IBM Software Group

自动化测试技术交流

IBM 软件部　　聂健

Agenda

- 自动化测试的原理
- 自动化测试框架
- 如何扩展
- 自动化测试的实施探讨

自动化测试的工作原理

- 录制 + 回放
- 录制:执行具体的业务操作,通过工具形成脚本
- 完善脚本
  - 添加验证点
  - 添加数据驱动测试
  - 设计各种流程
- 回放:还原被测试环境,回放脚本,查看日志

- 工具适用的前提:
  - 能够识别应用的各种对象

不同的工具,不同的适用范围

Robot	***Rational Functional Tester***
Dephi	Java
PowerBuilder	VS.NET
VB	Siebel
VC,VC++	Web(IE,Mozilla Firefox1.5)
Oracle Form和其他应用类型	SAP
Web	AJAX
Java	Flex

对象识别技术

- 严格的按对象识别
  - A.click()
- 半严格对象识别
  - Click,a,(10,20)
- 位置识别
  - Click,10,20

Agenda

- 自动化测试的原理
- **自动化测试框架**
- 如何扩展
- 自动化测试的实施探讨

传统的工作方式

- 等待一个可用的构建……
- 耐心等待……
- 得到第一个构建，录制脚本，插入验证点(VPs)
- 回放脚本
- 如果新的版本有明显的改动，重新改进相关的脚本，也许要重新录制相关的脚本

这种简单方式将会遇到的挑战

- 界面对象、执行步骤和验证点掺杂在一个脚本中
  - à ***代码不能很好重用！***
- 脚本很长并且按顺序录制
  - à ***很难调试!***
- 要录制脚本必须等到有可以执行的构建
  - à ***延迟产品可以发布的时间！***
- 测试脚本和被测系统紧密关联
  - à ***代码不容易维护!***

解决方法

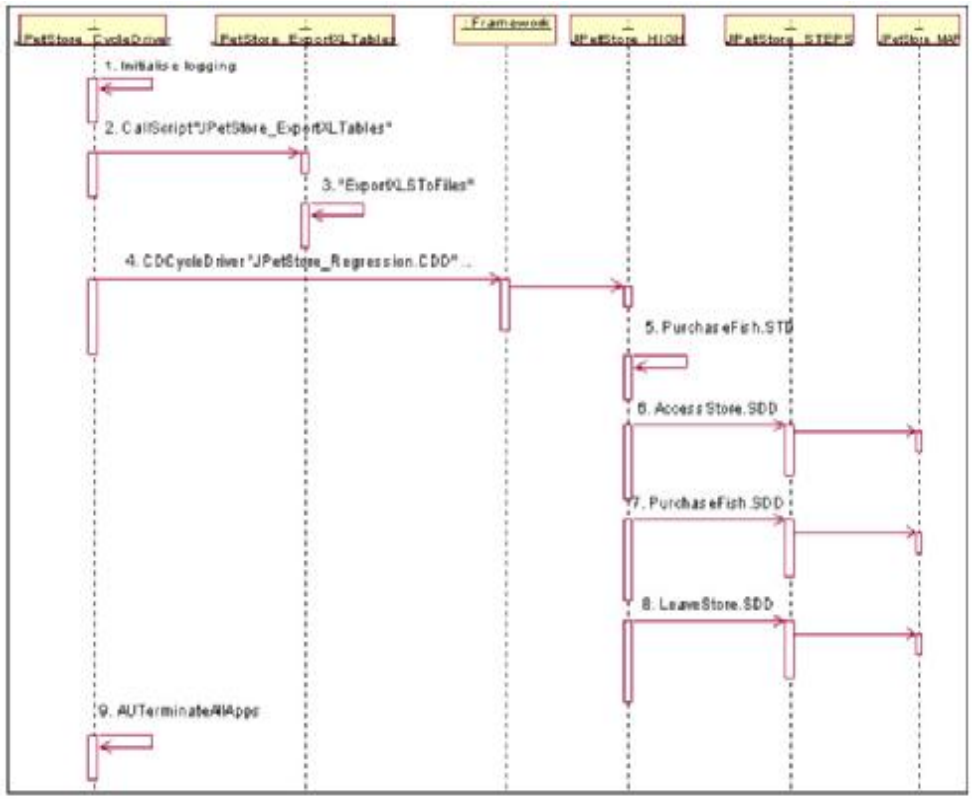
根据测试团队的侧重点和能力

- 业务角度
  - 业务测试框架(RRAFS,表驱动,Object map)
- 技术角度
  - 技术测试框架(IBM framework)

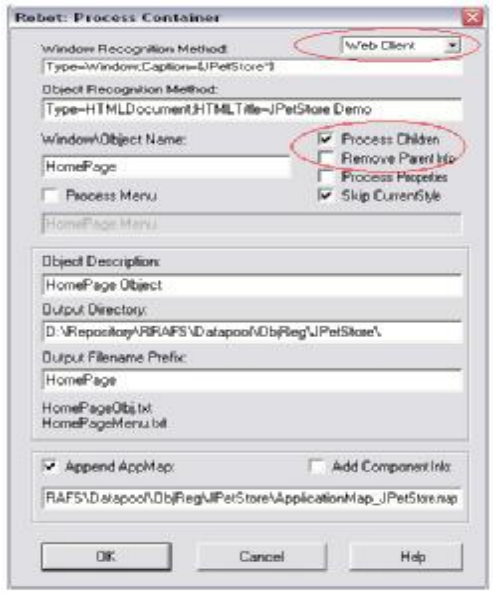
测试框架的好处?

- 代码重用…
- 调试…
- 测试脚本的维护 …
- 团队之间的协作…

业务框架(RRAFS)



RT	Window	Comp	Action	Arg	Arg
S	LoginPinPage	PinEditBox	VerifyProperty	Visible	True
S	LoginPinPage	PinEditBox	SetTextValue	^password	
S	LoginPinPage	SubmitButton	ClickButton		
Login_Pin					

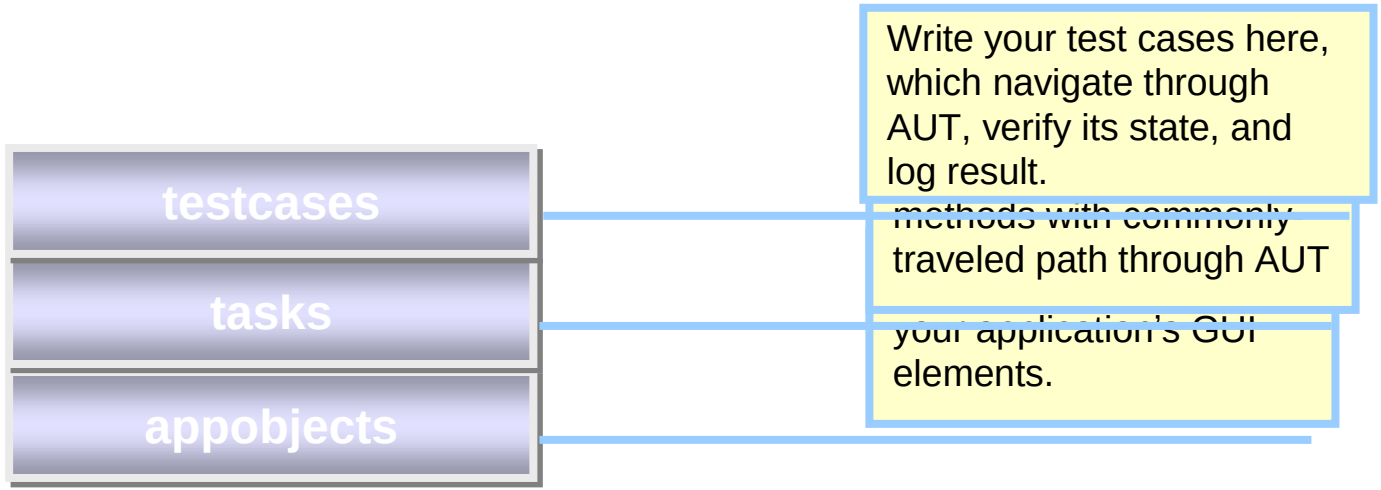


RRAFS的好处:

1. 全称Rational Robot Automatic Framwork Support),是基于Robot开发的开源测试框架
2. 通过框架分离了测试用例(High Step),重用的Step和Object map
3. 表驱动方式代替了传统的脚本驱动
4. 工具自动生成Object map
5. 通过工具生成Step

技术框架 (IBM Framework)

The three-tiered architecture is implemented through the appobjects, tasks, and testcases packages



Layered architecture separate the “what” from the “how”.

