泽众软件测试网
电子期刊

SPASVO

第十一期
2012年11月

上海泽众软件电子期刊

2012 年 11 月 第十一期

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
2012 年 11 月 第十一期 电子杂志期刊

半桶水谈什么是软件性能测试

首先在我的职业生涯中，做性能测试的机会不多，发现性能瓶颈的次数更少，确切的说只有2次。随着大型分布式系统，特别是 Web App 和云计算的推广，性能测试的需求会更加迫切。今天我这个半桶水就来谈谈性能测试的话题，欢迎大家一起来讨论。

什么是性能测试?

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。

负载测试和压力测试都属于性能测试，两者可以结合进行。

通过负载测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。

压力测试是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

中国软件评测中心将性能测试概括为三个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。

通常情况下，三方面有效、合理的结合，可以达到对系统性能全面的分析和瓶颈的预测。

注意这里提到了三个方面，但平常我们往往注意力集中在服务端的性能而忽略了客户端和网络的性能。

下面是我这个半桶水来谈谈一些体会：

一、目的/需求

为什么需要做性能测试或此次做性能测试的目的是什么?

1. 新项目/产品，首次发布，需要做基准测试

2. 使用中的产品，重构了某个模块/某个模块使用了新的技术，需要一个评估

3. 使用中的产品，用户量爆发了，用户量从百万级增加到千万级

4. 增加了一个或多个页面，需要对页面的 Latency 进行一次测试

在测试开始前，需要有准备工作，对此次性能测试的目的做详细了解，确定需要收集哪些数据，关注哪些性能指标。

二、环境部署

性能测试需要一个干净的环境，这个环境包括硬件、网络、操作系统、被测试的系统，数据库。

那在开始前，需要把这个环境部署准备好，最好是能模拟线上的系统，同时这也是一个排除干扰的过程，画出一张架构图。

1. 硬件，使用怎样的服务器，理想情况是和生产环境一样的服务器，需要配置负载均衡么

2. 网络， 是否需要模拟各种网络，是否需要双网卡，内部网络是否会影响到其他员工正常使用

3. 操作系统，windows/linux，是采用默认设置还是已经有参考的设置，哪些操作系统的服务需要关闭，linux 中的 ulimit 如何设置

4. 被测试系统，邀请架构师的参与，系统使用怎么样的架构，是否使用了 Web Server——IIS/Apache/Tomat，他们的最优配置是怎么样的

5. 数据库，站内搜索数据库是否和订单数据库分开，数据库是否有缓存，数据库是否使用主从式

三、场景和负载模式

此次性能测试需要模拟怎样的场景：

购物网站秒杀活动，使用恒负载模式，设定集合点，用户并发

系统正常的使用，从每天的监控系统中分析，例如看到购物高峰是中午，使用单步负载模式(Ram-up/down)，每间隔时间内用户增加多少，到购物高峰时间是恒负载，过了高峰后用户开始减少

系统改版，使用基于目标的负载模式，根据历史数据设定目标，或例如设定 CPU/Memory 最大到85%

确定好了场景和负载模式，才能正确的生成用例/脚本。

四、性能指标

此次性能测试需要收集哪些性能指标和数据：

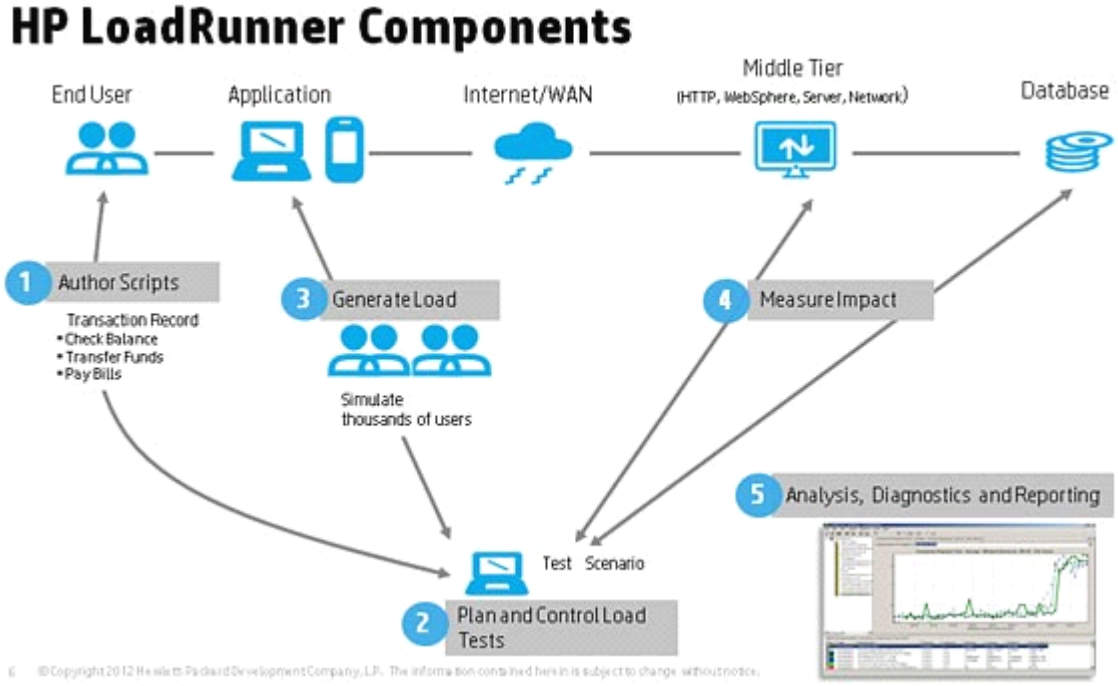
1. CPU 使用率
2. Memory 使用率
3. QPS
4. 响应时间
5. 网络 IO
6. 文件 IO
7. 数据库 IO
8. 最大支持用户数

