑覆盖可分为：语句覆盖、判定覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖与路径覆盖。

1.语句覆盖就是设计若干个测试用例，运行所测程序，使得每一可执行语句至少执行一次。

2.判定覆盖就是设计若干个测试用例，运行所测程序，使得程序中每个判断的取真分支和取假分支至少经历一次。

3.条件覆盖就是设计若干个测试用例，运行所测程序，使得程序中每个判断的每个条件的可能取值至少执行一次。

4.判定--条件覆盖就是设计足够的测试用例，使得判断中每个条件的所有可能取值至少执行一次，同时每个判断的所有可能判断结果也至少执行一次。

5.条件组合覆盖就是设计足够的测试用例，运行所测程序，使得每个判断的所有可能的条件取值组合至少执行一次。

6.路径测试就是设计足够的测试用例，覆盖程序中所有可能的路径。

2.2.基本路径法

基本路径测试法是在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例的方法。设计出的测试用例要保证在测试中程序的每个可执行语句至少执行一次。基本路径测试法包括以下5个方面：

1.程序的控制流图：描述程序控制流的一种图示方法。

2.程序环境复杂性：McCabe 复杂性度量；从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确定程序中每个可执行语句至少执行依次所必须的测试用例数目的上界。

3.导出测试用例。

4.准备测试用例，确保基本路径集中的每一条路径的执行。

5.图形矩阵：是在基本路径测试中起辅助作用的软件工具，利用它可以实现自动地确定一个基本路径集。

2.3.单元测试用例设计案例

好久前看过一篇非常不错的单元测试用例设计案例，但没有保留下来，感觉好可惜。一个好的单元测试案例对单元测试的入门是非常重要，希望这个案例能够帮助你对单元测试一些了解和对单元测试重要性的一个认识。

例如币种换算单元函数，要对这个函数进行单元测试。应该怎么去做个函数的单元测试呢？首先考虑的是该币种换算单元函数的输入输出参数，其次是输出结果 与期望值是否相等；再次建立单元测试处理类，并相应建立测试该函数的测试方法，按照 JUnit 规范一般是 test****()。

软件质量的“奥秘”——虚伪的质量

注：下面此文中提到的质量的行政与情感色彩，只是温伯格从心理学的角度揭示和探究组织内管理改进的方式，请勿以此来片面的理解“质量”，并作为自己无 法开发出高质量产品的借口。关于质量的客观定义，请参见朱少民老师的文章，建议先读朱老师的文章再读此文，以免对质量产生误解：

质量的定义总会带有政治的和情感的色彩吗？

什么是质量？似乎已经有了非常多的答案，从“质量就是零缺陷”、“质量就是满足客户需求”一直到“质量是满足客户需求的程度”，仿佛我们已经找到了答案。可是这些答案为什么总是无法解决我心中的困惑？

“满足客户需求是我们唯一的目标”作为公司的质量方针已经这么多年，可是为什么在软件开发中我们始终还只能不断的喊着“从客户的角度”的口号，而“从客户的角度”出发的思想却始终无法在开发团队中落实？

项目经理们一旦受到了进度的压力，什么质量、什么从客户角度出发就被他们毫不犹豫的扔到了九霄云外？

为什么在公司提高过程符合度的重压下，过程符合度指标急速上升，而有的团队质量却没有根本性的进步？

为什么有的产品终于实现了第一次开发达到了进度零偏差，但实际却偷偷拿着另外的版本提供给客户，并因此得到了公司的嘉奖？再后，进行审计时发现，获得嘉奖的产品其开发过程审计结果也一般般，甚至是同一个部门中比较落后的团队，其他开发团队对此更是嗤之以鼻！

……

最被大家认可的质量定义“质量就是满足客户需求的程度”，以前从未对此表示过怀疑，一切都如此的自然，就好像质量天经地义就应该是这样。我的思考和理 解也就停留在“客户是不同的，需要区分”、“满足的程度应该如何衡量”等诸如此类上，但它却始终没能解决我心中的困惑，我也没能成功将它和我们软件开发中 存在的各式各样的奇特现象联系在一起，除了在无计可施时，向着开发和测试人员喊喊“你们要从客户的角度考虑问题”的口号，就毫无办法！直到一天看到了大师 温伯格的三卷套（《质量.软件.管理》），才明白这一切的根本，没有谁比温伯格更深刻的揭示了“质量”的奥秘。

而这奥秘对我来说，无异于当头棒喝，将所有的困惑打在了一起，又一个个解了开来，也能让我更冷静的看待开发中存在的种种问题，从而能够更好的处理这些 看似简单却复杂的质量问题。究竟什么是质量的奥秘呢，上面的质量定义中究竟隐藏了什么让我们看不真切呢？答案竟然如此简单和虚伪：“质量就是对某个人而言 的价值，它的背后是行政和情感！”。“行政和情感”意味着，质量好并不仅仅是缺陷少、功能多或者是服务好，一切在于客户的感受！一个容易忽视的例子是：我 每次去买衣服时，最害怕的是什么，是怕服务员太热情，总是一见面就拉着我介绍这介绍那，她们的介绍总是那么详细以至于我总是心里发毛，为什么？

