业界自动化测试框架实施经验及实例研究

在软件测试日新月异发展的今天，自动化测试正在成为软件测试领域里的一个非常瞩目的趋势和潮流，很多软件公司正在或已经在企业测试团队内部实施软件自动化测试流程和框架，同时也把自动化技能作为人才衡量和业绩考核的重要技能指标，比如国外的微软，IBM，国内的金山等软件企业，都已经开始实施了自动化测试框架。

这些都不是偶然的事情，因为软件企业面临着一些难以逾越的困境。

1. 软件质量的重视和提高。

软件产业虽然只有短短几十年的历程，但是其应用范围已经从最初的科研专用转变为渗透入我们社会中生产生活各个方面，起着非常重要的作用，我们人类社会对软 件的依赖正在越来越强，根据牛顿第的反作用力定律，那么软件问题对我们的影响也在越来越大。比如 2007 年发生的奥运订票网站在开通首日不能登陆的问题， 导致上百万人购票失败；中国工商银行黄金交易系统出现漏洞，两名大学生通过低买高卖获利三千多万元，这样的新闻在报纸或网络上还能找到更多，要避免这样的 事情发生，就要使这些问题能够在软件上线之前被发现和解决，换句话说，软件质量必须要提高，而软件测试就是保证软件质量的一个重要并且非常有效的手段。因 此现在软件公司越来越重视软件测试，体现在老板那里就是对软件测试”舍得花钱，舍得投人“，测试执行起来就是”多测“，”测多”，“多测”就是时间上测试 执行频率加快，以前一个版本测一轮，现在一次编译就要测一轮，“测多”就是测试更加完整，覆盖更多的功能模块。这就为软件自动化测试提供了强有力的需求和 生长空间。

2. 软件系统规模的扩大，复杂性的提高

我们记得在单机系统时代时，几千行代码就能写出一个商业软件，比如 WPS，CCED 这些一代软件枭雄。但随着网络时代的到来，分布式系统的发展，软件系统 越来越重视交互和协作，多个模块服务的交叉调用，网间的交互安全等等，这大大提高了软件系统的复杂度和规模。Oracle 曾经开发过一个邮件客户端 Outlook 的插件，这个插件是安装在 outlook 上，提供一些常用功能，比如收发 mail，calendar 创建等等。但 oracle 的测试部门仅 仅为这个插件就设计了 6000 多个 test case！这个数目是如此巨大，使得测试执行和产品周期产生了深刻的矛盾。这个矛盾体现在：当每个新版本发布时，如果做一遍完整测试，一个人手工测试执行 一遍 6000 多个 test case 就要半个月，而产品版本的发布周期也就一周左右，也就是说测试的速度远远跟不上产品的发布速度。在这种情形下，如果没有自动化测试帮忙，手工测试 只能望洋兴叹了！

以上两个软件的根本现状，决定了软件测试自动化的趋势不是人云亦云，昙花一现，而会蓬勃发展，强劲有力，成为势不可挡的潮流。很多软件公司已经看到了这个 潮流，很早就开始做软件自动化测试的预研和实施，比如微软，oracle 等已经在企业内部测试团队整合了自动化测试流程，实施了自动化测试框架，并且已经 按时更新换代，进入稳定应用的周期。但在国内，由于软件自动化测试时间不长，测试人员技能水平等因素影响，软件自动化测试的研究和实施大多还处于一个起步 摸索的阶段，我们看到普遍的情形是”做的人不少，成功的不多“，这种现状一方面有技术的问题，另外也有方向上的问题。因为在业界可供借鉴的成功实施经验或 案例比较少，所以在起步阶段可能就会走弯路，这是摸索必然要付出的”学费“。

