如何最大化软件测试效能？

Maximize Value!

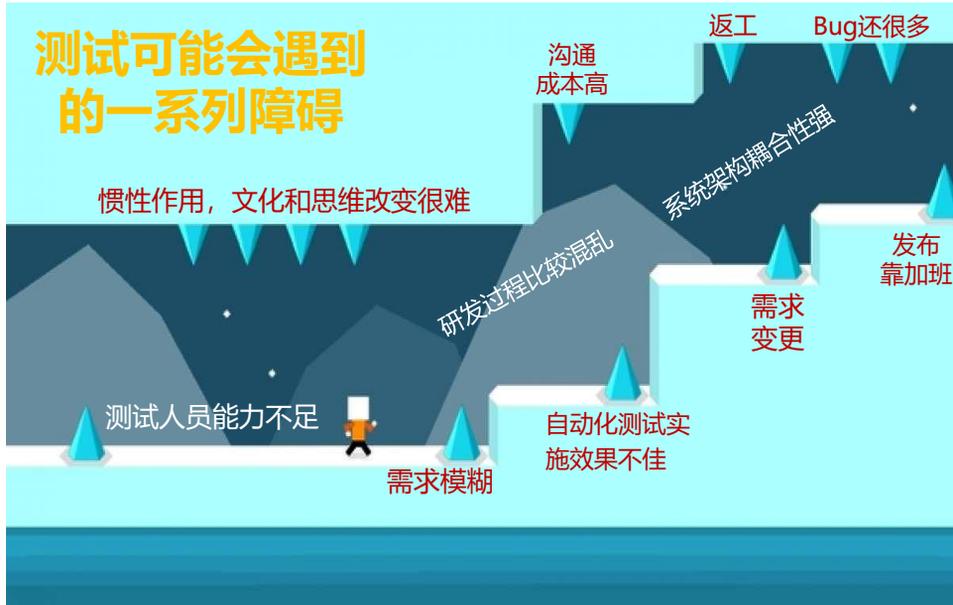**朱少民**

《全程软件测试》作者

QECon大会发起人

问题

软件测试工作瓶颈在哪里？

环境 数据 沟通 用例 设计
准备 测试 缺陷 分析 报告
需求 测试 脚本
变更 执行 开发

测试可能会遇到的一系列障碍

惯性作用，文化和思维改变很难

测试人员能力不足

需求模糊

研发过程比较混乱

自动化测试实施效果不佳

沟通成本高

系统架构耦合性强

返工

Bug还很多

需求变更

发布靠加班

如何跨越这些障碍？

如何不让测试成为绊脚石？

1. 测试思维：突出团队对质量负责
2. 测试周期：测试左移、测试右移
3. 测试自动化：全生命周期覆盖的高度自动化
4. 测试环境：虚拟化、云化、容器、CI集成
5. 度量：全过程可视化
6. 测试与开发融合，设立Test Owner/测试教练角色
7. 组织的支撑：组织改革、文化革新

5

今天从测试自身角度看
如何最大化测试效能?

