目录

测试介绍

测试现在被普遍认为“保证产品质量”这个笼统的说法下，而测试本身是什么呢？今天我们就测试本身跟大家一起讨论。

测试是在研发产品的整个过程中的一个跟踪活动,他在各个阶段报告给人们当前项目的状况,能够督促和提示项目经理或者高层经理对项目的关注点.

国内的测试的定义,一般是在产品的研发后期,对产品的功能进行验证的一个系列活动。

国外的测试已经发展比较成型了，而国内的测试现在还处于摸索阶段，至于超着那个方向去发展，我觉得大家目前还是处于比较迷茫的阶段。

主要原因是：国内软件产业起步晚，而且质量意识不强，造成了软件工业发展缓慢，配套行业（测试发展缓慢），我觉得这个很正常，因为从人类历史发展的角度来看，这个是必须经历的阶段，从有这个概念到摸索，目前国内的测试应该处于沉思期，主要是没有一个全套的指导思想，另外一个原因是行业发展方向不明朗。

国内存在对测试的误解，所以造成了测试现在成了大家进入企业的跳板，要么就是觉得自己的能力还不够，目前只能从事测试，要么就没有编写程序的能力，但是同类产品比较

了解，所以做测试。

我对这个问题我有自己的看法，我觉得在企业发展的同时，个人要发展，那么个人怎么发展呢？（我说的是测试人员），在国内，对应的测试的技术总结的相对很少，并且现在国内的企业测试都是把测试过同类产品当成了经验。我个人觉得这并不是错，但是我个人觉得有点偏颇。因为真正从事测试行业的人都知道，测试是需要一定的技术功底的，在国内虽然对测试这么要求，是因为国内的大环境有质量的意识形态所导致，对质量的意识还只是停留在理解和使用上，不是从设计或者原理或者其他方面保证质量的。如果我们说对测试的定义是对某类产品的经验上，那么这个人是对和他合作过的程序员和设计师比较了解，而且能够总结出来某些人在那些方便容易出错，但是当更换环境以后，这些经验是否还能有用呢？

如果我们把测试的方法整理成技术，那么他形成一个规则或者说是一个标尺，我们只是分析什么样产品的需要用什么方法来测试，而且需要了解的知识架构是什么？怎么把这些知识穿插起来，那么积累就不会被约束，但是不能撇开经验，因为经验本身是设计出好的案例的基础，但不是唯一的基础。

我们再来看看测试案例的设计，测试案例的设计在国内现在是一些刚刚入行的不会写程序或者程序功底比较差

的人在写案例，那么这些人设计出来的案例只是包含了整个测试过程中功能测试的一部分案例而已，因为他们不懂得或者不理解程序，不是从原理上去分析产品，不是从逻辑上去分析产品，而是从用户使用的角度去分析产品，这样设计出来的案例的可行性和可信度多大呢？大家可想而知了。所以我们在整个引导大家的过程中，从技术和方法，结合具体实例和针对不同类型的产品的测试方法进行跟踪和描述。

首先，什么叫测试？测试干什么？

测试，是在开发过程中的一种活动，它是分白盒测试和黑盒测试。在不同的阶段不同的人所承担着测试这个角色，我们把整个活动统称为测试。

测试的工作内容主要包含了设计测试计划，设计测试案例，执行测试，进行测试总结。

执行测试是在产品开发的整个过程中进行的，包括了单元测试，系统测试，集成测试，系统测试和验收测试，那么不同的阶段测试的重点不同。

单元测试的重点是函数级，包括需求，包括算法，包括接口预留等内容。

集成测试是指把小模块结合起来，测试的重点是输入输出数据，参数的处理，错误预处理，接口规范，参数约束等测试内容。

系统测试的重点是功能性质，它的测试重点是按照需求

来对照测试， 主要是功能实现的情况，包括功能使用逻辑和操作逻辑，操作系统，兼容性（软件和硬件）等内容。

验收测试，主要是合同性质而言的，在国外现在软件外包情况比较多，那么双方按照合同规定履行自己的职责，把功能按照合同约定的形式条条比对。这是主要方面，那么在企业内部，验收测试是除了功能验收以外，还包括易用性，软件的亲和度等方面的内容。

序

我是一个充满激情的人，我把所有的激情投入到生活的每一个空间。

我是一个不停折腾的人，因为生活在不停的折腾我。

我是一个不服输的人，因为我知道这个社会不会同情弱者，只有不停的折腾，才有可能把握自己的命运。

我是一个傲慢的人，因为我把自己已经当成是行业的开拓者。

我是一个平和的人，因为我和不同的角色在对话

我是一个开放的人，我会将我知道的或者了解的用无私的信鸽传播

测试蓝本（初稿）

王振刚（2004-4-14）

测试的分类

单元测试

单元测试是在测试过程中的最小粒度，它在执行的过程中紧密的依照程序框架对产品的函数和模块进行测试，包含入库和出口的参数，输入和输出信息，错误处理信息，部分边界数值测试。

这个部分的测试工作在国内现在是开发人员进行的。我相信未来的发展应该是测试工程师来做这个事情。那么需要测试人员需要深刻的理解程序，理解需求，理解设计，这样才能发现问题。

还有一种在国内先在操作的方法，就是当一个模块给某个开发工程师以后，需要他给大家讲解他要完成这个模块或者函数的整体流程和思路，进行统一评审，使得问题能够暴露的更充分些，这样做的目的有以下个，第一，使得大家对设计者的思路明晰的理解，以便以后调用或者配合的时候能够真切的提出需求或者相对完美配合。第二，在评审的过程中，如果发现问题，那么大家可能没有犯过，这样就会更加提高警惕，如果犯过，就会回想当时自己怎么解决的或者规避的，使得大家能够在错误的过程中快速提高。第三，可以对平常犯错误进行一个积累，我觉得这是生动的教科书，可以使得新的人员在新上手的时候遇到这样的问题以后，我们就可以给他一个解决问题的方法或

者方向。

回顾，我们上面给大家介绍了两种方法，第一种就是通过在开发的过程种进行测试，由开发（测试）工程师写测试代码，对所编写的函数或者模块进行测试，第二种就是通过代码互评发现问题，将问题进行积累，形成知识积累库，以便使得新人在同样的方面不至于再犯错误。

单元测试非常重要，因为他影响的范围和宽度比较大，也许由于一个函数或者参数问题，造成后面暴露出很多表象问题出现。而且如果单元测试做不好，使得集成测试或者后面系统测试的压力很大，而且项目的费用和进度可能就会飚升。

对单元测试，现在用 CPPUnit 的比较多，市场上也有其他对应的产品，他们在不同的软件单位不同的阶段。正确的理解单元测试的重要性是意识，需要在过程改进种不停的总结，慢慢的积累，将质量意识渗透到整个开放过程中的各个环节。

保证单元测试顺利进行，需要渗透软件工程的很多思想，把 CMM 和跟踪机制建立起来，问题的分类、跟踪，如果把软件环节整个活动都渗透了，那么产品质量的意识自然就增强了。

COM 思想现在在大的项目现在体现的淋漓尽致，因为如果不采用 COM 机制，维护和升级以及修改测试的成本很大，所以现在大型项目基本上都采用 COM 的组织形式。

说了这么多，单元测试做什么呢？单元测试主要是做一下几个事情：

第一，模块或者函数的设计稿

第二，代码规范，其中包含代码书写规范，对齐方式

第三，代码的注释。非常重要

第四，参数类型，数据长度,指针，数组长度大小

第五，输入输出参数和结果

第六，创建对象后是否删除了，如果在这里没有删除，请注明在那里删除

第七，是否应用了没有初试化的变量，如果是，请指明该变量在那里初始化

第八，变量是否声明，声明是否按照要求进行

第九，调用此函数需要的满足条件需要注明

第十，在此函数或者模块中用到了系统或者其他第三方插件函数，需要满足的系统条件

上面我只是列举了一些在测试过程中发现或者隐藏的问题，我想可能还有很多情况引发问题，请大家补充，以便在工作中有操作性。

集成测试

集成测试是在保证单元测试进行后进行的一个动作，能否集成的标志不是所有的代码编译通过了就算是可以集成了，而是所有的能够在这个虚拟环境下能够正常运转。

在集成测试种一般采用的方法是数据驱动或者桩驱动，因为集成测试不能看到产品的表象，因为他是一些数据流的中间段，我们渴望

能够对中间数据进行分析，就可以知道或者就渴望知道流程或者算法中有什么不妥当的地方。

集成测试比较适合做成自动化测试，当然首先我们分析了适合做自动化的条件是满足的，我这里就不讲详细的方法，到后面的自动化测试介绍中，我会提到这个方面的问题。下面和大家一起揭开测试自动化的神秘面纱以及给大家讲一些构建冒烟的概念。

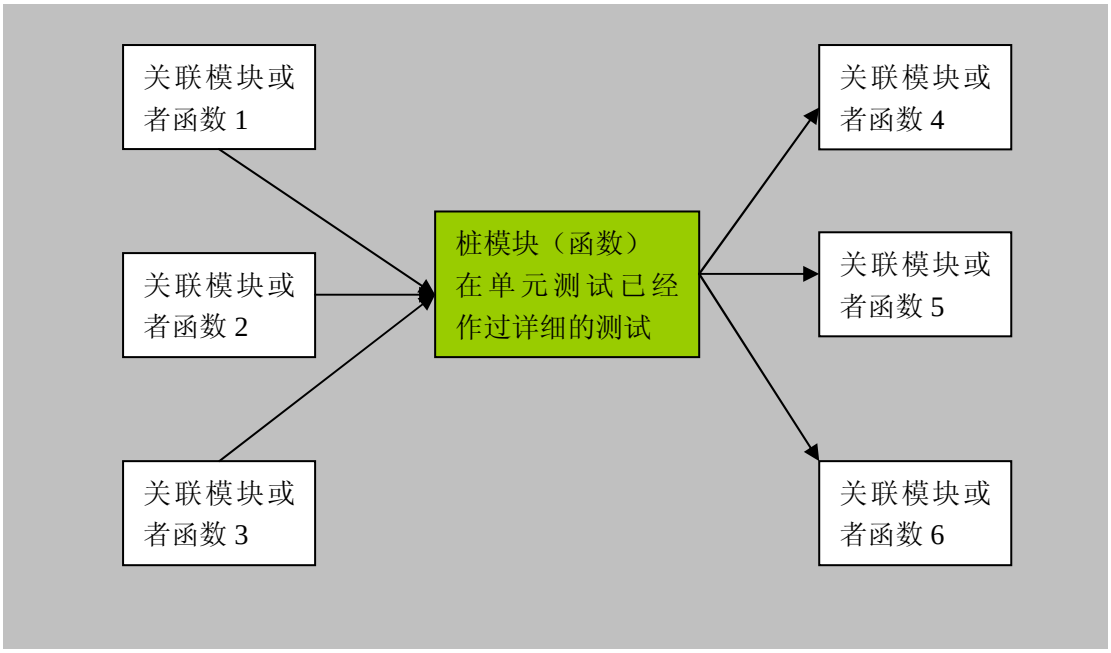
冒烟测试的出处是，由于生活习惯等原因，人们会定期的做某个事情，就像人们会约定成俗的认为 12:00 是吃饭下班的时间。那个时候大家都会做饭，哈哈，自然会从烟囱冒烟。在软件行业里面的约定是当产品到达某个阶段之后，为了验证产品的各个部分的衔接程度，为了验证项目的进展程度，为了验证产品的（已完成）功能的全面稳定程度，由测试主导的一种测试方法，主要的操作就是，在产品开发计划定制完成以后，依照开发计划指定完整的编译计划，按照开发计划和编译计划，各个单位按照要求完成自己的作业，然后在编译点上验证完成程度。

集成测试也是不可缺少的一个部分，很多单位为了赶进度，会将这个部分省略掉，就甩手给测试小组，如果没有对应的测试小组，就会是程序员进行简单的使用后就交付市场，危险，这是个定时炸弹。因为他时刻有可能产生市场对企业影响的额度，以及企业本身的声誉问题。

集成测试是在单元测试完成后进行的测试环节中的一个测试，主要是测试各个模块和函数之间的相互衔接情况，互动情况，输入输出

情况。所以集成测试也很重要。

那么集成测试一般采用什么方法呢？集成测试一般采用桩驱动的方法，因为在单元测试我们检查的相对比较详细，那么在集成测试的重点其实要保证逻辑上了。我简单的介绍桩驱动的实现方法。



请大家看上面的图，这个一定是一个有意义的组合。是函数间形成一个简单的逻辑关系。

从上面的图上我们可以看到，如果中间的模块或者函数是经过我们进行过详细的测试，基本上可以保证没有什么错误，那么如果这个逻辑出去，导致出错，一定是输入的过程或者接收了输出参数处理出错了。使得我们问题很好的定位。

我们可以定义很多个桩，使得测试效率提高。

我们把上面的内容进行简单的总结：集成测试就是测试各个组件之间的配合情况。所以集成测试是为系统测试提供了一些基

本保证，但是不要完全依赖。

采用的方法给大家介绍了，这样可以采用测试或者程序编码的形式实现测试。

系统测试

系统测试是测试过程中的一个转折点，因为在现在国内的企业中，不同的产品面对不同的用户群体，所以有的企业经过第三方产品的验收测试，有的企业则没有通过验收，而是一些工具类或者通用类的产品，那么他的验收测试是经过广大的用户群来做的，也就是说凡是通用类产品的系统测试必须严谨测试以后，才可以投放到市场。但是对于对企业或者其他专业性单位定制的产品我们必须进行验收测试。

