网络管理自动化测试应用

序言：随着网络应用以及各种网络设备的普遍，网络管理将会越来越大，以及越来越复杂，但靠人工的测试，其成本将会越来越大，因此，网络管理方面的自动化测试应用将也是一个趋势。也许这方面的内容电信方面的测试人员会感兴趣点，但是个人觉得，其实对于软件测试人员来说，软件是基于应用的，其交互是通过协议的，但是协议的数据流又是如何在网络底层运作的，这对了解整个网络架构以及扩充自己的知识面，对其职业的发展未尝不是一件好事。

一、 一、网络管理简介

1、网络管理现在应用是分为两类。第一类是网络应用程序、网络上的用户信息和存取的管理，这些是与网络流量监控、网络软件应用有关的网络管理问题，这里不作讨论。

2、网络管理的第二类是由构成网络的硬件所组成。这一类包括工作站、服务器、网卡、路由器、交换机、光网络设备等。因为这些设备分布在各个地方，所以需要一个网络管理程序实时的进行监控和管理其运作，因为其相应的网络管理软件应运而生。

其管理系统中的对象通常包括为四类：

被管理节点（或设备）：即你想要监视的设备

代理（agent）：用来跟踪被管理设备状态的特殊软件或固件（firware）

网络管理工作站（中心）：与在不同的被管理节点中的代理通信，并且显示这些代理状态的中心设备

网络管理协议（SNMP）：被网络管理工作站和代理用来交换信息的协议，其是在应用层上的。

通俗点讲，网络管理就是每个设备中有一个代理模块，其代理模块存储了其相关设备的信息，然后通过网络管理协议与网络中心进行交互的一种管理方式，由网络中心进行统一的控制和显示。

二、网络管理模型简介

1、管理体系结构

- 管理工作站
  - 必须有管理应用程序，用于数据分析、故障修复
  - 给网络管理员提供接口
- 管理代理
  - 能够响应管理工作站的信息或者操作请求

– 向管理工作站提供重要但未经请求的信息(trap)

• 管理信息库

– 网络环境下资源的表示

– MIB (management information base)被管对象的集合

• 网络管理协议

– 管理工作站和管理代理之间的通信机制

GET-REQUEST,GET-NEXT-REQUEST 和 SET-REQUEST 是由管理进程发送请求。

GET-RESPONE 是被管对象响应管理进程发送上诉三个请求的回馈。

TRAP 是被管对象自主发送的状态信息。

2、 TMN 管理模型

1）TMN 定义的管理层面：

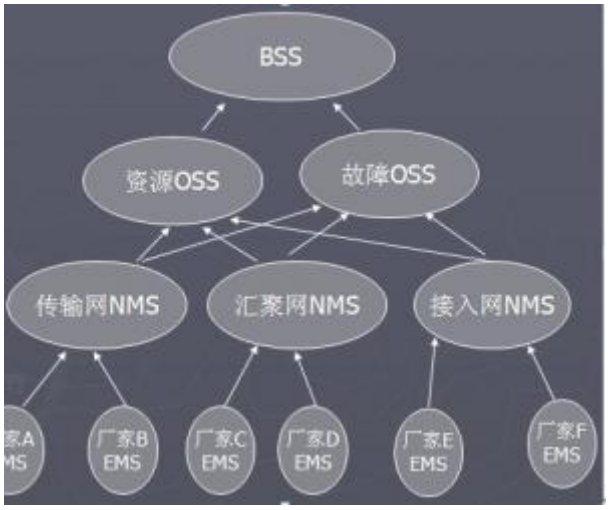
u 网元层：

u 网络层：

u 运维层：

u 业务层：

2）根据 TMN 模型，在网元层、网络层、运维层、业务层都有相应的管理系统。



u 网元层：EMS，俗称厂家网管或者专业网管，关注设备上的配置、资源以及故障信息。

u 网络层：NMS，构建在 EMS 之上，汇总全网的资源、故障信息形成一张完整的电路。不同的专业网有不同的 NMS。

u 运维层：OSS，构建在 NMS 之上，不同的运维部门有不同的运营支撑系统。

u 业务层：BSS，业务支撑系统，构建在 OSS 之上，为运营商提供运营管理以及运营决策。

3、网络管理应用

•故障管理(fault management)

–探测、隔离和修正 OSI 环境下的不正常操作

•计费管理(accounting management)

–记录网络资源的使用，控制和监测网络操作的费用和代价

•配置管理(configuration management)

–初始化并配置网络；控制、识别被管理对象使其实现特定的功能或者使网络达到最优；

•性能管理(performance management)

–对被管理对象的行为、系统资源的运行状况和通信活动的效率进行评价

•安全管理(security management)

–管理对象信息保护和访问控制

根据相应层次的不同，其网络管理程序的功能管理也有所不同。

4、对 MIB 的简述

MIB 为信息管理库，其可以用来表示一系列设备的相关信息，并且进行传输，其需要按照一定的规则进行规定其 MIB 格式，采用的方式是用树的节点形式进行表示的，为节点名和节点值的形式。

三、网络管理程序设计

在这里可以应用 java 的 SNMP4J 包进行对 SNMP 程序的设计。

你可以应用 SNMP4J 的 API 进行一个监测服务器、agent 代理以及实现相关的 pdu 的构造和各种 snmp 操作。这里不进行详说，直接下一个 SNMP4J 的 jar 包，里面有其 API 的 javadocs。

四、网络管理测试类型

1、 界面验证测试；即测试界面的字符正确性。

2、功能配置管理测试 ;即测试其网络管理程序能够对相应设备进行远程控制，其测试的是与 agent 的交互功能。

3、 MIB 功能测试；即测试是否能够通过 MIB 对网络管理程序进行 get 以及 set 的操作

4、 Trap 功能测试；即测试所有告警能够上报在网络管理界面上。

5、 兼容性测试；即测试网络管理软件能够应用在不同的操作系统上。

6、 国际化测试；即需要测试各种不同语言版本。

7、 性能测试；即测试网络管理软件的稳定性以及性能管理方面的功能等。

8、 Corba 测试；即北向接口方面的测试，需要测试其规则性与对通性等。北向接口相当于兼容各种公司不同网管的一个接口，用一个综合的上游软件平台可以管理各种不同公司的网络管理平台，例如：电信运营商需要华为、中兴等公司的网络管理平台都提供其北向接口，然后由第三方的软件公司开发一个综合网络平台来管理这些不同的网路管理平台，从而达到监控管理不同公司的设备的作用。

当然，还有一些测试方式和项目正在探索中。

五、网络管理自动化测试应用

1、 对于上述的界面验证测试，可以直接应用自动化测试平台，调用 GUI 自动化测试工具进行界面回显的判断，一般这方面的测试内容可继承在功能配置管理测试中。

2、 功能配置管理测试，一般为前后台交互型测试，即前台调用 GUI 自动化测试工具控制界面进行设备的配置命令的下发。然后后台应用脚本技术进行设备的命令行或者 API 控制，判断是否命令下发成功。

3、 MIB 功能测试，即应用 Java 的 SNMP4J 包编写一个模拟的 agent 客户端，对网络管理软件进行自动的整体的 get 与 set 的操作。

4、 Trap 功能测试；trap 功能，即 agent 上报通知的功能，一般是告警故障方面的主动上报，所以可以应用 Java 的 SNMP4J 包编写一个批量的 trap 告警发送的程序，然后应用 GUI 自动化测试工具监测其告警的上报的准确性。其 trap 的类型可以应用 MIB 的 OID 进行表示。

5、 兼容性测试与国际化测试，一般来说，即是在不同的环境下将功能进行遍历一遍，因此这一部份的测试根据其测试关注的侧重点的不同进行上述的自动化测试。

6、Corba 测试 ,可以应用脚本模拟上游端进行请求的发送和回复信息的接收。

六、网络管理自动化测试发展

随着发展 ,网络中的硬件设备的种类和数目会越来越多 ,其功能也会越来越多，因此对网络管理方面的要求也会越来越高，对网络管理软件的功能和性能要求

指标也越来越高，而电信设备公司或者专业的第三方网络平台设计公司将需要更多的人力成本去进行这方面的测试，因此这个时候自动化在网络管理方面的测试将会越来越重要，但是如何应用自动化测试将其做好，不仅是一个技术上的问题，更是一个需求分析和测试定位的问题了。

总之，自动化测试的设计的思想和原理是相通的，不同的是应用的方式和测试类型的定位，所以，做自动化测试，最终还是要归于到其本质上来，那就是如何更好的做好测试！

不断发展中的电信自动化测试

序言：做电信的自动化测试已经有一段时间了，算算，从接手到整体完成了流程的落定，虽说已经初步完成了一个阶段，但是我知道真正的挑战才刚开始，

一直一直在探索着到底什么才是自动化测试？

一直一直在思考到底自动化测试如何才能做成规模？

一直一直想把自己的经验分享大家，想大家能够少走弯路，能够真正将自动化测试落到实处,可是奈何自己水平有限，也是在不断学习，生怕去误导大家。

但是，请大家相信，自动化测试要做的话，是能够做得成功的，不关工具，不关技术，只关思想。

个人以为，思绪有多远，就能走多远，但最关键还是一定需谨思慎言。

一、 电信自动化测试简介

电信自动化测试，与单纯的软件测试项目不同，其测试涉及到产品的软件和硬件两个方面。系统测试在整个测试阶段占了很大的比重，因此自动化测试应用也主要应用在系统测试阶段。

电信产品的控制，主要包括网管控制与命令行控制，即大家熟悉的 UI 与 CLI，而 API 则是视公司而定，有的公司为了方面测试，而专门给出了 API，这方面测试，个人觉得在软件测试中要常见一些。

电信产品的产品架构为：硬件产品—软件嵌入式平台—具体功能软件实现（包括 MIB 与命令行）—网管软件（ ，包括 C/S 与 B/S 架构，有些电信设备商没有网管）。

而，系统测试主要是功能测试与性能方面的测试，即，主要控制 CLI 或者网管 UI 进行测试，电信自动化测试则主要是模拟 CLI 或者 UI 测试，而由于 UI 测试的界面变动性与维护性比较大，因此，主要还是采用脚本控制 CLI 的方式进行自动化测试。

另外，在进行性能测试时，需要涉及到分析仪器以及测试仪器，而很大部分这些仪器可以应用脚本进行控制的。例如，现在 IXIA 可以应用 Tcl 进行控制，而思博伦现在也可以应用各种脚本技术控制其仪器；安捷伦的一些分析仪器，也可以应用脚本通过 LAN 或者 GPIB 的形式控制，因此，大大方便了自动化测试的推广与应用。

二、 电信自动化测试要求技能

做一个电信自动化测试项目，其主要会用到两种技能，一是脚本编写技能；二是业务技能。

1）脚本主要有 ruby、tcl、python 等，

Tcl：由于其易用性与易学性以及在对字符判断和处理上的强大优势而在一段很长的时期被奉为电信自动化测试项目中的主要应用脚本。其在 80 年代开始就在 Motorola 使用，后来被思科采纳，并在自动化测试领域得到了广泛的应用。但缺点在于 :tcl 在设计之初是不支持面向对象 ,虽然后来出现了 iTCL 来弥补不足，但是常见的基于 tcl 的测试脚本开发还不是基于面向对象的。对脚本的可维护性和可复用性带来了一些挑战。

Phython：此脚本语言发展速度很快，因为其作为一种脚本语言而言，其相对支持库的代码水平较高，对于软件开发的各个方面的第三方库（如图像处理，网络通信，Web 技术等）都有非常好的支持。Python 本质上就是面向对象的脚本语言。在国外的很多公司开始应用其作为自动化测试的主要脚本语言。

Ruby：其是一种完全面向对象的脚本语言，Ruby 语言也受到了大型 IT 公司的测试团队的关注。在自动化测试领域得到了越来越广泛的应用。有一些大型公司正在由其他的脚本语言向 Ruby 转换，华为等一些大型电信设备商也开始应用 Ruby 作为脚本开发语言。

2）业务技能涉及则广，不同的产品线要求的技能不一。TCP/IP、SDH 协议、ETH 协议，以及近期火热的 OTN、PTN 等。其实个人认为，既然选择了做自动化测试，则不是去专研协议技术了，虽然要懂，但也是从整体的方面去分析。以上两者只是基础，真正要将自动化测试做成一种规模，那还需要别的很多方面的技能，做好电信自动化测试项目，至少还要从以下几个方面考虑：

1） 测试脚本的可重用性和可维护性

2） 测试脚本的可易用性与管理

3） 测试脚本的开发流程

……

所以说，根据这上面几点，则要求自动化测试人员拥有软件工程方面思想，而不是一些脚本编写技巧就能打发的；还需要对版本管理和配置管理熟悉；需要对测试流程与软件开发流程熟悉，最重要的要学会如何去将两者结合起来，协调好关系等。

所以说，一个好的自动化测试人员是很需要不断学习的。

三、 电信自动化测试发展

据我了解，自动化测试发展经历了一些阶段

1）手工辅助阶段：刚刚兴起自动化测试，系统测试人员借助脚本技术进行手工辅助的工具，并没有形成一定的规模。

2）应用阶段：一些大型的电信公司开始将自动化测试应用到测试阶段中，即统一应用脚本进行产品的系统测试。

3）发展阶段：自动化测试开始得到重视和发展，各种框架和技术开始滋生，C、VC 与脚本技术的结合与应用。

4）流程阶段；自动化测试开始有了自己的流程、并且与公司的开发与测试流程相结合，更好的促进产生的快速发布。

四、 电信自动化测试框架设计

关于电信自动化测试框架，其主要是从以下方面考虑

1）项目脚本的可重用性，由于产品线的丰富，不同的产品一些功能是可以重用的，而如何保证开发的脚本能够在不同的产品上进行重用。

2）项目脚本的可维护性，由于一些命令行参数的更改、功能的更改等原因，造成了脚本项目的变化性，因此需要增强项目脚本的可维护性。

3）项目脚本的可开发性，需要建立一系列的库，方面脚本项目的开发。

4）项目脚本的可易用性，自动化测试项目要得到推广，则需要更多的测试人员的参与，而如果在可易用性上不够的话，则会降低测试人员对自动化测试的积极性。

5）项目结果的易分析性，脚本运行完毕后

因此，基于以上考虑，可以搭建一个简单的电信自动化测试框架如下：

l 脚本方法库：则是脚本编写的一系列的方法，即将一些常用的方法封装起来，方便调用。

l 功能点测试执行层：与自动化测试用例的具体执行功能步骤相关。

l 测试设备设置层：此层对应测试中的测试设备的具体设置，其主要关注测试设备的业务配置。

l 测试执行设置层：此层对应测试执行时需要的设置，其主要关注执行方式的设置。

l 界面管理层：此层对应平台的界面控制层，其主要应用于对应用界面对全局参数进行设置、对测试项目进行操作以及查看测试结果等，可以用 VC、swing 或者脚本技术实现。

五、 电信自动化测试流程分析

说到流程，这才是做自动化测试最难以及最关键的部分。

为什么要制定一个流程，如果不将其自动化测试从流程上把握的话，那么其永远无法成为一种规模。

个人想法：其自动化流程是指明一个大的方向，将阶段和相关角色指明出来，而其中涉及到一些具体的细节，则可以以策略的形式表现出来。

个人想法：电信自动化测试可分为两个分支：

1）手工辅助过程；其主要是为了辅助测试人员更加快捷的进行测试执行，是伴随着手工测试用例的。

2）例行测试过程；其主要是有固定的环境和固定的平台，应用于回归测试，是为了保障产品质量，加速产品发布周期的，其有相应的自动化测试用例伴随，但其自动化测试用例也是来源于手工测试用例的。

个人想法：电信自动化例行测试流程可分为：

自动化需求分析设计—自动化测试用例设计—自动化测试脚本开发—自动化测试执行阶段。

涉及角色则可分为：

系统测试角色，主要负责自动化需求收集与用例撰写等。

测试开发角色，主要负责自动化测试脚本开发等。

测试架构角色，主要负责自动化测试项目管理与平台架构管理。

测试执行角色，主要负责例行测试执行与结果反馈。

在此，只简单的说明一下，具体的执行流程就不说了，不知道大家在电信自动化测试流程执行方面，有没有好的经验，可以拿出来分享一下。

六、 电信自动化测试总体分析

1、 大型的电信设备公司，从很早就开始投入自动化测试，例如思科、Juniper、H3C、华为、中兴、华赛、Topsec 等公司，他们的自动化测试已经形成了一定的规模，他们大都数的底层控制是用 C 语言，然后提供接口由脚本进行控制，而且采用了多线程的模式，在稳定上做的很好，而且在不同开发模式上的自动

测试的探索和应用也比较超前，例如，从开始 RUP 方式到后续的持续集成方面自动化测试的应用等，总之，在这些大型电信设备厂商，自动化测试的比重越来越大，应用也越来越广。

2、 中小型的厂商，由于其开发能力的问题和产品线需求较少的原因，主要还是应用脚本的开发，因为其开发周期快，投入周期可以较短，收效可以较为快速一点，能满足很大一部分需求。因此，其所应用的主要是基于脚本技术的框架设计，伴随着库思想、分层思想等，然后在流程上进行规范和统一。

总而言之，个人想法，不同的公司，对待自动化测试，需要采取的策略不同，

1) 大型的公司以长期投入为主，其根据相应开发模式和产品战略，将自动化测试打造成实验室平台级。

2) 中小型公司以步步跟进为主，要将自动化测试做成规模的话，则需要一点一滴的思考的积累。做自动化测试，太快，容易迷失脚本，太慢，又不能形成规模，因此需要以需求为导向来求发展。

电信自动化测试的不断发展，让我们对自动化测试的期待更进了一层，希望电信自动化测试走的更远

自动化测试项目实践回顾

最近很长一段时间都致力于公司部门的自动化测试工作，现在主要是自动化测试框架的搭建与自动化需求总负责。

主要是公司电信产品的系统测试

系统测试，包括其功能测试以及一些性能测试，主要从系统级别考虑，测试的是产品之间的相关性和功能业务稳定性。

一、刚开始，自动化测试主要定位在例行测试和验证回归测试，其主要目的是提高系统产品测试的覆盖率以及节省系统测试人员的重复性工作，解放系统测试人员的一部分工作量直接面对测试用例的维护和改进工作，然后反作用于自动化测试用例；这就是一个基本的自动化测试定位的流程。

刚开始，主要从寻找需求开始，先把一系列的需求安排出来，然后计划完成的百分比、可能遇到的问题，需要研发配合解决的 DFT 需求等

之后，设计例行测试平台框架，主要将框架搭建成了四层：最底层直接面向的是测试点，只关注其测试方法和步骤，不轻易进行修改；第三层面向的是设备的配置以及各个设备测试所需要的一些局部参数；第二层面向的是测试端口的

选择和小的测试用例和功能的选取；最高层是例行界面平台，传递的是一些全局参数以及大的自动化测试项目的选择。
再之后，便是项目的集成，需要达到的目标是：1、每一个自动化测试项目之间互不干扰，每一个项目运行完毕后都恢复到干净环境。2、测试结果报告的完整性。3、测试记录填写的完整性，每一次例行测试都有其详细的记录。
二、以上主要是例行测试平台项目，还有包括一些通用自动化项目，即不加到例行测试平台中的，则需要与测试用例进行同步管理。其通用自动化项目直接面对的是系统测试人员，为提高其工作效率服务。
三、网管自动化测试项目；包括网管界面的测试与基本功能下发测试。可以利用自动化测试工具进行实现，已经完成预言工作，关键在于其定位，现在只将其定位在基本配置下发和属性验证测试。具体开展还需步步为营，因为设计到 GUI 自动化测试项目，则很容易迷失，关键问题在于界面的变动性和脚本的维护性。
现在自动化测试达到的效果为：例行测试平台运行 OK，通用性的自动化项目开展的积极。网管自动化测试初步有进展，还需反复考虑。
问题：1、自动化测试一些资源文件的管理（需要定期更新，用版本管理软件进行管理）2、自动化测试需求的跟踪问题，需求引领自动化测试技术。3、自动化测试的推广问题，如何与研发部门的配合开展。
总结：1、自动化测试工作不复杂但也不简单，其需要自动化测试人员既懂业务也懂技术。2、对自动化测试看法过低以及对自动化测试要求太高，都是因为其盲目性，一个懂产品技术和自动化测试技术的工程师，是很快能定位其自动化测试需求和开展的方法。3、每个公司有每个公司自己的特点，调研和需求分析很重要。4、自动化测试框架不难，难的是细节。5、自动化平台很重要，没有一个平台，其自动化测试只能流于形式。

让自动化测试更“智能化”

序言：快元旦了，没想到世界末日居然也这么过去了，下一个世界末日，我是等不到了，呵呵，做完了部门整体自动化测试和性能测试方面的年度总结和下一年的具体计划后，今年还算部门测试自动化和非功能性测试整体上有所进步，很多点都有了，而且自动化已经算是正式上路了，但离大型的测试平台建设和改进过程却还得有一段距离，不过已经不是这么遥远了，有时间可以分享讨论一下，现在先偷得一小段时光，调侃一下如何让自动化测试更人性化吧，大家

只当看着乐呵，不一定说的对，因为大部分没有经过项目实践，只是想法，能起到拓展大家的视野的作用，我就觉得不错了。

一、所谓“智能化”

智能化一直是现在的一个趋势，记得大学曾做过一个课题，叫智能家居项目项目，其实简单理解就是将模拟电信号转换为数字信号，从而应用一定逻辑去控制我们的家居环境，例如：室温和照明，可以跟随随意调节或者根据天气变化而变化。

在我们的互联网应用上，更重视这种智能化，现在流行的数据挖掘和机器学习技术，就是智能化的一个技术基础，直接表现在一些推荐功能和程序的自主学习功能，通过记录你平时的一系列操作作为输入数据，然后基于一定模型学习到你的喜好，达到主动推荐需求的效果。

二、调侃如何让测试更”智能化”

简单思考了一下自动化测试的“智能化”表现，其实简单而言，就是如何让自动化测试更加贴近人的想法，一则更加像人一样思考，二则更贴近人的需求，提高人的测试效率。

1、人具有变化性，让自动化测试更具有判断性与场景处理

自动化测试一直不被用来提倡发现问题的最大原因在于他是将测试用例固化实现，只关注期望关注的，那为什么手工测试会比较容易发现问题，可以对以下场景进行想象：

1）**手工测试遇到问题时，往往为了验证此问题，会进行一系列相关的操作，而根据测试中的一项现象就是：基于 80/20 原则的缺陷集群性，即版本发布前进行测试所发现的大部分缺陷是由于少数软件模块引起的**，也许这样测试人员在验证过程往往容易发现更多的问题。而这种方式的话，首先，基于功能点进行关键字驱动封装，并且指定其关键字验证错误时，它的走向如何，即需要去调用哪些操作去验证功能，或者调用另外的关键字功能封装进行验证，这样一层一层套一层，直到某个关键字的验证完全通过为止或者开设另外一个线程规定和监测搜索的期限，否则容易造成搜索无止境。这种机制是为了对与之验证功能点相关的功能点进行探索，发现更多的发现问题。需要注意的是：这种机制是建立在关键字测试驱动框架上，而且调用的关键字注意的是在保证验证后恢复到调用前的初始环境，另外这种机制与程序异常机制还不一样的，异常机制是程序运行时候程序抛出的异常，当遇到异常时，测试脚本无法进行下去，

则会基于异常跳转到异常处理部分，一般会是清除测试环境，由框架转到另外一个测试用例。

2）**探索性测试**，在写这篇文章的时候，突然简单的浏览分析了一下探索性测试的相关书籍，突然发现，探索性测试就是一种强调以人为主观的测试，但是却又基于一定的规则，具**体实施可以围绕 SMART 规则，测试目标、指标评估、可实现性、当前语境的切换以及合理可接受的时间限度。**而探索性测试，按我的理解，更多的是一种思维模型，个人觉得，测试自动化和探索性测试并不是相对的，各有优缺，能相互结合，也可以利用程序建模完成，基于自动化测试手段开展一些探索性测试，当然，这只是浅见，需要再好好的学习和实践探索性测试后，并基于实现一定的模型，才能有发言权。

2、人具有学习性，让自动化测试更具有学习性

当今业务脚本的很大的一个弊病在于缺少动态性，而软件测试是具有杀虫剂效应，而人在测试过程中是具有学习性的，能够感知软件和产品的变化，快速做出相应的调整，例如：哪些功能模块是非常稳定的，可以少测试，哪些是重点，需要多测试等。

1）学习是一个有特定目的的知识获取和能力增长过程，一个学习系统中可分为:环境—学习环节—知识库—执行环节—学习环节。而学习策略则可以简单分为记忆学习(即我们学习过程常见的死记硬背)，归纳学习(即在学习过程，由大量的数据统计出规律，基于平均与概率求解)，解释学习(分析学习过程，通过对某一个问题求解分析进行学习)

2）在自动化测试过程中，当测试功能模块越来越多，没有太多的时间去全部覆盖，我们可以采用归纳学习的方式，基于自动化测试的执行结果或者手工测试执行的结果为数据输入，然后基于一定的模型(例如:以通过率和模块的重要率计算的平均值)得出测试推荐模块，或者当要执行一个功能模块时，基于历史数据和模型(bug 出现的错误相关性 ,功能相关性等)计算出与该功能模块相关性最大模块，并推荐测试。

3）还有一种机器学习广泛用于白盒测试中测试用例的自动生成，这个就不详细说了，有兴趣的可以查阅一下。

总结：当然，以上说的都是个人想法，除了基于关键字框架中的探索测试外，其余的很多都没去写代码实现验证，更不用说实际项目应用了，因为这种学习过程是建立在大量的输入数据上的，而现在的自动化测试水平还不足以支撑起如何庞大的工程，所以大部分只能为想法，而且自动化测试需要从简单做起，

一步一步一个脚印。**测试是一个很好的领域，不闭门造车，放飞思维，跨领域结合，不实践不批判，共勉之。—散步的 SUN**

测试工具设计开发之旅

言：一说到自动化测试工具，大家很多人都会想到的是 QTP、LR 或者 selenium 之类的工具，要大家一开始设计一个这样的工具，其实确实很有难度，因为其包含的功能细节太过庞大。当年的我，开始设计开发工具的过程中，走了很多弯路，例如：做工具的界面技术的历程，刚开始用 tcl/tk 脚本语言，用 tcl 写底层框架，用 tk 写图形界面，后来发现 tk 虽然构造图形方便，但可拓展性实在太差。就开始学用 java 的 swing 写界面，当时傻傻的从界面的布局，到界面的 MVC 框架，然后是各种图形的数据结构都是自己一点一点写出来的，但确实锻炼了能力，后来就开始掌握一些现成的图形框架，例如：java 里的 RCP、python 的 WxPython 和 PyQT。到现在更喜欢的是简单 web 框架，所以，现在将自己的工具开发之路简单分享一下，希望过来人不要走我的弯路，这篇文章虽说是说工具开发之路，但更多的是是一种学习思路，而且这一段旅程还很漫长，我也继续探索，也希望大家能得到一点启示，互相学习。

一、自动化测试工具浅析

在做自动化测试的这段日子里，现在也单独设计开发了一些公司对内和对外级别的工具，也设计开发过 C/S 和 WEB 方面的自动化测试平台，回首看来，从之前很简陋的工具，到现在客户应用级别的工具，真的颇觉时光飞逝。

1、界面自动化测试工具，我们往往入门的时候都是用的商业或者开源的工具，例如：QTP、RFT 之类，这些都是界面级别的自动化测试，界面自动化测试的有一定开发难度，但是确有不少的开源库可以提供，你完全可以基于以上库开发，或者有一些开源的工具很成熟了，你所做的就是基于以上进行一下更改。例如：测试 java 界面的工具就有 aboot、swbot、mathron 等开源工具，测试 web 界面的有 selenium、watin 等，测试移动端的有 robotium、monkey 等。要能二次开发这些工具，主要是需要理解抓取对象和回放的原理，然后是一些配置文件的处理，对象库里主要是 XML 的处理，一般录制功能我觉得可以忽略。

2、白盒测试工具，一些代码级别的测试工具，例如：对代码覆盖率的分析、对代码质量的分析等，这方面涉及较浅，就不随便造次了。

3、接口自动化测试工具，接口自动化测试工具在开发的时候，首先需要明确业务接口类型，然后掌握一定的接口工具的应用方式，一般的接口工具都是会解析某种接口定义文件，然后将接口文件以界面的形式展现出来，可以通过对界面接口的操作：对某个接口填写参数，然后发送到服务器端，查看响应，或者直接 get 接口返回值。例如：SoapUI 工具是针对 WebService 系统的测试，主要是解析 WSDL 接口定义文件。Jmeter 和 LR 也可以做接口测试工具，例如：java 接口和 HTTP 接口等。之前，开发过的接口工具包括：SNMP 接口和 corba 接口工具，其原理也是解析 mib 和 IOR 接口定义文件，然后可以对接口进行 set 与 get 操作。所以，开发这类的工具，一定要明确什么是软件接口、然后接口描述文件是什么，最后是如何去对接口进行操作，日志和结果的展现等，还有一些就是额外的功能了，例如：录制，将测试人员对接口的操作录制下来，成为工作流等。

4、性能自动化测试工具，看到性能测试工具，大家很容易想到 LR、Jmeter 之类，这方面的工具，我用的较少，但是会基于自己公司内部的产品一些特殊性能场景方面的测试，会专门开发一些这样的工具，例如：开发一个发送 SNMP 网络报文的工具，模拟告警最大接收和并发性能，开发一个网元模拟器，能够模拟大量不同 IP 的网元 ,可以在公司网元管理器上测试同时管理的最大网元等。所以，性能测试首先要与业务场景相结合，然后掌握一定的性能基础和指标，分析好相关的接口协议和需要模拟的业务，就可以快速开发相应的工具了。

5、系统应用级别的自动化测试工具，这种工具需要明确应用场景，即明确需求，例如：我之前开发一些部门内部工具集合，专门提供给测试人员进行脚本录制（ ）、公司级别的有采集和巡检工具。（对外支持），这部分工具带来的效益是很大的。所以说，千万不要将自动化测试局限在测试方面，其实提高测试与开发的人员的效率、以及对公司产品的质量保障方面的工具都是能给公司带来直接效益的。也许几行代码也是一个能提升效率的好的工具。

当然，还有很多方面的测试工具，因了解有限，就无法一一列举了，大家可以补充。

二、如何快速开发一个自动化测试工具

1、**定位自己，发现目标**：首先要看，你是否对软件开发感兴趣，其实我们测试人员往往把开发看得太深，所以很容易就因为觉得困难而不敢开始，如果你对软件确实有一些兴趣，何尝不试试，我们做的，不是要去开发一个多大的

系统，我们的目标是能够做提高我们工作效率的事情，让我们的工作变得更高效、更有乐趣，学习知识的同时，还能带来价值，何乐不为。

2、简单开始，立即上手：首先，在工作中积极发现需求，需求不需要太大，有时候一个点即可，找到需求点后，你可以向领导提出来自己的想法，然后进行可行性分析和立项，另外，很多人都担心因为不懂技术无法得到领导的认可，其实在首先最重要的是你的热情和决心，然后自己平时简单学一点入门知识就可以了，如果第一次尝试开发一个工具，千万不要一开始就把面铺太大，很多时候，我们总会被漫天的资料给淹没，天天在看书学习中渡过。我在公司带着测试人员做自动化测试项目时，往往告诉他们的是，不用害怕，直接上手，不会了再反过来查询资料或者咨询别人，千万不要一开始就拿着一本资料从头学到尾，这是我们大学应试带给我们的弊端，让我们往往忽视了实践中学习。当然，这是在有人带的情况下，如果没有人带你，那么你就找一个简单的语言，开始从最简单的实践起，大学我不是学计算机专业的，当年工作是从 tcl 脚本开始的，用 tcl 实现了简单的线性测试脚本、简单的测试框架开发等等，当时我采取的策略就是明确需求，然后拿着教程，一点一点开始攻克，困难肯定是有的，挨过去就好了。

3、适合自己的流程才是好流程：我现在一般开发工具、平台都是采用流程为，需求分析+设计流程+设计模块+接口定义+开始开发，在开发中调整具体架构和细节。记得当年刚学到软件建模的时候，很喜欢用 UML 来定义我的开发流程，结果后来发现，小型的系统和工具，采用这种方式其实更是浪费了时间，UML 的主要好处是团队沟通和交互，将系统抽象到大家都能理解的地步。所以，后来我明白了，不同的环境采用不同的流程，适合自己的开发流程才是好流程。

4、技术应用，步步深入：刚开始，不需要掌握多好的框架细节、多好的设计模式、多强的算法、多好的分布式和并行，也许刚开始，只需要线性编程，一步一个脚印即可，也可以尽可能的使用一些现成的框架，不需要太专研到框架的细节中去，例如：RCP 界面框架可以让不用面对更少的界面布局的情况，让你的工具界面和 eclipse 类似，或者 C++的 MFC 足以让你应对很多工具。当然，随着后面的工具难度的加强，你需要开始积累自己的知识，例如：专门有一个自己的库，包括收集和应用一些开源的库，例如：作业调度库、界面框架库、持久层映射库、日志和结果处理库等。还有一些自己的算法和功能库等。随着后来，建议可以多看看 web 系统方面的东西，毕竟这是个趋势。然后，多站在标准化和接口层次考虑问题。所以，我讲究的是顺其自然，踏踏实实，打

好底层基础，对待新技术和框架，少追风，多思考。这样就会逐渐形成自己独特见解。

5、抓准测试：一个工具开发过程中，你也需要进行版本管理和配置管理，你可以学会利用 git 和 svn 进行代码管理 ,学会利用 maven 和 ant 进行 build，学会在开发工具的同时也学会一些开发和测试自动化流程。而在测试过程中，因为开发的工具不是非常系统化，所以可以主要从功能点(按照需求列好功能点测试)、异常分析(例如：合法性测试、异常操作测试等)、兼容性(之前写的 C/S 工具，因操作系统不同会有一些问题，而 B/S 工具，会因为浏览器的问题，而出现一些展示方面的问题，所以需要明确应用和测试环境)

6、快速发布：千万不要将工具做到很完美才想着发布，没有什么是完美，我们所做的就是利用迭代的思想，一步一步去完善。所以，定义好阶段，快速发布，然后在发布中收集问题。毕竟是内部使用，所以能够及时反馈。

总结：不管测试也好，开发也好，大家都是为了提高效率，找到自己的价值感而努力，但是很多时候，我们往往有所想，而无所为，就是因为总是把很多事情想的太复杂或者太简单，当然，这是我们每个人必须经历的过程，而我觉得，**我们需要做的就是寻找自己的乐趣，如果我们迷惑了，就应该果断开始行动，只为相信我们做的事情是有价值的，积累的力量是很强悍的，在不知不觉中也许我们就挨过了很多困难**，获得了很多东西。另外，大家如果看到这篇文章，是否能够也能留言分享一下各自在测试过程中自己开发和应用过的内部工具的想法和心得呢，或者说说自己的学习想法和迷惑吧，大家互相讨论。

自动化测试工具—“魔爪”的束缚，“思想”的剥离

序言：有朋友留言希望能写对 RFT 写一些基础的实践操作的文章，想想，也是，可是提笔时，却发现写不下去，是因为其违背了我对工具的理解；刚用 RFT 时，曾很短的一些时间用过录制，后很快弃之，一直在应用它底层的 API 以及自行编写的组件来构建自动化测试，之后在想学学 QTP 的时候，看了看其关键字驱动与描述性语言等原理后，直接对其 QTP 的轻量级框架学习了一下，发现你若是对 RFT 的 API 和框架层次用了一段时间后，QTP 的框架还是挺好掌握的。

所以，发现，自动化测试工具，看的浅，其对你而言，是“魔爪”的束缚；看的深，则是“思想“的剥离。

一、 工具的使用想法

对于那些真的想在自动化测试上做出一番成就的人来说，不是简单的用用工具就行了的，其实很简单的想一想，要是自动化测试只是用用工具，利用工具去搭建一下框架之类的，那么现在自动化测试肯定就已经形成一股很大的规模了，毕竟，能够节约成本的事情谁都想做的。

做自动化不是一件简单的事情，在开始做的时候，想采用哪种工具的时候，就要想到尽量少的把自己的自动化测试开展工具都集成给工具，而是要自己构建一个基础的架构，把工具的自己需要的那一部分给当成插件的形式组合进来。所以，在掌握工具的基本使用后，就得去研究其底层的东西，因为越上层会越不灵活，拓展越难。想用好一个工具，那就把它的基本精髓给剥离出来，然后构建到你的平台之中，举个例子，就像 STAF 与 ROBOT Framework 一样，他们称为自动化测试框架

，提供的就是一个可以让你随意组建的平台，其余的各种服务（组件）都是可插可分的，是以需求为导向的。

二、 工具的学习想法

学习一个工具，当然首先是要学习其基本使用的，但是其基本使用只是一个过渡，为剥离其核心思想的一个铺垫。

从 RFT (Rational Function tester)来说吧

学习第一步：使用入手，掌握思想

1、 刚开始，你使用 RFT，知道其可以进行界面的录制，然后回放，将其都操作了几遍。

2、之后，你又学习了其数据池的使用、静态点与动态点的验证、scriptAssure 的使用等

等到，将这些都掌握之后，这个工具的使用层面你就 OK 了，但是学习第一步才开始：

录制此是对象的静态映射的思想；

回放是将抓取的消息重新实现的思想；

数据池在此是一种数据驱动的思想；

静态点与动态点验证是一种控件属性获取和比较的思想；

scriptAssure 是一种属性匹配阈值计算及分割线比较的一种思想；

当你掌握这些思想，你会发现，QTP 在其使用上的思想其实是通用的，无非在形式上的表达不一样而已，多了关键字驱动思想与描述性编程，不过这个也可以用 IBM 的开源框架 SAFS 来实现了。

所以，工具学习第一步，从使用去挖掘工具的设计思想，这样，才能做到举一反三。

学习第二步：研透 API，剥离核心

等到上述核心思想进行学习后，开始要接触 RFT 的底层 API 了，其 API 才是其 RFT 的最核心的东西，RFT，说明白点，其操作界面就是一个 IDE 环境，你可以完全的脱离这个环境，去用 API 来构建其测试过程。我将从其常用的分层架构上来明确一下其 API 的使用

1、对象层 AppObject：一般都采用其 RFT 提供的 find(),即，动态搜索的方法进行对象控件的定位；原因为：做 GUI 自动化，最大的难度在于其界面的变动性，因此，静态映射的方法局限性太高，而动态搜索的方法则可以将其查找关键属性定位在其变动性小的属性上，甚至，研发人员能够对每个控件提供唯一的标示 ID，这样可以有助于对象的识别以及测试步骤的定位。

2、方法层 AppLib:在这一层中，主要是自己构建方法，自动化测试的重点，需要明确：

1）、测试的执行，需要做到无人值守的情况下能跑完一个测试集。

2）、测试的结果报告，能够有统一的测试用例集执行报告，并且每个测试用例有详细的 LOG，帮助测试人员快速定位问题。

因此，要做到以上两点，需要利用 RFT 的一些 API，

1）你无法做到多线程去监测测试的执行，那么你可以应用其 RFT 提供给你的 RationalTestException 类，这是一个所有异常的父类，所以，在每个方法中构建一个 try-catch 机制，抓到此 RationalTestException，则会关闭测试用例，继续进行下一个测试用例执行即可，并将当前测试用例置为 fail。

2）而对于 log 信息，则可以利用其 RFT 的 LOG api 进行构建，但我建议的是，这一部分最好可以自己编写一个 log 的服务，这样拓展性强。

3、用例层 AppCase:则不需要多说了，你可以任意自己构建了，可以做成关键字驱动的形式，也可以做成方法库调用的形式，原理上不变的。

学习第三步：挖掘原理，回归基础

其实，不知道大家现在能够理解我的意思了吗？

学习完工具之后，你就要学会剥离和应用，而我建议，现在应用 RFT，只需应用其对象识别的搜索机制即可，对于 log 的查看、运行的控制等，最好能够自己以组件的形式构建到自己的一个轻量级框架中，轻量级框架集成的组件和服务多了，也就成了重量级框架了。

使用工具越底层、按需求使用，你就能越能脱离工具的“魔爪的束缚”，脱离其限制性，真正做到为你所用，为需求而用。
到了这一步，你要学的就是回归基础，编程基础、数据结构、数据库这些东西都是最底层的东西，只有将底层的东西弄通之后，你才能随心所欲的构建上层，灵活的玩转你的框架，否则，一切都是枉然。
总之，为什么越底层的东西越值钱，因为那才是精华的所在，越底层的东西越容易被大家忽略，其不知，那才是构建整个架构的核心思想，正所谓，大家都会的，你也会，大家都不会的，你会了，那你就值钱了，道理即如此。希望与同道共勉之。

第一部曲:RFT 基础篇—录制与回放的使用和实践分析

预言：自动化测试，工具只能是辅助，思想更为重要，集大成者能够随手拈来工具，将其整合成一个框架和流程． 我不会去泛泛的讨论什么是 Rational Function tester，因为这可以在很多资料中找的到，我只是通俗的谈一下我的理解而已，一步一步的深入。RFT 表面应用说白了就是一个录制与回放界面应用程序的工具，它抓取界面的对象，然后存入对象库中，当回放时，其依靠阈值的方法判断和识别对象。
方法：使用 RFT，很简单，就是点击录制，然后操作你要控制的界面应用程序；操作完毕后，即可以脚本（JAVA 或者,net）的形式保存对界面对象的操作。点击“回放”，将脚本重新运作一遍，即将刚录制的操作重复一遍。
项目实践：大多数测试人员的对工具的认识水平就停留于此，认为依靠录制与回放就能替代很多人工的操作，但是其以下缺点足以说明这种方法的必然失败性：
1）界面的变动将导致大量的维护性，例如：你录制了 100 个脚本，100 个脚本中都含有一个名叫“确定”的按钮，当开发人员因为客户需求，将“确定”改为“是”之后，那么你的 100 个脚本就全都得更改一遍。
2）连续录制的脚本，如果脚本某一处因为对象识别不出的原因，就会导致整个脚本的使用上出现问题。
3）灵活性太低，如果测试用例某一处进行了更改，进行了某一功能的添加或删除，那么脚本修改后就不一定能够重新使用。

因此简单的录制与回放的方法就导致很多测试人员宁愿重新录制，也不愿意再去修改和维护以前的脚本，可这样做话，自动化测试就只定位在简单的回归测试，那么就只能慢慢流于形式化而最终失败。

基于 RFT 实践四部曲—构建企业级的 GUI 自动化测试平台

前言

Rational Function tester

根据我的一些项目实践经验和个人心得，想分享一下利用 IBM 的 Rational Function tester 来进行一个企业级的 GUI 自动化测试平台的构建，当然这不是告诉你一定要用 RFT 去做你的自动化测试，个人认为如果有能力的话最好能自己去开发底层的控件识别，当然难度有一点大。所以这里主要去体会的是一种思想，一种如何去真的将自动化测试实现规模化、平台化与流程化，工具和技术是一个次要面，关键在于如何去与你的公司流程与产品联合起来去做好自动化测试。