以上指标并不是都要收集，要根据具体的场景来决定。

五、性能测试工具

提到性能测试，很多测试人的第一概念就是工具，比如商业流行的 Loadrunner，开源流行的 Jmeter，但是很少人注重上面提到的四点，所谓“磨刀不误砍材工”，没有上面的设计分析，仅仅使用工具跑出结果是无法分析出性能的瓶颈，不可靠的数据结果会大大增加排查工作，这些数据往往会受到很多质疑。

工具的使用：



参数化

Loadrunner 选择哪个协议

Loadrunner 的关联

Loadrunner 的 Controller/Agent

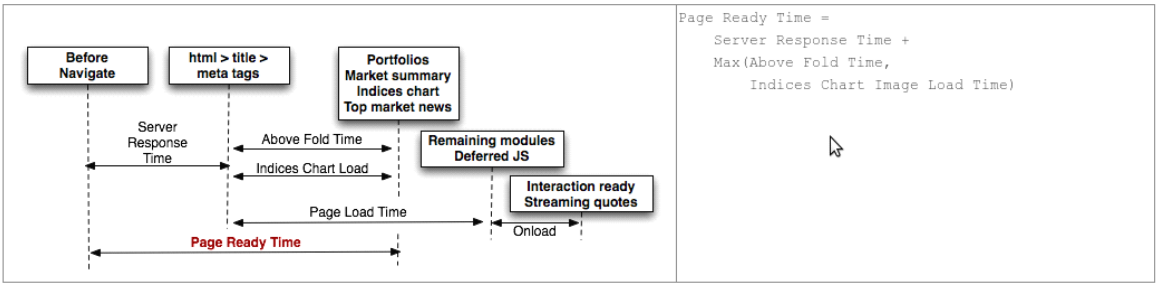
Jmeter 的 Remote testing

工具的使用技能仅仅是性能测试里的一小部分，切不可只关注工具使用。

是否每次的性能测试都需要用上大型的工具呢，某个页面的响应时间是否可以使用其他轻量的工

具，例如一些浏览器插件 Httpwatch，Yahoo YSlow，Google speed tracer;

是否可以自己写些有针对性的小工具，根据实际情况定义出真实的 Page load time，而不仅仅是服务器的响应时间。



六、数据收集与分析

性能测试的最终目的是通过数据收集分析出系统是否存在瓶颈，所以数据收集和分析是一个很重要的过程。

分析的过程需要团队成员的参与，例如架构师、DBA、开发人员，是一个长期的过程，通过调整测试脚本，生成不同的数据对比。

七、我所发现的2次问题

1. 双网卡问题，一个新改版的项目，上线后在峰值的时候总有机器崩溃或性能大幅下降，最后发现是服务器只使用了单网卡，这个明显就是上面提到的排除干扰没做好(配置没有检查)，这次性能测试是失败的。

2. 子进程崩溃，64位系统基准测试，通过与团队成员的不断沟通，排除各种设置干扰、确定硬件和软件配置、加入 profile 工具，与32位系统的对比。

八、后续优化

性能测试仅仅是个开始，性能测试最终目的是发现和解决系统的瓶颈，这就涉及到优化，而优化的过程往往在系统设计阶段就需要考虑。

1. 异步获取数据

2. 建立缓存

3. 分布式

4. 文件分解

为了发现数量众多的 Bug 而欢呼？

这不是一篇关于软件测试人员的工作评论方面的文章。最近参加了一个测试总结会，至少在两个项目汇报过程中发现，开发管理者感兴趣的一个度量指标是：你 们发现了多少个 bug？然后，当得到的回答是一个很高的数目或是一个很严重的缺陷（如：3个严重的 bug！），就会得到热烈的鼓掌。

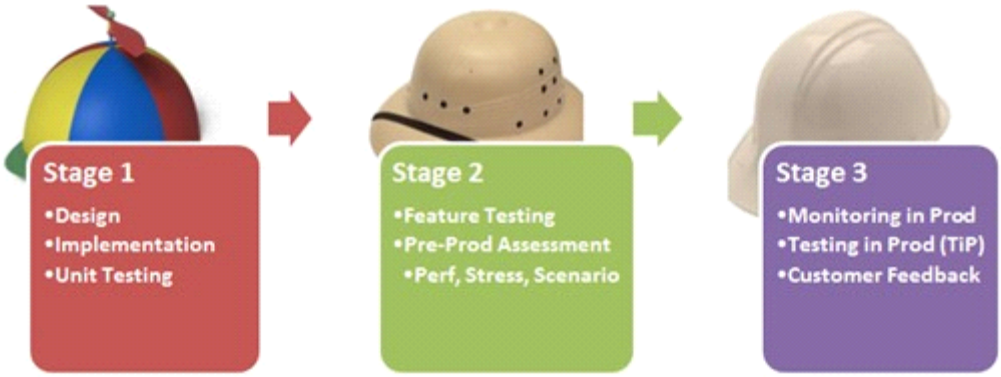
不过，我感觉这样做是不对的！我的第一反应是，好吧，让我换种说法……

“在我们的产品中，我们在设计和实现过程中出现了多少严重的缺陷？”“仅仅3个？”“太棒啦！（鼓掌）”