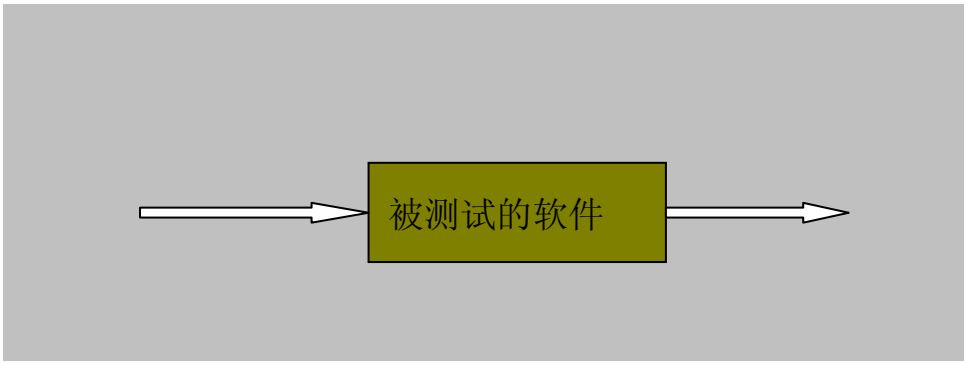
系统测试工作是一个重复老动很多的工作，需要在工作种把握几个重点，系统测试是保证系统能够正常运转，包括了功能，易用性，健壮性，压力，边界数值设定等各个方面的内容。要想在这个阶段的工作种找到乐趣，就要不停的摸索，找出能够将机器代替人的所有的东西，找工作的快感。

系统测试需要有广泛的知识面，对测试工程师的要求需要了解和掌握很多方面的知识，需要了解问题可能出现的原因，已经出现这个问题可能是由于什么原因造成的，以便我们能够及时的补充测试案例，保证或者降低产推出的风险。

目前软件测试行业发展还不成熟，大多数系统测试都在测试组做，而且测试组几乎到系统测试测试阶段才会接触到产品。我们也把系统

测试简单的说明一下。

目前系统测试基本上采用黑盒测试方法，而且基本上局限在手公测试上面。



从上面简单的图形可以看出来，我们不知道被测试软件是怎么实现的，做了什么事情，我们只知道我们要它做什么，我们想得到什么，至于程序内部怎么实现，我们并不关心，我们只是关心结果。这是一种纯粹黑盒的测试。

这个阶段是测试发现问题的主要阶段，最少从目前市场上的产品情况来看是这样的。在这个阶段 60%的问题会暴露出来，如果不进行单元测试和集成测试，这个阶段的测试量和测试点很重要。

黑盒测试的核心是需要找到测试的重点在那里？测试的切入点在那里。系统测试重复的工作量比较大，而且如果是一个大型的项目，涉及的内容相对比较多，而且如果组织不好，或者没有找到重点，需要一遍遍的重复。所以需要自动化测试的需要合理的设计，使得我们的重复工作尽量减少，以提高工作效率。

验收测试

验收测试类似于客户验证产品的质量，在软件行业发展的过程中，各种承包项目类似于国外的外包项目将会不断的出现，那么外包项目的质量问题需要大家共同讨论。

外包项目的操作流程是当承包方提出具体的需求，然后有承包商来按照需求来开发项目，包括单元测试，系统测试，集成测试等各个方面的测试，经过被承包商测试后的产品提交给外包商的时候，需要进行验收测试，验收测试可以是外包商本身提供一套测试方案，然后对照具体的需求，进行产品验证测试。也可以是双方找一个共同的第三方，进行产品的验证测试。

验收测试的测试重点主要是产品是否按照需求开发的，而不从针对功能进行的测试。所以验收测试基本上不需要多少专业水平，也可以是承包商找到使用该产品的用户，来体验该产品是否能够满足使用要求。这样以来使得双方可以有一个共同的平台，避免商业矛盾的产生。

验收测试的测试手段目前来说还是靠用户体验。对照合同的需求进行测试，是第三方按照双方达成的共识来跟踪和测试软件是否能达成的需求。

测试方法

黑盒测试

顾名思义，黑盒就是外面不知道盒子里面在作什么，怎么作，只知道我的输入他需要有反应，无论是正确的还是错误的，都需要有回馈信息。黑盒测试需要懂产品的使用方法，操作方法，把尽可能多的情况暴露出来，通过这种方法进行测试。

黑盒测试的随机性比较大，在大部分案例执行完成以后，大概能够测试 40%的功能，据美国一个官方的数据说，20%的问题是在开发过程中发现的，80%的问题是在系统测试和集成测试过程中发现的，其中 80%的比例我们还是需要在细分，20%的是使用的问题，20%是程序的问题，5%逻辑问题,剩下的都是莫名其妙的问题，这样的数据对测试的一个引导是：要想发现更多的问题，需要更多的思考，更多的组合。这样无畏的增加了很多工作量，人们在疲惫的执行着测试案例，渴望从中发现新的问题。

这样的案例设计思想使得我们在开发一个大型的产品或者延续性产品的时候，整个测试案例的延续性很差，重用性也很差。所以我们在这里需要纠正一个概念，黑盒测试不简单的使用，案例设计也不是无谓的组合。

那么如何设计好的测试案例呢？如何在开发过程中很好的结合 2/8 原则呢？当前包括以后，不可能出现一个积极完美的产品，一个错误也没有，但是我们作为软件工程师，肯定渴望自己参与开发的产

品稳定，易用，人们寄予无限的称赞，这是一种奢望，那么我们把这种奢望修改一下，就是渴望我们参与的产品，能够满足当前大多数人的需求，稳定是否更合理呢？

白盒测试

白盒测试是一种高技能的测试，它是一种基于源代码下的测试，这种测试要求对程序的要求很高，需要了解程序的构架，具体需求，以及一些编写程序的技巧，能够检查一些程序规范，指针，变量，数组越界等问题，使得问题在前期就暴露出来。

一般程序所容易犯的错误,没有定义变量,无效引用,野指针,超过数组下标,内存分配后没有删除等,无法调入循环体,函数本身没有析构,导致循环实效或者死循环.参数类型不匹配,调用系统的函数没有考虑到系统的兼容性等.

白盒测试一般是以单元或者模块为基础的。目前的做法是把他归结为开发的范畴，用转人或者兼职的人去看代码或者利用部分工具，例如 Rational 系列，Boundchecker 等工具，他们可以帮助人为的发现变量没有初始化，指针错误等。大大的减少了人力。

我下面讲讲 BoundChecker 最实用的东西。

例如：我们发现一个现象：有个软件再 Win98 下运行不起来，或者总是出现莫名其妙的错误，再 Win2000 下或者更高系统下运行正常。上面是一个现象，是我们再日常测试中经常遇到的现象,我们分析可能导致出错的原因：操作系统本身出错的原因，导致软件无法再此系

统下运行，另外一个原因：软件中用到了某些函数不支持此操作系统。还有一个原因，软件运行的硬件环境再此系统下不能满足

灰盒测试

灰盒测试是介于黑盒测试和白盒测试之间的一种测试.这个阶段的测试重点是各个组件之间的逻辑,这个时期的测试重点是各个 DLL 文件的参数和逻辑是否正确,测试的重点是 DLL 本身.然后采用桩驱动,把各个 DLL 或者函数按照一定的逻辑串起来,达到在产品还没有界面的情况下能看到一种既定情况下的结果输出.

灰盒测试的要求相对白盒测试来说要求相对较低,对测试案例的要求也相对较低,只要求能够检测 DLL 处理输入和输出的能力,异常情况下的处理而已.

测试方面

案例设计问题

分析：因为现在从总体上看，案例设计很细，但是重复和不必要的东西太多了，个人认为原因有三个：

第一、 设计案例的不了解产品设计的框架（从程序概念上讲）

第二、 案例的设计没有一个反馈，涵盖情况不知

第三、 开发产品质量意识淡薄，测试压力太大

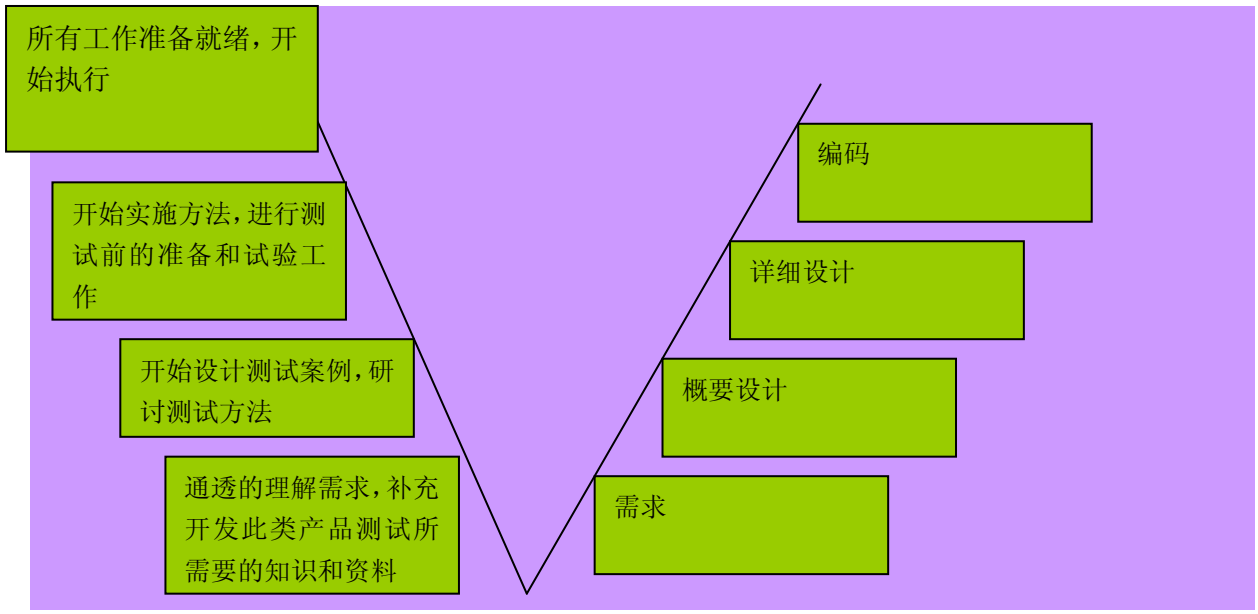
第四、 测试人员的素质分析没有，我们看不清问题出现在那里

进度问题

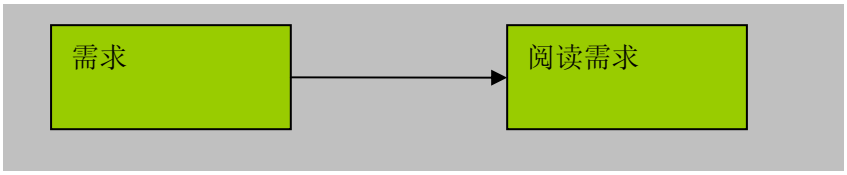
第一、 测试的整体计划里面没有重复考虑风险，时间问题紧迫

第二、 回归测试无法保证

结合开发模型，跟大家一起分析各个时期要作的时期



大家可以看到这个图和一般的书本上描述的不太相同，这是我结合具体的业务来分析。



怎么样阅读需求呢？

我们在测试的时候，我们需要通篇的阅读需求，那么怎么阅读需求呢？需要了解什么内容呢？实际的可操作的在那里呢？

我详细说我的认识，需求我们需要了解我们需要做什么类型的产品，这种产品需要什么样的基础知识，我们应该补充学习那些基础知识，市面上是否有同类型的或者相似的产品，他们曾经出了那些问题等，把自己先充实了，这是看需求的主要目的和重要目的。

测试改进方案

以上对存在的问题进行了分析，我们需要找到自己的弱项在那里，那么从现在看来，我们现在测试队伍没有建立，没有形成相应的体制。主要表现在一下几个方面：

测试工作需要回馈

回顾一，测试案例执行跟踪和统计不明确。

问题：如果测试案例不进行跟踪，无法证明或者检测我们案例设计的好坏，无法改进工作方法或者改善我们的思路，所以需要通过这里把自身问题看清楚，这样有利于工作的开展。

在我们日常的生活中，存在这一种现象，因为这种现象导致了测试一些列的发展。大家普遍认为，测试的含金量不高，导致了测试工作就是一些不愿意做开发或者没有能力做开发的人来做，其二，他们对测试设计的测试案例从不认真的审查，认为就那么回事情。出现这种问题的愿意是由于开发还没有清楚的认识到测试是一个服务部门，是为他们服务的，从私利的角度来讲，我们抛开项目的关系，测试的主要工作是为了帮助开发将自己写的代码更实用一些，让市场更认可一些，让开发人员的成就感强一些。如果大家都从这个角度考虑问题，那就可能缓解或者解决上面的第二个问题。

关于测试含金量不高的说法，我不赞成这个说法，在目前国内的大环境下，测试是这样的，但是它在朝自己预想的发展。而开发的发展除了新的语言在发展以外，思想或者体系我们能增加或者能设想的空间已经不多了，而对于测试是一个全新的行业，他发展首先需要支持，需要理解，我相信国内测试在 5~10 以后，发展更加迅猛。因为就算是现在很小的软件企业，已经开始重视测试了。