人　　流程　　技术　　管理
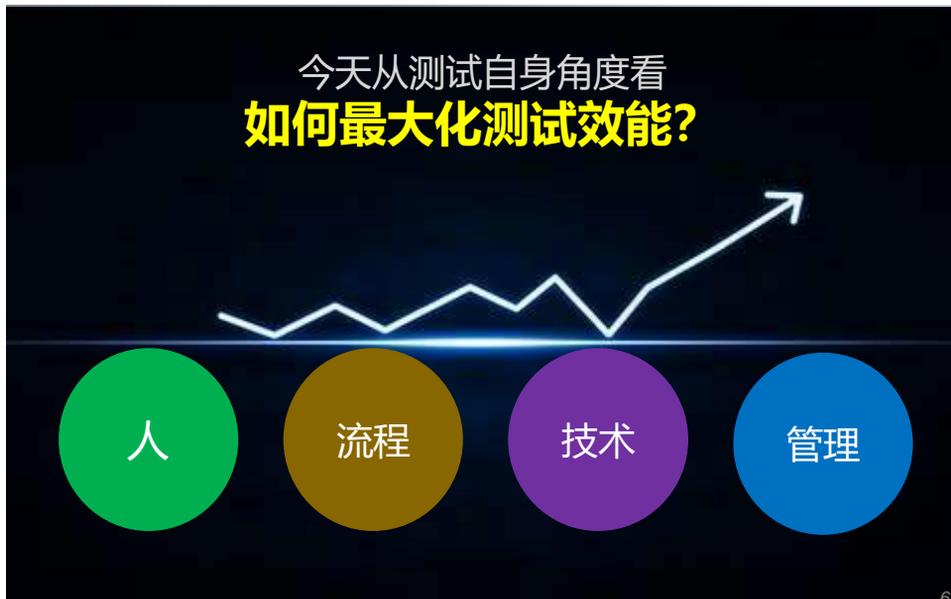
6

人是决定的因素

个人态度与能力、团队协作、组织文化等

Weapons are an important factor in war, but not the decisive one; it is man and not materials that counts.
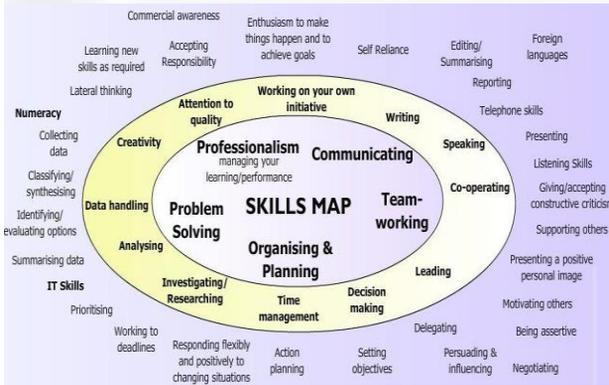
— Mao Zedong —

问题

卓越的测试人员的价值是一般测试人员 多少 倍?

"一个卓越的车床工可以要求比一个普通的
车床工多拿数倍高的工资，但一个卓越
的程序员的价值是一个普通程序员的
10000倍。 "

－－ 比尔 盖茨

卓越的测试工程师是怎样的？

- 热爱测试
- 良好的思维能力
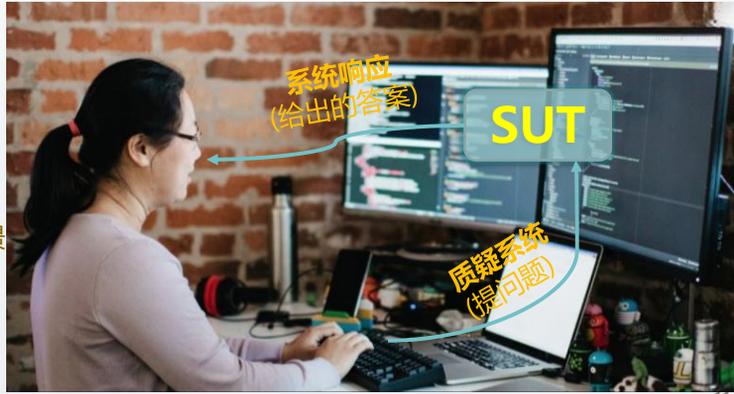- 熟练运用测试方法
- 软件技术功底深厚
- 业务熟练、知识面广
- 良好的沟通能力
- 观察力、洞察力
- ……