AppObjects


```
public class LoginPage extends LoginPageHelper
{
        public GuiTestObject getText_LoginID() {
                return Text_LoginID(ANY, NO_STATE);
        }

        public GuiTestObject getText_Passwd() {
                return Text_Passwd(ANY, NO_STATE);
        }

        public ToggleGUITestObject getCheckBox__PersistentID() {
                return CheckBox__PersistentID(ANY, NO_STATE);
        }

        public GuiTestObject getButton_SignIn() {
                return Button_SignIn(ANY, NO_STATE);
        }
}
```

tasks

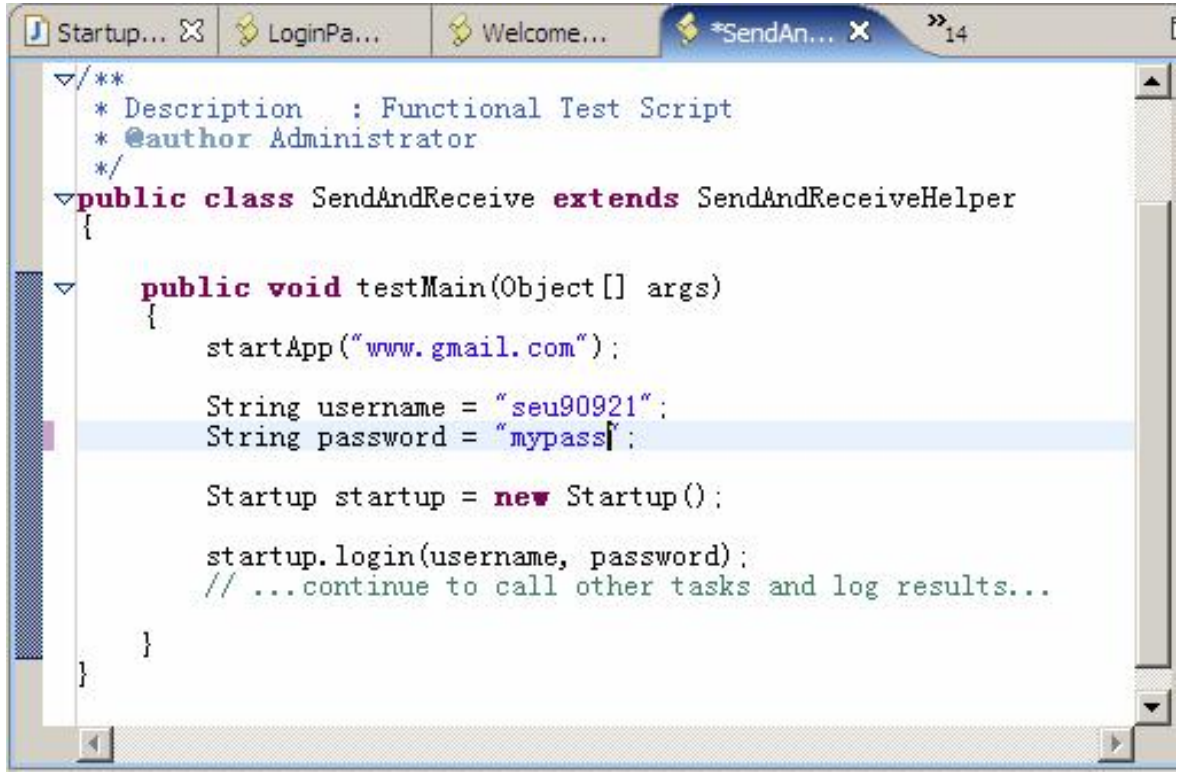
- Tasks 的任务是调用appobjects提供的方法来操纵GUI元素以完成一些通用的Task。反过来，tasks的方法会被testcases调用
- Tasks的力度要把握提高代码重用性以及对testcases屏蔽下层的实现细节

Startup... | LoginPa... | Welcome... | SendAnd... | 14

```
*/

public class Startup extends StartupHelper
{

    LoginPage loginPage = new LoginPage();
    WelcomePage welcomePage = new WelcomePage();

    public boolean login(String user, String password)
    {
        loginPage.getWLoginID().setText(user);
        loginPage.getWPasswd().setText(password);
        loginPage.getWSignIn().click();

        welcomePage.getStaticText_WelcomeMessage().click();
        if (welcomePage.getStaticText_WelcomeMessage().exists())
            return true;
        return false;
    }

}
```

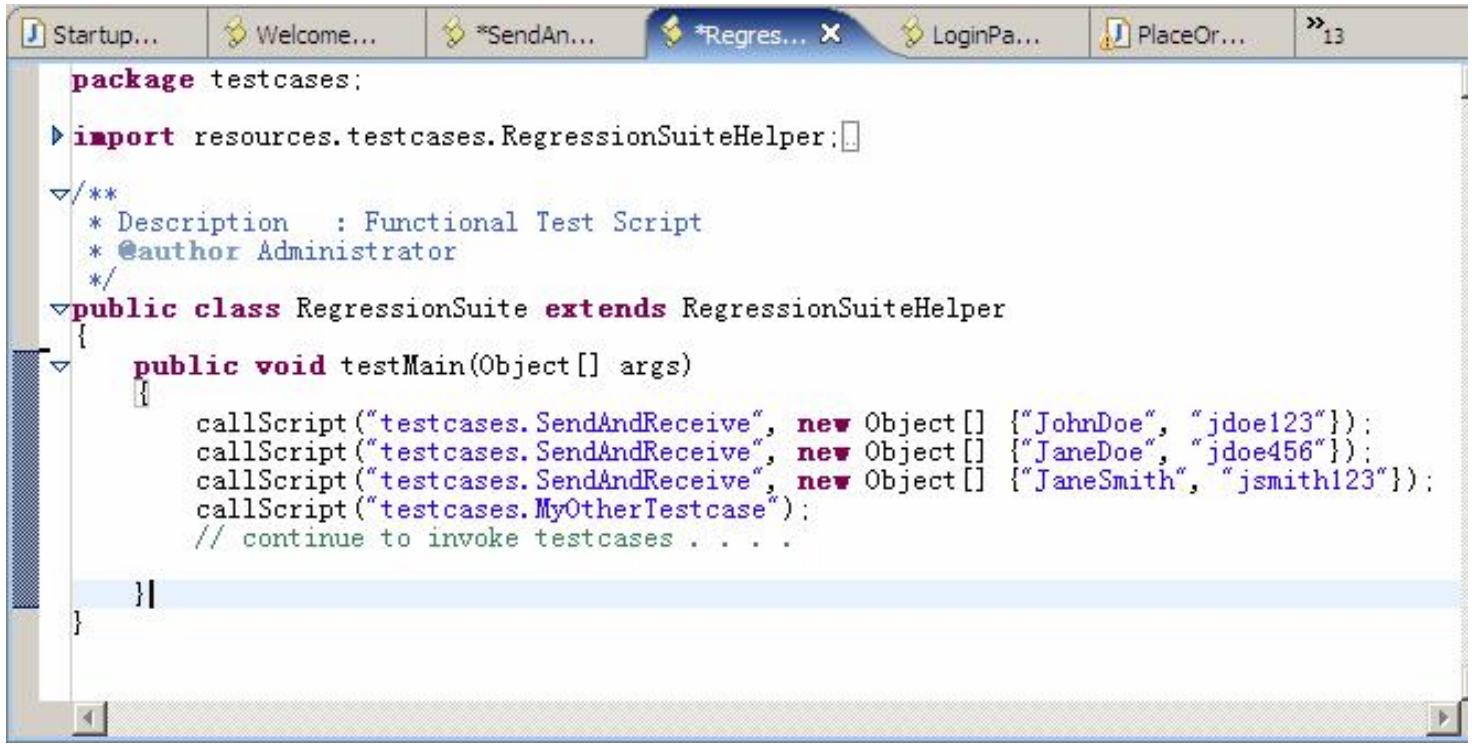
testcases

- Testcases 调用 tasks (如果需要的话传入需要的数据), 验证条件和记录结果
- Testcase应该仅仅包含简单的逻辑和控制流，其他的都应该在tasks这个包中实现



testsuites

- Test suite类可以通过从RationalTestScript继承过来的callScript()方法定义。Testsuite类可以放在testcases包中或在testcases中新建一个包来存放。你可以定义好testcase，然后用很多不同的数据作为输入以达到数据驱动测试的方法。



IBM 类包

- Ibm类包提供了很多类库能帮助你实现测试的自动化
- Ibm类包包含了：
  - Ibm.loggers: 提供了测试脚本可以直接调用的loggers
  - Ibm.recovery: 提供记录一个特定的状态并在需要时可以恢复到需要的位置
  - Ibm.tools: 包含了ClassGenerator类，可以用来从脚本的object map中生成所有的getter方法
  - Ibm.util: 包含了一些常用的方法方便使用RFT
  - Ibm.widgets: 扩展了TestObject功能 -- *Most valuable!*

The only visible present from IBM framework !

lbm.widgets

- 通常，测试对象的操纵如下(例如：在一个输入框中进行输入)：

```
Text_name().click(atPoint(35,10));
Browser_htmlBrowser(document_C(),DEFAULT_FLAGS).inputKeys
("{ExtHome}+{ExtEnd}{ExtDelete}");
Browser_htmlBrowser(document_C(),DEFAULT_FLAGS).inputKeys("test");
```

你会喜欢吗?