回顾二，测试需要和开发有共同语言

当你开心或者兴奋的发现问题以后，你能告诉我这个问题发生的原因吗？当你发现问题以后，你能告诉我问题可能在那个环节发生的？你能告诉我类似于此类问题可能在那里还会发生。

所以，当你进行测试的时候，渴望测试人员完全了解被测试对象的架构，然后针对此类软件需要补充基础知识，把自己补充起来，不至于开发人员给你讲任何事情，你不理解，或者很难理解，那么如果真的是这样，我对你个人设计的测试案例会打一个问号。

靠自己的基础知识，详细拜读设计稿件，从设计稿件中如果能发现问题或者风险，你就有长足的进步了。

回顾三，补充测试案例

我相信大家都有这个体会，在设计案例的过程中，大家想到这里，想到那里，总之想的很详细，但是在真正做测试工作的时候，总是发现一些 bug 与我们设计的测试案例无关。

怎么会发生这样的事情呢？因为我们设计案例是寄予自己的

经验和对软件的理解去设计案例，势必会造成这样的局面。现在我推进一种方法。就是在设计测试案例的时候，渴望每个人把自己负责的模块划个流程图出来，包括所有的出口和入口，包括信息流怎么流转的，如果把这张图形能够完全的划出来，说明你的理解要深一步，那么设计的案例含金量会高。

测试工作需要总结

测试的总结机制没有

i. 测试案例的执行情况
ii. 测试案例发现问题情况
iii. 测试案例的冗余情况
iv. 测试周期内的曲线项目进展情况

需要交流平台和形式

信息交流平台和积累

v. 资源共享
vi. 信息共享
vii. 提高自己在开发中的信心，不要总是喊狼来了
viii. 人和人之间需要沟通和认同，团体也一样

测试人员交流什么呢？

在一个组织中，应该让所有的人熟悉每个模块的设计思路和测试思想，让每个设计的人告诉大家，宣讲出来，这样大家在看这块的时

候，就知道那里容易出问题，出了那些问题。如果进行测试是最有效的，如果设计案例能抓住问题的核心。在设计阶段，如果把设计的案例给开发人员看看，能规避一些设计上的缺陷。

在应该团体中大家都应该有共享的概念，我个人推荐的学习方法，是偷，我别人学了很多年的思想精华部分，在很短的时间内接受并吸收，这样会提高整个团队的素质。如果每个人都在共享，那么每个人都在进步，所以需要交流，包括思想，包括方法等。

采用的方法

让别人给服务说话，清楚认识自己

让开发人员说话，让对应开发人员给我们的测试案例提出相应的意见，保证测试案例的覆盖面，以把握重点。

在整个开发过程中，由需求，开发，测试完整的团队，准确的说还有市场部分，我们都把它归结为需求的搜索和定义部分。那么在整个产品研发的过程中，各个部分需要完整的配合，否则整个产品都不能按时上市。作为为开发和需求服务的测试部分，应该摆正自己的位置，我们是一个团队中的一部分，是不可以缺少的一部分。

人贵有自知，也难有自知。只有在认识自己的基础上才能选择好自己的生活道路。首先要认清自己的能力。人的能力可以有天壤之别，但只要不辜负自己这块材料，也就可以问心无愧了。认识自己尤忌自大，这会使你为自己订立高不可攀的奋斗目标，到

头来高不成、低不就。其次要认识自己的本性。心理学家把人分成六个类型：经济型、理论型、社会型、审美型、宗教型和权力型。要选择一个适合自己本性的生活目标。

看清楚了自己，就可以很好的改善，也能把自己的事情做好，同时呢，才能更好的服务。

自己回头看

让执行测试案例的人员反馈给我们数据，说明案例的冗余情况，这样会慢慢提高自己的设计水平。

因为人们习惯于谈成绩，问题在成绩中可以淡化，我不同意此观点。

其实在现实生活中，大家都经历了很多事情，都学会了总结，可是同样的错误在现实中会多次出现，为什么呢？是因为回头了多次，没有总结，总结了没有执行，执行了没有改变方式，改变方式了但是没有认真考虑，还是错的。

把自己犯的错误列举出来，然后找出出现问题的真正原因，才是自己最大的进步。如果淡化错误，将来可能就会将成绩磨灭掉，所以积累，回头是工作中需要重视的问题。

还有一种论点，说公司多么多么重视开发，不重视测试，我对这种论调积极反感，这只是个人感觉。为什么这么说呢？

对公司来说，要靠项目和产品维持生存，对吗？从这个方面来说开发重要，产品质量不重要吗？这个问题很多人问我，我回答

说，重要，非常重要，那为什么测试的价值体现不出来呢？主要是两个方面的原因，一个是公司引导不正确，各个部分的同事为这个项目负责，而不是开发为这个项目负责，其二呢，主要是因为我们是维护，而不是创造，如果你告诉老板，这个产品我们改变测试策略，能够提高工作效率，这个产品可以提前 2 个月发布，而且我保证质量。我相信你的价值也即体现出来了，如果不可以，说明还是没有找到合适的方法。

了解同类产品

让市场人员反馈同类产品的问题以及市场对我们产品的需求。测试过程是反映当前产品的质量，为什么要研究竞争对手的产品呢？

首先，测试中包含易用性测试，测试什么内容呢？就是测试怎么好用，客户是怎么用的，我们怎么设计更贴近用户，那么不研究竞争对手，我们怎么可能占领上风。

其次，了解竞争对手的产品，有利于测试工作捕捉重点，使得工作开展有利有节。

可谓知己知彼，百战不殆，所以在现在的市场竞争中，了解同类产品才可能发现对方的缺点，给以打击，发现对方的优点，快速学习，闭门造车必定失败。

提高自身素质

从程序的概念理解产品，这样测试案例可以设计的比较有针对性。常言说得好，“识重于才”，而见识却往往是生活阅历造就的。对于一个初出茅庐的人，智者的指点是至关重要有时甚至是决定性的。回想我十年来的经历，很多失败其实是没有人指点而造成的。要寻找一个精神上的导师，他可以是你的父母，也可以是其他师长。他阅历丰富而又不拘泥于自己的老经验；他能在紧要关头给予你原则上的指导和精神上的支持。有时候仅仅是他失败的经验就会使你受益匪浅。

如何提高程序能力

耳濡目染

让开发或者设计人员在讨论开发方案的时候参与旁听，耳濡目染。其实这只是一种辅助的手段。

电视剧《霍元甲》播出以后，得到大家的欣赏。原因是因为他本人身体虚弱，所以父亲从小不让练武功，而生长在那样的环境中，他天天可以看到兄弟们在练功，招式已经记忆在心理，但是苦在没有练功的机会，他利用体力劳动的过程中，改变劳动方式，趁机练功，后来发展到独创“迷综拳”。

程序设计和开发是一个硬功夫，也是一个长远的事情，它是一个积累的过程，不能一蹴而就，需要苦心练，多些理解，多些思

考。

面对程序开发，不要有太多的压力，因为程序开发就跟你学说话一样，因为语言本身有很多通性，高级语言和低级语言本质上差别不大，所以扎实的从基础的东西学起，这样才能完全的积累下来。

计算机发展速度很快，各种概念，各种语言发展都很快，掌握实质，不断学习，才能把握。所以还是需要多看，多想，多练。

自己练内功

从自身做起，了解程序架构和开发模式，努力提高理解和产品的单元测试或者组件测试能力，这样以来可以了解程序的很多算法，使得在产品的开发过程中就能把问题发现并且能够得到及时的解决。

其次能够提高大家参与到项目的荣誉感，因为在测试本身是一个服务性的行业，那么服务行业的特点是不停的改变思路，改变服务模式，提高服务质量，当服务做好了，那么在整个研发中就可以找到自己也是其中一个分子的感觉。

其三，练好内功，为自己将来提高工作效率，进行一些自动测试以及从程序架构的概念上设计测试案例提供了技术保障。

以上是自己练好内功的用途。

在过去社会中，有很多擂台赛，目的是切磋技艺，弘扬中华武术，各个门派直接交流和学习的过程，为了在擂台赛中取的很好

的成绩，我们需要努力练功，其次是多学本门派和其他门派的武功，或者自创武功，在擂台上能够发挥的淋漓尽致，因为武功的最高境界就是没有招式，要达到这个境界，需要内功深厚，避免走火入魔，需要毅力，需要创新。

理论就是理论，无论在那里看到的理论都是一定的基础的，因为所有的理论基础需要一个证明此理论的平台或者条件，所有一定要看，想，用。看别人是怎么用的，在什么情况下用的，用的目的是解决什么问题，在什么样的环境下能够做出来，需要什么样的支撑；想自己现在目前是否有这个环境，就目前的环境能够做什么，如果要搭建对方的环境需要多长时间，这个做法中存在什么不托的地方，有什么需要改进的地方；在自己工作的环节中找找看，看自己是否适合用这个东西，如果适合，怎么用，用到什么程度，如果非常认可别人的做法，需要衡量需要多少资源和时间，努力找自己的结合点。

千万不要再我们看到一个理论或者方法的时候就去推动它，或者原理实践过一个什么思想就想在新的环境下实践他，都是不可取的。好的事情或者好的做事方式他需要一些条件支撑，一旦硬套，就可能出现问题。

实践中检验

尝试做一些灰盒测试部分（目前暂时是想法，但是还不完善）。灰盒测试是界与白盒测试和黑盒测试之间的一种临街状态。

测试发展

测试在国内还是处于摸索阶段，在过去的发展阶段，大家只是初步针对不同的软件产生了不同的测试方式，但在操作方法，操作流程等方面还需要继续摸索。对潜入式软件来说，行业内始终认为潜入式软件是最难进行测试的，因为他需要很广的知识面，需要对各个点的设计原理进行分析和测试。

在目前国内开发眼中的测试还没有形成概念，我们需要不断的改变形象，加深他们对测试的印象，以便我们获取更多的帮助和协助。

测试未来发展需要两条腿走路，这样能够在各个环节保证产品的质量。

第一步，系统测试继续练内功，将案例设计的能力提高

第二步，需要进行灰盒测试，对产品进行代码级的测试

第三步，需要进行部分白盒测试或者由开发人员进行执行

要达到一定的认同和发展,测试人员需要努力学习,打下扎实基础,这样才能一步步的成功.

如何提高测试

提高测试需要从几个方面着手，其实只是自己的一些感觉，不一定就需要按部就班，需要找自己适合的点。

制定完备的测试计划

清楚的认识测试计划，测试计划是一个文档，能够保证整个研

发过程中顺利执行的一个指导性文档，它描述了几个方面的问题。

第一、　描述了项目的目的

第二、　描述了项目的开发周期

第三、　描述了在测试中遇到的技术

第四、　描述了测试案例的设计周期

第五、　描述测试案例的执行周期

第六、　描述了测试过程中用到的工具或者技术

第七、　描述了测试过程中用到的资源情况

第八、　描述了测试过程中可能遇到的风险以及规避方法

提高案例设计水平

明确了解现在目前流行切实用的几种案例设计的方法，因为在不同的产品不同的要求有不同的设计手段，我们需要不断的学习和总结，在为了测试领域中，许多新鲜的词语都会出现。

这种方法类似与工业领域的随即抽取统计分析法，但是工业性质牵扯到损坏或者人为原因，统计出来存在这偏差，但是应用与软件方面，虽然存在着偏差，但是不可能象硬件那么偏差很高。

等效法

明确测试的目标，一般适合用到的范围是，制定被测试的对象是在满足某个条件的区间内的所有的所有数据。

案例设计方法：从其中区间数据段中选择任意一个或者两个数据，只

要这个数据满足了，那么其他的数据就是满足的。

我现在举一些例子，来说明等效法在测试过程中如何应用的。

范例 1：在登陆某系统需要验证用户名，要求是长度是最小是 6 位，最长是 14 位，名字中可以包含数字，但是不能以数字开头，可以包含各种符号，不能包含中文。

1、随意字母组合成一个 12 位的姓名，测试是否可以通过验证。

2.、随意生成一个长度 12 位的姓名，测试是否可以通过验证

3、测试以任意一个数字打头 12 位的姓名，测试是否可以通过验证

4、测试姓名长度位 12 位且包含中文情况，测试是否可以通过验证

5、测试长度不满足条件情况下，是否通过验证

6、如果长度不满足，是以数字开头的，提示信息验证

7、如果长度不满足，姓名中包含中文的，提示信息验证

……

(注：)这个可能比较简单，但是说明一个问题：为什么随意生成一个 12 位姓名的,其实你选择 8 位姓名长度或者 10 位姓名长度是一样的，所以这种情况下考虑采用等效方法比较合适。

范例 2:有这么一个需求，要求选择 1~12 之间进行调整，手机的背光就会随着数值的变化而变化。总体的是数值越大越暗。

以上需求是大家经常可以看到的。

测试案例设计：清晰记忆 1 的情况，然后随意调整一个数值，因为要求是变化了，至于变化成什么样子，变暗到什么程度才正确，没有明确的指标数值，所以只需要记住临街点 1 的情况，然后随意调整一个

数据，然后和当前调整后的数据进行比较。

（注：）没有明确的说明，只是含糊的结果，但是总体的结果是在变化，那么这个时候比较适合使用等效法。

因果分析法

需要有一定的程序基础，了解程序的架构，就是当问题发生以后，能够有效的补充相关的案例或者筛选相关的案例。因果分析的核心是从自己的理解去分析问题所在的真正原因。

范例 1：删除磁盘上某个文件失败，分析原因：如果是管理员权限，那么可以随意删除，无论这个文件的属性是只读的还是存档的，那么如果不能删除磁盘文件，除非是坏道上的文件。分析完成以后，使得测试案例设计有针对性，而不是盲目的将所有的文件格式都去尝试一次。