软件产品一般会用到下面三种不同类别的接口：命令行接口（command line interfaces，缩写 CLIs)、应用程序接口（API）、图形用户接口（GUI）。有些产品会用到所有三类接口，有些产品只用到一类或者两类接口，这些是测试中所需要的接口。从本质上看，API 接口和命令行接口比 GUI 接口容易实现自动化，去找一找你的被测产品是否包括 API 接口或者命令行接口。有些时候，这两类接口隐藏在产品的内部，如果确实没有，需要鼓励开发人员在产品中提供命令行接口或者 API 接口，从而支持产品的可测试性。

而且为了让 API 与 CLI 接口测试更为容易，应该把接口与某种解释程序，例如 Tcl、Perl 或者 Python 绑定在一起。这使交互式测试成为可能，并且可以缩短自动化测试的开发周期，当然这里不是重点。

当你的项目是基于界面开发的，而且项目已经成型，并没有提供你 API 的接口可以进行调用，那么你就很难做到两层的开发，你要考虑的是如何去做基于 GUI 的自动化测试。

很多测试人员因为上级想要推广自动化测试或者本身对自动化测试的热情而去推广自动化测试，而很多公司没有很大的人力物力去开发自己的自动化测试工具，因此就需要借助于一些自动化测试的工具，例如 QTP、winrunner 以及 RFT 等。

实际上，很多测试人员对这些工具的掌握都是靠着自己的摸索来一点一滴的用到部门的自动化测试项目中去的，而在这个过程中，又有很大一部分测试人员仅仅浮在这个工具的表面，以为测试人员所做的只是去用这个工具罢了，孰不知，很多自动化测试项目失败的原因并不是这个工具的原因，而是自身对自动化测试平台和流程的掌握问题。

因此，这次，将从 RFT 入手，来一步一步讲述如何去搭建一个企业级的自动化测试平台，所谓平台不仅包括测试工具、更是包括了流程和框架的思想，通过流程化，则能更加的管理自动化测试项目，才能更加的监测自动化测试项目的实用性；通过框架化，则能更加的增加自动化测试的灵活性，降低其维护成本。

请牢牢记住：一切皆平台，而且要达到的要求最好如下：

1、自动化测试平台是为较稳定后的版本服务的，永远不要用自动化测试替代第一轮人工测试（即一个新功能发布的版本测试），因为人工测试，因为一个产品的第一个版本的某些固定功能的 BUG 的 85%是在手工发现的，而 15%是在之后的版本回归中发现的。可以在需求设计的时候，即设计出用例，然后在进行第一次测试的时候，转换成自动化测试用例和脚本。这样可以用于到以后新版本时，将这些功能继续测试一遍。

2、根据人工发现的 BUG 分布率和人工测试的普遍率去定义好自动化测试项目选取范围，这样的话可以提高自动化测试的收益价值。

3、要达到的效果是建立一种平台机制，将测试人员当成用户，能够让一个不会自动化测试的人员来反复对这个平台产品进行反复使用和维护，并且稳定性较高。

接下来的四部曲，将分别介绍：

第一部曲:RFT 基础篇—录制与回放的使用和实践分析；

将说明录制与回放这种方法的使用，然后根据实践说明这种方法的弊端，因此这种方法只能小范围使用，而且只能用在很单一的回归测试中。

第二部曲:RFT 提高篇—RFT 的高效技巧及应用分析；

将介绍 RFT 的一些高效技巧，包括脚本模块化、数据驱动、ScriptAssure、动态搜索方法等。不会片面的介绍这些原理，而是从实践例子出发介绍一下这些方法，只要将例子操作一遍即可，这些方法都是可以在后期框架和平台的搭建方面能够用上的。

第三部曲:RFT 进阶篇—基于 RFT 的自动化测试框架的搭建

将介绍如何利用 RFT 进行框架的搭建 ,当然其思想不一定局限于 RFT 这个工具，其实自己开发一个也是基于这种思想的，就是尽量将 UI 层和逻辑层进行分开，搭建好框架，基本上，基于 UI 自动化测试就可以开始应用了。
第四部曲:RFT 完全篇—基于 RFT 的自动化测试平台的搭建
搭建好自动化测试框架是不够的 ,为了追求 GUI 自动化测试的实用性与易用性，需要的是建立好一个平台，其实即使用 swing 或者 VC 编写一个界面，然后用界面进行测试流程的管理。
OK，接下来，将尽快具体介绍这四部曲，希望能凭着自己的一点领悟，对大家在自动化测试生涯上有所帮助，也希望能够对大家的自动化测试项目能够起到真正应用，希望能够有志之士交流。

应用 SDH 分析仪的自动化测试

这几天，一直做着应用 SDH 分析仪的自动化测试项目
SDH 分析仪，属于程控测试仪器的一种，而这种仪表都集成了一种 SCPI 的命令集，Standard Commands for Programmable Instruments 的缩写，即程控仪器（可编程仪器）标准命令集。SCPI 是一种建立在现有标准 IEEE488.1 和 IEEE 488．2 基础上，并遵循了 IEEE754 标准中浮点运算规则、ISO646 信息交换 7 位编码符号（相当于 ASCII 编程）等多种标准的标准化仪器编程语言。它采用一套树状分层结构的命令集，提出了一个具有普遍性的通用仪器模型，采用面向信号的测量；它的助记符产生规则简单、明确，且易于记忆。
其实通俗点说，SCPI 就是提供一个接口、一个规范；一个可以远程控制程控仪器的接口，一个程控仪器命令通用的规范；这可以类似于一种协议，有国际标准，也有公司自己的私有的标准；国际标准主要针对一些常用的通用 COMMANDS ,语法和原则进行了规定 ;具体的实现 ,每个公司的程控仪器都有所 不同；因此 ,对于一些程控仪器 ,可以应用脚本进行 SCPI 命令的控制进行自动化测试，这是一种思想，也需要框架与集成，具体实现由于时间就不多说了，谢谢

测试技术发展小思

序言:算算,有三个月没发布日志了，这段时间对测试技术的研究很少，重点的是放在了整个测试平台的推广上，从测试部门推广到开发部门，然后到整个产品线 ,一个很有意思的现象 ,就是开发比测试更容易接受自动化过程(未尝试单测)，一是因为开发每天都需要自测，自动化能够替代他的部分工作。二是开发本身

对代码更有感觉。三是项目进度越来越紧的原因，静下来，根据自己的所看，简单想想测试技术类型的发展吧

一、测试发展

1、 测试前置

测试环节一般分为单元测试(代码块或者单个模块的测试)、集成测试(代码模块接口间或者服务接口间测试)和系统测试(系统产品功能和性能测试)，测试越来越多的是深入到测试的前期，首先通过静态测试(代码走查、技术评审、代码审查的方法对软件产品进行测试，之后在各个层次引入更多的自动化手段和动态分析进行结合，从代码层以量化的指标进行度量测试的充分性

2、 线上测试(监控)

对于互联网产品，现在很流行的灰度发布系统(从产品用户群中按照一定策略选取部分用户，让他们先行体验新版本的应用，通过收集这部分用户对新版本应用的显式反馈（论坛、微博）或隐式反馈（应用自身统计数据），对新版本应用的功能、性能、稳定性等指标进行评判，进而决定继续放大新版本投放范围直至全量升级或回滚至老版本)，一是基于客户端注入监控脚本以及服务器端的监控，能够快速的定位用户行为和发现问题，二是获取将此用户的行为数据和用户流程，例如:tcpcopy 可以在 tcp 层导入用户流量，从而模拟背景数据测试。另外，对用户行为的分析和建模、或者对历史 bug 的数据分析也会导致测试越来越有重点和针对性，对风险的管理和控制会变得越来越重视。

3、 测试细分

测试技术的能力将越来越细分，测试探索能力、业务分析能力、测试设计能力、自动化测试能力、白盒测试分析能力、测试专项技术(大数据测试、安全测试)等

4、 软件交付流水线

持续集成的应用随着自动化、性能等各个测试技术的发展，将越来越被接受和推广，这里就会涉及到配置管理、环境部署、数据准备、自动化技术应用、程序发布和回滚等各个环节

二、我们如何面对

个人想法:

1、 首先从广出发，要深刻了解软件发布体系，熟悉各测试场景，从而可以根据公司的状况，知道公司的测试重点，对症下药

2、 之后，可以选择从一个点开始精通、可以是自动化、可以是业务分析、或者是白盒测试、也可以是配置管理，一个点一个点进行攻破，然后连成线
3、 最主要的，用心做，世上的事情只要用心，有一股创造其价值的信念就肯定没问题

自动化测试阶段和软件设计思考

序言：发现好久没写博文了，前段时间，发现很浮躁，别想办法的静下心来，踏踏实实的思考，踏踏实实的做事，一直也在写总结，但却很零散，现在理理思绪，这一段时间，自动化测试很多东西都已经上道了，测试人员也能够去独立完成很多自动化测试任务了，我能够将更多的精力放在软件工程的思考上，那就暂且以软件设计为题，说这一段时间的心得吧。也许认知有限，请指点

一、自动化测试的阶段认知

很多人都将使用自动化测试工具当做了自动化测试，这样理解也没用错，个人现在看来，自动化测试的几种阶段吧

1、使用者阶段，能够去使用工具，到能够利用工具完成自动化测试任务。这个过程中，也许你需要的是将工具的使用帮助看懂，能够结合你的部门自动化测试需求去使用好工具，对测试理论有所理解。

2、 半开发者阶段，能够基于工具进行拓展，例如：基于 QTP 和 RFT 等写一系列的框架，这种阶段，就要求你能够懂一些自动化测试思想了，且对工具的 API 和脚本语言有一些理解了。

3、开发者阶段，能够脱离一些不灵活的工具，大千世界，各种测试开源工具的包能够为我所用，例如：你可以用 selenuim 操作 web 的 api ， abbot 操作 java 界面的 api 或者写一个脚本驱动库调用 CLI 来作为一个对象操作底层，自己封装一层数据驱动和关键字驱动层，然后调用然后调用 robot 的结果 api，最终也可以加上 husdon 来做一个测试任务的触发，根据自己的测试需求，应用各种开源包定制自己的自动化测试架构，当然，你需要能够很深刻的看待测试和自动化测试，能够对各种测试开源工具的原理有所理解（这种理解也是建立对软件开发知识的理解程度，例如操作系统、虚拟机系统、web 服务等）

4、设计者阶段，我以前，现在更是认为：自动化测试的大部分效益不是一定来源于一个多大的平台，多少个用例，而是来源于平时的各种测试活动中，无所谓自动化测试，也所谓手工测试，能够找出测试中的不足，能够抽象出测试中的某种理论或者模型。

5、 设计开发者阶段，我觉得，所谓的设计开发者，就是知行合一，能够快速的将繁杂的测试需求用自动化脚本替代，能够将一些测试的理论用软件工程的方式验证，能够基于某个测试任务能够快速的开发出易用性的自动化测试工具，不仅提出疑问，而且能够去抽象，去快速实践和证明。
6、 商业型阶段，所谓商业型，即是能够真正让整个领域产生巨大价值的推动，这个阶段，我也迷惑，但我相信肯定会有的。
注：也许以上的阶段有的看似脱离了自动化，但是我觉得，自动化是为其测试理论服务的，无论自动化测试还是别的测试技术，都是为了推动测试商业化，能够让测试良好的运作起来。
二、软件设计的思考
再说一说对软件设计的思考吧
很多时候，我们把软件设计想的太复杂了，从而让我们畏惧止步不前，最近在思考，领域是相通的，那么软件设计如何与我们最简单的认知相通呢。
1、 软件设计是否好比我们写文章，我们一开始学会文字，不管是学汉语也好，还是学英语也好，我们刚开始都是学语法，就好比软件设计，我们刚开始也是选择编程语言（java、C++、C），不同的语言有不同的应用环境，然后学习编程语言的语法
2、写文章，我们学会了语法，认识了字，但是我们还写不出文章，我们要学习写句子，学编程也是，我们首先要学习写简单的线性代码，很多人认为一开始要学习高深的软件思想，其实不好，为什么 C 语言基础，因为 C 语言是教你怎么一步一步写句子，然后组成记流水账似的文档，虽然不好看，但实用和基础。
3、 之后，我们踏入了学习写文章了，这个过程，就像我们写一个系统，没有人能一开始就能写长篇小说，每个人都是从最简单的文章开始，我们写代码也是，必须一步一步来，有的人写文档需要打草稿，其实就相当于编程过程中，说的好听可以叫建模，其实就是定义一些接口，组成系统的架构。
4、 写文章有很多大纲模板，就相当于写代码有很多框架，你要写成什么样的文章，需要你对某一个情景什么样的感触，编程也是，你能写成什么样的系统，就需要你对业务和协议的理解程度了。
5、 所以，软件设计和写文章道理很是相通，领悟力和苦功夫都是必需的，需要我们钻研进去但又不能拘泥于其中。写文章要多写才能出真文采，则软件设计也是一样，要多实践，不要老是望而远之，找借口确实比实践来得容易的多，我们往往太看重结果而不敢上前，但是实际上闭着眼睛只要迈出一步，会发现

原来这也是一件很容易的事情，刚开始的话，可以临摹，可以仿照，之后脱离自己写，到最后自己去思考架构，思考文笔，思考“写作”的系统流程。

总结：其实个人觉得：很多人都说，厉害的测试人员不一定要写代码，其实我也同意这种说法，但是，我认为更厉害的测试人员他一定懂软件设计和工程，并且有了一定的理解力，**测试人员可以是一个文章的读者，也可以是研究者，挑剔读者能读出文章的好坏，但是却无法指点，而研究者不仅知好坏，还能进行保障，会成为写文章之人的良师益友。**

从项目、产品、运营型看发展

序言：自己身处通信界一段时间，偶然网上看到一些公司的正积极的转型，突然将公司的发展对照着个人的发展以及自己自动化测试工作的进展，深有感悟，也许：一个怎样的公司能持续发展，渐渐在市场上占有一席之地；一个怎样的个人才能持续进步，渐渐的在人群稍有突显；一个怎么的自动化测试策略才能持续提高，渐渐在软件中真正起到至关重要的角色。这都是需要思考的吧。现在就说说自己一点认知吧。

一、从项目、产品、运营看公司的发展

首先声明，这些都是我听到的、查阅的再加自己的一点感悟，因为视野局限性，不一定适用，但却是值得去感悟一下；

在通信业界，公司分为三类，项目型公司、产品型公司以及运营型公司；

1、 项目型公司，项目型公司指那些业务是以项目运作方式为主的公司，其针对不同的客户需求提供不同的项目，排除公司规模性的方面，据说这种公司盈利很辛苦，因为其针对每一个的需求就需要对应一个项目，这就是一个开发成本；然后需要对开发过的每一个项目进行维护，这就是一个维护成本。所以，每一个项目都会耗费大量的成本，成本越高，盈利越少。

这种项目型公司需要的是将其项目的技术或者框架进行抽象，使得其项目间能够共享、甚至可以将其满足一定需求的项目拓展成产品。

2、 产品型公司，即拥有一定产品型组织结构，针对一定市场有专门的产品线支撑。在通信界。很多设备提供商都属于这类，而产品型公司的追求向服务型公司转型，其服务型公司是以顾客的需求为中心、以产品为载体、为顾客提供端到端的完整服务，其利润总额，提供服务所创造的利润将占很大一部分；例如：看看大家总说的 IBM，其 Rational 的软件产品，你会发现其主要介绍产品的方

式是从能够提供的服务方向来介绍的，而其主要服务理念是为其打造一条完整的软件交付平台。

3、 运营型公司，在通信业界，其三大运行商就是属于运营型公司，其需要从网络角度知道网络运行状况，还需要从服务角度知道网络运行状况。此外，他们需要在提供多媒体服务和应用时有效利用网络资源。在通信业界，因为其特殊性，其运营型体制无可替代。但与通信界不同，互联网的特殊性又决定了一个个大型的运营型公司的兴起，他们以互联网资源为优势，打造了一系列的运营平台，提供给广大客户一个享受服务的通道。

二、从项目、产品、运营看个人的发展

也许很多人都有过的疑虑，我在这个公司所学的知识出了公司之后是否还能使用？到底如何建立我的核心竞争力？

1、 项目型个人，即只针对当前公司的状况而获得的技能，而等到了出了这个公司，还需要去学习新的技能。例如：做手工测试，只会对当前公司的产品进行黑盒测试，也许你可以针对这个产品的特性测试的很好，可以出了这个公司，这个技能就不一定适合与另外一个公司了。

2、 产品型个人，即拥有针对一定范围的行业或者公司而拥有的技能，例如：编程的技能、撰写测试用例的技能、性能测试的技能。为什么有一些知识面很广的人却没有竞争力，那是因为就如一个公司，他把市场铺的太大，他也无法去顾及其主要市场，所以一个人不断学习，然后针对其公司和行业需求，抽象出自己的核心竞争力即可。而什么样的核心竞争力，就需要自己对其需求的把握了。

3、 运营型个人，个人认为，此人是一个整合型的人才，也许很多人有疑问，为什么有的人不懂技术，却能位于很高的位置，那是因为他能利用好人力的资源，就像互联网中能利用好互联网资源一样，将其各人的优势进行整合，打造了一个运转的平台。

三、从项目、产品、运营看自动化测试的发展

从以后可以联系一下，自动化测试的发展

1、 项目型自动化测试；即脚本开发式，测试开发人员负责给不同测试的测试需求开发不同的脚本，这样就导致了测试开发人员总需要帮助测试人员维护脚本，而且由于开发效率不高，导致自动化测试效果缓慢。

2、 产品型自动化测试，即开发了一系列的自动化测试框架与自动化测试工具，测试人员可以应用这些框架和工具去自己进行自动化测试，测试开发人员只对这些框架与工具负责。

3、 运营型自动化测试，即一个整体的自动化测试平台，这需要与软件产品流程相结合，利用软件开发的资源，把握住软件开发的每一个过程，例如：现在流程的每日构建感觉就是这么一个概念吧。

总结：以上只是我的个人一些看法，同意或者不同意的都可以一起探讨，我觉得对与不对，都值得思考，思想碰撞才能提高自身的一些眼界吧。

软件设计与自动化测试学习历程感悟

序言：最近一段业余时间都在进行 web 编程设计，采用的是 JSP 技术，虽然 JSP 在网站设计上过于复杂，可是其能帮助学习 java 的思想，而且觉得在理解自动化测试方面颇有些帮助。自动化测试设计也是软件产品设计的一种，不过为了在此区分，一个为被测试软件的设计，一个为测试软件的设计。前者是面向特定用户使用的，后者是面向测试人员使用的，前者是为了帮助特定用户实现某个场景、提高生活效率。后者是为了帮助测试人员完成测试工作，提高测试效率。

回想自动化测试过程和软件设计学习过程，**后来看了一个人所谓的软件设计学习历程，颇有**感悟，当然，只是在这里说说自己的感受，也许说的有点乱，读者需要保持一颗自我和清醒的心。

软件设计学习过程：

某位人士 23 岁毕业，对 Java 的优雅设计情有独钟，其 Java 技术之旅开始了。

1、最开始三个月，开始接触 Java，比如接口、继承、封装等，买了本《Think in Java》天天啃，并且同时做项目实践。猛学了三个月后，对面向对象编程 OOP 熟悉了，原来脚本式思维和对象思维确实有差别。

2、三个月后，开始啃《Core Java》，《Effective Java》，对 Java 有了更深入的了解，回调的概念也有了，逐渐接触到更高的层次，面向对象设计 OOD，这时又看了一本书《Head First Design Patterns》，感觉设计模式特别有趣。再写代码，已经不是面向实现编程，而是面向设计编程。感觉写 Java 代码太简单了。逐渐了解了 WebWork 等 Web 框架的使用。

3、六个月过去了，Java 瘾越来越大，逐渐开始往更高层次攀登，这时，又看到两本书《企业应用架构模式》、《UML 和模式应用：面向对象分析与设计导

论》，已经开始从设计往面向对象分析 OOA、架构攀登了。Hibernate 已经比较熟悉了，了解 Hibernate 背后的持久化技术、Spring 背后的 IoC 容器、组装技术原理。

4、一年后，他逐渐脱离了 Java 语言，开始看这类书《面向模式的软件体系结构卷 1》。这个阶段持续了一年，并且对以前的学过的设计模式，如命令模式、观察家模式有一个更深入的了解。因为两年的企业应用开发，他已经熟悉了 Java EE 的十来种规范，对 Web 容器和 Servlet 规范的关系有很深的理解，对 JDBC 规范和数据库驱动程序的关系也很了解。他正在经历 Java 开发的快速上升期

5、两年后，他突然发现，他学的很多东西都没用，都是纸上谈兵，比如，在自己的企业应用开发中，Command 模式、Template 从来没有用过。他还发现，本来 100 行写的一个功能，花了 1000 行，就是为了所谓的设计优雅性：可扩展。而实际上，还没有等到扩展，该系统就已经废掉了。他发现原来设计模式主要用在系统框架开发，而不是应用开发，一般开发人员不用，只需要理解。他还发现，他认真学过的 JMS、JCA、JTA、EJB 像是从来没有用过。突然他想通了，JMS、JTA 可能是一种无奈的选择：处理遗留系统。当他开始对自己两年学到的知识进行反省、批驳时，他已经有了技术辨别能力，知道技术推广也不是那么纯洁，也有商业炒作。这时候，他已经不限于 Java 了，开始了解 C#，Ruby，发现 Java 可能并不太适合互联网开发，PHP 可能更适合，ROR 开发更快但需要在牛人的手里。两年后的这个时候，他才开始真正驾驭 Java，他已经不再限于 Java，而是企业应用。这个时候，技术提升的速度越来越慢了，因为不知道还可以学习什么新技术。因为他发现，原来这些东西，最深层次的，都是几十年前的技术概念：消息系统、异步通讯、事件机制等等....

6、三年过去后，他已经不再限于企业应用，而是解决方案，技术只是一种解决问题的方式，比如企业信息化成功的关键，恐怕不是技术，而是企业本身的业务流程成熟度；企业信息化成功的关键，不是处理好了技术，而是处理好了几位企业高官的利益。这时候，对 IT 行业新闻，逐渐有判断力和免疫力。他突然发现，技术的力量很有限，商业才是最大的驱动力量。而此时，他已经不再钻研技术细节，比如 JVM 的垃圾回收机制，如果他在一个技术研发型公司，比如普元，可能还会深入挖掘技术。如果他在东软这类行业应用开发企业，这类企业的口号是 Beyond Technology，这时候他再执迷于技术而轻业务，恐怕

不太受欢迎。这个时候，技术的提升，就会进入一个平台期。

自动化测试学习过程：

自动化测试的整个学习过程中，不断在探索，虽然时间不算很长，但是确实是在一直在思考、一直在观察、一直在领悟：

1、刚开始的时候，是从手工测试入手，偶然之间开始了自动化测试之旅，发现自动化测试很神奇，在进行自动化测试用例撰写过程中（命令行的填充），对脚本技术（tcl）猛学了几个月。

2、若干月后，随着自动化测试用例加多，偶然开始有了结果的输出、日志的记录以及脚本库。在不清楚所谓的框架时，形成了一个简单的“框架”。

3、之后，需要对界面进行测试，开始研究自动化测试工具，之后在领悟了其神奇之后，开始疯狂的学习商业自动化测试工具（RFT、QTP 等），因为主要是研究 RFT，被其 ITCL 的框架深深打动、后来在实践过程中，脱离了录制，开始迷恋基于工具的各种框架。RFT 的 ITCL、QTP 的轻量级框架以及各种工具的自动化测试框架，后来也学会了自己去拓展这些框架。

4、之后，因为对 RFT 的学习，开始喜欢上了 java 设计，每天都享受应用 java 设计，开始沉迷于技术，想着如何去用技术完善自动化测试，开始不注重那些已经搭建好的脚本环境。

5、到了现在，突然回过头发现，自动化测试最害怕的事情就是一群疯狂技术者的游戏，其实最基础的还是踏踏实实的把需求做好，以前所设想的搭建的业务分层现在不是主要问题，以前设想的如何去跟踪命令行的变更问题，到现在也不是主要问题，其主要问题是一个简单的技术是否实现了其需求正常的落地了，发现现在真正用起来的东西才是最好的。而更多的技术只是在需求不能得到满足的情况下去拓展的。

6、当然，工具、编程、框架都是必须的一个过程，关键是不要纠结于一些技术细节而不去向前，要看到主要和本质的东西，其工具、编程、框架、流程最终都需要转换成思想，不管何种方式都信手拈来，成为满足需求中的一部分。所以接下来，我觉得，自动化测试的学习道路有两个阶段，第一个阶段，以技术学习为基础，不断进行技术方面的探索。然后，每隔一个阶段，跳出技术的视野，去挖掘一定的自动化测试需求，其痴迷的细节不是技术方面，而是自动化测试需求方面，从另一个角度说，也是测试的方式和需求。

7、而在学习 java 的过程，也接触过 J2SE、J2ME、J2EE，用了 swing 界面，也进行 web 设计，进行最基础的设计、也应用了一些框架，在没有多少实践的

时候，就开始专研设计模式且一直以数据结构为伴随，后来在进行整个系统设计中，也学了一点系统建模以及数据库建模，但总的来说，还是处于模糊状态，也曾迷恋过，也曾迷茫过，一直处于一种不断怀疑、不断痛苦、不断惊喜的过程。

其实个人想说的是，上两种方式，并不是去评断其好坏，不管什么方式，都有其好坏性，大家看看热闹就行，但是每种方式、每种领悟都是一段过程的结果，最重要的是我们坚定一个学习的信念不断学习下去，学习但不要迷恋于技能、要总处于一种简单的自我怀疑状态，一切以价值实现为导向即可。

突然想起，以前看的一段话：人早期看山是山、看水是水；中期看山不是山、看水不是水、晚期看山还是山、看水还是水，也许就是说的我们这一段学习过程吧，刚开始因为单纯，所以我们能够简单应用那些知识实现我们的需要就行，后来学习的深究，各种技术交杂在一起，人难免会有点晕眩，不能把控好自己的方向，后来，才发现不论什么方式，其实最终目的还是为了需求，不管简单或者复杂，能够把控好自己，把控好整个流程就好。

所以，自己还在技术的迷乱期，需要的还是不断的学习，这就是一个过程，所以相信，最终还是会有一个接一个的领悟，但关键是坚持啊，坚持啊，不管迷茫、不管怀疑。

工作，思考中的感受

序　言：今天先不谈自动化测试了，说说这段时间工作思考后的感受，这段时间算是我对自身工作的一个小回顾，这段时间，我一直在思考其工作的意义，也许我的这些思考能否给大家也有一点启示，毕竟它让我在工作中变得平和而主动。写的有冲突的地方，希望大家海涵。

一、工作环境

曾有同事向我来抱怨，觉得在部门小组里的待遇不公平，因为很多时候，他在辛辛苦苦的工作，而有些同事却是“表面文章”，跟上级关系相处很好，平时也表现的加加班，但是也不正经工作，而他平时努力把工作做完，加班加的少，这样，他得到的待遇比那些不好好工作的人差了，所以，他很郁闷，很想明年跳槽，换个环境也许会好。

而我问一句：换个角度想，如果跳槽之后还是这种状况，那又怎么办，难道你要陷入无限的跳槽中？所以，对于现在来说，除非你在这里已经感觉学不到东西了，感觉自身得不到提高了，可以考虑换个环境，而这些“不公平的待遇”

不是导致离开的主要原因。什么东西都有值得挖掘的地方，更何况这也是对自身的一种磨练吧。

个人想法：不知道上面的话会不会引起大家愤慨，其实也许我们在工作中或多或少有很多不如意的地方，但是很多人选择一不如意就离开，这样，让自己陷入了无限的不如意漩涡中，当然不排除找到如意的那种。但是我想说的是，也许我们的眼光可以不局限于一个公司，一个部门，而是放在个人的发展上面，**我们不是简简单单为争一时之利而工作，而是为了不断进步，不断发展而工作。**

二、工作方向

前段时间，自己陷入了一个工作方向的迷茫，因为做自动化测试，所以纠结其是致力于测试系统方向还是技术专研方向，前者是以自动化测试需求和流程应用方面为主，后者是以自动化测试开发为主，而我本身一直做前者，需要的是更多信息的综合和调度，但是发现的是对技术的深入，对后者技术本身的兴趣越来越大，当然不局限在自动化测试相关技术上面。

前一段时间苦苦追寻答案，害怕方向的错误造成时间的浪费，后来忽的明白，其实走哪条路都是一样的，关键看你最终想要做的。

个人想法：其实每个人在自己的人生道路上时，都会遇到一系列的分岔路，而走哪条路，只要你的终点明确，那都是一样的，我也许发现，内心有个声音，总在不时的提点你往哪，不管你现在是否迷茫，不管你现在处于什么阶段，但是只要你有一个对未来的想法，对未来的一个期盼，那么你终会朝着哪个方向走去，所以，对自己的想法是最重要的，说句也许大家不喜欢听的，我一直觉得，不要因为工作而工作，因为这样很容易让自己迷失自己的想法，变得麻木吧。**尽管现在什么也没有，尽管现在很迷茫，但是若是你知道自己以后要做什么的想法，那么现在你就可以开始一步一步试着接近了吧。**

所以，我选择了技术，因为技术是以后要做什么的基础。

三、工作沟通

前段时间，自身在沟通方面做的不妥，具体原因如述：刚调度来的某上级根据他的经验对自动化测试提出了一些设计方案并开始进行实施，而我在对这些设计方案评估的时候，根据现在的自动化测试情况，总是提出了一堆的问题反馈，例如：方案 1 是不是在现在没有必要，因为现在的主要缺陷不在这，方案 2 的设计代码(java)是不是太差了，没有考虑到复用性等各方面问题，对于以后的代码维护不够好等等。后来一小段时间后，在不断问题的反馈下，终于发现上级

有点小“恼怒“了...再后来，我反思自己的做法，确实在沟通上有些问题，不知道大家能够看到有什么问题吗？

其实问题在于我的沟通方式，我往往在提出一堆问题后，并没有伴随着良好的解决方案，其实再完美的方案也有问题，抱怨和提出问题的人多的是，而你觉得将一堆问题丢给上级，特别是需要现在出成果的自动化测试。所以，我向上级道了歉，认识了自己的错误，并保证以后提出的问题中，必须结合各方面的深入，伴随其解决方案。否则，问题，你还是自己留着吧~，**所谓沟通必须有效的沟通。 个人想法：**看看周围，也许，发现和抱怨工作问题，发现流程问题，发现管理问题的等等大有人在，但是反过来想想，当你处于那个位置的时候，你能做的更好吗？所以，我们可以选择少抱怨，发现问题的时候，多思考这是不是能有更好的解决方法呢，真正的提高效率才是其本质吧。

总结:以后只是我这么一段工作的一点想法，**觉得工作中心态和思考方式的提升，往往比技能提升重要一些，最重要的对待事情都从容性才能让我们去发现对待事情的真正需要把。**

当然肯定有不妥的地方，但是也肯定有妥的地方，事情都有两面性嘛，希望大家一起探讨工作中总结出来的感受，好与不好都可。

为什么自动化测试还未成规模

序言：纵观中国大陆，能做到自动化测试规模化的公司有几何，奈何软件发展如此迅速，而自动化测试却总是迟迟跟不上脚步，何解，何解？

一、 自动化测试的形式

吾自从入道于自动化测试一小段时间之久，从当时的无知到现在的懵懂，也算是一个简单的过程，从当时的存脚本化，到现在的所谓平台，也有一段艰难的过程，不断探索，却发现领域之广，难度之大，超乎想象。也研究过一段时间大型公司的自动化策略，也探索过一些中小型公司的自动化情况，发现，原来，中国的自动化测试还远远处在探索阶段。一些大型的公司，例如：华为、中兴对自动化的测试探索长达八年之久，现在才渐渐的着手于自动化测试 LAB。而百度、搜狐等这些互联网企业从 06 年自动化测试兴起开始便慢慢渗透于自动化测试，到如今，也是有了自己的重量型框架。可是中国还有一大批的中小型企业却仍处在艰难的探索当中，漫漫长路，唯有希望伴随。

二、 自动化测试的阻扰

随便想想，个人觉得中国现今自动化测试未能成规模的原因：

1、国内的企业样式复杂，而自动化测试本身也是一个技术需求性较大的一个领域，即，其不是简单的技术复制就能成功的，需要结合企业实际情况和具体需求分析，才能有所建树的。而国内太多的公司往往不考虑自身公司的实际情况，就想复制那些已然成功的大型公司的自动化测试策略，那是会得不偿失的。
2、国内企业浮躁，其自动化测试是一个前期需要长期投资的过程，而很多公司因为一下看不到收效，所以放弃。但是这也难怪，中国小型公司的生命周期太短，往往看到的是短期的利益，而自动化测试的过程太长远，其规模化，不是很适合一些中国的小型公司。
3、对于层出不穷的自动化测试工具，很多公司简单的将自动化测试定位于自动化测试工具使用，因而太依赖于自动化测试工具，所以造成自动化测试失败，从而对自动化测试失去信心。可以想想，哪个有着成功的自动化测试经验的大型公司会把自己的自动化测试命运交给一个所谓自动化测试工具，例如：GOOGLE 就是首要用的就是开源 selenium，百度和搜狐等企业都是用的各种开源的自动化测试工具，再结合开源的自动化测试框架，不满足需求的则自己开发。
4、没有合适的自动化测试人才，合适的自动化测试人才不仅需要强大的技术能力，更需要其业务能力和流程管理能力。但是真正技术牛的人，很多去做了自认为更有意义的开发，而流程管理能力强的人，当然乐衷于做管理了，所以，自动化测试对人才的要求导致了其艰难。
三、　　自动化测试的发展
但是虽然其阻扰之大，但是一个领域的发展之兴起，往往是伴随着巨大的阻力的，没有其阻力，何来其推动。纵观世界上的一个领域，哪一个不是伴随着阻力，其趋势如此，不过是时间和机会的问题罢了。
1、 不论任何领域都是如此，国家治国之道亦是“先使一部分人富起来，先富带后富，最终达到共同富裕”，而可以看看，自动化测试亦是这样的道路，大型公司积累经验，然后分享经验，带动中小型公司走向自动化测试，虽然其规模化还有一段很长远的时间，但是也不会阻碍自动化测试在各个公司的热起。
2、 最近调查了一下，可以从这几个数据说明自动化测试如今正处于兴起的一个阶段，积累了很多年，自动化测试总算小露头角；
首先，看看这最近几年自动化测试的关键技术的发展，从最初的线性脚本到现在的关键字驱动，从最初的简单运行到现在的框架化、平台化。从最初的无学术概念到现在的学术探讨阶段，可以搜索一下文献，会发现最近一到两年，自动化测试的文献是一个增长的热期。

然后，看看自动化测试工具的出现，不论商业的或者是开源的，这说明了市场再开始慢慢接受自动化测试，不然，你以为这些公司吃饱了撑的，会去做赔钱的买卖？
之后，搜索一下百度知道，你会发现今年关于自动化测试的提问多了很多，以前基本没有，这说明很多人或者很多公司部门开始接受自动化测试，开始接触到自动化测试。
再看，最近，自动化书籍的大量出现，说明了自动化测试出书是挣钱的，所以自动化测试的兴起不可谓没有道理。
3、 软件模式的发展和新型技术的出现，虽然敏捷模式还在探索阶段，但是其趋势很明显，而自动化测试在其的作用更是明显。而云计算和云测试中需要自动化测试的作用更大了，技术推动领域，领域依靠市场。

总之，这是我个人的一点小看法，无所谓真，无所谓假，是也不是，只需等待，好也不好，只需静观，与同道之人共勉 ——散步的 SUN

测试之途，前途？钱途？图何？

测试之途，前途？钱途？图何？
序言：在一篇日志中，有朋友问我：“你认为测试这个职业里面哪一个方向最赚钱，至少可以与开发相媲美”，为此，我不禁陷入沉思，曾几何时，我也为这个问题徘徊过，没有谁，一开始就能看得清路的方向：
测试之途，前途？钱途？图何？
测试之路，路在何方；
测试之殇，奈何，奈何；
一、测试之途-测试领域的切入点
————无论做什么，需求决定导向，心态决定走向
踏入测试这个领域，经过一段短暂的新鲜期，接下伴随而来的肯定是单调与迷茫，肯定会有一堆的思索，一堆的无奈，一堆的叹息；透过测试这个行业，很多人都无法看到未来的路；
大家之所以迷茫，个人觉得，原因有几个
1、环境问题；在中国，对于测试的重视程度远远不够。
2、工作技能问题；很多测试，都包含着大量的重复工作，无法透过时间，积累到自己的核心竞争力。

3、个人喜好问题；毕竟测试是质量的监督人员，大都数人对于这个职业的兴趣不高，只是开发人力的欠缺，委屈求全而已。

很多人都喜欢拿测试与开发相比，其实这也是对的，但是大家在相比的过程却没有抓住本质，很多人看到的是工作事情的不同，到手的工资不同等表面。

测试与开发到底在本质上有什么不同，为什么在中国很多公司，开发就一定比测试拿的钱更多，相通点，其实就是开发为公司创造的价值看起来比你多，其给公司带来的收益看起来比你多

一个公司创业之初，最重要的人才是销售。

一个公司发展之中，最重要的人才是管理。

一个公司发展之大，最重要的人才是财务。

所以说，每个公司，每个阶段对人才的需求不同，关键在于如何去把握这个阶段和需求。

因此，测试之途，个人觉得，不是发现哪个行业最赚钱，不是要去与开发相媲美，而是去认识本质，抓住需求，这就是切入点，摆正心态，既然已经进了测试领域，如果你不排斥它，那么你就要相信它其存在的价值，关键是你如何运用它。

二、测试之途-测试领域的发展

————每个领域，你越是钻进去，你就会感受其庞大，感受自己的渺小。

个人觉得，其实测试领域哪个最赚钱，让谁说，谁也说不上来，因为本身就没有可比性；

有人说，测试发展分为管理和技术方向，这是对的，但是，我觉得，这两者是相辅相成，缺一不可的。

一个不懂技术的人是做不好管理的；一个不懂管理的人也是做不好技术的。（纯粹在商业和市场角度）

不懂技术，只懂管理，以何服人？不懂管理，只懂技术，如何做好需求与沟通？因此两者都需要不断的进行学习和总结。

测试领域潮流趋势，我想了一下：

1、云计算的发展，因为云端的原因，计算量和数据量的急剧扩张，测试的难度将会越来越大，因此测试专家将会越来越受到重视

2、持续集成的发展，因为其开发模式的变化，其对测试人员的要求越来越高，其有着扎实的开发能力的测试人员将会越来越受到重视。

测试领域职业发展，我想了一下：

1、产品或需求路线；由于在测试过程中对产品的认识，对客户需求的把握有了一定的积累，因此很容易走上这个路线。

2、管理路线；通过对测试流程的整个把握以及与人沟通方面的锻炼，走上技术组长、主管，甚至到经理的路线。

3、测试专家路线；通过在测试理念上的不断学习，对其专业测试技巧有很高的认识水平。

4、技术与构架路线；对自动化测试的研究，对自动化测试架构的把握以及自动化流程的把握，走上了测试开发、自动化测试架构设计直到自动化流程管理的路线。

5、QA 路线；通过对质量保证流程和规范的认识，对整个公司的质量流程进行监控的路线。

6、开发路线；软件测试人员在测试过程中展现了其良好的编码能力而走上了开发路线。

7、咨询或培训路线；

……

所以说，测试领域的发展路线很全面，其实，其发展宽度，比开发人员要宽，但为什么大都数测试人员比不上开发人员的原因就是其掌握的深度不够深，远远没能达到其提高的需求。就像做自动化，你永远如果只停留在录制和工具的使用上，不能上升到需求的分析以及框架和流程的把握，那么永远只是表面。因此，通过对自己性格和兴趣的分析，以及对整个测试行业的把握，最终确定自己发展的方向，一步一步来，以需求为导向，在需求中求发展。

三、测试之途-测试领域的积累

————积累是一个很神奇的东西，它能让水滴滴穿石板；能让铁杆磨成细针。

也许，无论别人怎么说，我们不自己经历，是无法看透本质，是无法懂得，因此我们还是迷茫；

但是，我们一起记住，我们迷茫，但是我们不放弃努力，不放弃积累，不做”温水中的青蛙”；曾记得大学时，自己也是迷茫，这个学点，那个看点，后来毕业时，很是后悔没有精通一个东西；后来才发现，上班时，很多东西都用上了，至少我发现，我的自学能力提高了很多，这个才是最重要的。

因此，相信自己，如果决定好了，那就一直积累下去，越迷茫，越是要积累，不断学习，不断总结，总有一天，等你醒悟过来的时候，会发现自己的翅膀早以丰满，是自己要开始飞翔的时候了。

如果不去积累，那么你的翅膀永远还是纤细无弱的。

不管如何，越迷茫越是要走得勇敢与坚定，相信，迷茫的日子终将过去，而自己准备好了么？

四、测试之途-测试领域的共勉

自己只是说了一些自己的感悟，也许正确，也许错误

自己也是一个寻路者，一个学习者，虽有心帮忙，但奈何经验不够，但很想携同有志之士能够一起去收获点什么，一起去感受点什么；

测试领域的海洋很大，不知什么时候自己能够真的能够把握住自己的航道，但是希望在这无边的海洋中，能够有一份希望，有一些船只一起航行而已。与君共勉吧。

自动化测试同行—你是否也有这么一种感受

自动化测试同行—你是否也有这么一种感受

预言：我在讲一个故事，一个关于“自动化测试”的故事……

故事开始在有一天，你知道了一个“自动化测试”的东西

某一天，你拿起自己编好的一个脚本（TCL、javaScript 等），看着其设备或者软件听你的话自动化运行着，会不会兴冲冲地在心中咆哮，“哥也是懂自动化的人了”

某一天，当你用自动化测试工具（QTP、RFT）录制了一个脚本之后，看着其软件或者 web 重新回放了一遍之后，会不会轻视的哼了一句，“自动化就是这么简单……”

后来，你知道了有一个叫做“自动化框架”的东西

某一天，你用脚本或者 VC 拓展某个脚本的包，写了一个界面，简单的讲脚本分了下层。看着其界面能够调用那些脚本写成的项目，你是不是会小小的骄傲的咧咧嘴，“我也做了一个自动化框架嘛”

某一天，你在自动化测试工具里面（QTP、RFT）里面用了一下所谓的分层思想，将所谓的框架分了个几层，美其名曰“对象层、函数库层、用例层”。看着其框架正常的运行和维护性的降低，你是不是会美美的笑一笑“我总算明白了框架的思想”