神啊，救救我吧！

ibm.widgets

- 为什么不把这些公用的步骤封装在一个方法中呢？

对！就是这样。但是你不必自己做，ibm.widgets 已经将大部分都做好了！

ibm.widgets

Class WTextField

java.lang.Object
  └com.rational.test.ft.object.interfaces.TestObject
      └com.rational.test.ft.object.interfaces.GuiTestObject
          └ibm.widgets.ancestors.Widget
              └**ibm.widgets.WTextField**

```
Text_name().setText("Test");
```

ibm.widget – 怎么用?

- 在appoject包中那些类提供的getter方法里构建widget对象

```
public WTextField getPasswordText()
{
    TestObject to = passwordText(ANY, NO_STATE);
    return new WTextField(to);
}
```

- 从 appobjects包的类中的getter方法得到这些widget对象以便操作:

```
WTextField passwdField = loginForm.getPasswordText();
passwdField.setText("Test");
```

ibm.widgets(续.)

- 对Java界面元素和Web界面，常用的GUI对象都已经有相应的widgets
- 对Java的swt GUI 对象, 最新的ibm包中也已经有子包中包含了响应的支持

Ibm.util

- Util这个包提供了对图象，浏览器，文件的一些常用操作。
  BitmapOps, BrowserOps, FileOps…etc.
- 最新的ibm包也提供了对基于终端的应用系统的支持

Ibm.util

- BitmapOps:
  - 截屏
  - 捕获一个特定的测试对象
  - 比较两个图片并找出区别
- BrowserOps:
  - 启动一个浏览器窗口
  - 关闭特定的或所有活动的浏览器窗口
  - 找到一个特定的浏览器窗口
  - 确定一个浏览器窗口是否已经完成
  - 在页面中找到一个特定的字符串
  - …

Ibm.loggers

- 如果利用ibm包中提供的日志支持，实现如下需求将会非常容易：
  - 确定输出目标：你可能想将日志写到HTML文件、文本文件、控制台或同时输出到文件和控制台
  - 自定义输出格式：你可能需要清除日志中一些额外的信息或添加一些自己的信息
  - 保存截屏：每次当自动化测试记录到一个错误时捕获并保存截屏
  - 实现一个被动日志(passive logger)：记录所有的动作和过程以减少调试时间
  - 实现测试用例追踪：查看当前多少测试用例通过了；多少失败了

Ibm.recovery

- 想象一下这个场景:
  - 100个测试用例需要一个一个的执行
  - 执行到第三个时，意外发生…这个测试用例执行失败
  - 由于没有必要的清除步骤，第四个测试用例由于运行环境问题而失败
  - 同样的原因，的五个测试用例失败….

- 必须要有步骤来确保在执行下一个测试用例的开始环境是“干净”的

Ibm.recovery (续.)

- 首先，根据应用系统特定的环境要求，定义并实现一个基状态：
  - 我需要多少个不同的基状态？
  - 基状态需要一些输入参数吗？
  - 我要接受不通的输入参数来重载基状态方法吗？

例子： **WebBaseState …**

Ibm.recovery (续.)

- 调用callBaseState方法转到定义好的基状态。根据你的被测系统，可以在一个测试用例开始时调用，也可以在结束时调用

```
public boolean login(String sUserName, String sPassword) {

      //set the default user name and password here,
      gsUserName = sUserName;
      gsPassword = sPassword;

      new WebBaseState("login.yahoo.com").callBaseState();

      LoginPage loginPage = new LoginPage();

      WTextField tfPassword = loginPage.getText_password();

      if (!tfPassword.exists()) {
            logError("Could not Login! Cannot continue.");
            return false;
       }
```

Ibm.tool

- 目前, 在这个包中只有一个类 (ClassGenerator),它可以用来帮助在appobjects中生成getter方法

```
Vector v = new Vector();

v.add(new PlaceOrderForm());
v.add(new OrderCfmBox());
v.add(new FailOrderBox());
v.add(new AppForm());
v.add(new LoginForm());

new ClassGenerator().updateScripts(v);
```

界面变化时如何避免脚本的改变(AppObject)

- 对appobjects和tasks包中的类进行单元测试可以在整体上节省很多时间
- 在一个新的构建版本出来时，对appobjects进行单元测试可以节省更多的时间
- 单元测试可以很简单，只要在每个类的下面添加testMain()方法，然后给每个对象传入一个消息

```
public void testMain (Object[] args) {
        // To invoke this page: start the app and make sure the default
        // Album tab is selected.
        try {
                getButton_PlaceOrder().getParent();
        } catch(Exception e) {
                System.out.println("getButton_PlaceOrder(): "+e.getClass());
        }
        try {
                getLabel_AlbumComposer().getParent();
        } catch(Exception e) {
                System.out.println("getLabel_AlbumComposer(): "+e.getClass());
        }
        (etc.)
```

Agenda

- 自动化测试的原理
- 自动化测试框架
- 如何扩展
- 自动化测试的实施探讨

如果无法识别对象?

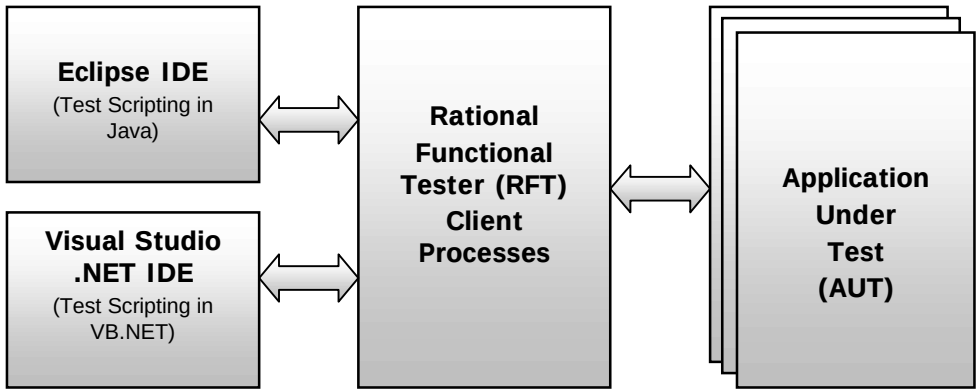
- 工具适用的前提:
  - 能够识别应用的各种对象
  - 系统必须提供相应的和外部系统的接口,便于工具扩展支持
    - RFT Proxy SDK
    - QTP 允许定义客户化对象

开始之前 – 基本术语

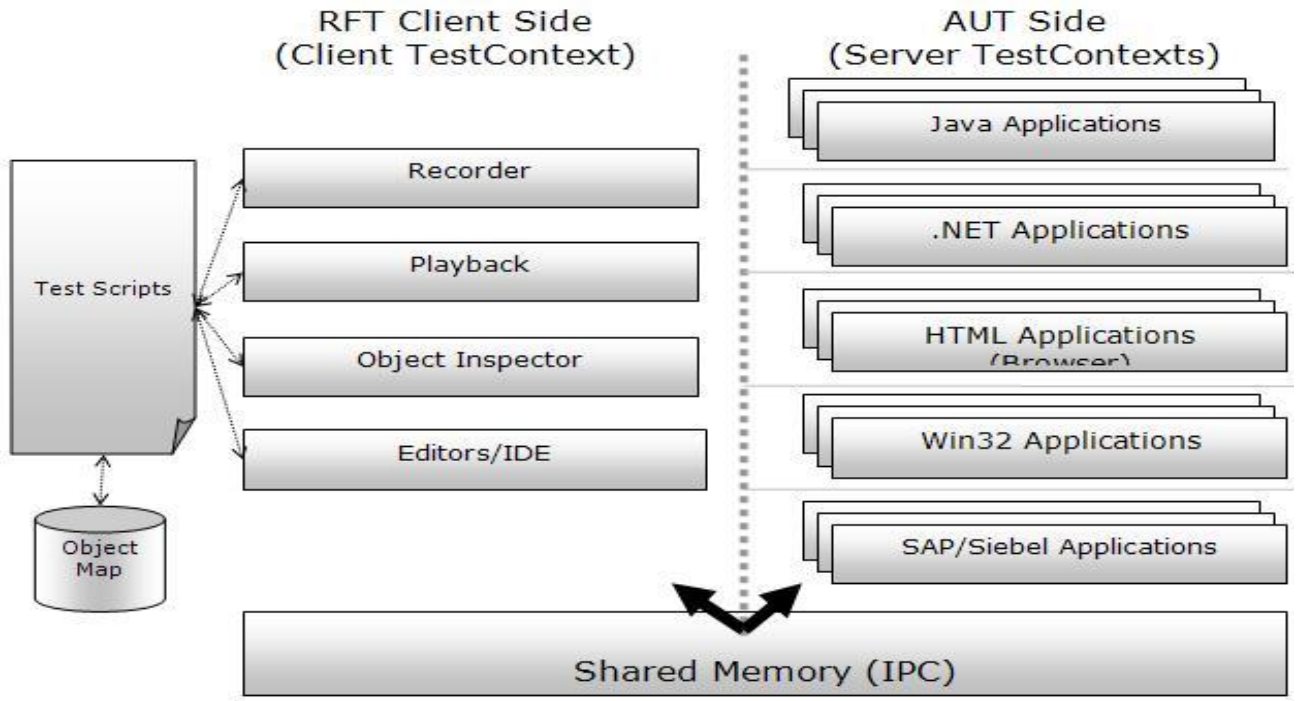
- AUT/SUT – 被测试系统或应用
- Domain(域) – RFT支持的环境
  - Java, .NET, HTML, Siebel, SAP, Win32 Controls
- Proxy(代理) – 被测试对象的包装(Wrapper)
- TestObject – Proxy的包装; 暴露在脚本中
- Server – 被RFT调用起来的SUT
- Client – RFT脚本端
- GUI – 图形用户界面
- Transaction(事物) – 录制/回放一个用户操作