范例 2：假设我们用 Excel 作一个计算，结果和我们用计算器计算的结果不同。

分析：Excel 的计算函数单独运算没有错误，然后插入一行，结果错误了，说明插入行导致计算错误，那么插入一行怎么会引起函数计算错误呢？原因是由于插入行后，导致传给计算函数的区域没有更新，所以造成计算结果错误，那么这个 Bug 就很明确了。

范例 3:假设我们平常在做讲座的时候发现在某台机器上就会死机。这是一种现象。

分析：为什么在这台机器上死，在其他机器上不死。原因有两个，

第一个先找系统原因，是否是我们的产品在当前这个系统下有 Bug, 经过验证没有，那问题出在那里？

其实演示产品需要的是硬件的支持，那就是显卡，如果显卡内存不够大，可能导致某些演示文件死。

（注）因果分析需要有广泛的知识面，使得我们在分析的时候能够拓宽面积，模糊的定位问题。

范例 4:用户给我发送一个文件，打印的时候发现是乱码。后来逼迫无奈，就让用户将这个文件传真给我。这是现象。

分析：为什么打印出现乱码？问题基本定位，系统字库不够，系统下打印驱动问题，打印虚拟内存问题，操作系统问题，软件本身问题？最后问题经过验证，最终归结为在此操作系统下，打印驱动程序有问题，使得文件不能正常打印。

（注：问题需要先框定范围，不要乱了套路。）

逻辑分析法

在逻辑分析方面，也需要有一定的程序理解能力。从程序逻辑和日常常识去判断问题。逻辑分析法其实就一堆假设的罗列，推论出系列结果的假设，然后将假设反推翻，问题就可以暴露出来。无论那种方法都是通过表现去分析问题的实质的。

范例 1:我们在做 MP3 播放器快进和快退测试中，要考虑的同步问题，就是我们液晶显示屏上出现的歌词进度，时间进度和我们耳朵听到的进度不同。我们分析一下，为什么出现不同步现象，为什么其他

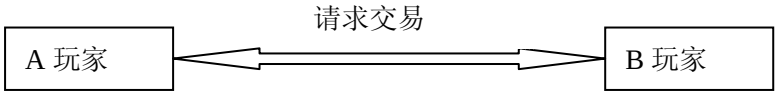
的能同步，就某一个或者某几个不能同步。

首先我们了解同步的算法：快进和快退是按照当前歌曲的数据流来计算应该到那里，它是以当前歌曲的数据流为系数，然后进行的一些调整，那么出现不同步的原因是由于系数不同造成的，所以考虑到同步问题，我们需要找不同格式不同数据流的歌曲，这样问题容易暴露，容易清楚的定位问题的真正原因。

范例 2

我们来分析网络游戏中的交易系统，就是在游戏两个人进行物品和金钱的交易。

怎么设计这里的案例呢？我们先梳理一下交易的逻辑关系图。



边界数值分析法

在测试案例执行的过程中，所有调节的数据都需要考虑到边界数值的测试方法，这里我就不在赘述。但是需要注意，边界数值的测试不是枚举，只是抽样的方法。

逃避测试的误区

市场需求引导产品质量

测试是为了验证需求，保证产品质量，无论如何你都不可能做成 100%的测试，不可能做成 No Errors。所以我们针对不同的产品，不同的市场定位，确定不同的测试方针。

因为企业面对的是客户，面对是企业长远利益，那么我们不可能仓促的推出产品为了迎合市场，而是需要研究，调查市场的真正需求，把用户所关心的功能提供给用户，使得其更加完善，更加稳定。

我们从企业来分析，首先任何一家企业要生存，必须需要市场空间的支撑，目的是为了盈利，我觉得没有必要说的那么冠冕堂皇，这是事实，但是在把握产品质量和市场需求的时候，我相信很多企业会选择市场需求的，因为这是机会，是把握企业生存的机会，特别是对于发展性企业来说。**（企业原因）**

我们从开发来分析，因为在开发的过程中，由于软件行业的高流动行和知识更新快的特点，风险加大，使得开发周期很难把握，这样使得产品测试时间很难控制。因为开发的进度包括市场提出需求的技术风险都很难把握。**（开发的原因）**

我们从测试来分析，测试在很多企业中是没有的，那么开发人员自己来做，如果有测试人员，那测试也是随意性非常强，造成产品上市后预留很多无法预估的风险，为企业的形象蒙上了面纱

（测试模式）

合理利用 2/8 原则

测试是列举，不是枚举，所以设计案例的时候全面是不可能的，那么需要灵活的运用 2/8 原则，使得测试重点清楚，容易控制。

基于产品在开发过程中的种种风险，我们在有限的人力和资源的情况下，合理的利用 2/8 原则，如何把握 2/8 原则？首先需要了解产品的特点，让所有参与测试的人员能够了解产品的特点，这样使得工作具有针对性，至于产品的噱头，我们可以进行充足的测试，因为只是我们的产品立足市场的点。

在时间有限的情况下，把常用的功能测试保证了，不要摊全，摊宽，这样到最后都无法总计产品的质量概念了。

以上这么说，是一种概况，在实际的工作中大家需要总结，把进度，时间，质量等进行权衡，以保证产品的顺利发布。

回归测试的概念

测试次数不是轮回，测试的不同次数不是轮回，而是为了验证问题，那么什么时候适合安排一轮测试，需要定义标准，否则耗时耗力。

回归测试是不可缺少的环节，在一个产品测试完成后，直接到用户手头的时候，需要千万小心，需要进行一次彻底的回归测试，这个时候包括所有的功能以及所有已经修正的问题。避免版本出

现问题。

其实在不同的资料中对回归测试有不同的解释，我就不在这里赘述。我想表明我的观点是，依照不同的开发模式，回归测试所在的时间段也不相同；当前的开发模式有瀑布型和迭代型，例如，在瀑布型的开发模式中，所有的测试活动（手工测试，系统测试，部分集成测试）都在最后进行的，而切所理解的回顾测试是为了保证在新的版本中测试修改后的问题，其实这个测试只是保证了其中一部分工作

测试的概念

测试不是为了验证问题，而是为了发现以前设计中没有发现的问题。

自动测试只是测试的一种手段，目的是为了提高工作效率

测试工具只是利用，不能依靠，因为工具本身没有智能的判断是否会有问题发生，自动测试不是利于测试工具，而是需要编写或者利于测试平台，编写适合自己的测试工作进展。

如何调整团队的作战能力

建议性质：因为曾经带过四个团队，而且这个经验最少在我身上是成功的。

形式分析

测试团队，测试团队在现在国内来说在慢慢的得到重视，之所以原来不重视是因为整个行业处于摸索期，不知道采用什么方法，什么技术，作什么事情等的情况下，使得测试员好像是一些没有能力人的集合（宣讲，不听的宣讲）。

目标计划引导

测试技术和未来发展规划，因为任何人的发展需要目标，那么一个人的发展目标假如它和这个行业相关，那么它会付出一切，努力的工作，所以需要大家认可一个目标，并且让大家认为是可行的，然后我们分步骤一步步的去实现它。让他或者大家能够看到自己所喜欢或者从事行业的发展方向。

过过老师瘾

因为在做任何事情的时候，每个人都有自己的想法或者步骤，讲出来就好，这就需要开始的时候我们以任务的形式下达，我相信，到后来大家愿意自己站出来讲了，我告诉你原因。因为人本身有羞怯感，怕几个方面，怕讲错，怕人多，怕提问。那么如果把这几个问题都解决了，是否羞怯感就没有了呢？

如何解决个人怕的问题：引导，因为一个人如果不能把自己的想法和思路讲出来，那么不可能把事情做的很好，其二，就是如果你把你的想法说出来，别人可能会指出你思路中走弯

路的地方，对个人来说可以跳高工作效率，使得思路更加完善。

其三，如果大家都把自己的思路说出来了，你不就节省的很多学习时间吗，另外你想过没有，当别人形成这个想法的时候，需要一定的积累，那是他的心血，这不就轻轻松松让你学到了吗？如果固步自封，那么你的思路有可能是错的，有可能是对的，但是你的知识面就只能局限在你所考虑的范围内，对个人发展不利。

定学习目标

在软件行业里面，要有发展，就需要不停的学习，不停的进步，不停的总结，才可能有长远发展，所以需要定义在这个行业阶段行的学习目标，让人感觉这个行业现有的水平只是维持，要发展，需要学习。

在工作中学习的方法，除了自学以外，就是“偷”了，所谓偷，就是要学会问问题，把你想知道的东西刨根问底，当别人回答你问题的时候，他一定是用他知道的东西的精华来总结，那么这样你在很短的时间内，把他总结的精华全给你了。

在学习的过程中，需要学会总结，把能总结的都整理出来，第一是经验的积累，第二呢能够做到分门别类，逐类旁通，使得相同或者类似的错误不要重范。

兴趣和爱好

一个人工作有两种情况，第一中是真正的工作，完成就算完成了，自己也在不断的学习，不断的总结，但是缺乏激情。第二中是把工作当成自己的事业，渴望自己在这个方面成为权威或者说业界能够说话的人，也是在不断学习，不断的总结，培养职业性，培养和引导大家的兴趣和爱好，因为只有你了解了兴趣和爱好，才能更融洽的调和整个工作组的气氛，这对测试行业的领导者来说是个挑战。

歪曲理论推理

“测试人员是由于技术不过硬，才去做测试的。”针对这个观点，我说说自己的看法。好，我给测试说几句，测试水平不过硬成立，假设成立，那么这是相对的，相对开发来说的，而且这种论调都是从开发那里扩散或传播出来的，测试在后期发现问题后，开发也许心理很痛苦，但是他不愿意暴露在脸上，使得有些问题越发变的严重，不得不修改。那么在产品后期暴露那么多问题，说明了什么呢？这么低水平的测试都能考虑到你程序设计的种种漏洞，那么说明了程序水平开发有待提高。

在 IT 现在的行业中，开发的流程，模式，以及各种约定成俗的东西越来越多，而且相对稳定，而测试是一个全新的行业，它需要大家摸索支持，需要大家共同建立起来。

其实，在原来的开发模式中，商家为了适应市场，为了保证利润的最大化，为了使得产品能够顺利的适应市场，那么采用各种方法，

使得产品质量的定位淡薄，而现在随着人们的要求越来越高，商家的意识越来越强，各个公司或者组织渴望成立测试部门，保证产品的质量，使得测试这个行业在最近几年才发展起来。

正确理解自动测试

首先，自动化测试是测试行业是技术，但是不是说用了自动化测试了就能发现更多的问题,它只是提高了工作效率而已.

自动化的概念是人们在工业生产的过程中，为了提高工作效率，不断的对操作方法或者技术或者工具进行改进，减少人们普遍的手工劳动，节省时间和成本。

而软件行业的自动化测试同样也有节约成本，提高效率的需求。所以所有的改进需要考虑到成本的问题。

自动化测试的大前提:

第一. 产品本身特征具有长期可维护性

第二. 产品本身非紧迫的大项目

第三. 产品结构相对复杂

第四. 资源投入相对充裕

那么作为成本，需要从几个方面去考虑，第一实现成本，第二，人力成本，第三，新技术的风险，第四，节省的成本，第五，被自动化的功能是否需要大量的手工劳动。

所以我们理解自动化一定从成本的概念上考虑， 最少从自动化测试概念的起步应该从这个方面考虑。那么自动化测试的重点就在于他

节省人力，节省时间，得到的数据更精确些，而且操作的可重复性和Bug 的可重现性更强一些。

下面我就对自动化测试做一个详细的解释，跟大家分享。

1. 简介

本文关注于一个实施自动化测试框架的组织的主要方面和影响。本文的意图是提供一些能够成功的实施自动化测试的指导方针。

2. 测试自动化的神话

有很多关于自动化测试的神话。其中的一些是真实的，而其他的一些是不正确的设想，这些不正确的设想会严重的威胁到实施自动化测试的成功。

2.1. 我们在时间上是紧迫的 － 项目已经落后了 － 让我们使用自动化测试吧！这种情况将不能成为现实。实际上，正确的思想应该是 － 我们时间急迫 － 我们决不应该使用自动化测试。

如果项目已经陷入到了麻烦之中，不建议实施自动化的功能测试。项目很可能因为需要大量的测试框架的准备和实施会被托跨。我饿建议将重点放在以下的事情上：

优化测试的过程。调查并建议在目前工作基础上的测试方法和过程。建议借鉴 RUP 的相关思想和过程。引进或者使单元/组件测试正式化。这是我们能够快速获得受益的很好的方法。如果正式的组件测试被使用，我建议可以使用 Rational PurifyPlus 进行单元或者组件测试。根据我的经验尽早的使用 Rational PurifyPlus 是非常值得的。在一个引入和 Rational PurifyPlus 的项目中，通常会在组件的级别得到

百分之三十的性能提升。仅仅在项目团队能够对 下列问题的回答是"Yes"时：项目能够被适当的推延。存在能够通过实施自动化测试被达到的精确的目标。项目具备建立适当的测试框架的必要条件。