后来，你知道了要弄一弄“自动化平台和流程”了
某一天，你发现框架只能作用在平台上才能发挥其最大的效用，于是你构建了一个包括版本管理、数据与资源管理的小平台，你低调的想一想“我应该建了一个平台了”
某一天，你将其都做成了一个流程，并开始了一些规范，你有点晕乎了“自动化应该可以开展下去了”
后来，你又不小心多学了一点，多看了一点，多交流了一点
某一天，你学习到了思博伦自动化测试，原来它们的底层架构用的是 DLL，它们的流程管理用的是 iTEST。
某一天，你看到了华为的自动化测试平台，原来他们的一个简单的工作就是你这个框架……而他们的平台是构建在整个流程上的。
某一天，你了解了百度的自动化测试平台，原来他们的大平台已经成熟得你只能仰望的程度
原来自动化还能这么做……
后来，你迷茫了，才发现自己好像什么都不会
后来，你无语了，才发现框架无所谓框架，是你自己一厢情愿罢了。
后来，才真正明白了那句话“越学越谦虚.....”，要么你就别学了，要么你就谦虚点吧。
后来，真的知道，做什么都像爬山，你使劲的往上爬，越高才看得越远，越远才发现世界越大。
后来，你开始学会了一句话“需求引领一切”
做自动化只能一步一步来，看的越多是件好事，但是能力没跟上，就是件坏事了，那么只能先从需求开始入手了，一个阶段满足一个阶段的需求吧，当需求临近饱和时，也许就是开始另一个阶段的时候了。

测试领域的发展和学习（我们都是温水的青蛙）

测试领域的发展和学习

------其实我们处在测试领域中，就都如处在“温水中的青蛙一样”

最近很长一段时间都很忙，突然对自己最近的学习方法做个简单的总结，以及个人对测试发展的整个过程的一个分析，希望对大家有帮助

一、测试发展过程

我认为一般的测试过程发展：测试执行->测试用例撰写->自动化测试实现->测试流程与平台实现。

《1》测试执行阶段要求：1、会看文档（即会看测试用例）。2、有一定的业务知识。3、有一定的工作操作和仪器仪表使用技巧。 而在此过程中也许你对自动化测试有一定的认识，觉得自动化测试就是依靠简单的脚本代替人的一部分手工测试。

《2》测试用例撰写阶段要求：1、对产品的认识和业务知识掌握到了一定深度。2、对测试理念和各种测试知识学习到了一定程度，至少对软件测试或者系统测试等原则和方法有了深刻认识（例如：黑盒测试中的各种方法、执行测试用例需能恢复到干净环境的原则等）。 也许在此过程如果你没有特别专研自动化测试的话，那么当然你还是停留在脚本执行的理念中，当然按照此路线向 QA 和管理发展也是不错的选择。但是如果你有想提高技术的想法，那么就得对自动化开始进一步进攻了。说句自己的理解：测试管理有点虚，如果你不是测试专家或者在管理确实有独特想法，那么你何以服众，技术为王吧。

《3》自动化测试阶段实现：整个测试流程，从单元测试->集成测试->系统测试->（回归测试）各对应有各自的自动化测试方法和工具。单元测试过程中，依靠的是 TDD 思想（测试驱动开发）。系统测试过程中包括：功能测试（B/S 测试与 C/S 测试），性能测试等。

自动化测试也有自己的一个过程：工具使用->工具实现（脚本开发）->框架搭建->平台与流程的建立。各有各的思想和理念，得好好学习，这里就不深究了，有兴趣的可以一起探讨。

《4》流程流程与平台实现阶段：测试的作用就是质量监控和保障，如果不建立一个良好的测试流程与平台实现，平台中包括手工测试执行与自动化测试实现，两者并行，手工测试执行负责发现问题，自动化测试负责保障质量。到了这一部，则需要对整个测试流程有很深刻的认识，可以考虑去学习一个测试管理工具的流程思想，将其的理念变为自己的理念。

二、测试技能学习方法

因为测试本身是一个涉及很广的领域，它的本质就是质量保障，所以它的要求不是要求你去专研某一个知识，而是去专研其实用性，如何更可能的节省测试成本、如何更可能的保证测试质量。所以其需要的技能是多方面的，因此我个人总结了一套学习方法，就从自动化测试这个领域来说吧

自动化测试涉及太广，真的想做好自动化测试，它不仅要求你对系统业务、编程、系统框架搭建有所专研，更是需要在测试流程与平台建设方面有所认识，而我觉得更行的学习方法为：

首先将学习当做一个流程，为学习建一个基本框架，明确自己要达到的目标，分别根据以下三部门进行列出。

1、实践精通式学习：这部分是你需要重点掌握的，像自动化测试过程中，需要重点掌握的就有：编程技巧（脚本语言，例如：javascript、tcl、perl、ruby 等，一种即可；面向对象语言，例如：JAVA、.net 等，一种即可；WEB 编程技巧：客户端与服务器端）；HTML 与 XML（XML 一定要学会）

自动化测试工具学习(单元测试工具 :xUNIT 等 ;WEB 自动化测试工具：selenium、QTP、Watir 等；GUI 自动化测试工具：RFT、winrunner 等；性能测试工具：loadrunner 等）

数据库知识（SQL 查询语言；一些数据库的使用）

操作系统知识（Windows:DLL、COM 组件、环境变量等知识一定要是掌握的，不然很多东西都很难了解本质）

业务知识（电信业务中的各种协议知识等）

流程知识（软件测试的艺术等）

当然还有很多，得具体看个人喜好了。以上是需要一变学习，一边在实践中操作的，光学习是没有用的，一定要学会把以上知识用到实践项目中，方能迅速提高。

2、目录审阅式学习：这部分知识不是需要精通，但需要认识的，不会有很大的精力花在上面，一般都是很多方面的经典书籍，例如：

单元测试过程中的书籍：Java+development+with+ant 等

自动化测试过程中的书籍：自动化软件测试—入门、管理与实现等

软件测试过程中的书籍 :有效软件测试——提高测试水平的 50 条建议。

像这些书籍，都是一些辅助性的提高参考书籍】因此可以采用“抄目录”的形式，然后对照目录快速找到自己需要的重点知识学习。我们时间太宝贵，需要学习的东西太多，自己斟酌着哪些知识需要采用这种学习方法，因人而异。

3、视野开阔式学习：学习活的，因此要多去各大网站，多到外面去走走，去认识，这样的话，才能更好的定位自己的学习方向。总的来说，视野开阔最重要，一定要保证自己的所学能够所用。

最后说实在的，测试领域其实是一个“温水煮青蛙”的锅，没有人逼着你跑，因为各个阶段都需要各种人，你想做什么样的人，就得看你到了什么阶段，千万不要再温水中“舒服”下去，当你时光不再的时候，你那点浅薄的技能，有什么资格借助“经验”这两个字与上面叫板呢，因为下面还有一群更年轻，更有活力的“青蛙”往里面跳呢，测试领域本来就不是一个靠吃经验饭就能呆住的地方，你得尽快找准方向，跳出这个锅，才能更好的求发展。

也许我们应该少想想企业为我们做了什么，多想想我们如何去为企业或者为自己创造价值，这才是真理吧，谢谢大家。

整体思考自动化测试发展和价值回报

很长一段时间，都在思考，怎么能通俗的看待自动化测试的收效

自动化测试到底能不能成为一种趋势？

自动化测试到底能不能形成一种规模？

自动化测试到底能不能成为我们的利器？

自动化测试到底能对我们的职业带来何种发展？

一些官方总是从各种数字上面来说自动化的收效，可是那种计算就真的是准确和令人信服的吗？每个公司的情况不一样，而且每个公司有自己独特的研发模式和流程，而不同的流程所带来的测试体系和观念又不一样。

但总的来说，自动化测试肯定是会有收效的，关键就是看你如何去做。因此我从整体上思考了一下自动化测试的发展：

1、信息科技增长对公司的影响来说：如今，信息科技大爆炸，中国开始走向国际，而国际上，不仅实体产品还是软件产品，不管设备产品还是服务产品，都令中国市场应接不暇；不仅仅是种类的增加，更多的质量的整体要求大大提升；但是从古至今，中国企业很大部分关注的是其产品的有无，而对后期的质量保证却并不是特别关心，这主要是因为中国以前本身市场的浮躁造成的。孰不知，当今市场，越是浮躁，看似产品上市快，但是其造成的后期维护成本占据了整个成本很大的一部分。大家都知道，一个 BUG，越早发现，其带来的影响程度越小，如果等到其投入市场，则会带来不可预知的大的负面影响；因此，可以想的到是，越到后来，其公司规模越来越趋于两极化，大的愈大，小的愈小，就在于其对待产品质量的重视程度上，其测试会越来越得到看重，但其对测试人员的水平会越来越要求高端。

2、市场发展对产品的影响来说：因为市场的快速，其要求产品的更新和发布也需要达到更快更稳定的程度；因此，单纯的靠手工测试已经远远无法保障其产品发布和发现问题的效率；因此，自动化测试必然会是一种趋势，而且后期这种趋势会越来明显。而且，敏捷测试流程以及云计算带来的数据规模也可以说明，其要求测试的效率和测试的问题发现要求越来越高，但靠人来进行手工测试是越来越不可能达到了。

3、大型企业的测试体系来说：再从大型公司收益上来说；软件服务公司，像微软、IBM 等巨头，都有自己一套自动化测试体系和平台，以前有人告诉我，微软真正的核心不在于其产品，而是在于其测试系统平台；很简单，因此，你可以去复制一套微软的操作系统，但是你复制不了其内部测试系统；你拿过来他的操作系统，更改一下的结果就是漏洞百出，完全不能用，那是因为不经过其微软的测试系统的测试，其产品稳定性根本无法得到保障。而像电信设备商，华为、中兴等，从很早就开始投资自动化测试平台，每一套产品都有自己对应的自动化测试平台，举个简单例子，华为一套网管自动化测试平台，就能每年为华为节省数千万级以上的成本，当然，这只是显性成本，不包括隐性成本（即后期因问题维护量带来的成本）

4、世界自动化测试发展来说：据了解，现在自动化测试已经越来越得到各个大型的企业的重视，自动化测试已经经历了很长的一段磨合期，而现在已经处于初步发展期了，很多大型公司开始致力于自动化测试的推广，抛开那些大型公司发布的的商业自动化测试软件不说，一些开源软件也越来越得到发展。而且，最近，很多大型公司共同成立了 NTAF 论坛（网络测试自动化论坛），其包括 CISCO、思博伦通信、IXIA、EXFO 等二十多家大型公司，而且 NTAF 论坛共享很多测试用例及框架，不过只能成员使用；因此，可以看到，自动化测试的重视程度越来越高。

4、公司自动化测试失败原因来说：为什么，自动化测试很多时候都是公司各搞各，很少能听到公司自动化测试能规模化的样例，很多公司将其归于投资成本问题或者自身独特问题，其实不然，自动化测试是需要投资，可是如果投资不到地方，就会造成很多浪费；很多公司，很多部门，只是脑袋一热，然后就一味的找各种工具，一味的编写各种脚本，缺少了前期的需求分析和整体上的把握。自动化测试到底给公司、部门带来的什么，完全对此没有一个清晰的概念，这样怎么能不造成自动化测试的失败。

5、自动化测试人员认识水平：作为一个自动化测试人员来说，如果公司部门将自动化测试的重任交与你，那么，你就要跳出单纯的自动化测试使用的认识上，而是要从公司的产品线和整体流程上把握自动化测试过程；自动化测试不仅仅是需要技能水平，更重要的是一个思考过程；不思考，不自动化测试。所以说，自动化测试的规模不是自动化测试本身的问题，也不是公司部门的投资与产品线问题，而是个人主观问题；

6、自动化测试技能发展水平：自动化测试人员，基本的技术水平应该是全面发展，而是单一发展，其不像开发，但是也要把自己在技术水平上尽量要求高，至少能懂得开发的思想和产品的思想；个人觉得，其需要懂得

1）产品业务的知识，设备还是软件等

2）流程管理的知识及各种研发模型

3）编程能力；精通脚本思想；熟练 JAVA 或者 C++等，在进行自动化测试脚本开发或者自动化工具开发的时候起到很大作用；数据库知识，能够帮助你建立一套自动化用例和数据管理体系；操作系统知识，windows 和 liunx 必须掌握；HTML、XML 等语言，反正一句话，知识不一定要精通，但一定要懂。

4）沟通能力；与研发和领导的沟通，看似自动化测试是在做技术，其实不然，其实那是在做产品，做服务，你要向公司和部门推销你的理念和框架，那么你就得一步一步拿出东西，不能太急，也不能太慢。

作为个人，如何帮助部门和团队快速建立起一套自动化平台体系，其技术和研发水平是一个方面，更重要的是策略，如何步步深入，各个击破，是一个很值得深思的过程。

7、最后，作为测试人员，眼光要放长远，带着一颗平和的心态，测试不一定比研发差，关键在于自己怎么去对待测试，很多人想做自动化测试，是因为其含有一定的技术水平，孰不知，其道理亦是相通，做一个只会用工具或者只会因为自动化测试而做自动化测试的自动化测试人员，和只会重复测试的手工测试人员，性质是差不多的；真的要想去提高自己，那么就得耐得住学习的寂寞，慢慢学会从整体上把握；你只有从整体上给公司真正来带来了成本效益，那么你的价值才会真正得到突出。因此，共勉之。

8、最重要的一条：我不是说，自动化测试会占据大部分测试流程与体系，相反，其人工测试还是在重点，但关键的是如何平衡自动化测试与人工测试，我一直觉得：我们测试人员最大缺陷在于很多测试任务都不是挑战性的、有意义性的任务，而只是重复性的、纯手工性的测试；因此，个人觉得，测试人员与自动

化测试的分工，就是自动化测试去完成那些大部门没有意义的事情，测试人员从整体把握质量流程、包括测试用例的更新和维护、产品异常测试、自动化测试小工具的开发等。真的想将一个公司带来成本效益，关键在于测试人员与自动化测试的一个定位和分配问题，这也得从整体上把握。
总之，自动化测试发展是大势所趋，其发展到何种程度，还得看中国市场的发展和广大自动化测试爱好者，和测试从业者的共同的努力。

自动化测试开展策略分析

序言：一般而言，刚开始自动化测试时，很多时候，很多人都不知道如何入手或者还有一部分人都信心满满，决心要建设出一份大的平台，可是事实在于自动化测试面临的问题一在于技术，二在于环境形势。每个公司有不同的需求、有不同的环境、不同的人员支持，所以做自动化测试所需要涉及的外界因素太多，就如黑天鹅效应中的说法，你所自认为的白天鹅中也许就隐藏着一只黑天鹅，它的出现会导致你的整体计划崩盘。所以，做自动化测试也一样，所依赖的东西太多，就能引起的整体变化太多，所**以我觉得我们的基本策略就是：不断预测、不断总结，然后是拥抱变化**
总结的从开始到一定阶段的建设自动化测试的策略如下**(麻烦有不同想法或者别的策略的朋友帮忙补充)**:
1、 分析需求并且细化需求，自动化测试是急不来的事情，不能指望用他来解决所有问题，所以必须明确需求，将需求一步一步写下来，然后从简单到容易开始击破。
2、 评估资源，围绕人力支持、部门测试流程情况以及产品业务来决定自动化测试要先从哪一步开始走，并哪一步为阶段。自动化测试必须最终与整体的测试流程相结合，才能发挥作用，否则只会越走越远。
3、 从最小的需求开始入手，也许是一个工具或者是一个线性脚本。总之，先解决一点需求，然后从点到面。获得一个面后，将其授权，然后再做点，这样一步一步进行铺张，其实说白了，也是一个自动化测试信心和价值建立的问题。
4、 记：简单。要将一个东西发扬出去，那么它必须简单，技术人员的思维有时候总是把东西做的很复杂，因为有时候会觉得很炫，但需要做好一个东西得到发扬的话，则需要将一个复杂的东西让人看起来很简单。一个工具或者一个框架，最好只有一个修改入口和一些 API 拓展机制。让测试人员用起来和拓展起来都很简单。

5、 CBB：CBB 在软件开发中俗称“软件模块共享”。而在自动化测试中也是一样，要建立自己的开发库，不仅提供给以后的测试开发使用，更是给测试人员使用，能够在其基础进行很快速的共享和拓展。

6、覆盖率分析：单纯的用例自动化很难突显自动化测试后，其到底覆盖了多少点，通过了哪些点，一个用例的拓展也不是很好拓展。因此需要划分为点的方式，即可以每个脚本对应一个测试点，每次测试，可以统计覆盖了多少个点，通过率如何，即产用产品—模块—测试点统计覆盖率的方法。

7、 ROI 分析：在自动化测试一定阶段后，做好自动化测试 ROI 分析，绝对不打没有目标性的仗，我们到底为了什么做自动化测试，很多人会说是为了保证质量，首先，大家都明白，自动化测试不是用来发现问题的，这个说法没有错，但是问题在于，好不容易做起来的自动化测试，结果没有好的 ROI 分析机制，乱做了一通，该做的测试用例没有自动化，不该做的做了一堆，结果导致自动化测试的开发和维护成本很高，收效成本很少，所以，到中期阶段，需要有一个 ROI 分析机制帮助评估自动化测试脚本的建设。

8、 成熟度模型：现在业界有一些人已经提出了自己的自动化测试阶段，这些阶段在一定基础上是值得参考的，但是上面也说了，每个公司、每个部门的情况和需求是不一样的，其依赖的因素很多，所以在自动化测试发展过程中，可以从一个试点产品或者一个项目中不断分析抽象，建立一套自己的成熟度模型，然后进行推广。在每个阶段评估不同项目、不同产品的自动化测试成熟度。

总结：做自动化测试不是一件容易的事情，但也不是一件值得怀疑的事情，它有它的价值，正所谓存在即合理，也许我们做的是要找到正视它的价值，不浮夸它，也不贬低它，踏踏实实做它应该带来的效果。

自动化测试推广经验分析总结

序言：又夜深了，每天都在告诉自己不要总加班，但不知不觉就加班了；每天都在告诉自己不要晚睡，但不知不觉就到深夜了，这是不是我们 IT 人员的共性呢。

近段时间，一部分工作在进行自动化测试开发，一部分工作在于资源整合，一部分工作在自动化测试推广，而感触最深的就是推广了，自动化测试难点一在于应用，难点二则在于应用推广，再好的技术底蕴，再好的形式，没有一个好的推广过程，则也是不行的，根据这段时间和过去的一些教训，在此总结一下，希望不对的大家能指出。

一、现象分析

做过自动化测试推广的人员应该都会遇到这么几个现象：刚开始推广自动化测试时，测试人员都很好奇，都会积极的参与配合，但是测试人员的热情在一段时间之后会被自动化测试人员的不断变动性所打败，所以，作为一个测试人员，你应该把测试人员的信任值和热情度牢牢把握。

1、**自动化测试人员总是不断的更新框架**，从而导致测试脚本不断变更，造成测试人员不断去定位测试脚本和维护测试脚本，造成测试人员对自动化测试信任度降低。

2、 测试脚本在还未规模化应用的时候，则去**强行的开发大型的测试平台**，这样带来的问题就是后期的变动性造成了平台架构的变动，从而导致测试脚本和测试平台的兼容性差。

3、测试开发人员犯的一个很大的错误就是不断推出一系列不成熟的工具，即只想着去开发一个工具，而**把测试人员还是当成了“测试人员”，而不是用户**，从而造成了测试人员对测试工具的使用的畏惧性。

4、 **不够双赢**；很多时候，测试人员做的工作往往被忽略，例如：测试开发人员开发的脚本数量是一个绩效，造成对开发出的脚本质量往往不负责任，但是测试人员对别人开发的脚本和环境的维护则没法进行量化统计，这样测试人员还不如手工测试来的绩效高，这样，则造成了测试人员对自动化测试的积极性不高。

二、对现象的思考和实践分析

1、 **脚本的稳定性应该是放在第一位**的，不管刚开始的设计和想法多么宏大，你也必须克制自己，不管线性脚本，不管封装也好，从最基础开始，让测试人员首先感觉到脚本的运行顺畅性，而不是频繁的维护。这也是一个前期自动化测试的一个先锋部对，不求一下攻克，只求快速获得情报和前线支持，为后面的大军进发做准备。

2、 决定开发了一个框架的话，**那么就要先考虑到可拓展架构，或者快速做一个原型，采取功能迭代增加的方式**，一定记住的时，能抽像到底层的一定要抽象，千万不要把特性的东西写死在底层，而是要分离。不然你会发现：前期写死一个变量，也许会让你在后期的某一个拓展中造成你的所有脚本瘫痪。

3、 测试平台、持续集成平台之类的都是架子，个人觉得，要保证的是：测试平台和测试脚本是分开的，即测试脚本可以脱离平台单独执行，而不是脚本需要

依赖于平台执行，因为有时候移植工作其实是一个很大的工作量，你不能保证你的平台永远没有变化，平台好变，大量的脚本不好变。

4、**预期开发一堆的全面化的复杂的测试工具，还不如在一个点上把一个工具做精了**，让测试人员真正觉得好用，那么你的自动化测试才能慢慢被他们接受，从而配合你的工作。其实这种策略很常见：很多平台化的公司做大前都是从一个点开始，把一个点做精了，这样就可以以此为入口，从而建立起一个产品体系。**少做即是多做，这真是真理啊。**

5、 分享一个我推广自动化测试的例子：在部门内部，以一个自动化测试活动为一个项目，测试人员做为项目负责人，我则可以作为项目协助人参与。例如：一个关键字测试驱动用例项目，测试人员来设计测试驱动设计文档，包括划分的关键字点、环境、参数等。由另外的手工测试执行人员来测试执行过程中，组装关键字，最终形成测试脚本用例。关键字、文档以及脚本便作为整个项目的输出，以项目为活动颗粒的好处是职责容易细分，进度容易把握，并且最重要的是能够让测试人员主动性更强。所以，**在推广过程中，要采用各种策略让测试人员的能力提升和职业发展相结合，让他们的成果突出，这样你给他负责，他才对你负责。**

总结：其实推广过程中，最大的经验就是一定要站在不同的角度上思考问题；学会以辅助的角度去做人做事；学会珍惜别人的时间和劳动成果，学会将别人的成果最大化，学会双赢；有时候学会自己宁愿多做点，不要告诉比人；也许，这样，你得到的将是你意想不到的。

自动化测试方式策略分析

序言：上篇文章说的“自动化例行测试有效性策略”，在又一次进行例行测试的时候，安排了一个不懂脚本的测试人员进行了例行测试操作，结果这是最迅速的一次，而且还发现了一些产品问题，效果不错，验证发现，其有效性还是得到了一些提高。所以说，有时候，做自动化测试很害怕的是“完美主义”与“盲目主义”，局限于一些误区，所以一定要以“测试价值”为导向，不时的跳出来看一看其自动化测试的应用价值是不是进入了误区。这次，对一些不同的自动化测试方式的应用进行了策略分析，不同的自动化测试方式应用不同的场景，不管如何应用，提高效率则是最佳应用。

一、自动化几种测试方式

这次，自动化测试几种方式，我暂时按测试类型分成：

1、界面自动化测试

C/S 架构（或者桌面类型）界面自动化测试：包括桌面界面测试类型，当然有 windows 方面的软件界面、跑在 windows 操作系统上的虚拟机方式的界面，例如 java swing。前者采取的方式可以调用操作系统本身的 API 来构建自动化测试、后者可以采用虚拟机内的事件处理机制来完成了。

B/S 架构（或者 web 类型）界面自动化测试：其实现原理之一可以依靠 JS 来进行客户端的操作，然后寻找对象是采用了 DOM 解析技术，将 web 方面的节点进行解析定位。

手机方面（嵌入式产品）界面自动化测试：其实现原理之一也是可以依靠其相关系统提供的 API 来完成。例如：安卓的 monkey 就是安卓提供的一个提供动作操作的一个工具。

2、命令行自动化测试

CLI 自动化测试：即嵌入式的产品很都基于嵌入式操作系统完成，例如：电信产品中的 ciso 路由器就是最早的代表，而系统测试人员的测试大都应用命令行进行测试，所以，其自动化测试实现可以基于 CLI 的脚本控制实现。当然，对数据库 SQL 命令行的操作也可以归为此类。

3、API 自动化测试

API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节，其实就是软件中的封装性的思想。个人认为，API 测试是用来验证组成软件的那些单个方法的正确性，其可以包括为单元测试、单个组件测试以及接口测试。一者是对单个模块功能测试、另外一者是对测试系统组件间接口的一种测试。但本质上都是调用其 API 来验证相应相应功能实现或者交互。

二、自动化测试不同方式策略分析

以前总在思考如何能够最大化的应用界面自动化测试与 CLI 自动化测试或者 API 自动化测试的不同特点去实现各自的自动化测试最大利用率：

（1）界面自动化测试的特点在于：

1、 操作方式**复杂**，需模拟出不同的手工操作。

2、 其界面元素结构层次变动性较大。

3、 其容错性查，如果某单个测试用例中途遇到问题，则造成其单个测试用例无法进行下去。且与软件性能关系较大，容易造成软件测试中断。

4、 错误定位方面，需要靠直观的方式进行定位。

所以，综上，策略为：

1、界面自动化测试需要进行架构的分层，对于界面控件的查找需要基于一定的属性搜索定位，而常用的录制方式一般是依靠结构层次定位。

2、尽量将测试用例按测试模块细分，在中途出现问题，则可以进行 down 掉，再进行下一个测试用例即可，而且测试用例越细分清楚，则错误定位越简单。

3、当然，大范围做的话需采用框架，但小范围回归测试的话，则采用录制+二次开发的方式会比较高效。所以，不同的方式采用不同的策略是很重要的。

（2）命令行或者 API 测试特点在于：

1、 操作方式**单一**，都是进行命令行的输入。

2、 其命令行变化小，但是不同设备有不同命令行集成，脚本数量多，命令行更改也容易造成维护量大。

3、容错性较好，一般命令行自动化测试都是采用输入+回显判断的方式，所以，不易造成测试中断。

4、 错误定位方面，靠日志记录的方式即可。

所以，综上，策略为：

1、由于其单一操作性，对于一些简单的回归测试，则可以采用 CLI 录制的方式，生成一些简单的回归脚本，这样，测试人员也可以进行简单的自动化测试回归设计，辅助提高测试人员的测试效率。

2、 对于需要大规模进行例行环境建设方面，更多的是考虑维护量方面的问题，则可以采用关键字驱动的思想，将设备映射成一系列的关键字对象，将其属性进行封装，这样，可以提高重用性和降低维护量。

总之，个人觉得，自动化测试应用的最大策略就是“因地制宜”，主要是抓住其测试方式的特点，然后以不同的策略去对待，这样，才能真正快速应用自动化测试提高测试效率。否则，很容易陷入了“为做自动化测试而自动化测试”的陷阱。

自动化例行测试有效性策略

序言：自动化例行测试作为自动化测试中一种测试流程中的方式，对于产品线的质量有很大的保证作用，且对于节省人力成本也有其独特的方法。而在进行自动化例行测试有一定阶段之后，发现自动化测试技术是一个方面，但更重要的一个方面的其需求挖掘与应用分析的小策略方面，这些小策略对于提高自动

化测试的效率也许会起到一定的作用，但也不敢太过于保证，毕竟环境不一样，差异也是很大的。

一、自动化例行测试概述

自动化例行测试即部署一套自动化测试环境，针对某一个产品线，进行有针对性的回归测试，目的在于保证产品版本迭代稳定性。

二、自动化例行测试的的几个环节

这里从已经开展的角度上来说，并不包括自动化测试框架的开发阶段，其简单的几个环节包括：

1、**自动化例行测试项目选择阶段**：即对产品线中需要进行的测试项目进行选择投入到自动化例行测试环境的阶段。一般是从手工测试用例模块中进行选取。

2、**自动化例行测试项目运行阶段**：即自动化例行测试项目的运行的方式以及自动化测试项目之间的运行优先级的排列计算。

3、**自动化例行测试项目维护阶段**：即对自动化例行测试项目中的维护，包括自动化测试用例、自动化测试点、自动化测试脚本以及自动化测试环境部署的维护。当然，还包括自动化测试框架的维护，但这里不作为重点。

4、**自动化例行测试项目跟踪记录阶段**：即对自动化例行测试项目的运行情况、结果以及维护情况的一个跟踪记录。

5、**自动化例行测试项目淘汰阶段**：即对例行测试环境中的项目的一个周期性的选择和去除阶段。

三、自动化例行测试的问题误区与策略

1、 自动化例行测试项目选择阶段；

个人根据经验发现，**在开展自动化测试前期过程中，很容易进的一个误区就是覆盖率，认为覆盖率越大越好，这样就很容易导致自动化测试的失败**，原因在于自动化测试本身就是一个高维护性的领域，而越大的覆盖率就意味着更大量的维护问题，而且往往这样的问题是发生在自动化测试开展的前期，所以，其自动化测试质量都很差，这就造成了测试人员疲于自动化测试的维护，造成了自动化测试的废弃，以前听说，百度在刚开始自动化测试时，也是因为在前期看到自动化测试好处而一味追求覆盖率，最后自动化测试的覆盖率达到了 80%左右，最好却是失败，直到现在自动化测试的覆盖率已经降到了 30%左右，才趋于一定的稳定和作用了。所以，**宁缺勿滥是其自动化测试开展前期的一个策略，也是自动化例行测试项目的一个选择的策略。而其选择性则可根据产品测试周期、产品的核心模块以及产品的稳定程度以及测试所需资源来综合考虑。**

2、 自动化例行测试项目运行阶段；

个人觉得，自动化例行测试运行的几个关键在于：1）、测试环境的稳定性（最好能拥有自己的一套独立的测试环境，排除外界影响）。2）、测试运行阶段的稳定性（遇到不同的异常能够抓取，记录异常，且测试进程不能停止。3）、提供实时的监测机制（即遇到需要变更自动化测试环境拓扑或者网络不可连接的情况下，可以进行测试人员的通知）等。4）例行测试项目的优先级选择；一般而言，其优先级可以根据版本的改动影响情况以及上次错误出现项目情况等进行选择，在时间不够充足的情况下，应尽量保证其更核心项目的验证。

3、 自动化例行测试项目维护阶段；

个人觉得，**自动化例行测试的价值就在于例行测试回归的次数**，而在自动化测试前中期，我们很容易进入自动化测试脚本的维护误区，**即过于关注自动化测试脚本的缺陷**，一定要保证所有脚本没问题才有信心进行例行测试项目回归。其实一定阶段，脚本错误只是占项目的一小部分，所以在保证了大部分无错的情况下，可以先将错误项目排除，运行无错的项目，在运行过程中再进行更正，尽快保证例行测试平台可以得到利用，保证 100%的正确率是不可能的。则策略是，在保证问题项目在整个项目中的比重比较少的情况下，在测试过程中，**可以暂时将其忽略，尽快去保证其余正常运行的项目得到验证，然后边运行边进行错误问题的调试**，如果测试项目有需要，可以将手工测试结合起来，关键是保证其例行测试平台的运行。使其尽快发挥自身价值，然后可以进行其回报分析，这样，才能对自动化测试下一阶段的开展提供铺垫。

4、 自动化例行测试项目跟踪记录阶段；

需要有一个随时跟踪例行测试项目的记录文档，一是产品缺陷造成的脚本问题。二是无法修复的脚本问题。三是常见的脚本错误及修改策略。这样，可以保证以后脚本或者产品出现问题时进行良好定位。

在项目结果验证方面，一是提供一个统一的结果及链接。二是提供一个测试点的详细记录。三是提供一个具体的步骤日志，方便分析。

5、 自动化例行测试项目淘汰阶段；

其策略则是根据产品线的测试需求情况进行选择。

总之，个人觉得，在一定策略中，快速让自动化例行测试运作起来，然后在自动化例行测试开展过程中需要在**每一个过程周期期对其回报价值进行简单的定位**（例如：前期可以以测试点的验证程度或者测试用例的验证程度来进行判定，中后期可计算金钱回报率等方式使得真正可以让自动化测试的应用落到实处），

这样将自动化测试的应用以直观的方式体现，**能对自动化测试下一个阶段的预估开展起到更好的分析**。即是一种**边走边分析**的策略。

自动化测试用例设计

序言：自动化测试中，自动化测试用例是一个重点中的重点，个人以为，到底如何去定位自动化测试用例设计的形式和发展是决定自动化测试成败的关键，根据一些研究和看法，我写了一个自动化测试用例设计的一个大概情况，当然一家之言而言，当然，大家在测试过程中，接触过自动化测试的，肯定就接触过自动化测试用例，其是自动化测试脚本本身也是一种自动化测试用例，看看以下的情况大家遇到过么，希望大家有什么想法，提出来吧。

一、 自动化测试用例应用

手工测试用例是针对手工测试人员,自动化测试用例是针对自动化测试框架，前者是手工测试用例人员应用手工方式进行用例解析，后者是应用脚本技术进行用例解析，两者最大的各自特点在于，前者具有较好的异常处理能力，而且能够基于测试用例，制造各种不同的逻辑判断，而且人工测试步步跟踪，能够细致的定位问题。而后者是完全按照测试用例的方式测试，而且异常处理能力不强，往往一个自动化测试用例运行完毕后，报一堆错误，对于测试人员来定位错误是一个难点，这样往往发现的问题很少。所以，根据其各自的特点，需要将两者有很好的定位：手工测试是在软件版本前几轮测试的重点，目的是验证功能，发现问题；自动化测试是应用在后几轮版本，保证软件版本模块修改或者添加新功能后，没有影响开始的功能模块（因为软件中，各模块之间的接口以及类、函数方法等的互相引用，也是容易出问题的地方）。

二、 自动化测试用例设计发展

1、自动化测试用例设计发展前期

记得，当时的自动化测试开展是针对测试设备设计一套测试环境系统，用于自动化例行测试，根据此，专门撰写了一套自动化测试用例，并转化成自动化测试脚本。之后整套系统都失败了，失败原因包括：

a、系统太过于庞大，测试用例也过于繁琐，每次测试运行下来，测试结果都夹杂着一大堆各种错误，有可能是产品问题，有可能是脚本问题，也有可能是用例问题，这样下来，测试人员每次运行一遍都要面对大量的问题，维护的几次就放弃了，问题越来越多，根本没办法去定位，这样反而浪费了大量的成本和时间。

b、搭建的一套测试环境系统，各个产品功能模块都互相联系在一起，都互相有影响，容易造成问题。

c、更重要的是，由于是单独搭建的一套测试环境系统，其自动化测试用例与手工测试用例没有太大关系，这样就造成了其自动化测试很难对手工测试进行辅助。

2、自动化测试用例设计发展前中期

这时，自动化测试用例来源于手工测试用例，首先，自动化测试根据手工测试用例进行转换而来（举个例子：CLI 测试时，自动化测试用例是根据手工测试用例的步骤，将其需要输入的 CLI 命令和回显进行填写），之后，自动化测试脚本人员根据其自动化测试翻译为脚本。这样做的好处就是手工测试用例与自动化测试用例的结合，保证了自动化测试的明确性，后期的改进还包括

a、将自动化测试用例根据手工测试用例进行了分层，把一些共性功能和全局变量抽象到了更上一层，保证复用性和降低维护性。

b、设计的自动化测试框架的管理。

经过一段时间之后，问题还是很大，主要问题在于

a、前期为了追求自动化测试覆盖率，往往忽视了自动化测试用例的质量，大家想想，出产一个自动化测试项目，得经过手工测试用例—自动化测试用例—自动化测试脚本三个阶段，而自动化测试用例是手工测试人员来撰写，因为其懂业务；自动化测试脚本是由专门的自动化测试人员撰写，但是这导致了一个项目出产的周期长（需要经过两个阶段，而且还要反复调试），而且由于经过了两个阶段，所以伴随问题也多了，特别是接口这方面，往往由于两者的沟通和认知问题，使得自动化测试项目的发布后还隐藏大量问题，往往使测试人员疲于调试和维护。

b、虽然是按照手工测试用例设计出来的，但是由于手工测试用例开始没有考虑到自动化测试者一块，因此手工测试用例的每个测试模块往往测试步骤很长，而且测试前后不能保证测试环境的恢复，所以也造成了后期一个自动化测试项目的问题以及出产成本。（这里建议最好自动化测试用例的每个功能模块的步骤不长，而且每个模块之间不要有联系，这样是要从手工测试入手的）

3、自动化测试用例设计发展中期

在这个过程中，最好的是测试人员能够根据自动化测试框架与平台来自行进行测试脚本的设计，往往在前几轮的测试中，手工测试人员就能在手工测试的同时将自动化测试脚本设计了出来

a、录制就是这样一种思想，测试人员在测试过程中，本身就是在测试过程中，将测试人员的测试过程进行了记录，所以，我的想法一直是，录制的方向是没错的。

b、而针对 CLI 测试，也可以制作了这么一个录制工具，就是模拟制作一个超级终端，测试人员应用其进行手工测试，其工具自动记录其命令行操作，而且手工测试人员能进行回显的需要匹配的某一个字段进行选择，然后在后台根据模板，自动生成一个自动化脚本，之后测试人员自行进行维护即可，这样可以缩减环节，降低出错几率和维护问题。

c、当然，要做到这一步，一个强大的自动化测试框架和平台是其支撑，我曾经做过一个基于 GUI 的自动化测试框架，部门里面也动员过一次试用，每个产品线都出了一两个人进行脚本开发，测试人员往往对脚本开发没什么太大兴致，更多的是一种好奇，而且调试工作大多是我做的，整整 10 多个产品线，调试了我大部分时间，因此其难点本身不是在于脚本的开发，因为当时测试人员开发出这些脚本是很快的事情，其真正难点在其测试人员的调试和维护方面，而且测试人员往往疲于去进行这方面，他们在乎的是测试和业务本身，要协调这方面的冲突是一件很难的事情，需要不断思考和总结吧。

4、 自动化测试用例设计发展中后期

留一个中后期，现在的自动化测试用例设计发展还是很浅，还需要一个长远的发展过程，中后期发展成什么形式，因为见识有限，所以请大家积极猜想啦。希望能给我带来一些见识和想法。

三、 一些感想

下面感想仅我个人之言：自动化测试就本身而言，其实技术方面的前瞻性是需要的，但是自动化测试确实是一个长期的过程，对自动化测试的定位很重要，然后是一系列的细节问题，每一个问题都是需要值得重视的问题；定位、细节、技术真的都是缺一不可啊，而且还要把握这么一个平衡，想到自己以前注重需求，后到注重技术，后又因为技术怀疑需求，最后又回归到认识需求，短短时间，总有变化，现在想想，三者，技术是基础，需要不断学习，但却不能看的太远而忽略了现在的最重要的需求，在实现现在需求的同时，不断步步跟进，进行探索。自动化测试还真是一个“步步惊心”的过程啊。

自动化测试的效益分析之道

序言：做过自动化测试的人员是否对自动化测试的效果有一些疑问，到底自动化测试给我们的测试活动带来了何种效益，发现问题的能力实在不高，但只是简简单单的说是保证质量又觉得无法说服自己，更别说说服别人。我个人简单考虑了下其自动化测试直接和间接带来的一些效益，**希望得到点拨**。

一、自动化测试的几种效益

个人觉得，自动化测试按照重要程度的效益分级，分为以下几种：

1、 自动化测试加速了版本发布的周期

2、 自动化测试保障了产品质量

3、自动化测试发现了一些产品问题

至于自动化节约了人工时间，而其节约的人工测试时间其根本应用还是在于其以上三个目的

二、自动化测试的效益分析

1、 自动化测试加速了版本发布周期

加速了版本发布周期，我认为，这是自动化测试的根本目标，在如今市场竞争日益增加的情况下，软件功能越来越强，而对应的软件版本迭代周期却越来越短，在迭代的过程中，不可能由人工去覆盖大量的测试活动，因此在产品迭代过程中，则需要大量的自动化回归测试的保证。

为什么说自动化回归测试加速了版本发布周期呢，这是因为一个软件产品的迭代周期是根据市场的需求变化的，做过软件开发的人员应该知道，一个良好的软件产品，是要满足高内聚，低耦合的原则，而一个新产品往往就是按照这个原则去开发的，因此其维护起来也是相对简单，但是在不断的迭代的过程中，其软件的架构就会慢慢的被破坏，互相的引用往往造成了修改的难度，产品迭代过程越长，维护起来越是难度大，因此，自动化回归测试在测试过程中不断覆盖了那些因为时间或人工成本问题不能被涉及的测试点，越早发现问题，则会越容易保证产品迭代过程的稳定性，保证了产品快速的得到发布。

2、自动化测试保障了产品质量

这个就是自动化测试通常说的效果所在，让其产品的信任度更高。

3、自动化测试发现了问题

因为自动化测试是简单测试逻辑的实现，它不可能去拥有“创造性“，一个软件产品的问题往往在测试前几轮时就大多被发现了，而且这里很多问题是由于人工测试者进行了一些“创造性”的异常操作而发现的，因此，需要将人工测试与自动化测试在流程上很好的结合起来。

三、自动化测试效益估算

其实无论做什么，效益方面的问题最终还是要转化成最直观的“资金”问题，说白了，就是“钱”的问题，到底自动化测试能够从“资金”带来了多少效益

1、自动化测试过程前期效益分析—模糊方式

刚开展自动化测试时，不可能一下子就从钱的方面去考虑，那样只会造成虚而不实，这个时候，就需要有一个长远的认识，只需简单的模糊分析即可。

2、自动化测试前中期效益分析—时间方式

当自动化测试开展初具成果时，这时候可以从“时间”来进行简单效益分析，即自动化测试脚本开发时间与自动化测试运行时间*迭代周期的一个对比。

3、自动化测试中后期的效益分析—资金方式

这个时候，自动化测试已经有一定规模了，因此可以“资金”进行分析，这样才能对自动化测试有一个很好的评估过程，哪些手工测试需要自动化，哪些需求需要满足等。（成本=测试脚本开发工作时间*参与人数*每日工资金额<<效益=测试脚本运行时间/8*每日工资金额；当然，这只是一个简单的资金计算公式，还有很多隐性的因素需要包含进去，每个公司根据自己不同的情况进行估算）

总之，自动化测试过程开展，每个阶段需要对其每个阶段的目的明确，即一定明确其带来的效益，不管是隐性或者是显性，个人觉得，都需要有一套自己的分析和估算机制；**自己随意想了一些，肯定各位自动化测试同行有自己一个良好的分析机制，希望能够分享，谢谢。**

“软件测试层次”思索，你又到了第几层？

序：今天偶然发现一篇很好的说明自动化测试境界的东西，然后分析了一下，看看我们的自动化水平到达第几个境界了？

Bill 认为软件测试有 5 个层次。

1. **Manual**
2. **Automated Manual**
3. **Automated**
4. **Frameworks**
5. **Automated Frameworks**

Manualis where you perform. the testing entirely manually.
Don't underestimate this, some parts of your system will always be better tested manually.
（思索:手工测试执行阶段，其实自动化测试只是一种质量保证，而发现 BUG 的利器往往还是“自由性”的手工测试）
Automated Manualis where you automate what was previously done manually.
This is characterised as the 'Record/Playback' style. of automated testing.
It is NON-SCALABLE and is the dead end where most automation projects end up failing.
（思索：手工半自动化阶段，大部分公司都是这种模式，只是简单的依靠录制-回放就想达到自动化测试的规模，但是往往心急吃不了热豆腐，其实想想，要是真的这么简单的话，自动化测试早就成为一种现今必不可少的流程了，还会是如今挣扎的水平？）
Automatedis when you take the process seriously and apply software development techniques.
A major aspect of this is refactoring the code base so that duplicate code is eliminated. When an object in the AUT changes 'beyond recognition', you should only have to change one line of code in the entire code base.
（思索：自动化测试阶段，有了一些脚本化的用例库，有点维护性，但是需要专门的人员进行开发用例，后期开发成本高，拓展性弱，有一部分重视自动化测试的已经达到这种水平了）
Frameworksare where you open up your automation process to being driven by non-experts.
This is the only route to a scalable process with a good ROI.
（思索：自动化测试框架阶段，很少以部门公司达到这种水平，一般都是大型公司，或者一些中小型公司借助商业型的自动化测试工具和一些有自动化测试水平的员工搭成的简单框架；这个阶段需要达到的水平就是一个不懂自动化测试或者不懂脚本的测试人员能够自己组织开发自动化测试用例，且维护性很好）

