通过测试卓越来推动电信领域持续集成

诺基亚西门子无线网络平台
自动化测试教练林曙湧
@林曙湧
Sean.linsy@gmail.com

问题

- 你的产品线有多少自动化测试用例
- >100
- >1000
- >5000
- >8000+
- 10000+

自我介绍

- 1998-2001 UTStarcom 现场支持工程师
- 2001 加入系统测试团队
- 2003-2005 测试团队 leader
- 2005-2008 自动化测试Team Leader
- 2009– Now 加入诺基亚西门子担任自动化测试教练
- Belief: Testing Could be more creative and interesting than current ways of working

分享要点

- 产品及研发背景介绍
- 持续集成上碰到的困难
- 我们的破冰之旅
- 实践经验分享
- 方案回顾

我们的产品背景

- 移动通信核心网软件平台
- 超过1200万行的代码维护量
- 电信级的软件质量要求，面向全球上百个运营商进行发布
- 软件交付的周期很长
- 软件的交付延期以及交付后的的质量问题导致我们失去很多

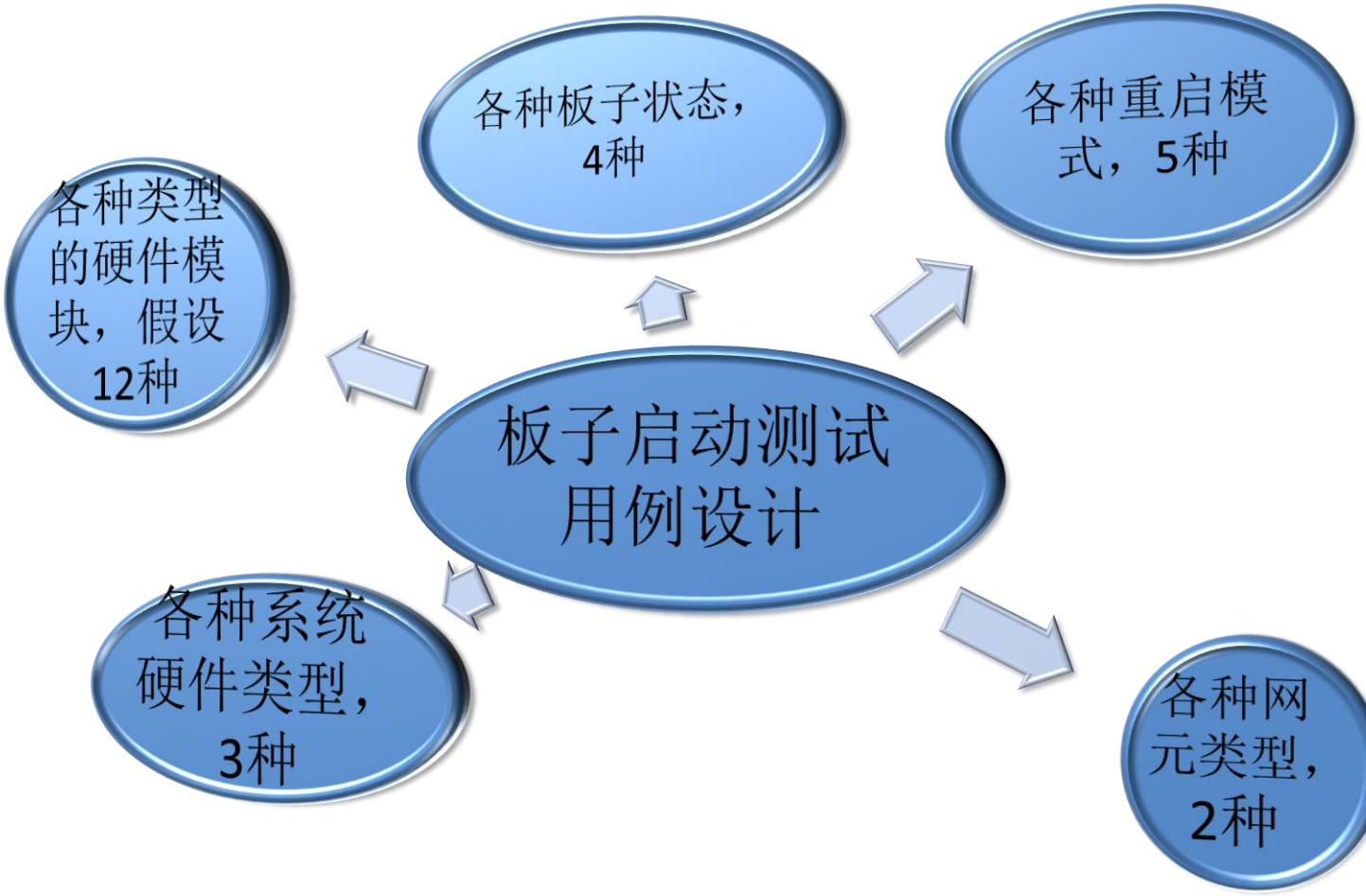
我们的研发背景

- 2006年开始导入Scrum
- 2007年产品线全面转型Scrum（1000+ people）
- 测试人员加入 Scrum team
- 开发人员参与测试用例的开发
- 自动化测试用例的完成作为我们Sprint 的Done 的标准
- 持续集成被广泛地使用来控制风险
- 听起来很不错？那还有什么问题呢？

持续集成的困难

- 为了达到高质量的要求，测试人员考虑了所有用户可能的使用场景
- 自动化测试用例数量很多（10000+）
- 测试执行速度从外 部已经很难优化
- 测试需要真实设备，数量非常有限
- 测试用例的维护工作量很大
- 自动化测试的杀虫剂效应

单板启动领域典型例子介绍



计算一共多少可能的组合

12*4*5*2*3 ==1440

在每个小的迭代我们真的可能执行所有的用例吗?

- 硬件资源有限
- 人力资源有限
- 需要时间做探索性测试
- 需要休息
- 需要考虑投资回报率
- ...