概览

- 自动化功能测试工具
- 支持Java, .NET, HTML, Siebel, SAP和Win32测试域
- 脚本上支持JAVA (Eclipse)和VB.NET (Visual Studio)语言

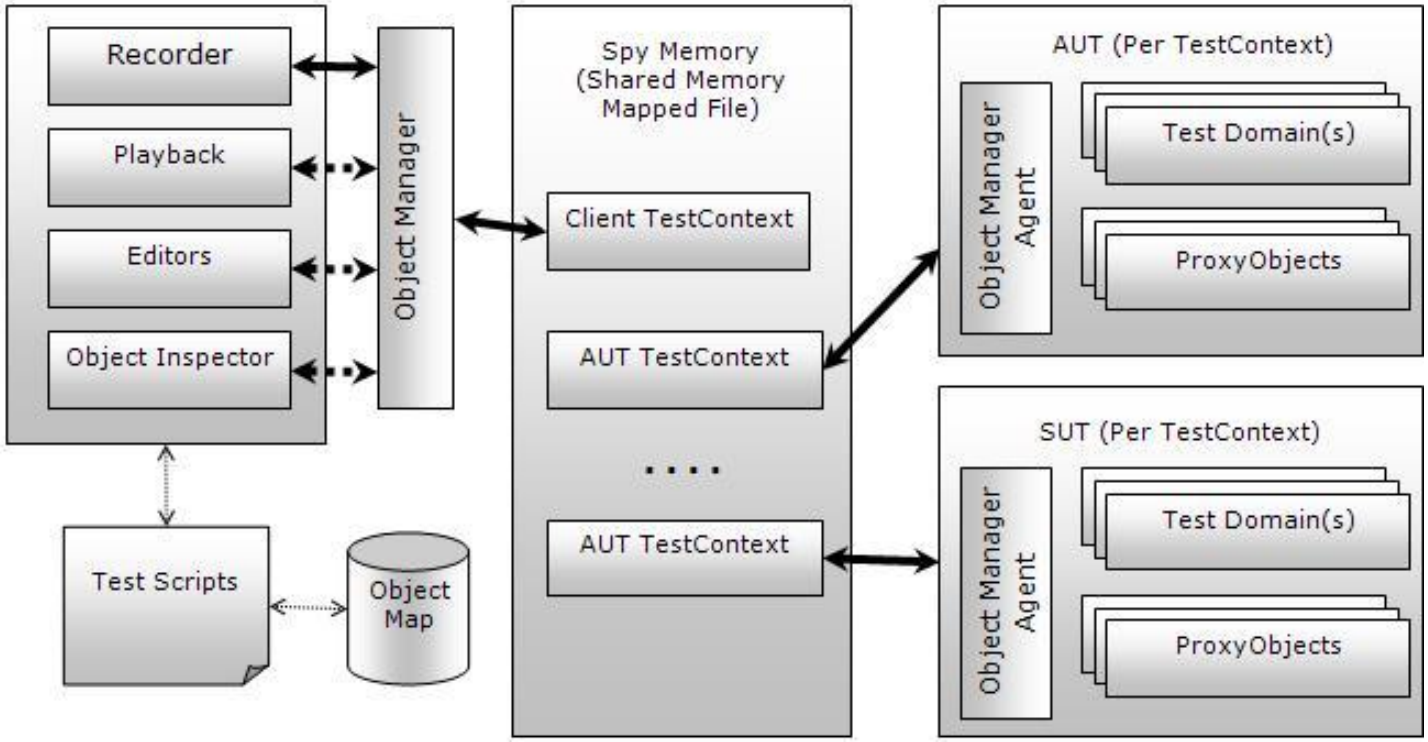


RFT 架构


RFT Client Side
(Client TestContext)
AUT Side
(Server TestContexts)
Java Applications
.NET Applications
HTML Applications
Win32 Applications
SAP/Siebel Applications
Recorder
Playback
Object Inspector
Editors/IDE
Test Scripts
Object Map
Shared Memory (IPC)

RFT 架构

- RFT对于每个进程(客户端或者服务端)生成了一个TestContext对象 并把它注册到共享内存区中(shared memory).
- 被注册的TestContext用于相应的进程间通讯的参照.



RFT 架构

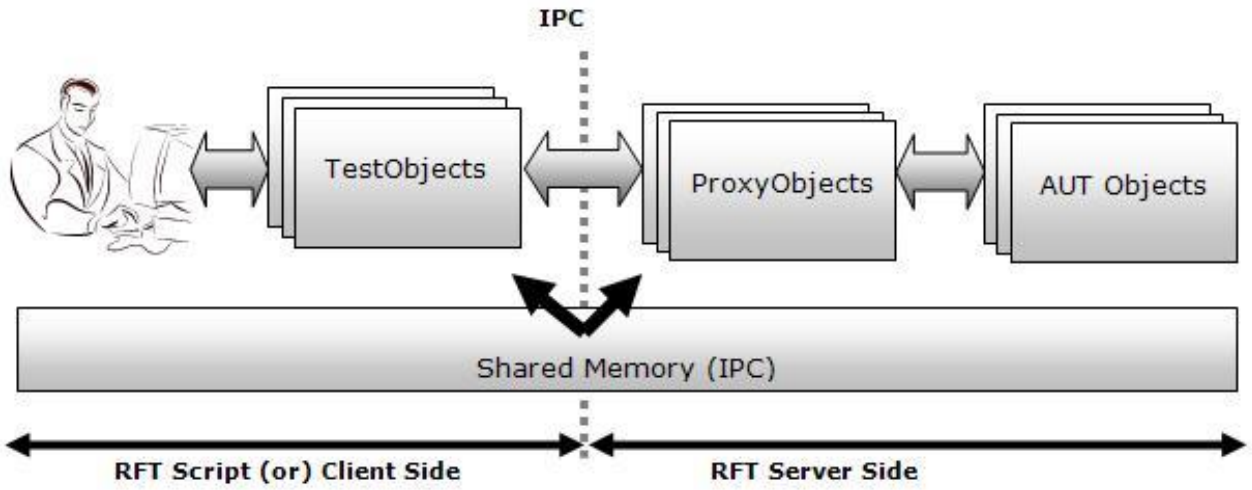
- TestContext:作为客户端和服务端进程间信息传递的传导. 客户端进程的TestContext如记录器,回放进程.

- ObjectManager: 管理从RFT服务端的通讯. 所有的录制和回放和AUT的核心交互都在这里发起. 和所有的被测试应用的test contexts交互.

- ObjectManagerAgent: ObjectManager类在被测系统中的体现. 帮助处理特定的Test Context下ObjectManager的操作.