我对名牌和款式实在是所知甚少，而她们的介绍几乎总是让我感觉自己是那么无知，而这让我很自卑（以前一直没意识到），于是每当看到太过热情的服务员我几乎每次都选择了逃避，我会告诉她，我只是随便逛逛。

是她们的服务不好吗？显然不是，我会说她们服务得很好，可是这对我来说却不是什么好的购物感受，我却不能说她们的服务质量很好，因为她们显然没有达到 她们的目的－卖出，而我也同样没有得到我想要的－买进。一切在于客户的感受，这和以客户为中心的思想并没有什么区别，只是事情的关键在于质量的行政和情感 本质，当这从软件组织向外看时，没有什么稀奇，但是一旦以它的目光看回开发组织的内部，事情就不那么简单了，它揭示了软件质量的“不确定性”之外的另一个 重要特性——层次性。而如何破除内部质量层次的封闭性将是一个软件组织实现真正的“以客户为中心”的关键，否则“以客户为中心”就只能沦落为一句口号。

“形象代言人”与“抽风式管理”

“形象代言人”与“抽风式管理”两者似乎没有什么联系，和程序员似乎更没什么关系。但软件行业内，这两样东西都是存在的。

先说“形象代言人”：

什么是“形象代言人”，成龙对于“霸王洗发水”，郭德纲对于“藏秘排油”，都是形象代言人。那么，一个软件开发团队的形象代言人是谁呢？他真的存在么？

一个软件开发团队当中，往往存在这样的一类人，他们不做任何具体的编码工作，而且极有可能不会编码，具体工作是陪各种领导及客户吃喝玩乐一条龙，独自 一人的时候在独立办公室内玩空档接龙和红心大战，闲着没事的时候，拿个点名册在办公室门口抓迟到的下属，或者把某个程序员叫到他的办公室内做批判性“谈 心”。

幸运的是，绝大多数团队中这类人不存在或者极少数的在；不幸的是，这类人往往会被冠以领导的头衔。

我把这类人，称之为软件团队的“形象代言人”。就好比成龙代言霸王洗发水，他只是一个提升广告效应的形象，而不会做洗发水。回想一下，你的团队中是否有这样的角色存在。

我个人不反对这样的“形象代言人”存在，但“形象代言人”的某些行为会让我感到反感。先用微软做个例子吧！

在微软的团队中，所有的测试人员，必须会编码！为什么？因为不能编码的人，无权评判代码的是否正确！

与此同时，微软中所有的项目经理，都有自己负责的编码任务！为什么？因为不能编码的人，无权指导别人的工作！

虽然我对微软的部分行为持保留意见，但对以上两个做法是十分赞同的！一个不懂得厨艺的人，如何在一家五星级酒店当厨师长？他可以是大堂经理，可以是服务生的领班，但一定不能是厨师长！为什么？我不解释！

回到原来的话题，如此比较，一个软件团队的“形象代言人”可以空档接龙，可以花天酒地，但他们可以指挥一群程序员进行工作么？

Of course……

not!

如果作为领导的“形象代言人”同志，将决策权下放给程序员，那么结果也许会不错。如果他行使了领导该有的决策权，那结果一定很糟糕。这结果也就是我接下来要说的——“抽风式管理”！

作为一个软件团队的领导，其实大可不必事必躬亲，但一定要做到了如指掌，对自己直属部下的工

作内容及进度了如指掌。也就是说，他作为自己的下属，也可以很好的完成任务，也就是可以“躬亲”。

中国有一种“大好”的风俗——外行人统领内行人，各行各业都存在，最典型的莫过于“叉腰肌”同志。最终只会让内行人耻笑。而软件团队的“形象代言人” 不断的做决策，以彰显自己的领导地位的话，其实也就是在招致内行人的耻笑，更何况程序员本身就是一群聪明的、有独立思想的“内行人”。

第一，作为外行人，他并不知道要如何去做，却在指挥内行人按照外行的方式进行工作，结果是只能让内行人变成外行人，或者仅能做到一个外行人的水平。试问你的领导一边问你什么是“视图”一边和你说这个东西要写“视图”处理，你会是什么感觉？

第二，“形象代言人”肯定无法做到了如指掌、事必躬亲。那么他插手管理及决策的效果就是打一枪换一个地方。无法做长期的规划和指导，也无法对每个下属 做有连贯意义的指引。也就是“抽风”。今天抽 A，明天抽 B，然后每天风向都会变，作为帆船的 Developer 在不断变化的风向中很难到达彼岸。

“抽风式”管理的最终结果是将一个团队抽成了散沙，每个人都在忙活自己那点东西，同时也将“专家”抽成了“雏儿”。 孙悟空保唐僧西天取经一 样，悟空同志火眼金睛，偏偏有个肉眼凡胎的师傅，老唐还偏偏要指导悟空，“亲爱的，那个妖怪不是妖怪！”于是，孙悟空的火眼金睛废了！