那么，你可以在一个时间紧迫的项目中适当的实施测试自动化。但是根据经验这种情况是很难发生的。总而言之，我只能说"对不起，银弹根本不存在"。

2.2. 测试自动化就是捕获和回放

在过去的日子中，自动化的测试工具只是被看作是一种捕获和回放的工具。当前这个神话仍然在很多测试人员的思想中。而事实上自动化测试已经远不止捕获和回放这么简单了。按照成熟度自动化的测试可以被划分为 5 个级别。

2.2.1. 级别 1：捕获和回放

这是使用自动化测试的最低的级别，同时这并不是自动化测试最有用的使用方式。优点：自动化的测试脚本能够被自动的生成，而不需要有任何的编程知识。缺点：你会拥有大量的测试脚本，同时当需求胡子和应用发生变化时相应的测试脚本也必须被重新录制。用法：当测试的系统不会发生变化时 — 小规模的自动化。

2.2.2. 级别 2：捕获、编辑和回放

在这个级别中，你使用自动化的测试工具来捕获你想要测试的功能。将测试脚本中的任何写死的测试数据，比如名字、帐号等等，从测试脚本的代码中完全删除，并将他们转换成为变量。优点：测试脚本开始变得更加的完善和灵活，并且可以大大的减少脚本的数量和维护的

工作。缺点：需要一定的编知识。频繁的变化可能会引起"意大利面条式的代码"，并且变更和维护几乎是不可能的。用法：当进行回归测试时，被测试的应用有很小的变化，比如仅仅是针对计算的代码变化，但是没有关于 GUI 界面的变化。你能够使用这种技术通过快速的编制一些测试脚本以检验你的想法来探索你的预定的测试设计。当我在没有任何象需求或者设计模型这样的文档的情况下第一次操作一个产品时和我需要获得一系列内部构建版本的稳定性的第一印象时，我使用过这种技术。通常如果适当的软件配置管理（SCM）与良好的内建设计相结合时，使用级别 2 的技术已经足够了。

2.2.3. 级别 3：编程和回放

这个级别是面对多个构建版本的有效使用测试自动化的第一个级别。你需要在实际的投资开始显现之前确保团队和客户对项目的安全感。如果没有对测试自动化工具的适当的培训测试人员将不具备到达这个级别的能力。在自动化测试工具中的所有测试功能都必须被很好的理解，并且要掌握测试脚本语言的知识。好处：你确定了测试脚本的设计。适当的设计是必要的。编码的习惯必须是适当的。使用与开发中相同的编码习惯是非常好的。这将开始搭建起测试和开发之间的桥梁。在项目的早期就可以开始自动化的测试。你能够在项目的早期就开始进行测试脚本的设计。与开发人员交并调查他们认为可能会存在问题的区域。确保了开发人员关注在获得能够被测试的方案上。缺点：要求测试人员具有很好的软件技能，包括设计、开发等。用法：大规模的测试套件被开发、执行和维护的专业自动化测试。级别 3 使你

能够使用自动化测试并构建不同的回归测试（重用已有的自动化测试用例）。根据我的经验在看到更多切实的回报之前，为了达到这个级别，有大量的工作和影响项目的活动必须被做。因此快速的建立和证明自动化测试的价值是至关重要的。找到乏味的测试（例如，边缘测试和特定的功能测试用例是首先进行自动化测试的良好候选者）。首先创建少量的能够测试一些基本功能（比如，登陆和创建用户等）的测自动化测试用例。

2.2.4. 级别 4：数据驱动的测试

对于自动化测试来说这是一个专业的测试级别。你现在要利用测试工具提供的所有的测试功能。你拥有一个强大的测试框架，这个测试框架是基于能够使你根据被测试系统的变化快速创建一个测试脚本的测试功能库的。维护的成本相对是比较低的。你在你的测试中会使用到大量真实的数据。优点：你能够维护和使用良好的并且有效的模拟真实生活中数据的测试数据。缺点：软件开发的技能是基础，并且需要访问相关的测试数据。用法：大规模的测试套件被开发、执行和维护的专业自动化测试。级别 4 要求一些非常良好的测试数据。一个测试人员必须要花费一些时间来识别在哪里收集数据和收集哪些数据。使用现实生活中的数据是最基本的以从测试中得到完全的回报。使用适当的数据将为你提供通常仅仅在项目的后期才会发现的或者是有客户发现的错误的能力。现在你能够通过使用现实的数据开运行大量的测试。

2.2.5. 级别 5：使用动作词的测试自动化

这是自动化测试的最高级别。主要的思想是将测试用例从测试工具中分离出来。这个级别要求有一个具有高技能测试人员测小的团队，这些测试人员能够将测试工具的非常深层次的知识与他们具备的较深的编程能力结合起来。这个团队负责在测试工具中生成并维护测试的功能性，能够使测试工具从外部的来源，比如 excel 表或者数据库中执行测试用例。这种测试概念最初是由 CMG 开发的。与 CMG 方案相比，其他的可能的开放源码的方案有被 Carl Nagle 和 SAS Institute 开发的 DDE。使用 DDE 的概念，关注点是当在 Excel 表中创建测试用例的时候，放置使用包括被使用的特定动作词语的一些类型的模板。执行的过程是从 Excel 表中读取测试用例，并将测试用例转换成为测试工具能够理解的形式，然后使用不同的测试功能来执行测试。这个概念变得越来越流，因为测试与用例一起使用是非常有用的。优点：测试用例的设计被从测试工具中分离了出来 — 关注在设计良好的测试用例上。允许快速的测试用例的执行和基于用例的更好的估计。缺点：需要一个具有工具技能和开发技能的测试团队，以提供并维护测试工程（框架）。用法：专业的测试自动化将技能的使用最优化的结合起来如果工具不具备使用内建的对象映射的可能性，那么这个方案对于消除与 GUI 相关的大部分维护成本是优秀的。在一些组织中已经创建了这种方案，并且他们其中的一些已经实现了高度的自动化（60%），并且他们都得到了巨大的回报。如果测试框架是适当的，我们能够使用 excel 来生成实际的测试用例。这个级别对于那些按照正规基础使用用例的组织或者项目来说是非常优秀的。有

多少测试用的估计是被需要的，并且当用例适当时需要做的工作也是非常简单的。你可以集中时间来生成第一个包含被需要的"对象映射"的测试用例（主流程）。依靠被测试应用的复杂程度，通常这会花费大约半天到一天的时间。后续的被需要的每一个测试用例大概会花费 15 到 20 分钟的时间，因为通常多数的测试用例可以复制已有的测试用例，并对其进行必要的修改，通常这种修改是有限的。动作词语框架能够通过使用用例使紧密的并行测试用例的开发变得可能。

2.3. 我们不需要培训！

我们所有的人都在某一些方面具有一定的经验，我们没有时间能够花费在使用新工具的培训上。当一个对自动化工具还不是很熟悉的组织或者项目团队开始实施自动化测试时，培训和指导是至关重要的。如果我们允许组织或者项目团队在没有关于应该如何做的任何知识的情况下实施自动化的测试，那将肯定会以失败告终。用于实施自动化测试方案的预算会被超出，测试会被延误并且更坏的情况是自动化测试将被放弃。组织和项目团队需要尽量避免一些认识上的缺陷，尤其是自动化测试的维护成本和当测试人员尝试和确认工具如何工作时产生的挫败感。你需要确保你的测试过程是适当的 — 如果测试过程是不合理的，引入自动化测试只会给软件组织或者项目团队带来更大的混乱。因此，我建议希望实施自动化测试方案的组织或者项目团队应该在实施之前建立"训练营"，并将重点放在培训测试团队能够很好的利用一个原型的项目上。为第一个原型项目制定一个实施计划，下面包括原型项目的最小化的描述：当先状态我们希望实现什么 — 建

立成功的因素期待的回报（第一次自动化测试工作被期望验证什么）找到一个"简单的"测试的痛处并尽力的通过自动化测试解决它，这可以被作为在同一时间上使测试运行在多个平台上的样例说明被需要的资源和时间.....

一开始你就要大声的说出成功的信心 — 让人们了解你所展示的进展。这将吸引更多的关注和资源。

2.4. 我们必须 100% 的自动化

从管理的角度来说，100% 的自动化目标只是一个从理论上可能达到的，但是实际上达到 100% 的自动化的代价是十分昂贵的。一个 40-60% 的利用自动化的程度已经是非常好的了。达到这个级别以上将增加测试相关的维护成本。由于对每一个构建版本的需求变化的复杂度，你将花费更多的时间在变更测试用例上以使他们能够正确的运行。在这种情况下，通过告知管理层 100% 的自动化目标是相当昂贵的来确立一个合理的期望值才是明智之举。对于决定自动化一个测试用例的一般规则是这个测试用例必须被运行 4 次以上。这个数字是基于用户对测试工具有良好的技能并且有一个良好的测试框架的。如果情况不是这样的化，整个数字能够是 10-20 次或者更高。一个例子，在一个项目中测试人员花费和两周的时间将手工测试的 23 天的任务转换成了自动化测试的用例。在完成使，项目能够在 4 个小时在多个平台上运行相同数量的测试用例。

2.5. 测试框架

测试框架对于产生成功的测试自动化的适当基础是重要的。很多考虑

必须被解决以使测试自动化更加有效地被使用。重点必须在：维护成本维护成本是成功的使用自动化测试的最重要的问题之一。维护成本直接联系到前面已经提到过的自动化测试的成熟度。组织或者项目必须至少要在成熟度的 3 级使用高度的测试库才能使维护和更新测试功能变得容易。

测试数据

什么样类型的数据将被使用？要为每一个测试用例生成测试数据还是使用在被测试应用中已有的数据。在很多的情况下一个测试数据被创建了，删除他们是不可能的。

可测试性

自动化测试方案能够有效的测试吗？例如，被适当命名的对象（不仅仅是索引 Id）。一个简单的例子是所有的对话框都有相同的 #id 和相同的标题，所不同的仅仅是显示的文字信息。当测试应该覆盖多种语言的方案时，对话框的测试就是一个挑战。

测试人员的技能

被包括在自动化测试的创建中的人员应该具有什么样的技能呢？如果他们具有良好的开发背景，那么成熟度 3 级是足够了。如果他们有很少的或者根本没有开发的经验，我们被迫使找到或者培训一个自动化测试专家的小组，并直接到达成熟度 5 级，在成熟度 5 级测试的创建与实际的测试执行被分离开。

一个好的构建过程

自动化测试的引入在"构建团队"上加入了一些约束。为了实现自动化

测试的高利用率（回归测试），要求具有一个高的构建频率。每周仅仅运行自动化的测试不是好的自动化测试的使用率。将回归测试增加到每天一次将帮助快速的发现新的问题并使开发人员更加容易的发现问题的根源，因为对测试的反馈时间是比较短的（开发人员能够记住他们昨天做了什么）。所有权不同的测试库的所有权的定义是重要的。一个好的方案会将测试库的组织划分为三个级别：

级别 1 — 全局的

这个一个通常的级别。被存储在这个级别的测试功能能够被所有的项目访问。通用的和通常的功能象登陆、创建一个用户都是这个级别很好的候选者。

级别 2 — 项目

在这个级别的测试功能是与特定的测试项目相关的，但是通常在项目中有用的比一定在项目外是有用的。通常级别 2 是级别 1 的功能的提供者。

级别 3 — 脚本

功能被直接关联到特定的测试脚本。 I 在这个级别中，通常一个测试功能的第一个版本是被开发的。在新的测试脚本的创建期间已有测试功能的重用性被发现，并被移到了级别 2 中。在这个级别上尽量最小化功能的数量，因为它将增加维护工作量。还有很多有关测试框架的问题，但是这里所提及的是一些基本的要被解决的问题。

3. 在哪里使用自动化测试

有很多的情况下使用自动化的测试可以降低测试成本。我将尽量的突

出在自动化测试中的不同的测试技术技术 描述 备注单元测试/组件测试 这个测试工作通常是开发人员的职责，很多不同的方法能够被使用，比如"测试先行"，它是一个测试框架，开发人员在编写代码前编写不同的单元测试。当测试通过时，代码也被完成了。 通过使用正式的单元测试，不仅能够帮助开发人员产出更加稳定的代码而且能够是软件的整体质量更加的好。冒烟测试 冒烟测试是一般验证别测试系统的功能性测试用例的集合。冒烟测试背后的思想是确保基础是可以工作的，以便"大的"测试工作能够开始。 在构建过程能够确保构建已经为测试准备好时，冒烟测试通常是自动化的运行。功能/集成测试 这里测试的工作关注在验证在不同的组件之间的集成上。 这些类型的测试通常是被测试系统的更加复杂测试的基础，大量的边缘测试被合并以制造出不同的错误处理测试。系统测试 — 用例测试 这种测试是通过执行用户场景模拟真实用户使用系统以证明系统具有被期望的功能的测试。 这里不需要进行自动化的测试。安装测试、安全性测试通常是有手工完成的，因为系统的环境是恒定不变的。回归测试 回归测试实际上是重复已经存在的测试。通常如果是手工完成的化，这种测试只在项目的结尾执行执行一到两次。 这里完全有潜力应用自动化的测试。你能够在每次构建完成后执行自动化的回归测试，以验证被测试系统的改变是否影响了系统的其他功能。性能测试 性能测试包括以下不同测试形式：

- 负载测试
- 压力测试

- 并发测试

-.....