或者是：“哦，测试团队，你找出了多少个我们程序的致命错误？”“才3个？”，带着得意的笑容，“感谢测试团队（鼓掌）”

这表明，询问“多少个 bug？”是种错误的方法，像这样的事情，我一旦发现会严厉地批评。但后来我意识到，这个特有标准的诱惑也曾让我深受其害，不仅 在我过去无知的岁月中，甚至更多的是现在。为什么呢？对“多少个 bug”的有效性来说，这意味着什么呢？ 下面是我的一些观点。

三个阶段



软件开发的生命周期可以按多种不同的方式进行切分。在这里，按我的观点，我会把它描述成 3个阶段（见上图）。

- 阶段1：设计、开发、单元测试
- 阶段2：功能测试、上线前的评估测试：性能测试、压力测试、使用场景模拟
- 阶段3：线上监控、线上测试、客户反馈

上面列出来的项目是我们在每个阶段参与的以质量为中心的活动。

当我们询问发现了多少 bug 的时候，我们是针对第二阶段。所以问这个问题本身不是错误的，错误在于我们忽略了第一和第三阶段质量的影响和贡献。

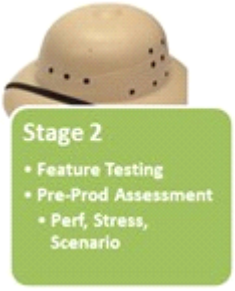
阶段1的质量

在这篇博客开头，我开了一些玩笑，我现在想说的是第二阶段的高 bug 数意味着第一阶段质量下降。当开发总是主导设计，一个可靠的质量将会来自测试（系 统的可用性和可测性）和项目管理（客户至上）的贡献。开发完成的质量不仅仅依赖于好的开发经验，当测试驱动开发（TDD）被用上的时候，还跟单元测试紧密 相关。除此之外，单元测试也能让我们对“所得是否所需”有个最基本的了解。

阶段1的质量指标：

- 开发和测试、PM 一起展开设计评审（双重检查）；
- 需要结对 review 的代码的百分比。我的观点是--100%。这不仅是为了指出代码中的错误，更是一种重要的方式，能让高级开发人员指导更多初级开发人员使用更好的开发经验，比如采用设计模式和代码重用；
- 单元测试代码覆盖率。咦，我有提到过？一些人可能会想象这是一个有争议性的论断。但就 像 bug 的数目没有尽头，而是一个未知数一样，代码覆盖率也是；
- 代码覆盖率缺口分析：那些没有覆盖的代码，我们是否遗漏了什么？
- 静态代码检查；

阶段2的质量



我想阐明的一个主要观点是：当许多软件专业人员想到软件质量的时候，他们就会想到这个阶段。这种观念的错误可以用一句谚语来概括：“质量不是测试可以测出来的”（如果有人知道这句谚语是谁说的，请告诉我下）。

这只是整个过程的一个阶段。有很多阶段1的质量评估方法在这有对应的部分：

- 测试计划是否被开发和 PM review？
- 测试代码结对 review 的百分比；
- 集成测试代码的覆盖率（和往常同样的说明）；

然后是这个阶段特有的部分：

- 有记录的测试结果：这个对性能测试和压力测试尤为重要，因为它提供了我们所知的能在生产中接受的基准指标。
- 所发现的 bug 数目和严重程度。重点就在这了，因为它是一个有效的质量/风险指示器。但它不能放在真空中，它必须和第1、第3阶段的结果在一起才能说明问题。
- 难道发现大量的、很严重的 bug，就意味着超级有效的第二阶段会把这个产品所有的风险排除？或者意味着阶段1质量非常糟糕时，我们可以期望更多的灾难折磨我们的客户？
- 难道一个很少的 bug 数意味着我们阶段2的工具是在浪费时间？或者意味着阶段 1非常给力然后带来了高质量的代码？

阶段3的质量

我之前讲过线上测试（TiP），它是一个有效的针对软件产品的测试方法。这种方法的接受程度（不是方法本身）还有点新。然而线上监控就不新鲜了。亚马 逊就是一个很好的例子，快速开发和良好支撑的监控工具，加上其它工具使得亚马逊能对产品发布作出快速响应（也就是补丁），这已经成为亚马逊各种服务的质量 保证制度的一部分。你也许会问，即使你能找到线上缺陷并快速修复，难道就允许将这些缺陷带到生产中？“质量”，是的，你只要问问亚马逊的用户他们是否遇到 过问题，或者看看亚马逊的用户满意分数就明白了。