9

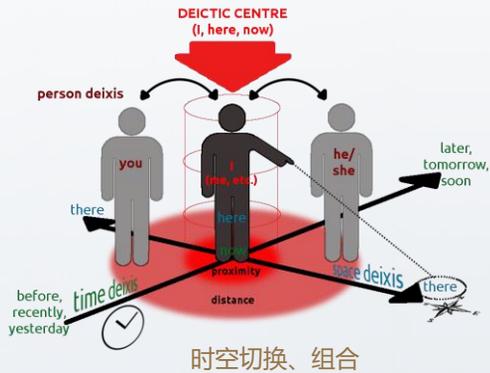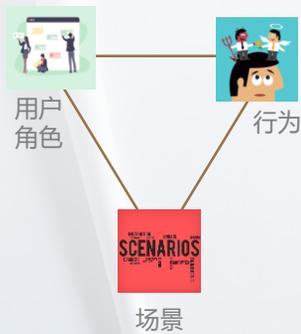## 态度决定一切，但我更愿意强调

测试思维决定了测试人的价值与成长

情境性思维　水平思维　反思　迁移　逆向思维

上下文驱动　产品思维　层次思维

工程化思维　分析性思维

发散思维　独特的视角　用户思维

系统性思维　创造性思维　抽象　类比　结构化思维

5

批判性思维能力

将 **探索式测试、发现缺陷** 做到极致

质疑假定、条件
持续反思、改进
不断深入系统
善于挖掘应用场景
......

系统响应
(给出的答案)

SUT

质疑系统
(提问题)

用户、行为与场景

许多同学傻傻分不清用户、用户故事/用例/用户行为、场景......

用户
角色

行为

SCENARIOS

场景

DEICTIC CENTRE
(I, here, now)

person deixis

you

I
(me, etc.)

