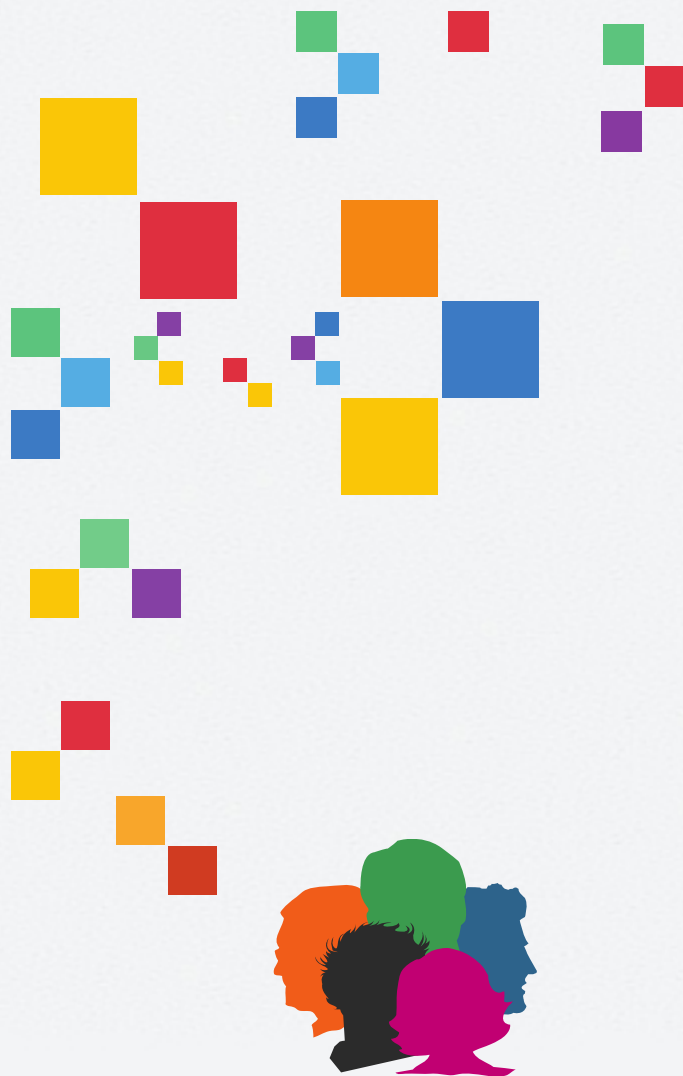


专业化测试项目实践





讲师介绍

郑兴宗

- 11年软件测试管理、自动化测试、性能测试，性能调优和测试咨询服务
- 前惠普高级技术顾问/测试架构师，在惠普从事7年技术咨询
- 为多家国内外世界500强和知名企业提供测试管理、自动化测试框架、性能测试、性能调优等培训、咨询和实施服务
- 精通TMMI测试成熟度模型，为客户提供卓越测试中心解决方案



目录

- 专业化测试提升产品质量与效率
- 自动化测试框架设计与实践
- 性能测试过程及性能调优实践



专业化测试提升产品质量与效率

软件测试体系：

- 软件测试基本概念和理论
- 软件测试流程
- 测试计划
- 测试策略
- 测试类型
- 测试用例
- 测试工具和管理平台（测试管理、自动化测试、性能测试和安全测试）
- 缺陷管理
- 测试报告
- 质量管理体系和测试成熟度



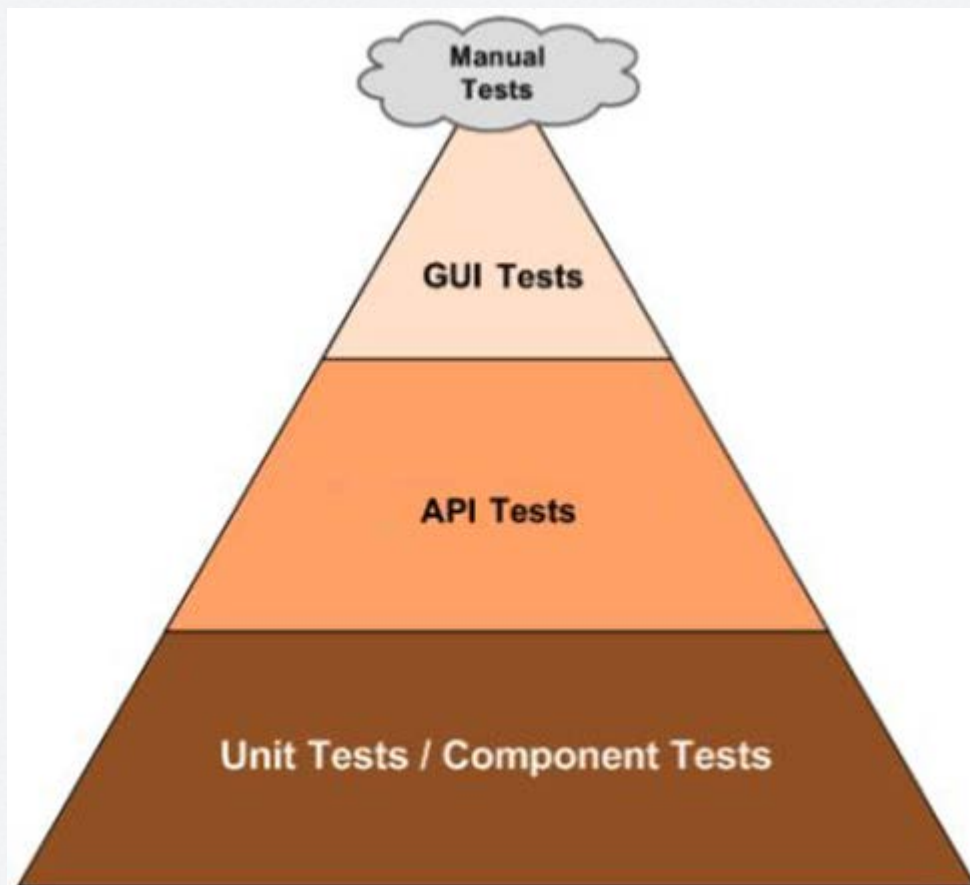
JPEG 图像

专业化测试提升产品质量与效率

专业化测试类型：

- 自动化测试
- 性能测试

测试金字塔：





专业化测试提升产品质量与效率

专业化测试流程:

- 接口自动化测试
- UI功能测试
- UI功能自动化测试
- 接口性能测试
- E2E性能测试



专业化测试提升产品质量与效率

自动化测试工具：

- UFT (QTP + Service Test)
- SoupUI
- Silk Test
- Rational Function Test

性能测试工具：

- Load Runner
- Jmeter
- Silk Performer
- Rational Performance Tester



专业化测试提升产品质量与效率

自动化测试：

- 提升软件测试效率
- 减少手工测试投入资源，把合适的资源投入到最优价值的地方
- 增加回归测试覆盖率
- 支持敏捷开发等高频回归测试

性能测试：

- 定位并解决产品性能问题，提高产品质量
- 提升用户体验，别让客户感觉到慢
- 减少资源浪费，投入更少的硬件资源，支持更大的用户量、业务量和数据量



自动化测试框架设计与实践



自动化测试框架设计

自动化测试框架设计原则

- 强大的执行引擎，真正做到无人值守
- 易维护的测试程序
- 能够管理可重用的公共资源和公共组件
- 能够调度/管理测试环境和测试资源
- 良好的测试结果生成和管理功能
- 业务测试人员和测试开发人员进行角色区分