1. 业界自动化测试实施状态探究

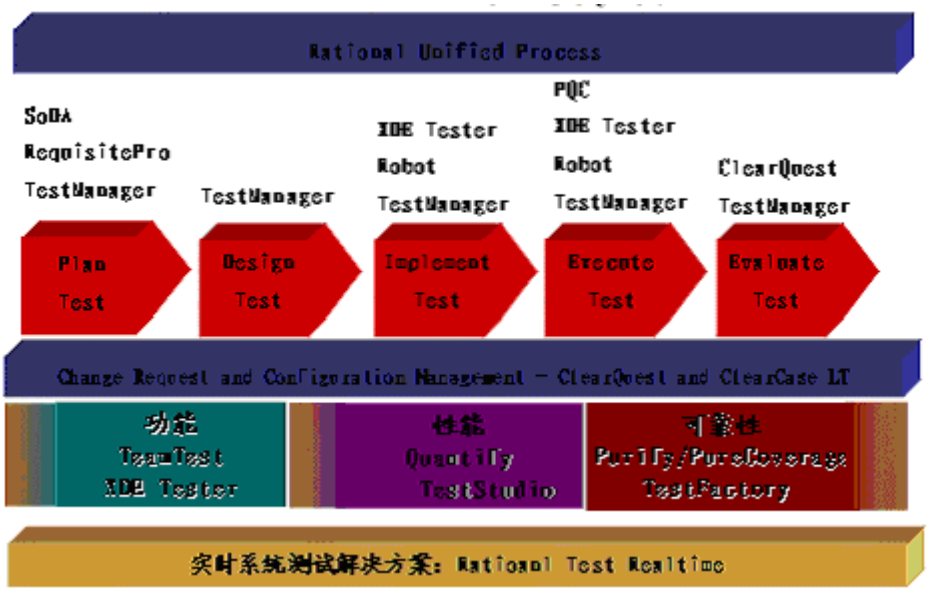
那么怎样能够把握软件自动化测试方向和思路呢？下面笔者结合业界的现状分析，以及未来展望，对软件自动化测试的作出三个层次的划分：

1.1 第一阶段：测试的自动化–以工具为中心

这一般是实施自动化测试最原始的起步阶段，其目的就是将原先手工测试所作的工作转化为自动化代劳。显著的特征就是以工具为中心，比如 QTP 的应用，原先靠人工来执行的测试案 例，比如点击，录入等，现在由 QTP 来完成，如果 QTP 不能支持我们的系统，我们就要寻找解决方案，或改用其他工具，总之大多数的自动化工作重点是在每个 case 上，也就是技术层面上的问题。

因此，这个阶段最明显的特征是所有的自动化测试实施工作是以工具为中心，在这个阶段的过程中，要完成的工作如下：

a) 软件测试团队学习和掌握自动化工具的基本知识，并能根据当前被测产品的软件特征和项目特点，选择合适的自动化测试工具族（详见 PEARL 模型 Evaluation 一节）。这通常以有效的工具知识培训或咨询的方式来完成的。如图 5-1 是一个 rational 测试工具实施整体解决方案，在测试流程里，不同的测试阶段都有相应的工具方案支持，我们可以根据测试的需求来选择相应的测试工具。



t 图 5-1 rational 自动化测试工具解决方案体系图

b) 测试工程师基于工具来开发生成自动化测试脚本，通过运行脚本，来完成测试案例的执行工作。比如 QTP 的应用，原先靠人工来执行的测试案例，比如点击，录入等操作，现在由 QTP 的脚本来完成，如果 QTP 不能支持我们的系统，我们就要改用其他工具。

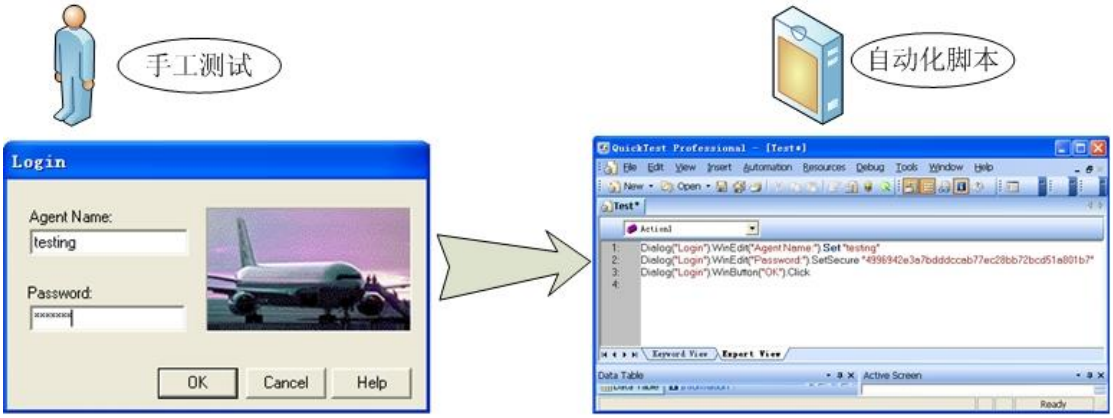


图 5-2 手工测试案例到测试脚本映射图

但随着技术上的解决，自动化测试规模的进一步增大，我们很快就面临下面的问题之一或全部：

（1） 如何提高的测试程序或脚本的复用性以获得自动化测试的高收益？

自动化测试实施成本包括前期的开发成本和维护成本。其中维护成本是一个自动化测试重点考虑的一个特别问题。显然，脚本数量越多，维护的工作量就越多。实际上，构建 lib 库，数据驱动和关键字驱动等都是为了提高脚本的复用性而采用的手段。