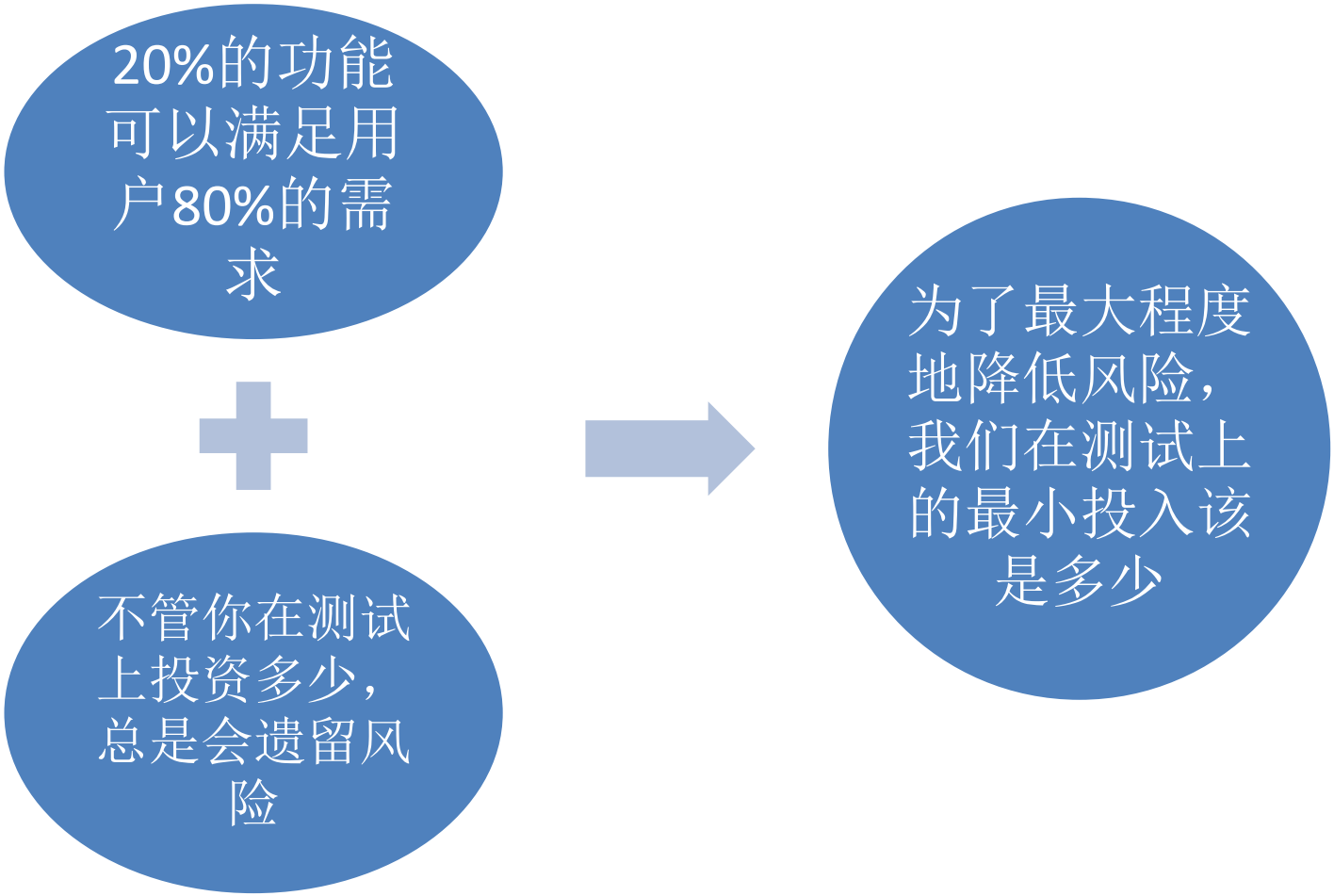
- 回答是否！持续集成陷入困境！

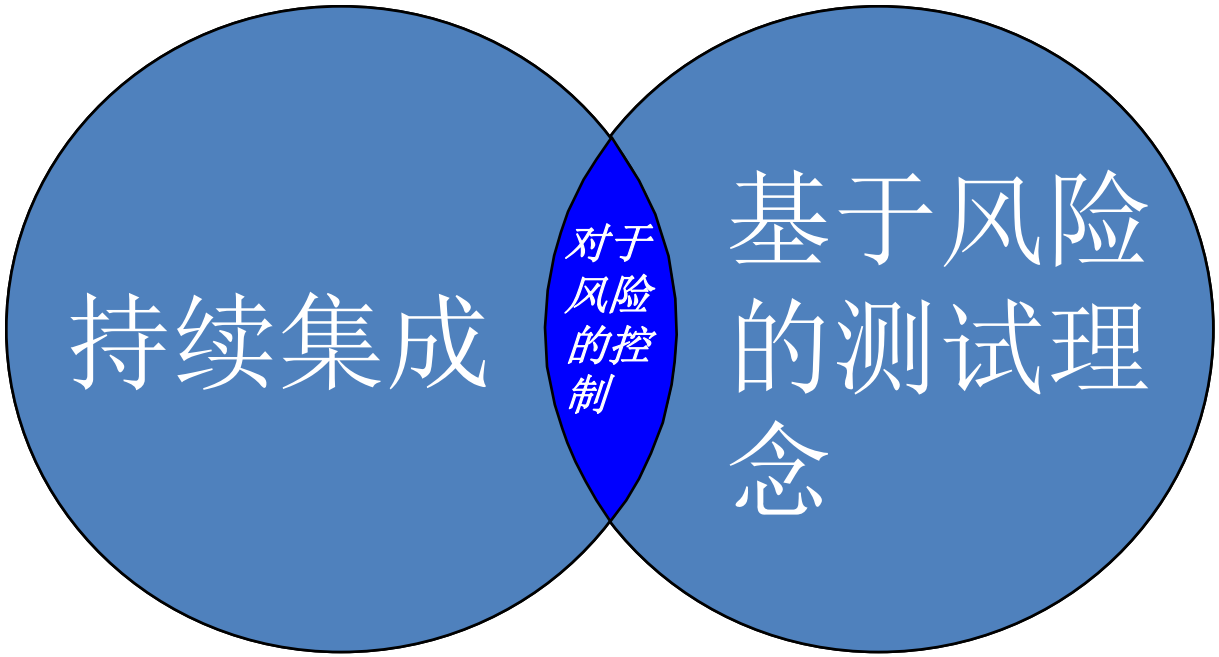
理想很丰满 现实很骨感

- 不同测试的重要性也许是不一样的？
- 一个问题往往会导致很多用例失败
- 我们的用例在重复地发现错误？

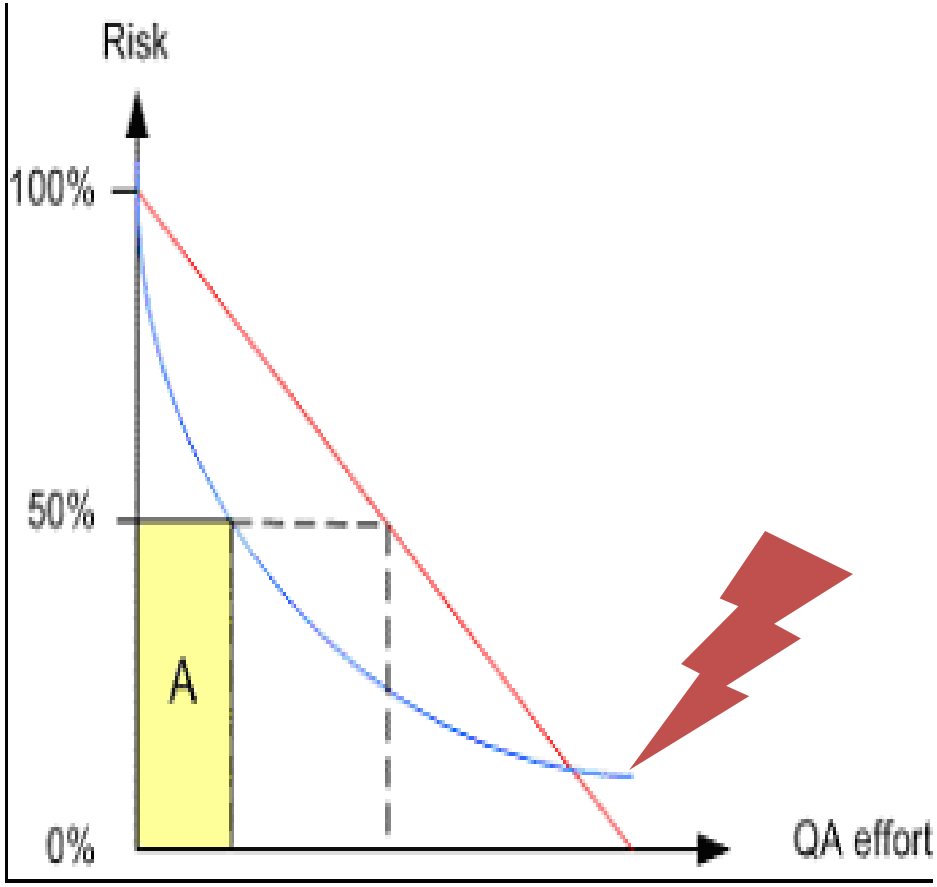
- 在每个小的迭代我们不可能执行所有的测试用例
- 那如何在控制风险的前提下来精简测试

基于风险的测试理念


20%的功能可以满足用户80%的需求
不管你在测试上投资多少，总是会遗留风险
为了最大程度地降低风险，我们在测试上的最小投入该是多少


持续集成
对于风险的控制
基于风险的测试理念

测试对于风险的控制曲线



针对核心功能或者高风险功能的测试，极小的投入就能让我们控制大多数的风险

测试的边际作用递减