自动化测试框架设计

测试框架组成部分

- 测试资源
- 测试组件及参数
- 测试用例及测试数据
- 测试用例代码生成及管理
- 测试用例集
- 测试执行调度器/触发器
- 测试执行收发和监控器
- 测试结果收集
- 测试报告生成

自动化测试框架设计

测试框架设计思路

- 测试分层：业务层，组件层和数据层分离
- 关键字驱动
- 公共测试资源、公共函数和场景恢复
- 公共测试组件，业务组件，参数化业务组件
- 测试用例：业务逻辑层调用业务组件和实例化测试数据
- 用例集组织和执行：组织测试用例集，支持测试执行调度器和触发器调用执行测试用例集
- 监控测试执行过程的事件和资源使用
- 自动收集测试结果，生成测试报告

自动化测试框架设计

测试框架设计逻辑





自动化测试框架设计

组件化开发

- 什么是关键字
- 关键字驱动的好处
- 关键字驱动脚本设计对人员的要求

低级别关键字

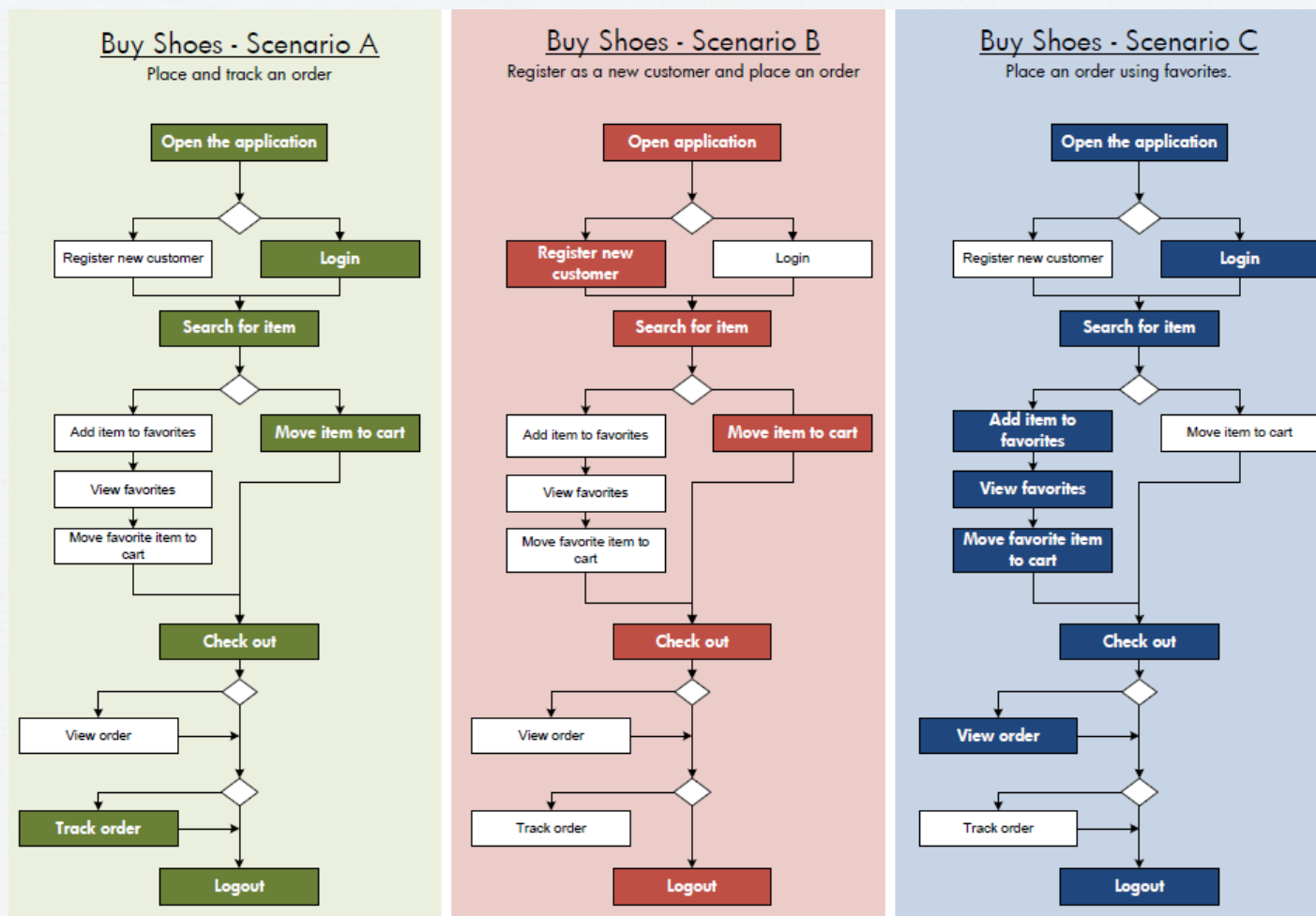
- 封装的关键字是接近测试步骤中的每个操作动作，就是用户的界面操作，例如：Click、Type、Select等。

高级别关键字

- 体现业务流程中的最基本单元业务操作，如登录、查询、下单等。
- 高级别关键字贴近业务，容易被手工测试人员、业务人员理解。
- 封装高级别关键字的组件，屏蔽底层操作，易用且重用度高，手工或业务测试人员只需要关注业务流和测试数据。