既然我们承认产品有一个合理的质量阶段，那为什么不在第2阶段把所有的问题找出来，而不用管第3阶段呢？问题的答案是成本。如果我们尝试用第二阶段的 大规模预先测试找到所有的问题，那我们就会因为不断增加的成本而得到越来越少的回报。在前面两个阶段的基础上，用上第3阶段是一个合理的、划算的方式，能 让各种产品的质量最大化。那对第二阶段的 bug 数这意味着什么呢？它意味着我们应该非常强烈地意识到找出那些 bug 我们所付出的代价，并确保它有所值。

结论

那些曾由于对 Bug 数目感兴趣，而被我在会议中严厉责骂的伙计们，在整个软件开发生命周期（SDLC）中， 只要你们能够承认 Bug 在不同阶段出现的数量及其原因，我也非常愿意加入到你们之中，并乐于接受这个结果。

隐形的软件质量

最近被问到了一个话题：软件的质量真的能完全看得到么？比如两款手机，一个是山寨机，一个是三星或者 Google Nexus，同样是安卓系统，同样的配置甚至装上同样的 App，那么它们从测试的角度功能可以说 Function 是完全一样的。它们的质量的差具又在哪里 呢？可能这个例子是手机还可以划分到硬件范围，那么举一个纯软件的案例，同样的两款软件，比如都是微博、或者浏览器，当他们外在的功能几乎完全一样的时 候，你如何衡量他们质量的区别在哪里？

这是一个很有意思的话题，互联网的迅速发展和变化，改变了许多软件设计、研发的模式，许多曾经红极一时的书籍里提到的最佳测试实践也好，测试理念和方 法论也好在当前的时代，效率其实是非常低下的，甚至在许多情况下毫无意义，并适得其反的拖延了抢占市场的先机。软件测试的传统的流程是：开发人员写好代 码，测试人员制定策略，编写用例，覆盖率，把所有的 bug 都找出来然后改正、回归再最终发布。但是当代的软件测试，特别是互联网测试，夹杂了两个特点对传 统的测试流程形成了致命的冲击：1. 需求很可能是时刻变化，传统测试的笨重流程根本无法适应需求的迭代更替。2. 市场 Timing 转瞬即逝，在许多情况下测试团队不会有足够的时间来执行全套测试，你可能面临代码提交之后2个小时要正式发布这样的挑战，如何在有限的时 间完成对质量的把关，倍加重要。总结一点：团队要敏捷的开发，更要敏捷的测试

在过去三年的测试实践中，我遇到了这样一种情况：我们的研发团队几乎都是加拿大很优秀的工程师，他们对需求的理解、代码的质量、单元测试都做得非常不 错。于是我发现在进行功能测试的时候，QA 在95%的情况下把功能确认了一遍执行正常就完成了测试任务，在5%的情况下发现了一些显而易见更多的是 UI 的 小问题，fire bug 并快速的跟这些负责任的开发修复 bug，测试任务也随之完成。其实从一个较长的周期来看，QA 总是在不断的确认功能，不能有效的发现许多有价值的 bug。这是为什么呢？因为开发人员本身具有不错的水准，他们接到某个功能研发的任务，能高质量的编写代码完成并本着责任心自己会多次确认（包括单元测 试），而在一个架构较为成熟的系统中由于解耦合，不同的开发人员在不同的模块中所写的代码，又不太容易产生交互的深层次 bug。这种情况下，QA 按照传统 的理念和流程，去一遍遍的安装、操作功能、确认功能正常后测试就 pass 了，仔细想来，效率低下，且意义真的不大。

遇到过几个案例，经过测试团队测试通过后发布的产品，在客户那里得到了一些不太好的反馈：比如网页加载速度不理想、比如有的功能在跑的时候网线掉了整 个 server 就挂了，比如一些莫名其妙的错误弹出来，还有特定的条件下例如在用户频繁操作了一个月以上才引发的一些性能问题和不好的用户体验……这些案 例无疑都是测试团队的警钟。并不是大家在确认功能交付的时候做得不够仔细或者工作不努力之类的，笔者觉得这是因为软件在表面的功能健康之余，还存在着一种 笔者叫做“隐形的质量”的东西。

所谓的“隐形的质量”，可以这样来理解，面对同样的需求使用同样的技术，一个国内的二流团队来开发跟 Google 这样的精英团队来开发。经过基本的功 能验收发布之后，可能从功能上来看，二者没有太多的区别。但是不用问也都知道，如果说二者的质量是对等的，那是痴人说梦！那么在表面功能都一致的情况下， 所谓二者的质量之差，就可以理解成“隐形的质量”了。它可能包括的范围包括了传统测试理论里的：系统的稳定性、性能、兼容性；还包括笔者所想到的：代码的 质量、架构的合理性、消息传输效率、数据存储搜寻命中的效率、软件持续运行的各项健康指标数据模型、甚至包括了 UI 的人机交互等许多概念。后面所提到的方 面，很少在测试团队中听到，但却都是在当代软件行业非常影响产品质量和成败的因素。笔者认为，往往那些优秀的团队和公司之所以能在竞争中屹立不倒，都在 “隐