可是核心的测试用例就已经很多了怎么办呢？

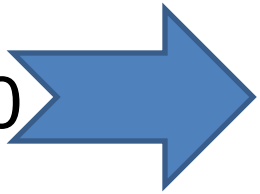
业界研究

- 1997 年Telcordia科技研究表明，假如一个软件系统由N构件组成(或者说由N个因素决定)，大部分的软件错误是由一个构件的错误所导致，或者由 2 个构件之间的交互错误导致
- 基于这个理论，构造测试用例就需要涵盖每个因素的所有状态，并且涵盖每 2 个因素之间的 所 有 交 互 ， 这 种 测 试 理 论叫 作Pair-wise测 试。

相关测试理念

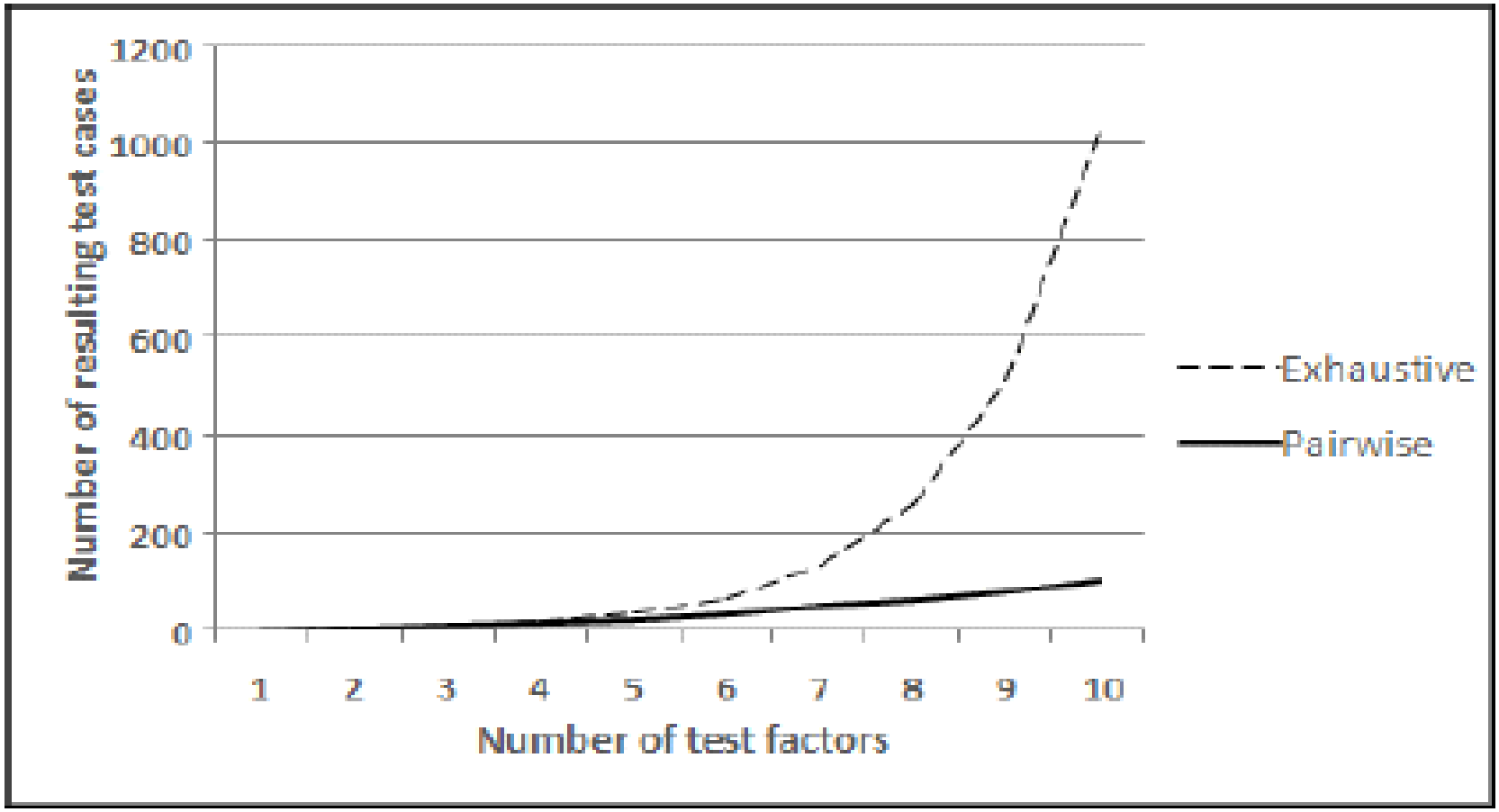
- 最简单的问题往往可以由一个输入参数触发
- 次简单的问题往往可以有两个输入参数的交互触发
- 必须三个或者三个以上的输入参数交互触发的问题往往比较少，要想全部抓住他们往往也很贵
- 全配对测试技术可以起作用的就是在第一和第二个领域，在风险控制和测试成本之间达到最佳的平衡