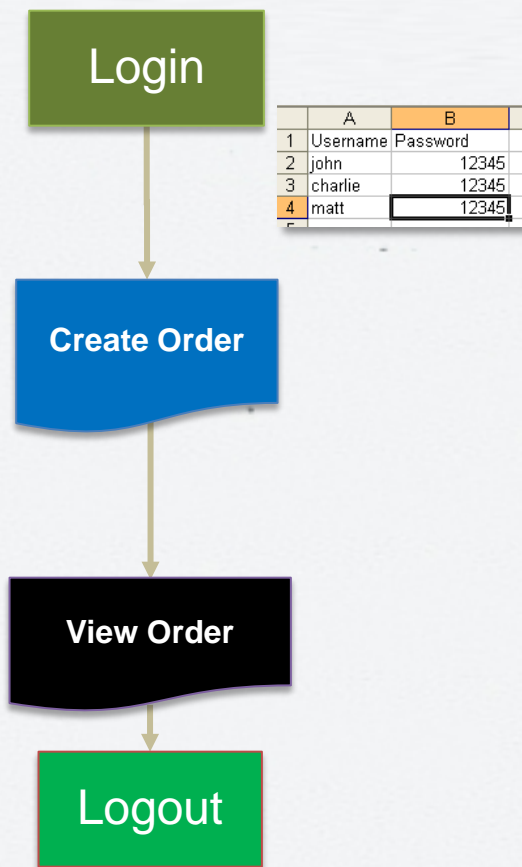
自动化测试框架设计

能够使用业务组件根据业务场景自由组合，并实例化测试数据

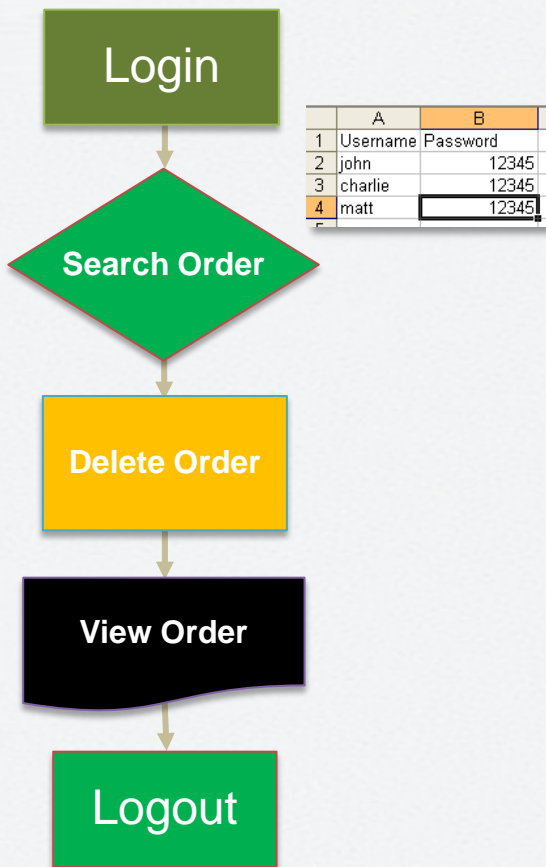


自动化测试框架设计

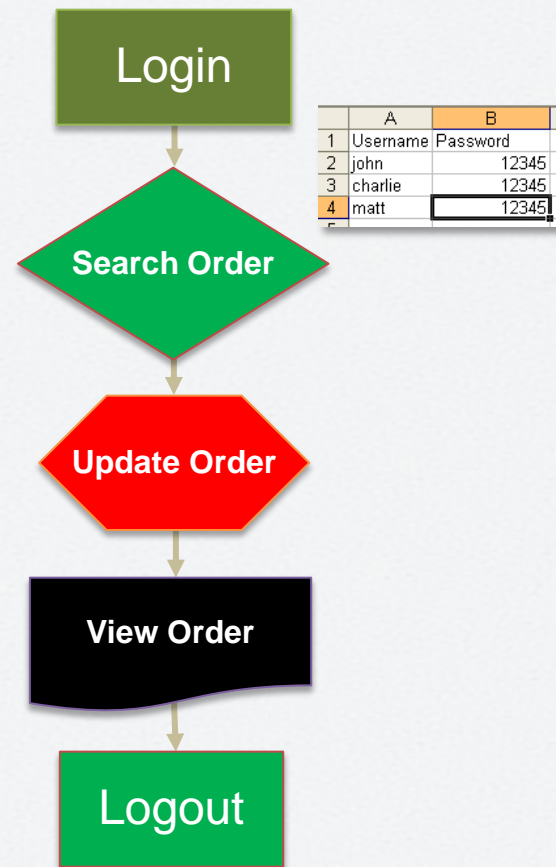
Create Order Test



Delete Order Test

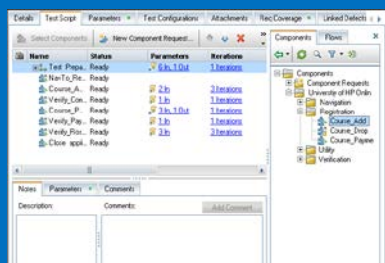


Update Order Test

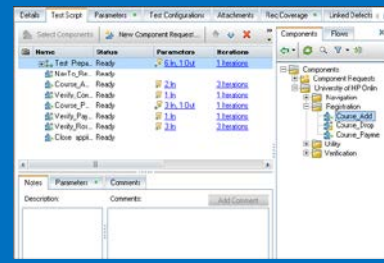


自动化测试框架设计

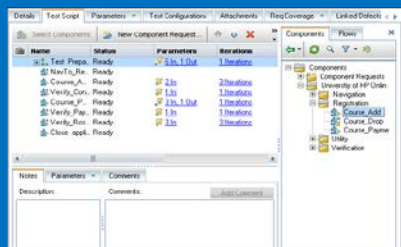
业务组件能够进行复用，当组件有变化时，实现一次修改，所有测试用例同时生效



测试用例1



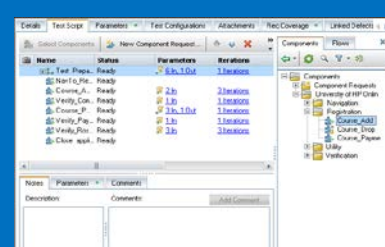
测试用例4



测试用例2

Step Name	Description	Expected Result
1	Open New Order Form	Click New Order Button
2	Step2 (auto generated)	Make the window active
3	Step3 (auto generated)	Click the "Register New Order" button
4	Enter Flight Details	Enter date, destination and departure cities