Automated Frameworksare when you realise that you can write scripts to AUTOMATICALLY GENERATE other scripts.
You can automatically detect when an object in the AUT has changed and adapt the code
（思索：测试自动化框架；与以上不同的是，这是一个测试自动化过程，而上面的为自动化测试框架阶段；这是已经从流程上实现了自动化，自动生成脚本、自动监测变化信息，自动生成测试结果等；这种阶段才是自动化测试的最高境界—“测试自动化”，也许在敏捷测试阶段的持续集成中有所模糊的影子）
总述：总的来说，自动化测试发展历程：手工测试—>半自动化测试—>自动化测试—>自动化测试框架—>测试自动化。最后那个阶段完全是从整体上实现了自动化，是我们大家都要追求的最高境界，可是这个境界离我们还是很远，而且需要花费大量的经历去做是完全不值得的，我一直觉得，满足需求的自动化测试才是最好的自动化测试，而一味的为了追求自动化测试而自动化测试则会有点得不偿失了，因此，对于我们大都数来说，追求第四个阶段已经足够，我们没有必要去做追求第 5 种境界的“科学家”和“烈士”，我们追求的“效率”和“实现价值”。但是，那第 5 个境界，我们可以改一改，我们不去实现测试自动化，但是要实现流程集成化，即将我们的自动化测试与我们自身的测试流程很好的结合起来，那么也可以给我们带来巨大的效率，所以，说白了，自动化测试不是说要达到多么高的技术水平，而是一个定位问题了。
说这么多的最终目的，还是请我们一起想想，你们公司到了什么阶段，最终又在追求哪个境界，或者，你自己又到了哪个阶段，你自己又在追求哪个境界呢？

[转] 自动化测试案例设计及读后感

自动化测试已经越来越深入人心，其重要性也是不言而喻的。性能测试中大规模并发的要求，压力测试中的大规模压力的模拟，回归测试中的大规模测试用例的反复执行都要求实现一个高可用、高可扩展性的自动化测试框架体系。因此，如何在一个开放的框架下，构建一个完整的自动化测试体系是我们需要研究的方向。

一个完整的自动化测试框架体系包含以下几个部分：1、自动化测试框架；2、测试脚本以及测试数据管理；3、测试脚本的执行管理系统；4、测试结果的显示与分析系统。其中最重要的是自动化测试框架部分。

第一部分，自动化测试框架。自动化测试框架要解决的问题，从本质上来说，是实现分布式资源透明化的过程。由于性能测试、压力测试的要求，我们往往需要构建一个分布式的测试环境，在这个分布式的测试环境中，我们需要多种测试平台（例如：多台 windows，多台 linux 等）。自动化测试框架的作用就在于将分布式环境中的各种资源变成相应的服务对象。例如一台 windows 机器，在自动化测试的框架中，我们看到的将不再是一台 windows 机器，而是绑定到某一个 IP 地址上的一个服务对象。通过这个对象，我们可以通过一个通用的调用方法（本地调用一个远程提供的方法，需要采用对象映射的技术），告诉这个对象，让它做我们希望它去做的事情，例如启动一个指定的测试脚本（这个测试脚本可能是我们日常写的某一个测试用例，也可能是其他操作）。在自动化测试框架的实现上，其主要是建立了一个以提供服务为主的底层的通讯网络。而在服务的应用上，我们可以采用插件模式，以及对象映射的技术，可以动态的无限的扩展我们的服务。根据我个人的实践，STAF + python 的开发模式可以很好的实现这个框架。STAF 主要构建了一个网络体系，使得各种机器资源之间可以自由的通讯。而 python 则可以在 STAF 的基础上进行二次开发，可以构建一个动态插入的服务体系。

第二部分，测试脚本及测试数据的管理。首先要选择一种合适的自动化脚本语言。一般来说，需要考虑以下几个方面：（1）高可读性，（2）无需编译，（3）可扩展性，（4）强大的第三方支持，尤其是对各种数据源的支持。我们可以采用 CVS 或者 SVN 的方式来实现对测试脚本和测试数据的管理。在这里，主要依靠高度组织化的目录结构来实现，尤其是需要和实际测试过程中的测试套件，测试模块以及测试用例的组织结构进行匹配，分级管理。形成一个完整的测试脚本和测试用例的资源库。对于测试脚本的编写，有一些基本的要求：1、形成一套测试脚本的编写规范；2、测试脚本采取分层设计思想，持久层（数据资源库，对象资源库，统一 IO），逻辑层（封装基本业务逻辑，实现 API 级调用），脚本层（实现测试用例过程，主要是描述测试步骤）。通过这些，测试工程师编写测试脚本将会变得十分轻松，测试的效率也会有大幅度的提升，大规模回归，甚至是在第一轮测试就实现自动化测试也不再是梦想。

第三部分，测试脚本的执行管理系统。大量的测试脚本编制好了以后，一个很重要的步骤就是大批量的执行这些测试脚本。通过 CVS 或者 SVN 的管理，我们生成了一个测试资源库，一个测试用例将是一个测试脚本。测试脚本执行管理系统的目的，就是要在用户定制的时间去执行用户选定的测试用例。测试

脚本执行管理系统也应该能动态的追踪到当前正在运行的任务的状态，例如执行百分比等等；还可以实现多用户管理，例如同时执行多个用户提交的测试需求。同时，测试脚本管理系统还应该实现测试环境自动部署的功能。一般来说，我们在进行大规模的自动化测试之前，需要准确部署测试环境，这里就要求用最新的代码版本来进行测试。因此，测试环境的自动部署也是很重要的。

第四部分，测试结果的显示与分析系统。通过统一的 IO 调用，我们可以将测试过程中产生的错误信息，日志信息，以及测试结果动态的放到我们想要存放的地方。测试结果的显示与分析系统正是基于这些数据进行处理的系统。每一个测试用例在执行的过程中，需要输出大量的日志信息，这些日志信息是非常重要的。通常，我们判断一个测试用例执行结束以后，是否有 Bug，常常需要深入分析这些日志信息。在测试用例执行的过程中，不光要打印相关的测试数据，实际获取到的数据，还要打印相应的测试步骤，这样才便于对测试结果进行分析。至于显示系统，主要是对测试结果的一个分类检索功能，可以生成各类报表，例如，一个 300 个测试用例的模块中有多少通过的，有多少是失败的等等。有一个基本原则是很重要的，自动化测试不是为了自动化，而是为了发现 Bug。如果自动化测试不能发现 Bug，那么花费大量的人力物力实现自动化，也是没有什么实际意义的。因此，深入收集测试用例执行的过程中产生的各种信息是非常重要的。个人的实践经验表明，这些信息对于发现 Bug 起着至关重要的作用（测试步骤的描述也不容忽视）。

自动化测试体系不是一个工具，一种自动化测试脚本语言就可以实现的。它需要一个完整的解决方案才能实现。个人的实践经验表明，自动化测试框架的引入、强大的资源整合能力和有效的自动化测试体系的设计将是实现自动化测试的十分重要的因素。

特别喜欢这篇文章，从整体上把握了自动化测试，我们好多人做自动化测试，一开始就缺乏全局观念，但全局观念的修炼却又是很难很难的，不仅需要懂得各种技术，最关键的是其快速的学习能力，以及从宏观上把握整个流程的能力，因此，如果真的想帮助公司把自动化测试做大做好的话，不是一个自动化理念，也不是一个自动化工具，而是一整套的自动化测试解决方案；如果真的从这上面出发的话，我相信，做好自动化测试不是一个遥不可及的东西。

畅想网络自动化测试平台与流程管理

昨天 ,有幸参加了 spirent 的北京的全球研讨会 ,主要关注其的自动化解决方案，

后来与其公司自动化专家经理长谈了一次，颇有收获.

其自动化测试主要是面向于通信网络测试的自动化测试，其框架与流程很值得借鉴。

网络自动化测试与单纯的软件自动化测试不同的是，其物理拓扑环境的搭建，以及对一系列网络设备和仪器的自动化控制，其难度之大，有甚于软件自动化测试。

思博伦通信从收购 FanFare 之后，就出了这套整体的自动化测试流程，即一个全局的自动化测试 Lab。以前的网络测试仪的自动化测试方案，无非就是基本三层—硬件设备层、适配层（脚本调用接口参数）、二次封装层（对适配层的脚本的二次封装），然后通过自动化测试人员根据 API 自行编写脚本控制其测试仪进行测试。

现在其提出的整体方案称为：NoCode 自动化解决方案，其分为物理架构、应用环境、测试例以及测试周期。物理架构说白了就是利用一个物理交换机来控制自动化测试环境的插拔线以及断上电过程，进而控制测试拓扑；应用环境即集成了一个测试用例的编辑工具；测试例即集成了图形的脚本编辑；测试周期则是对测试结果的管理等。

后来，与其自动化测试经理进行交谈的时候，主要从针对现在中小型公司的自动化测试流程与架构的问题进行了交流，像华为和中兴、思科等这些大型的企业，因为其研发水平和投资的缘故，其自动化测试做的都比较成功，成功在于其从平台的角度上把握了整个自动化测试，其实道理很简单，为什么很多中小型公司自动化测试并不很成功，原因在于其只是将自动化测试当成了一个简单的工具，其自动化测试只是发生在执行流程上，而不是从整个流程上把握自动化测试；真正要做到自动化测试，则需要从很早开始就要从整体上进行把握，不一定要做到很大，但是要从整个流程出发，关键在于

1、自动化测试用例的管理流程

2、自动化测试用例的执行流程

3、自动化测试用例的结果管理流程

而我们很多公司只是单存的将用例编程自动化脚本就觉得 OK 了，孰不知一堆的维护量在等着你（因为自动化用例的更新问题，因为通信产品命令行或者功能的更新问题）

因此，个人觉得，对于中小型公司做自动化测试而言

1、刚开始，以需求分析为主，不要急着为了做自动化测试而做自动化测试，

而是要从整体上进行分析，例如：产品自动化测试的比率（其成本价值），产品自动化测试的定位以及更重要的是自动化测试与人工测试的关系（如何均衡和使这两个部分配合达到最佳的效果）。

2、中期，等自动化测试用例达到一定数量后。可以先做到将其自动化测试执行分层次，典型的可以分为测试方法层、测试配置层以及界面层（界面层可以适当而做，像华为自动化测试平台，其界面与逻辑层做到了很好的分开）

2、中后期，当测试用例进行添加到一定数量后，开始需要从整个流程上进行把握，其实最重要的开发一个平台，集中管理自动化测试用例、集中管理自动化测试脚本以及集中管理自动化测试结果，可以专门负责对自动化测试用例的修改和完善，然后直接作用到自动化脚本的修改，以及对结果的输出与查看分析等。

这样的话，从整体上进行把握整个自动化测试流程，则会将自动化测试真正应用到实处。

总之，无平台与流程管理，那么自动化测试只能流于一种业余形式；而平台并不见得多大，而是一种思想；流程管理，也不见得多难，而是一种把控；

因此，做好自动化测试难点就在于既要从整体大局出发进行把握，还得需要从细节出发进行分析调控。

浅谈一些中国电信自动化测试模式与框架

最近突然想思考一下电信的自动化测试模式与框架，请指正

1、大型的公司，从很早就开始投入自动化测试，思科、Juniper、H3C、华为（部分 office）、中兴、华赛、Topsec 等公司采用 TCL 脚本。原因主要是由于 TCL 的可维护性和简单性，在加上其与 C/C++的易结合性；Z 大概在 06 年时开发出了 VC 的单进程的脚本平台模式，至后往多线程发展；而 H 在这上面投入比较大，随着开发的模式而有所更改，其主要为第三代—基于关键字的驱动模式，其趋势似乎是持续集成之道，有熟悉的希望可以探讨下。而个人认为，这些大型公司的自动化测试框架则是一种从开发到测试的流程平台，这个平台集成了单元、模块和系统的自动化测试。

2、中小型的公司；其主要还是利用存脚本的开发；主要用来进行例行测试及一些核心功能的验证；在其所处的境遇，自动化测试主要是对其核心系统功能起着例行与回归验证。其需要的框架是脚本集成的框架，是具体实现测试的一个框架，每个层次对应不同的步骤和配置。其主要关注的是中小型公司的特点，

以中期收效为主，步步为营。
3、单元、模块、系统、网管都对应着不同的自动化测试方法，而 CLI 与 GUI 测试各有不同的方法，总之，网管与单存的 GUI 测试不同之处在于需要结合 CLI 进行测试。
总之，不同的公司对应不同的自动化测试境遇，以上只是简单的总结一下，具体的实施都因地制宜，自动化测试最大的特点就是其没有一个通用的实现方法，只有一些核心的实现原则和框架；以上总结由于水平和一些别的原因，因此说的有些生涩...

分布式系统设计和测试总结

序言：在当今，有一门技术很是热门，那就是分布式技术，也许很多人对分布式技术很疑惑，但是实际上，你总是与分布式技术打交道，**我们若想更好的把握住以后的**测试**机遇，那么则要让自己不断的对这些系统基础知识了解，这样才能去创新**。有不对的地方请指教，谢谢啦

一、分布式系统和编程
1、 分布式计算：将一个大型的高难度的计算拆分成若干个子进程进行计算，然后计算完毕后，回收结果。这个需要区分的是并行计算，并行计算是指并发执行，分为空间和时间的并行，空间则是我们常说的多核，而时间则是利用了流水线，错开时间。他们的共同点是解决对象上，都将大任务化为小任务，这是他们共同之处。区别在于前者的每个任务具有独立性，并且更关注的是任务间的通信。后者的任务包是一个划分，之间的联系很大。
2、 分布式操作系统：一个分布式系统是若干个计算机操作系统组成，但是对于用户而言，就像一个巨型计算机一样。
3、 分布式编程：简单而言，,其主要特征是分布和通信。即将一个大型软件系统，分割成若干个模块，然后模块之间利用规范好的接口进行通信。
例如：我常用的是 Java 的分布式编程，包括：RMI、Corba 以及 SOAP
RMI 是 java 虚拟机模块之间的通信。
Corba 是通过 IDL 描述接口，不同编程语言模块之间可通过 Cobra 服务进行通信。
SOAP 是靠 XML 来描述接口，通过 HTTP 协议进行通信。

4、分布式存储：将一个大型计算机资源可以分配到不同的存储系统上，整体可以看做一个大型存储系统。例如：现在很火的 Hadoop 的 HDFS 就提供了这么一个功能。

所以，总之，分布式重点就在于：能够统一管理和分配资源，协调好各个分布式模块之间的通信。

二、分布式应用

举几个简单的例子说说分布式

1、 我们常用的 web 也是分布式的，浏览器和服务器之间的交互是通过 http 协议，而可以将各个资源分配到各个服务器，而我们常说的云计算就是这么来的。

2、 在一些 C/S 系统中，也利用了分布式，将计算和资源进行分配，互相之间利用接口进行通信。

三、分布式测试和应用

常见的测试范围

1、 模块测试；对分布式系统中的单个模块进行测试。因为分布式模块中的模块有一定独立性，所以先保证每个模块的功能。这部分可以考虑用白盒测试来实现一部分测试。

2、 接口测试；因为分布式系统之间的通信主要应用接口，所以专门针对接口进行测试。这方面可以考虑用通过自动化进行请求和相应实现一部分测试。

3、 系统测试；全面将各个模块进行部署，按照功能需求进行测试。这部分的话，可以依靠一些上层的自动化测试，例如 :WEB 的话可以是 ui 级别的自动化测试。

4、 性能压力测试；高压力和高并发下测试，各个模块之间的响应速度和接收程度等。例如：web 系统的高并发测试，C/S 系统的多个客户端同时请求服务器。

5、 模拟测试；即尽可能真实模拟应用环境。如果系统太过庞大，也可计算比例进行模拟。

四、分布式自动化测试平台

1、 测试资源可以分配到各个执行端进行测试。

2、 测试结果可以统一收取集中到服务器上进行查看。

3、 可以多个服务器保存测试资源。

4、 可以监控和跟踪自动化测试过程

总之，即**测试执行资源可以进行分配、测试数据可以分配、测试结果可以分配。但是对于整个外界而言却是一个整体系统。**

总结：不同的软件系统有不同的测试方法，**我们如果要对一个系统能够充分进行测试，一定要了解其运作原理、然后从原理出发，合理划分其测试类型和测试需求，根据测试类型和测试需求然后找到具体的测试方法，这样才能一步一步提高我们的测试效率。**不能做其然而不知其所以然吧。

基于云测试的一点思考

序言：最近看了一些比较多的云测试的应用，觉得：测试资源开始向着服务集成化和动态分配化发展，可是不禁堪忧，云测试其实也是基于自动化方面的测试，而我们本地化的自动化测试都没有做好，那么云测试真的就这么好用吗？

一、云应用

相信我们每个 IT 人或多或少对“云”有所了解，从 google 提出这个概念后，就一直被议论着，现在更是掀起了一股云热潮，现在的云应用也越来多，从这个方面来说：

1、 基础设施即服务（IAAS），一些厂商搭建服务器，应用分布式+虚拟，提供服务器的租借。而现在一些公司提供的网络 U 盘服务，我觉得也可以算是一种这样的服务，它提供的是一种硬盘存储资源，例如：苹果的 iCloud 和现在流行的酷盘。它通过这种方式实现了一种资源绑定，更好的将用户绑定在它的云端。

2、 平台即服务（PAAS），平台是向外提供服务的基础，这种方式将软件开发平台按需提供，不需要你为了开发某个软件功能去购买，你可以应用服务器上的软件开发环境。开发出所需应用之后，可通过服务器传递到用户。其实也可以说是一种另类的 SAAS。而我觉得云测试算是一种 PAAS。

3、 软件即服务（SAAS），这种方式，即某个中间商通过浏览器或者某种途径展示和推广软件应用，用户按需选择自己的软件运行即可，这样，用户可以不用为购买此软件付费；举个浅显的例子：某款收费的单机游戏，我们可以不用交费，通过远程在服务器上玩，当然，这还需要一个过程。

而最想说的是：360，360 的软件宝库，其通过了 360 安全卫士和安全桌面将这些软件展现在用户的面前，用户通过其下载试玩，其只是将软件保存在本地，而它绑定用户的方式是可以把 360 安全桌面安装在 **prayaya v3** 里面，然后在 prayaya v3 文件目录拷贝到 u 盘或者移动硬盘里面 ,这样以前的软件也可以保存，这也是一种另类的云端吧，但是这种方式总是比用户与资源绑定来得不够巧妙，当然，这都是自己的想法，只是说说而已，我能想的，大多数人都有自己更好的想法。

二、云测试应用

云测试可以称得上是 PAAS，因为其云测试就是利用云端的平台资源，包括硬件资源和软件资源。我对云测试的产品和应用收集分析了一下：

1、 利用云端的服务器的性能

一些公司进行性能测试时，需要利用大量的服务器资源，而为了一时的性能测试，实在是浪费成本，因此这些公司可通过网络将软件上传，按需租借服务器进行测试。

2、 利用云端的服务器中的测试软件进行测试

现在有一些公司应用服务器专门提供商业自动化测试软件或者自身开发的自动化测试系统，通过远程桌面连接到服务器内的虚拟机，直接打开虚拟机内的测试软件和 WEB 浏览器来测试本机或者公网上的 WEB 软件。测试您本机 WEB 软件或者客户端软件时可以下载 VPN 客户端。

3、 利用云端的服务器中的环境系统

利用云端的服务器环境系统，多用于兼容性或者国际化测试，例如：现在手机应用有一个难题就是需面临各种手机操作系统，而一些公司通过 web 网络和服务器上提供手机测试平台，测试用户可以通过浏览器选择相应的操作系统，然后上传自己的 APP 应用或者打开自己的手机 web 系统进行测试。

三、云测试的发展思考

1、 云测试，其实也是一种资源整合和动态分配的方式，这种方式保证了测试资源的最大化利用，将测试资源放在云端，而给我们提供一个水龙头去按需提取这些资源，这在一定程度上让我们的测试资源成本减少，但是却加深了测试的技术成本和人力成本，所以也要求测试人员会更快速的构建好的测试环境和执行更完善的测试用例。

2、 云测试，会推动自动化的测试，但是相反的，现在本地自动化测试都没有做到非常好，所以云测试方面的自动化测试软件的应用推广会比较难，而且现在商业软件都有试用版本，如果那些云测试软件商能够提供一种途径，即提供一种自动化测试系统或者解决方案，能够快速的帮助进行自动化测试；所以，自动化测试系统解决方案会随着云端的开展变得尤为重要。

3、 **云测试的发展，对我们测试人员来说是一种机遇，也是一种挑战，云端的测试系统将会变得复杂，虚拟技术、分布式技术、操作系统技术将会是一定热点，本地端将会变得简单和复杂并存，如何选择是我们的想法了。**

总之，以上只是我对现在云测试和发展状态的一些想法，不一定很对，**现在时代发展飞快，如何从海量数据中提取信息，也许也需要一种技巧吧**。共勉

基于模型驱动的自动化测试设计

基于模型驱动的自动化测试设计分析

序言：一直在思考一个问题：业界，自动化测试被大家认为应用的场景主要是产品稳定、周期较长的测试项目，而且自动化测试不是用来发现和定位问题的。大家应该知道，“杀虫剂效应”，虫子对农药产生抗体的，那产品何尝不会对反复不变的测试产品抗体呢，那么这样，自动化测试的效果只会越来越差，根本上只能去更改自动化测试脚本，让其变得更有适应性。这其实自然界的一种规律。而更好的方法，我觉得是只需要在自动化测试系统中添加相应的因子或者是模型，系统则会自动适应生成一系列的自动化测试用例，业界内流行的机器学习也是让程序变得有学习型，前两个月对自动化测试中的基于模型的技术进行了研究，还弄了个小专利，不过没有推广，因为其要求测试人员的一些编程功底，而且成熟度不够，但是却让我更相信证一个事实，那就是自动化测试是可以用来发现更多问题的。

（PS：序言好像写的有点长了...）

一、人工测试的场景

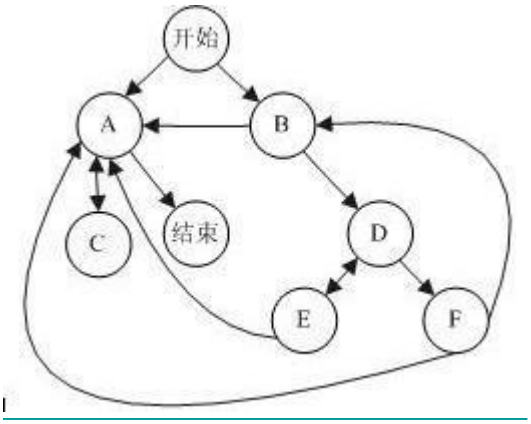
为什么人工往往比自动化测试更具有效率，分析一下：

1、 人工具有反应性，发现产品异常不是去继续按用例执行，而是根据异常结果进行不同的处理，因而能快速发现问题。

2、 人工具有变化性，能够自我学习，并且基于功能，快速发现新的测试用例，从而让测试变得更有充分性和适应性。

二、基于模型驱动的应用

所谓的 model，其实是一系列的状态机，一般都是有限状态机，是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型可以，可以将一个有限状态机看做是一个特殊的有向图（这个在算法导论中可以参照），它包括一些状态（节点）和连接这些状态的有向弧。每一个有限状态机都有一个启始状态和一个终止状态和若干中间状态。每一条弧上带有从一个状态进入下一个状态的条件。下图是我画的一个说明图：



从开始模型起，箭头表示上一个模型能够根据不同的条件触发进入下一个模型，经历若干个模型后，到达结束模型止。从而获得了一系列的遍历路径。这么一说，也许大家就能想到如何将其技术小用在测试中了吧。

三、基于模型驱动的自动化测试设计

在自动化测试中，每一个模型相当于一个测试场景，不同的测试场景之间的触发有不同的条件，一个整体的功能测试，比有开始和结束两个状态。因此，设计自动化测试系统可包含这几个测试模块：人机交互模块、总体控制模块、模型驱动模块、数据库交互模块、测试用例组装和分析模块、执行模块、测试结果分析验证模块。

重点是测试用例组装和分析模块，其可以根据不同的测试方式进行测试用例的组装，第一种按输入指定的测试序列直接进行模型组装生成用例，第二种测试方式是在测试过程中模型不断根据输出状态和触发条件进行组装和生成用例。

应用就是

1、随机序列，则无需人工去构造用例，而是根据测试模型，应用深度或者广度优先遍历算法，生成所有用例，例如：你从北京去上海，有几种途径选择，测试时，你只需定义好各个城市节点状态，则可自动生成从北京去往上海的路径，可以快速应用到实际测试中。这样，保证了测试的充分性，也节省了人工构造用例的时间。当然，最后生成的用例也需要人工审查保证。

2、指定序列，则可以按指定的序列去检验功能，例如：北京到上海，指定的路径是：从北京到南京，再到上海，主要是测试这条路径。

总结：当然，上述的很多系统很多工作还未完成，实践上也是颇为简易，而且适合的场景也很有限，所谓的数据驱动和关键字驱动已经很适用，而且驱动还有太多的思想还没有挖掘清晰，**我个人觉得：技术很重要，但是落地的应用更重要，技术是为了服务需求和实践的，所以有时候高明的技术在某个时候效率不一定比得上基础的技术应用。**

由单元测试看功能自动化测试

最近，在看单元测试框架和研究 Junit4 的源码，看单元测试框架的目的是为了研究单元测试的测试方法是否可以应用到功能自动化测试上 ,看 Junit 源码的目的是写了一段时间的 java 代码了，可惜现在总感觉无法有一个很好的上升，特别是在接口以及代码规范、模式上面没有很好提高，所以只能通过读源代码的方式提高了，之前读过 JDK 的源码，但因为不是一个纯粹的产品，所以从根本上也提高不大。

一、Junit 的框架分析

Junit 应用上而言，没有什么太大的难度，其思想确实值得借鉴：

1、 **测试执行模块：**Junit 的测试单位有：

测试集合(testsuite)：测试集合可以包含若干个测试用例，按顺序排列执行。

测试用例（ testcase ）:可以说一个测试类就是一个测试用例，每个测试用例，包括若干个测试方法。每个测试用例可以包括：BeforeClass 与 AfterClass，即用例测试前配置和测试后的清除。

测试方法（ testMethod ）：测试方法是具体的执行测试的函数。每个方法可以包括 before 与 after，即方法测试前的配置和清除。

分析：在做功能测试框架的时候，也可以采用这种方式，即测试集合+测试用例+测试功能封装。而且测试前配置和清除很重要，在功能测试可以是环境配置和环境恢复等。

2、**测试结果模块：**可以从测试集合与单个测试用例为入口执行测试，测试完成后，可以看到测试执行的结果和时间，即执行用例的个数、error 的个数和 fail 的个数，并且能够快速跳转到错误方法的位置。Error 和 fail 的区别在于 error 是被测试程序的异常。Fail 是验证被测试程序的期望结果的失败。

分析：自动化测试中，结果部分是非常重要的，最好能够有结果统计以及具体结果的链接功能，一定要区分出 error 和 fail，帮助快速定位问题。

3、**测试异常模块：**Junit 可以使用 expected=Exception.class 来期待一个预期的异常，即监测输入异常数据，是否能使被测试方法抛出异常。而且我测试了下，当被测试方法中有异常的时候，当前测试方法跳出，但是不影响下面的当前测试方法的测试恢复方法和下面的其他的测试方法。

分析：在功能框架，测试的稳定性很重要，测试框架的一个作用就是保证一个测试用例执行错误后不会影响到后面的测试用例的执行。测试执行的稳定性是自动化测试的基础。

4、参数化测试模块：@Parameters 指定一个集合存储测试数据列表，主要包括期望值与实际值，然后测试方法遍历列表，这种方式适用于测试方法一样，但参数不一样的形式，理论上是一种数据驱动的思想。

二、让脚本具有“可信赖性”

在单元测试中，有一个很重要的技术是 mock 与 Stub 技术，他们的原理都是去替代那些被测试代码所依赖的，但确实不可信赖的东西，例如：你写的一行代码中需要与数据库或者配置文件连接，你是否能保证数据库的连接类与外部资源都没有问题。或者你是否能保证引入的其他的 class 是没有问题的。如果不能保证的话，这些都是属于测试代码不可信赖。

所以，有一句话说的很好：一个好的测试必须是“值得信赖“的测试，而值得信赖包含两层意思：

1、 当它通过时，我们有信心说被测试代码一定正常工作。

2、 当它失败时，它一定证明被测试代码是错误的。

在功能自动化测试中，我对照“可信赖性”的原则，进行了反思：

1、外部环境的影响；由于我的自动化测试不仅与软件环境有关，还与硬件设备测试环境有关，所以，这些环境是否能够保证是值得信赖的，即不会影响脚本的运行状况。

2、 封装的应用；我们包括命令行封装和关键字功能封装，命令行封装的好处是防止命令行接口变化导致脚本失败。业务脚本是由关键字功能封装组合而成，是所有脚本公用的，因此保证一定的正确性。**其实，从另一个角度说，这也是让自动化测试变得更可控制性。**

3、 环境参数与业务逻辑脚本的分离；最近帮软件开发部门搭建自动化测试环境的过程中，遇到的最大一个难题是脚本的移植性问题，很多脚本由于以前涂一时方便，将环境参数写在了脚本里，所以移植的过程，由于拓扑环境差异造成的错误大大增加。

其实，总的来说，我觉得用一句话概括：拥抱变化，让自动化测试变得更可控和更信赖。

总结：个人浅见，知识是相通的，存在即合理，互相借鉴，也许会收到不小的启发。作为一个技术人员，不能局限与技术，如果能够在精于技术的基础上再学习点别的知识，那么技术创造性也许才会提高一些。共勉之~

STAF 的应用总结和分析

自动化测试框架的应用分析系列

之 STAF 的应用总结和分析

序言：我想主要从实际原理和应用上来说说测试框架，这些框架包括：关键字测试框架 robot，基于各种语言的分布式 STAF 框架，集成测试框架 Fit（husdon），以及 elipse TPTPd 性能测试框架等。这不是一套工具教程，而是一套应用的简单思想，个人难免有局限性，见谅。

一、STAF 介绍

STAF 全名叫 Software Testing Automation Framework,即软件自动化测试框架，我觉得，STAF 可以完全称得为一个测试框架，其基于软件开发中“系统+组件”的思想，系统是一个架子，各个组件都可以插入到架子中成为其功能一部分。即，STAF 是一个这么一个架子，提供了组件插入的规则，不管你的组件是用什么语言写的，只要符合 STAF 的组件规范，就都能做为 STAF 的组件服务与其他的组件服务进行通信，即其提供了轻量级分发机制，负责将请求转发给服务，从而调用这些服务的功能。

二、staf 原理

staf 的基本运作原理如下：

1、启动 STAF 时，STAF 做为系统的一个守护进程开启，然后同时加载其架构中的服务，这些服务可以是 DLL 文件、JAR 文件。当你设计一个 STAF 的外部服务的时候，即继承 STAF 提供你的接口，你只需要撰写一个类实现接口的方法即可。也可以这么理解，STAF 就像一个操作系统，而这些服务就像这些操作系统上的应用功能软件，每个软件之间可以通过内存或者管道或者网络通道进行互相通信。所以，你若是设计一个测试框架时，可以参考 STAF 或者操作系统的这种理念，定义一套规范接口，以便于框架的灵活拓展和实现，而不是将一套框架写的死死的。

2、STAF 在加载服务的时候，为每个服务分配了一个句柄，即 STAFHandle，这个句柄即作为服务的唯一标识。类比于操作系统会给每一个进程一个进程 ID，你打开系统的任务管理器，能看到很多进程在运作，而你可以通过每个进程的 ID 对这些进程进行操作。

3、STAF 之间服务的交互则可以通过命令行的形式或者向服务的队列读写的形式进行。命令行可以看做为每个服务向外公开的消息接口。

三、staf 的应用

staf 的缺陷在于掌握困难，你需要很清楚它的机制以及其提供的一些服务的作用后，你才能真正的把他用好

1、分布式多执行端应用：STAF 采用 P2P 架构，即没有服务器和客户端之分，任何安装了 STAF 的机器之间可以互相通信，可以利用 STAF 的这种特性实现分布式执行的功能，例如：你有 10 个测试用例，一般都会将 10 个测试仪用例在一台执行机器上串行运行，脚本少的话当然不会有什么问题，但是当脚本数量庞大到一定程度时，那么你需要一个分配机制，能够将这些脚本分配到不同的空闲执行端运行，那么 STAF 可以帮助你实现这些机制，可以指定一台 STAF 机器为脚本任务分配端，然后指定一些 STAF 机器为执行端，那么任务分配端收到任务，即一系列测试脚本后，它可以寻找到空闲执行端，应用 STAF 之间的通信，将这些测试脚本分配下去，在每一台机器上执行，然后返回结果。这样，就可以大大节约一些测试时间了。当然，你也可以自己开发属于自己的网络分配系统，可以采用 socket 机制实现。

2、STAF 几个比较常用的的内外服务：

文件系统服务：即 FS 服务，你可以采用 FS 服务进行不同机器之间或者一台机器上的文件之间的传递。例如：一些测试脚本和测试结果、日志等都可以采用这种方式传递。

时间驱动服务：即调用此服务来按特定的时间间隔发送 STAF 命令，从而调用别的 STAF 功能服务，这其实相当于在自动化测试中，每隔多少时间，进行一次测试，这种概念跟持续集成的按时间间隔进行 build 有些相似。

事件驱动服务：即由发生的事件来驱动进行通信，从而执行 STAF 命令，事件是通过队列形式发送，每个服务都有自己的一个队列，服务通过接收队列中的请求从而进行功能操作；这种概念跟持续集成的按事件驱动进行 build 有些相似。

邮件服务：即发送邮件的服务，当测试完成或者失败时可以触发这个服务发送邮件。

日志服务：即可以将一些测试结果或者日志信息按照表格序列的形式存在指定的 STAF 机器上，然后可以随时进行读取。这样可以当一台 STAF 机器的测试执行完毕后，将测试结果存在本机或者专门存放日志的 STAF 机器上，然后随时可以读取显示。

压缩服务：调用 STAF 命令对文件进行压缩。

名字空间服务：这个服务挺重要的，就是可以将一些数据存在 STAF 的指定内存中，然后独特的名字对应着独特的数据，这个与哈希表有些类似，你可以随时取得这些数据，这些数据在后台都是保存在本地 XML 文件中的。

当然，这些服务只是一些基本服务，为了拓展你自己定制性的测试框架或者平台，你可以很好的利用这些基本功能。

3、**STAX**，很多人都不清楚 STAX 和 STAF 的关系，其实从本质上而言，STAX 和 STAF 本身没有太大关系，即 STAX 只是调用了 STAF 的内外部服务来构造了一个测试执行引擎。STAX 整体机制是：你通过在按照 STAX 的 XML 格式定义测试工作流，可在其中嵌套要执行的测试用例，然后由 STAX 导入，然后执行 STAF 命令，调用相关的执行服务执行测试，STAX 能够监测服务的状态，并且读取指定机器的日志服务并显示。**不过我认为：STAX 在执行操作上太过于繁琐，不用也罢。你可以自己设计控制界面，来下发脚本，并且可以读取日志服务或者可以获得 STAF 的一个句柄，从而可以接收 STAF 发送过来的回应，将结果和执行状态显示在界面上。**

4、总而言之，STAF 只是一个请求消息的分发机制，而那些基于 STAF 规则的服务才是实现整个测试的重点。你需要拓建多大的测试框架，那么你就得找到更多的需求，将其转化为服务，并且定义好他们之间的请求消息，让它们很好的协同合作。

总结：

1、一个好的测试工具或者框架，简单的使用并不难，如果真想在自动化测试领域得到进一步的发展，那么学习这些框架的思想，因为这些框架都是前人在实践基础上构建出来的，学习其思想，不仅能掌握自动化测试的理念，更能在软件思想上更进一步。

2、 想当时学习 STAF 框架时，实践很少，对于自动化测试框架一些认识都限于理论，所以学习了很长时间，但是在一段时间后，再去学习 robot 框架以及别的框架，都很快的从里到外认识，所以，学习很多工具不如先把一个学通。

Robot famework 的应用分析

自动化测试框架的应用分析系列

之 robot framework 的应用分析

序言：很多人都对自动化测试框架痴迷，我曾经也痴迷过一段时间，以前觉得自己对框架说的头头是道，现在回过头来看以前，说归说，但在如何应用还是

欠缺，这一段时间，自己经历了一系列框架的构建和应用的时期，所以，我想主要从实际原理和应用上来说说测试框架，这些框架包括：关键字测试框架 robot，基于各种语言的 STAF 框架，集成测试框架 Fit（husdon），以及 elipse TPTPd 性能测试框架等。这不是一套工具教程，而是一套应用的简单思想，个人难免有局限性，见谅。

首先，看看 robot framework 吧

一、robot 介绍

大家上网查 robot，相信能查到这么几个资料;

1、robot 的快速入门，就是如何简单使用 robot，这是翻译过来的教程。个人觉得，一般人拿到这个教程容易懵，因为你是在不了解其原理的基础上操作，所以会遇到很多问题而无法向下进行。

2、 robot 的源码分析，讲讲 robot 怎么一步一步工作的，现在 robot 框架不断升级，这个源码分析教程有点旧了，所以代码有些不一样。

Robot 是一个完全基于关键字测试驱动的框架，它即能够基于它的一定规则，导入你的测试库（例如：其集成了 selenium 的测试库，即可以理解为操作 web 控件的测试底层库），然后基于这些测试库，你能应用 HTML、TXT 等文档形式编写自己的关键字（这些关键字即你的库组成），之后，再编写测试用例（测试用例由测试关键字组成）进行测试。例如：一个简单的登陆测试由：登陆+输入密码+登出三个关键字组成，也可以由一个关键字登陆组成，关键字颗粒的大小可以自行定制。

二、robot 原理

robot 的基本运作流程如下：

1、robot 开始测试是从 cmd 命令输入，robot 初始运行程序接收命令字符（主要是用 TXT 或者 HTML 写的测试用例集）

2、接收之后，robot 先生成初始化全局变量配置，主要是定义一系列的字段名称和文件名称（例如：日志文件名称等）（由 settings.py 中的类完成）

3、之后，开始解析用例文件,生成数据对象，数据对象中包含了测试集的各种数据，例如：测试用例集名称、各个测试用例名称、各个关键字名称等。

4、将测试数据对象传送给测试集合类处理，生成测试集 suite 对象.