- 12种硬件模块，4种状态，5种启动方式，2种应用平台，3种硬件平台，要多少次测试才能做到所有的两两交互都覆盖？
- 12*4*5*2*3 ==1440 ？ - NO



1440 59

全组合 vs. 全配对

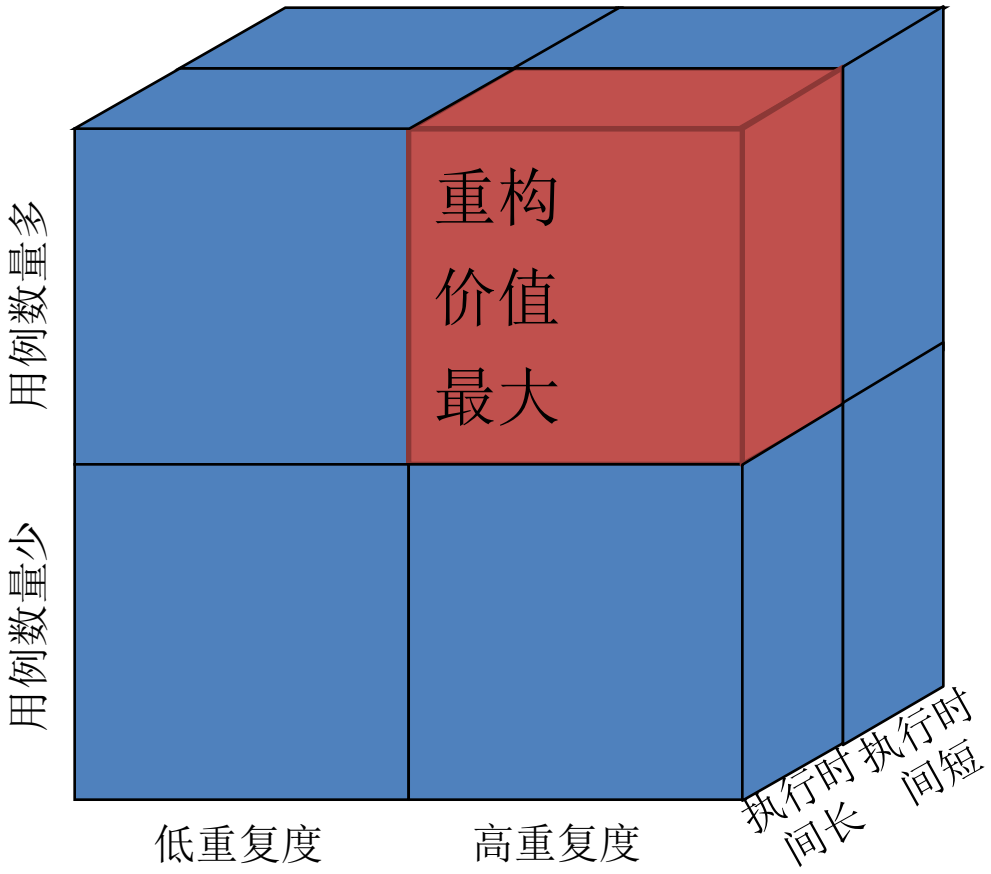

Number of resulting test cases
1200
1000
800
600
400
200
0
1
2
3
4
5
6
7
8
9
10
Number of test factors
Exhaustive
Pairwise

实践经验分享

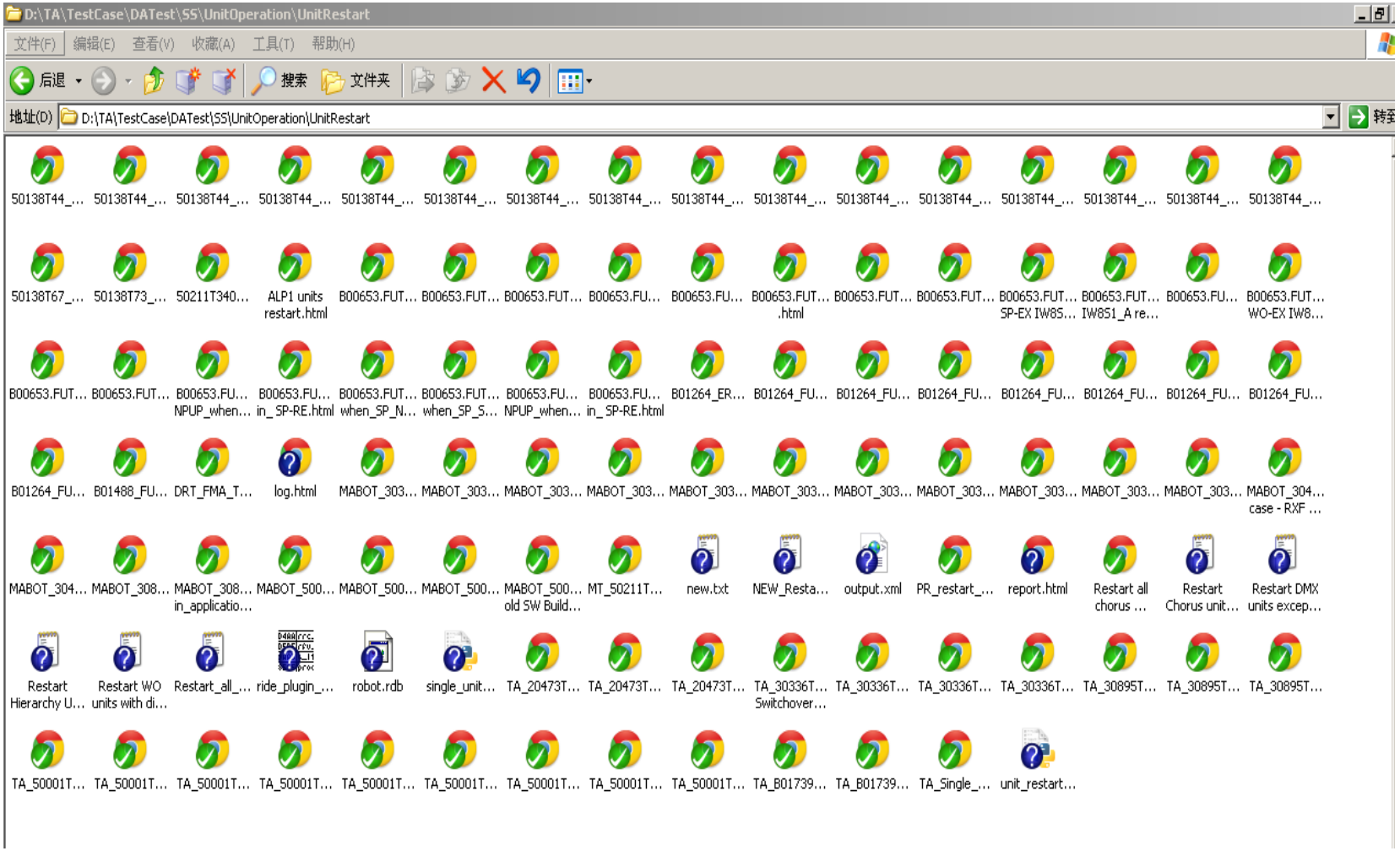
- 重构原有的测试用例
- 如何利用
  - 统一的测试过程
  - 测试数据
  - 组合算法

  来自动化的生成用例
- 如何部署于持续集成环境

如何选择测试用例重构的范围



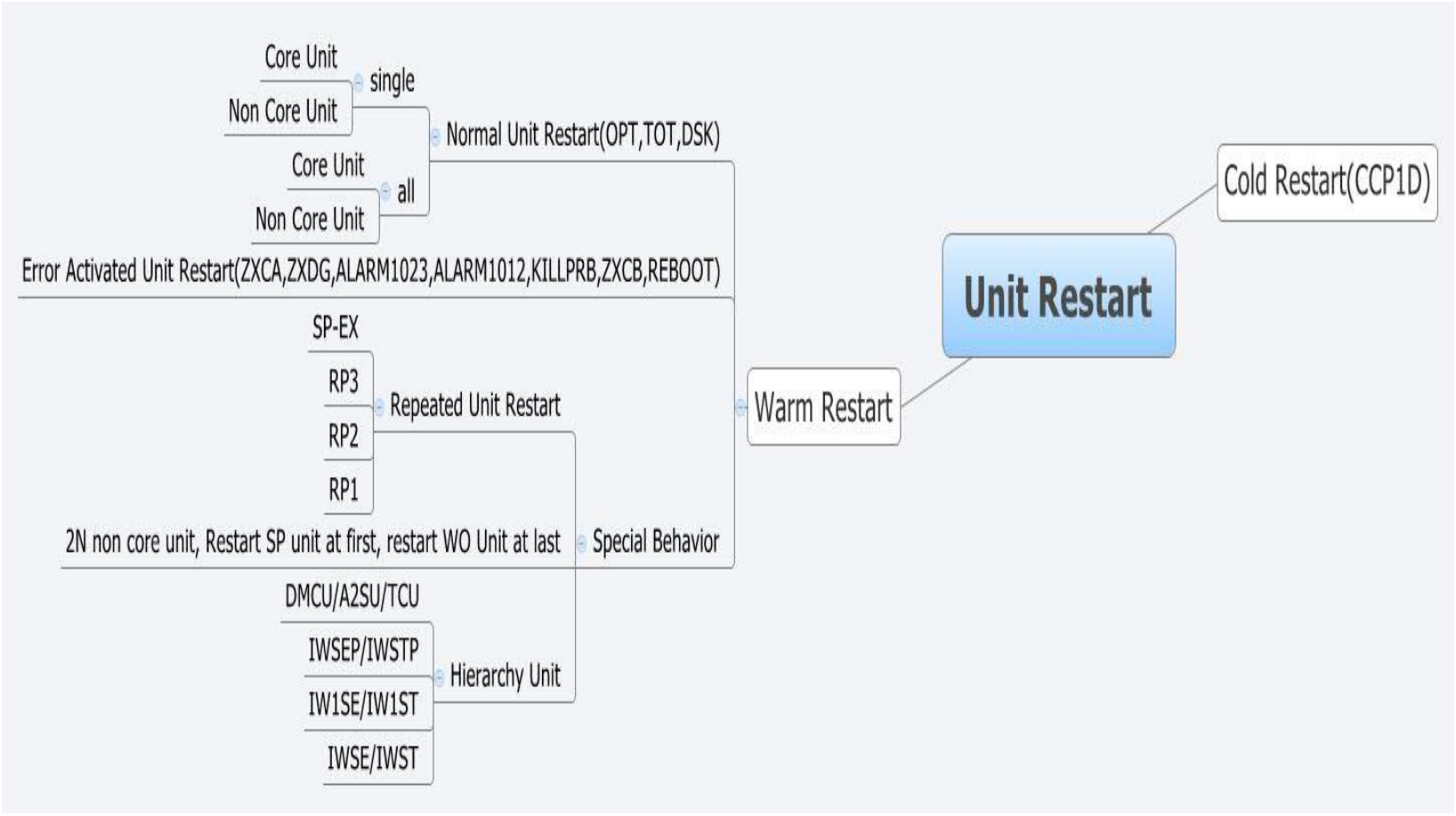
板子重启的1000+ 测试用例



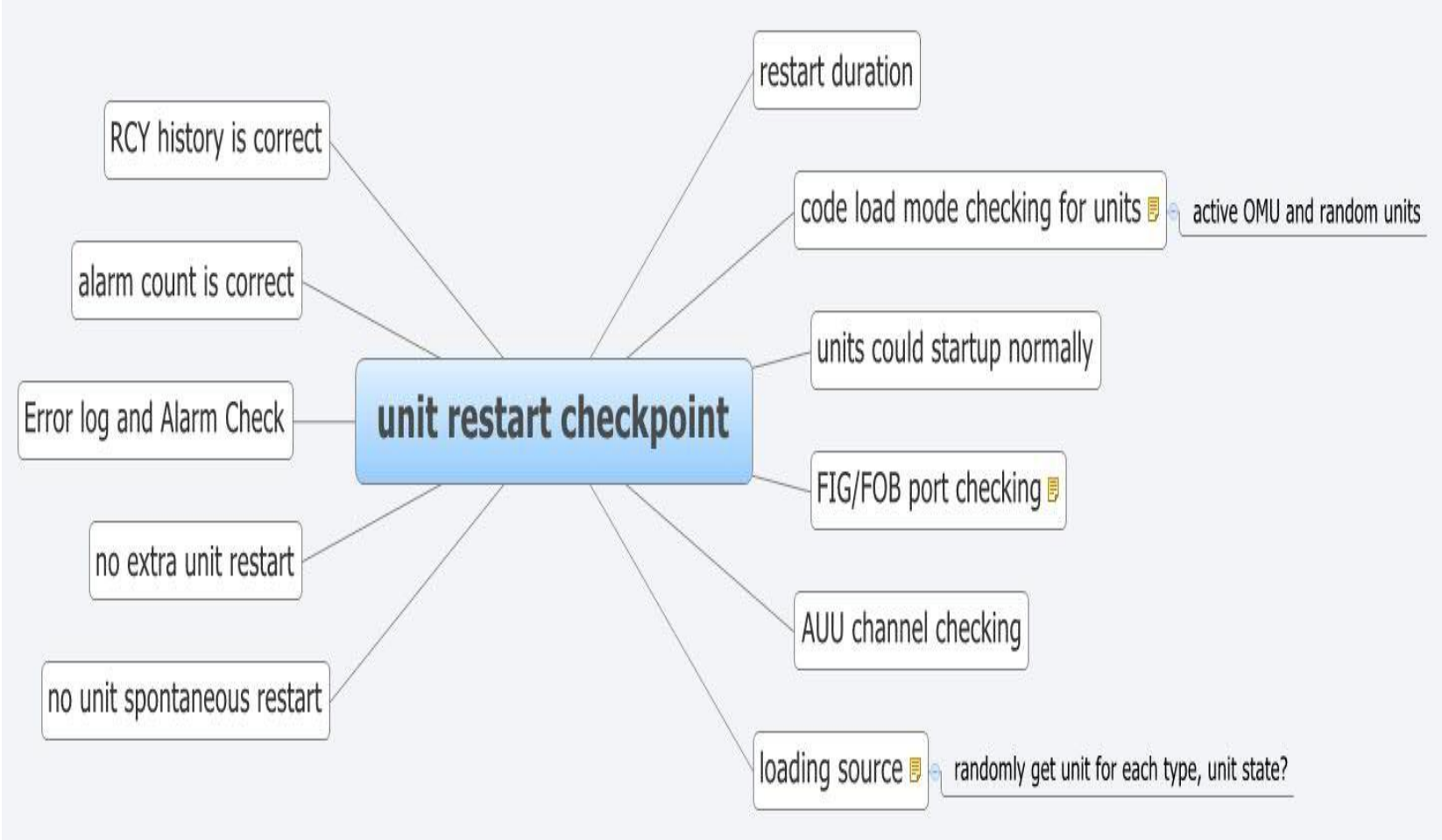
测试重构的困难

- 原来的测试用例是多年的积累，不能轻易舍弃
- 里面包含着很多有价值的点
- 重构的测试覆盖率不能低于原有
- 保证我们已经覆盖了所有重要的功能

测试过程的设计


Unit Restart
Cold Restart(CCP1D)
Warm Restart
Normal Unit Restart(OPT,TOT,DSK)
single
Core Unit
Non Core Unit
all
Core Unit
Non Core Unit
Error Activated Unit Restart(ZXCA,ZXDG,ALARM1023,ALARM1012,KILLPRB,ZXCB,REBOOT)
Special Behavior
Repeated Unit Restart
SP-EX
RP3
RP2
RP1
2N non core unit, Restart SP unit at first, restart WO Unit at last
Hierarchy Unit
DMCU/A2SU/TCU
IWSEP/IWSTP
IW1SE/IW1ST
IWSE/IWST

测试检查点的设计


unit restart checkpoint
restart duration
RCY history is correct
code load mode checking for units
active OMU and random units
alarm count is correct
units could startup normally
Error log and Alarm Check
FIG/FOB port checking
no extra unit restart
AUU channel checking
no unit spontaneous restart
loading source
randomly get unit for each type, unit state?

实践经验分享

- 如何对原有的测试用例做重构
- 如何利用
  - 测试的输入数据
  - 测试输入的相互约束
  - 组合算法
  - 测试行为的描述

  来自动化的生成用例
- 如何部署于持续集成环境

测试需求的抽象描述

- 需求领域: 系统与硬件模块的启动

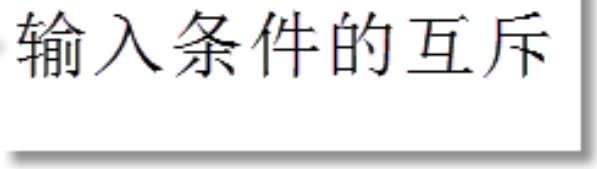
考虑到电信软件高可靠性的要求，

在不同的硬件平台+上层应用程序的组合环境中

所有可能的硬件模块

在各种可能的状态

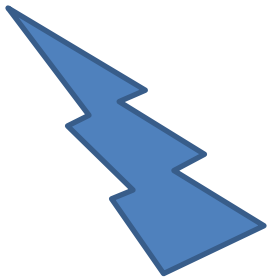
以各种可能的模式重启



并符合各模块的测试点以及通用检查点。

业务领域的测试输入数据

```
parameters = [ [ "ICSU","MPS1","MXU","SFU","MPGEP","DMCU","OMU","RSMU","MPS1P","A2SU","MIS1P","GTPU"]
        , [ "WO-EX","SP-EX","TE-EX","BL-ID"]
        , [ "OPT","TOT","DSK","ZXCA","ZXDG" ]
        , ["RNC","MGW"]
        , ["SF20H","SF10E"]
        ]
```



It is all about data!

Behavior is also one kind of data!

```
pair_wise = all_pairs(parameters)
all_line=[line for line in pair_wise]
print len(all_line)
```

业务领域的相关依赖

```
def is_valid_combination(row):

    n=len(row)
    if n>=len(parameters):

        # not core unit state is wo and is 2N unit,only restart unit with mode OPT,TOT,DSK
        if row['state'] =="WO-EX" and row['unit_type'] in (Hot_Redundancy_unit_types + Nplus1_unit_types) and row['restart_mode']
            return False
        # core unit in wo state not restart
        if row['state'] =="WO-EX" and row['unit_type'] in CORE_unit_types:
            return False
        # Single Unit and SN+ Unit don't have SP-EX unit
        if row['state'] =="SP-EX" and row['unit_type'] in (SN_unit_types + Non_Redundancy_unit_types):
            return False
        # only TCU,DMCU,NIWU have BL-ID state
        if row['unit_type'] not in Unit_has_BL_type_list and row['state'] =='BL-ID':
            return False
        # TE state set alarm 1012 unit will not restart
        if row['state'] == "TE-EX" and row['restart_mode'] in ["ALARM1012"]:
            return False
        # TE state dxm kill prb not affect
        if row['state'] == "TE-EX" and row['restart_mode'] in ["KILLPRB"] and row['unit_type'] in DMX_unit_types:
            return False
        #hierarchy unit restart by reboot,killprb,will be included in hierarchy unit restart cases
        if row['unit_type'] in Hierarchy_unit_types and row['restart_mode'] in ["KILLPRB","REBOOT"]:
            return False
    return True
```

创建统一测试过程

- ✓选取符合类型和状态要求的硬件模块
- ✓执行要求模式的重启操作
- ✓模块A的相关检查点
- ✓模块B的相关检查点
- ✓模块C的相关检查点
- ✓各模块的公共检查点
- ✓如果一旦错误，按照要求抓详细的消息及日志并将系统恢复到正常状态

测试行为的描述

common_kw | model | unit_restart_example | state_change | generator | unit_restart_lib

```
10      test_unit,previous_state = kw('choose one unit for test',unit_type,state)
11      set_test_start_time()
12      unit_info = FunctionUnit(test_unit.name,state)
13      CHECK_SP_UNIT = False
14      '''if restart unit is wo ,and is 2n unit ,have sp unit ,should check sp unit'''
15      if state == 'WO-EX' and unit_info.is_pair_unit():
16          pair_unit = unit_info.get_pair_unit_name()
17          pair_unit_info = FunctionUnit(pair_unit)
18          if  pair_unit_info.get_unit_state() == 'SP-RE' or pair_unit_info.get_unit_state() == 'SP-EX':
19              CHECK_SP_UNIT = True
20      kw('restart_unit_with_mode',test_unit.name,restart_mode)
21      time.sleep(1)
22
23      '''if unit in BL-ID ,after restart unit state will change to TE-EX ,need change back to BL-ID by manual'''
24      if state in ['BL-ID']:
25          kw('Change Unit State',test_unit.name,'BL','')
26
27      '''if unit is hierarchy unit,we should wait all subunit in expected state'''
28      if unit_info.is_hierarchy_unit():
29          kw('Wait until units list are in states',[test_unit.name],[state,'SE-NH'],'14min','both')
30      else:
31          kw('Wait until units list are in states',[test_unit.name],[state],'14min','unit')
32
33      test_start_time = get_test_start_time()
34      run_keyword_continue_on_failure('restart_figfob_check_point',unit_info)
35      if restart_mode not in ['ZXDG']:
36          run_keyword_continue_on_failure('rcy_check_1001_alarm_history',\
37                                          test_unit.name,state, restart_mode,test_start_time,1)
38          run_keyword_continue_on_failure('check_recovery_history_for_unit_restart',\
39                                      test_unit.name,state, restart_mode,test_start_time,1)
40      if CHECK_SP_UNIT:
```

根据风险调整测试强度

- 全组和
- 随机
- 全配对
- ...

自动化的生成用例

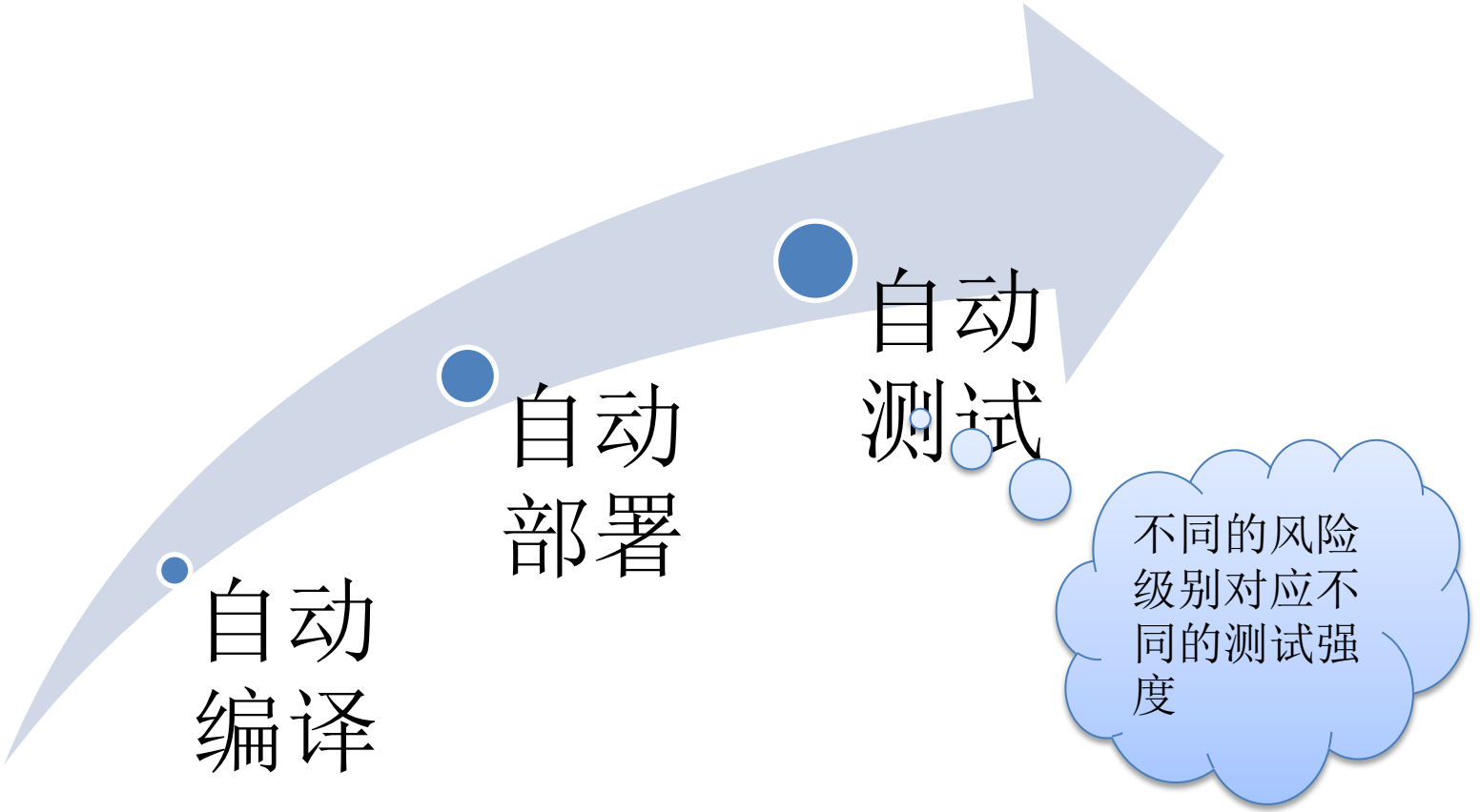
- 直接创建可读可执行的自动化用例
- 测试用例的数据和测试环境自适应
- 每次插入随机因子生成不同的组合
- 实例演示