形的质量”方面做到了一定程度的保障。

自动化也是在测试行业一个被反复炒作的概念。其实道理也是一样的，花了许多时间建立了基于功能的自动化测试，这本身没错。但往往在优秀的开发团队提交 交付的代码中，仅仅基于功能的自动化是发现不了多少 bug 的。自动化的目的是代替人的繁琐重复的操作，这一点没错，但考虑到维护的成本和变化更新的代价， 需要谨慎为之。如何在“隐形的质量”这个领域中探索测试的方法和流程，并有效的结合自动化来快速的完成测试，这可能才是未来软件测试的发展趋势。笔者听过 的两个案例分别来自 Facebook 和 Google：Facebook 没有测试工程师，或者说所有的开发工程同时也是测试工程师，他们编写对自己提交代码 的自动化覆盖测试，但更重要的是他们研制了各种工具来监控采集上线产品和后台 server、数据库在运行期的各种关键指标，由此来反馈产品的质量。而 Google，他们有自己的测试工程师、也有测试开发工程师，在 Google 里不会把开发和测试分成不同的学科，测试和开发是齐头并进的，测试不是独立的 实践活动而成为了开发的一部分，它们认为质量是当你把开发和测试搅拌均匀分不清彼此的时候才能产生的。他们甚至把测试当成了产品的一个功能从而让整个团队 对产品的质量负责。总结一句：Facebook 无测试人员所做的自我监控分析，和 Google 少而精的挑选测试人员融入开发团队不分彼此共同保障产品质 量，正是它们为了保障“隐形的质量”做的特别好的方面。

说到这里，笔者觉得当代的测试行业已经从一个相对统一的测试实践和理念的经验圈子中跳了出来，在各自摸索中前进探索最适合自身的测试工具和测试流程。 更多关于这个话题的思考和讨论，笔者也会继续写在 blog 中，推荐一本书“How Google Test Software”，里面介绍了作为全世界发展最快的顶级互联网公司，Google 的测试工程师是如何来完成测试任务并保障产品质量的。

关于软件测试团队的管理

在测试团队管理上所存在的问题总结以及制定相关的措施：

1、责任。测试作为项目提交的最后节点，直接影响到最后项目的结果，一旦出现差错，到底是谁的责任呢？现在团队的测试是程序直接对口测试，也就是说程序开发完包，直接发给了相关人员测试，中间缺少了相关的监管环节，一旦出现问题，不能直接责任到人。

问题分析：无论是经理也好，还是测试组长也好，必须要对整个的测试进行有效的计划，把控，分配以及检查。这个是一个简单的事情重复做的情况，所以以至 于一些人在思想上不想去做这个事情，因为这个对自己的业务发展并没有多大提升，一直在一个阶段重复的做这样一个事情。而且事情多的时候，容易把这个事情耽 搁下来，所以需要制定一个有效的策略，减少花费在相关的管理上的时间，同时要保证管理的有效性。

2、奖惩，作为测试的工作可以说，不出问题就是最大的成功，一点出现问题就是工作失误。现在就是觉得没有问题就安然无事，一旦有问题都是测试的问题。这样测试出现的都是问题，不会有成功的表现。

问题分析：对于测试的奖惩就不能以常规的思路来进行衡量，需要一些数据来进行相应的分析，并且建立起相应的标准，对测试的效果进行有效的判断；同时也是对相关测试人员的一种评定。

3、方法，很多人测试很努力，但是结果却总是不尽如意，主要的在测试的方法上出了问题。

问题分析：需要制定一个测试的流程，也就是测试的过程。这样可以让管理者很轻松的了解到测试进度，也便于大家在相同的语意下进行沟通，同时也有助于规范测试方法，至少在一个框架下做事不会偏离的太远。

由上面的一些问题的分析可以得出，下阶段需要做以下的几个事情。

1、有效的管理：需要一个工作表单，将相关测试计划，人员分配进度情况进行有效的罗列，一定的时间内更新，方面检查和工作记录。

2、制定标准：通过相关的数据，表单得出最后的结果，与相关的标准进行对比，得出测试工作的有效性。

3、制定测试流程：将测试的过程分成不同的阶段，并且对相关的测试人员进行培训，使其在相应的流程下进行有效的工作。

软件测试工程师成 IT 业国宝

近日，中华英才网发布了2007最热门行业，互联网、电子商务和计算机软件三大行业位居排行榜前三，相关人才已成为行业内的“抢手货”。其中，软件测 试工程师的人才供需比已达1：50，人才缺口正向30万挺进，受软件产业30%以上的年增长率影响，供需缺口还将不断扩大，而导致这一结果的原因主要在于 产业先行而教育滞后。