（2）如何解决测试自动化的健壮性问题？

一个不健壮的自动化测试会给测试工程师带来很多麻烦，它们会受到种种干扰因素的作用，有时会 pass，有时会 fail，这大大增加自动化测试 debug 的工作量。尤其在 UI 自动化测试中，环境的差异，弹出式对话框都会直接干扰自动化测试运行。如何保证自动化测试的健壮性？面对健壮性问题，可以考虑从两方面入手，第一从代码上可以考虑引入 transaction 机制等来运行，第二是建立良好的自动化测试开发规范。

（3） 自动化测试脚本的类型和数量越来越多，怎样有效组织管理和调度这些脚本？

比如有 1000 个脚本，有 QTP 的，有 Winrunner 的，还有 perl 的等等，每种类型的测试脚本又实现了多个测试案例，那我们怎样统一管理和调度这些脚本，使之能够组成一个宏大的自动化测试目标？要知道单个的测试脚本和单个的测试案例一样，对于公司的老板来说，他们是不关心这些细节的，只有它们组合起来成为一个壮丽的图本，才能体现出来它们的价值。也许老板们刚开始对自动化的执行感到新奇和惊喜，但当他们意识到这些并不能为他带来什么价值时，他就会厌倦并放弃。

（4）. 自动化测试如何与手工测试整合？

我们知道自动化测试是不能 100%完全替代手工测试的，那么自动化测试必须要和手工测试整合在一起，才能体现出其价值。这意味着，自动化测试要全方位地和手工测试整合，包括前期的案例管理，测试的执行，以及最后的报告呈现。

以上问题都可归结于自动化测试的一个根本矛盾，即对我们对自动化测试提高测试效率的目的要求与自动化测试高成本高投入的现实的矛盾。

这些问题是助产士，它们促成软件自动化测试第二个阶段的到来。

1.2 第二阶段：百家争鸣–形形色色的自动化测试框架

在众多书籍和网络论坛中，经常被提到的有代表性的自动化测试框架思想有以下几种形式：

1.2.1 数据驱动测试框架（The Data-Driven Testing Framework）

数据驱动测试是测试从数据文件（数据池，ODBC 源，cvs 文件，Excel 文件，DAO 对象等）中读取输入和输出数值并载入到录制的或手工编码的脚本变量中的一种框架。在这种框架里，输入数值和输出验证数值都使用变量。在测试脚本中编写贯穿程序的导航，数据文件的读取，记录测试状态和信息的日志的代码。

测试用例包含在数据文件里而不是在脚本中被 hard code，这种框架和表格驱动测试有些相似；脚本只是一种“驱动器”（driver）或传送数据的机制。尽管导航的数据不包含在表结构中，但和表格驱动测试还是不同的。在数据驱动测试里，只有测试数据包含在数据文件中。

如果使用 SQABasic 语言和 IBM Rational 的数据池功能，IBM Rational 工具集里有自带的数据驱动功能。为了演示这种框架的使用，我们将测试一个简单的应用程序中的订单表格。

图 5-3 订单应用程序

录制对以上窗口的操作，得到以下 SQABasic 脚本：

```
'Data Driven Framework
'Test Case Script

Sub Main

    'Make An Order
    Window SetContext, "Name=frmOrder", ""

    'Card Number
    EditBox Click, "Name=txtCreditCard", "Coords=16,9"
    InputKeys "3333444455556666"

    'Expiration Date
    EditBox Click, "Name=txtExpirationDate", "Coords=6,7"
    InputKeys "12/2005"

    'Place Order
    PushButton Click, "Name=cmdOrder"

    'Confirmation Screen
    Window SetContext, "Name=frmConfirm", ""
    PushButton Click, "Name=cmdOK"

End Sub
```