ProxyObjects

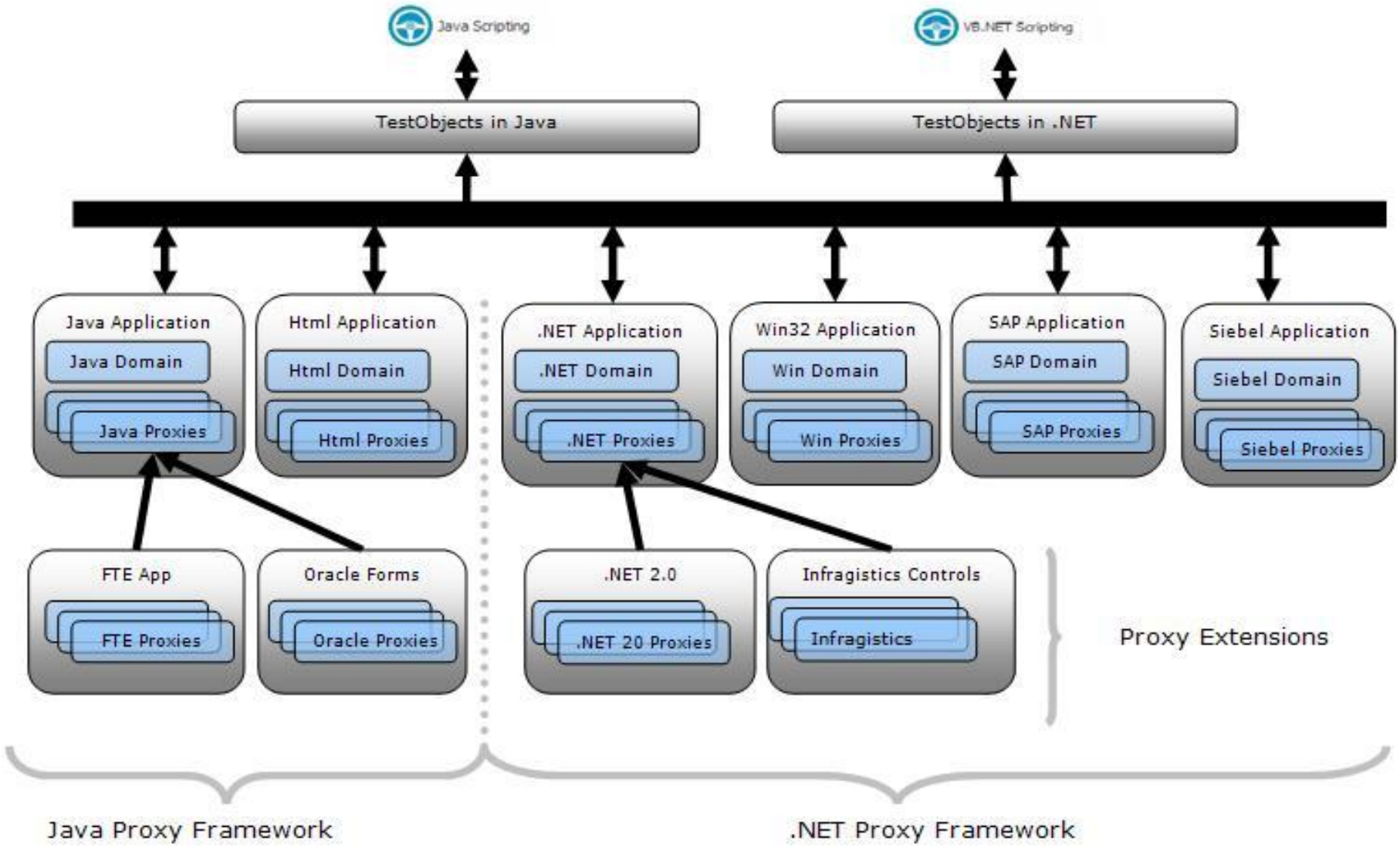
- 封装了单个的被测试对象
  - hWnd, Java control, COM Object, DOM element, Process etc.,
- 运行在被测试对象的运行环境中
- 针对需要交互的每个被测试对象动态生成
- 在被测试对象访问的地方生成(进程/线程)
- RFT交易结束后终止
- Proxy Object的方法是RFT框架预先定义的
- Proxy实现取决于被测试对象和它所在的域



TestObjects(GuiTestObjects,TopLevelTestObjects)

- 被测试对象的用户界面接口
- 用Java (Java scripting)和.NET (VB.NET Scripting)实现
- 对ProxyObject的封装
- 对象图中的节点
- 将对它的操作转给ProxyObject

TestObjects(GuiTestObjects,TopLevelTestObjects)



Proxy/Test Objects的关系

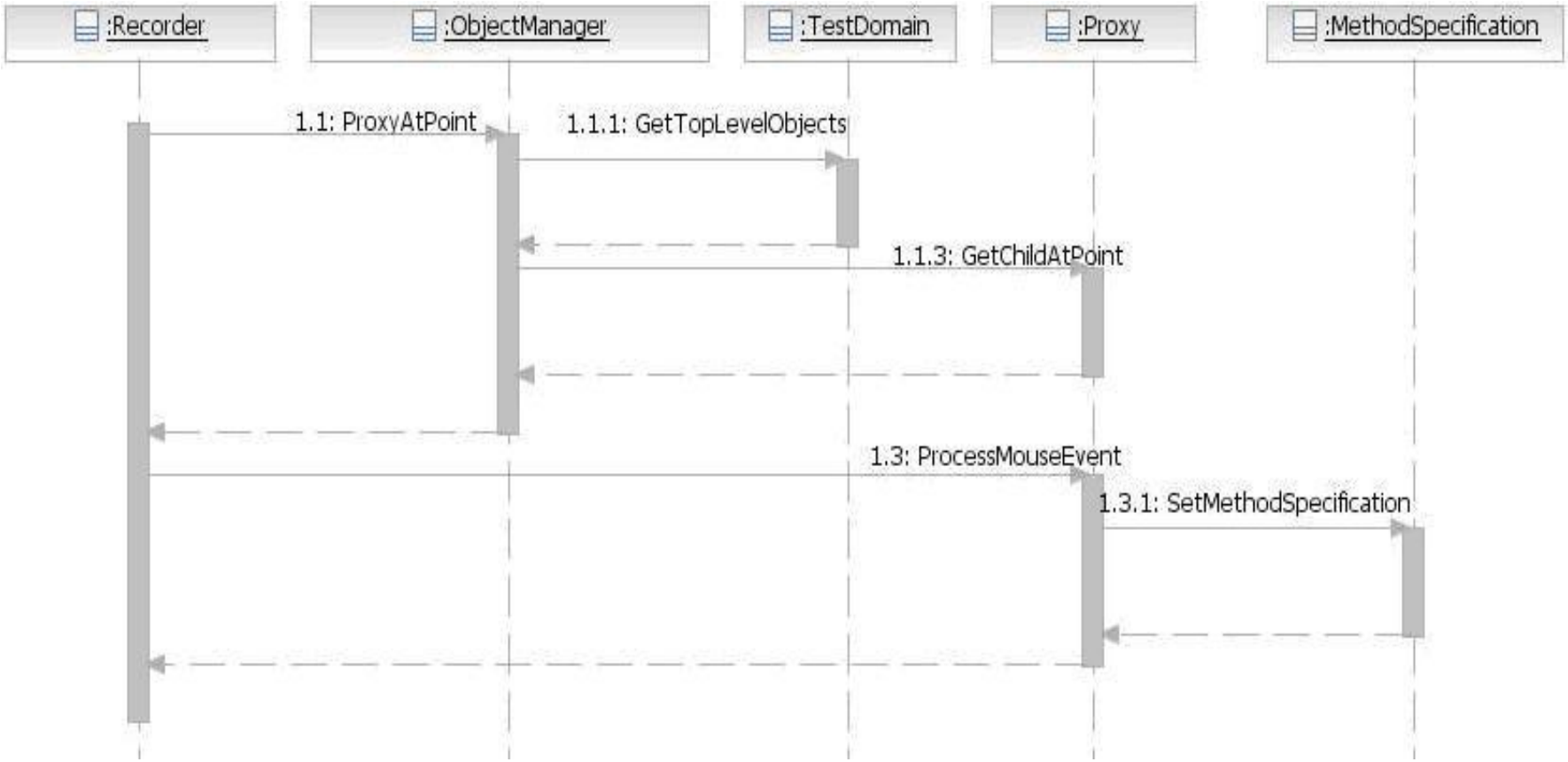
- Proxy Object是存在在被测试的环境中
- Proxy Object是实现于Domain领域.
- Proxy Objects是RFT根据需要动态建立的
- Proxy Objects和SUT Object是一一对应的
- TestObject在RFT的客户端