5、之后，进行 suite 中的用例测试，然后调用关键字，找到关键字对应的库文件，进行操作。

6、每一个操作和结果都写在输出的 xml 文件中（有专门调用对输出 xml 进行操作的类）
7、测试完成后，调用转换类将 xml 文件转换成相应的 HTML 日志报告，测试完成。
三、robot 的应用
robot 的缺陷在于不够灵活，有些地方对我们普遍不适用，所以，**我们需要会将 robot 分解，取之我们能用的**，以下提供几个分解方式
1、robot 的需要用命令行的方式去启动测试，我们可以更改为自行写一个测试执行客户端界面，选取相应测试用例集合，下发到 robot 框架的程序入口。
2、robot 的用例格式需要固定，而我们很多时候都有自己的自动化测试用例编写规则，这样的话，我们可以写一个转换器，可以将自己的测试用例解析文件解析为 robot 可识别的用例文件。
3、第 2 个的另外一种解决方法就是：做一个数据解析器，将我们的测试用例生成传送给测试集合的数据对象的对象规范
4、Robot 的日志文档很好，如果我们有自己的关键字测试框架，但需要调用 robot 的日志 api，则可以：1）因为 robot 是解析 xml 文件生成 Html 的，我们可以调用 python 的 xml 操作库生成符合 robot 的 xml 格式的输出文件，然后调用 robot 的转换库生成 Html 即可。2）我们可以直接重构 robot 的 xml 输出类，简化成我们的形式即可。
而在实际脚本的应用好处就是
1、关键字思想驱动的好处就是应用封装的思想，保证了上层脚本的变动性可控，增强了维护性。
2、容易定位问题，一般我们定位测试问题，首先，是从用例—功能—单个操作。而线性脚本则是相反的，所以会不好定位。
总结：
1、很多框架不一定能拿来就用，我们要做的是分析源码，学会拆分框架，取之自己能用的，开源的框架一般代码结构都挺良好，接口定义挺清楚。**尽可能的不要造车的同时还要自己造轮子。**
2、从 robot 上面，我们可以很好的学习到关键字测试驱动思想和数据测试驱动思想的应用。

自动化测试开源策略

序言：不知道有多少人对开源社区真的很有了解，个人以为在自动化测试中，开源也是一个很好的利器，往往商业性的工具针对普遍人群，而自动化测试是“定制型”的，不一定特别适合，而且自动化测试是预言型的，所以从长远或者灵活性上而言，可以考虑开源。我个人觉得：自动化测试在追求发展过程中，要学会借助各种工具提高效率，而不是仅仅局限于一种。还是那句话，能提高测试效率和工作效率的才是王道，“摘花折草即可伤人也”。

一、自动化测试中的开源软件分类

开源工具因为其零许可费以及开放和自由的理念逐渐得到了大家的认可和广泛的传播，而且由于自动化测试的差异性，其开源软件的灵活性更能在自动化测试中很好的体现，而且随着开源软件和自动化测试的发展，其开源工具在自动化测试中也形成了一股应用的趋势。其实，在工作中，我们都在不断的与自动化测试打着交道。

在自动化测试过程中，我们与之打交道的开源工具，可以分为

1、 编程语言与平台，即在自动化测试过程中应用的语言和操作系统。

1）Andriod ,大家应该都有所了解 ,其是以 Liunx 为内核底层来支持不同硬件，并在其上搭建一个类 java 的运行环境，其大概有几层，包括：linux 内核、底层库、JAVA 框架（包括其 API）、Andriod 应用程序。

2） LINUX，大家熟知的开源操作系统。

3）脚本语言：Python、ruby、perl 等，这些都是在自动化测试中因为其简便性与动态性多有用到的编程语言。这些语言的维护、开发和发展都是通过开源社区和开源标准组织(例如 ISO 和 Ecma)进行的，所以它们称为开源语言。而 java 因为受 Oracle 支配，所以称得上开源不开源，我也不是很清楚…

4） Flex:是在 FLASH 基础上做的一层封装，提供了组件库，开发人员可以直接编写 MXML ,即在 FLEX 中布局用户界面组件的一种 XML 语言来搭建用户界面。同时，完成负责数据逻辑的 ActionScript 脚本，最好编程成 FLASH 文件。所以，有些用户界面是用 FLEX 开发的。

……

2、 开源开发工具，

1）Eclipse，这个用过 java 的一般都很熟悉吧，是一款很好的 IDE。其是基于”OSGi”的“即插即用”理念，所有功能以组件形式存在。其理念我觉得非常好，其插件只要遵循其平台的规范，就能集成到其中应用。例如：Pydev 是一款 python 的插件，jython 是一款 Java 与 python 集成的插件，还有 andriod、an

t 等集成的插件，当然，我觉得可以的话，最好先应用一下独立版，再去在 eclipse 中应用，这样，可以更好的了解其运作原理。
而且，我觉得这种理念在自动化测试中也可以很好的应用，使得各个工具之间能够在一个平台上作为模块互相通用，而且也能独自使用。其 IBM rational 开发的 jazz 平台也是基于一种这么理念的。
2） Ant,这个大家也许不是很熟悉，但是开发过 java 应用程序或者做个持续集成的也有有所了解，它就是一款构建的工具，即用 XML 描述任务的形式，自动完成其定义的工作，例如：可以帮助开发人员自动完成编译、单元测试、打包、发布等工作。
3） Maven，Java 开源项目的开发管理工具，涵盖了项目构建、文档管理、报告生成等方面，与 Ant 功能类似，其差别在于 ant 每一个项目需要独立维护一个 XML 构建描述文件，而 Maven 能够帮助快速搭建一个项目框架，而无需从头编写，其是一种“约定胜于配置”的理念，即先抽象出一个原型。这理念也可应用在自动化测试中的，即先提供一个脚本模板，然后根据这个模板，搭建一定的测试流程。
4） 版本管理工具，例如：SVN 和 CVS，其都能够应用脚本控制其代码版本的签入和签出，在其自动化测试中也能有一定应用，方便管理脚本与代码程序。
5） Bugzilla，缺陷管理工具，可以管理和跟踪缺陷，即，可以在自动化测试中应用来管理相应结果或者缺陷跟踪等。
6） Junit，大家都恨熟知的吧，单元测试的一款工具，即事先规定好单元测试模板，开发人员只需去根据被测试代码，搭建其测试代码即可。
7） TestNG,与 junit 类似。在自动化测试中也能有所应用。
3、 编程及测试框架与库
1） 在 J2EE 开发中，大家熟知的 SSH，即 Spring、Struts、Hibernate。具体的大家可以去查阅相关资料，我想说的是，如果大家深入学习的话，会发现，其开发理念和自动化测试思想很相似，像 Struts 的 MVC 思想，与自动化测试的分层理念可以很好的结合。Hibernate 的数据库持久层思想也可以用于自动化测试的数据管理应用，总之，了解这些软件设计框架，对于加强自动化测试思想的理解很有帮助。
2） Selenium，大家都很清楚的 web 自动化测试框架，很多人都说这是一种工具，其实说工具也行，框架也一样，其提供了一种测试 web 的自动化思想，即采用绕过 web 中“同源策略”的方法，用 JS 来控制 web 的操作。你可以编写

脚本应用其 API，来控制 web 的相应控件的操作。一般是集成在你的自动化测试管理框架或者系统平台中的。

3）Robutium，andriod UI 测试的一个自动化测试框架，理念类似，只是应用场合不一样。

4）Abbot，测试 java UI 的一个自动化测试框架，其录制的测试用例是用 XML 进行描述，其只能用录制的方式生成 XML 文件，而且其 abbot 只能去读取 XML，所以你可以自己写一个脚本库去生成相应的 XML 去控制 aboot,其在测试 java UI 方面的稳定性还是不错的。

5）自动化测试管理框架或者平台，STAT 与 Robot FrameWORK,前者是提供一个分布式平台且可以将一些功能作为其服务插入到相应平台中。后者重点在于测试用例的管理上。

当然，还有各种各样的编程框架与自动化测试框架，但是随着接触的多了，你会发现其理念都是一样的，所以要学会自己从各个工具中提炼其思想与共性。

4、 应用服务器软件

1）web 服务器，举一个例子，大家熟知的 Tomcat，其中也继承了 J2EE 中的 servlet，其 web 服务器的作用主要是提供 HTTP 协议操作，将 web 客户端提交的页面请求进行处理后，然后动态返回相应的 HTML 页面即可。

2）数据库，MySQL,开源的关系数据库系统，在一般的中小型项目还是很好用的，数据库设计在自动化测试中，个人认为也很重要，如果将自动化测试设计成一个平台的话，需要涉及大量的测试用例与脚本、测试用户权限的管理等。所以，数据库设计需要在自动化测试平台设计之前，定义好表以及表之间的联系，方便以后拓展使用。

二、如何去保证开源工具的应用

1、在自动化测试开展过程中，首先要对其测试需求以及对自动化测试的开展程度进行分析，包括自动化测试的规模、自动化测试的紧急程度以及实际需要应用程度、自动化测试的成本考虑等因素。

2、 之后，就去根据相应的需求，在不同方面采用不同的测试软件或者工具，不需要局限和死专于一种，哪种能提高效率，就尽快采用。

3、 总之，在这些软件或者工具的基础上，如果要规模化的话，你需要有一个自己定义好的平台进行规范，各个工具软件框架都可以以模块化的形式存在，当然，我建议最好要慎重考虑其“高内聚、低耦合”的思想。

三、开源工具的应用策略

我大概想了一下，其在自动化测试应用中，这些工具都扮演着不同的角色，对推动这个测试，甚至说软件行业都起到很大作用。

1） 常规的开发和测试流程

当然这个自动化测试不会起到主要性的作用，但是能提高一定的效率。

2） 持续集成的流程

需要搭配单元测试框架、构建工具、以及持续集成管理工具（例如：cruisecontrol）

3） 敏捷开发与测试流程

敏捷开发中我觉得自动化测试是很重要的一个角色，其能够快速保证其发布周期。

4） 云端测试流程

现在出来的云端提交测试，需要自动化形式提交以及进行相应的处理，其都是在 web 上面进行提交与返回的。

总之，“预先善其事，必先利其器”，但是，在众多的自动化测试软件工具和框架中，我们要保持一个清醒的思想，要能够去抓到本质，真正能为我所用，就像武侠小说里面似的，侠客之路，从手中有剑到手中无剑、从有招到无招，从无心到有心。共勉之。

自动化测试环境拓扑管理

简易自动化测试设计 之（四）

——自动化测试环境拓扑管理设计

序言：不知道大家有没有对自动化测试环境拓扑管理方面有所涉及，但是这往往是自动化测试规模化一个难点，这也是作为自动化测试平台中的一个模块（服务），以下的解决方案更多的是不成熟，分享出来，希望大家能够帮助开动自己的想法，提出见解，当然，其实这些东西不一定受限于你的技术认知，相反，脱离了技术的束缚，你才能从更高的角度提出问题和建议，所以，希望大家提出应用的需求，帮助我开拓眼界，大家也许也能够提高思考能力。

一、 自动化测试环境拓扑管理简介

在电信项目中，自动化测试环境拓扑的管理往往是一个难点，刚开始进行自动化测试项目时，因为用例过少，所以环境信息一般都是硬编码在脚本中，后来随着项目的增多，就采用了脚本分层的方法，将经常变动的业务配置提了出来，写到了一个单独的脚本中，随着拓扑的越来越大，各种产品线都有各自的测试

拓扑，而对于其都是测试人员去进行管理的，缺乏了一种部门级的统一管理这些拓扑的机制，并且对于单个项目来说，其项目的拓扑并没有抽象出来成一个连接关系，因而不方便后期的项目拓扑的维护和管理。

而在软件测试中，其实原理也是一样，软件测试中有一个概念称为：自动化部署，这是持续集成中的概念，个人觉得，当然也适用于各个自动化测试，持续集成是脱离不了自动化测试的，部署是指将软件移动到它被测试的地方，或用户指定的某个位置，准备送给客户。最早前的部署都是手工过程，部署工程师从某处拿到部署文件，再把它放到目标机器上，然后开始正式的安装过程。然而，这种手工过程会比较慢，而且部署失败率也可能要高一些。自动化部署的目标是“持续部署”，即自动部署到生产环境中而无需手工干预：得到一个版本后，自动部署到一系列的测试环境中。经过整个构建管道中的所有阶段，并且能通过所有的测试后，自动部署该版本到生产环境中，而且具有自动回滚和相应的监控手段。

与上述自动化部署类似，自动化测试环境拓扑管理也能够自动根据项目综合拓扑中的环境参数与设备信息自动的生成脚本所需的环境配置文件，脚本运行时会读取这些环境参数和设备信息调用相应设备进行测试。

二、 自动化测试环境拓扑管理挑战

1、 如何保证统一测试拓扑。

2、 如何保证映射更新条件。

3、 检测版本，进行版本的自动检测和升级。

4、如何与设备进行联系，即拓扑中的节点如何与实际设备的控制信息关联起来。

三、 自动化测试环境拓扑方案

1、采用拓扑映射机制，即将项目的测试拓扑抽象出来，不具体表明端口，只是表明一种连接关系，然后采用映射匹配算法（匹配的条件可以根据自己需要进行设定，可以按设备的连接关系、设备型号、测试版本型号等进行匹配）。将标准拓扑中所需要用到的连接关系在综合拓扑中找到，然后生成一个设备匹配表。

2、匹配到相应拓扑之后，则测试项目可以调用其拓扑关联生成一系列的配置信息。

3、之前，一直思考如何将拓扑管理与设备关联起来，后来思考，即，采用软件工程中的思想，将一款共性的产品线作为一个高度抽象的类，然后下面派生

成一系列的具体的产品类。产品线类拥有共性的属性（例如：设备的端口号等可以作为一个共性属性，还有一些共有的命令行），然后产品派生类拥有自己特性属性。当然，还有一个共享的基本类（里面包含拓扑中每个产品线共有的属性，例如，IP 地址、串口号等）。

4、拓扑中的产品节点可以依靠产品线类型与具体产品型号关联实际环境拓扑中的一款设备，当测试项目调用这款设备时，测试项目就可以找到相应的配置文件。

5、配置文件相当于一个中间件，需要在每次映射时进行自动的更新。

6、拓扑的表现形式可以用 XML 节点的形式或者 UML 图的形式进行表示，个人觉得，图的表现形式是个发展。

7、软件自动化测试中的环境部署也可以应用此关联的方式，不过还是需要具体情况具体分析。

四、 相关软件产品分析

对一些相关的产品进行了使用研究：

1、 1、FanFare 的 iTEST,现在被 spirent 收购，在其拓扑关联方面，提供了一个 Topology 功能。其应用 Port 连接来表示设备的互连关系，每个设备对应一个或者多个 session，可以直接在拓扑图里，根据这个 session,管理到设备，但是并没有与 case 关联起来，所以其主要功能在于拓扑的直观性。

2、 2、IBM 的 RSA 与 RTC 产品，其理念是模型驱动理念，用过 UML 的人应该知道，其将测试拓扑中的设备抽象成一个个模型，然后每个模型关联一个脚本模块，具体情况还没有过于去研究，有兴趣的人可以了解参考一下。还有一款软件 Insight，是可以应用在智能数据分析，从而选取最佳的测试优先级，这个一直觉得好复杂，对数据存储和挖掘分析要求很高。

3、 3、以上只是自己的一点简单总结，因为是很匆忙，所以不一定正确，需要自己去进行使用分析。

4、4、个人觉得，以上工具，因为需要涉及到应用普遍性，再加上做自动化测试的特殊性，所以也许你能用到的少之又少，但是你可以根据自己需求，结合其工具的设计来得出适合自己的自动化测试方案。适合自己的才是最好的。

总之，自动化测试拓扑解决方案很难有一个完美点的方案，所以如果对这方面有兴趣或者有研究的人，可以一起讨论下，提出一下想法，感激不尽。

基于 STAF 的分布式架构模块设计

序言：这里将简单说说基于 STAF 设计的分布式模块，当然，首先需要明白什么是 STAF，那就以 STAF 为主，讲讲其分布式一些简单的设计方法，当然，不一定适合各个情况，只做参考而已。

一、分布式架构设计

基于 STAF，其主要是设计了三个部分

1、 控制客户端，其主要是设计了一个界面，应用来控制测试用例的点选、测试用例脚本的打开和编辑、测试统一结果的查看以及测试请求的下发（即与 STAF 的交互）

2、 执行客户端，其主要是作为 STAF 的一个外部服务设计的，即执行端的 STAF 接收到控制端发送来的请求信息，进行执行端上的测试执行，并且将测试统一结果返回到控制端。而在这里，我设计的不是直接返回到控制端，而是通过中间件的形式，这个中间件可以是 EXCEL、CSV 表格，或者数据库表格，测试用例框架将测试结果信息写入到其中间件中，然后测试执行端解析这些中间件状态，传递到 STAF 的日志服务，控制端可随时读取到其结果值，测试完毕后，给控制端一个状态值，控制端收到状态值后，将执行端的具体测试日志通过 STAF 的文件服务传送到控制端即可。

3、 测试服务器端，保存测试用例脚本以及测试之后的日志，可以通过控制 SVN 进行随时更新或者 STAF 的 FS、FTP 服务都可。

二、STAF 的应用说明

STAF(software test automation framework ,IBM 开源的自动化测试框架)，IBM 软件设计的理念是很强大的，他的 rational，现在是基 jazz 协议平台的，即遵循这个软件交付平台的协议工具，都可以互相通信使用，使其灵活性大大增加。而 STAF 也是遵从与这一理念，所谓 STAF，按照我的理解，就是一个提供各种通信和服务机制的平台，遵循这个规则的服务都能在这个平台上互相通信和使用。我花费了很长一段时间才真正理解到这种方式的强大性，其实我觉得做自动化测试就是需要这么一个理念，一个可插拔的框架，任何机制模块都是一个基于框架的组件。

STAF 应用的几个部分：

1）STAF 分布式架构：STAF 的分布式架构提供了分布式功能，每一个 STAF 其实是一个分布式组件，其间可以互相通信，传递请求信息。然后，其提供了一系列组件模块服务插拔的机制。

2）STAF 的服务：STAF 的服务包括内部和外部服务，通过这些服务，可以完成一系列自动化测试中的基本功能，例如:日志的服务、时间驱动服务、资源池服务、文件传递服务等，依靠这些服务，可以进行任意服务的组建工作，构建出自己合适的自动化测试体系。
3）STAF 的定制服务：STAF 的定制服务，就是其应用 java 或者 c++设计一个遵循 STAF 服务规则的功能模块 ,这样 ,就可以将此功能模块插入 STAF 架构中，与其他的 STAF 服务进行通信。
4）STAX：STAF 的执行引擎，其也是作为 STAF 的一个外部服务，执行方式是读取 XML 文件撰写的测试用例来进行控制 STAF 的执行，如果对这种撰写用例的方式不是很了解的话，也可以不用。当然，脚本的执行命令和环境变量在各个平台是不同的，具有很大的差异性。为避免将这种差异硬编码到自动化脚本中，并使自动化脚本具有很好的透明性，采用基于 XML 的配置文件可以覆盖多种平台的差异性。
后面有机会的话具体说说基于 STAF 的设计方法。
三、B/S 架构的分布式设计想法
想过用 JSP 设计一个 B/S 架构的分布式模块
1、 控制客户端；都是基于 web，保证了其安装的繁琐性，测试人员用起来比较方便，可以在 web 界面添加自动化测试用例驱动表。
2、 执行客户端；执行端也就是后台，即是解析测试驱动模块，可以解析 web 控制客户端的执行用例驱动表 将统一结果信息和具体的日志信息写在 web 上。
3、 服务器端，通过 FTP 方式，进行测试用例脚本的调用。
总之 ,基于整个 web 构建的自动化测试框架的话 ,个人觉得确实能够方面很多，但是要基于此从头做起的话，因为业界有了很多这方面的自动化测试框架，例如：STAF、Robot Framework，这样的话，确实有点得不偿失了。不过对于那些没有分布式功能的框架，可以考虑一下。
这段时间，发现自己差点迷恋于框架和平台的设计工作，而离真正的自动化测试应用越来越远，有时候发现，想的太远反而不是件好事，呵呵，这才是真的得不偿失，也是一种浮躁的表现吧，共勉

自动化测试—工具、框架、平台……

序言：发现自己做自动化测试很难受，因为每天都在打击自己，每天都在更新自己的想法，每天都要思考着自己需要做的事情，有时候，虽然有了工具、虽

然有了框架、虽然有了平台，但是却发现其能够满足的需求性很少，只能重新回归测试细节，有时候，决定成败的却是那些繁琐之事。到底有没有一条明路呢。

一、从公司发展看工具、框架、平台

这里先不说自动化测试、突然记得以前的一个想法，那些大型公司的“大楼”是如何能够这么大的屹立在这片土地上。以前的腾讯 QQ、以前的百度搜索又是如何成长成现在这个样子的，难道他们一路上就是顺利的吗？答案肯定是否定，没有谁的道路不坎坷，相反，当要成长成巨头时，其面对的压力和挑战反而更大。

也许，我可以这样理解，腾讯 QQ 和百度的搜索引擎，刚出来的时候，其面对的是整个不知道这个东西的用户，对于用户来说，这些东西刚开始只是“工具”，而当腾讯 QQ 和百度搜索根据客户需求，结合一系列的逼不得已的需求考虑，而将其改善成了一个“框架”，里面集成了一些围绕着 QQ 聊天或者百度搜索的应用，能够更好的为这个工具服务，在这个阶段中，其也是艰难的，因为用户对这款工具很陌生，你只能从他的平时表现中挖掘需求，而到了后来，当这个框架慢慢被用户接受后，那么用户开始乐于使用这个“框架”了，但是，框架虽有，隐患还是很大，因为公司的生存太过于依赖这个“工具”了，那么接下来，他们从“价值回报”考虑，肯定就要做更多的事情了，那么“平台”的诞生也就理所当然了，像腾讯 QQ 和百度都已经不能说是纯粹的聊天和搜索定位了，他们已经慢慢拓建自己的“多元化平台”，加入一系列能够改善用户体验的活动，开始引导用户的“需求”了。“平台”之后，将会是无限期的平台……而很多公司，要么太过于依赖自己的“工具”，而忽略了用户的“需求”，自己搞自己的。要么发展到“框架”之后，虽满足了需求，但也只能被用户引导，很容易沦丧，而达到“平台”之后，就能开始引导“用户”的需求，去创造用户的体验价值了。

二、从自动化测试看工具、框架、平台

自动化测试与上述过程有点类似，虽然前景很美好，但是道路很艰难。

刚开始开展自动化测试的时候，测试人员就是“用户”，他们对自动化测试没有概念，而且刚开始一般都基于“自动化测试工具”来开展自动化测试的，在后面的发展过程中，测试人员发现测试工具不能满足需求时，就想着做一个所谓的框架了，框架出来之后，确实改进了很大部分的测试需求，但是却还是得

基于着“自动化测试工具”，所以，从后期发展考虑，“自动化测试框架”也会将成为“自动化测试平台”的一种，最后打造出一个插件式的可拓展的平台。当然，愿望越美好，阻碍越大，只能说是从整体上把握整个过程，在细节上进行调控，慢慢的进行挪动，过于依赖或者停滞不前总是不可取的。

三、简易自动化测试设计

一些简易自动化测试设计，慢慢的将这些设计进行整理，慢慢的将其进行整合，最终看是否能打造成一个平台，也许若干年后，这些设计还在拓展，也许，将会消逝，但是只是希望无论拓展也好、消逝也好，总会留下那么点东西就足够了。

1、 自动化测试工具架构模块
2、 自动化测试执行驱动模块（也可以说是 agent 端）
3、 自动化测试控制端模块（包括界面控制端）
4、 自动化测试分布式模块(其是最主要的模块，它可以将所有的模块联系起来)
5、 自动化测试服务器模块
6、 邮件服务模块
7、 ……

每一个模块可以说是一个服务，以一个主模块为主，有一个主要的机架，然后各个小型模块。围绕这个机架，就最终形成了一个“简易自动化测试平台”

自动化测试执行驱动模块设计

简易自动化测试设计之二

自动化测试执行驱动模块设计

序言：本次想说说自己设计的自动化测试简易驱动机制，这里包括前台与后台的两种驱动机制，前台是用 java 写的，后台是用脚本语言写的，这里都是属于自动化测试的模块系列，像之前所说的基于 RFT 或者别的工具的层次也是属于一个模块，其是属于在工具接入模块的，后面会说的。而它们围绕着有一个能够调用这些模块的机制，逐渐基于这个机制扩大，慢慢的，就能从一个轻量级的自动化测试框架到一个重量级的自动化测试框架，然后再加入分布式、物理测试环境及资源的调用方式紧密结合起来，构建成一个自动化测试平台，即 LAB。

废话又说多了…还是看看执行驱动模块设计吧。

一、自动化测试驱动模块包含部分

首先声明，这里最重要的是一些中间件：CSV 表或者 MySQL 数据库，他们的作用是作为一个中间方式，连接各个模块的信息，保证信息的传递。
执行驱动模块需要有一个配置文件读取机制，可以读入到数据表与用例驱动表中放置的路径。
这里的自动化测试执行驱动模块包括：
1）自动化测试用例驱动机制。
前台：我应用 java 设计了一个读取和写入 CSV 表与 MySQL 数据库的机制，这里有四个类，CSV 读写类与数据库读写类，这个驱动机制就是解析这个 CSV 表或者数据库表中的关键字。表中的关键字包括：
Testid、TestName、TestStatus、TestPostion、TestFinish、TestResult、TestStartTime、TestEndTime、TestIntervalTime
用例驱动机制在开始执行测试时，首先由一个 Init()方法，将其驱动表中的状态值置成初始状态。然后发送“调用工具测试请求”信息执行测试，测试过程中进行等待一 TestFinish 状态为完成状态后，才执行下一个测试，TestFinish 状态由测试工具层次架构设置。
后台：应用脚本同样设计了一个读取和写入 CSV 表与 MySQL 数据库的机制，不过不同的是，其驱动机制发送的不是“调用工具测试请求”，而是直接调用后台架构进行用例执行。
2）数据驱动机制。
数据驱动机制，即是将数据存入 CSV 表或者 MySQL 数据库中，在测试工具框架中或者后台测试架构中调用即可。
3）结果读取机制。
这里的结果读取机制，即是能够将 CSV 或者数据库表中的完成状态、结果状态以及测试时间读到测试执行驱动模块中。
4）工具接入机制。
这里，很关键的是工具接入机制，即是利用自动化测试执行驱动模块能够调用不同的测试工具架构，例如：RFT、selenium、QTP 都只是作为其的一个插件，能把能插，我在这里设计了一个解析机制，即能够将这些工具的 API 写到一个配置文件中，由驱动机制解析，然后发向不同的工具调用其进行测试执行。
二、自动化测试执行驱动模块说明
所以，我这里只是设计了一个简易的自动化测试执行驱动模块，它也是作为自动化测试平台的一个模块，也可以说是一个“服务”。整个模块设计过程中，C

SV 表或者数据库的读写操作是最重要的，另外工具接入机制也是，如果真想将此模块做的更有通用性，那么一定要找准一个通用的中间件以及设计一个好的工具接入机制，那是漫漫长路。

三、自动化测试平台中的模块

1、 自动化测试控制端模块（包括界面控制端）

2、 自动化测试执行驱动模块（也可以说是 agent 端）

3、 自动化测试分布式模块(其是最主要的模块，它可以将所有的模块联系起来)

4、 自动化测试工具架构模块

5、 自动化测试服务器模块

6、 邮件服务模块

7、 ……

每一个模块可以说是一个服务，以一个主模块为主，有一个主要的机架，然后各个小型模块。围绕这个机架，就最终形成了一个“简易自动化测试平台”

基于 RFT 的自动化测试层次

简易自动化测试设计 之一

—基于 RFT 的自动化测试层次

序言：基于 RFT 的 swing 界面的自动化测试，这里不是说怎么去使用工具，而是怎么将 RFT 融入到自动化测试设计中，其实 RFT 在整个设计中只是一个辅助的角色，如果时间允许的话，谁也不想用这么昂贵还这么不灵活的 RFT，接下来，你读完这个系列后，你就知道为什么我这么说了。

一、简易自动化测试架构说明

很多人都应该了解了 GUI 自动化测试的通用架构划分如下：

1、对象层 AppObject，存储基于 find()搜索控件的具体方法,我分成了两大类，一类是通用的基本 java 控件、一类的不同产品线的相应的自己拓展的控件类。

2、 方法层 AppLib，存储简单的操作函数，在这里，我分成了两大类：Lib 与 Method 类，Lib 类用于对象的基本操作方法，Method 类存储用于一些基本的自己拓展的操作方法，例如:外部文件读写操作、日志写出操作、模拟键盘鼠标操作（Robot 类拓展）等。

3、 用例层 AppCase，调用测试方法进行测试用例的组织，不同的产品线对应不同的测试集，一个简单的测试功能模块对应一个测试用例。

二、简易自动化测试架构的实例

以下可以看一个整体的简单实例操作：

1、AppObject 文件夹中有一个 object.java 文件，存储一系列的通用的控件查找方法：

实例如下：

```
////////按钮组件/////////
        //用于搜索其上的按钮
        //输入：按键名称
        //类别：GuiTestObject
        public GuiTestObject  getButton(String  buttonName)
        {
                RootTestObject  root  =  getRootTestObject();
                TestObject[]  to  =  root.find(atDescendant(".class","javax.swing.JButton","accessibleContext.accessibleName",buttonName));

                return  new  GuiTestObject(to[0]);
}
///////文本框组件//////
//用于搜索文本输入框
        //输入：无
        //类别：WTextField
        public WTextField  getTextField(String  LabelName)
        {
                RootTestObject  root  =  getRootTestObject();
                TestObject[]  to  =  root.find(atDescendant(".class","javax.swing.JTextField",".priorLabel",LabelName));
                return  new  WTextField(to[0]);
}
```

a、GuiTestObject 是直接继承于 RFT 的所有对象的父类 TestObject，其里面包含了一些简单的操作，而我只想用 Button 的 click 方法，所以我这里就直接返回的是 GuiTestObject 类。

b、RootTestObject 也是直接继承与 RFT 的 TestObject，不过它与 GuiTestObject 类不同的是，它是直接指向被测的应用程序的，即是被测试应用程序

的控件的一个全部的 view，是按数据结构中的树的方式进行储存的，其中有方法为 find() ,可以基于属性用来在这个 RootTestObject 中查找你所需要的控件。

c、find（ ）方法查找的方式有三种，我重点说一下 atChild 与 atDescendant，前者是在查找根节点的直接子对象、而后者是查找所有的子对象。后面当然是要查找的控件的具体的属性了，在这里，我们可以定义一个基本不变的属性，然后定义一些需要在测试用例中改变的属性，例如：按钮，这里我设计的就是基于按钮名字来进行查找了，如果名字一样的话，你就得用一个别的方式了，例如：index。

d、WTextField 是 IBM 的 ITCL 提供的一些控件包裹类，其实就是继承了一些 RFT 的基本类，然后自己写了一些方法而形成的类罢了，你也可以自己拓展一些控件类嘛。

2、AppLib 文件夹中有一个 Lib.java 的文件，其类存储对查找到的控件对象操作的基本方法。

实例如下：

```
//下面是对对象的初始化
public AppObject.Rc_Object getObject = new AppObject.Rc_Object();
//下面是对共同方法的实例化
public AppLib.Rc_Method getMethod = new AppLib.Rc_Method();
//////////////////////按钮组件操作//////////////////////////
  //采用动态搜索的方法进行找到 Button 组件，并且执行 click 操作
      //输入：ButtonName
   public void ChooseButton (String buttonName)
      {
       getObject.getButton(buttonName).click();
       sleep(3);
       try {
      //向外部日志写其完成测试的步骤
            getMethod.logWriter("完成"+ buttonName + "的 ChooseButton 方法;");
      } catch (Exception e) {
        //可以调用异常处理模块进行异常处理
  }
```

```
  }
//////////////////////文本框组件操作//////////////////////////
      //采用动态搜索的方法进行找到 TextField 组件，并且执行 click 操作
      //输入：TextFieldName
       public void ChooseTextField (String LabelName,String TextNam
e)
      {
     sleep(3);
        getObject.getTextField(LabelName).setText(TextName);
        //用 JAVA 的语句将其 UserName 拆分成一个一个的字符进行输入
        try {
              getMethod.logWriter("完成" +LabelName+" ChooseTextFiel
d 方法");
      } catch (Exception e) {
              //可以调用异常处理模块进行异常处理
      }
      }
```

a、ChooseButton (String buttonName)方法就是找到一个名字为 buttonName 的按钮进行点击操作。ChooseTextField (String LabelName,String TextName)方法就是找到一个前缀名为 LabelName 的文本框进行输入 TextName。

b、此处有一个 getMethod.logWriter 是用于向外部 LOG 写入具体的控操作信息的。当然你也可以不写在方法中，写在测试用例中也可以，不过需要其返回一个 button 的名称，然后在测试用例中一个方法获得这个 name，然后再将信息写到日志中，个人认为后者方式较好，因为其可拓展性强。

c、这里以前设计的是在此抓取异常，然后进行异常处理操作，你可以选择调用一个共同的异常模块，我的异常处理方式是：抓取到异常，则关闭整个测试程序，重新进行下一个测试用例。

3、AppTestCase 文件夹中有一系列产品线的文件夹，然后每个文件夹对应每个功能模块，这里就不多说了，根据自身公司部门的情况决定吧。

三、一些需要注意到的地方

1、 用动态搜索的方法是脱离 RFT 的对象库的，所以无需使用 RFT 的对象映射机制了。
2、 你可以在外部基于测试方法进行测试用例的编写。
3、 因为整个系列只用了 RFT 中的 RationalTestScript 类中的方法，所以我本想直接用 eclipse 导入含有这个类的 jar 包，然后基于 eclipse 进行其测试用例的编写，但没成功，因为其回放的机制中还用到了很多具体的类，这些就是不可见的了，看样子商业工具在这方面做的很严密啊。
4、 根据这样，你就可以想尽办法，利用 RFT 提供的 CMD 接口，将 RFT 隐藏起来吧。
5、 其实一个自动化测试框架而言，RFT 最好能做到成为其框架的一个插件，所以用到的 RFT 东西越少就越好了。
6、 想想原理挺简单的，可是自己摸索的时候发现却要走了很多弯路，但是摸索的过程对于以后对整个测试活动理解还是很有帮助的，所以，个人觉得，不要怕走弯路，关键要摸索。
四、自动化测试平台中的模块
1、 自动化测试控制端模块（包括界面控制端）
2、 自动化测试执行驱动模块（也可以说是 agent 端）
3、 自动化测试分布式模块(其是最主要的模块，它可以将所有的模块联系起来)
4、 自动化测试工具架构模块
5、 自动化测试服务器模块
6、 邮件服务模块
7、 ……
每一个模块可以说是一个服务，以一个主模块为主，有一个主要的机架，然后各个小型模块。围绕这个机架，就最终形成了一个“简易自动化测试平台”

录制，到底给我们带来了什么

序言：最近与自动化测试同行讨论时，对于刚开始接触自动化测试工具的同行来说，总会有一些问题：没有感觉到写脚本相比录制回放的优势在哪里?用 RFT 来说，大家都很难感受到到底用动态搜索的方法构建的脚本与应用录制的方式形成的脚本到底有什么区别，到底其能够对于我们有什么提高，难道仅仅是提高了我们的脚本功底和对工具掌握的功底？也许，当你有野心的时候，事情往往不是这么简单。

一、自动化测试同行的疑问

记得我刚开始使用 RFT 进行 gui 界面自动化测试时，也是用的录制和回放的手段，后来一小段时间后就放弃了，也许很多刚接触自动化测试工具同行和我有一样，也是从录制入手，有的，也许和我一样一段时间放弃了录制，有的也许一段时间后还是坚持录制，好与坏，短时间谁也说不清楚，但是我觉得，自动化测试，不走到最后谁也说不清楚，或者自动化测试本身就没有头。

我把某位同行的疑问说一下（用的是 RFT 工具）：

“我没有感觉到写脚本相比录制回放的优势在哪里?当然我知道肯定是有优势的。首先录制简单，插入验证点，数据池都很直观，执行速度又快。那么我们录制完成，再在脚本中加入一些诸如 try catch，sleep()，logInfo()等防止意外和记录的代码来完善，那不就是很完美了，何必全写脚本呢？何必要动态查找呢？还慢。录制时自动把控件映射到 TestObjectMap 里了，直接用就行，根本不用 find()，录制完再改也方便。录制也是根据控件属性来定位，应该都不会出错。你说复用性，再复用也得写脚本，有那时间我再录制一下也够了；维护？维护写的脚本比录制的方便在哪，录制的脚本增删几个控件应该是很容易的事。”

总体来说，疑问可以分为以下几点：

1） 录制简单，手工在录制的脚本上根据需求进行二次开发就很完美了，何必全写脚本。

2） 录制的控件映射也很强大，无需用动态搜索了。

3） 复用性方面和维护性方面，有写脚本的时间直接再录制一遍不就好了。

二、 录制的优势

当然什么事情要以辩证的观点看问题，我先说说录制的优势吧

1 操作简单，测试人员容易上手，这对于大多没有编程经验的人来说是一件很有诱惑的事情。

2 构建测试脚本快速，在手工测试的同时，也可以将测试脚本生成好。

3 如果是商业型软件，就不用操心其测试工具带来的问题，大多都可以进行技术支持。

三、 我眼中的“录制”

以我个人的看法，做自动化测试，最大的也是最容易进入的一个误区，就是太拘泥于自动化测试工具，因为自动化测试的需求变化性很大，而商业型的自

动化测试工具的重点是面向于简易的操作性，这样往往是以失去韧性为代价的。因此，商业型工具中的“录制使用”则是自动化测试“误区”中的“误区”。

1）“录制简单，手工在录制的脚本上根据需求进行二次开发就很完美了，何必全写脚本。”，为什么会有疑问说录制比写脚本简单，那是因为你没有建立起一个合适的自动化测试框架，如果有了这个框架之后，你会发现你可以脱离 RFT 的 IDE，在任何时候任何环境下可以进行你的脚本开发，而此“测试脚本”在直观上就是等同于“测试步骤”。

2）“录制的控件映射也很强大，无需用动态搜索了”，静态的控件映射，其实是一种很不灵活的方式，其机制为了满足对象定位的精确性，而失去了其灵活性，想想，举 java 界面来说，其静态的映射方法是将其 java 界面映射成一系列的树节点关系，若是树的结构有所变化，那么其对象查找就会受到影响。并且其静态映射还需对一个控件的各个属性进行阈值计算，得到一个值后才确认是否识别。那么我们用动态搜索方法的话，就可以依靠我们的界面的特点，进行基于某一个特别的属性进行匹配即可。总之，越简单则越灵活。

3）“复用性方面和维护性方面，有写脚本的时间直接再录制一遍不就好了“，自动化测试定位很重要，为什么会有这么多失败，就是因为对以后会发生的情况的预估不够。想想，也许十个用例脚本维护量很小，可是等到成千上百个脚本的时候，你还能有心思进行维护，你可以说重新录制，那么这样的话，你会疲于录制，而最终放弃自动化测试。

总之，为什么录制会带来这么大的维护量？因为个人觉得，自动化测试的重点，也是难点就是一个适应变化过程。而录制方式，将其对象的查找、逻辑方方法的处理、测试用例的组织一锅端的放在了一个脚本中，当出问题时，牵一发而动全身。所以，分层是一件很重要的事情，有编程经验的人一定知道，在设计模式中提倡的原则就是，“面向接口编程，而不是面向实现”，也有一种说话“抽象不应该依赖于细节，而细节应该依赖于抽象”，即说软件编程模式，要做到职责细分，增加其拓展性与可维护性，如果将其抽象与实现写到一起的话，那么到后期，此软件在重用性和可拓展性方面则会很差。所以说，做自动化测试其实就是一门软件设计的艺术。

四、 如何应用“录制”

这里我不在框架上说如何去应用了，而只是说如果你只想先依靠自动化测试工具的自动化测试做一些事情，没有人力和精力去投入，虽然构建一个简单的自动化测试框架不需要很大的精力，个人以为，那么你就要在流程上把握了。

1）当一个产品不稳定时，那么别用其来构建自动化测试项目，可以尽量将录制用在需要进行很多次重复操作的过程，例如：需要对某个界面的某个输入框进行数值输入遍历测试。

2）如果你的产品界面很稳定，那么恭喜你，你可以试着去用，但是我还是不提倡，因为录制是不可能形成自动化测试规模的。

3）如果你开发组能够配合你不去刻意修改你的脚本中录制的界面，那么也恭喜你，但是这是不可能的...