如果没有自动化的测试工具，你将不能执行通过模拟用户的负载实现的高密集度的性能测试。

4. 什么时候使用自动化测试

我对什么时候应该使用自动化测试和什么时候应该使用手工测试进行了一个概要的总结：使用自动化测试 使用手工测试项目没有严格的时间压力具有良好定义的测试策略和测试计划你直到要测试什么你知道什么时候测试对于自动化测试你拥有一个能够被识别的测试框架和候选者能够确保多个测试运行的构建策略多平台环境需要被测试你拥有运行测试的硬件你拥有关注在自动化过程上的资源

被测试系统是可自动化测试的 没有适当的测试过程没有一个测试什么，什么时候测试的清晰的蓝图在一个项目中，你是一个新人，并且还不是完全的理解方案的功能性和或者设计你或者整个项目在时间的压力下在团队中没有资源或者具有自动化测试技能的人没有硬件如果你正在从事自动化测试，那么一定要记住要关注将自动化测试与手工测试结合起来使用。首先，对于自动化测试率的目标是 10/90（10% 的自动化测试和 90%的手工测试）。当这些目标都实现了，可以将自动化测试的使用率提高。记住创建自动化测试的测试用例要比创建手工测试的测试用例花费更多的时间。不要将你所有的测试时间都用在自动化的测试用例上。同时也要记住在测试期间对每一个被发现的错误都要花费一定的时间去处理。

5. 自动化测试的好处如果你正在你的组织中引入自动化测试，记住有很多不同的方面被包含了进了。今天在测试工作如何被进行上有很多不同的视图。为了能够成功的实施自动化测试你应该提出这些问题：

测试覆盖什么？ — 我们没有覆盖什么？

由于遗漏的测试我们没有发现的"bug"会带来什么样的成本？由于不好的测试，破坏已有功能性的成本是多少？如果"琐碎的"测试被每天的运行，对于你的项目意味着什么？如果我们能够每天向开发人员提供他们最近代码变更相关的反馈，对项目有怎样的影响？这些问题都能够被自动化测试满足。你必须从自动化测试成熟度的级别 1 或者级别 2 开始，并开始测量结果。根据我的经验快速的向开发人员反馈并每天运行测试对于向自动化测试成熟度的级别 4 或者 级别 5 是非常有好处的。

自动化测试有以下的贡献：

降低风险 — 你知道你测试了什么和没测试什么测试能在项目的早期开始并随着时间一直扩展快速的反馈 — 自动化测试用例能够随时的运行在多个平台上的测试能够同时进行

更好的估计 — 你能够对测试进度和被使用的时间有更好的了解优秀人员的集中 — 你能够得到一个专家的团队，并将他们的知识传播给其他的项目喜悦 -你和你的团队正获得着成功

测试工具介绍

下面我介绍市面上相对比较广泛的测试软件，Rational 中的 Robot 和 MI 公司的 WinRunner 的具体的用法。

Robot，俗称机器人。它有几个特点，第一，Robot 它的语法相对简单，是一种类似于 VB 的语法，所以上手快；第二，Robot 的脚本组织结构类似于 C 的结构，相对容易理解；第三，Robot 运行环境和可参数化，所以容易维护；第四，要求的机器配置相对较低；第五，Rantional 系统集成的很多产品很优秀，而且适合面很广；第六，相对价格比较低廉。

WinRunner 的功能相对 Robot 相对来说有一下几个特点。

第一，WinRunner 的语法结构是类似于 C 的语法，理解相对容易；

第二，WinRunner 的脚本组织形式是以目录的形式组织的，所以相对来说比较好管理；

第三，WinRunner 支持当前市面上的很多系统；

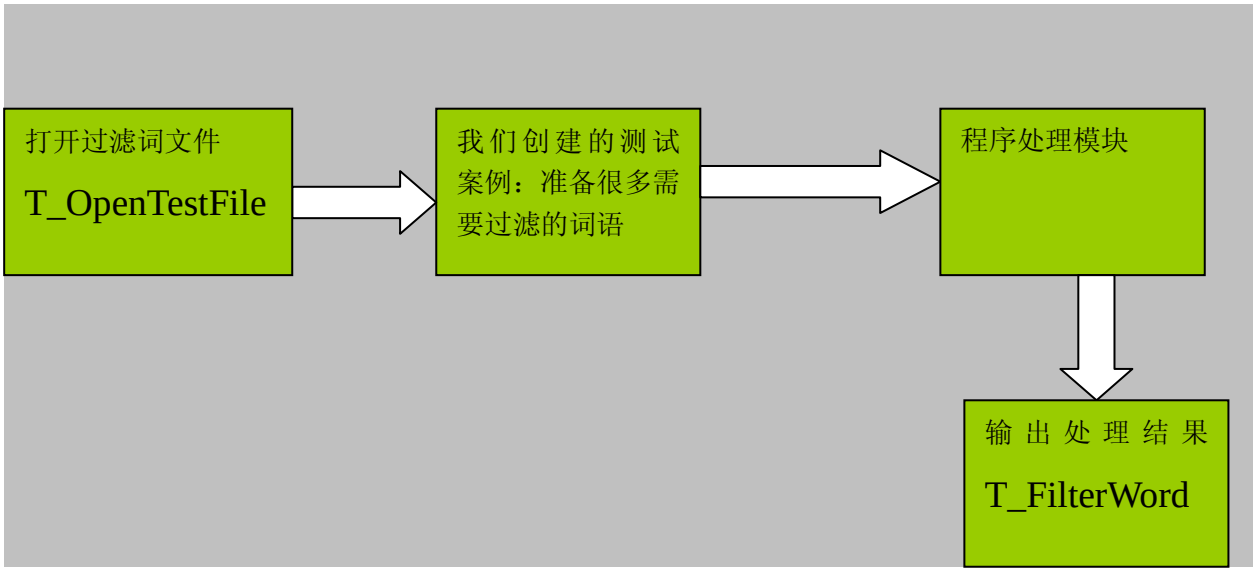
第四，可以随时 Update 检查点的内容，使得脚本的维护量减少

第五，机器配置相对要求高些；

第六，集成了很多优秀的组件。

下面我就尝试写一个 Robot 的脚本。

题目：请检查各种我们规定的字符是否过滤掉了，如果那些词语没有过滤掉请记录下来。我们先看测试这个功能的逻辑关系图应该如下：



测试脚本：

Function T_OpenTestFile(Filename as string)as integer

{

。。。。。。。。。。。。。

}

Funciton T_FilterWord(getword as string)as integer

{

。。。。。。。。。。。。

}

以上是两个 SBL 文件，相当于 C 中的函数实体

那么我们要建立一个 SBH，相当于 C 中的头文件，红色的部分类似于一个类库，我们把相关的或者具有相似属性的操作或者函数定义在同一个类库中。

Declare Function T_OpenTestFile Basiclib T_FileOpertion(filename as string) as integer

Declare Function T_FilterWord Basiclib T_FileOpertion(getword as string) as integer

我们现在建立一个脚本文件 REC

＃include T_FileOpertion.sbh

Main()

{

。。。。。。。

}

在此脚本中就可以调用上面那两个函数了。在以后的测试中，我们只需要维护过滤词文件库就可以了。这样提高了工作效率，保证了测试的正常进行。

下面我就举个例子，来尝试用 WinRunner 的使用方法，其实例子相对很简单，主要是想告诉大家这种语言脚本的组织方法是怎么样的？

测试工具在实际工作中的应用

现在在测试行业说的最多的有两个问题，第一个问题是测试的待遇和地位，第二个问题是自动化问题。我这里不谈测试的地位，只谈自动化测试。

如何做好自动化测试？这里给出几个分析方法，第一，被测试产品是否有延续性？第二，被测试产品的维护频繁度？第三，被测试产品自动化的难度如何？

其次，如何选择自动化工具，选择容易上手的，测试脚本和测试数据容易维护的，价格低廉的。

正确的理解测试工具需要我们去引导，我们应该清楚的知道那些应该自动化，那么能自动化，那些自动化程度会更高等。

工作中的自动化

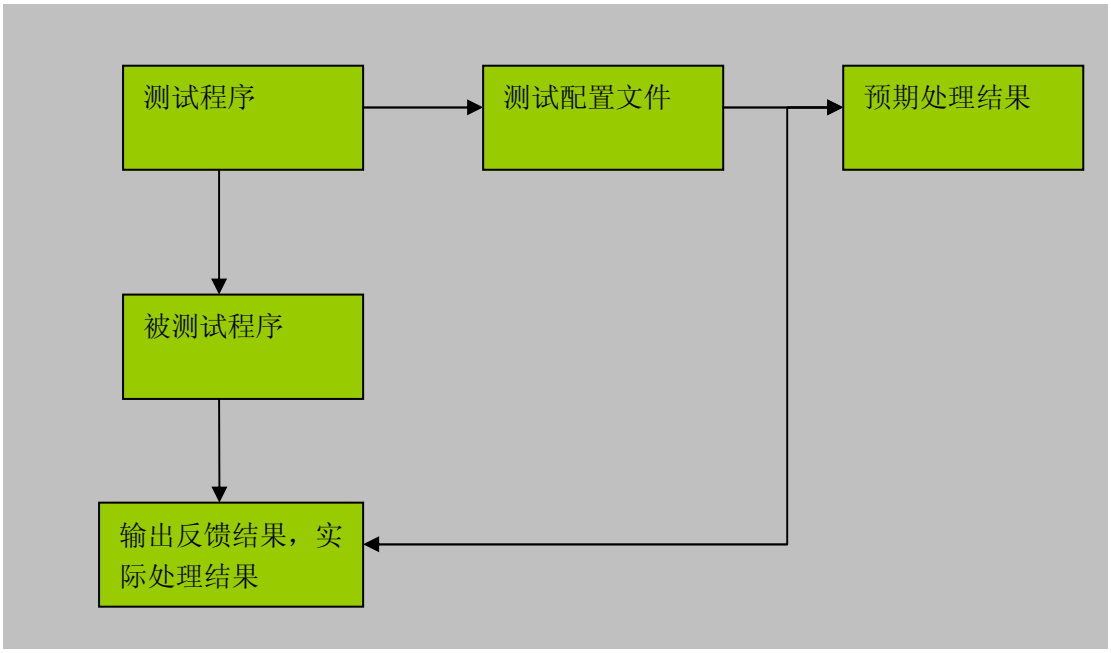
测试的几中方法

数据驱动法

数据驱动是测试中最常用的一种方法，它是在不修改测试程序的情况下或者稍微修改测试程序的情况下，不段的增加测试案例来进行自动测试的一种方法。

举一个具体的例子：例如我们检查登陆验证（用户名称），

如果我们采用自动化测试，我们只需要在测试配置文件中增加或者删除修改配置文件就可以达到测试的效果了。



只要将两个文件进行比较，就可以得到被测试程序的处理能力和结果了。

桩驱动

桩驱动也是一种常见的测试方法。这种测试方法是把某个模块假设是正确的，然后把这个模块作为主核心，然后测试他传出的参数和数据给其他模块来处理，查看处理的结果。

我那 EXCEL 中数据举一个现实的例子，大家都知道，当你输入等号的时候，Excel 认为你需要计算，那么处理流程就应该是接受键盘消息，然后处理。如果我们把输入做为一个桩，那么来查看计算结果的话，就可以知道那些信息处理了，那些信息没有处理。

桩驱动的核心其实很简单,就是首先对桩模块做一个假设和详细测试,认为桩模块是足够健壮的;第二,桩模块是驱动机,能够和很多模块发生调用关系,是核心部件.

关键字驱动

关键字驱动和桩驱动有些相似的地方，关键字驱动的核心是

网络方面的测试方法

数据库测试要点

网络游戏测试要点

下面我和大家探讨网络游戏的测试方法。网络游戏是一个信息相对集中的一个网络产品，它牵扯到客户端和服务端以及客户端之间的通讯。下面我们就结合具体的例子来分析

我举一个 Mmog 的例子（注：mmog 是大型网络游戏的缩写），大型网络游戏主要是即时战略性质的，所以通讯上采用 TCP 协议来发送数据包。TCP 协议发送数据包的优点是尽量能够减少数据包的丢失，他是对数据包的一个精包装，一般的发送数据包采用实时处理的形式，即采用堵塞式的处理模式，而不是等到一定空间的数据包满了以后才一起发送（非堵塞式的处理模式）。

上面我们只是简单的讲了 mmog 游戏数据包发送的形式。下面我们分析网络游戏服务端测试的重点。

网络游戏服务端的主要作用：下发玩家信息，仲裁胜负信息，仲裁装备信息，服务器之间中转信息，状态存盘，玩家信息存盘等。我们来一个个的分析，服务端要处理这么多的事情，在处理什么事情的时候压力最大呢？那肯定是下发玩家信息最大，为什么这么说呢？因

为在下发玩家信息的时候，服务端会把每个人的信息和地图上的所有的信息，而且这个信息是当地图上的玩家和元素的增多而增多，当地图上的人越多的情况下，服务器的处理信息的量就很大。那么分析这么多，我们主要测试什么？测试同步消息的传递。就是在很多玩家的情况下，信息能否顺利的通知给各个玩家。