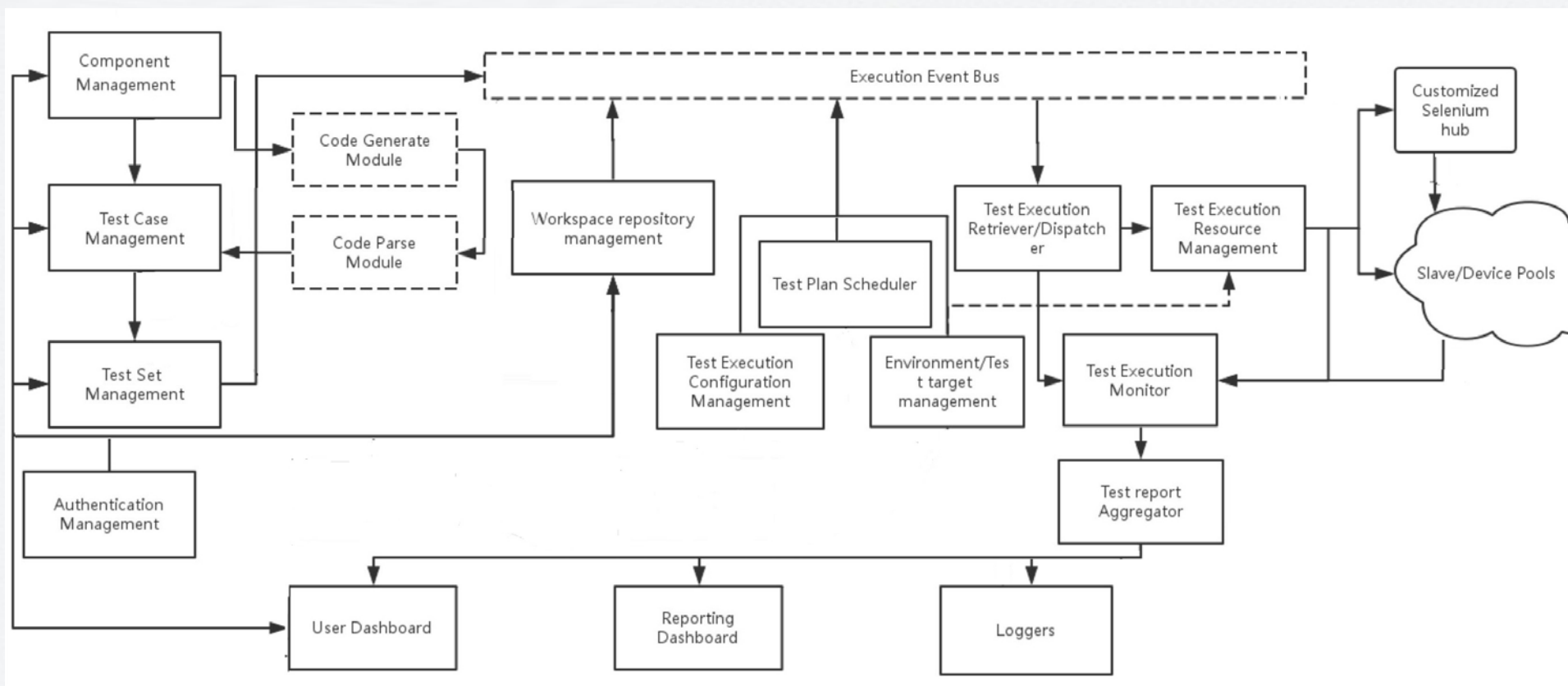
Update the component



测试用例3

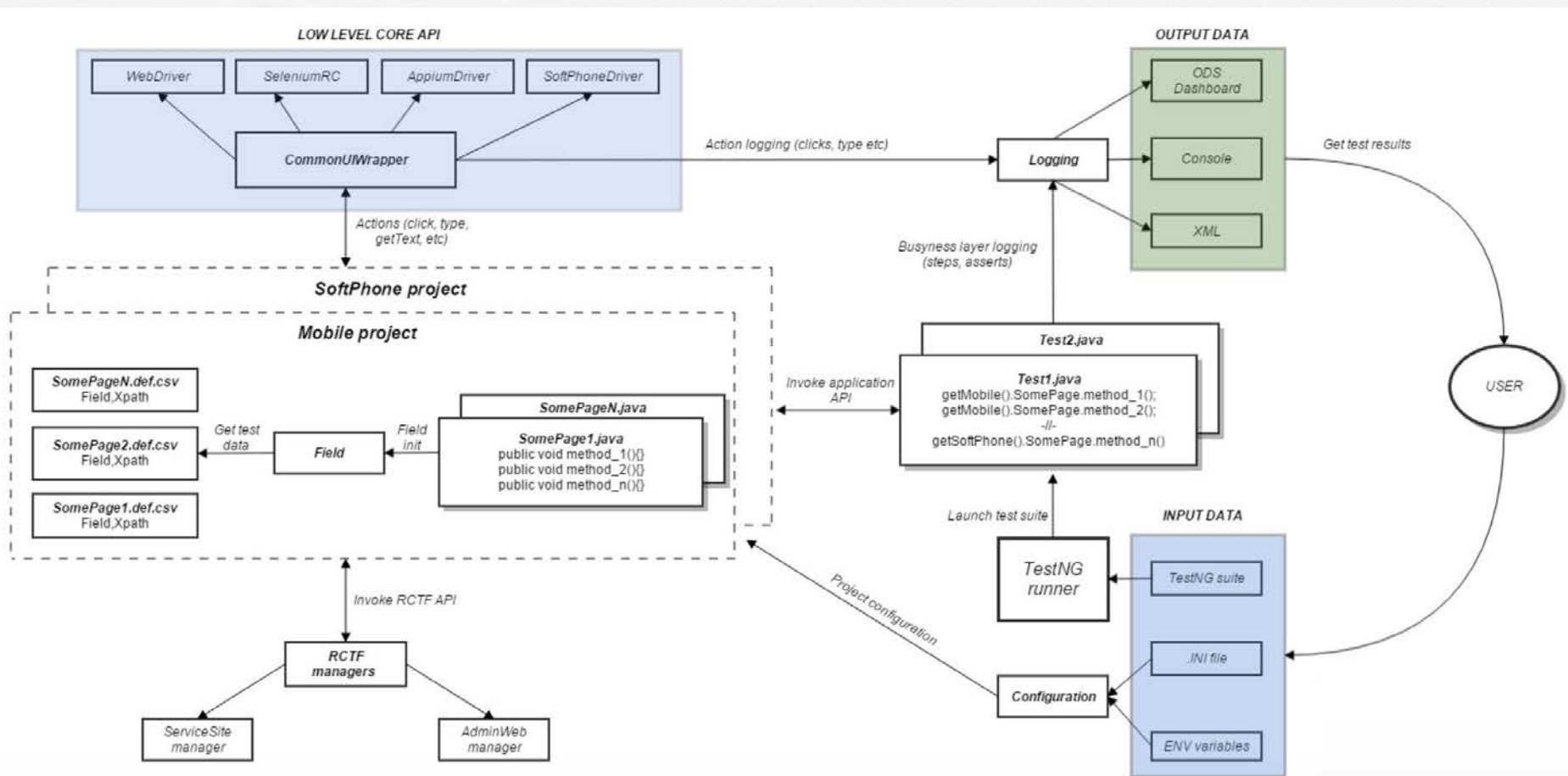
自动化测试框架实践

接口及UI自动化测试框架架构图



自动化测试框架实践

手机自动化测试架构图





自动化测试框架实践

项目级自动化测试框架

Test Management

Design

Preview

Test Plan

Bulk Update

Feature List

Env Script

Test Execution

Test Management: Service Web

DesignPreviewTest PlanBulk UpdateFeature ListEnv Script

root / loginMoveEditExecuteSubmit For ReviewHistoryDeleteBack

Login

Test Cases

Test Case: 'WEB-2346109': Login to SW with RC US account

WEB-2346109

acc(alias=Account1,scenario=swatBasicAccount,brand=1210,tier=4488)

ARC_US

Given [Login] account "Account1"

When [Login] set phone number

And [Login] set password

And [Login] submit

Then [Main Menu] is opened

自动化测试框架实践

- **Cucumber 是什么**
 - Cucumber 是 BDD 模式下实现可执行规范（Executable Specifications）的开源工具，但是它的使命并不局限于做自动化验收测试，更加重要的在于其能够在团队成员之间构建统一的交流基础（feature 文件）、规范交流用语（Domain Specific Language）、提高各个利益相关方（Business Stakeholders）沟通效率和效果，从而达到提升产品质量、做成客户期望得到的产品这一最终目标。
- **如何使用 Cucumber**
 - Cucumber 有很多种语言的实现版本，例如 Java、Ruby、.NET、JavaScript 等等，并且 Cucumber 可以和主流的测试框架很好地集成，常见的 Selenium、SpringFramework、Ruby on Rails 等，能够方便地引入到您的测试工作中去，几乎没有任何门槛。以 Java 测试项目为例，使用 Cucumber 的 Java 语言实现版本：Cucumber-JVM。

自动化测试框架实践

- 将 Cucumber-JVM 依赖加入到项目中
 - 如果您的项目是使用 Maven 管理所依赖的第三方依赖 jar 包，那么引入 Cucumber-JVM 将是一件优雅而且轻松的事情，只需要简单的将如下的 Code Snippet 加入到项目的 pom.xml 的 “dependencies” 下即可。<https://cucumber.io/docs>

Cucumber-JVM Report

Feature	Scenarios			Steps							Duration	Status
	Total	Passed	Failed	Total	Passed	Failed	Skipped	Pending	Undefined	Missing		
GL-14275	1	1	0	10	10	0	0	0	0	0	1m 34s 991ms	Passed
GL-14288	1	1	0	7	7	0	0	0	0	0	31s 070ms	Passed
GL-14304	1	0	1	11	2	1	8	0	0	0	25s 870ms	Failed
GL-14311	1	0	1	14	0	1	13	0	0	0	000ms	Failed
GL-14361	1	0	1	66	0	1	65	0	0	0	000ms	Failed
GL-14512	1	1	0	11	11	0	0	0	0	0	1m 42s 942ms	Passed

自动化测试框架实践

Cucumber 支持语言输出内容

1		feature		"功能"	
2		background		"背景"	
3		scenario		"场景"	
4		scenario outline		"场景大纲"	
5		examples		"例子"	
6		given		"*", "假如", "假设", "假定"	
7		when		"*", "当"	
8		then		"*", "那么"	
9		and		"*", "而且", "并且", "同时"	
10		but		"*", "但是"	
11		given (code)		"假如", "假设", "假定"	
12		when (code)		"当"	
13		then (code)		"那么"	
14		and (code)		"而且", "并且", "同时"	
15		but (code)		"但是"	



自动化测试框架实践

Cucumber描述测试用例

@PART-II-2

Feature: PART-II-2

@P0 @ios @realDevice @callStabilityTest

Scenario: PART-II-2:Receive incoming call, end call on caller side before accept it

Given [Welcome Page] Switch ENV To Dev

Given [Common Action Page] Sign In with RingCentral Phone Number "18552785561" Extension "703" And Password "Test!123"

Given [All Messages Page] Navigate To Settings

Given [Settings Page] Incoming calls section set Mute Incoming Calls Off

Given [Settings Page] Click Back Btn

Given [Phone Page] Kill app, then account "18552785561" password "Test!123" make call to user "703" and accept

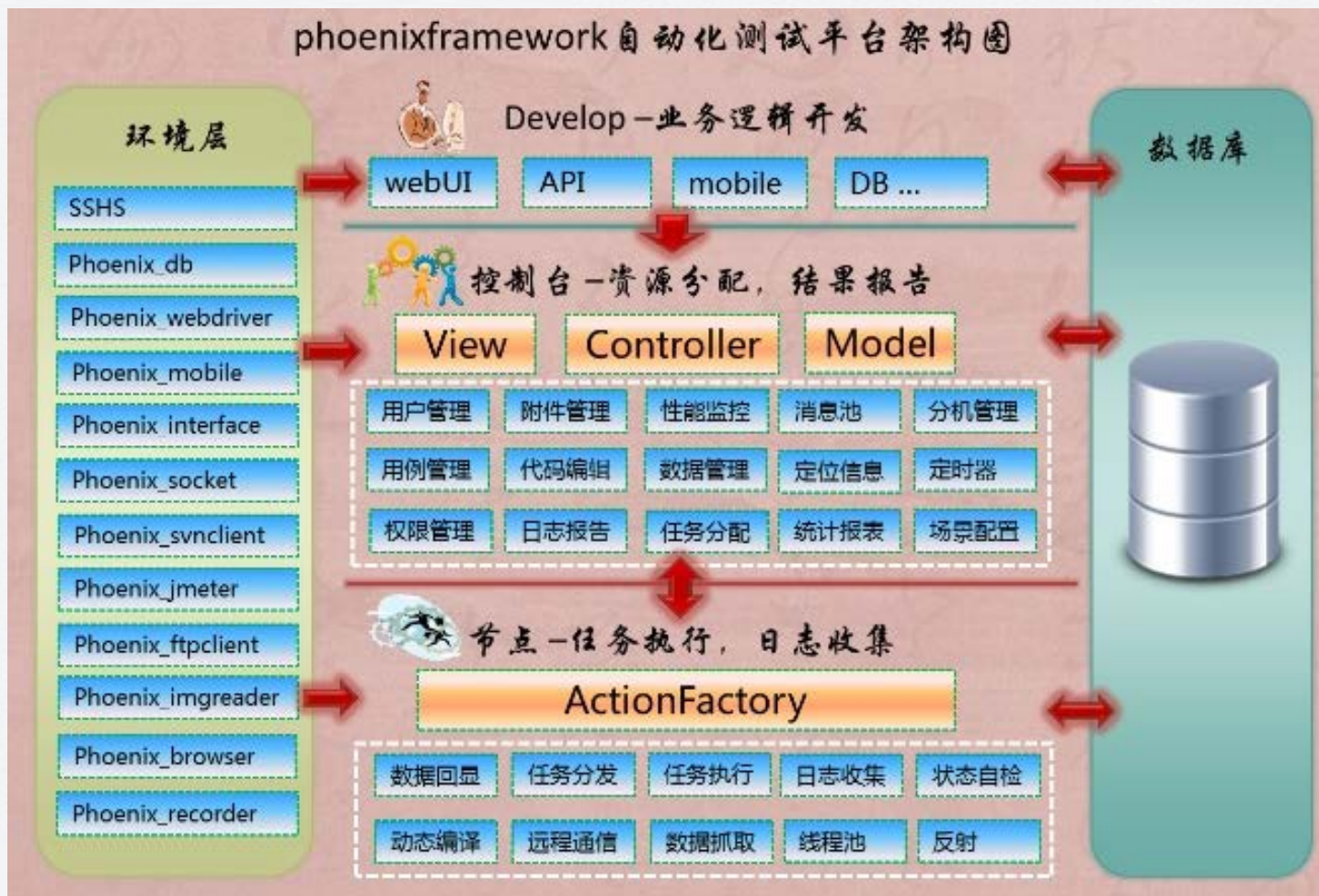
测试脚本实现

```
@Given("^\\[Common Action Page\\] Sign In with RingCentral Phone Number \"([^\"]*)\" Extension \"([^\"]*)\" And Password \"([^\"]*)\"")
public void sign_in_with_ringcentral_phone_number(String phone, String extension, String password) throws Exception {
    getGlipAuto().getLoginPage().clickSignInLink();
    getGlipAuto().getSignInPage().clickSignInWithRCBtn();
    getGlipAuto().getSignInAsRingCentralWebViewPage().waitPageLoaded(60);
    getGlipAuto().getSignInAsRingCentralWebViewPage().signInWithRingCentral(phone, extension, password);
    getGlipAuto().getEmergencyServicesDisclaimerWebViewPage().clickAccept(20);
    if (getGlipAuto().getAccountSetupWebViewPage().isPhoneNumberFieldDisplayed()) {
        getGlipAuto().getAccountSetupWebViewPage().inputPhoneNumber("16509375829");
        getGlipAuto().getAccountSetupWebViewPage().clickDoneBtn();
    }

    GlipWrapper.getInstance().getAlertDialog().clickOkAndAllowIfNeeded();
    getGlipAuto().getMessagesPage().waitPageLoaded(60);
}
```

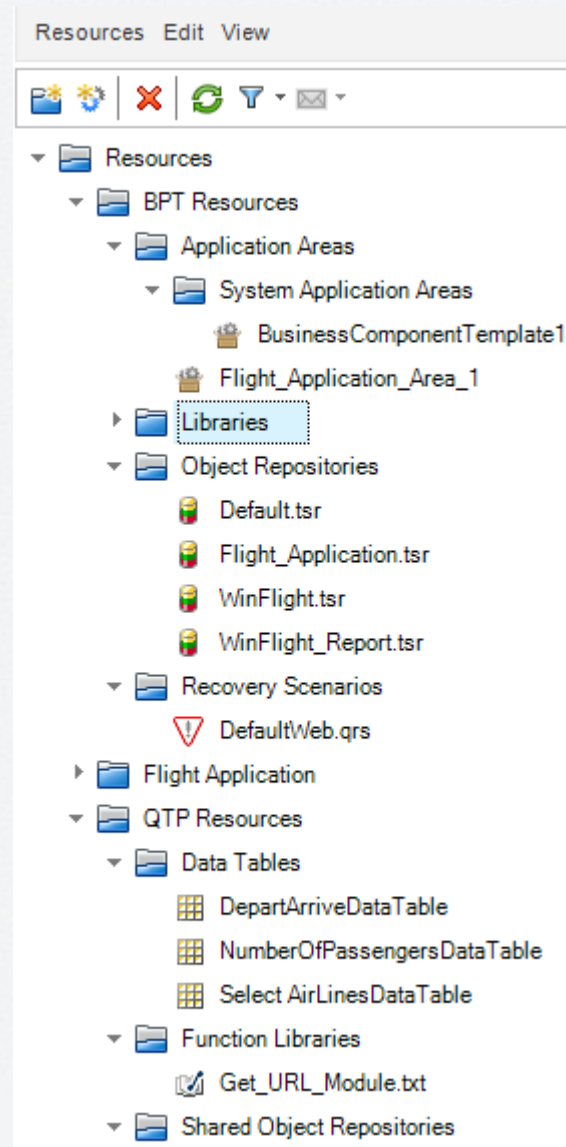
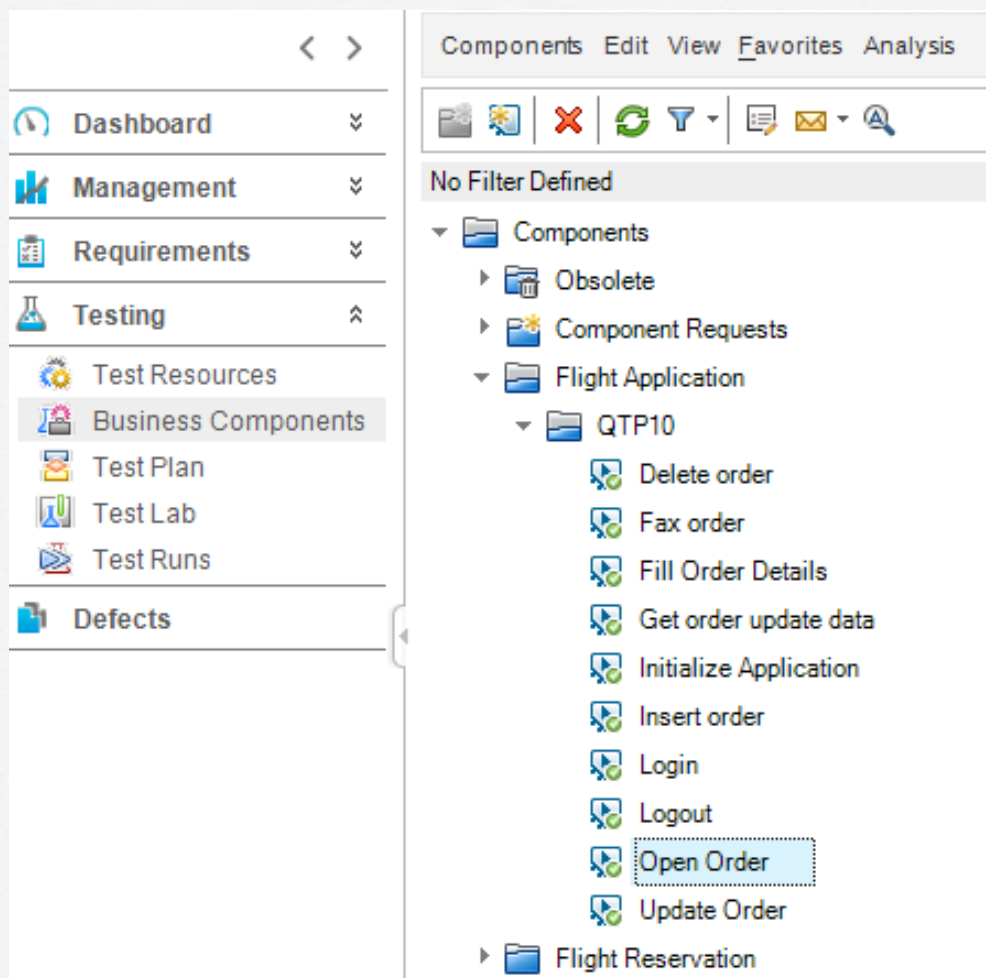

自动化测试框架实践

- Phoenix Framework 自动化测试平台



自动化测试框架实践

企业级自动化测试框架--BTP框架





自动化测试框架实践

企业级自动化测试框架--BTP框架

Defined

Name

- Subject
 - Unattached
 - Cruises
 - Flight Reservation
 - Book Flight
 - Address Options
 - Billing And Delivery Address
 - Book Flight
 - Book Flight Page
 - Book Flight Preparation
 - Create and delete order
 - Create order
 - Create Order Flow
 - Credit Card Expiration Date
 - Credit Card Number
 - Credit Card Owner
 - Delete Order
 - Departing And Arriving Location

Details | Test Script | Parameters | Attachments | Test Configurations | Req Coverage | Linked

Select Components | Canvas View

Name	Status	I/O Parameter
Login [1]	Ready	2 In
Group1		
Initialize Application [1]	Ready	
Open Order [1]	Ready	3 In, 1 Out
Delete order [1]	Ready	
Logout [1]	Ready	

Components | Flows

- Components
 - Component Requests
 - Flight Application
 - Business Component
 - Delete order
 - Fax order
 - Fill Order Details
 - Get order update data
 - Initialize Application
 - Insert order
 - Login
 - Logout
 - Open Order
 - Test1
 - Update Order



自动化测试框架实践

讨论：

对比传统开发模式和敏捷开发模式下的自动化测试



性能测试过程及性能调优实践



- 什么是性能测试？
- 为什么做性能测试？
- 如何做性能测试和性能调优？

什么是性能测试？

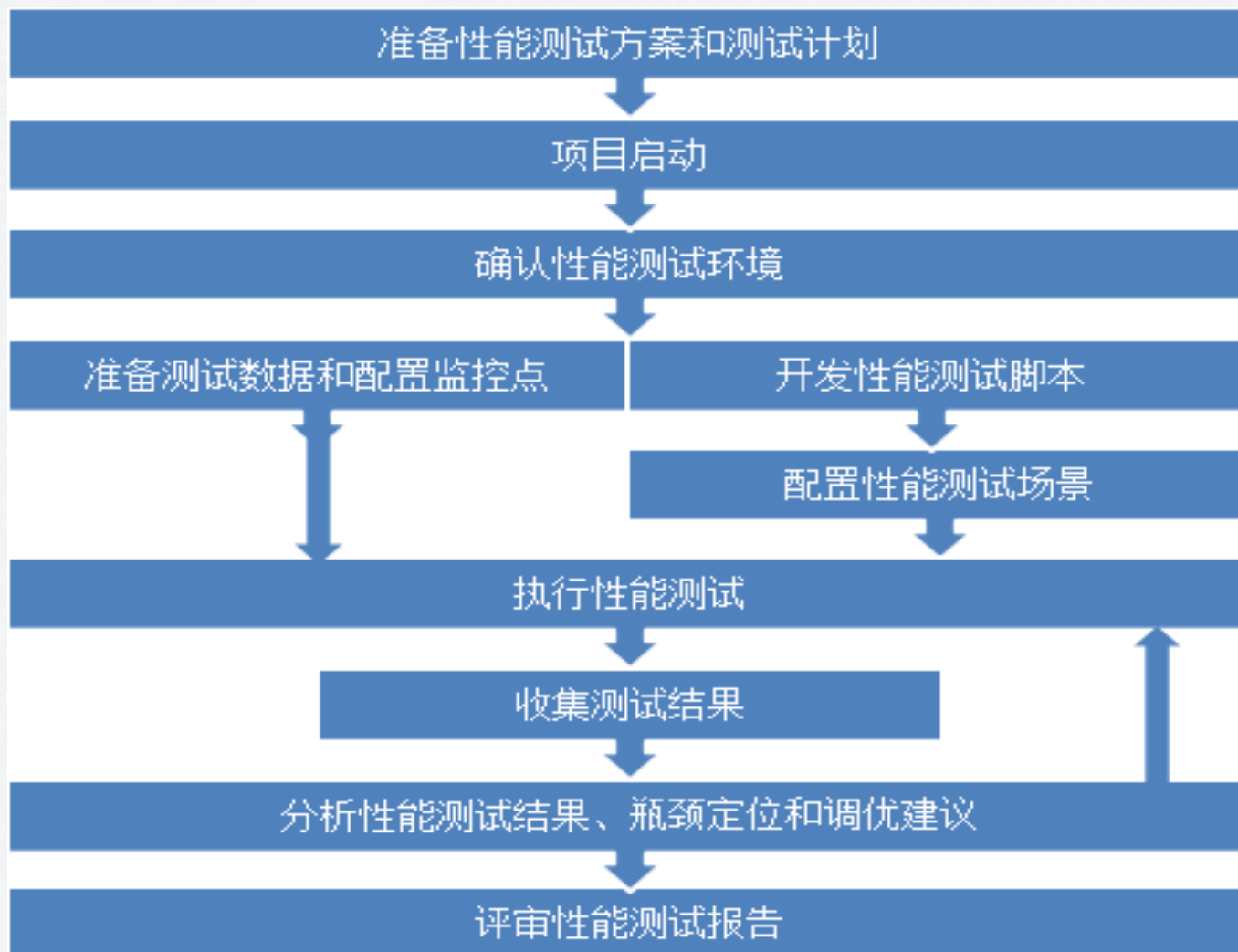
- 性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。基准测试、负载测试、压力测试、稳定性测试和容量测试都属于性能测试，可以根据实际场景结合进行。
- 基准测试：在用户数较小的压力情况下，系统的性能基准。
- 负载测试：确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。
- 压力测试：通过不断加压，确定系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。
- 稳定性测试：模拟生产环境真实高峰低谷时间段的情况下，不同的压力长时间运行，检测系统稳定性。
- 容量测试：获取系统能支持的最大用户容量、业务容量和数据容量。

为什么做性能测试？

- 评估系统的业务处理能力
- 找出系统的性能瓶颈点
- 为系统性能调优提供数据支持
- 验证系统的稳定性和可靠性
- 性能是用户体验中重要的指标
- 性能测试是验收测试中重要的部分之一
- 性能配置测试可以作为IT成本优化的重要手段
- 性能效率是质量模型中最重要的特性之一
- 系统的运营维护需要性能指标数据支撑
- 性能指标是项目指标中不可或缺的重要数据

如何做性能测试？

性能测试流程



如何做性能测试？

应用场景测试过程：


































性能测试类型：



如何做性能测试？

性能测试开发工具之Load Runner:

- 模块：脚本开发、场景执行和结果分析
- 支持很广泛通用的协议
- 支持C, .NET, Java等多种开发语言
- 能够集成Site Scope, Diagnostics, Performance Center等监控工具和性能测试平台

- +  Utility Functions and C Language Reference
- +  AJAX Click and Script Functions
- +  Action Message Format Vuser Functions (AMF)
- +  Citrix ICA Vuser Functions (CTRX)
- +  COM Vuser Functions (LRC)
- +  Database Vuser Functions (LRD)
- +  DNS Functions (MS_DNS)
- +  Flex Functions (FLEX)
- +  FTP Vuser Functions (FTP)
- +  Internet Messaging Functions (IMAP)
- +  Java over HTTP Vuser Functions
- +  Listing Directory Vuser Functions (MLDAP)
- +  Media Player Vuser Functions (MMS)
- +  MS Exchange Server Functions (MAPI)
- +  Multimedia Messaging Vuser Functions (MM)
- +  Oracle NCA Vuser Functions (NCA)
- +  .NET Vuser Functions
- +  Post Office Protocol Vuser Functions (POP3)
- +  RealPlayer Functions (LREAL)
- +  Remote Desktop Protocol Vuser Functions (RDP)
- +  SAP Click and Script Functions (SAP)
- +  SAP GUI Functions (SAPGUI)
- +  Simple Mail Transfer Vuser Functions (SMTP)
- +  Terminal Emulator - RTE Vuser Functions (TE)
- +  Tuxedo Vuser Functions (LRT)
- +  WAP Vuser Functions (WAP)
- +  Web (Click and Script) Vuser Functions (WEB)
- +  Web Vuser Functions (WEB)
- +  Web Services Functions (SOAP, WEB_SERVICE, Silverlight)
- +  Windows Sockets Vuser Functions (LRS)
- +  XML Functions (LR_XML)