当然，时代是发展的，技术也是发展的，也许有一天，录制的这种方式可以很稳定之后，但是我坚信的是，录制再发展，其也是基于我们的框架上的，因为我一直相信“需求引导设计”，只有自动化测试框架才是我们需求的集合，而录制只能是我们自动化测试框架上的一种实现方式而已。

分布式自动化测试平台设计

序言：经过了一段时间的探索，对于自动化测试平台项目的设计，已经进入了一个初步阶段，即自动化测试平台的基本形式已经实现，想想，自动化测试平台从无到有，自动化测试框架从基于需求环境的探索到基本实现真的是一个很值得玩味的日子，且将实现方式与大家分享一下，希望大家有什么好的想法或者好的建议也一起分享和讨论一番。

感觉，***这篇文算算是我自动化测试工作以来对自动化测试设计和理解的一个阶段性过程吧***，分享出来与大家共勉吧。

一、自动化测试平台定位

我一直崇尚的自动化测试观点是“**需求引导设计**”，只有将需求分析清楚，将需求定位好，才能着手去进行所谓的自动化测试框架与平台的设计。

而所谓的自动化测试平台（也可以认为是一个集成的自动化测试框架），我认为可是是若干个小型框架的组成，而每个小型框架有各自独有的作用，他们都是作为自动化测试平台的一个组件或者可以说是一个服务。

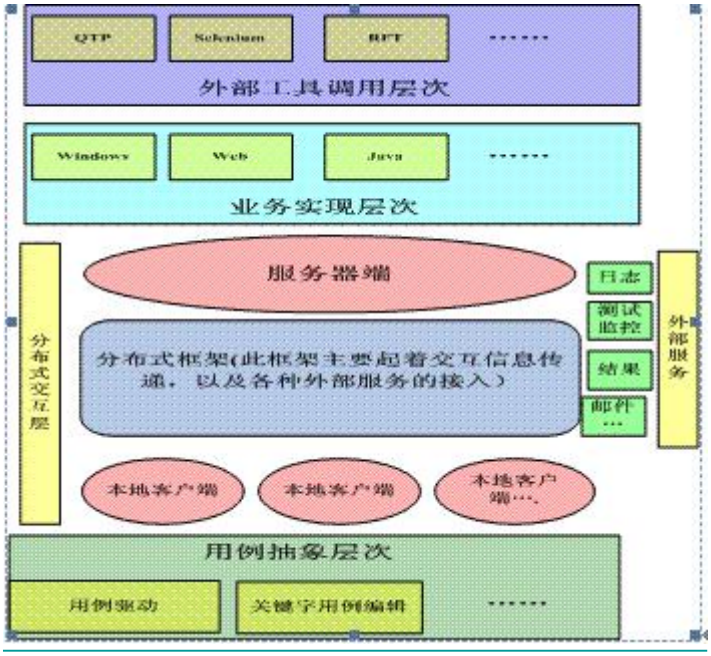
因此，其**自动化测试平台的原则**就是

1）1）普遍的可适用性。

2） 2）灵活的可插拔性。

二、自动化测试分布式平台设计

自动化测试分布式平台设计基本整体架构图如下：



按照层次分的话，可以分为：

1) 用例抽象层，此层功能为：

a) 可以进行测试驱动，即调用测试用例驱动表进行测试（测试用例驱动表可以存储在本地端，存储方式可以为 EXECL 表格形式或者数据库形式）

b) 可以在本地客户端进行用例脚本的编写和生成。生成的脚本可以通过转换至与外部工具调用层次中的工具想对应的脚本。（其统一的脚本形式可以为 XML 实现）

2) 业务实现层，此层功能为：

a) 可以实现各种业务，即抽象出业务，通过不同的业务实现方式调用不同的测试框架实现。

b) 可以通过不同的业务实现方式调用不同的测试外部工具实现。

3) 分布式交互层

a) 客户端与服务器端进行交互的通道

b) 各种外部服务的接入的通道，例如：日志服务、测试监控服务、邮件通知服务等。

4) 外部工具调用层

a) 运行不同的驱动命令，调用不同的外部工具进行测试。

三、自动化测试服务器端设计

1）自动化测试服务器端的功能

a) 本地服务器保存测试集脚本。（也可以说各种工具的小型框架都是保存在其上的，例如我将 RFT 分成的三层框架，都是保存在服务器上的）

b) 本地服务器调用外部工具进行测试（外部工具安装在服务端）

c) 进行用户的权限识别。

2）自动化测试服务器端的设计

设计一个服务器，可以接收分布式框架发过来的请求信息，并且此服务器端依靠“多线程”处理机制，集成了各种外部工具的操作命令。

四、自动化测试客户端设计

1）自动化测试客户端的功能

a) 进行用例驱动表的解析。

b) 进行自动化测试用例脚本的编辑和生成。（采用关键字驱动，即测试人员可撰写和识别的脚本）

c) 统一结果的导入与查看。

d) 详细日志的导入与查看。

2）自动化测试本地客户端设计

设计一个客户端 ,main 程序是实现一个界面 ,通过其界面进行测试用例的选择，测试用例脚本的撰写和生成、结果和日志查看等。

五、自动化测试平台应用分析

优点：

1）通过此平台，层次定位清楚，将测试执行与测试驱动分隔。

2）通过此平台，流程结合容易，在流程上可以和产品的测试周期相结合，把握好其进度。

3）通过此平台，职责细分清晰，最重要的是能够将职责细分，这个在后期的想写的一篇文章将阐述。

4）通过此平台，用例复用性高，统一用例适用于各种不同的语言。

5）通过此平台，业务拓展灵活，可以任意进行各种工具调用，能够进行各种业务的测试。

6）通过此平台，结果分析简单，可以方便进行结果查看和日志调用查看。

缺点：

1）因为服务器端安装的工具，所以在测试人员在调试上很麻烦。

2）根据详细的日志往往不能够清楚定位问题，还需要测试人员人工定位，不能一次性定位出问题。

3）还有很多问题等待大家挖掘，提出宝贵的意见……

六、自动化测试平台设计心得

1）这次设计，对编程而言，是个很大的挑战，其实个人觉得，java 的设计模式与其自动化测试框架的思想很是相近，其在这次自动化测试本地客户端设计上就采用了其模式的思想，因为 java 设计模式的理念就是**降低耦合，增强灵活性和复用性。所以，学习软件思想对于自动化测试设计很重要。**

2）自动化测试，就算有了平台，但是在进行**自动化测试效用分析**上上才是一个很大的难点，因为只有将效用定位清楚，才能将平台用到实处。

3）实践指导理论，再多的理论只有实践之后才知道，而且不能过多的追求技术完美，不然等到技术研究透彻，黄花菜都要凉了

如果有测试同行对自动化测测试有兴趣，欢迎一起学习探讨，

自动化测试细节设计之道

序言：自动化测试，个人以为，很重要的两个方面：一是能够做到无人值守的情况，自动化测试过程能够顺利进行。二是能够做到有详细的结果报告。将其具体在细节上分，则有几个重要的部分，一是：自动化测试用例的组织管理。二是：自动化测试的执行方式。三是：自动化测试对异常的处理。四是自动化测试的日志。五是自动化测试的报告反馈形式。接下来，结合 RFT 来从技术实现上说明自动化测试细节的设计之道。

一、自动化测试工具的应用方向

个人觉得，RFT 的优点就在于其丰富和开放的 API，你完全可以基于这些 API，脱离 RFT 的 IDE，因此，最好的方式是尽量将 RFT 提供的功能进行删减。

个人认为：伴随着这么些强大工具的推出，为什么还有好多一些公司的自动化测试一直不能规模化，究其原因是因为自动化测试的具体“战术”是根据环境变化的，而自动化测试工具设计的时候只是为了做出一款集成的测试工具，是根据工具设计人员对自动化测试理念设计出来面向大众的，大多中小公司拿到工具之后，以自动化工具为中心去做自动化测试，而渐渐脱离了做自动化测试的本质。

所以，做自动化测试，需以公司需求为中心，尽量将自动化测试工具给拆离（拆离的方法便是从底部开始，例如 RFT，就可以从其 API 入手），然后将其满足需求的部分应用到自身的自动化框架中。

二、自动化测试框架重要方面简述

其重要方面，一为执行无人值守。二为结果定位准确，

其实，自动化测试，个人认为，分为两种情况：一是辅助手工测试；二是回归版本测试。第一种情况就是在手工测试的执行过程中，有一些重复的测试步骤或者很繁琐的步骤（做自动化比手工测试执行的时间和效率高），这种自动化测试其实也可以认为是半自动化测试，它主要是帮助测试人员解决一些无味的工作。第二种自动化测试可以俗称为“例行测试”，它必须做到无人值守的情况下执行，其主要作用是回归版本，加速产品发布速率，并且保障产品质量。因此在这种测试过程中，其结果的定位尤其重要，在例行执行完毕之后，需要有一份总的结果报告，并且有问题的情况下能够根据 LOG 记录详细定位问题。

三、自动化测试框架的具体细节设计

1、 自动化测试用例的组织管理

这是自动化测试框架设计很重要的一个方面，管理的方式可以通过“EXCEL 表、CSV 文件、数据库等”，然后应用“自动化测试框架执行引擎”去解析“自动化测试用例表”去执行自动化测试。而其中自动化测试用例表中可以依靠一些关键字对自动化测试用例的执行方式和状态进行定义和查看。

2、 自动化测试执行方式

通过“执行引擎”去解析自动化测试用例表，根据其用例表的“执行状态”的关键字判断其是否需要执行。并且下一个测试用例的执行还需要读取上一个用例的“完成状态”为“完成”之后，才执行，否则，一直处于等待过程。

3、 自动化测试对异常的处理

自动化测试过程中往往会出现一些异常造成自动化测试过程的中断，因此如何去处理这些异常，保证自动化测试的不间断执行，个人设计的方式是：在具体的测试用例中加入异常判断处理（try-catch）,然后判断过程为只要抓到 Exception,测试过程则会关闭整个测试过程，并且在“自动化测试用例表”中将相应的“测试完成”状态置为“完成”，将“测试结果”置为“失败”，并且进行下一个测试用例。

4、 自动化测试日志

RFT 的日志反馈太过于凌乱，并且不好与自动化测试框架的结合，因此，个人认为，可以将其日志去掉，建立自己的 LOG 机制，在每个方法中设计其 LOG 的输出，而当遇到异常时，则进行异常的输出，这样的话，就能够从全局上把握其测试的结果了。

5、 自动化测试结果

对于结果的查看，可以在“自动化测试用例表”中进行统一查看，由关键字“result”定义，若有错误的，则再根据具体的日志去定位问题。

总之，以上只是一些自动化测试设计的细节想法，当然，一定还有人有更好的方式，希望能够共同探讨。但是，个人坚信，工具本身是为测试需求服务的，所以，大家一定要警惕自动化测试工具的陷阱，不能陷入以“自动化测试工具”为中心的漩涡中，而要以自身的自动化测试框架为中心，将自动化测试框架插入其中，这样你才能做到“收发自如”。

顺便一句：其实我们的人生又何尝不是这样呢，吾以为：人生的最大追求为能够掌握自己的内心，真正做到“收发自如”，但若限制太多，抓不住本质，又何能做到此呢。

正所谓：

花开花谢，云卷云舒，静看日出日落，卧听雨滴天明

以何吾能达此状态矣？吾问吾心，答曰：自然而为之~

与君共勉之…

基于前后端交互的自动化测试框架设计

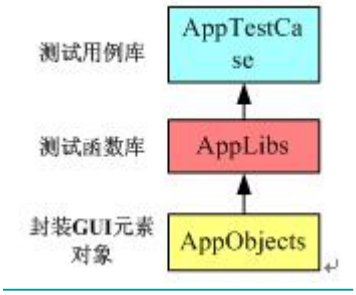
序言：一般对于 C/S 架构与 B/S 架构系统而言，都会包括前端界面或者页面的验证测试，也包括后端设置是否生效的测试，当两者结合测试时，就得有一个好的前后端交互机制来进行协调控制，进而保证测试的有序性与完整性。此，介绍一下自己对于应用 RFT 来进行前端 swing 程序的控制以及其后端脚本控制设备来进行交互的框架的研究与设计。**希望有同道之士能够对其提供意见即可。**

一、前后端测试简介

在一些测试中，例如，电信自动化测试项目中，需要进行前后端的测试。如果前端是 C/S 架构，则需要在客户端上进行命令的操作和下发，然后在后端终端上应用 CLI 进行命令下发是否成功；B/S 架构类似。而前后端的测试方法各有不同，所以在交互上需要有一套良好的机制来保证测试的有效运行。

二、前端测试方法

前端测试，主要是采用 Gui 测试工作进行其应用的控制。
C/S 架构的工具，包括：RFT、QTP、abbot(java 的自动化开源测试工作，主要是针对其组件的应用)等。
B/S 架构的工具：包括：RFT、QTP、selenium 等。
在此，以 RFT 测试 java 程序为例子展开说明：
根据 AUT 的具体情况，将其测试架构分为了三层，如下：



1）AppObjects，可以采用动态搜索（find）的方法进行界面组件的查找定位，也可以应用 getter（即取得组件容器）的方法。比较如下：
1、采用 find 的方法，增加了编程的复杂性，但是在属性的查找定位上更加灵活，拓展性和识别能力也更强。若是有公司产品研发能够做到提供对其组件提供唯一标示 id 作为识别属性 ,那么其在组件定位及错误定位上得到很大的帮助。
2、采用 getter 的方法，编程简单且直接面对组件对象，但是其组件查找属性已经固定，所以界面的改动会容易造成组件查找不到。
所以，可以针对不同的组件应用不同的方法，具体在实践中总结。
2) AppLibs，测试函数库，可以分为组件调用方法库与通用方法库。组件调用方法库是针对 AppObjects 传过来的组件进行操作；通用方法库则是一系列在架构中需要用到的一些通用方法，例如：模拟键盘的输入、文件的读取、数据库的读取方法等。
3）AppTestCases,测试用例库，具体划分针对不同产品，例如：有的按产品线划分，有的可以按同一产品不同型号划分；每个用例为一个模块，能够保证单独执行；且可以建立一个通用的用例库，任一个用例可以进行调用。
三、后端测试方法
1）对于后端测试方法，可以应用脚本技术即可实现。例如：在电信测试项目中，应用脚本技术测试后端的优点如下：
a) 脚本编写简单(命令+参数的形式)，应用方便，直接使用解释器执行即可。
b) 与测试仪、分析仪的结合使用方便。
c) 处理字符的能力较强，容易满足需求。

2)对于应用 java 等语言实现后端测试亦可，不过在编程上实现较为复杂，成本较高，具体情况可以针对公司具体情况分析。

四、前后端交互方法

其交互框架图如下所示：

执行平台调用其用例驱动表进行测试。测试驱动表可以用数据库、EXCEL、CVS 等方式进行存储，其关键字段包括：

1、 TestCaseName:识别和调用测试用例的名称。

2、 TestPosition:测试的位置；此字段识别是前端测试（front）或者后端测试（back）。

3、 TestStatus:测试用例执行的判断依据；此字段是用来判断测试是否执行的依据，yes 为执行，no 为不执行。

4、 TestFinsih:测试完成的状态；此字段是用来判断测试是否完成，也是提供下一个测试用例执行的判断依据。

5、 TestResult；测试结果的状态；此字段是用来最后查看测试的结果；测试通过则置 pass,失败则置 fail

前端与后端的交互则也是通过此测试驱动表来执行的：

1、 执行平台开始执行测试，首先从驱动表中读取第一行用例与用例的状态，如果 TestStatus 为“yes”,则根据 TestPosition 与 TestCaseName 调用执行用例。（RFT 与 QTP 都有直接调用 command 的形式执行测试）

2、 若执行的为前端测试用例，执行过程中，若需要调用后端的程序，则直接调用其后端测试用例，并进行循环状态，并且不断进行其相应后端用例在测试驱动表中的 TestFinsih 状态，若其相应的后端测试用例完成，则其后端测试程序调用测试驱动表，将其 TestFinsih 状态置为”yes:,前端程序读取到后，将继续执行其相应用例。

3、 最重要的一个机制，就是异常处理机制，若测试执行过程中，测试遇到异常后，其相应测试用例的测试不能向下进行，则可以根据异常关闭当前测试过程，将 TestResult 状态置为“fail”，将 TestFinsih 状态置为”yes”，则可以继续执行以下的测试用例。

4、 测试全部完成后，可以进行其测试用例的 TestResult 状态查看，测试通过的 TestResult 值为”pass”，测试失败的为”fail”,然后可以去进行 fail 的测试用例的具体错误信息的查看。

如上，基本上就是一个简单的交互过程，可以参考，当然也可以应用程序线程同步的机制实现，不过其要求编程功底高，另外，处理方面也较为复杂。

五、测试人员体验设计

当然，如果要将其自动化测试过程开展下去，光靠以上框架是不够的，其测试执行平台的设计是一个很重要的环节。

个人研究发现，最好的测试过程是能够每个测试人员能根据提供的测试平台去编写和生成自动化测试用例脚本，要做到这样，个人以为，测试平台可以包含：

1）脚本编辑模块：不能去指望测试人员花多大的精力去学习编程，因此测试执行平台需要以最简单和直观的形式提供测试用例脚本的编辑环境：

a) 执行前端自动化测试脚本的编辑和生成；能够自动的生成前端测试脚本，并且能够自动进行编译操作。（

b) 执行后端自动化测试脚本的编辑和生成；对于脚本，则可以省略编译的操作过程。

2）测试驱动表读写模块：此模块用于调用测试驱动表并进行测试用例的调用读写。

3）结果查看模块：此模块用于对测试驱动表中的具体结果信息进行查看；例如，对于 fail 的测试用例，则可以应用执行平台去调用其相关的具体测试结果进行查看和分析。

当然，最重要的还是定位好测试需求，这需要根据产品和部门情况，了解到如何去将框架用到最佳处，一般这方面的测试应用在回归测试中比较好，不能指望其重点为发现问题，而是保证质量和加速版本的发布进程。

大话“自动化测试框架思想与构建”

*序言：*也许到现在大家对所谓的“自动化测试框架”仍然觉得是一种神秘的东西，仍然觉得其与各位很远；其实不然，“自动化测试框架”从理念来说，并不复杂，但其之所以神秘，是因为其运用起来很是复杂，每个公司，每个部门其产品线，其运作流程都是不同的，所以就导致了在想运用“自动化测试框架”去完成自动化测试时产生了很多不定因素，导致了很多自动化测试项目的失败，让人对“自动化测试框架”开始敬而远之。

而自动化测试发展也有一段时间了，为什么到现在虽见其火热，但难见其规模，关键是大家对其的定位，很多公司以及很多人都知道做好自动化测试不简简单单的靠一个工具，而更需要一个框架，但其总是对“自动化测试框架”缺

乏清晰的定位，很容易将其定位成了一个固定的框架，其实个人理解不然，自动化测试框架不是一个模式，而是一系列思想的集合，是将各种自动化测试框架思想集合应用去搭建成的一个分层组织。

一、简述自动化测试框架

也许很多人印象里的自动化测试框架就是一个能够进行自动化测试的程序似的。其实这不全面，真正的自动化测试框架可以不是一个程序，它仅仅是一种思想和方法的集合，说白了，就是一个架构，大家应该都知道操作系统其实也是一个架构吧，你可以把其理解成一个基础的自动化测试框架为一个简单的操作系统，它定义了几层架构，定义了各层互相通信的方式。通过这个架构我们才能在上面进行拓展我们的测试对象（核心体）、测试库（链接库）、测试用例集（各个 windows 进程）、测试用例（线程）,而其之间的通过参数的传递进行通信（即相当于系统中的消息传递）。

二、自动化测试框架思想

接触过自动化测试的，一定不会对以下几种“自动化测试框架思想”陌生吧。

l 模块化思想

l 库思想

l 数据驱动思想

l 关键字驱动思想

很多人都将以上定义为“框架”，而我却觉得它们都只是代表了一种自动化测试的思想，不能以纯粹的框架定义。

首先，我们来看看自动化测试的一个发展，就能更加明白这些思想的真谛了。

a) 第一代自动化测试，即自动化测试思想刚开始诞生时，依靠的是传统的“录制-回放”技术，这种技术与现在的工具的“录制-回放”思想不一样，其其实就是一个“模拟”的过程，即模拟你对 PC 的操作而形成的，其基于你对键盘的输入与对鼠标的操作，原理与按键精灵等类似，这种机制对环境的依赖性太强，对变化性太过于敏感，因此不可能发展成一种规模。

b) 第二代自动化测试，即脚本化的自动化测试，利用脚本进行结构化的自动化测试，此可以应用于 CLI 与 API 的自动化测试，在其就开始集成了模块化与库思想。

c) 第三代自动化测试，开始产生了各种自动化测试思想，包括数据驱动与关键字驱动思想，其伴随着对象化思想的产生，而且也造就了现在一系列的

自动化测试软件，其实其中都集成了这些思想，从这时候开始，自动化就开始实现了一定的规模，开始运用在各个行业，并且发展趋势越来越快。

现在将一一根据自己的个人理解来介绍这些“自动化测试框架思想”：

1、 所谓模块化思想，就是将一个测试用例中的几个不同的测试点拆分并且将其单个点的测试步骤进行了封装，形成了一个模块。

例如：一个测试用例要对一个登录程序进行测试，其中包括：用户名输入、密码输入、以及确定登录；

那么就可以将用户名输入、密码输入、确定登录、取消登录四个操作分别封装在四个不同的模块中。测试时，只需调用其模块即可。这样的话，当一个模块有变化，你只需单独维护那个模块即可，也可以根据模块的不同组合成不同的测试用例。

2、 所谓测试库思想，就是模块化思想的升华，其为应用程序的测试创造了库文件（可以是 APIs、DLLs 等），这些库文件为一系列函数的集合。其与模块化思想不同的是，其拓展了接口思想，即可以通过接口去传递参数，而不是一个封死的模块，可以说是一个多了一个“门”的交互型模块。

例如：还是以上那个测试用例，只是将用户名输入、密码输入、确定登录、取消登录封装成一个库，这个库含有一个函数 Login，这个函数 Login 接收两个参数“用户名、密码”，对输入不同的用户名和密码可以进行不同的测试用例。也可以另外一个函数 Cancle。

3、 所谓数据驱动思想，众说纷纭，很多人都觉仅仅依靠用 EXCLE 表进行不同数据的读取仅是一个高级的参数化，其实怎么理解并不重要，关键是其思想能够好的应用到你的框架中。而我的理解就是变量不变，数据驱动结果，不同的数据导致了不同的结果的产生。而对于数据的导入，可以通过很多方式，例如：EXCLE 表、XML（用在 WEB 中）、数据库(DB)、CSV 文件、TXT 等都可以。

4、 所谓关键字思想，这个思想，我曾经一直思考，它与面向对象的关系，与交互模块化思想的区别。后来个人理解，其实关键字驱动就是一种面向对象的思想，例如：QTP、RFT 中，对象可以为一个数据或者一个关键字，对对象的抓取，可以将其测试对象封装为一个关键字（即可以将 gui 元素封装成了一个个关键字），这样可以对其关键对象进行各种操作了，不同的对象可以驱动不同的测试流向与结果。

简单的应用的方式可以用一个 EXCEL 表，里面包括“对象类型”“对象名称”“对象操作名称”“判断方式”“预期结果”。这样的话，可以通过导入不同的对象类型和名称、不同的对象操作来构建成了一个测试用例表了。

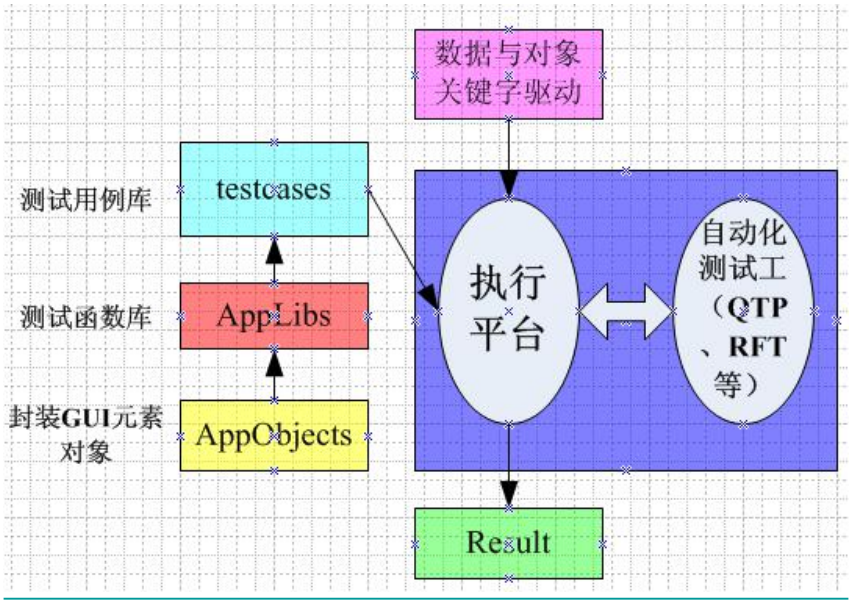
以上只是对这些思想的个人理解，做好自动化测试，不是说你掌握了一个框架，而是要掌握其自动化的思想，然后根据这些思想，结合你不同的测试环境和流程来构建你自己的自动化测试框架。

三、 构建自动化测试框架的策略

1、 永远记住，你的“自动化测试框架”是给测试人员用的，如果你真的想把自动化测试做成一个规模，那么你需要将测试工程师当做你的用户，你不能指望他们有耐心的去编写测试脚本或者指望他们能够像你一样对这些思想有良好的掌握。你要将他们当成什么都不懂的用户，因此你的框架必须是“一切简单化”的化身，简单的操作、简单的维护、简单的拓展。

2、 做一个自动化测试框架主要是从分层上去考虑，而不是简简单单的应用一种思想，它是各种思想的集合体。

例如，做 GUI 自动化测试，简单的一般就将其分为三层，其框架如下图所示：



而其中，可以贯穿着自动化测试的各种思想，例如：对象层中有关键字的思想、可以将对象库标示在 Excel 表中进行管理 ,或者应用动态搜索的方式传递对象识别参数。tasks 层中可以封装各种方法，形成一个大型的方法库，而每个方法中可以应用上数据驱动的思想。

3、 真正的自动化测试框架是与流程上结合的，而不简简单单的靠技术实现，技术其实不是很复杂，关键就在于对其架构和流程的深刻把握，而这需要很长

的一段时间，所以不要指望一口气能吃成胖子，只能一步一步按需求来，需求指导思想的应用。

四、自动化测试框架的发展趋势

个人认为，自动化测试从初始诞生到至今，已经经过了一段漫长的日子，而其仍处于上升期，特别是现在软件大爆炸、敏捷模式、云端的开始热门，测试难度和质量保证的难度开始越来越大，自动化测试的比重也会越来越大，而单存的自动化测试是无法实现规模化的，因此，自动化测试框架热门化的趋势化的必然的，那是，在各种框架思想的集合中，各种框架将散发出各自的璀璨，来帮助我们快速的完成各种测试。

以上仅仅是至今，个人对“自动化测试框架”的理解，也许在以后的日子，因为认识的加深而会有不同的火花蹦出，但至少觉得现在的框架对自己的项目能够进行应用，也许某一天，需求饱和时，那么，新一轮的远征探索就又要开始.....

希望，我们大家在自动化测试的征程上能越走越远，也希望自动化测试能真正成为测试流程中“不可缺少”的一部分。共勉之。

自动化测试实践经验和教训

自动化测试**实践经验和教训**

序言:在部门做自动化有好一段时间了，经历了自动化从无到有，然后到框架，到现在的平台，以及持续集成，回顾发现由于自己之前经验太浅，走过的弯路太多，现在也还在谨慎的前进着，上次又回顾了一遍”软件测试经验和教训”里的自动化测试章节，发现早前很多懵懂的经验，现在稍稍清晰，于是想着结合自己的历程精简出一些经验吧。现在经验还是尚浅，如果有更深认识的朋友，互相讨论，谢谢

一、所谓自动化是为了软件发布服务的，并不只是为了测试服务

来源:自动化测试目标建设

以前一直怀疑自动化测试的用处，我们之前花费大力气开发了大量的基于关键字方式的脚本，用来提高测试的覆盖率，每次测试耗费大量时间，但是发现的问题少之又少，虽然说，自动化测试不是用来发现问题的，是用来验证软件没有问题，但是有一个矛盾在于我如果不做自动化测试，问题还是那么少，那么做自动化测试我们难道只是为了追求一个心理感受吗？这个概率问题怎么平衡

后来，这个经验是在与开发一起合作冒烟测试建设，到现在的持续集成建设，开始明白，自动化测试的好处是为了增强开发的灵活性和保证软件开发流程的有序性

1)快速检测新版本的不稳定变更,即冒烟测试，能够快速验证当前 build 版本是否可以继续下一步或者提测，此处冒烟测试可以是单元测试、集成测试和基本功能覆盖测试，常用的框架和工具:Junit、TestNG 和接口测试框架(soapui、httpClient 等)、界面测试框架用于基本的界面测试(QTP、RFT、selenium)。

2)尽可能的暴露回归程序的错误，即例行回归测试，测试部门在开发提测后，基于开发的测试单，手工重点测试变更内容，利用自动化有选择性的覆盖其他功能。此处更多的是到了系统测试层面。

3)验收测试，在发布前应用自动化的各种手段进行一次基本功能验收测试，通过率在多少以上才可以提交发布，而且此处环节还可以加入非功能测试。

所以说，我们做自动化测试的目标一切都是为了软件发布服务，也可以理解为部署软件生产流水线，在这里，推荐一本书，《持续交付-发布可靠软件的系统方法》

二、不要事后去计算人工替代率，而是要参考自动化测试有效性

来源:自动化测试度量和 ROI 分析

之前在年底做自动化测试度量的时候，要求每个产品线将自动化测试执行次数和时间进行统计，这样做的目的是为了统计自动化测试执行时间，然后要求测试人员算出每个模块的人工耗时，然后进行换算，得到自动化替换人工的时间，后来算完后，发现这样的数据其实是没有意义的

1)测试的本质是不可以比较的，一次手工测试执行有可能比自动化测试执行也许是更有意义的。

2)大多数的人工耗时都是拍拍脑袋想出来的

3)运行只是表明自动化测试被执行，但不能证明这次执行是有效的，只有真正本来需要手工的测试被自动化执行时才有意义。

所以，在度量上，一定要保持维度一致，可以横向比对，即不同模块之间进行比对，有效性上可以从自动化测试模块的应用场景和执行次数、执行频率去评估，成本上，即是自动化测试开发和维护成本

三、度量一个自动化测试的可实施性可以从其可控制性或者可测试性上来考虑

来源:自动化测试可测性分析

有的人说单元做自动化比较好，有人说接口测试做自动化比较好，但奇怪

的是也有人说他们做界面自动化比较成功，其实说接口测试和单元测试比较好的人，单元测试往往是开发来定义的，所以开发来控制可测性，接口测试的话，接口协议都是约定好的，所以可控制性和可测性有保证，而界面的话，有些产品的界面可测性控制比较好，例如都有唯一的 debug id，或者界面变动性很小，那么也是考虑可以用界面自动化的。所以说，我们在推广自动化测试过程中，不是人云亦云，而是要结合当时的环境，从可测性上去考虑自动化测试的开展。

四、试点推进自动化测试

来源:自动化测试推广

自动化思路重要，但是做自动化本身也是具有一定机遇性的，其环境相关性很大，所以在开展自动化测试前，一定不要过度自信，铺天盖地的推广，而是要找好试点项目。之前做过一个接口录制工具给测试人员推广使用，由于不是试点分析，每个组的测试人员拿着这个工具开发了一堆一堆的线性脚本，导致存积了大量的接口线性脚本，然后后来由于脚本的维护性和不稳定，慢慢的流于形式，甚是可惜。如果采用试点分析的方式，先在小范围进行使用，并且跟踪效果。

业内有一个分层测试的理念很好，可以基于 IP 和地域选择性的发布新产品，根据使用效果，然后再根据情况逐步推向全国。

五、自动化测试框架的重要作用之一是为了职责分层

来源:自动化测试推广和框架建设

所谓软件框架功能，我觉得很重要的一点是能够将每一层次的责任细分出来，数据驱动框架就是将数据单独抽离，让测试人员能更有效的管理数据，例如:可以构造重复的、组合的、随机的或者大量的数据包;关键字测试框架就是将对象封装、任务封装、业务组装分层，这样，可以让代码能力稍强的人员面对对象和任务封装，让业务能力稍强的测试人员面对业务组装，这样可以通过框架将各个人员的职责细分，做自己所擅长的事。

关键字框架推荐 robot framework,利用其可视化界面 ride，其中还有一些拓展包，例如 selenium2Lib 可以应用在 web 测试。

六、可以的话，让开发一起参与自动化测试

来源:自动化测试可测性分析

曾几何时，我基于 RFT 和 QTP 都试点推行了公司的界面自动化，但是效果不佳，究其原因还是界面的变化太大，加上一些自定义控件或者一些自定义图片展示的动态搜索无法查找，只能用存储对象的方式，后来大力发展接口自

动化，将界面自动化先搁置了一段时间，后来开发自己做了一部分自动化，说是效果甚佳，去学习才发现，他们有一些图形和事件是基于 xml 配置定义文件描述的，这样利用 JDOM 技术，解析出文件，然后根据 QTP 的脚本编写规范，生成 QTP 脚本，而测试动作基本上涵盖增删改查。这种情况是我们测试之前无法知道的，因为开发流程原因，我们无法知道开发所采用的开发技术，所以说，我觉得可以和开发一起来评估自动化测试。

七、自动化测试是可以成为一种习惯的，尽早自动化

来源：自动化测试推广

之前，和几位华为的开发朋友聊天，问他们一个问题：你们自动化怎么样，我原来以为开发是不太了解自动化的，没想到他们给我的答案是，什么怎么样，这是必须做的工作呀。原来他们华为的持续集成过程，开发自己写测试脚本做冒烟测试，以前这是华为的规定，现在是他们开发人员的一种习惯。

所以，我觉得，推广自动化不一定一开始就要大张旗鼓的，也不是所谓的等到一切时机成熟，自动化测试是一种辅助测试的方法，不同的情况可以采用不同的方式，几行脚本，一个 bat 部署文件，一个数据统计，也是自动化的一种应用。**用另外一种说法:不管黑猫、白猫，能抓到耗子的就是好猫。**

八、自动化应该立即见效，见效后应该慎重评估

来源:自动化测试推广

有时候如果我们对目的不够清晰，不能抓好本质问题解决，就很容易走极端，有人说录制不好，我们就反对录制，有时候我们抓到一个录制工具就以为遇到了神器，孰知后面是一个一个的大坑；而我觉得，录制不是不好，而是如何将它用在合适的地方，录制的好处是无需编程技能即可快速上，但是他的缺点是相对脆弱，一旦 UI 或者接口变化测试就会受到影响，分散的脚本不可重用且难以维护，而且系统在测试前必须可用（也就意味着无法使用 A-TDD 方法）。因此这种方法并不适合大型自动化测试。而关键字驱动，将数据与关键字结合来描述如何使用数据执行测试，这种方法是具备数据驱动的优势，同时非编程人员也能建立新类型测试。所有测试由同一个框架来执行，无需不同的驱动脚本。然而初始成本很大，但是可以使用开源方案，因此非常适合大型项目。所以，我们刚开始的时候，选择合适的项目，可以采用脚本录制这种方式，让自动化测试立即见效，小范围推动项目进度，然后，就严格控制，开始设计框架，把握可测性。

九、不要指望用自动化测试平台一下子解决所有的问题

来源:自动化测试平台建设

当年在作自动化测试过程中，遇到瓶颈，就想当然的认为要开发业内所说的自动化测试平台，即先弄出一个线，然后再想办法去线上发展每个点，后来历时几个月，加班加点的，总算基于 STAF 开发出了一套分布式的自动化测试平台，但是后来用着用着就不了了之了，分析出原因是问题本身和目的都没有明确，单纯的开发出了一个架子，然后再去找衣服强塞进去是不可行的，架子本身是为放衣服服务的，所以平台本身是为让自动化更有序的进行而服务的，之后我们大力发展点，而平台是等点都满足需求后，开始将点串起来，自然而然的形成了一条线，然后再加以修饰，就成为了平台。所以说，不要舍本求末，追求花哨，而要踏踏实实的做好每一个点，以需求为导向，自然而然的才能把架子搭好。

在这里也推荐几个用来搭建平台的较好的工具和框架:STAF，Jenkins,selenium grid 可以用来做分布式测试执行和测试节点管理，sonar 可以用来做质量管理**平台，更多的是面向**白盒测试**分析。Toast 也是一款不错的自动化测试平台，我觉得它的理念更多的像项目管理+jenkins。**

十、将所有的测试资源都版本控制起来

来源:自动化测试配置管理

这个很重要，在我们的自动化建设过程中，因为我们是多产品线的，每个产品线的有些功能模块是共用的，我们的方法是将每个模块的功能抽象成关键字接口，所有产品线共用接口，只是实现上不一样，而这种方式前期的问题是版本管理的问题，因为一个产品线开发出一个模块后，各个产品线开始互相迁移，导致迁移混乱，还有一个问题是产品功能模块也在不断的更新，不同的产品版本要对应不同的测试脚本，所以后来，加入版本管理，而最终的测试脚本版本用 V1.0.0 进行跟踪，第一位表示接口版本、第二位表示产品特性，第三位表示当前产品线的脚本维护。

版本控制中重要的两个概念就是分支与合并，随着开发的版本控制的不断深入，测试的版本管理也需要跟上，常用的版本控制工具是 svn、cvs 以及分布式版本控制 git，和基于流的版本控制系统 Clearcase，个人觉得，前期可以用手工的方式定义好版本，等到开发协作强大到一定程度后，可以采用这些版本控制系统。

十一、 时刻反省自动化测试过程

来源:自动化测试推广

自动化测试是一个长期的过程，我们即要低头走路，也要不时抬头看路，也需要不时的休息一下，回顾自己走过的路程，整理之后才可继续向前，所以需要我们每隔一个阶段，就要好好反省自动化测试的开展和推广过程，大家的时间都是很宝贵的，珍惜自己的时间，更要珍惜大家的时间。

总结;因为时间仓促和水平有限，有些经验和教训不一定很全面，或者还有朋友在推广或者平时观察到有更好的自动化测试实践经验和教训，希望可以一起总结，总之，个人觉得，学习过程最有效的方式是理论和实践相结合，知行合一，在实践中发现问题，然后去推敲理论，解决问题，最终总结从而变成自己的东西。

软件架构与测试架构分析论

序言:在做自动化测试的过程中，曾努力的对软件设计学习过一段时间，间间断断的，突然发现接触到了软件流程，软件建模（包括数据库建模）、web 设计，当然，更多的是用的 java 设计（数据结构、模式应用、算法等）。虽然不够深入，但是将这些细碎连接起来，在软件架构上也有了自己的一点思考和总结，其实不管做测试也好、做自动化测试也好，还是进行软件设计，都有着一套理念类似的架构学。

一、生活中的架构

相信大家都知道“**马斯洛需求层次理论**”，在其理论中，把人类需求分成生理需求、安全需求、归属与爱的需求、尊重需求和自我实现需求五类，依次由较低层次到较高层次排列。这里不讨论其理论的正确性，这也是一种架构，将人类需求实现的各个层次进行了细分，每一个层次之间都有详细的职责细分，每个层次之间都有清晰的关系接口，因此普通人能够快速基于此架构拓展自己的特性需求。

在我想来，人的思想意识也需要自己的架构，你的最底层是你的原则，也就是做人方式，是固定的，不会轻易更改，变动越大，影响越大，然后依次是做事方式和表现方式，可以随着环境的不同进行适应。

二、软件设计架构

1、软件设计流程：

一般的软件设计基本流程会是：**需求分析—架构设计—系统设计—系统开发—测试上线。**当然，从产品整体上而言，需求分析前期还有战略制订以及可行性研究，上线后期还包括应用数据分析等。

2、软件架构介绍：

软件架构简单而言，觉得有一句话说的很好：在一定的**设计原则基础**上，从**不同角度**对组成系统的各部分进行**搭配和安排**，形成系统的多个结构而组成架构，它包括该系统的**各个组件**，组件的外部可见属性及组件之间的**相互关系。其实说的简单点：软件架构就是从整体上定义软件系统的运作流程、各个模块以及模块间的接口。**

3、软件架构作用

软件框架的作用是使软件设计层次清楚，可以达到不同模块细分到不同个人开发的效果，各个接口之间能够无缝连接。**但是架构虽好，也需要看实际情况，不能一味为了追求花哨的架构，而忽视了其成本。**软件架构的真正作用是分离关注点、提升软件需求灵活性和维护性。但是相应的，其前期复用的成本效益也显著提高。因此，需要根据软件项目的规模综合考虑，选择一套合适自己的架构，只是为了临时住而搭建一个草房子，就没必要又是设计地基、又是设计扩展性等问题了。

4、软件架构师发展

从一个从软件实现角度上说，写代码并不是其最重要的工作，最重要的工作就在于其架构的设计，当架构设计好之后，其具体代码细节则是一个填充的过程。就像做房子，房子图形架构都设计好了，那么水泥和各材料就只是按照其形式搭建而已。

而根据我的总结，要成为一名软件架构师：则必须经历这么一个过程：

软件业务学习——开始能够写函数和类（即能够写得了过程性的函数和对象性的类集合，能够应用数据结构和算法实现）————写模块和引擎（能够写一些给外部调用的模块和引擎，能够应用模式，能够定义好自己的接口）——写框架与平台（能够应用抽象思想，善于利用各种架构以及搭建自己的架构，能够区分和选择技术重点、快速验证技术实践等）。自我认为自己还在第一个阶段。反思以前开发的测试项目，大都接口定义太少，更多的是面向类实现的编程，而不是面向接口的编程，最近一直在看 java 的源代码，才深有体会面向接口的编程和一些简单模式的真正应用——**软件业务分析。**

三、测试架构

1、测试架构介绍

测试架构，业界常分为“测试管理架构”和“测试技术架构”，也许很多公司，大多数测试经理就担任着这两类的测试架构师，他们往往对测试流程设计负责，也对测试技术的应用选型负责。

其作用不言而喻，但是很多人将架构误认为测试框架，其实不然，测试架构不等于测试框架，其架构还包括测试业务应用定位，例如：一个公司刚开始做自动化测试时，最容易犯的错误是没定位好实施自动化测试的架构流程，一开始就大规模的套用测试框架，然后开始调用测试人员写脚本，到头来却发现这套框架根本不适合测试的形式，其实脚本录制算是一个必然要经历的阶段，只是这个阶段可以大可小，可以通过录制去满足好一部分的测试业务需求，然后等需求稳定和饱和之后，再设计发展适合自己的框架等。所以说，凡事都如造房子，得一步一步来，才能建出好的房子，否则一味的求快，架构不牢固，只会吹散在风中。

2、测试架构师发展

测试技术架构，**从业务角度而言**，可以是以可重用的测试框架来思考测试点和测试方法，例如：采用的测试方式是功能测试、探索性测试、例行测试、实验局测试还是性能测试等。然后规范出各个测试设计模板。**深入技术角度而言**，就是对测试方法的技术重点的选择和部署，例如，功能自动化测试和性能测试架构设计。因此，**测试架构在技术上是与软件架构理念相通的，而成为一名技术型的测试架构师，需要经历为：**

测试业务执行学习—开始写测试脚本或者使用工具进行测试——搭建自己的测试框架或者基于工作设计框架——设计测试平台，能够从整体运营上把握整个测试执行过程——**测试业务分析。**

四、分析论

1、大家注意到上面的红字，我个人觉得，每个人都会开始于业务，回归于业务，再好的架构只是架子，而架子上没有衣物的话，太大的架子反而是浪费，所以，角度上而言，架构是服务于业务的，业务是驱动架构的，即：**测试业务驱动技术，技术带动测试业务。**

2、 **架构的能力发展不在于架构本身，而在于一个的分析和抽象能力的发展**，我们平时总觉得做测试没有技术含量，但是个人觉得，恰恰相反：一个开发人员，编程是他的工具，如果他只会工具，那么他永远只是个”码农”；一个测试人员，测试执行和测试软件是他的工具，如果他只懂工具，那么他永远只是个“测

试执行者”。一个开发人员，如果他永远只会架构，那么他只能算是一个开发者。一个测试技术人员，只懂得设计框架和平台，那么他只能算是一个测试者。真正的架构师们都是能够将实际业务相结合，能够分析和提取各种信息和资源所用的。**或者，可以这样说，每一个架构师都是一位管理者，他们掌握了“合理分配和利用资源的能力”。每一个架构师都是一位思想者，他们掌握了“思考业务和架构的最佳融合的方式”。他们以最小化的成本实现了最大化的价值。当然，这只是我的一番杂乱说辞罢了，是对是错，还待研究。**

3、对于我们个人人生架构设计，则是一个出发和回归的过程，在这个过程中，我们只需保持不急不躁，一步一步的打好基础，不间隙的慢慢向前即可，这样，也许才能走的更远更稳。

数据库设计在自动化测试的应用

简易自动化测试设计 之（五）

—数据库设计在自动化测试的应用

序言：大家对众多软件都应该操作过，不管是 c/s 架构还是 b/s 架构，个人觉得，很多时候，软件的很多问题的出现其实是在于数据库的设计方面的遗漏，而一些性能问题也大多与数据库方面的处理有关，前段时间，对数据库的东西进行了了解，因为自动化测试平台需要一个集中的服务器，而服务器上需要应用数据库管理。

一、数据库在自动化测试应用

自动化测试发展到一定规模后，需要有一个自动化测试平台的服务器进行集中管理，其中服务器上保存所有产品线的测试脚本、各个日期的测试结果以及综合环境拓扑表、各个产品线的拓扑表（当然包括环境初始化参数）。

而这样，对应的数据库中则需要保存测试脚本的特性（为测试脚本建一张表，每张表里的属性为用例 id、脚本名称、脚本路径等）、测试结果的特性（属性为项目 id、结果名称、结果状态、结果路径等）、测试拓扑环境表（拓扑 id、拓扑产品线名、拓扑对应路径等）。这样，依靠这几张表，就能在服务器上管理测试中所需要的资源，并且不断更新内容，也可以调去历史记录。

二、Web 设计简单介绍

根据自己业余的有限的项目设计经验说说软件设计过程，以前觉得看例子倒还好，但是真正自己设计的时候，才发现比想象的难，其关键不是在于编码如何，而是在于需求规划与设计方面（概要设计与规划设计），例如：我设计一个 we

b 网站，复杂的话，可以考虑用 UML 建模来设计用例图、协作时序图，当然简单的网站，我建议直接列出功能模块即可。

首先，进行系统设计，包括系统前台与系统后台的设计，简单列出系统前台的功能模块与系统后台的功能模块，之后在详细说明。

之后，进行界面设计，根据系统前台与后台功能模块，将其需要的父页面与子页面都进行简单设计，这个过程，是用 HTML 与 JS 实现。

之后，进行数据库设计，其实数据库设计是一种 E-R 建模，可以用的工具有 ROSE 与 powerDesigener,简单项目，我推崇 powerDesigener。建模之后，直接由 CDM 生成 PDM 即可，其中自动生成了你所选的 DBMS 的 SQL 脚本。

之后，就是**后台交互的设计**了，我用的是 JSP，可以用 selevet+javabean 的模式或者直接应用 SSH 三大框架都行。

最后，其 web 在经过本地 web 服务器部署测试 OK 后，就可以上线应用了。

当然，C/S 架构其设计理念差不多，例如：应用 java 的 swing 界面设计时，也是差不多的步骤，也许随着流程不一样而不一样吧。

三、数据库模型设计

首先，我根据我的一些学习很浅显讲述一下数据库的大概设计方面：

1、 现在的数据库模型采用的是关系模型，即具有列和行的表。（与 EXCEL 表是一样的，不过 EXCEL 表的容量有限，而且最主要的是表与表的无法建立起好的联系，性能也不是很好）。其表的列就是属性，代表其键（例如：testID 就是一个键，标示性的作用），而后面对用的行就是其属性的值，也叫记录。还有一系列的术语，我就不一一说出，大家可以参考数据库资料。

2、数据库设计第一步是根据你的系统规划你数据库中需要的表、以及表对应的属性、表之间的联系。

3、 然后应用 powerDesigener 设计 E-R 模型，在 powerDesigener 中，有几种模型文件，分别为 CDM、LDM、PDM、OOM。设计简单数据库模型，我觉得用 CDM 与 PDM 即可，先应用 CDM 将实体（表）的属性以及其联系描述出来。然后转换为 PDM（PDM 是可以连接具体的 DBMS，例如：mySQL 与 DB 就是两种不同的 DBMS）

4、 我依据 51testint 的博客简单设计了一个数据库：

其中有 blog（博客管理）、article（文章）、links（友情链接）、feedback(评论)、photo（相册）、music（影音管理）。其中每个表之间有联系，例如：blog 与 article 就是一对多的联系，然后，article 表中包含 blogid 属性，

说明其所属的 blog，这样就将这两张表联系起来了，建立了一种主从模式。之后，可以将图中的 CDM 模型转换为 PDM，选择 mySQL,就可以生成对应应用于 MySQL 的 SQL 脚本，可以生成此数据库，并且在数据库中说明了表间的联系。