实践经验分享

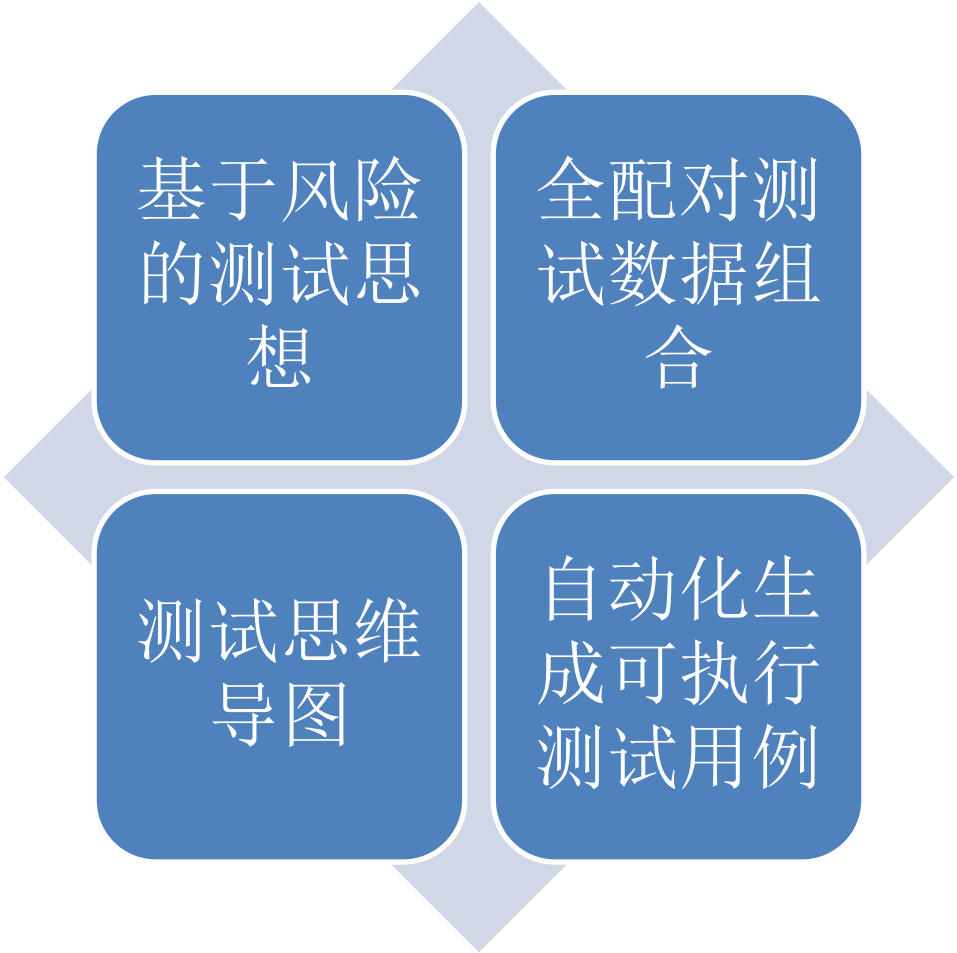
- 如何对原有的测试用例做重构
- 如何利用
  - 统一的测试过程
  - 测试数据
  - 组合算法

  来自动化的生成用例
- 如何部署于持续集成环境

如何部署于持续集成环境



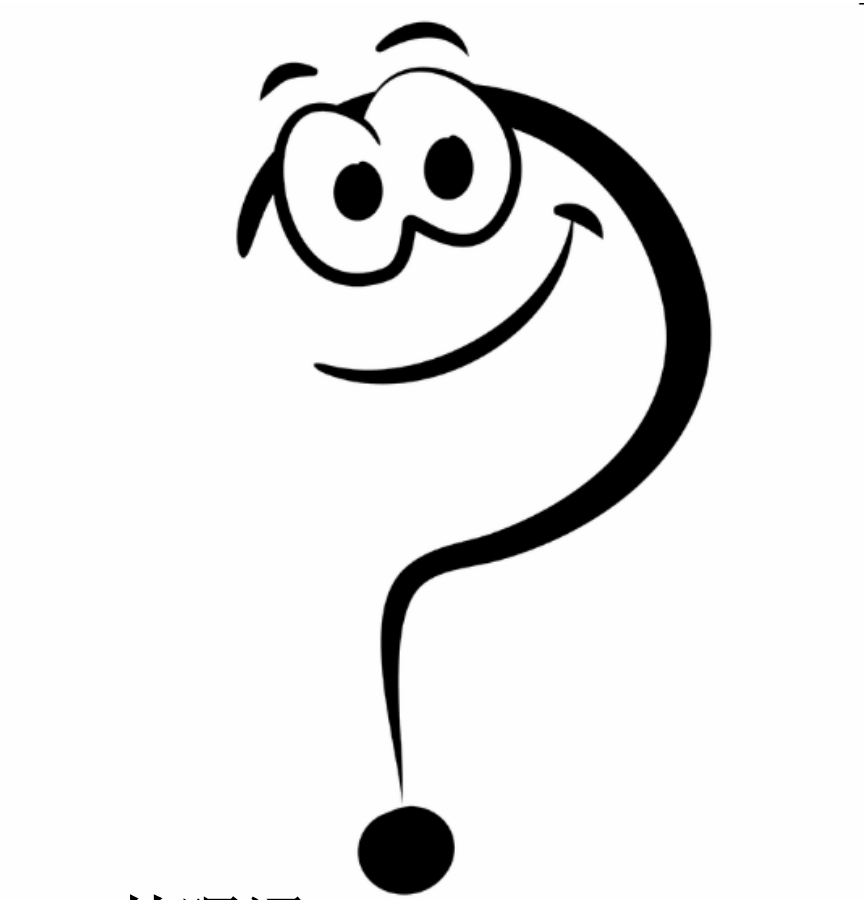
测试技术回顾



经验回顾

- 思维的转变是最重要的第一步
- 更多的测试而不是更多的测试用例
- 测试模型和测试数据的分离
- 以频繁的测试来弥补每次测试覆盖率的不足
- 测试领域知识的集中存放
- 不同的风险对应不同的测试覆盖率
- 探索性的自动化测试

Q&A



@林曙湧

http://www.cnblogs.com/blue_energy/

Sean.linsy@gmail.com

Robot Case Example

Setting	Value	Value	Value	Value
Resource	03_resource.html			
Test Setup	Open Login Page			
Test Teardown	Close Browser			

Variable	Value	Value	Value	Value

Test Case	Action	Argument	Argument	Argument
Valid Login	[Setup]	Open Browser	${LOGIN_URL}	${BROWSER}
	Title Should Be	Login Page		
	Location Should Be	${LOGIN_URL}		
	Input Text	username_field	demo	
	Input Text	password_field	mode	
	Click Button	login_button		
	Title Should Be	Welcome Page		
	Location Should Be	${WELCOME_URL}		
Higher Level Valid Login	[Documentation]	Same as earlier test but with higher abstraction level.		
	Input Name	dave		

个人对测试卓越的理解

- 1: Demand Technical Excellence
- 2: Promote Individual Change and Lead Organizational Change
- 3: Organize Knowledge and Improve Education
- 4: Maximize Value Creation Across the Entire Process