这就是“抽风式管理”，而抽风式管理的出现是因为有一个“形象代言人”存在。JoleSpolsky 说过：坐在独立办公室里看《星际迷航》的项目经理，没有决策权！

我表示同意！当然，“形象代言人”的出现未必代表一定形成“抽风式管理”，前提是成龙同志不指导“霸王”如何做洗发水！

工作不能用“生产效率”这个词来衡量

通过反复的交谈，Bill Caputo 最终说服了我，让我相信了一些不可思议的事情。这些事情改变了我整个看问题的方式，也让我重新思考如何更好的工作。

软件开发中没有“生产效率”。

几乎正如10年前 Martin Fowler 发现的，用生产效率来衡量软件开发工作没有任何意义。原因就在于，它们不属于同一范畴。换句话说，生产效率不具有作为衡量软件开发工作的适用性。“今天创 造了多少代码/软件?”这是一个没有意义的问题。即使可以这样测量，软件开发工作上的生产效率也不能以任何有意义的方式估计出它的商业价值。

这是因为，软件开发这种工作并不一定非要生产出什么东西。让我来举个例子：比如说，碰巧有两个程序员分别在开发两个完全一样的项目，他们在同一天被分 配了相同的任务。第一个人，弗兰克，回到电脑前，写出了一个有1000行代码的框架，完美的解决了问题。代码规范书写，全面测试，有详细的文档描述部署和 操作的流程。第二个程序员，皮特，转身去了公园，在哪里，他一边喂鸽子一边思考问题。大概在下午4：45分，皮特溜达回办公室，删掉了200行代码，并部 署了他的修改…问题就这样解决了。

这两个程序员，今天的“生产效率”谁的更高?答案是：这无关紧要。紧要的是，皮特解决了问题，同时为团队消减了长期维护的成本。弗兰克同时也解决了问 题，但他因为生产了代码，提高了维护成本，所以，(在其它方面完全等效的情况下)他的方案差一些。而把皮特称作更有“生产效率”，则完全从实效性上扭曲了 这个比喻。

我认为，优秀的程序员，他所做的事情应该是去除问题。而相对的则是生产出什么。所以，技术上的生产产物，例代码，文档，数据等，对于实现“去除问题”的目标来说，都是必要但有害的。这就是为什么有时候，这最有效的解决方案是5分钟的交流沟通。

对这种思考模式最有力的支持：当你用这种思维去看待软件开发后，很多棘手的、能看得到但无法测量的问题突然间变得很容易理解。例如，为什么当程序员和 他们的客户隔离开时会显得缺乏效率。难道让他们避免打搅不会提高工作效率吗?答案是不会，按常理这会使他们更有效率…但也会造成他们更没效率。因为他们的 工作是为客户解决问题，与客户的隔绝导致他们无法找到问题，确定问题。相反，跟有问题的人保持沟通能更有效的解决问题，甚至有时候你一天8小时手指根本不 需要碰键盘。

这将我们引向了另外一个问题：为什么软件开发中维护成本相比起其它方面的成本显得很难接受?为什么我们永远无法在第一次做出“正确”的东西?一种解释 就是，软件是一个对可能变化的问题的固定解决方案。当问题发生变化时(或我们对它的理解发生变化时)，问题和解决方案之间就出现了裂痕。这种随着问题的演 变而不停的修补产生的缝隙的活动代价高昂。这也解释了为什么相对于其它软件项目，视频游戏通常的维护成本较低。这是因为它们需要解决的问题(让人们去买这 个游戏，玩这个游戏)基本上是根据人类心理学，而这是不常变化的。

好的程序员和坏的程序员之间10倍之差的“生产效率”又是从何说起?每个人都说这是事实，但事实上没有人能直接的测评。我们的理论同样能解释这个问 题。相比起工作效率来说，“解决问题”是一种更容易“调控”(金融词汇)的东西，使得产生一个数量级差别的效果很容易实现。解决问题需要的是信息和洞察 力。你要么有，要么没有。不需要原材料，没有生产能力限制。并不是差的程序员打字速度慢。

并不是如果他们努力就能做得更好。他们是缺乏这种高效解决问题的 眼界和必要的信息。也许无法测量好程序员和差程序员在生产效率上的差别的原因就在于没有东西可测量。

还有很多现象都可以用这个理论来解释。如果你去找，一定能发现一些。最近我一直在搜罗这方面的案例….试一试，看看这个理论是否也体现在你的工作中。 每当发现自己在说提高“生产效率/工作效率”时，问问自己是否是在用正确的方式解决问题。铭记在心：如果不通过生产任何东西就能解决问题，那生产出的任何 东西都是一种浪费。

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。

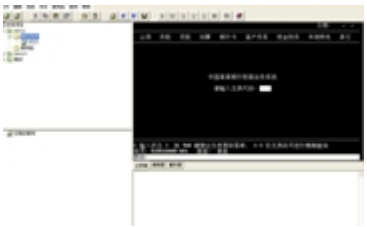
在线体验		产品租用	
企业版	免费版	在线申请	详情

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