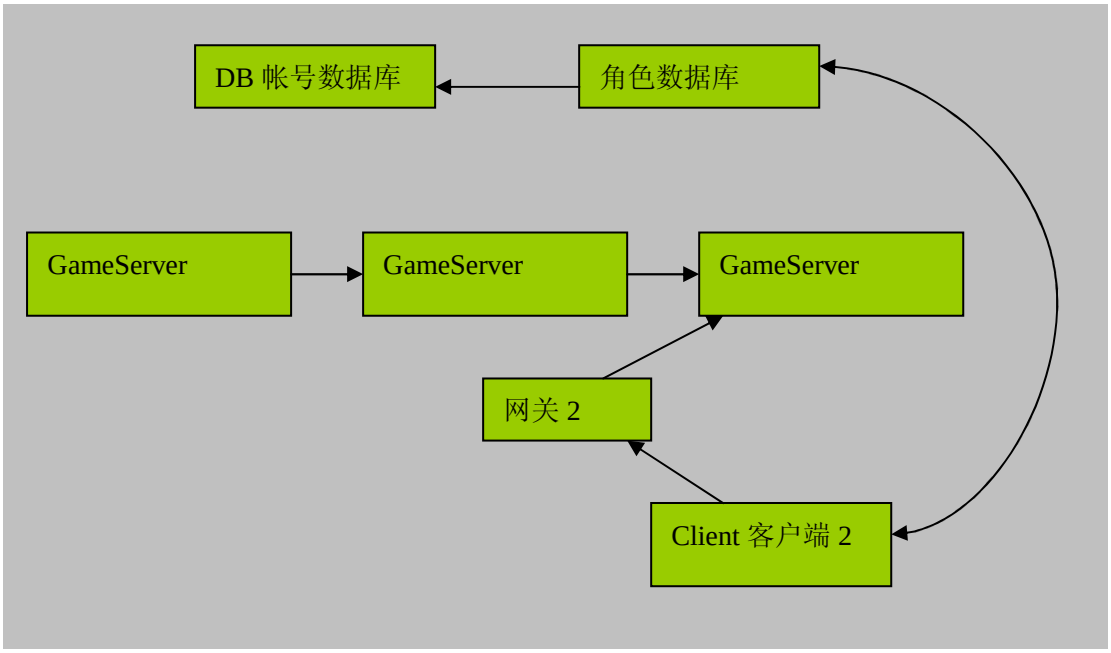
如果做这样的事情呢？根据经验（可以另外想办法），建议编写一个机器人，我们能够操作机器人，这个机器人需要处理什么事情呢？希望机器人能够在以玩家为中心的地方随意走动，能够跟随玩家跨地图，能够按照设定施放技能，能够聊天。通过机器人的每一个动作我们来观察服务的各个性能表现。关心那些性能表现？大家可以参考 LoadRunner 提供的业界比较官方的指标进行核对。

网络游戏服务端的测试相对比较复杂，对测试人员的要求比较高，测试人员必须了解用什么协议通讯的，怎么通讯的，服务器都处理了那些事情才能具体的分析测试重点和测试方法怎么做。

对于客户端的测试应该相对简单些，我这里说的客户端的测试不包含数据平衡部分。客户端主要是验证功能逻辑。例如我举个例子，任务系统，现在每个游戏都有任务系统。如果测试任务系统，我们应该怎么测试？首先完成任务的条件是什么？完成任务需要什么道具？玩家能得到什么道具？道具是否可以交易，拾捡？道具是否可以买卖？玩家跟 NPC 交付任务断线处理？NPC 在不同的状态下所说话的内容？是公共任务还是门派任务？在得到道具过程中储物箱满了会怎么处理？在负重超出了本身能力的情况下怎么处理？等等，把细

节的问题考虑清楚了，这样设计案例执行就是了，没有自动化的必要（但是可以在组件测试中进行自动化，系统测试没有必要自动化）。

客户端还有一个测试点就是地图，因为地图是游戏的门面，所以地图的检查点：绚丽度，整个地图的亮度，不同分辨率下的地图显示，地图的拼接，地图上 NPC 或者怪物的分布等等。网络游戏的测试是积极具有挑战性，包括能力，耐力，创造力的人奋斗。



还有一种是休息类游戏，例如盛大网络的泡泡，客户端之间主要是通过 UDP 进行通讯。关于 UDP 通讯的检查和测试我在这里先不做详细描述，以后进行补充。

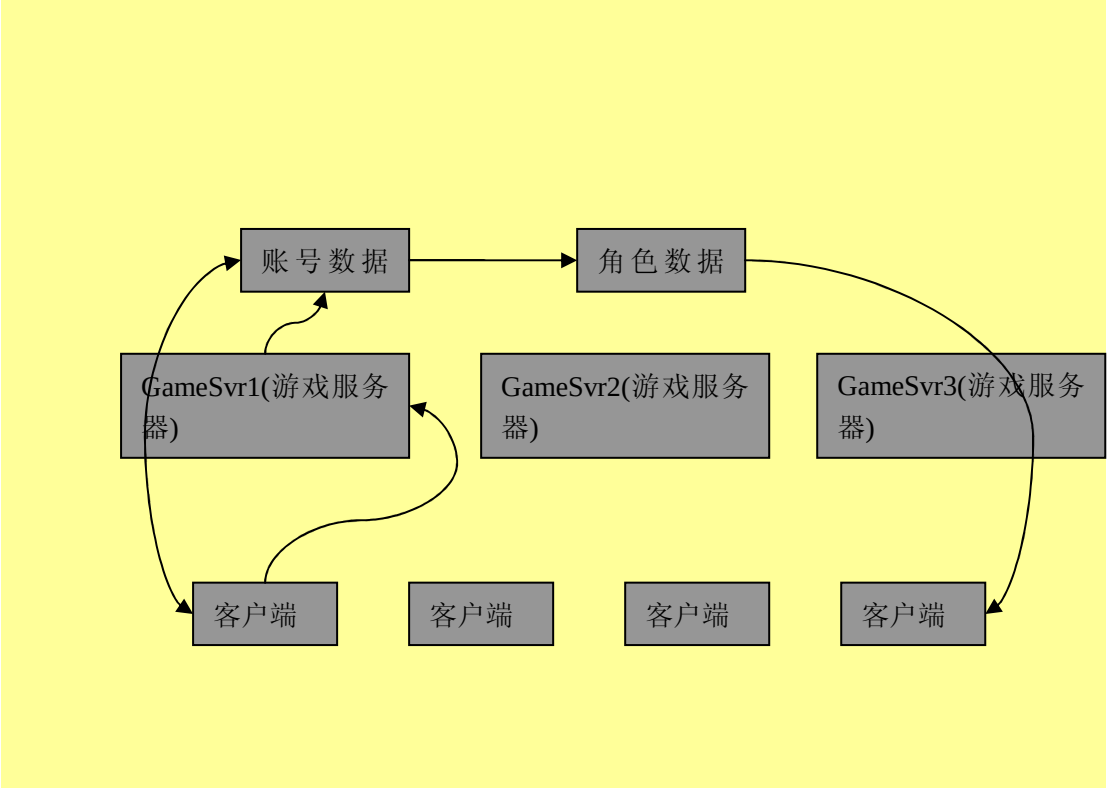
C/S 结构测试要点

网络游戏现在是 C/S 架构的一种，但是我们平常所说的 C/S 是一种类似于银行系统，购物系统等，这种系统要求：服务器处理能力强，

反馈及时，但是有一个特点，这种服务器压力是出现在业务繁忙的时候，所以在测试这种软件的时候，需要完全弄清楚业务逻辑，业务点在那里？信息同步在那些方面需要关注？服务端数据处理流程是什么？如果把以上问题搞清楚了，测试的重点也就出来了，这里我不再举具体的例子，因为 C/S 架构的软件是相通的。

下面我就举个例子,给大家看看,关于 C/S 架构的 mmog 类型的游戏测试,服务器的压力那里,测试的重点在那里?

请大家看下面的图形,这个图是目前市面上 90%以上 mmog 类型游戏采用的结构.



我把上述的图形描述一次:

当客户端要求登陆服务器的时候,通过账号数据库验证,然后取出该账号对应的角色,然后告诉客户端你是我们的用户,然后由网关(我在图上没有体现)把该玩家连接到响应的 GameSvr 上.

我们来分析这个图:客户端向服务器发送登陆请求,如果采用非阻塞式,使得各个客户端的请求形成非队列式,那么服务器压力不会很大,如果采用阻塞式,那么客户端感觉服务器处理速度就会很慢,玩家不答应,所以大多采用非阻塞式,处理速度快.而且玩家和账号数据库不是长期保持连接,只是保持一个心跳而已.这样就得出一个结论,用户在登陆的时候,对服务器不会造成多少压力.

那么什么时候会造成压力呢?

玩家走路,大家都知道,网络最主要的特点是同步信息,也就是说,当一个玩家走动以后,服务器首先要计算一次,当前玩家指的坐标是否正确,其次,服务器要下发消息,告诉周围的玩家,他现在在那里,怎么样的姿态存在.因为服务器会给一定范围内的玩家通知这些信息,造成服务器压力增多,因为服务器(GaemSvr)和玩家保持的是一个 TCP 的长连接,如果当前服务器或者一屏(或者说一定范围内的玩家相对比较多)的情况下,服务器要处理当前服务器上类似于上面描述的若干个这样的情况,造成服务器压力巨大.

除了走路,还有什么会造成服务器压力大呢?

聊天,为什么说聊天会造成服务器压力比较大呢?因为当 A 对 B 说话,服务器需要寻找,当前 B 在那里,然后通知 A,B 在,你可以说话给他,这个时候 A 才可能和 B 说话.如果人相对比较多的话,可能造成服务器压力大.

跨服务器行走,也会造成服务器压力巨大?

因为当你从一个服务器跨到另外一个服务器的时候,服务器需要

把第一个服务器上玩家的数据拷贝的另外一个服务器,然后初始化一个实体,然后再告诉 DB,把这个人的数据存盘一次,然后再尝试跨服务器,如果跨不成功,如果成功,则把该数据初始化给本玩家,如果不可以,则另外处理,这样造成服务器的压力大.

总上所述,不是所有的 C/S 结构的产品,我们以上来就模拟人多的情况怎么样,而是要分析,那块的压力真正的大,压力大的情况下会影响到游戏逻辑和数据在那里?需要冷静的分析这些情况,才可能有效的执行测试.

综上所述,我们知道网络的测试重点是相同通讯,数据同步和存储,那么在进行测试的时候,需要熟悉的了解各个部分相互通讯的模式是 TCP 还是 UDP 模式,那么各个模式有各个模式的特点.

TCP 是通讯协议是把数据包进行精包装,然后在网络链路中进行发送,它需要进行三次握手才可以确定数据包的发送与否,丢包的几率很小,但是自身有补偿机制,使得包数据传输的过程中可靠性大大加强但是缺点是速度相对很慢.

UDP 通讯是另外一种数据包传送模式,这种方式的特点是传送速度快,但是容易造成数据包丢弃.

所以除了了解各个部分的通讯协议之外还需要了解,当丢包了以后,对数据包进行补偿的两种方法,第一种方法是把从丢失的第一个包到当前包作为一个整体的包,进行包的重播,这样的方法会使得从客户端看到通常我们在游戏中所出现的”卡”现象,还有另外一种补偿机制,就是把前面的数据包丢弃,然后播放最后一个数据包信息,这样就会在

其他的客户端看到一些莫名其妙的现象.

WEB 测试要点

Web 测试的要点其实很多，但是 Web 测试的着力点其实不多，因为基于 Web 现在有两个重点，一种是基于平面的静态的，只是更新新闻，那么这个的测试点相对简单，主要是各个链接正确性，服务器浏览量等，还有另外一种就是基于 Web 的 P To P 的通讯，需要注册，牵扯到用户信息，那么这些测试重点在于 P To P 通讯的顺畅性，数据的可靠性。

嵌入式软件的测试方法

手机软件测试

手机现在在社会上应用很广，同时针对手机上的功能增加也非常快，例如，蓝牙，例如，短信，例如，信息下载，手机游戏等，这说明在未来发展中，手机将人们带在一个孤独的社会中去。

简单回想，我们的上辈人开始用手机，那是后简单的目的：就是通讯，而且还必须在既定范围内，价格高昂等，这些只有很富有的人才能染指的东西现在在社会中的每个角落都可能。放羊的。。。。

那么手机软件怎么测试呢？测试什么呢？怎么组织呢？

那么我做简单的分析：首先确定手机软件是嵌入式软件，那么我们抛开 IC 测试，抛开盲区，耐久性，耐碰撞性测试不提，就软件本

身测试需要考虑的内容。

我们说了这么多，我们整理出来手机软件测试的核心是数据的时效性，怎么理解呢？不能说我发了个短信，对方一天都无法收到？或者我打电话，对方没有任何反映，这就说明有问题。

在例如，通过手机的红外线，能把地址薄中的资料传送的电脑中，如果在传送的过程中出错，或者对操作系统有要求，或者在传输的过程造成数据丢失，导出来的数据是无法识别的格式等，都会造成该功能无效。

再如，手机信息的存储问题，因为手机本身有自己的系统，有自己处理信息的方式和格式，那就牵掣到的问题和外界信息交互的问题，外界需要了解其格式，阅读其格式，就需要和外界有比较信任的通讯。

其次，是测试本身的功能，例如，能够正常拨打电话，能够正常编辑信息，能够正常保存朋友电话，能够有历史记录，能够正常闹钟和各种设置，能够正常有提示信息等。

我再举个例子，手机中游戏软件的测试，我们先了解手机中游戏开发语言 Java，其次，由于手机中的内存大小优先，所以游戏的大小以及运行的各种指标都很小，所以用 KJava。还有一个特点，手机中的游戏是回合制，所以类似的同步要求不是很高，所以测试重点就有所改变。