“90年代初期，国内软件企业刚起步，承接和编写的软件项目大多很简单。然而，随着软件产业竞争加剧，软件企业开始由单打独斗的小作坊式生产向分工合 作的软件工程形式过渡，虽然提高了软件的开发效率和复杂程度，但各模块间的 bug 大幅增加，导致软件整体质量下降。”康赛普特信息技术有限公司总经理王亚 智介绍道，核心竞争力的丧失让企业意识到软件质量的重要性，开始加大对软件测试的投入。“从我在微软工作的经历来看，很多大型的开发项目，测试会占据项目 周期一半以上的时间。以 IE4.0为例，代码开发时间为6个月，而稳定程序花去了8个月的时间。”前微软亚洲研究院博士、软件测试专家陈宏刚指出，除了时 间、资金的投入外，人才的支持也是实现目标的关键。然而，就现实情况来看，国内市场供给和人才培育速度远远落后于企业需求。

“目前，国内每年为企业培养的人才年供给量不过几千，与30万的人才缺口相比仅是杯水车薪。” 软件测试专家郑仁杰教授在接受记者采访时如是说。人才的稀缺让企业开始寻找多种渠道解决问题，如海外引进、内部培养等。然而高额的人力成本难以让这些方式 大规模普及，企业将更多的注意力转向 IT 培训。一些 IT 培训机构也相时而动，把握市场对人才的需求趋势，邀请国内外知名的软件测试专家共同开发系统的培训 课程。目前，IT 培训机构已为企业培养了近6000名专业软件测试工程师，成为该领域人才培养的主力军。

在 IT 职业培训大力发展的同时，一直默默无闻的国内高等院校也开始试水。在去年8月，由教育部软件工程专业教学指导委员会、上海交通大学软件学院、清 华大学出版社等组织的第一次软件测试教学研讨会在沪举行。会上首次明确提出“软件测试”是软件工程的核心课程之一。会后，各大高校就软件测试专业开设问题 进行深入探讨。对此，曾参与微软 Windows95、Internet Explorer 5.0等项目开发与测试工作的陈宏刚博士认为，虽然高校能部分缓解软件测试人才培养的一些压力，但借鉴国外发展历史来看，职业培训仍将是专业人才供给的主 流渠道。

在各界的努力下，我国软件测试人才供求失衡的局面也得到了一定程度的改善。“我原来所在公司的软件测试人员和开发人员的比例大致在1：4，虽然和国际 先进水平的1：1还有一定差距，但是比起最初的1：8有了很大的提升。”原武汉弘智科技有限公司测试经理冉春娟这样说。冰冻三尺非一日之寒，软件测试的人 才缺口不是年内就能弥补，需要教育、企业、人才三方的通力配合，才能逐步实现供需平衡的良性发展结构。

软件测试职业发展的各个阶段

软件测试职业发展的各个阶段

这是国外公司的职位分布，国内一些走在前列的公司，也差不多在国内可能晋升要快的多。

初级测试工程师

刚入门的拥有计算机科学学位的个人或具有一些手工测试经验的个人。
开发测试脚本并开始熟悉测试生存周期和测试技术。

测试工程师/程序分析员

具有1-2年经验的测试工程师或程序员。编写自动测试脚本程序并担任测试编程初期的领导工作。进一步拓展编程语言、操作系统、网络与数据库方面的技能。

高级测试工程师/程序分析员

具有3-4年经验的测试工程师或程序员。帮助开发或维护测试或编程标准与过程，负责同级的评审，并为其它初级的测试工程师或程序员充当顾问。继续拓展编程语言、操作系统、网络与数据库方面的技能。

测试组负责人

具有4-6年经验的测试工程师或程序员。负责管理1至3名测试工程师或程序员。担负一些进度安排和工作规模/成本估算职责。更集中于技能方面。

测试/编程负责人

具有6-10年经验的测试工程师或程序员。负责管理8至10名技术人员。负责进度安排、工作规模/成本估算、按进度表和预算目标交付产品。负责开发项目的技术方法。为一些用户提供支持与演示。开发一些特定领域的技术专长

测试/质量保证/开发（项目）、经理

具有10多年的工作经验。管理8名或更多的人员参加的1个或多个项目。负责这一领域（测试/质量保证/开发）内的整个开发生存周期业务。为一些用户提供交互和大量演示。负责项目成本、进度安排、计划和人员分工

计划经理

具有15年以上开发与支持（测试/质量保证）活动方面的经验。管理从事若干项目的人员以及整个开发生存周期。负责把握项目方向与盈亏责任

简析研发项目经理的管理

管理的基础是知道要干什么。开发经理的基础就是要安排好工作计划。管理不好任务计划，就没有时间干别的事。实施的问题提交给你，问你什么时候交东西，你 不知道，无法有个明确的答复，实施就会跟你抱怨，我们需要和客户沟通具体的实施时间，你连个什么时候开发出来都不知道，我们怎么去和客户说。

开发经理是 IT 界最底层的管理人员，开发经理是一类特殊的 IT 民工，他不仅仅要去做好管理，而且还必须承担一定的开发任务。开发经理一般都是由开发人员晋 升而来，开发人员做开发的时间长了，有些固定的思维，做开发经理以后还是把开发的工作放到工作的第一位，认为只要承担起主要的开发任务就是已经做好开发经 理了，所以开发经理大多数感觉做在这个位置上感觉很累。几乎都快要崩溃了。开发经理必须转变思想，管理比开发更重要，必须要抽出时间来做管理。抽出时间来做管理了，有时候感觉不得法，为什么那？因为我们没有清楚的认识到管理到底 是什么？