5、 之后，你博客登陆后，就能依靠这个 blogid 去定位到你的博文等信息。

上面只是简单介绍，具体如何设计操作及工具如何使用，就不说了，只想说，数据库设计是很重要的环节，而且了解了数据库设计，对于软件测试也是很大的帮助，可以帮助你去关注到软件设计人员在软件设计时容易遗漏的地方，例如：在设计数据库时，你如果没有在 article 中建立一个 blogid，那么所有人都能查看到文章。

—散步的 SUN

自动化测试平台化策略 之“即插即用”型策略

自动化测试平台化策略

之 “即插即用”型策略

序言：什么是自动化测试平台？这个是没有一个好的定义的，个人拙见，自动化测试平台就是根据自身公司或者部门的流程，将自动化测试的需求融于上述测试流程，然后提供一个软件平台的形式表现出来，也就是用规范和协议的形式表现出的一套自动化测试体系。

一、编程工具中的“即插即用”型

Eclipse 平台是 IBM 向开发源码社区捐赠的开发框架，其是一个成熟的、精心设计的以及可扩展的体系结构。这个平台允许任何人构建与环境和其它工具无缝集成的工具。工具与 Eclipse 无缝集成的关键是插件。除了小型的运行时内核之外，Eclipse 中的所有东西都是插件。从这个角度来讲，所有功能部件都是以同等的方式创建的。

你可以在安装好的 eclipse 的文件夹下有一个 plugins 的文件夹中有其各种插件，eclipse 的核心较小，几乎都是由插件组成，而所有的插件库有四个基础库：

· 标准 Widget 工具包（SWT）：Eclipse 中处处使用的图形化组件：按钮，图像、光

标、标签等等。布局管理类。通常这个库被用于代替 Swing。

·JFace：菜单、工具条、对话框、参数选择、字体、图像、文本文件的类和向导基

类。
· 插件开发环境（PDE）：辅助数据操作、扩展、建立过程和向导的类。
·Java 开发者工具包（JDT）：用于编程操作 Java 代码的类。
基于这个基础库，然后遵照其 eclipse 开发插件的过程，你就可以将自己的工具与 eclipse 集成起来，即根据自己的需要去定制自己的开发平台的需求。

二、软件交付平台的“即插即用”型

IBM 其软件产品有一个词叫 jazz，之前很不理解这种想法，后来慢慢的为其庞大的理念而感到心动，虽心动却也只能研究一下。

Jazz 是一个用于整个软件生命周期的团队协作平台，旨在支持跨所有软件生命周期阶段的任务的无缝集成。Jazz 平台的主要作用是为工具编写人员提供要使用的机制和要遵循的规则，这些机制和规则可产生无缝集成的生命周期工具。这些机制通过定义良好的 API 来公开。

Jazz 是一个基于客户端-服务器体系结构的平台。通常在受保护的服务器级计算机上运行的 Jazz 服务器承载一组服务，并在其存储库中存放数据。远程客户端通过网络使用 HTTP 与 Jazz 服务器通信。

个人理解的话：jazz 提供了一个开放式的平台，其中基于了一些国际上的组件规范（例如：OSGi 等，OSGi 称做 Java 语言的动态模块系统，它为模块化应用的开发定义了一个基础架构。这样，一个大的系统可以被划分为很多模块或组件，其通过标准化的接口进行交互通信），然后，IBM 的大多数工具可以集成到这个平台上成为软件交互生命周期的一个整体，尽量使得各个工具在使用上能够进行交互，之后，可以根据自身的开发流程情况，基于软件实现定制自己的开发和交付流程。

三、自动化测试平台的“即插即用”型

自动化测试中因为其应用特殊性，所以，会有各种工具的使用（界面测试工具、命令行测试工具等）以及各种自动化测试的模式（例如：回归测试、例行测试等）来提高测试效率。所以，个人觉得，自动化测试也需要提供一个开放式的平台来集成这些工具和测试模式。

可以参考的是：开源的自动化测试框架 STAF，其提供了一个“即插即用”型的概念，任何工具或者模块只要遵从其规范，则能作为其中的一个服务与构建与其上的各种服务进行通信。其还是作为一个分布式的框架，其意思即每台运行 STAF 的机器都是等同的，都可以拥有各自的功能模块与数据，也可以在分布式网络中进行共享与交互。或者，不基于 STAF 也可以自己进行类似框架的开

发，需要的是提供一个标准的接口形式，各个模块能通过这个标准的接口互相进行交互。

当然，以上的形式需要根据自身的情况来定，是在自动化测试需求发展到一定程度上，如果连自动化测试需求和流程都没有定义下来，那么，开发这套平台的意义将会变得很空洞，而且容易脱离实际需求，反而越走越远，浪费了成本，所以，**“效率为上、需求为导”，不同的时候应该采取不同的策略来应用自动化测试来提高自身的测试效率。**

自动化测试测试平台策略 之 模块交互策略

自动化测试测试平台策略

之模块交互策略

序言：要做一个自动化测试平台，越强大的平台，其模块之间的交互越难，也就是各个模块之间的接口定义越难，而如何采用一种策略去规范各个模块的接口、消息格式和交互方式更是难，这一点，我觉得可以从学习网络协议中找到一丝灵感，那些协议的交互方式以及消息的格式传递都是值得学习的，以前觉得学习协议纯粹是为了了解，现在学习真的是想掌握其几点精华思想，突然能够想象到：一群人在一起思想的碰撞，不断的去总结，去发现，去实用，才有了现在的协议标准。觉得，**不一样的领域都去发现才能有所感悟。**

一、自动化测试平台中的模块

1、软件产品是具有一系列特定功能的组件组成，其系统可以被分为一系列的功能模块，每个模块所特有的信息处理过程都被包含在模块的内部，如同一个“黑箱”，这就是“封装性”，然后模块与模块之间按照一定的规则相连则成了一个复杂的系统（一个系统也可以作为一个模块，去组成更复杂的系统）。

2、而在自动化测试平台的系统开发中，首先，按照其抽象出来的自动化测试流程和方式划分一系列的功能模块，这些功能模块都能脱离系统独自使用，有的模块独自使用能够提高一些效率（像 CLI 测试中的基于脚本的自动化测试框架，GUI 测试中的基于工具的自动化测试框架等都是一种模块 ,还有一些测试工具），然后在此些模块的基础上，我们定义一些交互规则，将他们以最好的方式进行安排在自动化测试流水线上，然后提供一个统一管理的界面，慢慢的，就构建成了一个自动化测试平台。（**关键点：必须在测试功能模块独自提高效率的基础上，能够在测试流程中加以应用，平台的作用伴随更多的服务，将他们流水线化**）

二、自动化测试平台平台构建难点

1、 如何在测试流程中提炼自动化测试流程，然后抽象出自动化测试流水线。

2、如何能够统一规划整个测试部门的测试资源，这需要一个服务器去统一存储和调度管理。

3、 然后，很关键的一步：如何将已有的功能测试模块结合起来，即建立一套良好的交互协议和交互消息格式，使其能够很好的交互和互补，真正完成自动化测试流水线的运作。

三、自动化测试平台模块交互策略分析

以下策略是自己对测试平台建设中的一些提炼，不一定完全适合，可以参考

1、 策略 1：服务注册机制

即定义一个框架，其框架提供相应的 API，所有的模块利用这个 API 遵循一定的规则都可以作为服务注册到这个框架中，注册的同时，也对该服务的发送和接收的消息进行了规定，不同的消息可以调用其模块不同的功能，然后，模块则可以以这个框架为媒介，以传递消息的方式互相控制。当然，对于数据的存储，需要单独提供一个空间，可以是服务器，也可以是内存机制。

2、 策略 2：消息分发统一机制

即定义一个消息分发模块，模块的交互都会经过这个消息分发模块，这个消息分发模块将消息进行解析，然后传送到对应的模块。

这里，两种策略都需要统一规定好消息的格式。可以参考网络协议中的一些消息格式，例如一个简单的：

源头	目的头	时间	消息类型	消息内容

总之，个人上次听过一句话，如果你在一个行业领域已经停滞不前，那么就将眼光放宽，去另外一个领域去看看，往往这样，会激发你无限的灵感。这也许是真理啊，乔布斯不就很爱好艺术吗？马云不就是个金庸迷吗？**当然，不是说要去复制，个人觉得，说的是不要闭门造船吧，而是学会提炼共性，在思想中升华吧。**

自动化测试平台化策略 之 平台质量策略

自动化测试平台化策略

之平台质量策略

序言：在应用软件开发自动化测试方面的产品也有一段时间了，某一天，与一位研发中心好友聊天，其问到：你们的自动化测试产品有人测试吗？我不禁哑然，随着自动化测试水平的提高和自动化测试的深入，才发现，自动化测试已经完全作为一个成熟的软件产品，但是这个成熟的软件产品却没有一个很好的测试流程去保证其质量。

一、自动化测试平台质量方面

自动化测试平台从里到外，可以包括几个部分，当然这也是一个自动化测试的发展过程，从小到大的一个过程，而这个几个部分所要关注的质量方面为：

1、 自动化测试用例/脚本

记得，从最开始的只有脚本化的自动化测试时，那时候脚本的质量是自动化测试的一个最大的问题，测试人员往往要花费大量的时间去分析到底是产品问题还是脚本质量问题，而脚本质量的问题的保证不仅需要测试开发工程师的能力决定，还需要其产品的变更性作为保证；而由于开发工程师不是很懂业务，这就造成了脚本在测试方面有很大的难关。而一般的保证其脚本质量的措施是：测试开发人员自测—测试人员使用测试—测试人员提出问题—测试开发人员解决问题—测试人员使用测试（反复性），这样反复的过程就容易造成大量的维护成本，而自动化测试与其他软件产品一样，也有自己的生命周期，所以很容易造成了其自动化测试脚本的淘汰。所以，自动化测试脚本的质量保证是要靠测试人员，而不是主要靠测试开发人员。

2、 自动化测试框架

框架是整个或部分系统的可重用设计，表现为一组抽象构件及构件实例间交互的方法。而自动化测试框架的作用，我认为，是为了保证自动化测试产品（脚本）的可维护量、可重用性等，即保证脚本的使用质量。而在自动化测试框架开发过程中，涉及的问题主要是在于其框架的定位、框架的异常（包括脚本的异常以及框架本身的异常）的处理、框架的结果产生方式等。

3、自动化测试平台

自动化测试平台作为一款完全的自动化测试产品，其涉及模块很多，所以，其重点在于保证平台的模块之间的交互性，而其主要问题也在于各个模块的交互性方面的问题。

二、自动化测试质量保证方法

1、自动化测试脚本

自动化测试脚本的质量保证有两个方面：

1）一个良好的生产自动化测试脚本的生产框架，且将逻辑操作和变化封装，由测试开发人员去保证脚本本身质量。
2）然后将业务组成部分交由测试人员。由测试人员去保证测试脚本的业务质量方面。

2、自动化测试框架

自动化测试框架的质量保证方面：

1）个人觉得，首先将自动化测试框架的层次划分清楚，每个层次对应的关系要明确，例如：举几个简单的例子：CLI 的脚本框架，就要区分为：底层的脚本生成的方法（生成脚本库，包含最基本的框架方法）—中层的脚本库—上层的业务用例（由脚本库构成）等；界面的测试框架，一般区分为：控件搜索层或者定位层—中层的控件方法层—上层的业务用例。
2）完成框架的定位和分层之后。则需要建立一套好的框架异常处理机制，即能够很好的区分是框架本身的异常，还是脚本的异常。则测试人员只关注脚本异常方面，测试开发人员则要关注框架本身方面。

3、 自动化测试平台

1）、在保证了单个框架应用都没有问题后，则由一个测试平台去贯穿，所以，在开展自动化测试平台时，其模块的交互，即接口方面的定义是重点。保证了接口方面，则其平台的稳定性则能大大保证
2）、在开发过程中，需要很好的利用开发的一些模式和原理，个人在开发过程中，有一些经验：设计先行（定义好外部接口、内部模块之间的接口、内部模块之间的数据结构、实现代码的层次要分清楚，例如在界面开发上，多采用 MVC 的思想，保证其动态的变化性；数据库或者外部未见的读取则可以定义一个持久层，之后要定义一些模式：单一模式和工厂模式一般用的比较多；也不能为了模式而模式，这样反而增加了代码的冗余性）—编码实现（实现过程中要注意首先搭建好基本框架、每隔一段时间要注意集成测试）。

三、自动化测试质量保证的几个策略

总体来说：

1、 因为测试人员与测试开发人员的技能的区别性，所以想办法的将职责细分，各自保证各自擅长部分的质量，要充分利用好广大测试人员的资源。
2、建立一套完善的测试自动化相关产品的测试流程，其也需要问题的回归和收集，但其重点在于产品本身，而不是脚本。所以自动化测试脚本需要作为其自动化测试框架的产物。

3、 自动化测试产品开发是很重要的，一定要站在软件实现角度，即要有良好的软件外部和内部接口、良好的软件设计框架，即保证软件的高内聚、低耦合等原则。

4、个人觉得，自动化测试产品的质量保证方式可以结合敏捷等一些开发方式中的优势，例如：迭代式开发，持续集成等；因为自动化测试是一个与测试人员共舞和拥抱变化的开发过程。但是其需要一个懂测试业务和精通开发的人员把握，

所以，个人觉得，自动化测试的人才是一个能够结合测试流程和开发流程的人才。

自动化测试平台策略 之 自动化测试管理平台

自动化测试管理平台

序言：不同的自动化测试平台所面向的重点不一样，一个好的自动化测试平台是来源于测试流程，而归于测试流程的，所以，首先定位测试流程以及自动化测试在测试流程的几个应用点是很重要的。在此，我想根据我个人的理解介绍几个不同的测试平台来理清一下自动化测试在其测试流程中的应用点。

一、自动化测试管理平台简介

自动化测试管理平台其重点是突出在测试管理方面，在这里可以从不同的角度进行分析，确定其测试平台所拥有的测试管理功能：

测试职责方面

1、业务测试人员的职责:其主要负责对测试项目的管理、测试用例的编写、测试场景的设计以及测试数据的准备等。

2、测试执行人员的职责：其主要负责对测试场景的部署、测试状态的监测以及测试结果报告的查看和分析。

3、自动化测试开发人员的职责：其主要负责对测试平台的自动化测试框架库进行添加和维护，设计开发不同的工具适配。

测试流程方面

1、测试管理：包括测试项目管理、测试场景管理、测试用例管理、测试对象管理以及测试数据管理等。

2、测试调度：包括测试场景的分发、自动化部署以及驱动执行（包括时间与事件驱动）

3、测试监控：包括测试环境与测试执行情况的监控。

4、测试结果管理：包括测试结果分析与缺陷管理。
5、测试系统管理：包括权限管理以及各种系统功能管理。
二、自动化测试管理平台系统架构及分析
其自动化测试管理平台的系统架构如下图所示：

三、自动化测试管理平台实现分析
1、 前端管理平台的实现方式可以是 B/S 与 C/S 架构，个人以为 B/S 架构更合适，其瘦客户端的形式，可以更好的方便测试人员随时进行测试管理以及测试结果的查看。而实现技术方面，则可以应用 JSP、PHP 以及 ASP 等技术。
2、 测试用例的设计可以采用拖拽的 web 技术方式实现，先自己新建一些 web 控件，包括各个测试步骤以及各个箭头，然后应用 JS 技术实现拖拽。
3、 自动化测试脚本设计：应用测试对象库在界面上进行脚本设计。
4、 测试用例与脚本的关联：记得有一个模型驱动的思想，即新建的测试模型，每个模型关联一个测试场景；当然，这里不是，即只是将用例与脚本的一个关联。关联技术可以应用到数据库了。
5、测试对象设计；这个是自动化测试开发人员的职责，即设计底层的测试对象，每一个测试对象关注一个测试方法或者一个共性测试步骤。然后这些测试对象各对应一个测试数据库。例如：RFT 的三层框架中，其方法层就可以看做一个对象库，QTP 也一样，但是 RFT 与 QTP 应用的脚本不一样，所以在这里，我们可以用一种语言进行统一，这就是适配层的功能。
6、 测试调度设计；测试调度即测试环境的部署与执行，当然，这里可以一个测试任务为最小颗粒，测试执行人员根据不同测试项目与测试需求来新建和安排其测试任务。其测试执行人员需要从测试用例库中调用其测试用例来组建测试任务。（测试用例已经与测试脚本关联了）。测试执行分发时根据其执行机器的状态进行分发。
7、 自动化测试监控：可以定时由测试执行机来返回其状态。以及执行机器上各测试任务的执行状态
8、 自动化测试结果管理，执行完成后，可以由执行机将结果上传，并通知相关人员和存储到数据库中，方便历史查询。这里需要提供统一结果和具体结果分析报告。
9、或者分析出来的问题，可以通过集成的测试管理工具与测试缺陷工具进行报告的提交以及具体缺陷报告的生产（包括其缺陷分布图等）

四、从测试流程几个点分析

个人觉得，总结起来，一个好的自动化测试管理平台有几点

1、 接口集成性，能够提供方便的接口来适配各种流行的测试工具。

2、 统一协作性，能够提供一个统一管理、统一存储与统一规范的功能。

但是一个测试管理平台，不是说做就做的：

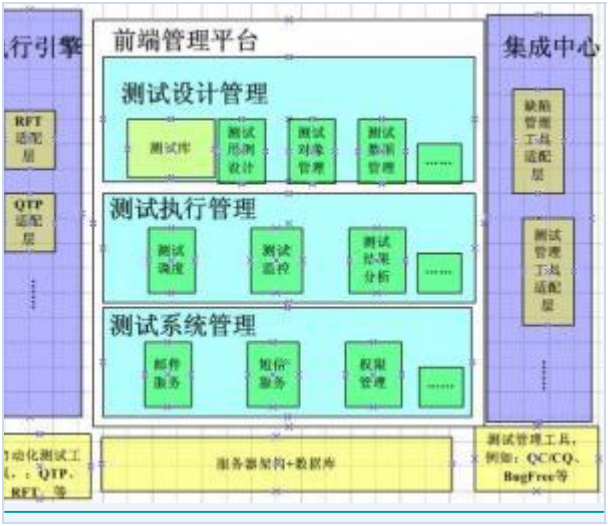
1、 需要测试流程清晰

2、 需要自动化测试发展到一定水平，即有了明确的测试框架，能够起到一定效用，自动化测试平台只是将这些点的结合。

3、 需要明确自动化测试的作用点和效果，自动化测试辅助测试的阶段。

4、 好的人员，能够懂得测试和开发的人员。

我的个人观点：自动化测试的最终目的是协助测试人员操作规范化，建立一个统一的测试管理平台，自动化测试中的真正核心还是测试理论。



自动化测试平台策略 之 电信测试系统策略

自动化测试平台策略

之 电信测试系统策略

序言：与一般的自动化测试平台不同，由于电信硬件设备的拓扑复杂性和人工参与性，电信领域的自动化测试系统平台的重点在于环境拓扑管理，所以，**如果能够将设备拓扑管理那一块做好，那么电信自动化测试算是成功了一半。**也可以这样说，电信自动化测试策略就是对测试资源的一个管理策略。根据我的一点经验和想法，谈一谈电信方面的自动化测试平台系统。

一、电信自动化测试系统设备管理简介

电信领域的自动化测试主要是对设备的系统功能测试，而对设备的操作管理分为三种方式

1、 CLI 管理

传统和大多现在的电信设备，都需要通过命令行进行设备的控制。

2、 C/S 管理

C/S 架构的网管系统，例如华为的 T2000 或者 N2000。这些网管系统是起到对网络系统设备的设备以及其连接关系进行监控和管理的。

3、 Web 管理

B/S 架构的设备管理系统，例如：一般的家庭网关设备或者家用路由器设备，都会有一个 web 界面去配置，所以其系统主要起到配置单个设备的功能。

二、电信自动化测试系统的体系架构

第一种解决方案：（如图 topo1 所示）

其自动化测试系统会完成自动化从任务计划、任务调度、资源调度、拓扑连接控制、设备配置管理和控制、设备之间的通讯、自动化脚本解释和执行、日志收集和查询、缺陷报告整个自动化过程，然后处理用户角色管理、TESTBED 管理、数据库管理等工作。

这种解决方案的特点是提供控制网络，由控制网络根据拓扑配置文件来负责构造脚本运行所需的拓扑，一般而言，这种方案比较适合于 ETH 交互和路由设备的网络，因为构造逻辑网络的控制方法可以依靠划分不同 VLAN ID 来形成。

1、 管理客户端：用户界面负责与用户之间的交互。主要是管理测试设备、管理测试拓扑和新建测试 Job，查看结果等。

2、控制网络：控制网络由控制网络交换机、PC 节点机、终端服务器、电源控制器等设备构成根据测试脚本的测试拓扑文件，选择合适的 TESTBED。

3、测试网络：测试网络由数台测试网络交换机、PC 节点机、测试仪、待测设备等构成。在自动化脚本运行前会根据测试用例所定义的拓扑配置文件通过控制网络在测试交换机上自动划分 VLAN 将测试用例所需的空闲测试设备选取出来组成独立的逻辑测试网络，并且根据测试用例地址配置文件的需要通过控制网络初试化各测试设备测试接口的 IP 地址、网络掩码和路由。

4、服务器集群：保存用户信息和各测试结果信息。

第二种解决方案：（如图 topo2 所示）

与第一种解决方案不同的是，这种方案没有控制网络，每个脚本有一个标准拓扑文件，而对于一个大的综合环境有一个真实拓扑文件（包含各设备具体信

息），可以应用算法匹配的关系寻找符合条件的设备进行连接测试，这种方案可以应用于大多数拓扑，而且测试灵活。

1、 管理客户端：用户界面负责与用户之间的交互。主要是新建测试任务、测试设备查看管理以及测试拓扑查看管理等。

2、测试管理端：由管理 PC 构成，包括设备管理、拓扑管理以及任务管理中心，用来进行测试任务的调度和测试拓扑的映射查找。

3、测试网络：测试网络由 PC 执行机、测试仪、待测设备等构成。进行测试脚本的执行和结果的返回。

4、服务器集群：保存用户信息和各测试结果信息。

三、电信自动化测试系统的设计策略

第一种策略：采用 B/S 架构进行设计，这种方式的好处是平台环境简单、测试用户可以随时依靠 web 进行测试的管理。例如：有的网络设备系统测试时只需监测不同包的处理，则可以用一个自动化构造网络包，自动化进行打流来完成，那么可以用 B/S 架构设计，随时在 web 上进行流的构造和结果查看。

第二种策略：采用 C/S 架构进行设计，这种方式的好处是可以考虑应用分布式环境，然后搭建环境灵活。

四、电信自动化测试系统的操作步骤

第一阶段：管理员工作

1、 搭建测试床或者测试拓扑环境。

2、 搭建服务器或者测试执行机环境。

第二阶段：测试工程师工作

1.设计和编写脚本（测试用例）

2.设计和编写配置文件和拓扑配置文件。

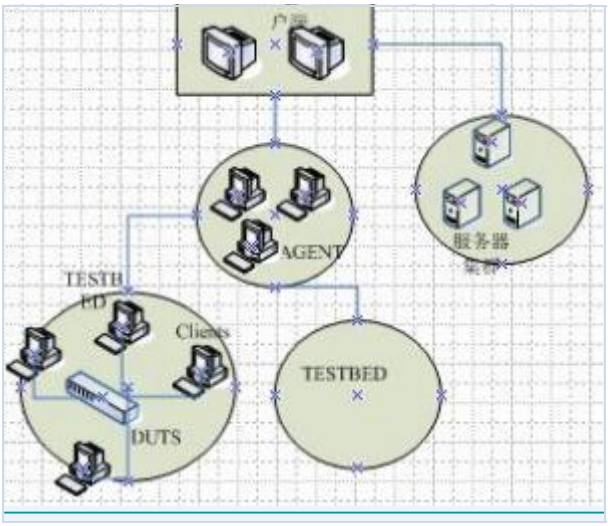
第三阶段：

1、新建测试任务或者测试 job，提交作业将排队，在等待作业运行过程中，可以做其他的事情。

2、运行作业，可以查看实时监视报告和日志。作业完成，可以查看最终完整版版的报告和日志

总之：电信自动化测试平台系统的建设不是一早一夕之事，需要综合不同的测试环境进行不同的考虑，而没有一个能够统一的策略，所以做自动化测试难点就在于此，但不管什么策略，能够提高测试效益就是好策略。

topo1



topo2

自动化测试平台化策略 之迭代式开发策略

自动化测试平台设计策略
之迭代式开发策略

序言：一般对于一个自动化测试平台而言，需要包含的模块和服务太多，特别是要做一个分布式的平台，那么其涉及的平台模块及交互信息则是非常庞大，所以在开发流程上的把握则特别重要，选择一种合适的开发流程，对于把握整个测试平台的进度以及提高自动化测试的有效性是很有帮助的。

一、迭代式开发的介绍

所谓的迭代式开发，与瀑布式开发方式不同，在软件开发的早期阶段就想完全、准确的捕获用户的需求几乎是不可能的。实际上，经常遇到的问题是需求在整个软件开发工程中经常会改变。迭代式开发允许在每次迭代过程中需求可能有变化，通过不断细化来加深对问题的理解。迭代式开发不仅可以降低项目的风险，而且每个迭代过程都可以执行版本结束，从而更好的发现问题。

二、自动化测试平台开发流程的关键点

1、自动化测试平台是一个非常大的项目，我们的一个难点在于自动化测试的变化性，所以平台的应用发布的周期不能太长，即公司部门等到一个庞大的“完美”的测试平台的出现，其成本耗费量太大，大多数公司等不起。

2、 自动化测试平台的直接客户是测试人员，其客户需求获得容易，所以，也意味着必须要做到拥抱需求的变化性。

3、 自动化

三、自动化测试迭代开发策略

根据其关键点，大概的自动化测试迭代策略如下：

1、 首先根据测试部门情况，制定一个大的测试平台计划，即需要实现的大致模型以及对需求的汇总。

2、 然后，根据其需求，定义一个大的模型，然后根据总体模型，设计出自动化测试平台包含的各个模块。

3、 然后撰写各个模块的概要设计，特别是对模块之间接口的定义要清楚，统一接口形式或者消息格式。

4、 之后，根据总体模型，进行迭代的阶段划分，将总体模型抽象出一个基本简单的模型，定义好次模型实现的基本功能，然后把握好其实现进度，并且控制好集成阶段。

5、 等第一个版本出来后，就可以用来给测试人员试用，在试用的同时，一边收集需求，一边进行第二次迭代，在第一个版本的测试人员使用需求上的分析的同时，将其他的功能开始开发。这样，迭代的好处就既能保证自动化测试的及时应用上，也能同时保证需求的有效收集。

注意的是：

1、 不管迭代到哪个版本，其平台实现的流程是一样的，也就是说，各模块的交互流程形式都是一样的。

2、 第一个迭代版本，一定要将其平台的基本功能包含进来。

例：一个大型的分布式的平台，包含功能为：用户权限管理、测试任务管理、环境部署管理、测试执行管理等，包含模块为：界面控制模块、测试任务分配模块、测试执行模块，环境部署管理模块等，那么在第一次迭代时，可以先以最简单且最基本的方式实现，第一个版本可以在包括：界面控制模块、测试任务管理以及测试执行模块且可以在本地实现，但是实现流程是按完整的运作流程，而第二版本则可以添加环境部署管理模块，第三个版本则可具体到分布式以及用户权限，而每一次的版本，都会发布给测试人员使用，并进行需求收集和改动，而改动的需求则会尽快添加到下一个版本中。所以，在版本迭代之前，模块之间的基本运作流程还是要先定义清楚了的，也就是说弄清楚了要做的东西，只是将做的东西分阶段进行。

总结：个人以为：不管做什么事，都需要根据不同的情况和场景进行分析，然后采取不同的策略，不管小事或者大事，细节决定很多，在实际项目中，才发现，细节往往真能决定成败，一个好的耐心是很重要的，也许坚持这么一小步就好。

自动化测试平台设计开发心得

自动化测试平台策略

之自动化测试平台设计开发心得

序言：负责自动化测试平台产品的开发已经有好一段时间了，从开始着手需求分析到概要设计、然后到原型的开发只至到现在出了第一个 beta 版本。月底要发布正式版，趁着现在平台测试基本稳定之时，总结一下设计过程吧。

一、平台的设计需求来源

什么时候需要平台？当自动化测试发展到一定规模后，即根据不同的测试需求，有了一些零散的自动化测试框架（包含脚本）、一些零散的自动化工具（满足某些特性功能的测试）。测试人员在利用这些脚本和工具提高了一些效率，但是却没有一个统一的平台将这些框架和工具集成起来，按照一定程序去进行测试。

例如：QQ 是一个通讯工具、QQ 空间是一个 SNS、QQ 游戏是网络游戏，这些单独分开的娱乐方式，腾讯公司提供了一个平台，即接口，将这些方式进行了整合，你打开 QQ，能通话之外，还能去访问 QQ 空间和游戏，而联系这些的就是 QQ 账号。

所以，创建一个自动化测试平台的目的，就是使这些工具和功能框架无缝链接，由测试人员按照各自需求，去定制自己的测试方式，而联系这些的就是测试用户账号，而账号联系的就是各个定制的测试任务。

总之，**平台只是一个空架子**，例如：腾讯门户没用 QQ 用户，再多的功能需求也没法，所以必须**测试需求到达到一定程度，平台才是有需要的。**

二、平台的设计开发过程

平台的整个设计大致过程为：

1、进行**平台方案的讨论**，务必要确认平台的设计定位，我们是定位为分布式的自动化测试平台，然后划分模块（或者称为服务），我们这里划分为控制端、任务管理端、脚本管理端、执行端、拓扑管理端、设备管理端等，并且确定好模块间的通信方式。

2、之后，进行**平台模块的概要设计撰写**，这里务必要做好的是定义好各个模块的具体接口消息。

3、之后，进行**具体开发**，首先，按照定义好的迭代式设计过程，先快速开发出一个原型，能够满足基本功能，在原型的基础上，根据所需要的测试方式去迭

代开发；这种迭代开发过程中，特别注意的是程序的注释必须清楚，否则对维护和交接带来巨大的工作量。

4、当平台迭代到一定需求之后，则可以发布 beta 试用版，在**小范围试用**。必须注意：测试人员的耐心是有限的，而且对一个产品问题特别敏感，所以要尽量的保证测试平台的稳定性。

5、之后，**发布平台正式版**，需要注意的是平台要提供接口方便以后测试工具和功能框架的集成。

三、平台的设计开发心得

1、整个平台设计开发比预计划节约了大量时间，可以说是与**整个团队的紧密合作**是分不开的，一个良好的团队不论在做什么事上，都会是快速的。一个人的力量再强大，但其局限性不仅体现在个人思路上，更是体现在心灵上。一个人遇上挫折容易放弃，但是一个互相依靠的团队能够通过鼓励，快速排除困难；而且一个团队不是人越多越好，而是有一个平衡，通过这次平台设计开发，真的是对团队的合作有了深刻的认识。

2、 做产品一定要**保持谦虚严谨的态度**，我之前在这点上没做好，容易陷入自己的思路，记得，刚开始，对于我刚开发出来平台的主界面，上级指出其界面布局的不合理性，但是我却从代码的角度上考虑，这样更改太过于麻烦，不予理会，结果到了后面，询问几个测试人员后，都觉得这个主界面带有迷惑性，不够简便，只好进行了更改，结果却导致更加麻烦，所以，在听到不同意见后，一定要严谨考虑，进行多人询问和调研，这样有助于扩展思路。

3、 在**平台的设计过程中，一定要紧密结合需求**，每隔一段时间，要讨论现在的平台设计方向是否和公司需求脱轨，大多时候测试开发人员对测试需求了解不够深，容易在设计开发平台模块时陷入自己的思路，从而脱离需求，所以每天短暂的交流和一段时间的讨论是必须的。

总结：平台开发完后，将要回到部门继续负责产品线的自动化测试工作，那时会以需求、流程和应用为主，**其平台到底是一个空架子，如何用上还得不断调研和分析**，路途遥远，共勉之。

自动化测试是一种累赘吗？

自动化测试平台策略

之自动化测试是一种累赘吗？

序言：平台的第一版开发工作已经完成，昨天已经正式将其打包了。打完包后，与原部门某测试组长聊天后，发现原来现在的自动化测试工作确实还存在很大的问题，不少测试人员抱怨随着自动化测试规模化，自动化测试工作越来越繁重，还不如手工测试，其实以前去开发平台之前就已经预料到这样的问题，只是当时也不是特别突显，但是随着自动化测试规模的变大，自动化测试反而成了一种累赘，看样子回去之后，自动化测试的引导工作还有很长一段路，做了这么一段时间的自动化，才发现自己原来是转了个圈，思想和认识上重新回到了原点。

一、自动化测试成本

为什么自动化越做越大，反而却带来了更大的成本，根据目前的状况，我分析了下，自动化测试课程带来“累赘”的某些因素：

1、 测试人员在本机上安装自动化测试运行环境的成本
2、 测试人员掌握自动化测试脚本或者工具使用的成本
3、 测试人员运行脚本的成本
4、 测试人员部署自动化测试环境的成本
5、 测试人员分析测试结果的成本
6、 测试人员维护测试脚本的成本
7、 自动化测试结果的输出成本

以上几个方面，在刚开展自动化测试时不会突显，但一旦规模化，则会将自动化测试做成一种包袱，所以，必要的时候，一定要精简，这样才能在自动化测试上越走越远，越走越轻松吧。

二、自动化测试的问题分析和解决

那么根据以上的原因，对照着分析现在状况存在的问题进行分析和解决

1、 对于安装测试软件造成的成本，可以搭建执行服务器，采用 C/S 或者/S 架构的形式，将执行软件端和测试代码都放在一台或者多台机器上，某个小组或者部门共用，测试人员需要下发脚本时，只需利用控制端进行测试任务下发即可。

2、最大化的将测试工具集成在平台中，让测试人员尽量关心只是测试的输入和结果的返回（输出）。例如：测试人员如果想测试断电稳定性测试，则只需从平台库中提取，下发测试任务，返回结果，其关心的只是测试的目的、测试的分析，而不需太关心测试的手段。

3、 搭建固定的自动化测试环境，一般电信行业会搭建一个大型的拓扑图，而对于软件行业可以搭建测试集群，应用拓扑管理或者集群管理，且集成版本自动升级，。

4、测试结果分析的成本主要来源测试脚本的过于冗余，一般测试脚本按模块来划分，往往一个测试脚本几十行，这样造成测试人员分析结果困难。例如：二十行的脚本如果是第十九行错误，则其需要根据前面十九行来进行判断。这样，很多时候测试人员需要将前面的步骤重新操作一遍来定位问题。

所以，在此，测试脚本需要细化到测试点，例如：一个登录模块，维护量大的方式：是写成“登录—各种输入判断—登出”。而我觉得应该细分为“登录—输入正确—登出”和“登录—输入错误——登出”，当然，这里只是简单举例，测试点越细分，则越容易定位脚本问题，也越容易维护。

5、 维护测试脚本方面，可以集成断点调试功能，即测试人员可以采用平台脚本调试器给相应脚本添加断点，无需测试人员手动进行错误步骤前的配置。

6、 自动化测试结果需要加入分析统计功能，而不是单纯的一份结果。这样在技术上虽然实现成本高了，但是在应用效果上反而降低了成本。

三、自动化测试平台定位

基于以上，我对以后自动化测试平台的定位主要为两个方面：

1、资源整合，利用平台尽可能将测试资源进行整合。

2、辅助性，利用平台尽可能辅助测试人员开展测试。

总之，个人觉得，自动化测试的精简之道就是“辅助”和“实用”，最大限度的辅助节约测试人员的时间和精力，最大化的融于平时的测试中，这样，自动化测试才不是一个绩效的数据，不是一种包袱，而是真正帮助提高测试水平和规模的利器。

这也许也和我们的学习过程息息相关，我们的学习路上，需要不断的反省、总结和精简，这样，才不会导致越学越多，反而越学越累的情况发生。

自动化拓扑管理设计

自动化测试平台策略

之平台拓扑管理设计

序言：在设计开发的自动化测试平台中，按服务功能模块划分，有一系列模块：包括控制端、任务管理服务、执行端管理服务、拓扑管理服务、脚本管理服务、结果管理服务等。在此先说说拓扑管理。希望大家有不同意见提出。

一、拓扑管理概述

在电信业界中，要规模化实现自动化测试，搭建一个实用性的自动化测试平台，那么需要对应一个规模型的测试实验室（TestLab），其实验室重要部分则由测试拓扑构成，那么测试拓扑的管理则是测试平台中的关键：

1、 每个脚本都对应一个脚本拓扑。

2、 测试实验室由若干个 TestBed 组成，每个 TestBed 都对应各自产品线的大型拓扑，大型拓扑中包含了脚本拓扑所需的拓扑。

3、 脚本运行时，根据标准拓扑获取到综合拓扑中的各个设备参数信息，并传送到脚本中。

4、其脚本拓扑描述的是抽象的连接关系。大型真实拓扑描述的是实际的连接关系。其描述可以通过 XML 文件描述。

二、拓扑管理应用

1、 可以保证拓扑资源动态分配

2、可以将环境的更改提取出来，直接抽象成一个拓扑，而无需具体到每个脚本，大大提高了维护性。

3、 测试人员无需对脚本配置负责，而只需对拓扑变更负责。

三、拓扑管理设计

1、 前端界面拓扑设计模块

前端界面拓扑设计模块提供给测试人员进行拓扑图的设计，我在此采用的是 java 中 jgraph 开源包进行设计，采用拖拽式放置控件，并且每个节点（GraphCell）属性设置界面，并能将这些信息和后台数据文件关联。界面如下图所示

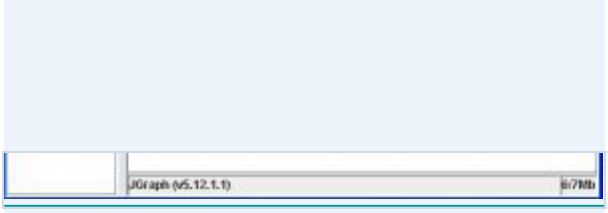
2、 中间持续层模块

中间持续层，即将前端界面的拓扑图与每个节点的属性能够导出为 XML 文件，并且按一定格式保存，并且可以导入反应到图形界面。

3、 后端拓扑映射算法模块

后台拓扑算法则是对后台文件的处理，根据脚本生成的脚本拓扑来与生成的大型拓扑进行匹配，主要匹配设备类型和连接方式，来获取符合条件的配置参数。从而部署好脚本运行的测试环境。

总结：软件设计的最高抽象度就是让用户人员（测试人员）面对的是实际模拟的环境，让他们对自己所感兴趣和所擅长的东西负责，而不是一堆一堆生硬和陌生的脚本，而那些脚本的管理则可以交给后台擅长的人所负责，各有有长，各使其责。下一节将简单说说其余的服务。



topo

自动化测试与项目的结合之路

自动化测试平台策略

之 自动化测试与项目的结合之路

序言：测试开发部门做自动化测试很容易脱离测试项目去做，仅仅为了做自动化测试而做。开发部门做自动化测试很容易考虑不够全面，但是在自动化测试开发、维护性和复用性以及可测性上结合比较好。测试部门去做自动化测试，对测试项目理解比较深，对自动化测试的应用也比较能理解，但是却无法开发应用比较好的自动化测试系统与框架为之服务，自动化测试只是作为测试的一种辅助手段，而每个人都术业有专攻，**我们需要的是理性认识到自动化测试，不能本末倒置，以项目为驱动，将自动化测试与项目结合，真正将自动化测试在整个项目环节中应用起来。**

一、自动化测试开展在整个项目中存在的一些问题

上一年度，总结和与各个产品和维护测试组的技术组长进行面对面交流，总结出来在项目测试中，自动化测试存在的几个问题和一些他们的想法。

1、**产品测试组认为自动化测试开发效率过慢，**新的产品开始进行测试时开始开发自动化测试脚本，赶不上测试项目的进度，有可能版本都迭代完了，自动化测试脚本还没有稳定下来，根本就指望不上。

2、维护测试组之前将自动化例行测试应用于维护测试项目中，之前**耗费大量精力开发了大量的模块**，后来却发现在版本迭代过程中，这些模块之中大多数都没有应用上，而开发关注的往往都是那些非常基本的功能模块。

3、**产品测试组和维护测试组之前没有配合**，各个组开发的业务测试脚本互相不知道，也很大程度导致了重复开发，并且开发出来的脚本不具有很好的迁移性，也间接到导致了自动化测试资源的浪费。

4、**自动化测试度量**不够量化，虽然年末也做了自动化测试指标度量，但是很多数据都是测试人员拍拍脑袋想出来，数据都不够量化，因此也不具有太多的说服力，而且在项目上的作用也不够突显。

二、自动化测试与项目结合之路

针对以上问题，如何改善，总结几点

1、**将自动化测试真正融于到测试项目中去的方法就是在项目前期就要开始考虑自动化测**试，而不是等到项目中或者项目后，这样就能保证自动化测试能够快速应用在测试项目中，具体方法如下：(当然，需要注意的是不推荐产品测试的前几个版本应用自动化测试)

1）分析测试项目周期，选择需求要自动化测试的模块

2）在项目前期加入自动化测试考虑和评审

3）在项目开始前，便开始对新的模块，基于测试点撰写关键字驱动测试方案，并生成关键字接口库，这里特别注意的是，在关键字驱动测试方案中要指明测试用例中还需要手工测试的比率和部分。

4）在测试用例完成的同时，基于测试用例组装开发业务脚本

5）在产品接口确定的同时，撰写关键字接口实现库，调试通过

6）自动化开始应用于项目，部署在测试系统上，实时统计与跟踪

7）项目完成，对项目中的自动化测试项目进行度量与分析(单个模块覆盖率、产品特性覆盖率、产品 CBB 率)

8）将项目完成的模块按需添加到例行测试环境

2、 **宁缺勿滥，精做与迭代式。**制定自动化测试模块策略，以相应的几个维度去分析此模块是否优先自动化，CBB>基本功能>产品特性

3、 **建立统一管理的自动化测试功能模块库**，各个测试组都能够通过此系统库查看到已经自动化测试覆盖的模块和应用的产品线，而且基于关键字测试驱动开发，统一标准，迁移性好，产品测试组在项目中开发后的模块可以快速迁移到维护测试组。

4、 **CBB，统一标准，统一度量**，手工测试用例和自动化测试用例都是基于一个模块测试点标准，方便细化和统计。

三、自动化测试平台之项目系统建设

基于以上想法，在自动化测试平台已实现如下:

1、 在测试平台上加入项目管理机制，基于项目建立自动化测试任务并运行。

2、 模块管理机制，自动化测试任务基于模型运行。

3、 Master-slave 机制，每个 slave 对应相应的测试模块。

4、结果分析和报表管理机制，能够基于产品线和项目维度去查看自动化测试指标，方便阶段性分析。

总结：有人说测试没有技术能力，但是我觉得测试很需要有技术能力，虽然测试与开发关注的角色不一样，前者关注于测试设计与质量保证，后者关注与软

件设计与编码，但是大家都是基于软件工程和项目管理，因此在基础上大家都是站在一个起点上，前者如果忽略基础，很容易变成纯手工或者过于追求具有技术含量的东西。后者如果忽略基础，很容易变成无意识的”码农”。所以，如何学好基础，就需要我们在工作中好好去学，多发现问题和本质，提出有效方法。