ProxyObject和TestObject的映射关系

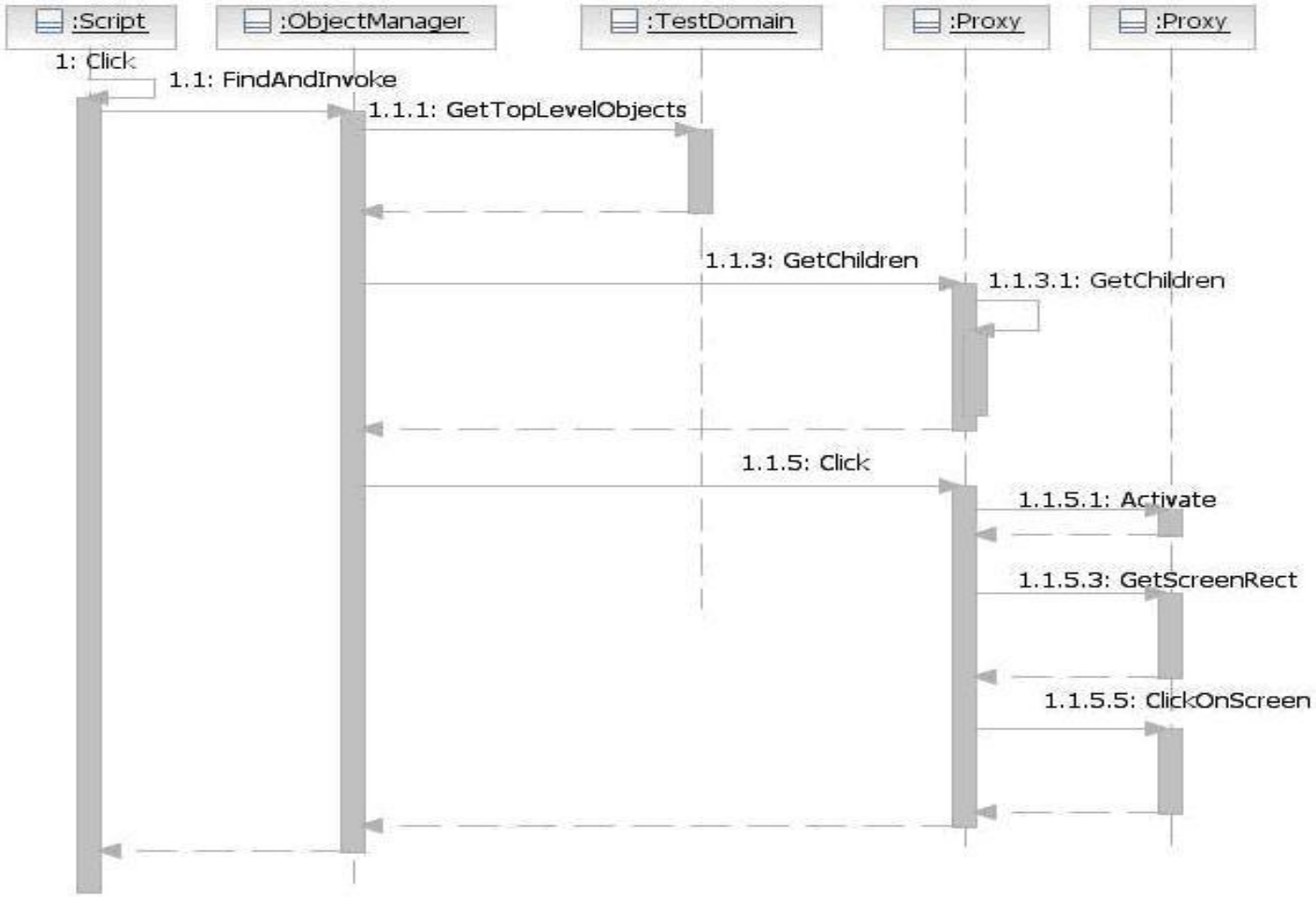
- 映射信息是通过存放在RFT安装目录下的customization (*.rftcust)文件实现的. (主要的mapping文件是rational_ft.rftcust)

```
<Obj L=".Proxy">
<ClassName>[WhidbeyControls]Rational.Test.Ft.Domain.Net.DataGridViewProxy</ClassName>
        <Replaces/>
        <UsedBy>[System.Windows.Forms]System.Windows.Forms.DataGridView</UsedBy>
</Obj>
```

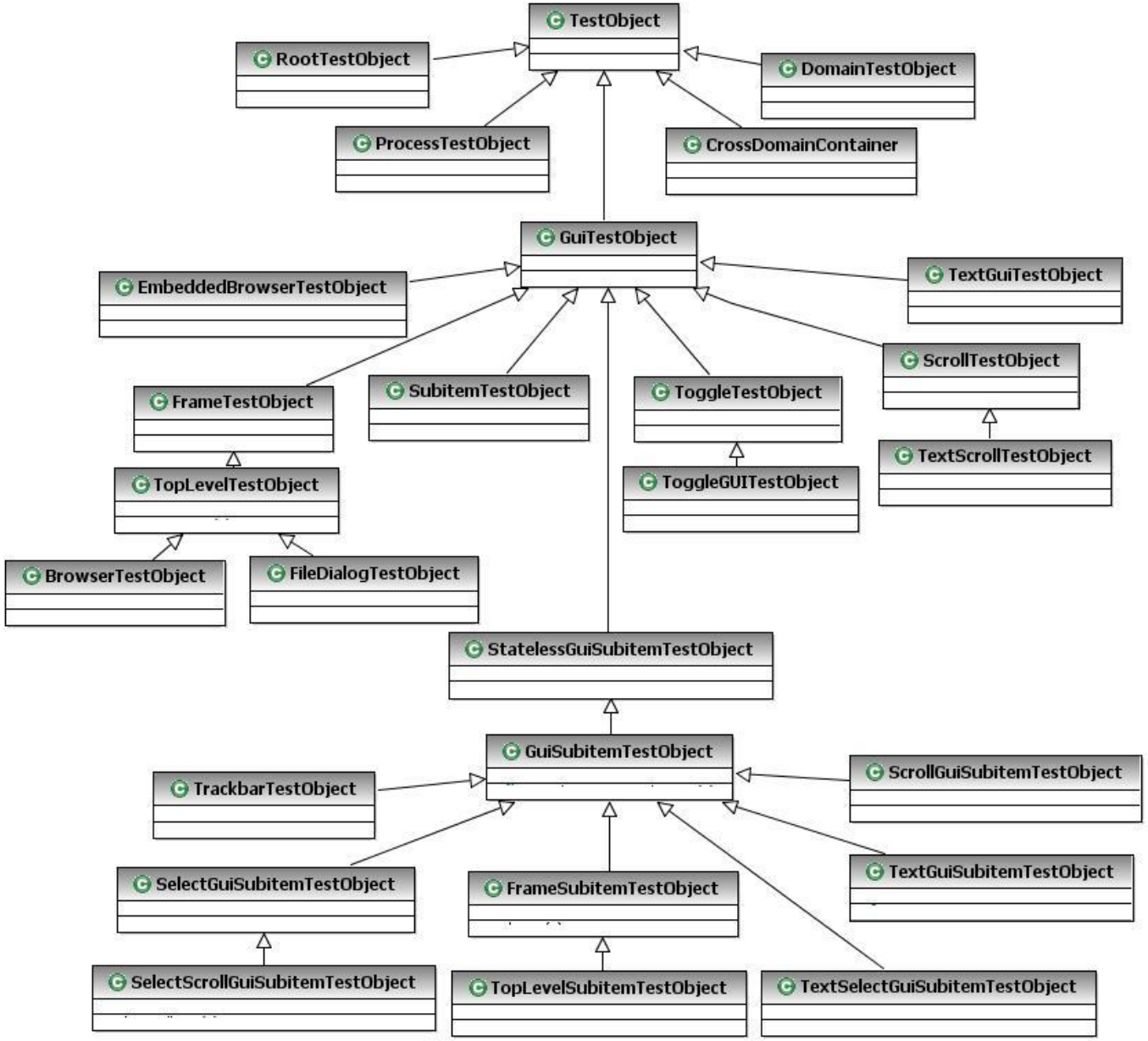
录制阶段被测试系统的交互



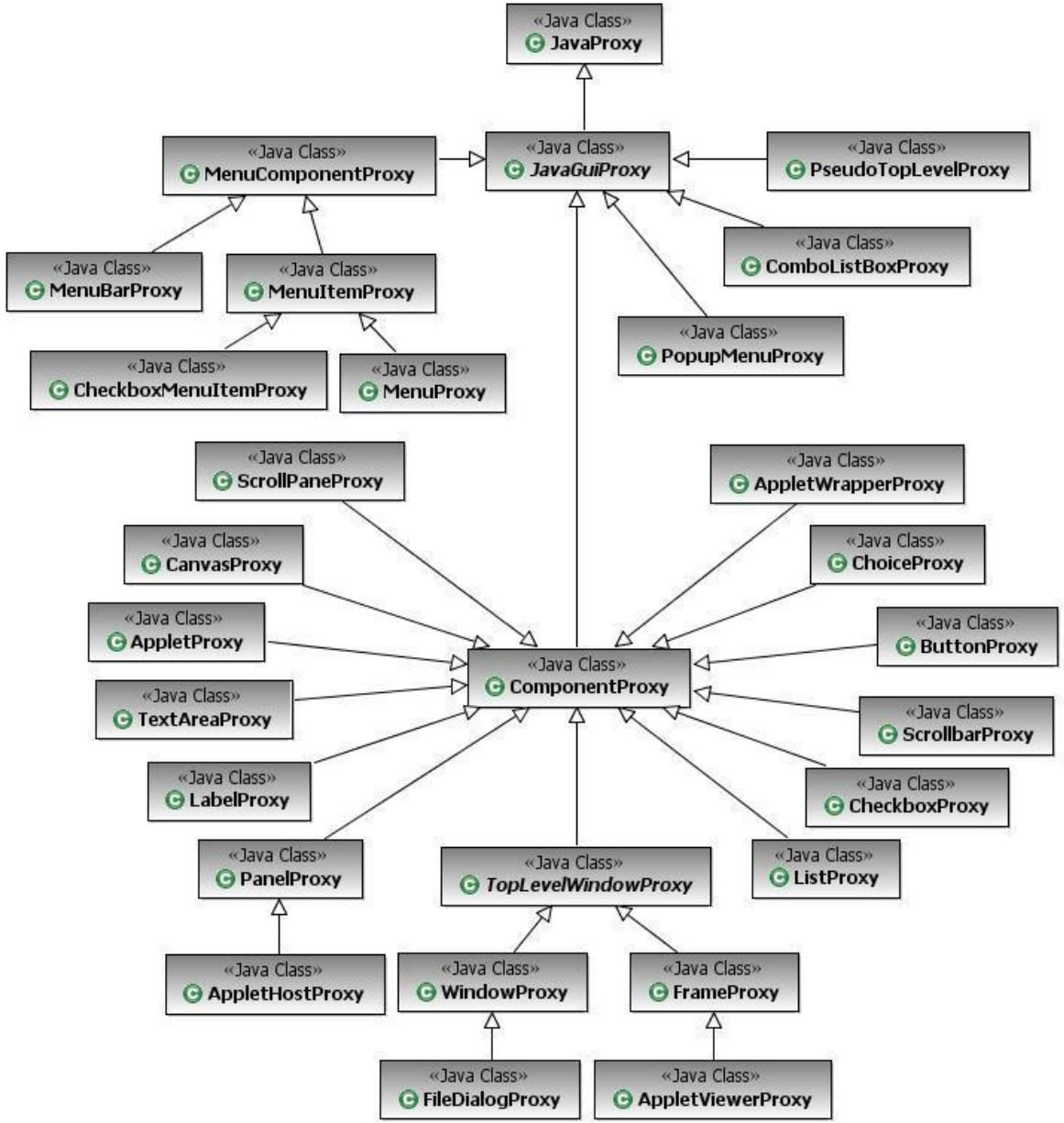
回放阶段的交互


:Script
:ObjectManager
:TestDomain
:Proxy
:Proxy
1: Click
1.1: FindAndInvoke
1.1.1: GetTopLevelObjects
1.1.3: GetChildren
1.1.3.1: GetChildren
1.1.5: Click
1.1.5.1: Activate
1.1.5.3: GetScreenRect
1.1.5.5: ClickOnScreen