部门经理那天给你安排一任务，问你可以插入到任务计划中吗？你不知道；客户有 BUG 需要修改，而且很着急，你干不干，你一干，你的任务计划又不知道到什么 时候开始，什么时候完成了；那天公司的大领导要部门的软件开发必须跟的上时代的发展，去设计和开发新的产品，你只能说没有时间；而且团队成员不仅仅需要你 去帮他们解决问题，而且还需要你来安排具体的开发任务，解决不了问题，开发没有办法进行下去，影响了进度，任务没有安排好，做到那是那，有的人闲死，有的 人忙死。

每天被这些事搞的焦头烂额的，被烦也烦死了。整天疲于应付，感觉做管理也不是管理，做开发也不是开发，到头来，开发没有做好，管理也没有做好，真是难为死 了。给实施的印象不好，给领导的印象也不好，手下人的时间一长就有想法了，不想干了，感觉没有意思。为什么会出现这种情况那？问题的根源就是在于没有做好 计划管理。

管理是沟通。沟通在软件开发是很重要的事，真的很重要很重要。

不沟通，你做的功能很可能就不是人家想要的或者没有满足客户的需求，到头来费事费时。不沟通，团队成员来来回回的改功能，改一次还行，短时间改第二次、第三次，时间一长人就疲惫了，而且知道改功能是因为你没有与客户沟通清楚，很质疑你的领导能力。

不沟通，团队成员没有和你交流技术的机会，团队成员不了解你的想法，你不了解团队成员的想法，时间一长，大家各自为政，各说各的，越来越没有团队意识。还 有不沟通，上级领导无法了解你再干什么，领导还以为你干自己的事，建立自己的小世界，培养自己的势力，你要让领导了解你再干什么，没有不怀疑员工的领导， 做开发经理，要尽量的获取上级领导的支持，那样你的工作开展起来就没有那么多麻烦。误解，往往是自以后相互了解。

管理是左右逢源。管理要平衡与你打交道的各方面的人的利益，这样的工作才好开展。人都心情不好的时候，人都有烦的时候，人都有累的时候，人是感情的动物，克服内心的感情色彩，不要掺杂到工作中去，不然会让别人对你感觉很差，认为你喜怒无常。

管理是争强团队凝聚力。提高团队凝聚力的方法有很多，最主要的是要让团队的成员工作的开心，有成就感，感觉到自己再成长。

管理不能陷入细节。管理应该从大局着眼，不能老是陷入到一些细节中，特别是开发经理。要是纠结于细节，就没有时间去把握住全局，你把握不了全局，就是给别人一种不可靠地感觉。

管理要放手，管理要让合适的人干合适的工作。管理是要倚重团队里面的人。开发经理一边管理一边开发，就想要承担

起所有的开发任务。因为团队的可能对业务部熟悉，也可能技术不太成熟，肯定没有你开发的快，但是开发经理要给团队成员犯错成长的机会。永远不要低估一个人的潜能，人永远也不知道自己的潜能有多大，不在特定的情况下，人的潜能是激活不出来的。

团队的成员就是要分担开发经理的工作，所以要让团队的成员成长起来，反过来还可以提高团队的凝聚力。你永远不知道你的能力是怎么来的。要善于利用自身的优点去生活，这样你才能左右逢源。

管理也永远没有定式，自己有找到适合自己性格的管理，才能做好管

六种全面质量管理工具

一、鱼缸会议

这是一种组织会议的方式。不同的群体本着合作的精神，一起分享各自的观点和资讯。因此，让销售部门与客户服务部、或高层管理人员与管理顾问碰头，这种做法一定管用。

何时用：鱼缸会议使某些群体与顾客、供应商和经理等其他与之利益攸关的群体加强沟通。

何时不用：如果用这种方法不能明确地分清各群体的职责，就不宜使用。

培训：会议召集人需要接受培训。

能达到何目的：迅速增进了解、扫除误解。

注意事项：这类会议影响巨大。可能会暴露实情，使内情人和旁观者感到受威胁，因此需要精心组织。

使用程式：把与会者安排成内外两圈。内圈人员会上比较活跃，外圈人员则从旁观察、倾听，必要时提供资讯。会议结束时推荐改进方案，取得外圈人员的赞同。

二、横向思维

这是一种爲老问题寻找新[b][url=http://www.ltesting.net/html/81/category-catid-381.html]解决方案[/url][/b]的工具。

何时用：由於老方法、旧思路不再管用或已经不够好，需要寻找新方法、新思路时使用。

何时不用：种种制约使这种全新的思维方式无法发挥作用时不要用。

培训：建议读 Edward de Bono（爱德华）写的 Lateral Thinking for Management（管理中的横向思维）一书。

能达到何目标：开创新思路，激发创意，找出可行的解决方案。

注意事项： 需要传统的逻辑思维加以支援。爱德华建议，只有10%的解决问题过程采用横向思维。

使用程式：确定问题。运用幽默、随机排列和对流行观念的挑战来制定横向思维解决方案。对找到的各种想法加以适当的提炼和取舍。

举例来说，某工业缝纫线轴制造商的传统市场已经消失，公司不得不另寻出路。对此，公司经理们的本能反应是，从常规思路出发，爲産品找新的出路、新的市场或新的销售手段。不过，事态的发展很快表明，他们需要一种彻底的解决方案。

公司召开了一次集思广益会，对叁加者不加任何框框。思路应能用得上现有的技能和经验，但只能把它作爲起点。结果，横向思维把他们引向高尔夫球，成爲一家成功的高尔夫球制造商。

三、帕雷托分析法（Pareto Analysis）

该方法强调爲80%的问题找出关键的几个致因（通常爲20%）。

何时用：凡是一个问题的産生有多个变数因素并需要找出其中最关键的因素时，都可使用这一方法。在一个改进专案的开始阶段尤爲有用。

何时不用：如果设置有更完善的系统就没有必要使用此法。

培训：需具备基本的统计知识以备分析之用。

能达到何目标：非常直观地展示出如何确定问题的优先顺序，将资源集中在何处才能取得最佳效益。这种展示让企业各级一看就懂。

注意事项：仔细分析结果总是很重要；不仅靠资料还要利用常识来找出问题的原因和优先顺序。

使用程式：找出问题和可能的原因。收集有关原因的资讯。绘制帕雷托分析图，横坐标表示原因，纵坐标表示问题，以出现次数、频率或造成的成本来表示。找出最关键的几个原因。依据重要性排序，利用改进技术消除産生问题的原因。

例如，某洗衣机制造商出现质量危机。在一次广泛的可信度测试中，一家大型杂志将其産品排在末位元并建议消费者不要购买。

该公司具有完善的失误记录，列出的失误种类达22种。但运用帕雷托分析法表明，仅其中的4种失误就占了所有记录的83%。

四、质量功能分布图（QFD）

这是一种産品和流程设计工具，可以用於把顾客的呼声转化成産品或流程的特点。采用该方法能防止企业仅因爲某些观念似乎有效就予以实施。

何时使用：用以设计或重新设计産品或流程，保证提供顾客切实需要的産品特性；专爲制造业设计的，但也可用於服务业。

何时不用：如果问题的优先顺序已经分明、流程设计卓有成效或设计团队经验老到，不要采用该方法。

培训：该方法运用特定的惯例，建立相关的栅阵图和计分标准。在这方面有必要进行培训。

能达到何目标：有能力分辨基本的産品与流程特色和所期望的産品与流程特色，这样便可以看清高成本的技术或工程投资在哪方面将有回报。同时，还提供了一个评估産品或流程变化影响的框架准则。

注意事项：花时间通过市场调研来找出顾客的真正需求所在。

使用程式：研究顾客的需求。找出符合顾客需求的流程设计特色。建立一个栅阵图，将顾客的需求与设计特色进行比较（即性能／方案栅阵图）并加以计分。选 取5个左右分数最高的设计特色，然後再按3个层次建立栅阵图：设计特色和关键部件特点、关键部件特点和制造工序、制造工序和生産要求。

例如，某割草机制造商耗时费资重新设计其畅销割草机的控制性能，却发现顾客对此毫无反应。因此，公司经理人在计划改进另一较老型号时，想要确保所做的改善的确是顾客想要的。

研究结果表明，顾客感兴趣的是性能。因此改善马达、驱动链和刀的效率比改善控制性能可以産生大得多的影响。

五、关联树图

这种图示工具对关联项进行层次分类。这是一种不错的思维工具，因爲它提供了一种快捷的方法把各种想法总括出来，并在相关的枝叉出现时可随即增加细节。

何时使用：使用该图示可以爲同一目标寻求多种不同的实现途径。

何时不用：不可用於详细比较各种方案。它只用於从总体上探索新的方向。

培训：无需正式培训，但设置一个协调人员会很有帮助。

能达到何目标：该图示能很有逻辑地揭示出该采用什麽方法来实现目标，它们要求哪些行动和资源。

注意事项：如果你选用的方法经不起分析，要随时准备回到关联树图上来。

比如，一个发展中的小公司运用这种方法来考虑员工的托儿问题。许多员工大学一毕业就加入了公司，现在都供养着子女。

公司开会讨论各种选择方案。结果，大家都赞成建一个日托中心。但树形图显示，潜在的成本太高，需要满足的地方法规要求太多，很难实行。於是公司选择了托儿津贴计划，让有子女的员工有选择的馀地。

六、方案效果分析法（solution effect analysis）

这种图表用於分析手头解决方案可能產生的效果。

何时用：在提议变革时可运用这种方法。它能让你看清各解决方案的效果。

何时不用：你所提议的不是根本性变革的话，不要使用。

培训：无需正式培训，但如加以辅助很有用。

能达到何目标：一种向前看的思维方式并能预见所建议的方案会造成什麼影响，避免未能预见的效果。

注意事项：人们对你正在致力的变革前景看淡时，不要阻止他们。他们并非有意发难，也许他们是对的。接受辅导会减少自己受威胁的感觉。

使用程式：记下正在考虑实施的解决方案，放在图的左侧，箭头则指向右方。在主箭头两侧用分箭头标出各种重大效果。通过集思广益，找出所有可能的效果并添到图上。计划实施行动以确保该方案行之有效。

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