he/
she

there

here
now
proximity

later,
tomorrow,
soon

before,
recently,
yesterday

time deixis

distance

place deixis

there

时空切换、组合

任何行为或操作
都取决于其上下文

进一步提升测试能力
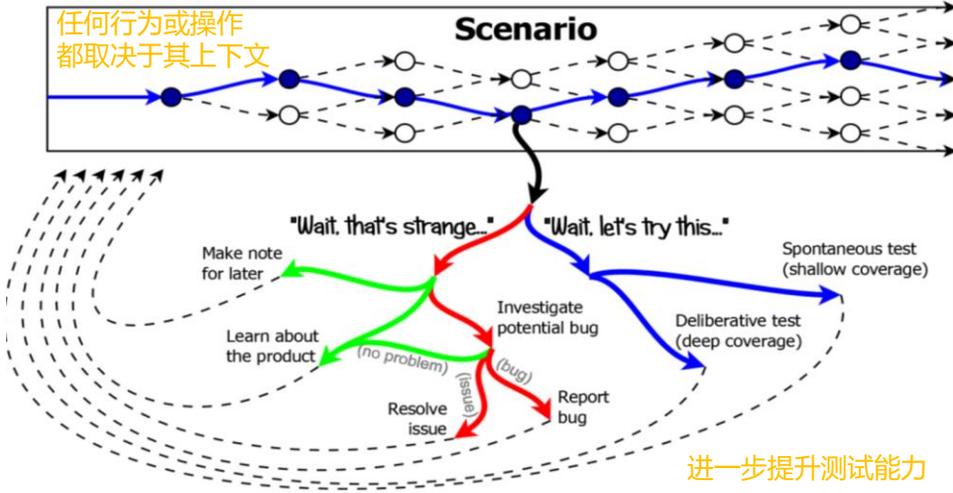
卓越的团队

团结起来就是力量，

但"能做"和"做得又快又好"还差十万八千里

全功能、学习型团队

	功能A	功能B	功能C	性能	UX	安全性
M1	●		●			●
M2		●			●	
M3	●			●		
M4			●		●	
M5		●		●		

开放・尊重・协作・反馈　　简单・勇气・承诺・专注

15

卓越组织的质量文化

一切以用户价值为依归

质量是团队尊严的起点

有条件的情况下，每周从客服提取10条客诉录音播放，**听听用户肆无忌惮的谩骂和侮辱，品品内心撕心裂肺的疼痛和羞愧，在破坏的基础上重新建立起踏踏实实的尊严**

ZERO DEFECT

Quality built-in
Design for Quality
Design for Test

16

流程 **卓越的测试流程是怎样的?**

CONSISTENT PROCESS + RELIABLE DATA

= BETTER RESULTS

有什么流程就有什么结果

尽快地、持续的揭示质量风险

质量风险揭示的越早越快,
产品越好、项目风险越小、成本越低

产品经理、用户等

反馈质量

客户需求 → 需求 → 设计 → 代码 → 系统 → 运维 → 客户满意

质量问题反馈

开发人员

测试流程力求简化、轻量化

质量文化：缺陷预防、quality is built-in　　　　全过程的协作、有效沟通

测试需求 验收标准 → 一页纸的 测试计划 → 测试点列表 → 测试执行 → 😊

提炼 ↑ 测试需求

支撑 ↑ 自动化 测试

良好的基础设施（TF，email，IM，wiki，…）和开发平台、项目管理平台
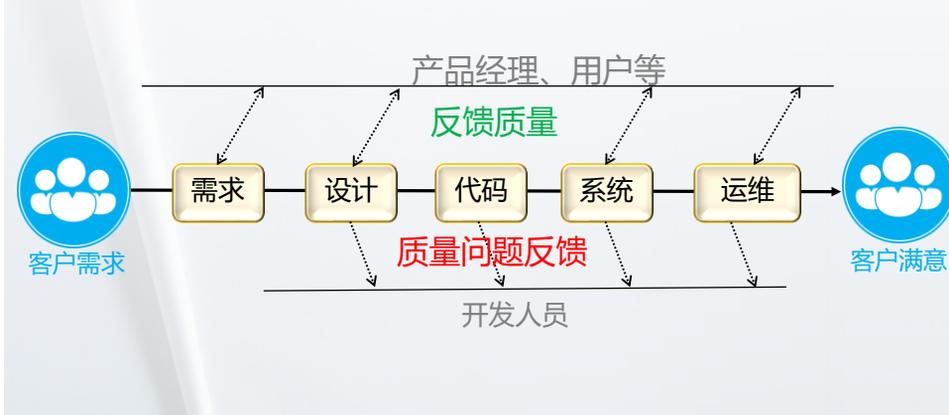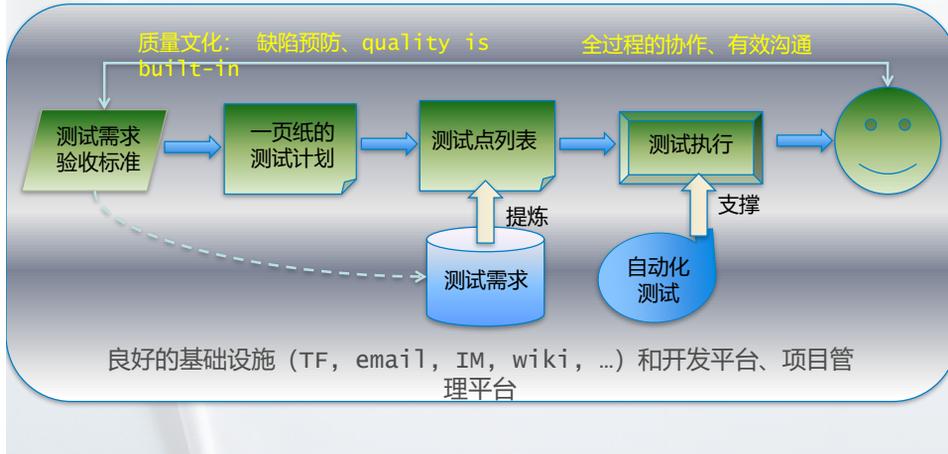
一页纸的测试计划

最大程度地 简化文档

抓住测试计划的 核心、要点即可

USABILITY TEST PLAN DASHBOARD

20

人更具有灵活性、适应性，**能应付各种需求的变化**

人能理解业务，也更具创造性，举一反三，**发现缺陷能力远远超过工具**

但所有做过的测试，都是已知的，100%自动化

工具 执行回归测试守护质量 倒是不折不扣

测试 = 已知的检**测** + 未知的**试**验

迭代

开始　　持续的新功能测试 + 回归测试　　结束

新功能探索式测试 @I_n

为 @I_{n-1} 已有的功能特性开发TA脚本

回归测试自动化

执行(I_1 ~ I_{n-1})

彻底的左移：**需求即测试** QR 软件质量报道 QualityReport
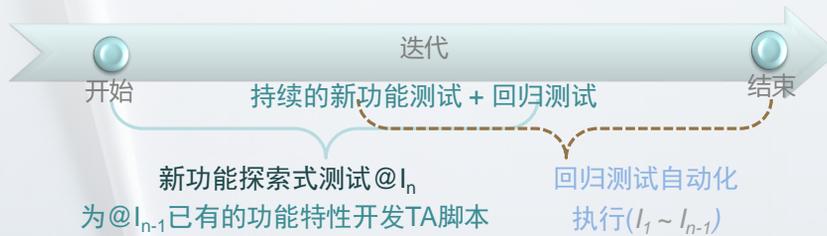
验收标准 (Acceptance **C**riteria**)**

□ 操作人/用户

□ 行为

□ 可观察的结果

Given [Precondition]

When [Actor + Action]

Then [Observable Result]

AC ──成为──→ 测试（自动化）

细化

验证

需求
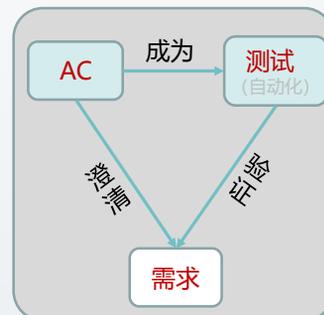
测试右移进一步缩短测试周期

最终形成**质量改进的闭环**

代码分析、评审
与单元测试

业务度量,
DoD
质量度量/
KPI仪表盘

环境验证
部署验证

Code

Plan

Deploy

Dev

Build

Release

Ops

Operate

容量
QoS
管理

Test

Monitor

持续的迭代测试

质量监控/客户反馈分析

24
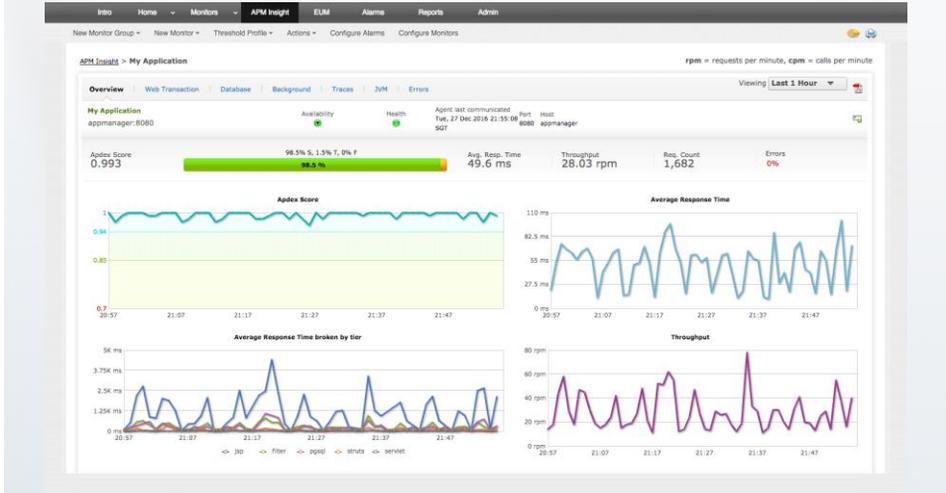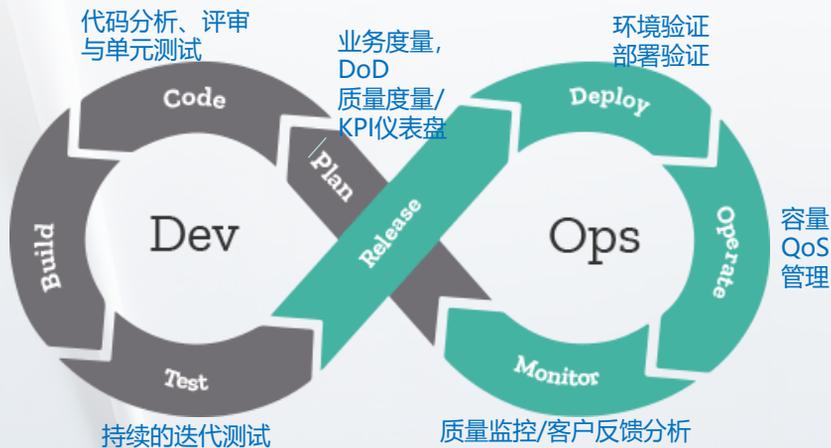
技术

Testing Techniques
卓越的测试技术

自然会想到它

QR 软件质量报道 QualityReport

自动化测试

How can you **effectively** implement **Test Automation?**

良好的自动化测试策略

从而获得稳定的、有效的自动化测试

❶ 金字塔策略
❷ 全生命周期覆盖
❸ 持续实施、持续改进
❹ 与CI/CD集成
❺ 不变的测试环境
❻ 采用AI 技术
❼ ……

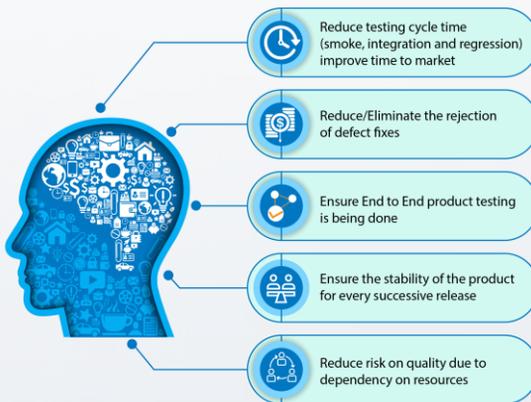Reduce testing cycle time (smoke, integration and regression) improve time to market

Reduce/Eliminate the rejection of defect fixes

Ensure End to End product testing is being done

Ensure the stability of the product for every successive release

Reduce risk on quality due to dependency on resources

良好的自动化框架/基础设施

业务需求即测试需求，如用户故事的验收标准，并且借助类似GWT的DSL来开发测试脚本，让需求可执行
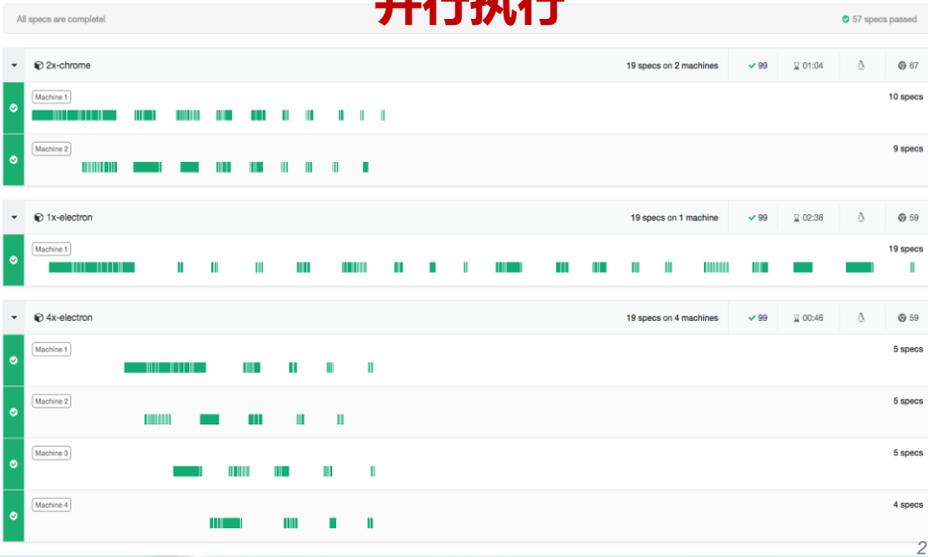
```
Feature：查看课程分销收益详情
做为一名参与课程分销的用户，我想查看分销的收益详情

Scenario：用户查看课程分享的收益详情
    Given 我成功登录在线教育网站
    When  我进入我的账户
    And   我进入课程分销中心
    Then  我应该看到我的分销收益详情
```

```java
package com.example;
import io.cucumber.java8.En;
import cucumber.api.java.en.And;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;
import sun.security.util.PendingException;

public class StepDefinitions implements En {
    @Given("^我成功登录在线教育网站$")
    public void 我成功登录在线教育网站() throws Throwable {
// Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @When("^我进入我的账户$")
    public void 我进入我的账户() throws Throwable {
// Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @And("^我进入课程分销中心$")
    public void 我进入课程分销中心() throws Throwable {
// Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^我应该看到我的分销改益详情$")
    public void 我应该看到我的分销收益详情() throws Throwable {
// Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }
}
```

并行执行

更彻底的自动化测试：MBT

软件质量报道 QualityReport

根据需求自动生成测试用例、数据，实现彻底的自动化测试

	结构	功能	行为	人机交互
操作		因果图　决策表		IFML　用例图
组件	类图　数据流图	业务流程图	活动图　序列图	系统生态图
系统	部署图　E-R图		有限状态机	

基于MBT来提升测试效率

Improved Efficiency
40%
faster development cycle

Higher Quality
50%
more defects found

Reduced Cost
400%
return on investment

AI + MBT

Codeless自动化测试

不是没有代码，而是测试人员不用自己开发测试代码，借助工具生成可以执行的测试用例集，且易于管理，具有良好的**可视性**。**更强大的**脚本录制能力，并具有脚本**自愈能力**

卓越的测试管理

管理

规范模板

仪表盘

绩效牵引

任务调度

策略

赋能

协作

因地制宜采用测试策略

- 基于风险的测试策略
- 基于用户使用的测试策略
- 启发式测试策略
- 组合策略

Perceivable
Information and User Interface components must be presentable to users in ways they can perceive
01

Understandable
Information and the operation of User Interface must be understandable
03

Operable
02
User Interface components and navigation must be operable

Robust
04
Content must be robust enough and should be interpreted reliably by a wide variety of user agents including assistive technologies

36

示例：两段论和立体作战

发现缺陷，强调效率　　　验证功能，降低风险

单元/集成测试

自动化每日验证（SVT/BVT）

关键里程碑

探索式测试　　　　　回归测试

非功能性的系统测试

新功能验证　　针对有风险区域的功能测试　　功能验证测试　　剩余缺陷验证　　确认测试

37

让测试成为服务：赋能团队

测试建模服务

测试设计服务

测试数据服务

测试结果呈现服务

TaaS

任务调度服务

Mock服务

测试执行服务

测试环境服务

整体卓越 = 持续测试

技术手段/方法	探索式测试	测试自动化	测试设计方法

中间过程	基于风险的测试分析	测试影响分析

实现基础	测试数据管理	DevOps工具链集成	服务虚拟化

持续测试

QECon
全球软件质量&效能大会
GLOBAL SOFTWARE QUALITY & EFFICIENCY CONFERENCE
深圳 | 5月28-29日
主办方：QECon组委会
QECon Organizing Committee

最后是广告时间

Q & A

Thank you

软件质量报道