TestObject

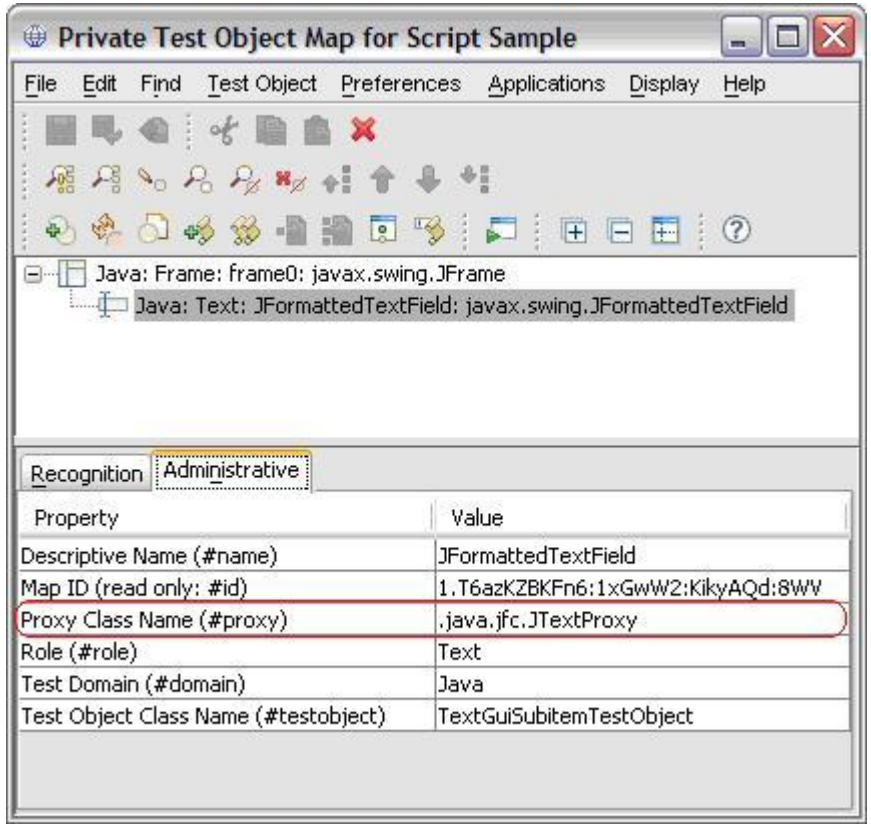

TestObject
RootTestObject
DomainTestObject
ProcessTestObject
CrossDomainContainer
GuiTestObject
EmbeddedBrowserTestObject
TextGuiTestObject
ScrollTestObject
FrameTestObject
SubitemTestObject
ToggleTestObject
TextScrollTestObject
TopLevelTestObject
ToggleGUITestObject
BrowserTestObject
FileDialogTestObject
StatelessGuiSubitemTestObject
GuiSubitemTestObject
TrackbarTestObject
ScrollGuiSubitemTestObject
TextGuiSubitemTestObject
SelectGuiSubitemTestObject
FrameSubitemTestObject
SelectScrollGuiSubitemTestObject
TopLevelSubitemTestObject
TextSelectGuiSubitemTestObject

AWT Controls Proxys



在什么地方能看到Proxy类?

- 查看用到了什么proxy



理解目前的类结构

- AUT控件(TestObject)的层次结构
- RFT的映射表
- ProxyObject的层次结构

javax.swing.JFormattedTextField (UI Control's) inheritance hierarchy

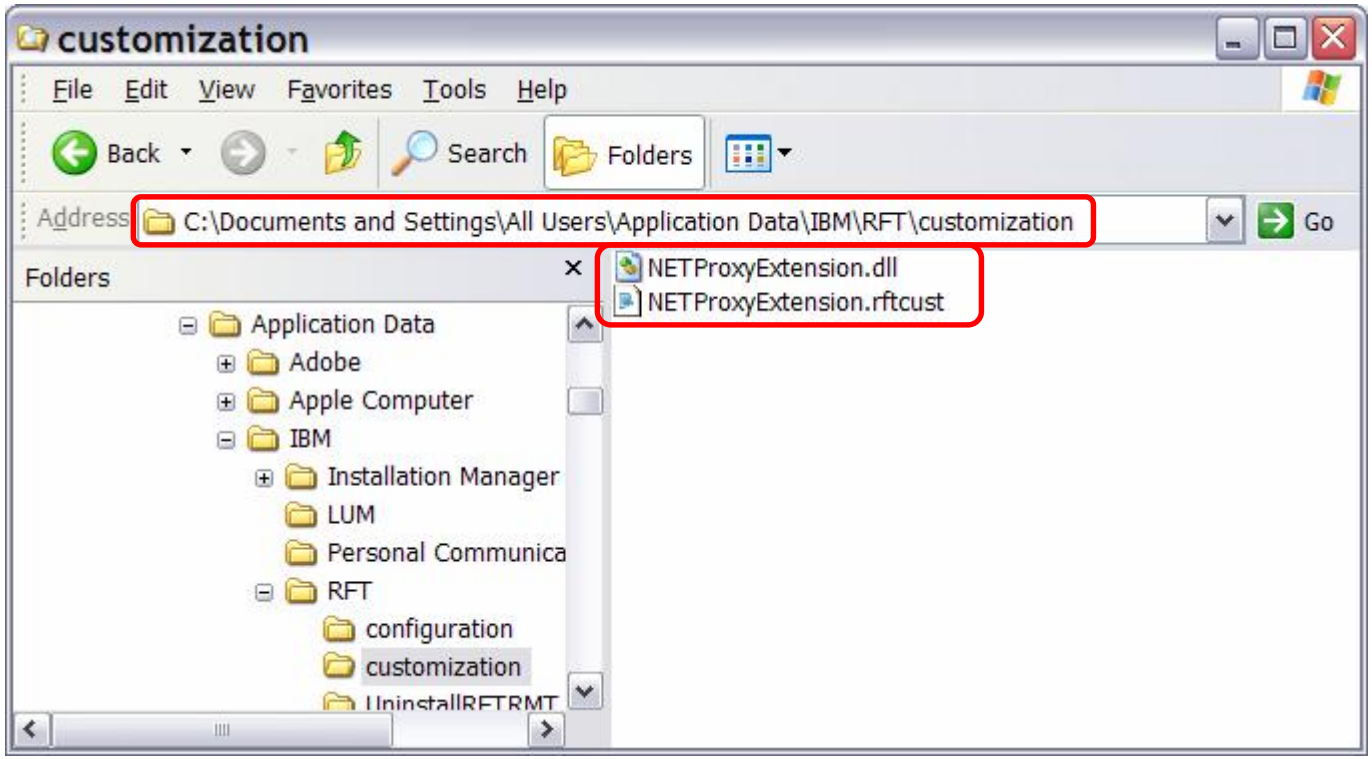
```
java.lang.Object
    java.awt.Component
        java.awt.Container
            javax.swing.JComponent
                javax.swing.text.JTextComponent
                    javax.swing.JTextField
                        javax.swing.JFormattedTextField
```

```
<Obj L=".Proxy">
      <ClassName>com.rational.test.ft.domain.java.jfc.JTextProxy</ClassName>
      <Replaces/>
         <UsedBy>javax.swing.JEditorPane</UsedBy>
         <UsedBy>javax.swing.JTextArea</UsedBy>
         <UsedBy>javax.swing.JTextField</UsedBy>
         <UsedBy>javax.swing.JPasswordField</UsedBy>
         <UsedBy>javax.swing.JTextPane</UsedBy>
</Obj>
```

JTextProxy (RFT ProxyObject) inheritance hierarchy

```
ProxyTestObject
  JavaProxy
    JavaGuiProxy
      awt.ComponentProxy
        jfc.JComponentProxy
          jfc.JfcGraphicalSubitemProxy
            jfc.JScrollPaneProxy
              jfc.JTextProxy
```

部署


customization
File Edit View Favorites Tools Help
Back
Search
Folders
Address
C:\Documents and Settings\All Users\Application Data\IBM\RFT\customization
Go
Folders
NETProxyExtension.dll
NETProxyExtension.rftcust
Application Data
Adobe
Apple Computer
IBM
Installation Manager
LUM
Personal Communica
RFT
configuration
customization

Agenda

- 自动化测试的原理
- 自动化测试框架
- 如何扩展
- **自动化测试的实施探讨**

自动化测试误区

为什么自动化的功能测试始终无法执行?