产品、市场与测试

测试，人人都是产品经理

之产品、市场与测试

序言：产品、市场与测试，大家看到这个题目，不知道是否会有些疑惑，他们之间的关系会是什么，在做自动化测试已经有这么一段时间了，在这段时间中，我深深体会到，不管自动化测试，还是类似的测试技术，如果真要能够快速占领公司的测试市场，技术底蕴是一个方面，产品和市场策略也很重要，这几个月，空闲的时间，都会在看产品管理以及数据分析的知识，然后基于两者去看待测试，似乎也颇有一些风趣。

前一段时间，我曾负责做过两个产品：一个是自动化测试平台、一个是自动化测试执行端。在做完前者后，我被调出负责部门自动化需求挖掘、推广和应用。也很好的有机会去使用之前的测试平台，在对其推广和应用过程，才逐渐发现平台的应用效果很差，最后测试人员在使用一段时间后，逐渐放弃，原因如下：

1、产品原因：平台功能强大，包括：执行界面、任务管理分配、拓扑管理、脚本管理、执行端管理，但是却显得非常复杂且没有重点，对于测试人员反而加重了熟悉和使用负担。并且测试平台需要

2、市场原因:没有对测试部门的整体现状好好分析，就急于开发平台，当时测试部门的情况是测试框架还不统一，测试脚本执行状况不是很良好并且测试覆盖率不足。可以理解为市场需求还未发展到那一步，这也许跟很多“创新型”公司正因为太过于创新，超脱于市场需求太大而死掉的原因类似吧。

3、当然，还有一些其他的原因，包括技术原因、人员原因，但是这些都是次要的，就不详说了。

因此，后来我在部门放弃了测试平台的使用，采用产品管理的手段重新做了一款自动化测试执行端，至今看来，效果还是不错的，至少测试人员都对它有所依赖了。

1、“市场状况分析”：首先将所有技术组的自动化测试状况收集分析，发现现在他们的测试脚本都存在各种问题，每个技术组都有一个单独的测试框架，因此，我先引入软件配置管理（svn）,将测试框架和测试脚本进行统一，如果移植工作量太大，则继续按前方式进行，但是先花力气将测试脚本稳定，尽量让脚本可以单独运行无误起来。

心得：一个产品的崛起，需要的是市场某个阶段的稳定。基础不牢，一味的追求吹嘘上层应用，那么泡沫就这样产生了。

2、**“市场需求调研”：**对几个小组的自动化测试接口人进行调研，发现他们以前用平台的时候，觉得最方便的还是在于执行部份，所以依托了于这样的需求，我好好写了一份自动化测试执行端的需求分析报告，包括：需求点、功能点以及使用分析（解决的问题）。然后，在设计上，采用了可拓展似的 tab，可以不断添加新的功能集合。

心得：一个好的产品就是踏踏实实的为需求服务的，并且是可拓展的，这个可拓展的原则就是基于一个思想即可。这个自动化测试执行端的理念就是：让自动化测试执行一条线（从导入—配置管理—执行—结果都能在界面上查看）

3、市场应用和反馈：之后，快速完成编码，这个过程就忽略吧，高科技营销魔法之父，摩尔有一本书叫：《公司进化论：伟大的企业如何持续创新》，其能告诉你：产品与对应的市场、用户都是有自己的生命轨迹的。一个产品生命周期里有五种用户群：创新者—早期追随者—早期主流用户—晚期主流用户—落伍者。而在每个测试技术小组，自动化测试接口人都是很好的早期追随者，他们因为强制性的要求参与自动化测试，所以他们的需求目的性强，需要迅速能够解决问题的测试产品，但是这里有一个问题在于他们有时候虽然对产品某个地方不满意，但也会将就着用，而不会反馈，所以就需要你的积极沟通了，只有这样，你才能将早期的追随者的价值发挥到最大，从而能够让你的产品继续推广。

心得：一定要区分用户群，发挥每个用户群的最大价值，要做一款细致的产品，那么一定不要用无所谓的心态，而是要把每一个点都做好。现在的产品层出不穷，特别是互联网产品，也许你因为某一秒的延时就错失了很多用户，从而让你的产品的用户群无法从早期追随者跨越到早期的主流用户。。

总结：也许很多测试人员总觉得上面两个产品与自动化测试的从业人员或者测试开发人员相关，与做手工测试的没有什么关系，但我个人觉得不然，**如果我们在测试这条路上走行进的时候，就像做产品和市场一样，能够更多的想想**

我的测试需求目的是什么，我的核心测试策略在哪？我的测试能够带来多大的价值？也许会更有收获呢，共勉之。

之后，我想总结一下：**数据、分析和测试**，看看数据能给测试带来什么？

自动化测试需求的奋斗史

测试，人人都是产品经理

之自动化测试**需求的奋斗史**

序言：一直觉得现在作自动化测试更像是经营一个“产品”，从需求的收集、分析，自动化测试设计、自动化测试推广，测试用户的使用和反馈以及自动化测试的改进等都需要来参与，有时候，你需要提从测试用户的想法，收集测试需求，有时候，你有了一项新的自动化测试技术或者产品，需要引领测试用户的改变。有时候，你的自动化测试产品设计的功能再强大，却因为其应用复杂性而热门一时却最终流于无人应用的下场。有时候，一个简单的自动化测试工具，却能够被测试人员欣然接受。为什么会出现这样的事情，都是值得我们进行深刻思考的，所以，今天谈谈做“产品”，也是做自动化测试的第一步：自动化测试需求的奋斗史。

一、产品需求的奋斗历程

产品需求可以来源于用户调研，也可以来源于产品经理自身对用户需求的理解。像前者的典型代表就是史玉柱，他就是一个热衷于用户需求调研的人，从脑白金到黄金搭档，从征途到征途 2 等商业游戏，他都会查看市场反应，而且更是亲身参与到征途游戏中，与游戏者一起体验，而后者典型的代表就是乔布斯，其的名言“从不搞市场调研，只相信自己”在业界津津乐道。而从客观上讲，不管哪种方式，他们都成功抓住了用户的需求和体验。

一个需求的奋斗历程：大概会经过，需求收集—需求分析—需求筛选。

1、需求收集则主要是对用户的研究，形式上包括定性和定量收集

定性即从自身或者市场角度感性认识用户的需求和心理，而定量，则尽量用量化数据说话。收集方式上包括说与做，说，即以与用户交流的形式，定性的说，可以用访谈的形式，定量的说，则可以用问卷调查等形式。做，即用实际行动来进行，定性的做可以采用可用性测性，即开发一个原型让用户用着，然后反馈。定量的做可以采用数据分析的方式，即将量化的指标采用数据的方式展示，尽量让数据指标变得具有可预测性。

2、需求分析，即将定性化和定量化收集到的数据进行分析，得出结论

需要明确这种需求解决的问题以及实现

需要明确这种需求带来的商业价值，可以先做方向性的假设，然后提取相应的数据分析概念图，找出现象，然后尝试着解释和预测方向和价值，最后通过用户调研修正方向和商业价值。

3、需求筛选

需求有价值，也需要进行筛选，即进行性价比分析，即根据现有条件，得出一个商业价值和实现难得的一个分析，对需求进行剪枝，马云说过一句话：少做即多做。有时候，把一个东西做精了往往比做全了更有意义。

三、自动化需求的奋斗心得

对于我们自动化测试工程师而言，一定要感谢现在的机会，因为，将我们的测试部门比做一个市场的话，我们有一批忠诚的用户，我们能够参与市场，也能服务于市场，我们是一线市场人员，也是幕后的设计开发者，我们是市场的驱动者，也是市场的观察者。

我们一定怀着做一款好产品和做一个好服务的心态去开发和服务这个市场。

1、需求收集，我们收集测试用户的需求，主要是通过测试用户反馈、测试用户的访谈等形式实现， 主要是从定性上说和做，而没有定量的说和做。所以，不能很好的把握好自动化测试的发展方向。

测试用户的访谈，我觉得自动化测试人员，包括我，很容易犯的一个毛病就是不够倾听，很容易站在设计开发的角度上思考，而不是使用者角度，例如：我以前老犯的一个错误就是，我就觉得，测试用户用习惯了就好了。访谈时老说的一句话就是：你不觉得这样设计非常好吗？而总想去引导别人接受自己的想法，所以，现在的我们需要改进的就是认真倾听，对测试用户 每一句话进行斟酌。**务必要做到：需要认真倾听用户，但别照本宣科。**

缺乏定量需求方式的收集，个人觉得，要做好自动化测试，也要掌握好数据分析的技巧。能够从大量的测试用例中提炼可自动化的需求，能够从大量的测试结果中提炼自动化测试的发展需求，像自动化测试中的测试覆盖率统计就是一种数据统计的方法，所以，自动化测试的发展还是需要从定性发展到定量的。

2、需求分析。对需求收集分析尤为重要，可以参照产品理念中的需求分析方法来进行，当然在其基础上也有根据条件形成自己的分析风格。

测试用户的需求反馈是需要值得好好分析的，而不是拿到一个需求就开始实现，需要从需求角度分析，其实现的难度和价值是否合算，其是否能够将一

个特性需求拓展为一个共性需求，为什么会有这种需求（需求分析三要素：why、what、how）,尽量的提炼多的信息，将其最大化利用。

可以的话，为每一个需求做好需求分析报告，你的自动化测试是向上级汇报的，需要用最通俗的话和图表来表达需求的形式和发展方向。例如：你开展 web 自动化测试，那么第一步你需要找好自动化测试工具，是开源的 selenium 还是商业化的 QTP，那么你不仅需要从使用设计角度分析，还需要从价格和公司角度上分析，需要分析报告不需要大量的说技术（不知是否这是技术人员的通病吧，我以前写过一个自动化的需求报告，结果写成了概要设计类似的文档），而是性价比的分析和发展趋势的分析。

3、需求筛选。不是每一个需求都需要做的，做自动化尤其这样，一定要结合现在的条件进行，例如：自动化测试的覆盖率还没达到一定程度，你就要盲目的去做平台的需求，那么最终会顾此失彼的。所以，少做即多做，务必将现今最重要的需求实现好，然后在预言下一个需求的发展和实现。即在稳定中前瞻未来。

4、在使用自动化测试工具或者软件的时候，我往往在想，在中国测试市场，有多少测试软件能够给各个公司提升效率，从而带来效益，你随意扔出一个测试工具，是很强大，可是仅仅强大在其功能上，可是现在的境况是我们对自动化测试具体带来的价值体系都没有清晰化，那么现今阶段，测试工具仅仅只能带来的是更多的探索，一个公司能够支撑这个成本，那么它会投入探索，如果支持不下去，那么就会渐渐荒废罢了。所以，自动化测试为什么还没有发展起来，个人觉得，是因为没有一个真正的自动化测试的运转体系和盈利体系的形成吧。说简单点，就是自动化测试的需求利益转化不够明确吧。

总结：无论做产品、做测试还是做自动化测试，我们都要用一颗做好产品的心去对待，那么前提就是根据过去自己的做法，对自己的心理进行多次的访谈，对自己做好定性和定量的分析，然后从数据中提炼出自己的性格和想法趋势，找准自我需求，找到适合自己的做法，这样，才不会在信息的浪潮中迷失吧。

沉淀后，看自动化测试发展

序言：今天，才发现一些自动化测试思想只有待到真正的进行应用之后，才能从根本上理解，做了长时间的自动化，每天分析总结后才发现一些东西慢慢的才开始理清楚，沉淀后，才发现做这个是一个从简单到复杂，再从复杂到简单

的一个过程，对比以前写过的一些自动化测试框架和设计思想，发现原来很多东西都是可以更简单来表达的。

一、自动化测试发展建设

脚本发展阶段：

1、 线性脚本，即，现在工具纯录制产生的脚本，从上到下，没有逻辑判断，代码量大，维护困难。

2、 结构化的线性脚本，基于线性脚本上，增加了一些逻辑语句，可以循环操作或者判断操作，提供了一些代码效率，但还是维护量大。

3、 基于配置文件的脚本，在以上脚本的前提下，增加了共用的配置脚本，将一些变化量大的参数提取共享，降低了一些维护量。

4、 数据驱动脚本，即基于以上脚本，将可变数据流提取出来，测试逻辑不变，这样可以不同数据驱动测试一些测试功能点相同的测试，一定程度上提高维护性，降低了脚本数量。其实也就是结构化线性脚本的发展，只是将数据变量用数据库概念替代了。

5、 关键字驱动脚本，即将一些共性的测试功能点提取成为关键字，然后测试脚本共享关键字，这种方式也是从复用性上考虑。

框架产生阶段：

6、 等脚本发展到一定阶段后，测试框架产生的作用就是能够帮助统一管理这些封装库，快速帮助生成脚本以及对脚本结果生成和管理的方式。

平台产生阶段：

7、 等框架发展到一定阶段后，测试平台产生的作用就是从整体上把控测试流程，彻底将自动化测试作为一个公司的软件产品进行运营。

测试设计阶段：

8、等自动化测试应用到一定规模后，进而引导测试设计的发展，因为业界的一个测试理论“杀虫剂”理论，即产品会对自动化测试产生抗体，**事先设计好的测试不会起到真正改善产品质量的效果，最多也就是保证产品质量**，因此引入模型驱动脚本，即基于测试场景的封装，然后用图的思想去自动组装、部署各个场景，驱动不同测试。

二、自动化测试发展沉淀

从有效性来看：

自动化测试负担运行—自动化测试有效运行—自动化测试保证产品质量—自动化测试改善产品质量

从产品角度来看：

测试附属—测试产品—测试设计

类比个人的发展

刚进公司，你是附属公司的发展——之后，你辅助公司的发展——最后，你引导公司的理念发展。当然，一般做到第二个阶段已经不错，往上走很难，而自动化测试亦是一样。

总结：每个公司自动化测试脚本发展都会经历以上的过程，只是时间长短而已，**只有一个量变导致了质变，才能引发下一个过程的发生**，因此，做自动化不能急，但也要有前瞻性，保证一个阶段有一个阶段的效果。到最后，**沉淀下来的也许越少越好，大道至简。**

测试，人人都是产品经理

测试，人人都是产品经理

之测试产品的选择和创造

序言：明天新的一年的的工作开始了，在晚上写这篇文章，也算是对自己一年工作的一个简单的总结以及对今年所想做的事情作为一个开端吧。这次回家，疯狂了一把，不管测试、不管自动化、也不管技术，只知道与朋友们欢畅，踏上回来的途中，却反射性的重新拿起了书。每个人也许想知道自己的价值在哪，无论在哪，我觉得每个人都是自己的产品经理，而定位自己的需求，寻找产品的价值都是一件很难的事情，首先知道自己要什么，再知道自己可以设计出来？最后还要经过反复的实践和测试，才能诞生出一个让自己感到稍微满意的产品，因为这些文章，大多数看到的人是测试人员，所以将测试作为一款产品的角度上说说吧—测试，人人都是产品经理。

一、测试产品的选择之路

刚开始，我也做手工测试，在手工测试的同时，我还记得为了锻炼自己测试的速度，曾经一次性练习插拔机箱产品单盘几百次。那时候的我，说心里不躁动是不可能的，除了在夜晚打电话给家里发发牢骚，也许是别无选择而我却又是一个懒惰的人，所以没有放弃测试却让我走上了在公司从事自动化测试的道路。从事测试的人也许觉得测试是一件枯燥而不能快速提升自己的职业，但是，根本问题从事测试是你的选择，你就要为这个负责。我其实一直是个浮躁的人，想我大学当年，因为选择太多，什么样的技能都去学习，但没一个学通的，但后来，我的代价就是没有过于精通的，好处就是是懂非懂了一堆，对现在的工

作学习还是帮助。所以，大学是可以重来的，工作了却不可重来，放弃需要慎重。

我们年轻的时候都如刚出山的“孙猴子”，浮躁和自大，而暂时的外界是束缚，如一段取经历程，一个头上紧箍咒，也许能让我们更加指明本心，从而变得意志有力。而我当年误打误撞的选择了系统测试，就如给我浮躁的心系上了紧箍咒，逼迫的我不得不静下心思考自己的出路，不得不踏实下来慢慢寻求方向。所以说，如果你现在选择了测试，如果你对这份职业不满的话，可以的话，请不要轻易放弃，而是借助这个机会，思考一下自己的方向，当年既然盲目的选择了测试，那么我们就不能继续茫然了。

二、测试产品的创造之路

当我们认定了测试作为我们的产品之后，我们的产品经理路径就开始了。

所谓的产品经理，大多数人都应该了解，一个公司的组织包括销售、研发、测试、服务。而研发体系中有项目经理，而产品经理却是贯穿与全部体系的。一款产品的诞生到一款产品生命周期的结束，必然都会有相应的产品经理参与其中。而我们，在选择了测试这个产品之后，我们就必须开始为这个产品负责了。

业界一句话“产品经理应该以创造用户价值为使命”，所以，一款产品的出现必然伴随着相应的价值导向

我曾经和一位研发的好友交谈，我突然有过这么一个问题“一个公司就如一个生产体系，一款产品就是一个流水线，我们都是流水线上的环节，当你编写出一个模块之后，你懂的只是你这个模块的功能，而当你将这个模块放到这个生产线中之后，你的模块也就是这个产品的零件，而这个产品到底从何而来，去往何处，产生了多大价值，我们都不得而知”。在这里，我不是说这是一个悲剧，但是我觉得却值得我们深思，像以前的福特创造的工厂流水线制度，让工厂的人们只知道机械式的干活，所以，我们在工作的同时，也去寻找自己的需求和价值，每天每年都有不同的进步，这样，我们所做的事情才会越来越清晰吧。

一款产品的概念指的是它要创造什么样的用户价值、满足用户那些方面的需求，而你选择测试这款产品，你需要满足自己怎样的价值，怎样的需求，有没有大体的概念，也许你是看重了测试行业的入门简单、工资发展比较快，这也是一个概念，但却不是一个长期的概念

一款产品的诞生伴随着概念出现、之后做的就是概念过滤，只有过滤的概念才是最核心的概念，每个产品最好的方式是一句话能表达出来的方式。

概念的过滤就是将一个产品这些零散的大概念集中起来，从可行执行性和可应用性上进行分析，过滤出一个核心的概念。例如：淘宝要做最大的网上交易门户，麦包包网则要做中国最大的时尚箱包在线商城；以前的游戏：《征途》的成功在于定位一个有钱人玩的游戏等。你看到百度，自然联想到中文第一搜索，看到腾讯，自然想到最大的即时通讯门户等。

所以，我们需要将测试产品集中起来考虑，我们做测试产品的核心在哪？是管理者方向—测试经理？是测试业务类的专家—产品工程师？还是设计类的专家—测试架构师等。你看到自己，最快能联想到什么呢?

当然，**业界有一句话“用户的历史行为比他们当前的意愿更有价值”，也就是说，有时候你自己所想出来的需求或者概念，其实并不完全代表你会此买单，因为你并没有亲自尝试，所以，办法一就是参考自己的历史行为；二就是在“行进中开火”，**只有在了解它，并经过一些低风险尝试，才能在内心有所明白，想是不够的，必须做，产品都是在做的过程中逐步完善，所以，我们做测试时，也许很多知识是不了解的，我们只有先找准一个方向，慢慢摸索。

一个产品经理最重要的财富是将概念具体表现出来的图纸和文档。

当你的核心概念提取了之后，你就需要将其落地，将其设计具体化，当然，一款产品技术如何实现不是产品经理所关心的，在这里，我就仅以产品经理的角度出发，我们需要的是告诉实现人员实现一款什么样的产品；我们做测试产品，我们需要我们达到一个具体什么样，成为这样的角色，需要会什么，伴随着怎么样的功能和性能。而实现的技术过程，这就是在以后项目经理的安排了。而作为产品经理，所记录的最重要的就是需求文档和设计概要。后期设计的变化，也要及时更新这些文档。

实现过程中，一个产品经理做的最大工作就是沟通

当我们将概念具体化之后，接下来就是实现了，而在实现的过程中，我们要综合分析，实时拥抱变化，调控实现进度，所以，产品经理需要不断的沟通，前线和后线的沟通；而做测试这款产品，我们也要实时把握行业动态，实时更新自己的知识，实时的与自己打交道，不然很容易陷入不断学习，却学无所用的死循环。

当一款产品最终失败，他也有失败的价值

没有什么产品做出来一定成功，腾讯的 QQ 以前因为收购没有谈拢才有了现在的门户；百度的搜索因为 YAHOO 没有看上才有了现在的平台。**所以，当我们**

怀着一颗做产品的心出发时，即使失败，也会有很多经验在手。换换需求，我们重新可以开始。

后记：后面有机会会和大家交流一些做自动化测试产品的想法，自动化测试产品，与传统产品一样，只是面对的用户的内部测试人员罢了，曾研究产品设计一段时间，发现做自动化测试产品跟传统产品真的很多核心概念是一样的，为什么以前做的自动化测试没有很好推广，为什么以前做的自动化测试产品怨声载道，原来都是有根据可寻的，**错了不要紧，错了不总结就要紧了，对了不要紧，对了不总结更错了。所以，想写一个系列，从产品的角度出发，涵盖测试职业发展、自动化测试产品设计、测试体系等方向**

记二期测试交流总结心得

二期测试交流心得总结

序言：这次，我们北京测试技术圈子里面的人进行了第二期的测试技术和发展交流，地点在百度大厦，主要是纯测试技术的交流。然后会针对一些话题讨论，之后便是一段时间的心得和问题讨论解决。现在将交流的内容和心得做了一个简单的总结，分享给大家，希望大家能有所收获，也希望对测试、对技术、对创造和分享有同样热爱的人能加入到我们圈子里来。

一、主题交流

1、 路径覆盖率—思寒

首先，如何评估测试充分

1） 需求管理

2） 测试用例管理

3） 代码覆盖率统计，而在代码覆盖率统计中包含行、分支、条件、路径以及返回值覆盖率统计。

心得：他这里主要阐述了如何保证测试的充分性和有效性，他指出了在黑盒测试中，往往不能很好的评估测试的覆盖充分性，因此需要利用白盒测试让测试变得更有信心。

之后，他讲述了一下他设计开发的工具，我理解的原理如下：

1） 通过抓取的线上数据应用此工具进行回放

2） 回放后，生成思维导图，能够有效反应代码的运行路径

3） 最后能够统计出相同路径包含哪些测试数据，具有可统计的作用

4） 最终生成 EXCEL 表格，即自动生成了用例，可用于测试人员手动输入数据进行测试验证。从而保证测试的充分性。

心得：让我明白了白盒测试的另外一种方法，真是增加了见识，不过不知道我的以上理解对不对，所以，以后明白了线上数据和用户的数据都是最真实的数据，都是很重要的。

2、 Web 测试架构—jameshu

我只简单记录了大纲，后面有想更详细的了解请问 jameshu

1）、 测试工具

Jenkins(CI，这个我简单用过，之前叫 husdon)

Selenium

HTTP 协议测试（服务器端返回）

Jmeter(性能测试)

Jasmine(是不是写错了，没查到是什么工具，说是用于 JS 单元测试的)

2）、 后端服务监控

HeartBeat(服务主动发送 Heartbeat 包)

Topology Graph(拓扑的自动发现，怎么个发现法，这里我很想知道啊)

Graphine(服务器端指标监控)

Exception Email(短信提醒)

3）、 前端服务监控

Beacon(用户行为跟踪、第三方验证、性能监控指标)

可访问性(来自不同地域的请求:GOMEZ; PhantomJs:分析页面资源加载时间和无浏览器访问）

心得：后面还有具体的测试内容，不过都没来得及记下，这些都是 jameshu 和他的同事们 3 个人在两年时间内搞起来的，心中只有一个感受：都是牛人啊

二、话题讨论

1、个人发展讨论

Jameshu：要先将个人发展和公司发展相结合，在公司，一定要解决大家觉得是问题的问题，选择方向则要看好公司的关注。

MyDream 提出问题：如何从一个 公司跳到另外一个公司，还能保持技能的持久性？

思寒回答：1、软实力。2、行业资历（业务层面）。3、技术功底（要先过来技术关，数据结构、算法、测试思维等）

Jameshu：以前微软面试 QA 的时候，在两轮会抽开发来面试，所以技术功底还是很重要的。

John:每个公司在每个特定的时刻需要的人都是不一样的，所以找准自己的方向就好。（john，我这样理解可以吧，哈哈）

当然， 还有一些更多的讨论，我就不一一列出来了。

2、持续集成

1）GIT

2）单测作为指标通过后，才能继续提交到自动化测试

3）单测可以提高系统设计和可测试性

4）场景化和 BDD

…

三、尾声

最后，大家一起在辉煌国际吃了家常饭，哈，由于我选的位置不佳，所以让大家享受了一次“桑拿浴”，又没空调…又一堆锅子和烤鱼，幸好最好空调还是开了，抱歉了各位啊，期待下次的交流啊~

也谢谢刘哥最后不辞辛苦的赶过来，赞一个~他现在很希望招到合适的自动化测试人员，希望能有不错的人能来呀。

下次的交流，争取大家在交流的同时，能把这段时间积累的问题带上，大家一起讨论解决~

谢谢朋友们，测试的路上有了你们更美好~

记三期测试交流活动心得

序言：10.20 那一天，我们一些测试技术方面的朋友总算抽出时间进行了第三期的内部测试技术交流，本来想写一个总结，后来有参加的朋友已经写了，我就乐得偷一下懒，就从他的博客中的交流活动心得直接转了过来吧。希望没能一起参加交流的朋友，也能从他的心得中有所收获。

记 10 月 20 号贝塔咖啡的测试活动

周六，晚上有同学聚会，上午睡觉，下午就计划去参加一下 sun 他们组织的测试活动，事实上，挺有收获，在这里做一个总结，分享一下我的见解.

一、BDD 到底如何?

作为公司推动技术改革的 MIC 同学，为我们分享了公司的实践.

用一种新的方式肯定是发现了原有模式的问题，我听到的 MIC 同学讲的是，看到测试用例与需求，与自动化之间的脱钩，导致重复的工作，自动化的难于实施. 而 BDD 是解决这一问题的方法.

BDD 以 cucumber 为基础，完成相关特性(用例)的编写，利用二次编码进行自动化测试，根据 MIC 的分享，效果较为突出，这也为我们尝试改进提供了很好的借鉴意义和相关思路.

这个过程关键环节我认为有:

如何保证 BDD 的基础上，又让用例更为灵活.

显然，MIC 采用了后者，通过 Excel 的整合，与二次编码(想尽办法让需求人员或测试人员避免写复杂的正则）简化这个过程，让更懂自动化的人去完成编码细节.

如何推动测试人员在原有的习惯上完成这一转变?(其实对他们是变得稍不灵活了)

目前的办法是强推，习惯 3-4 天就好了.

不关开发们的事么?

MIC 分享说还没有去推动，我认为这点会导致效果较大的打折扣. 不过就整个情况来看，虽然没做这个，效果还是达成的.

我的个人建议:

如果你也存在这样的问题,需要注意:

你必须能有很强的话语权，否则难于推动.

流程规范远远比你的想法更重要，也就是落实度要强.

BDD 自动化框架要经过充足的预研和改进，以更灵活的方式让所有人员接收这个观点.

推进到各个环节，仅仅是测试是不够的.

二、QA 的下一步发展?

二、思寒 给我们的观点确实蛮新颖，不过仔细想想以前也有过类似的思路点，不过不够系统.

其主要观点是建立于 bug 容忍度高的基础上，我们可以将更多的测试内容交给直接用户真实使用和测试. 而 QA 则向前和向后各努力一把，将单元测试，**接口测试**，静态测试做的更出色；向前建立有效的监控,反馈系统,甚至于自动报表系统，让我们的工作重点更多放在**容错**,**发散**,**性能**等更需要思考的环节.

我的建议:

不是所有的公司都是这种方式，尤其是 B2B 模式的公司，我们的重点仍然在功能测试.
这个过程对 QA 开发能力要求很高，又需要调度各种资源来实施前端的用户反馈收集，不适用于目前大多数的 QA 团队.
如果你们的 bug 容忍度也较高，不妨更多利用敏捷改善. 如何敏捷,这是接下来的话题.

三、敏捷项目的实施?

对这个话题是非常感兴趣，Tom 讲的也非常出色，活跃了整个气氛，讲的很带感:)
实施一个成功敏捷项目，需要几个必备条件:
真正的持续集成自动化运转
需求可控
RD 质量意识到位
所以，Tom 一开始项目的悲剧不可避免地上演了，没有自动化的敏捷可以说是一个笑话.
没有准确的需求也就是像一个大楼从来不知道要盖成什么？要是一不小心又成为 苏州的"裤叉"，难免被用户们批死...
我的个人建议:
这个工作不是 QA 一个人能搞定的，想开始的时候要看好第 3 点，选择优秀的 RD
QA 的工作依然是"保姆"，不过更加主动，控制需求(偏测试)，实现持续集成的系统建设,推动团队的自动化建设.
你的编码能力要足够，我更希望你会一些动态语言,比如 Ruby. 开展会快很多很多.

四、自动化的度量管理

SUN 今天准备的不多，干货是较少的，不过还是足以回想起我的公司进展，我也有点想吐嘈两点:
总是看效果的领导不是好领导
不敢于承担风险的领导也不是好领导
这两点说起来简单做起来难. 自动化的推进是同样的道理，过于关注 ROI 会适得其反，浪费无谓的工作. 在自动化建设初期，这一点更加明显. 与其关注这些，不如:

培养合适的人才在合适的位置上
建立更加合理的激励机制，并且根据不同时期进行调整
自底向上自发的推进，走的弯路会更少.

五、其他

后来的 selenium 封装我就没有听了，实在是时间有限，相信这个是纯干货.
大家给我的感觉是十分的自在，感觉是自己在这个组织很久了，谢谢 sun,sihan、tom 等人的组织. 无以为报，写一些听后感想留给大家回想一下吧.
对活动的建议是:
1. 频率可以高一点点，讲 slideshow 时间可以少安排一点
2. 让大家多自由交流
3. 地点其实不错，就是不实惠，建议固定一些，没有水不要紧，大家可以自备，其他吃的也可以让大家准备.

转自：http://yafeilee.me/blogs/5082ea921563880ac000000a#disqus_thread(亚飞的博客，偏技术些，对 Ruby 有兴趣的可以多看下.)
我们的一些评论：
@思寒：写的太好了。太赞了。
Mic 已经放我们好几次鸽子了，本来以为他很不靠谱，但是今天下午，他为了给大家补上测试用例。特地又折返了一次，真是敬业啊。让我彻底改变了看法，分享的内容非常的有价值。这也是业界少数能够成功而踏实的应用 BDD 并且取得不俗成绩的实践。
Tom 的讲解，激情四射，神采飞扬，勇于剖析自己曾经遇到挫折的项目，并能在后续积极的改进测试方法和测试手段，让公司的测试发展飞上一个新的台阶。tom 充满感情的分享，给我留下了深刻的印象。
John 的分享是最实在，最实用的，他介绍的 selenium 封装方法和问题场景，是现在大多数公司都遇到过的，sogou 成功的解决了这些难题，并打造成了出色的独居产品特色的 selenium 解决方案。从技术上，实践上，产品上，sogou 对测试的打磨真的是匠心独具。
这次我和 sun 的分享，只是给大家穿插一些测试发展的介绍。还有好几个非常好的资料和话题要跟大家交流，可惜时间有限，无法都展示了。
@散步的 SUN
我同意那说的两点，我说的那五个自动化测试过程：
1、初试

2、应用
3、定义
4、度量
5、管理
每个阶段有每个阶段的自动化测试策略，而我分享的度量分析，则是到了第 4 个阶段，在自动化初期就是先把事做好，初有成效，之后摊子大了，再回过头来关注度量和效果分析，刚开始由手工统计，后续建设一个大的管理平台去自动统计。对于 BDD：由于之前接人，一直没有听得全面，但是我觉得跟我一个想法很想，就是帮助测试人员做合适的事情，他们懂业务、懂场景，就开发出一套框架和体系去帮助他们自由组装场景，而不是让他们去关注底层的语言之类。我们要做的工作也包括基于每个人擅长的事情不同，分析如何让不同的人做自己适合的事情吧。Tom 的分享让我对敏捷开发中的测试的实践有了更深的认识，敏捷开发不单纯的指快，更需要的是如何用有效的手段保证快，例如：自动化部署、构建和测试。在前期就需要计划好环节。John 的分享确实是干货，展现了他对 selenium 的灵活的应用 ,让我对开源的应用上的认识上升到到了另一个高度，原来开源真的是不一定就比商业的差，而是你如何将他运用的更加灵活。
@simple
我觉得 tom 的那个敏捷方式跟公司项目和人员素质有极大关系 ,不可一味模仿，高难度动作

JAVA 性能测试初体验

AVA 性能测试初体验

序言：做自动化测试的时候，一直没想过要去做性能，等到现在做性能的时候，才明白这本身就是一个必须都要经历的过程，就像编程一样，编写小型软件的时候，我们不用过多关注架构和性能，但是**等成长到一定时候，就会需要关注软件的可复用性(这是由开发成本决定，这点可以在软件架构上去改善，常说的自动化框架也是为了增强脚本的可复用性和可维护性)、性能瓶颈(这是由系统资源成本决定,空间和时间的调配)、可测试行(这能大大提高测试人员的测试效率，很多时候我们要求开发提供一种测试的接口来方便测试人员进行测试)、可部署性(利用 make、ant 或者 maven,能够大大提高软件发布效率，这也是持续集成中的一种手段)等，**因此，测试中的发展其实可以有很多的，不仅关注

测试手段，还要关注如何在更多的途径上提高测试效率。下面是对本次性能测试项目至今的一些简单总结，欢迎指正。

一、性能测试项目的背景

性能测试缘起于产品存在大量背景数据时，程序响应时间过慢，而且在特定的情况下有可能会造成一些数据上报丢失，所以需要定位。

产品为 C/S 架构，采用的协议是 snmp 协议，运行在 jvm 上。

二、性能测试的策略

1、 测试目的的确定

1）系统监控，包括 cpu、内存、线程使用情况,在大数据情况下，发现问题，帮助修正代码结构，系统结构，提高系统的运行效率。

2）确定软件运行资源需求指标。

2、 性能测试指标确定

1）确定指标来源，主要包括:产品规格、行业标准、客户需求与故障场景等

2）确定测试特性，例如:系统容量、及时性、稳定性、抗压性、资源利用性等，这些特性可以根据行业性能测试特性以及产品的相关特性来决定。

3）确定具体指标，包括数目和单位。

3、 性能测试技术储备

其实性能测试可以算得上是自动化测试的一种大数据测试

1）测试场景准备：准备测试场景，可以理解为对背景数据的构造，其实可以将这种构造理解为另类的接口测试，例如:我们的软件服务器是应用 SNMP 协议进行通信，设备端有一个 agent，专门用来与软件服务器端通信，那么可以虚拟出这么一个 agent，保存相应的设备信息，虚拟过程可以通过对在网的实际设备进行录制，然后生成。

互联网中，客户端与服务器的交涉是基于 http 接口协议，其一般的性能测试都是发送大量的 http 请求，其实这种过程有一个问题就是无法模拟真实的背景数据，因为报文过于单一，而印象很深的是新浪一位朋友开发的 tcpcopy 工具，在传输层，将线上数据复制到测试场景下，从而成功模拟了真实场景环境，这是一种很好的测试方法。

(还有一种准备工作就是对测试服务器的选型，包括操作系统类型、CPU 内核数目、内存数目等)

2）测试数据准备：这其实就是接口数据，在互联网中，这方面的模拟比较简单，用很多工具，例如 LR、jmeter、soaupi 等都可以成功构造模拟 http 报文，从

而查看服务器的响应。因为我们采用的是 snmp 协议，所以业内没有这样的 snmp 接口工具，所以就自己基于 snmp 协议包开发了其 snmp 报文模拟工具。

3） 性能测试监控：性能测试过程中，对软件系统服务器的监控是关键，例如: web 测试中，往往会对 web 服务器和数据库服务器、操作系统的指标性能进行监控，因为我们的软件是运行在 jvm 上，所以直接采用 jconsole 或者 jprofiler 监控服务器的内存使用、cpu 使用、各个线程使用情况，还有对数据库和操作系统的监控等。

4、 性能测试方法

1）基于指标，进行测试数据构造测试，查看系统是否工作正常以及监测是否没有问题。

2）基于指标，在基于测试数据测试的同时，由测试人员参与进行操作，测试在特定环境下的系统工作情况。

3） 客户场景模拟测试。

4） 随机测试，利用算法进行大量随机数据构造。

三、性能测试调优

1、 性能测试是一个不断探索和不断完善的一个测试过程。

调优步骤：衡量系统现状、设定调优目标、寻找性能瓶颈、性能调优、衡量是否到达目标(如果未到达目标，需重新寻找性能瓶颈）、性能调优结束

2、 衡量现状，系统性能主要存在问题

1） 内存泄露

2） 内存占用过大，响应速率慢

3） 线程数不断增加，出现死锁或空闲线程

4） 某些类实例化数目过多，占用多余的内存空间

3、 内存泄露

1)检验方式：内存泄露需要进行时长测试，既将监控界面及系统界面全部打开，进行长时间运行（如 12 小时），观察系统类的增长情况。

2)问题定位：若出现 JVM 的 Heap 持续增长或者 Memory views 经过时长测试，出现较大规模的红色部分（增长部分），且无法 GC。

4、 系统内存占用大

1）检验方式：进行某些特定的操作，系统进行大量内存占用或者数据读写操作。

2）问题定位：若系统内存数突发性的增长，且之后不回落，说明某些模块在持续性的占用系统资源。或者出现 JVM 的 Heap 有增长，虽不是持续增长，但一直无法回落。

5、 线程数目过多或死锁

1）检验方式：进行某些特定的操作，可以使系统产生大量线程操作。

2）问题定位：若系统线程数突发性的增长或持续增长，且之后不回落，说明某些模块在持续性的占用线程。或者观察是否有许多线程来自同一个模块、长期处于 waiting 或 block 状态

6、 性能调优原则

调优过程中，充分而不过分使用硬件资源、不要一遇到问题就去改善资源环境，然后，合理调整 JVM，需要重点掌握一些 JVM 的参数，并且要善于分析系统架构和 JVM 的底层工作机制。

总结：**性能测试是一个很漫长的过程**，不管是做 JVM 性能测试、WEB 架构方面的性能测试，其实道理是相通的，个人觉得，要做好性能测试，不仅要对测试理解，更要对软件架构和底层的服务器工作机制特别理解，不然，往往，你只能去简单做一些所谓的性能测试操作，但是却无法针对很多场景提供有效的测试策略和调优建议。**好的性能测试工程师应该是能够快速搭建场景定位问题、提供指标，并且能够对软件系统架构提出有效建议。**共勉之

并发编程的测试实践分析

序言:2013 年从家回来，就好像没写过日志了，一是忙着部门整体自动化计划，二是忙着 JAVA 性能，三是忙着公司的持续集成建设，其实最主要的是人懒了，写的东西的动力来源于认可，我一直认为，理论太虚，而自己之前写的也确实让人看得虚了点，有点雾里看花的感觉，所以尽量多以实践分析和总结为主，不管如何，再简单，自己能做着自己所认可的事情就算开心了，就算是写给自己看的吧，等到以后有一天，看着自己这些东西，也许自己也会为之小小感动一下。

一、并发编程分析思考

一般的并发编程问题包括:死锁、并发错误、系统资源不足、线程泄露(工作线程抛出异常而未处理以及线程被阻塞而等待不到数据)、任务过载

场景 1:

需求场景:以前的工具也用过并发编程，其实就是多个线程同时开启，但是最近，开发一个客户级工具的需求让我开始真正认识到并发编程的难度，需求是选择 EXCEL 表，导入数量能达到 1000 个的网元，然后同时开始运行过程，每次运行队列为 10 个，其余的网元处于等待，每完成一个，则将其等待的置为运行(类似于迅雷下载)。每一个网元有各自的结果数据库文件，但是每个网元都会同时访问一个工程数据库文件。

设计方法:直接引入线程池进行管理，指定工作队列以及工作队列中保存的队列数量、线程池中维护线程的核心数量以及最大数量，先统一将所有运行线程加入到线程池中，然后开启

问题和分析:当 1000 个队列开始运行后，会发现 10 个运行线程，一段时间后，会最终只剩下一个线程在运行，其余的 9 个线程都 down 掉，后来利用 jvisualvm 对程序的线程进行跟踪，原来是由于 10 个线程的同步机制问题，因为会同时访问一个工程数据库文件，虽然中途用了单例模式，只有一个对象，但是几个线程同时调用对象，读写数据库时，会造成 Lock 异常，然后由于程序中没有对 Lock 异常进行处理，则会造成线程中断，最后只会自相残杀到最后一个，这就是线程泄露所造成的，所以要注意资源同步和程序异常的处理。

场景 2:

需求场景:之前对公司软件的测试引入 SNMP 性能测试 ,开发的工具可以模拟网元，利用大量模拟网元构建背景数据，然后引入小部分的真实网元进行操作，从而达到性能测试的目的。

问题和分析:由于大数据下的测试，软件的很多问题暴露了出来，主要是大量数据的同步，即同时收发、然后读写数据库造成。主要的问题类型包括:线程挂起不释放(这是由于死锁、大量阻塞线程造成)、数据库自动关闭(数据库参数读写频率上调后，数据库读写过快造成)、客户端卡死(model 反应到 view 层的数据结构问题)、服务器异常退出(内存泄露造成)、cpu 过高等，而这些问题是在之前的功能测试活动中并没有发现的，所以测试并发程序，所面临的挑战是潜在错误具有很大的不确定性和随机性，需要更广泛的覆盖度、更庞大的数据量以及更长的运行时间，这就引申了性能测试和系统监控。

二、并发编程的测试总结

由于并发测试主要是为了发现隐藏的不确定性，因此，我个人总结的一些前期简单策略，可以分为序列测试、人工干预测试以及随机干预测试(欢迎拍砖讨论)

1、 序列测试，即基于某些指标，以相同的输入或者步骤重复执行被测程序，并潜在错误会在某次执行中出现；例如，我们对网管的不断同步，即希望能够在合理的大数据背景下，，自动反复执行一个并发通信和数据的读写工作，将隐藏的并发问题暴露在监控系统下。并且错误之后，能够保留现场。

2、 人工干预测试，通过人为干预程序的执行，即在某个序列机制下，同时进行一些人工干预的步骤，例如:在大量网元同步的基础上，进行一些对真实设备的操作，例如：告警查看、配置操作等。

3、 随机干预测试，即引入自动化手段，在序列测试的操作上，随机从测试案例中，输入一些自动操作，并监控。

总结:因为写的突然，也许不够深刻，但突然发现，一直编程下去，总有一天要应用到并发编程这样的技术，一直自动化测试下去，总会走上性能的道路，从上层走向底层，再从底层走向应用分析，所以，刚开始我们不用预料，只要朝一个方向屁颠屁颠走下去，终都会遇到啊