如何做性能测试？

性能监控及瓶颈定位：

- 系统层
- 网络层
- 中间件应用层
- 业务逻辑层
- 数据库层

性能监控工具：

- 开源软件
- 商业化APM软件
- 云监控平台

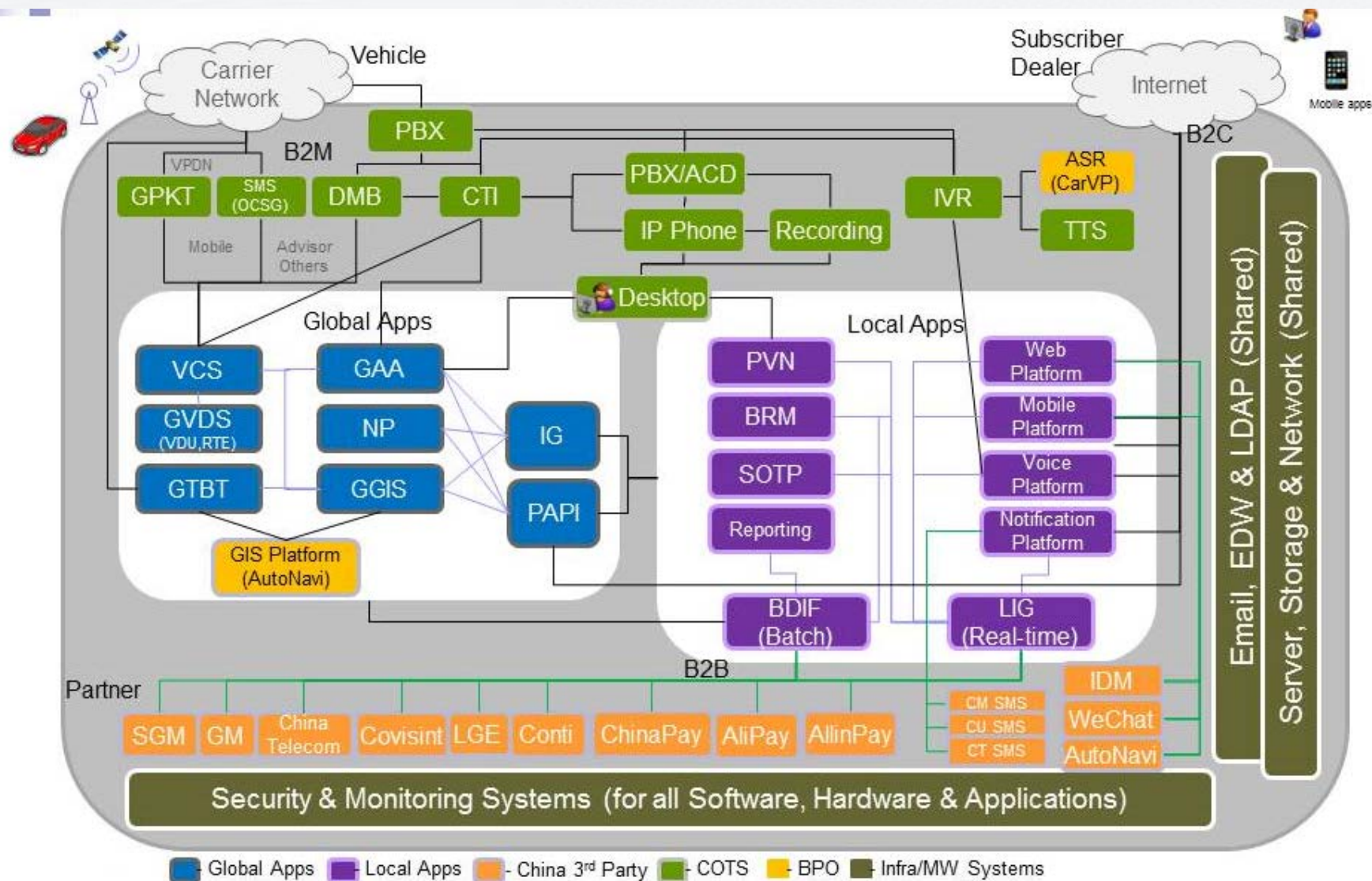


如何做性能测试？

性能测试过程交付内容：

- 测试策略
- 测试计划
- 测试场景类型
- 收集生产环境配置、用户量、业务量和业务场景
- 测试环境配置，与生产环境差异分析
- 性能测试范围、指标和调优目标
- 测试监控点分析和配置
- 性能测试脚本
- 测试场景配置
- 测试执行和结果收集
- 测试结果分析和瓶颈定位
- 测试问题解决及回归测试

性能测试案例分析



性能测试案例分析

性能调优举例

Tuning Approach - Weblogic



- System Memory
- File
- Process
- Thread
- JVM Memory
- Access Time
- Session lifecycle

- Access Log
- Error Log
- Query Log
- System Log

- Env. configure
- Multi-thread
- Parallel processing
- Ajax tuning
- Session tuning
- Request Tuning

Tuning Approach - J2EE

Data Monitor

- Thread
- Oracle Connection Pool
- Session Pool
- Bean Pool
- Persistence Object Pool

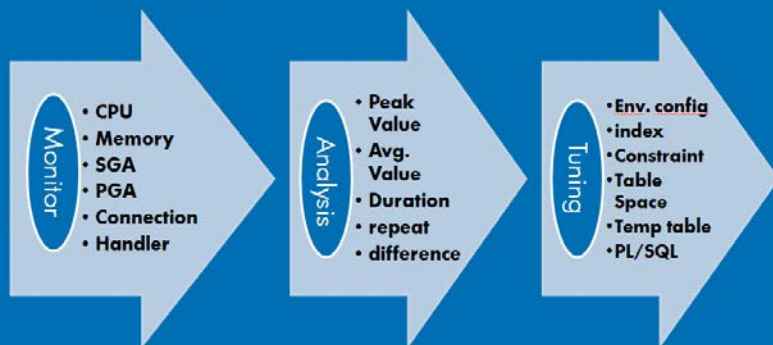
Analysis

- CPU Utilization
- Memory Analysis
- File System Analysis
- Dead Lock Analysis
- Persistence Object Analysis

Solution

- Connection Pool tuning
- Query Tuning
- Parallel Process tuning
- Object recycle
- Oracle connection recycle

Tuning Approach - Oracle



- CPU
- Memory
- SGA
- PGA
- Connection
- Handler

- Peak Value
- Avg. Value
- Duration
- repeat
- difference

- Env. config
- index
- Constraint
- Table Space
- Temp table
- PL/SQL



性能测试案例分析

阿里性能测试具备以下优势：

- 与生产环境达到1:1的配比
- 人力资源
- 技术和工具资源
- 硬件资源
- 数据资源
- 业务场景资源



性能测试案例分析

讨论：

对比传统开发模式和敏捷开发模式下的性能测试



感谢您参加本届MPD！

www.mpd.org.cn

400-812-8020