- 期望:

"我们今年购买了自动化的功能测试工具,测试团队不需要5个人了,我们计划1个月后把3位同事转到开发部门……."

"测试工具能覆盖100%的流程"

我们的期望

- 工具能够做所有事
- 工具能够帮助我们
  - 让自动化测试帮助我们覆盖尽可能多的流程,减少手工的干预
  - 工具能够自动的帮助我们统计数据,生成报告
- 花最少的钱做最多的事

功能测试的现状

- 一般分为常规测试和创造性测试
  - 常规测试是指按照功能说明书针对正常的流程和应有的出错流程而展开的测试
  - 创造性测试指依靠业务的经验,组合相关的测试用例,达到发现常规测试所无法发现的缺陷而进行的测试
- 测试团队中更多的人侧重于业务的理解和应用的理解
- 时间的开销
  - 重复性测试
  - 异常流程的测试
  - 创造性测试

自动化功能测试实质

- 自动化功能测试有自己的适用范围
  - 重复测试(回归测试),数据驱动测试,正常流程的测试
  - 无法覆盖100%流程
  - 数据准备

- 自动化测试的作用在于不断的增强测试人员的能力,从而帮助我们覆盖更多的流程,更多的重用,更科学的测试,从而更好的保证我们软件的质量.
  - 短期内不能帮助您减少测试团队成员
  - 改变测试团队的技术水平

- 自动化的功能测试往往由手工的功能测试工具辅助.

回顾期望,定义目标

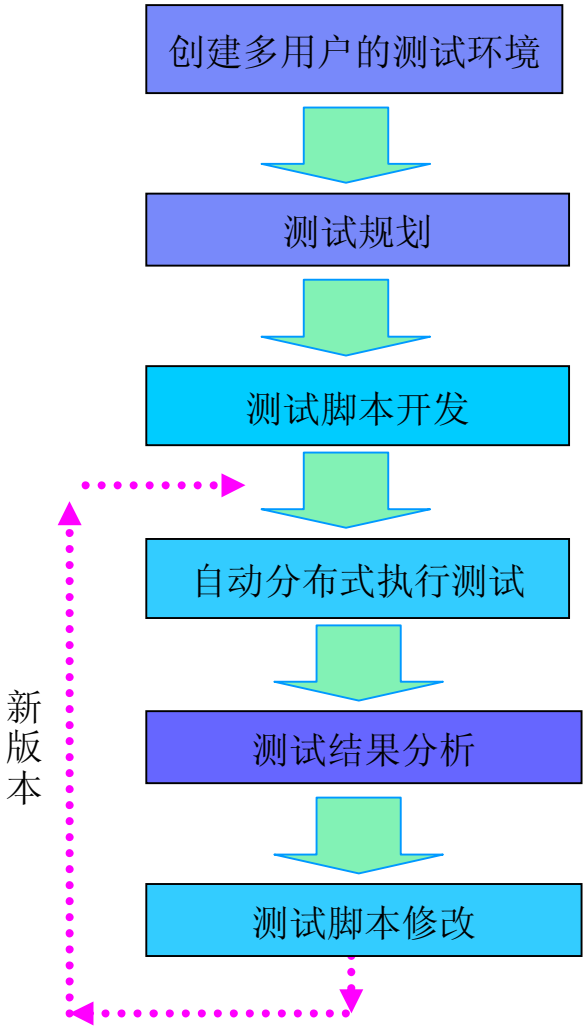
- ~~工具能够做所有事~~

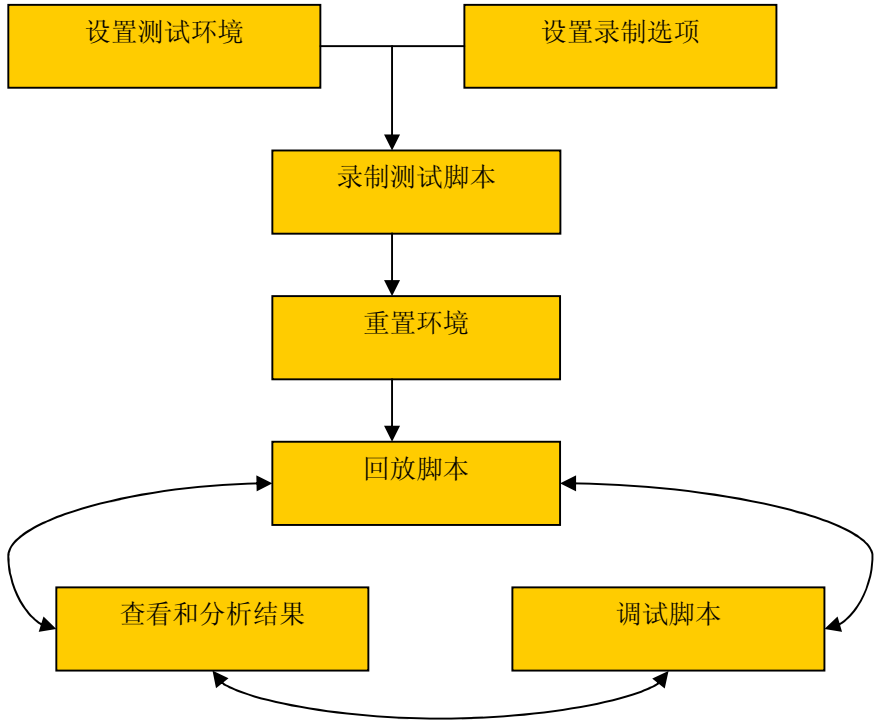
- 工具能够帮助我们
  - 让自动化测试帮助我们覆盖尽可能多的流程,减少手工的干预
  - 工具能够自动的帮助我们统计数据,生成报告

- 花最少的钱做最多的事

- **目标:不断改进和优化测试流程,积累测试资产,实现重用**
  - **重复性测试(自动化测试工具)**
  - **异常流程的测试(手工测试工具)**
  - **创造性测试(手工测试工具)**

自动化测试的技术实施过程



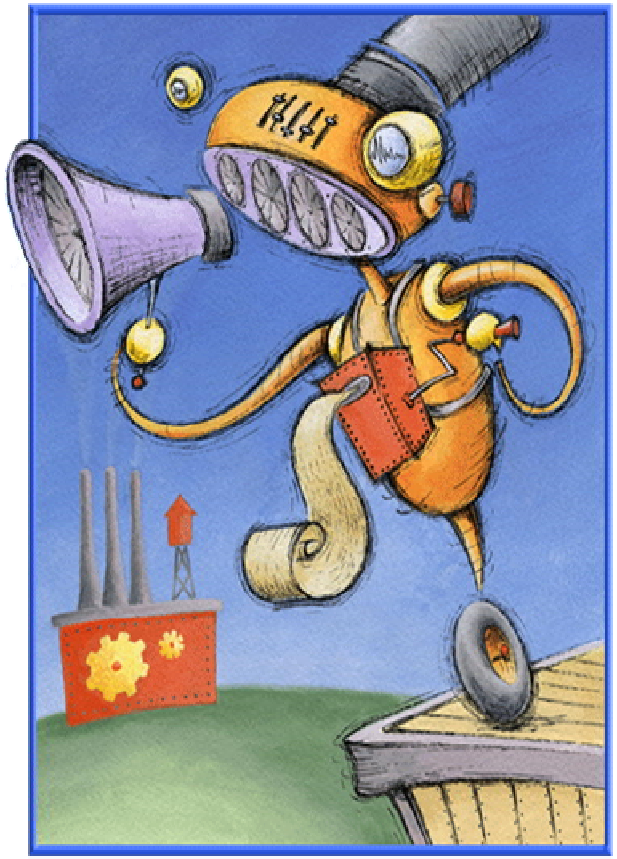


自动化测试的非技术实施准备

- 建立测试团队,明确角色和职责(技术测试人员和业务测试人员)
- 建立测试流程规范,并不断的改进
- 通过培训不断提高测试人员的水平
- 通过一些典型的项目切入,积累经验,制定计划,逐步推广
- 选取适合进行自动化测试的用例,逐步建立脚本库,对于不适合的流程,可以通过手工测试工具覆盖
- 建立命名规范,验证点约定,数据约定和重用模块约定
- 脚本库不断完善和全面,同时重用性也在提高
- 每周或每个阶段都需要提交项目质量报告和测试报告

根据阶段制定可行目标

逐步推广完善!



问题?

.....

...

..

.