Rational 录制生成的脚本

我们可以使用 Rational 自带的数据池来设置测试有效和无效信用卡号和过期日期的测试用例。例如，图 5-4 中是用于测试数据字段的测试用例中的数据池。

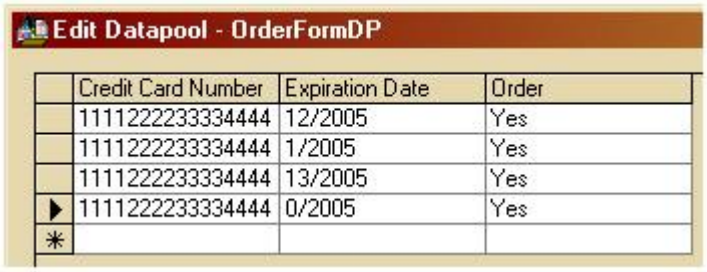


图 5-4 Rational 数据池

将这些数据应用到脚本中，即参数化后的脚本如下：

```
'Data Driven Framework
'Test Case Script

'$Include "SQAUTIL.SBH"

Sub Main
    Dim Result As Integer
    Dim DatapoolHandle As Long
    Dim DatapoolReturnValue As Variant

    'Open the datapool
    DatapoolHandle = SQADatapoolOpen("OrderFormDP")
    '...Add error checking....

    'Loop through the datapool
    While SQADatapoolFetch(DatapoolHandle) = sqaDpSuccess
        'Open Order Form
        Window SetContext, "Name=frmMain", ""
        PushButton Click, "Name=cmdOrder"
        Window SetContext, "Name=frmOrder", ""

        'Card Number
        Result = SQADatapoolValue(DatapoolHandle, "Credit Card Number", DatapoolReturnValue)
        '...Add error checking....
        EditBox Click, "Name=txtCreditCard", "Coords=16,9"
        '...Clear Value...
        InputKeys DatapoolReturnValue

        'Expiration Date
        Result = SQADatapoolValue(DatapoolHandle, "Expiration Date", DatapoolReturnValue)
        '...Add error checking....
        '...Clear Value...
        EditBox Click, "Name=txtExpirationDate", "Coords=6,7"
        InputKeys DatapoolReturnValue

        'Place Order
        Result = SQADatapoolValue(DatapoolHandle, "Order", DatapoolReturnValue)
        If UCase(DatapoolReturnValue) = "YES" Then
            PushButton Click, "Name=cmdOrder"

            'Confirmation Screen
            Window SetContext, "Name=frmConfirm", ""
            PushButton Click, "Name=cmdOK"
        Else
            PushButton Click, "Name=cmdCancel"
        End If
    Wend 'Go fetch next row

    'Close datapool
    Result = SQADatapoolClose(DatapoolHandle)
    '...Add error checking....
End Sub
```

为了使用数据池，这里增加了 SQABasic 命令，还增加了“While”循环来处理在数据池中每一行数据。必须说明一下，在“If…Then”语句中的 Ucase（SQABasic 命令）函数。Ucase 用于将参数（在这个例子里是指数据池返回的数值）全部转换成大写。这种方法不是大小写敏感的，所以代码更具有健壮性。

这个框架趋向于减少实现所有测试用例而需要的全部的脚本数量，并且在开发绕开错误的办法（Workaround）和维护方面提供了最好的灵活性。和表格驱动测试非常相似的是，表格驱动测试只需要非常少的代码就可以产生大量的测试用例。

1.2.2 关键字驱动或表驱动测试框架（The Keyword-Driven or Table-Driven Testing Framework）

关键字驱动和表格驱动测试是在数据驱动基础之上将进一步提高自动化测试的灵活性和扩展性的框架解决方案。在关键字驱动框架下，除了要设计应用程序的测试数据表，还要生成一个关键字数据表。这个关键字包含了程序的特性，或者是有关程序的操作。

比如，如果要映射出手工测试 Windows 计算器功能过程中用鼠标执行的操作，我们可以创建如下表 5-1。“Window”一列包含了我们执行鼠标操作的应用程序窗口的名字（在这个例子中，他们都发生在计算器窗口里）。 “Control”一列指出了鼠标点击的控制键的类型。“Action” 一列列出了鼠标的操作（或是测试人员的）。“Arguments”列指出了特定的控制键（1, 2, 3, 5, +, -等），如表 5-1 所示

Window	Control	Action	Arguments
Caculator	Menu		View standard
Caculator	PushButton	Click	1
Caculator	PushButton	Click	+
Caculator	PushButton	Click	3
Caculator	PushButton	Click	=
Caculator		Verify result	4
		Clear	
Caculator	PushButton	Click	6
Caculator	PushButton	Click	-
Caculator	PushButton	Click	3
Caculator	PushButton	Click	=
Caculator		Verify result	3

表 5-1 关键字驱动表

以上这个关键字数据表更像是一个测试案例，其中 arguments 是输入数据，control 和 action 则是本次测试案例中的关键字。测试人员只需设计和维护这张表，关键字驱动框架的任务是将这张数据表解析成脚本并执行。解析的 QTP 脚本如下：

```
Window("计算器").WinButton("1").Click
Window("计算器").WinButton("+").Click
Window("计算器").WinButton("3").Click
        Window("计算器").WinButton("=").Click
        Window("计算器").WinEdit("Edit").Check CheckPoint("4")
        Window("计算器").WinButton("6").Click
        Window("计算器").WinButton("-").Click
        Window("计算器").WinButton("3").Click
        Window("计算器").WinButton("=").Click
        Window("计算器").WinEdit("Edit").Check CheckPoint("3")
```

从这个例子中，可以看到关键字框架的优势是测试人员设计和维护自动化测试案例的工作量非常小，仅维护一张数据表，而脚本代码的工作交给框架来生成并执行。缺点是，对框架的要求很高，而且由于大部分程序逻辑被框架控制和实现，因此在使用起来，不够灵活，扩展性不强。因此，我个人更倾向于认为关键字驱动理念只是一种想法（idea），实践性并不强。

1.3 第三阶段：完整解决方案–自动化测试框架

如果说前面谈到的自动化测试实施还是在一个点上下功夫，那么本阶段就是在一条线上作战了。“自动化的测试”的内涵更加丰富，它意在将软件自动化测试中所涉及的各个环节作为一个统一的整体考虑，从测试脚本的管理，测试脚本的执行到测试报告的展现都有相应的策略，规范定义及自动化实现，故称这个阶段为“自动化的测试”。

简而言之，我们在自动化测试实施中会遇到各种各样的问题，有的是关于流程，有的是关于自动化测试本身，还有的是关于规范等等，这些问题不可能通过单一的自动化测试编程手段解决，而是需要一个有机的系统的解决方案，这个解决方案就是测试框架。

因此，测试框架相比单个的测试脚本具有以下特点：

a. 测试框架位于软件测试的战略规划层次，而非执行层面

很显然，测试框架在测试流程上表现为手工测试与自动化测试的整合策略，在组织上是一套自动化测试管理开发及执行的规范。这些改变对传统手工软件测试的工作内容和方式的影响是巨大而深远的。

b. 测试框架具有组织上的延续性和软件上的扩展性

测试框架在软件测试组织中的建立和成熟是一个长期的过程，一旦建立，测试框架就可作为组织的知识经验甚至文化的一部分，随着团队的发展而延续，同时，由于被测软件的更新和测试人员技能的不断完善，测试框架也是一个不断进行扩展和更新的发展过程。这必然就对软件框架要求在软件上具有较高的扩展性性。

那么自动化测试框架作为一个整体解决方案，到底包含哪些组成部分呢？

一提到测试框架，很容易想到 Junit，QC 等被我们称之为框架的测试工具，这给了我们一个错觉，以为框架就是一些代码，lib 或者脚本。其实不然，**软件自动化测试框架从本质来讲是一系列策略思想，规范文档和代码的集合。**针对上面的问题，框架的解决方案包括但不限于：

1. 策略思想
   a) 与手工测试整合策略
   b) 脚本存储和管理策略
   c) 任务和组件思想
2. 规范文档
   a) 脚本开发规范
   b) 脚本与框架集成规范
   c) 日志规范
   d) 报告规范
   e) 数据源存储方式和格式定义
3. 代码
   a) 测试案例自动化后的脚本

b) 测试框架拓扑结构的建立
c) 测试脚本库的建立
d) 案例底层驱动机制
e) 与其他系统的集成

1.4. 业界分析总结

目前国内软件公司软件自动化测试的实施情况大多处于第一阶段或从第一向第二过渡的阶段。

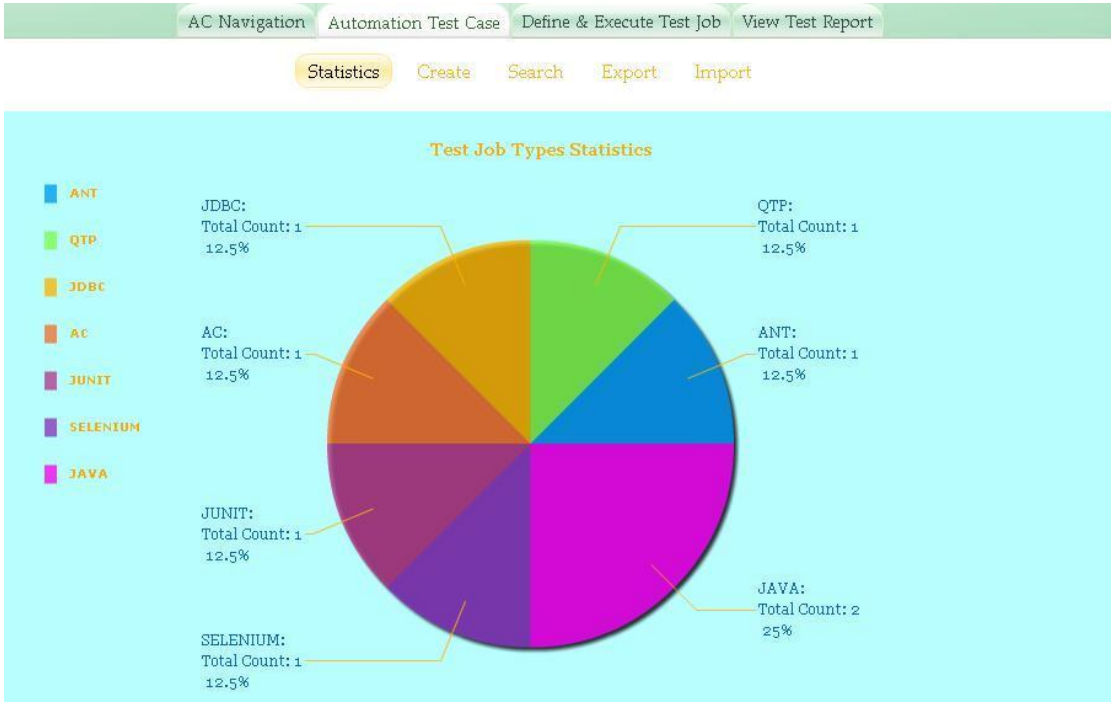
实施经验	第一工具阶段	第二框架阶段	第三完整解决方案
代表企业	国内多数应用软件企业，如中国移动研究院等等	国内一些纯软件企业。如华为，用友等等	大型外资企业，如微软，Oracle 等等
实施时间	1 年左右	1-2 年左右	3 年以上
应用技术	QTP,Selenium 等	QC，STAF 等	Automation Center 自动化测试框架
实施效果	团队工具技术获得提高，而收益成本比较低，无法获得真正成功	已经具备实施能力和条件。收益成本持平	自动化收益客观，真正减少测试工作量，提高测试效率

2. 测试框架解决方案效果一览

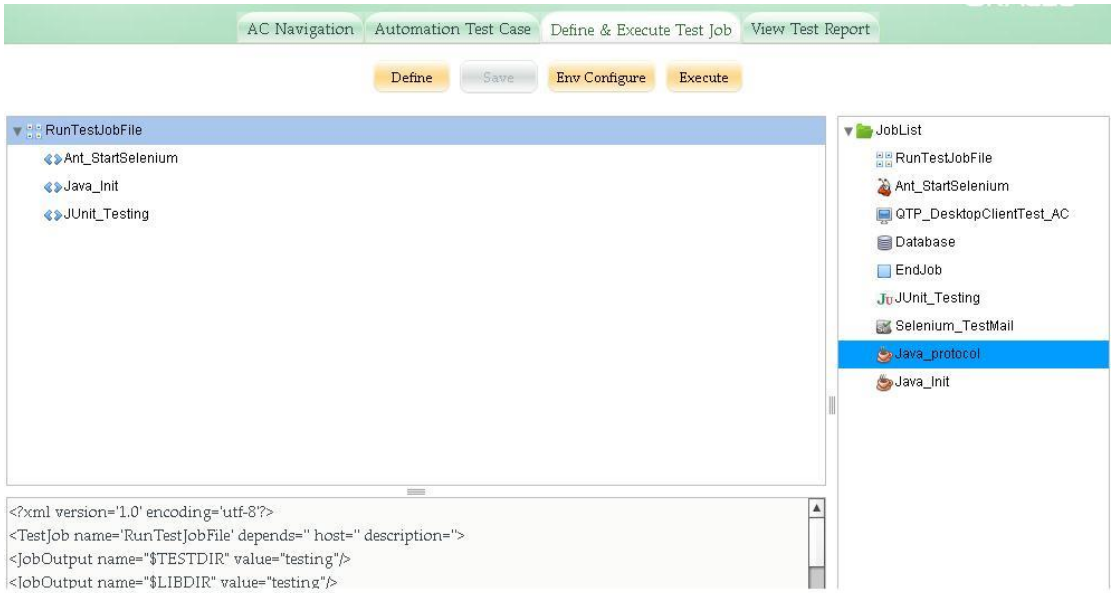
以 Automation Center 为例来说明企业团队可以通过实施框架，来达到提高测试效率，度量软件质量状态的目的。

2.1. 管理不同的自动化测试脚本

在 AC 中，可以管理 Java，QTP，Selenium，Ant，Junit，JDBC 等多种类型的自动化测试脚本，并能对其进行统计分析



2.2. 定时调度运行测试任务，无人干预的自动化回归测试



定义测试任务

2.3. 自动生成报告，邮件通知，web 呈现

生成基于 xml 数据源的自动化测试报告，以邮件通知方式，自动发送给相关人员

所有的 TestJob 运行结果将会被呈现在一份测试报告中。测试报告分为两级，总览报告和细分诊断报告。

总览报告的阅读受众为测试经理，质量总监等管理角色。因此，总览报告里主要提供运行案例数，成功运行数，失败数等概要信息。

Beehive - Client Automation Test Results

测试产品版本: OCSCORE_MAIN_LINUX_081209.1310
测试执行日期: 11.12.2008 15:30:47
测试系统用户: cdctest
操作系统信息: Microsoft Windows XP Professional, 5.1.2600
测试执行语言: de
测试执行时间: 1 小时 36 min 11 sec

测试结果汇总

测试集合	测试案例组			测试案例		
	通过	失败	无法运行	通过	失败	无法运行
sanity_obio	15 [75%]	5 [25%]	0 [0%]	30 [75%]	10 [25%]	0 [0%]

测试集合: sanity_obio

被测模块	测试案例组			测试		
	Pass	Fail	CNR	Pass	Fail	CNR
OUTLOOKINT	15 [75%]	5 [25%]	0 [0%]	30 [75%]	10 [25%]	0 [0%]

图 1-3 AC 概览报告

细分诊断报告的阅读受众为自动化测试开发人员，手工测试人员等技术角色。为失败案例的诊断提供详细的信息，比如检查点，日志，抓图等等。

测试案例: obio_createupdatepermanentlydeletetask

运行结果	Failed
被测模块	OUTLOOKINT
功能点	MS Outlook Integration - OUTLOOKINT (OBIO)
案例描述	Create/Update/Shift-delete regular tasks by specifying data in all available fields
执行花费时间	5 min 40 sec
案例名称	+CreateTaskWithAllFields - Failed +UpdateTaskByReplacingDataAllFields - Passed -PermDeleteTaskWithAllFields - Failed
time	1 分钟 9 秒
description	PermDeleteTaskWithAllFields
Check Point	Passed:Sucoessfully open Outlook Advanced Find Passed:Sucoessfully select Advanced Find search type [Aufgaben] Passed:Sucoessfully select folder [Mein Arbeitsbereich - shining liu;Aufgaben] Passed:Sucoessfully type [1229012201 EditTaskWithAllFields Subject] on object [1229012201 EditTaskWithAllFields Subject] Passed:Sucoessfully enter search string [1229012201 EditTaskWithAllFields Subject] and select search field [Nur im Feld "Betreff"] Passed:Sucoessfully select [0] item in window [Erweiterte Suche] Passed:Sucoeeded to delete #0 item permanently Passed:Sucoessfully access Aufgaben: Eigenschaften - [Verwaltung] tab
Failure	Failed:Failed to reset folder because there are pending upload changes Failed:Failed to reset folder Failed:Test:PermDeleteTaskWithAllFields:All attempts for Recovery failed.
Log	runtime log
QTP report	QTP report
screenshot	screenshot