那么像手机游戏的测试点在那里呢？了解这个方面的测试，需要了解几个问题，运营商对产品的要求：

运营商对开发商提出的要求，一般情况有这么几种。第一，应用程序的大小；第二，应用程序的命名规则；第三，应用程序版本控制规则；第四，明确提出所支持的机器的类型；第五，采用的编码格式 UTF_8 的编码还是其他的编码；第六，外部中断的处理逻辑；第七，比对压缩后的文件和先前文件的比较。

上面我提出了手机软件测试的七个重点，下面我逐步添加详细介绍各个部分的细节。

第一，应用程序的大小。由于应用程序在被下载时百宝箱会根据需要插入 5K 的程序代码，因此要求 SP 提交的应用程序其大小不能超过手机最大下载尺寸减 5K，下表列出了目前市场上部分 JAVA 手机对 JAR 文件下载的字节数限制情况（未列入机型请 SP 咨询手机厂家）

手机型号	最大可下载 JAR 文件大小	百宝箱限制 JAR 文件大小	备注
Nokia 60 系列	无限制	无限制	目前由于 WAP 网关不能下载超过 100K 的 JAR 文件，因此对于 WAP 方式下载应用的手机（T720 除外），百包箱限制 JAR 文件大小不能超过 95K。
Nokia 40 系列	64K	59K	
Motorola T720	无限制	无限制	
Motorola 388	无限制	无限制	
Motorola 388C	无限制	无限制	
Siemens 3118/2128/S57/M55	无限制	无限制	
NEC N800	60K	55K	

其次，由于应用程序在被下载时百宝箱会根据需要插入程序代码，这段代码在手机上执行时要占用 10K 左右的堆内存，因此要求 SP 提交的应用程序在运行时要预留至少 10K 的堆内存。

第二，应用程序的命名规则。应用程序的命名规则的字符长度不

能超过 12 个字符长度，而且命名有一定的规则，需要 SP 和提供商商定，但是必须依照行业标准。另外，在提交的时候，需要了解一些要求，由于各个 SP 开发商开发环境，编译环境等问题，所以运营商都提出严格的打包要求。

SP 递交应用程序打包要求：

1. JDK 使用 1.3.1 版本（国际版）；
2. 打包工具使用 SUN 公司提供的 J2ME Wireless Toolkit (midp1.0 版本，1.0.3 或 1.0.4)；

   由于 SUN 公司的 WTK 只支持标准的 midp1.0，不支持各手机扩展的 API，需要 SP 对所使用的 WTK 进行扩充才可以支持手机扩展的 API，方法是：将扩展 API 加到 D:\WTK104\lib\midpapi.zip（假设 WTK 安装在 D:\WTK104 目录下）中即可。
3. 不做扰码或使用 RetroGuard 进行扰码，不得使用别的扰码工具。

说明：1. SP 在开发时仍然可以使用原来的工具进行开发，但递交应用时请使用 SUN 公司的 WTK 进行打包；

2. 如果 SP 不按照要求进行打包，出现百宝箱不能识别 API 的情况时将按照测试未通过的情况处理。

第三，版本控制要求是，版本控制的格式 x.x.x，应该严格遵守这个规则。

MP3 软件测试

通用软件的测试方法

办公类产品测试

办公类产品的延续性相对较长，所以部分模块有利于自动化测试。我在这里介绍办公类产品的测试重点在那里？

办公类产品的特点是人们在办公中经常要用到，频率很高，所以简单，易用，方便操作容易理解成为办公类产品的首要要求。

其次办公类产品考虑的机器的硬件配置，例如显卡，打印机等外设

第三，办公类产品还需要测试兼容性，因为自身延续性长的特点。

第四，办公类产品需要保证数据安全

第五，办公类产品本身的功能逻辑

那么怎么才能做好办公类软件的测试呢？

首先办公软件的核心是办公，日常生活中要用到，那么按照各种排版格式自己去排版就好了，在这个过程中详细得出一些结论：第一可操作性；第二，各个模块的使用逻辑，例如：在文件中插入对象，对象能否体现出来，设置对象的绕排方式，能否设置成功，能否体现，设置对象是否打印；第三，存盘，数据安全考虑，看存盘后对象是否保存，各种字体设置是否保存，绕排方式是否保存等。第四，兼容性，因为用户使用的办公类软件很多，需要文件共享兼容。其次是本身高

低版本的兼容性，需要着重考虑。第五，文件大小和不同文件格式的存盘信息。第六，各种性能指标，包括启动速度，占用磁盘大小，在使用的过程中 CPU 的占有情况；第七，快捷键的设定和检查，不要和系统的快捷键冲突，并且支持系统的各种设置支持系统剪贴板功能；第八，支持网络传输，能够自由的通讯和共享，能够支持协同工作，能够充分体现修改数据的同步信息。

杀毒类产品测试

杀毒类软件是一种工具类软件，随着网络的普及，杀毒软件也越发显得重要，那么杀毒软件的测试重点是：

能够发现病毒

操作系统检查

能够正确的报告并且查杀病毒

占有的 CPU 占有情况

查杀病毒的内存使用情况

数据安全

软件冲突

病毒库升级

自动更新

那么在杀毒软件的时候，还有一个重要的东西，就是用户的数据安全，当杀毒软件能够清除含毒文件中的毒的时候，一定要包含用户文件的可靠性。

ERP 软件的测试方法

验证测试

测试管理工作

我没有读过管理方面，只是看了一些书而已，在几个团队中应用，效果还可以，所以拿出来和大家交流。

其实无论是开发管理和测试管理，其核心应该是“理”，如果理都不通，那么管只能是强制性质，那么在软件行业，创造性的企业中，可能不能维持的很久，所以需要先“理”清楚了，然后我们来管，到那个时候也就不用管了，因为理顺了，所以大家都知道该怎么样做，什么时候做了。

那么谈到“理”，我们需要理什么呢？理事情，把要做的事情理清楚；理目标，把要达到的目的说清楚；理思路，把做事情的思路和方法理清楚；理资源，把合理的资源调配到合适的位置上，让兴趣和能力结合。我觉得从大的方面就需要先将这些事情理清楚了，才可能使得一个团队具有非常的战斗力。

其次，还需要和善的沟通，做到无所不晓，因为在一个团队中，气氛非常重要，也许一个人的今天心情不好，造成跟他配合的几个人都无法工作顺心，也许是感应。所以需要和大家开诚布公，把能讲的问题，能讲的事情讲清楚，让大家觉得你可以依靠，可以把心思告诉你，你尽力为大家解决后患问题，让他们能够开心踏实的做事情。

另外，需要换位思考，充分让大家提出意见，因为如果一个团队要发展，是需要大家一起努力的，这是大家都熟知的道理，但是做起来很难。避免一言堂，让大家充分参与到设计中，在其中找到自我的感觉，找到这里没有他是不可以的，这样每一个人才能关心项目的每一个角落，而不是为了工作而工作了。

其二，需要在“理”的基础上帮助大家总结，大家在什么地方容易犯错误，犯什么类型的错误，犯错误的原因是由于我们的思想老化了，需要改进做事情的方式，还是由于工作能力或者经验的问题。那么就需要对各种错误进行统计，以找到问题的根本原因。就问题而讨论问题，问题的实质出在那里，然后帮大家改进，自己同时也会进步。

给大家讲我现实中的例子吧，我曾经带的一个测试组，大家做事情都很卖力，而且成绩也非常显著，但是在做的过程中总是会有问题发生，问题到底出在那里呢？后来找到了原因，就是由于工作环节中出现问题了，所以他总是在那里犯错误，因为他是把工作当成工作来做了，所以他就改这个地方，我后来找到他，也找到了问题的原因，一起把问题解决了。后来我就告诉他们，如果谁在同一个地方犯同样的错误，我们的结论是他不懂，教他，共同学习，直到他告诉我可以了；如果第二次犯错误，那么我们给的结果是，他忘记了，好再学，直到他告诉我，他可以了；第三次犯同样的错误，我们给的结论是故意犯错误的，那么就要全组请喝可乐一灌。

这个例子在现实生活中很多，主要是引导，因为如果把工作当成工作来作，可能不会用“心”，因为如果是用“心”工作他会找窍门，

会找方法，负责他会厌烦；如果把工作当成自己未来的事业来做，那么就会用心工作，这样就需要我们正确的引导。

开发方面

开发分析

因为我们是第一次做 MP3 的产品，所以从技术上还处于摸索阶段，第二我们对产品的质量意识还存在一点问题，第三，四方沟通或者说回馈不够.

问题分析

目前存在的问题

进度问题

四方配合问题，因为在任何一家公司做嵌入式软件的企业，存在硬件、软件开发，市场开发，测试准备四个方面的组织需要开展工作，所以那个组织的进度耽误都会严重影响产品的发布周期，在现在竞争日益剧烈的市场中，必须把握市场机会，所以需要对进度进行把握。

建议采用作战图的形式，对各个阶段点进行有利的把握，以给各个方面赢取更多的时间，明确职责范围，则权利的有利结合，激发各个方面的积极性。采用利益共同体的原则，将各个方面合理的结合，让开发关心测试工作，让测试关心开发工作，这样形成一个有利的整体，让市场及时反馈信息，使得项目顺利进展。

沟通问题

SPEC 设计完成后，各个方面在限定的时间内做出反馈，以及时跟进，以便对产品有个明确的定位，各个方面对方案认可后，切实执行，如果出现技术问题，及时通知有关人士，以便进度和方案进行合理的修改和微调。在产品开发阶段，需要及时的总结可能遇到的问题和对问题的解决方案，通知相关人士，使得项目组的人有一种归宿感觉。

开发需要给每个 Bug 做出合理的解释，或者对我们的测试人员进行相关培训，以便测试人员能够深切的理解产品，能够对产品进行合理严格的测试。

这里提出一个建议,在测试人员提出 Bug 以后,知道表现,一定要弄清楚问题发生在那里,是什么导致这个 Bug 发生,才能有效的分析结构,有效的进行案例的补偿和测试.

跟踪问题

跟踪问题牵扯到 Bug 的跟踪和进度跟踪，Bug 的跟踪需要一套流程，及时的出来 Bug,使得发现 Bug 和解决问题形成一个完整的流程。

进度跟踪的方法，建议采用阶段点产品，定义明确的时间点，明确各个时间点上完成的功能点，保证产品可以编译，可以抽查，可以运行以完成的相关功能。这样使得大家在各个时期能够看到大家都在进步，产品在进步，有利于整个团队的士气。

版本控制问题

定制严格的产品发布流程，降低由于时间问题，压缩测试时间，给产品发布带来的风险。建议采用的方法，由项目组定制编译计划，然后有相关人士进行讨论，各个方面达成共识之后，坚决执行。

版本控制和进度控制密切相关。

进行严格的代码管理，以保证公司机密资料，在代码 Check in 和 Check Out 的时候建议由专人进行审核后方能入库，以免误操作或者敌意的操作造成代码冲刷。控制版本有几个好处和优点。

关于 Bug 的回馈优点

第一、 开发方面回馈速度快

第二、 解决速度快

关于测试情况的总结

第一、 涵盖情况

第二、 Bug 的质量问题

开发体系问题

第一、 设计的分析做吗？技术难度？设计的合理行，进度的可行性？

第二、 单元测试做了吗？

第三、 CVS 的 Check In 控制了吗？

第四、 进度风险预估了吗？

第五、 进度控制如何进行的？

产品方面：

产品分析：整个产品在开发前期，市场已经找到了市场切入点，但是在现在产品要发布了，我们不能或者说很难用一系列的数据做说明，告诉高层这个产品可以发布了或者说不能发布，因为分析的数据太少了。因为数据是说明问题的基础，但是现在定了产品发布的标准，只是一个现存数据，没有预估分析可能性，对产品投入市场可能存在风险。

建议：

第一、 案例执行率

第二、 案例发现 bug 率

第三、 重复 bug 率

第四、 KLOC

第五、 产品日测试汇总图（察看历史曲线）

第一步、增强开发质量意识

做法：这个是个长远任务，但是我们可以做具体的事情，例如：检查开发单元测试情况或者单元测试代码等活动

对设计的模块案例检查回馈情况，把质量和开发邦定，不要让测试单独承担这个压力。

活动方式：宣讲，制度，执行，邦定

第二步、增强测试本身素质

做法：因为测试不成型，这是现状，所以我们通过提供一些数据来提高案例设计的能力或者说告诉测试人员我们的缺陷在那里。这个是长远的事情，因为需要人力去做这个事情。提供如下数据：重复报告率，案例和 Bug 的比率等。加强程序概念的培训或者框架的了解，尽力从程序的概念上理解产品，尽量避免例冗余。

活动方式：培训，数据，比较

第三步、对产品开发过程中版本编译的控制

做法：CVS 库权限控制，所有的 Check in 和 Check out 需要控制，检查代码后方可入库。另外需要制订完整的开发计划和编译计划，使得项目跟踪超向正规。

活动方式：专人控制，以面冲掉原来的代码

第四步、进度控制

做法：在产品开发周期中定制完整的编译计划，对阶段性产品进行测试或者抽查。以保后期测试时间的争取和质量的保证。

活动方式：认可，执行，调整

第五步、控制进度问题

做法：明确开发模式，列举详细的开发进度，详细提出开发的功能情况，树立作战详细进度表或者叫作战牌，以鼓舞士气，让每个参与者了解进度

活动方式：承诺，监督，跟踪

18 做法：统一平台，统一流程，统一做事方式，实现任务单跟踪

活动方式：准则，平台，优化。

问题: