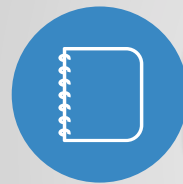


自动化测试设计与敏捷实践

黄冶 2016-07-18



一、初识自动化测试





概念介绍



自动化测试范围



敏捷测试自动化的原则



自动化测试的好处



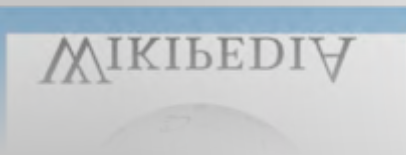
自动化测试的局限



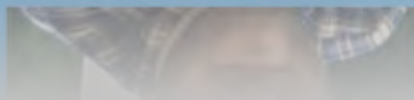


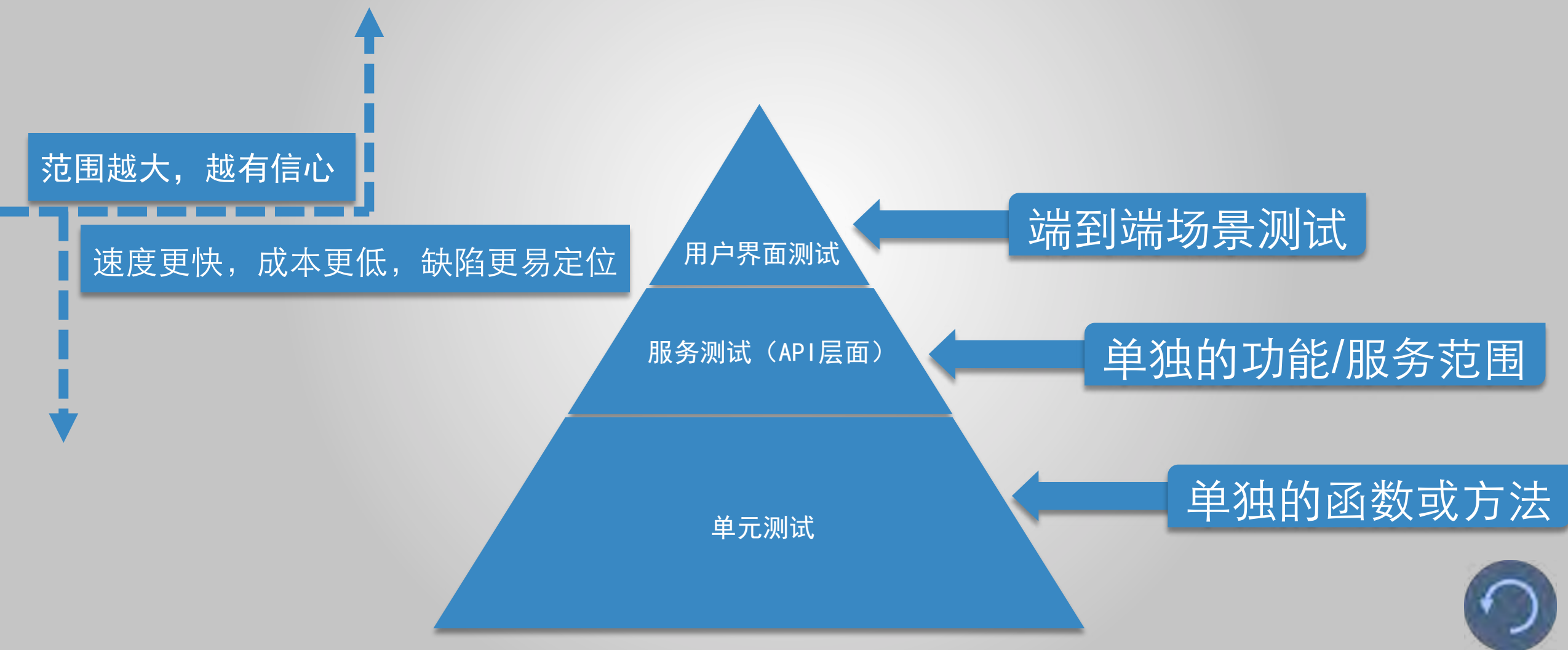
测试自动化

测试自动化是使用特殊的软件来控制测试用例的执行，并比较期望结果与实际结果。



测试自动化就是任何使用工具辅助测试的做法。测试自动化扩大了测试人员的能力范围。







自动化测试的好处

缩短测试周期

降低测试成本

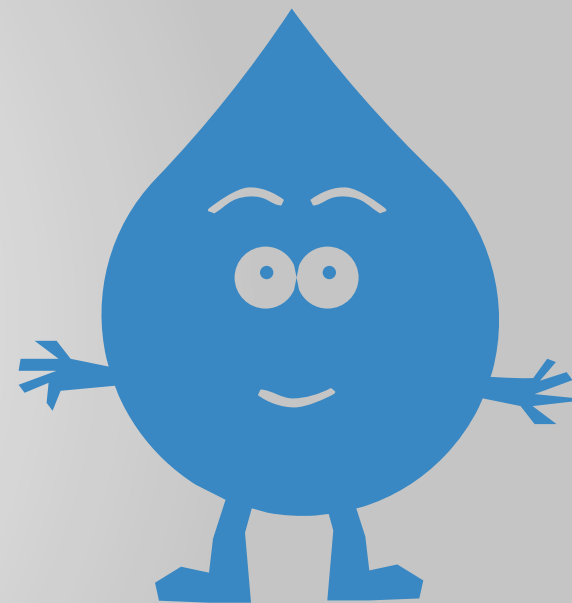
提高资源利用率

可以更频繁的测试，提高了覆盖率

更方便重现软件缺陷

扩展测试工程师的能力范围

降低新人参与实际测试工作的门槛





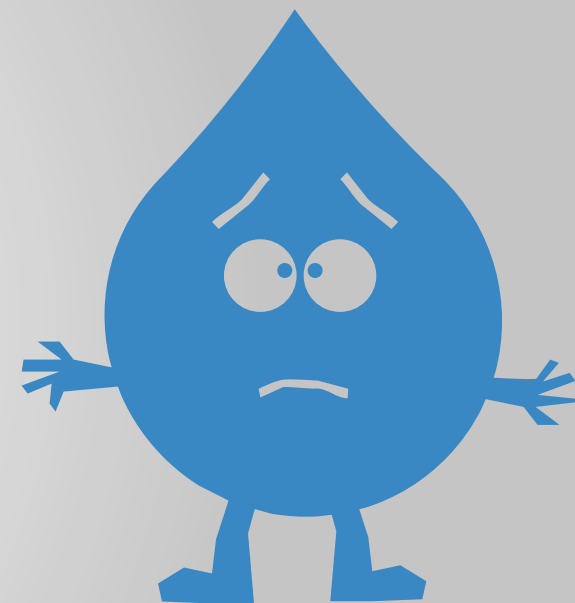
自动化测试的局限

前期投入大

后期维护开销大

不适用于业务规则复杂易变的测试

不适用于涉及人体感官和物理交互相关的测试





分配时间用于测试自动化

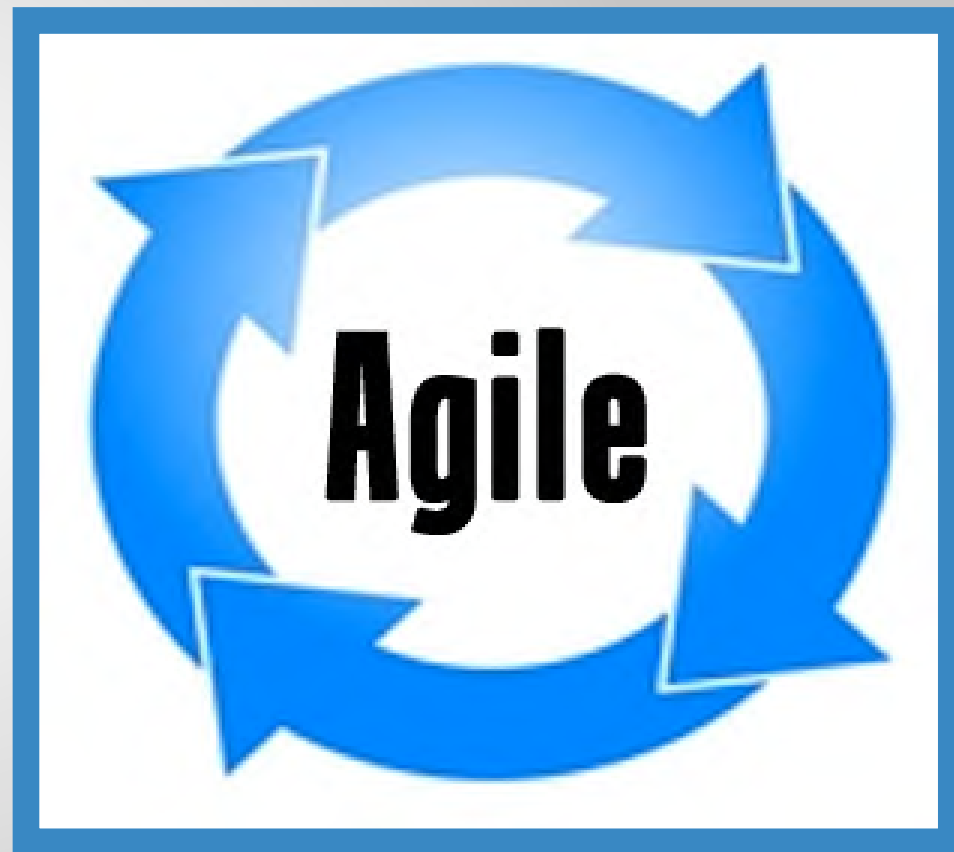
对测试生命周期自动化

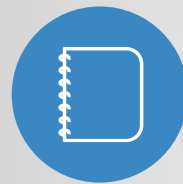
让自动化测试成为一种团队行为

持续开发和测试自动化代码

持续回归测试

用敏捷方式开发测试工具/框架





二、自动化测试的设计与实现





自动化测试的设计与实现流程



自动化测试之前的考虑

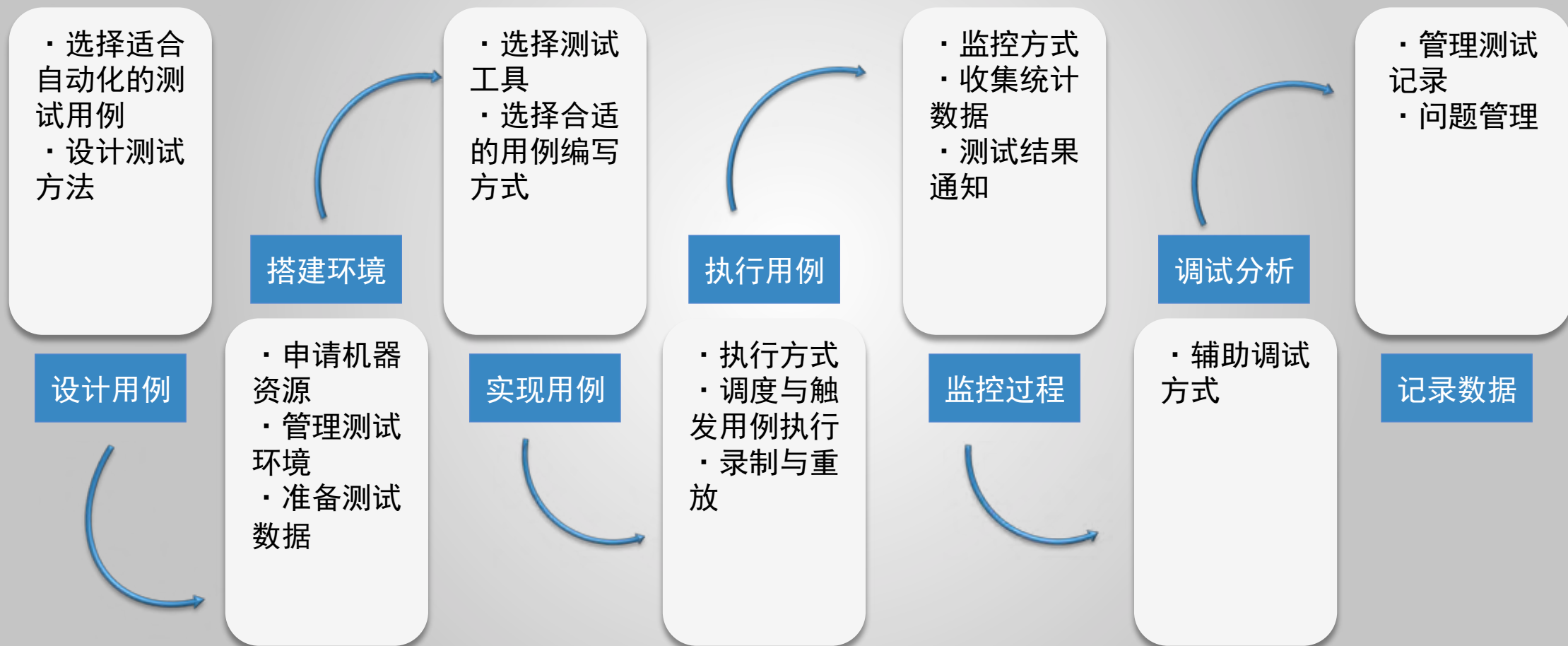


常见自动化测试方式



自动化测试工具







适合自动化的测试



单调、重复的测试工作

单元测试与组件测试

API 接口测试

负载压力测试

长时间不间断测试

GUI 底层的测试

GUI 测试?



不适合自动化的测试

可用性测试

探索性测试

一次性测试

不可测的遗留系统

功能和界面不稳定

特别复杂的业务逻辑





自动化测试之前的考虑



TiD2016

测试人员的技能水平

需要的软硬件资源

自动化方案的实现时间

能否提高测试有效性、发现更多问题

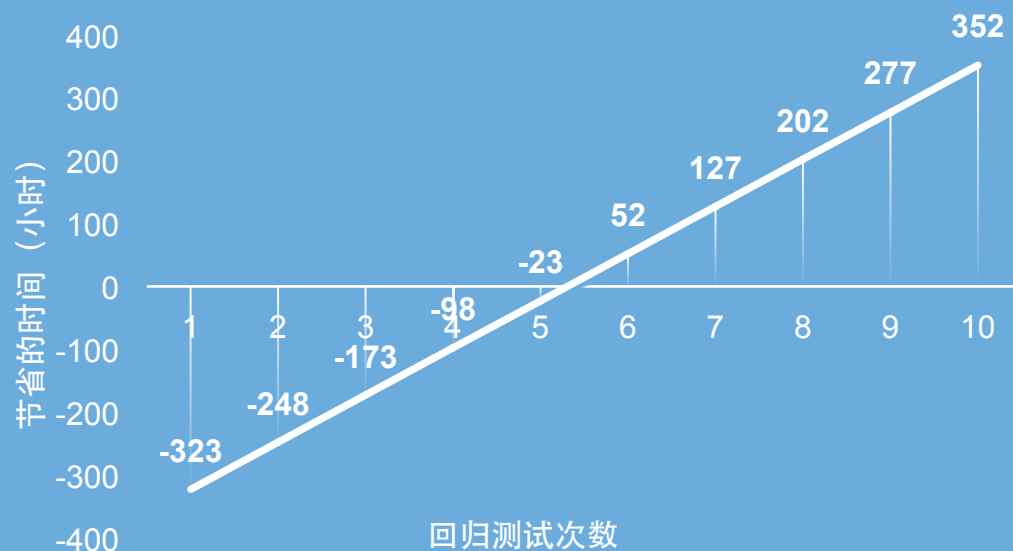
维护自动化测试的开销

组织层面的支持





自动化时间成本度量



1

- 每次回归测试产生441 (147*3) 个执行记录

2

- 基于经验估算：自动化一个执行记录需要0.9小时，手动执行一次需要0.17小时

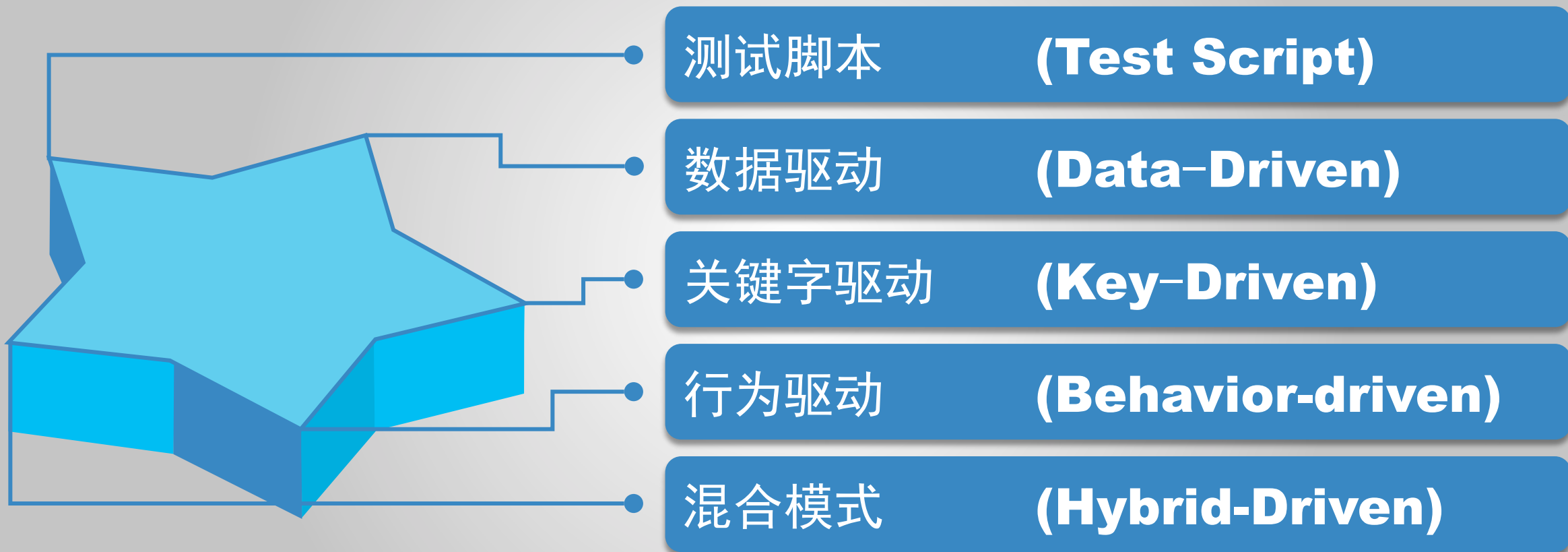
3

- 准备自动化测试的时间：441*0.9=397小时，手动测试时间：441*0.17=75小时

4

- 忽略提交bug与重现问题的时间







测试脚本



Test Script

```
Testxxx()  
{  
    set timeout 30  
    spawn rlogin -l user hostname  
    expect "Password:" {  
        send "password\r"  
    }  
    sleep 2  
    expect "# "  
    send "mkdir -p dir1/dir2\r"  
    expect "# "  
        send "echo \$\?\r"  
    send "echo \$\?\r"  
    expect "0"  
    expect "# "  
    send "rm -rf dir1\r"  
    expect "# "  
    send "exit\r"  
    expect eof  
    exit  
}
```





数据驱动



Test Restful APIs

```
- test: # create entity
  - name: "Basic get"
  - url: "/api/person/"
- test: # create entity
  - name: "Get single person"
  - url: "/api/person/1/"
- test: # create entity
  - name: "Get single person"
  - url: "/api/person/1/"
  - method: 'DELETE'
- test: # create entity by PUT
  - name: "Create/update person"
  - url: "/api/person/1/"
  - method: "PUT"
  - body: '{"first_name": "Gaius","id": 1,"last_name": "Baltar","login": "gbaltar"}'
  - headers: {'Content-Type': 'application/json'}
- test: # create entity by POST
  - name: "Create person"
  - url: "/api/person/"
  - method: "POST"
  - body: '{"first_name": "Willim","last_name": "Adama","login": "theadmiral"}'
  - headers: {Content-Type: application/json}
```





关键字驱动



*** **Settings** ***

```
Library      ShellCommand
Resource     %{ENV_ROBOT_PATH}/resource/resource.robot
Test Timeout 1 minutes
Suite Teardown GUI Logout
```

*** **Variables** ***

```
${ENV_HOST_URL} https://127.0.0.1
```

*** **Test Cases** ***

```
Terminal Test
  Issue Shell list Version
  Should Match Regexp    %{Shell_OUTPUT}    Host Version
```

```
Gui Test
  GUI Login
  View Host Page
```

... ..





行为驱动



mthread_server_static.feature

@multi

Feature: Multithreaded HTTP Server

Background:

Given the root path is "http://localhost:8088"

Scenario: Get the test page

When I visit "/test.html"

Then I should see "A test page"

Scenario: Get the Lorem page

When I visit "/lorem.html"

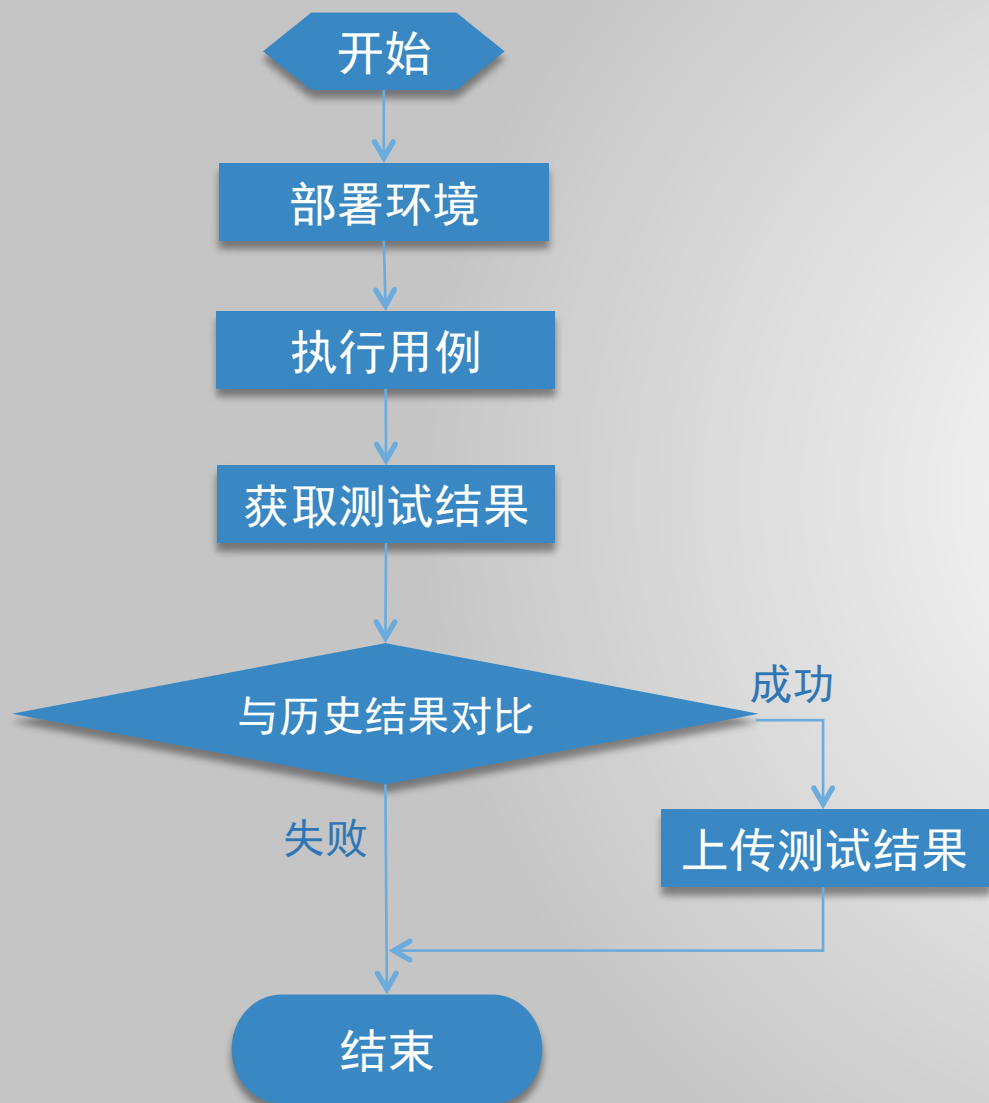
Then I should see "Quisque sit amet congue elit"

Scenario: Check CGI

When I visit "webclock"

Then I should see "00 +0000"





#1: 完全手动运行

工作量: $3 * 6 * 50 = 900$ 小时

#2: 编写少量工具

工作量: $3 * 4 * 45 = 540$ 小时

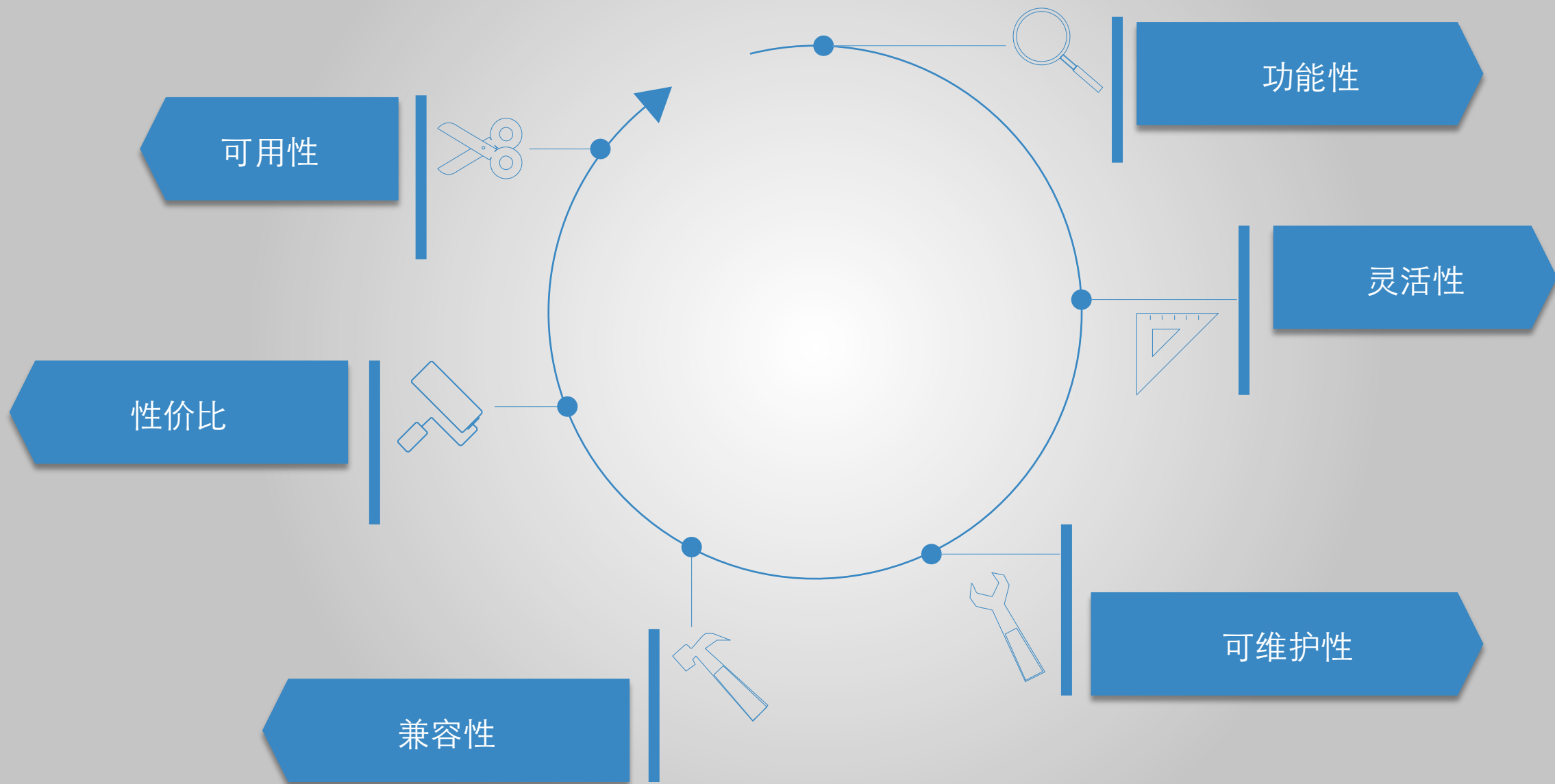
#3: 工具集成, 数据参数化

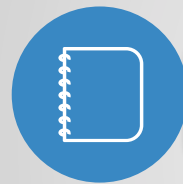
工作量: $2 * 4 * 40 = 320$ 小时

#4: 基于关键字模型自动化90%用例

工作量: $1 * 4 * 40 = 160$ 小时







三、自动化测试的管理与调度





测试资源的管理



为测试提供数据



测试用例的管理






测试用例的组合方式



测试用例的执行方式





-  测试执行过程的录制与重放
-  测试执行的触发机制与调度模式
-  自动化测试与持续集成



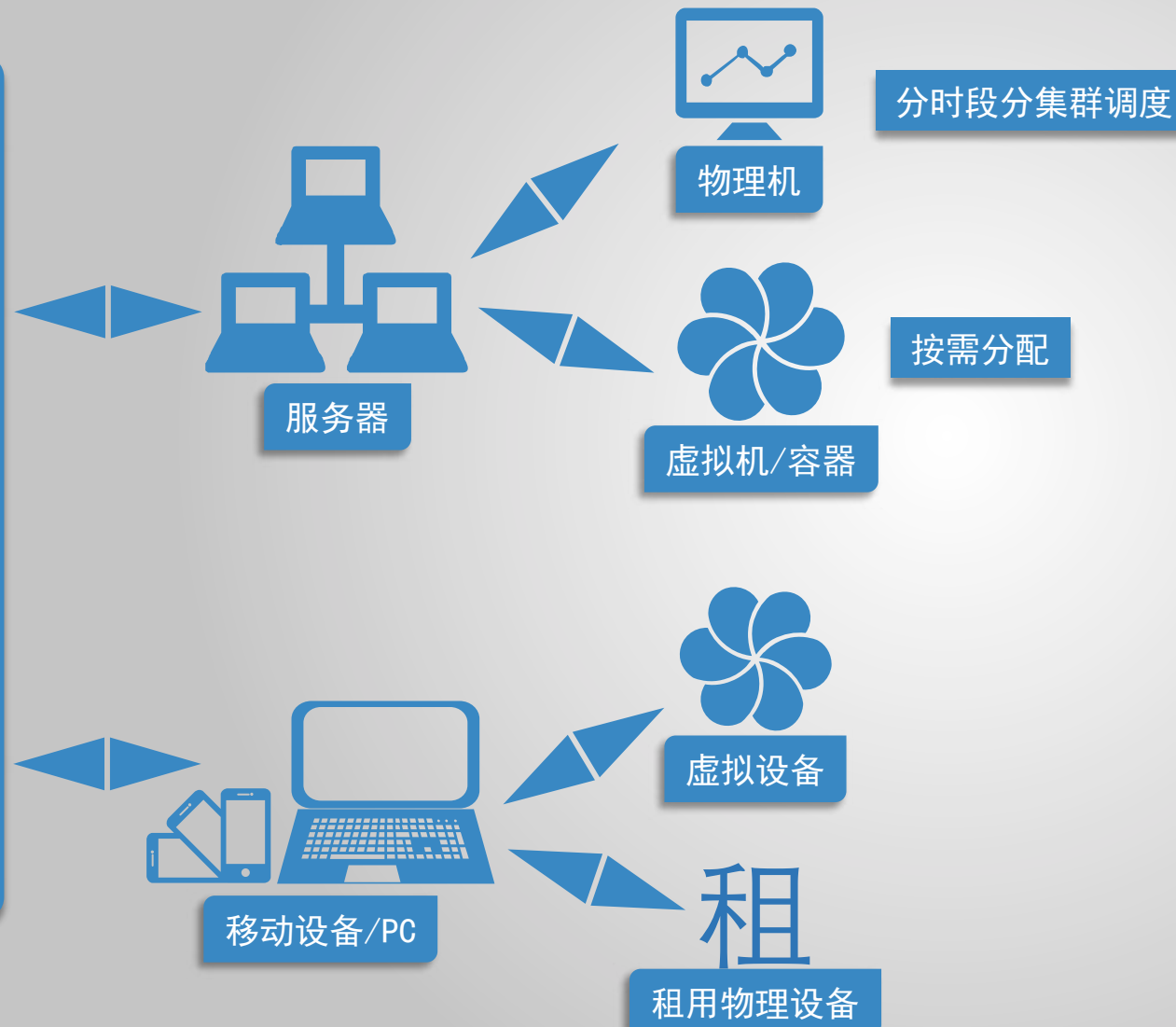


测试资源的管理



TiD2016

测试资源的管理方式



SVT Machine Scheduler

z/OS SVT Tester Dashboard :: Runlogs :: Regression Week 4/29/13 - 5/6/13 :: System Configuration

Running Configuration System Configuration Entries TN3270 Sessions Detests Dumps Logs System Performance Network States Workload States

Date and Time	CPC	Machine	LPAR	CPs	LPs	LAAPs	Reserved CPs	Storage (MB)	Reserved Storage (MB)	LPAR Storage (MB)	Load Factor	Release
2013-04-29 10:44:07	000000429.01	C0K	20	2	4	0	40.80	0	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000429.01	C0S	30	2	4	0	40.80	0	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000429.01	C0K	20	2	4	0	40.80	0	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000429.01	C0S	30	2	4	0	40.80	0	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000600.07	C0D	34	2	4	1	20.40	0	100 / 20 / 999	100 / 20 / 999		VSR11
2013-04-29 10:44:07	000000600.07	C0B	11	2	4	1	40.80	10.80	444 / 100 / 999	444 / 100 / 999		VSR11
2013-04-29 10:44:07	000000600.07	C0A	40	2	4	1	40.80	10.80	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000600.07	C0K	20	1	1	0	30.70	0	500 / 10 / 999	500 / 10 / 999		VSR11
2013-04-29 10:44:07	000000600.07	C0K	35	1	4	1	30.70	10.80	500 / 100 / 999	500 / 100 / 999		VSR11
2013-04-29 10:44:07	000000429.01	C0K	20	2	4	0	40.80	0	500 / 100 / 999	500 / 100 / 999		VSR11

Date and Time CPC Machine LPAR CPs LPs LAAPs Reserved CPs Storage (MB) Reserved Storage (MB) LPAR Storage (MB) Load Factor Release

SVT Machine Scheduler

Today

Legend: SVT FIT SVT Special FIT Special Unavailable * = bad time assignment

Date	Shift	C	Team
05/28 Sat	1 Now		Team
	2		Team
	3		Team





一个测试集群的组成

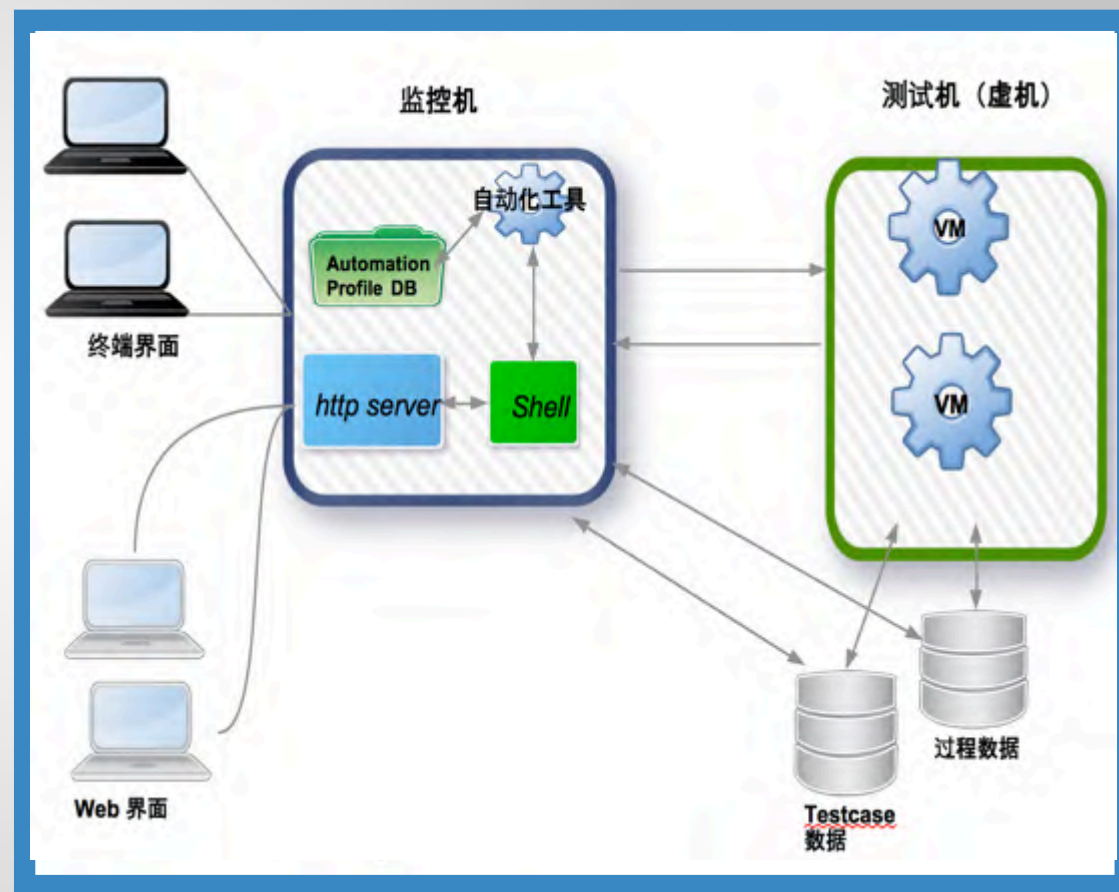
控制机：发起或停止自动化测试。

监测机：监控测试过程与结果，检查测试机健康状况。

测试机：虚拟机、执行测试用例。

数据存储：存储测试数据、脚本、环境配置。

共享存储：存储测试过程产生的log和dump。





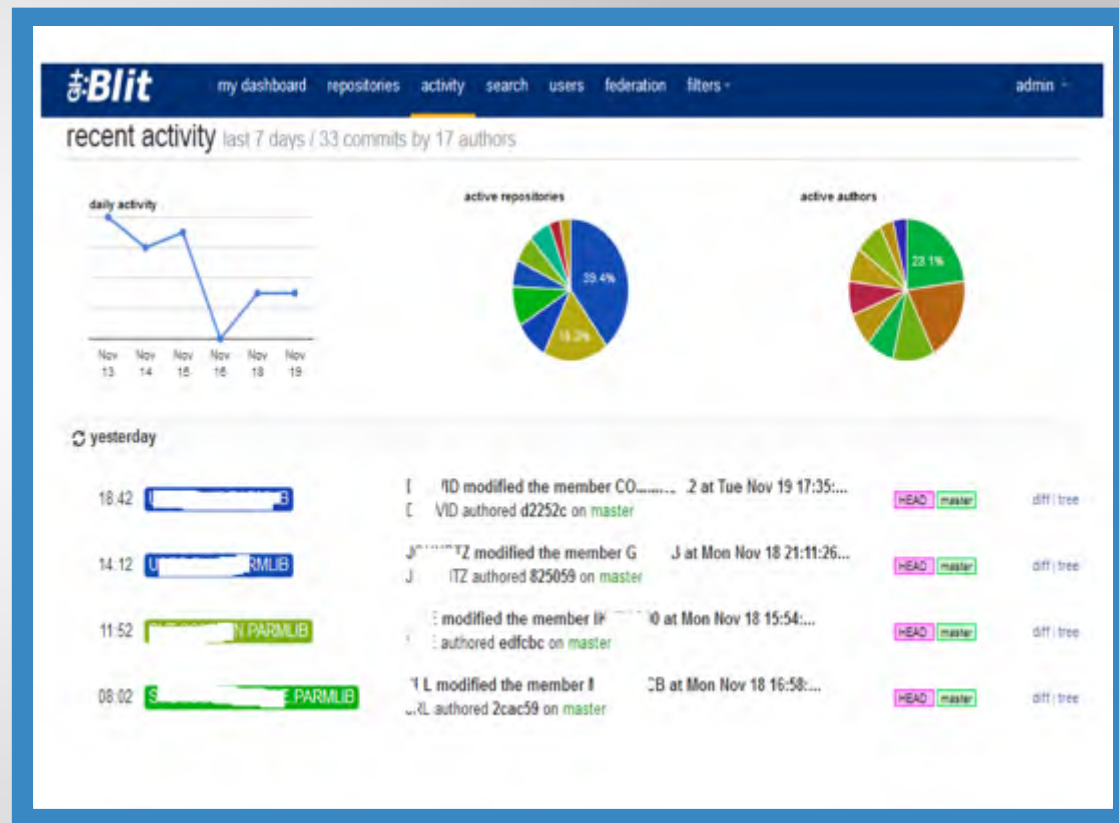
测试环境

环境配置版本管理

监控重要的环境配置文件

参数化环境配置

通过自动化工具布署测试环境





避免在测试脚本中将数据硬编码

为每次测试新建、销毁数据

使用内存数据库（文件系统）或者SSD加速测试

自动生成数据的方式

录制工具

脚本程序

Fuzzing（半随机化）

数据库

利用专用测试工具/包

调用产品接口生成数据





生成测试文件

```
dd if=/dev/random bs=1024 count=1024 of=test.file
```

```
dd if=/dev/zero bs=1048576 count=4096 of=4g.file
```

正规表达式生成

```
$ regen -n 2 '0x[\da-f]{16}'  
0x8f5858102a5ce124  
0x3e4c9fee6c9f419d
```

```
$ regen -n 3 '[a-z]{6,12}(\+[a-z]{6,12})?@[a-z]{6,16}(\.[a-z]{2,3}){1,2}'  
abxfcomj@uyzxrgj.kld.pp  
vzqdrmi@ewdhsdzshvxxjk.pl
```

利用ODM技术从 NoSQL获取数据

```
type Person struct {  
    Id          bson.ObjectId `bson:"_id,omitempty" json:"id,omitempty"`  
    Name        string      `bson:"name,omitempty" json:"name,omitempty"`  
    Password    []byte     `bson:"password,omitempty" json:"-"`  
    Type        int        `bson:"user_type,omitempty" json:"type,omitempty"`  
    Email       string     `bson:"email,omitempty" json:"email,omitempty"`  
    Team        string     `bson:"team,omitempty" json:"team,omitempty"`  
}  
  
func TestGetUserData(t *testing.T) {  
    u := mgodb.GetPersonByName(testUser)  
    // Test code below  
    ...  
}
```







- 测试用例分类管理

- 环境配置参数化

- 利用虚拟机运行回归测试集



- 事件/定时调度测试行为

- 实时监控测试进度

- 及时收集第一手问题数据

(FDDC)





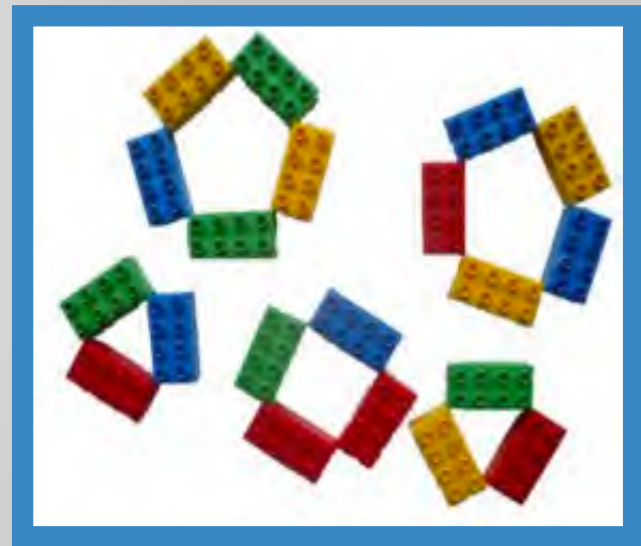
组合自动化测试用例的好处

- 快速生成新的用例
- 回归测试排序，早发现问题
- 动态调整测试对象和测试强度
- 提高覆盖率



组合方式

- 顺序组合
- 标签选取 (Exclude/Include)
- 根据分组与标签组合
- 随机组合
 - 普通随机
 - 带权值的随机
- 混合方式





同时利用Lottery scheduling算法和Tag组合用例

total = 20

random [0..19] = 13



↑
winner

```
$ wkldgen -algo lottery -exclude above -template uss
```

测试用例集成模板

Template for USS Component

Setup Jobs&Commands

[[SETUP BEGIN]]

SUBMIT 'SETUPJCL(SETUP)'

SUBMIT 'SETUPJCL(WORKLOAD)'

[[SETUP END]]

Environment parameters

[[ENV BEGIN]]

SUBMIT 'SETUPJCL(SETENV)' [[ENV:LOWMEM]]

SUBMIT 'SETUPJCL(SETENV)' [[ENV:64bit]]

SUBMIT 'SETUPJCL(SETENV)' [[ENV:32bit]]

[[ENV END]]

Testsuite

[[TESTSUITE BEGIN]]

KERNEL [[TICKETS:50]] [[TAG:UNIX]]

FILESYS [[TICKETS:10]] [[TAG:64bit]]

... ..

[[TESTSUITE END]]





- 一次执行一个用例：常用于单元测试与功能测试、确保100%的成功率
- 一次执行多个用例：压力测试\负载测试\并行测试
- 多核执行：性能测试
- 分布式执行
 - 分布式执行工具（Ex. Selenium Grid）
 - Docker 容器集群
 - 测试云

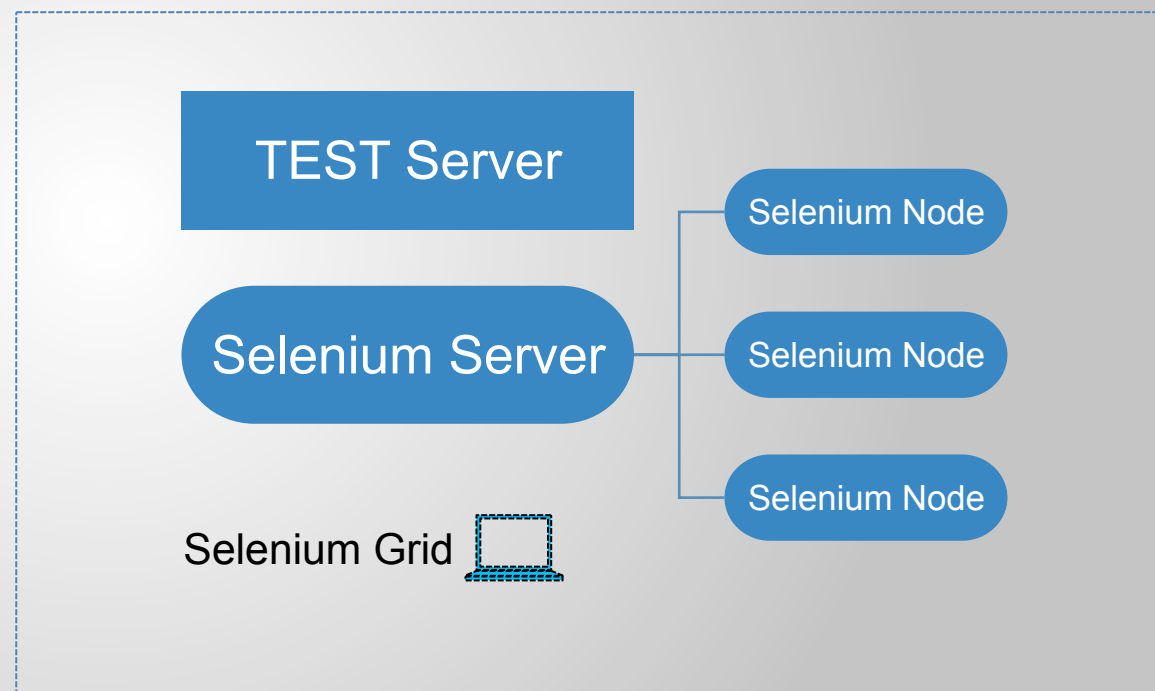


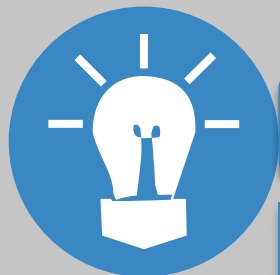


并行执行GUI回归测试，加快反馈周期

多机环境：Selenium Grid

单机环境：Headless Selenium





录制重放方式

- 日志录制与重放
- 打点记录过程
- 网络流量或者IO行为的录制与重放
- 将测试环境与录制脚本绑定
- 自动化测试脚本 != 录制/重放工具



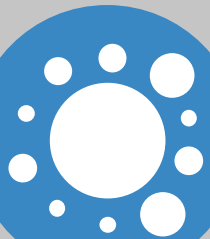
带来的好处

- 模拟用户行为
- 收集数据做Mock
- 重现问题
- 验证补丁





人工触发



事件触发

- 代码提交之后执行单元测试
- 构建之后执行功能测试
- 自定义事件



无人值守定时触发

- 执行一次 (Once)
- 重复执行
 - 按日、周、工作日 (Daily, Weekly, Week Days)
 - 重复次数
- 根据配置文件设定多种调度方式
 - 支持ISO8601 (时间日期表示标准)





调度工具



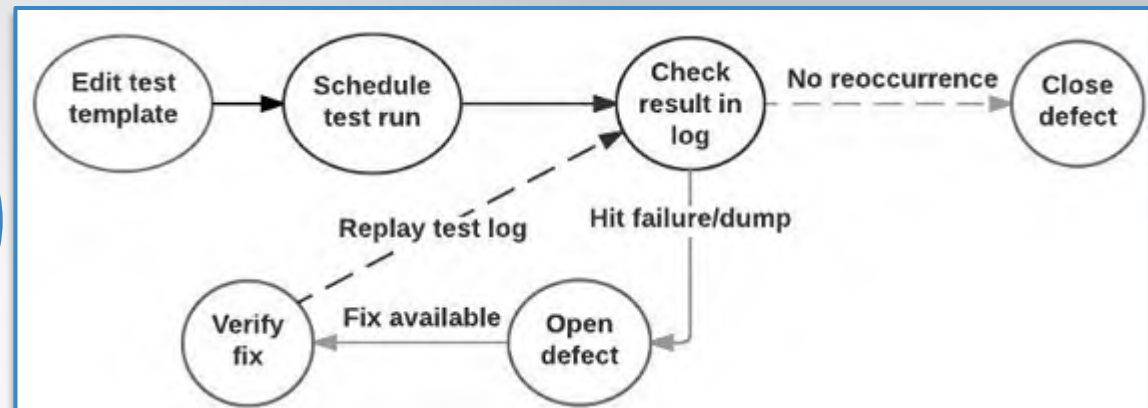
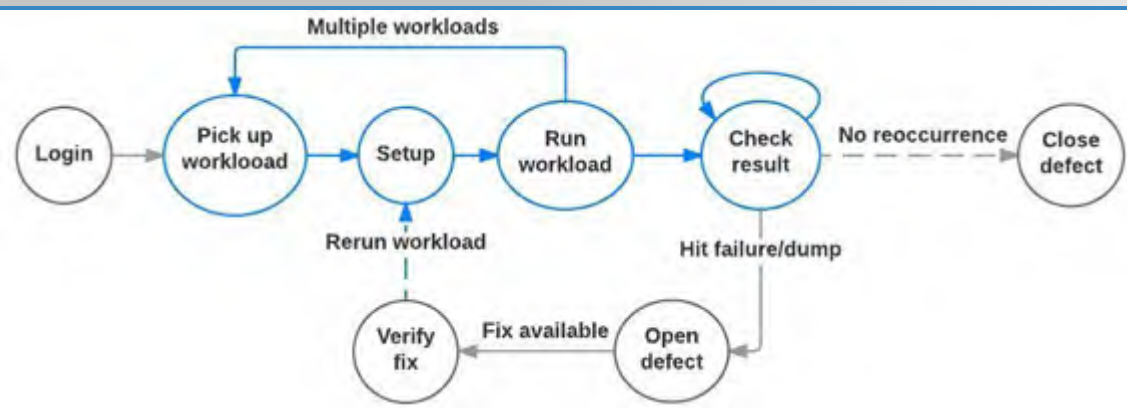
Crontab

Chronos (分布式环境)

CI工具, 如Jenkins

编程实现调度





通过模板跨团队共享测试集

根据标签和权重随机生成测试负载

读取测试log重放测试过程

用于执行长时间运行的回归测试

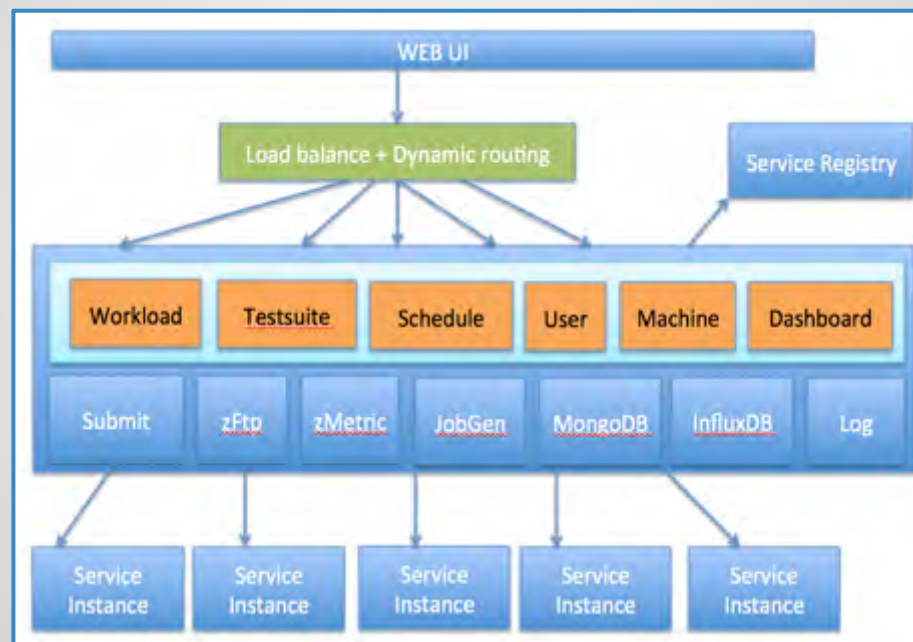
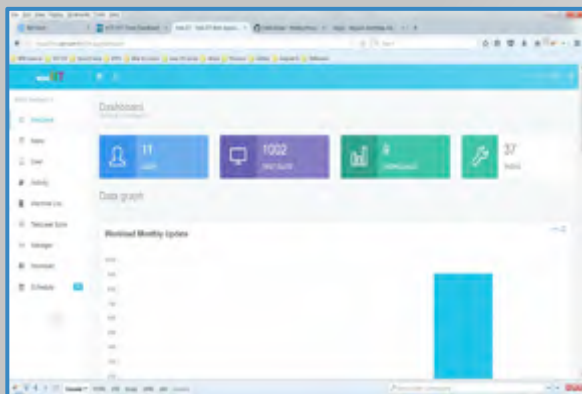
通过定时任务向多台机器提交测试任务

以模板的方式管理测试用例和测试环境



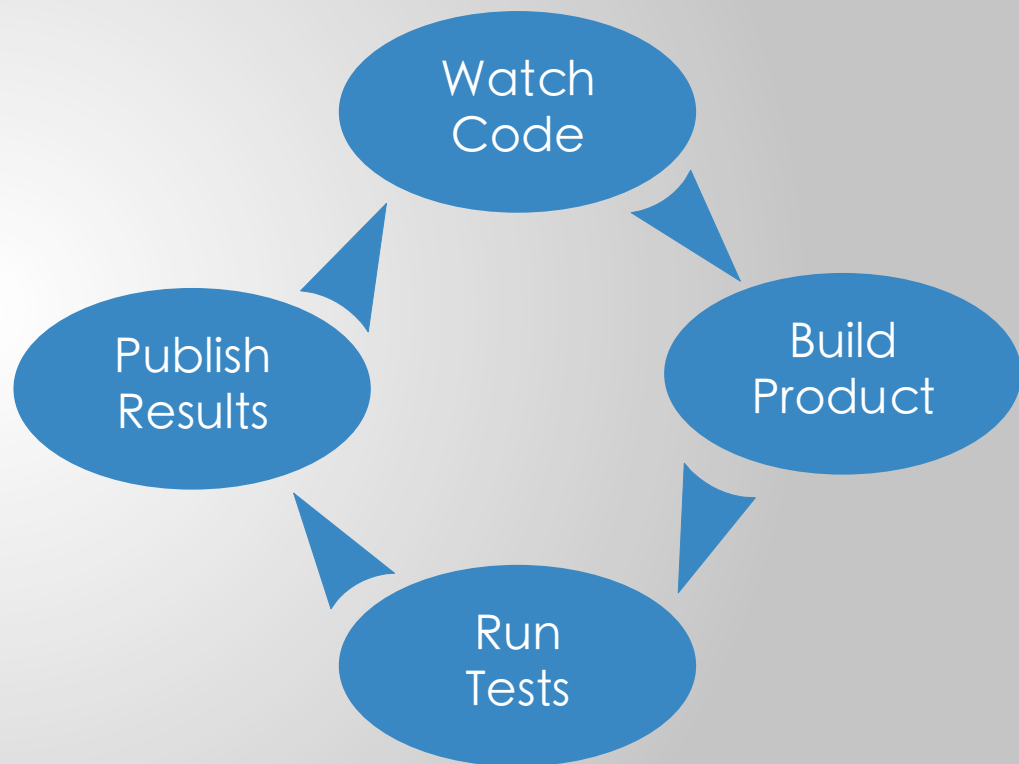


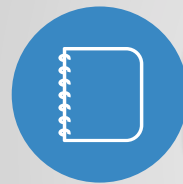
实际案例：TestHub





- 保证快速反馈
- 高覆盖率的单元测试和接口测试
- 适当引入自动化验收测试
- 防止引入不稳定的测试用例
- 设立提交阶段失败标准
- 运行自动化测试集之前检查环境
- 运用虚拟化技术（虚机/容器）并行测试
- 持续调整集成测试的方式
- 需要持续做压力测试和负载测试？





四、自动化的监控与记录





测试过程监控



测试结果通知



辅助调试手段



测试记录的管理



问题管理





结果监控内容

- 测试用例运行状态
- 被测系统运行状态
- 底层操作系统/中间件/硬件运行状态



监控相关技术

- 测试用例运行状态：框架提供支持，在测试脚本里加入Hook
- 系统状态监控：Zabbix/Nagios以及各种Agentless工具
- 日志收集与分析：ELK (ElasticSearch + Logstash + Kibana)
- Metric收集与显示：InfluxDB + Grafana





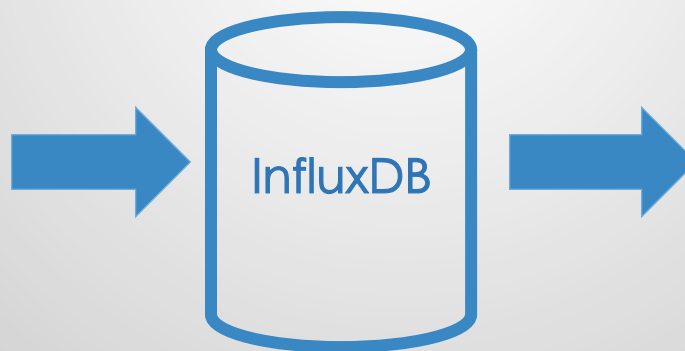
InfluxDB + Grafana

“We found a **BUG** this week that literally would have taken us weeks or months to discover if we didn't have Grafana”

——Bret Copeland, Team Lead, StackOverflow

测试脚本

```
#!/bin/sh
TIMESTAMP=`date +%s`
METRIC=metric.test.thing
VALUE=10
echo $METRIC $VALUE $TIMESTAMP |
nc HOST_OR_IP_OF_INFLUXDB_SERVER 2003
... ..
```





测试过程监控



TiD2016

操作记录Runlog



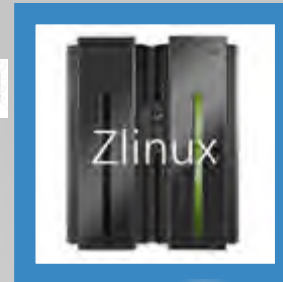
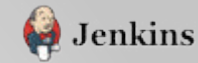
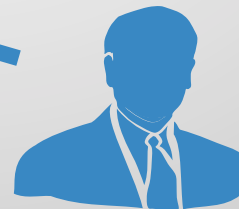
人工记录



自动记录-机器人程序



自动收集监控数据

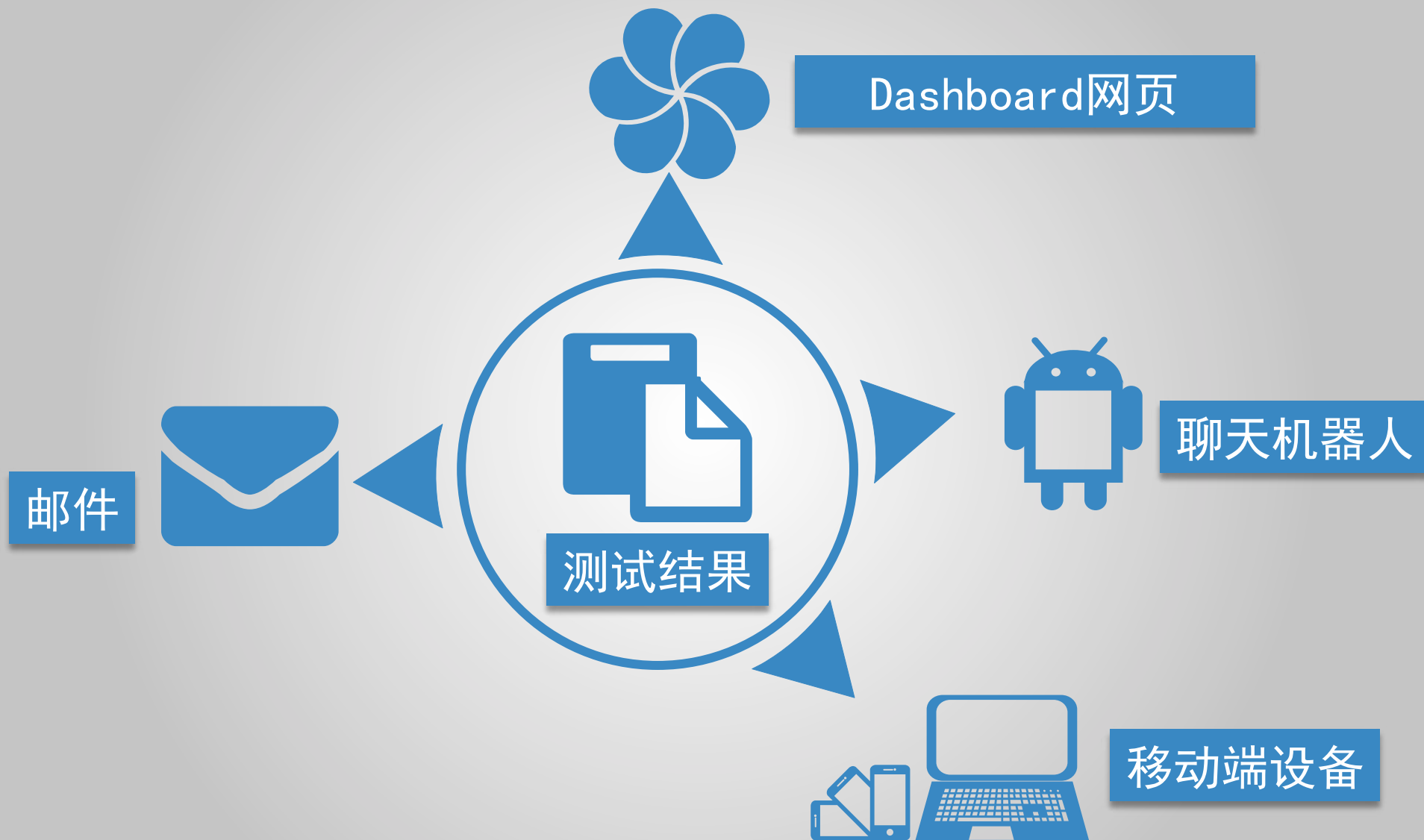




测试结果通知

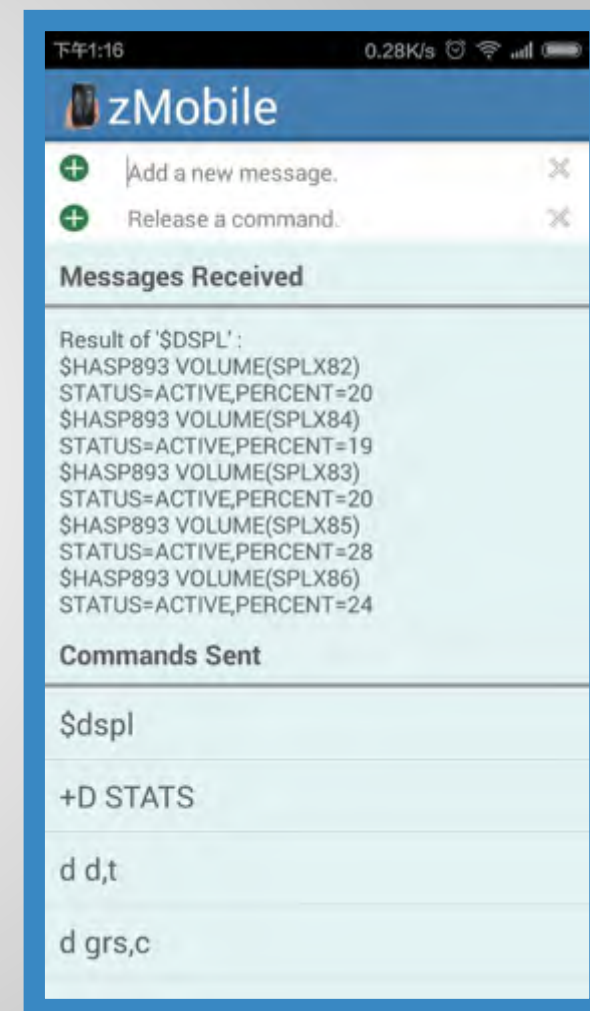
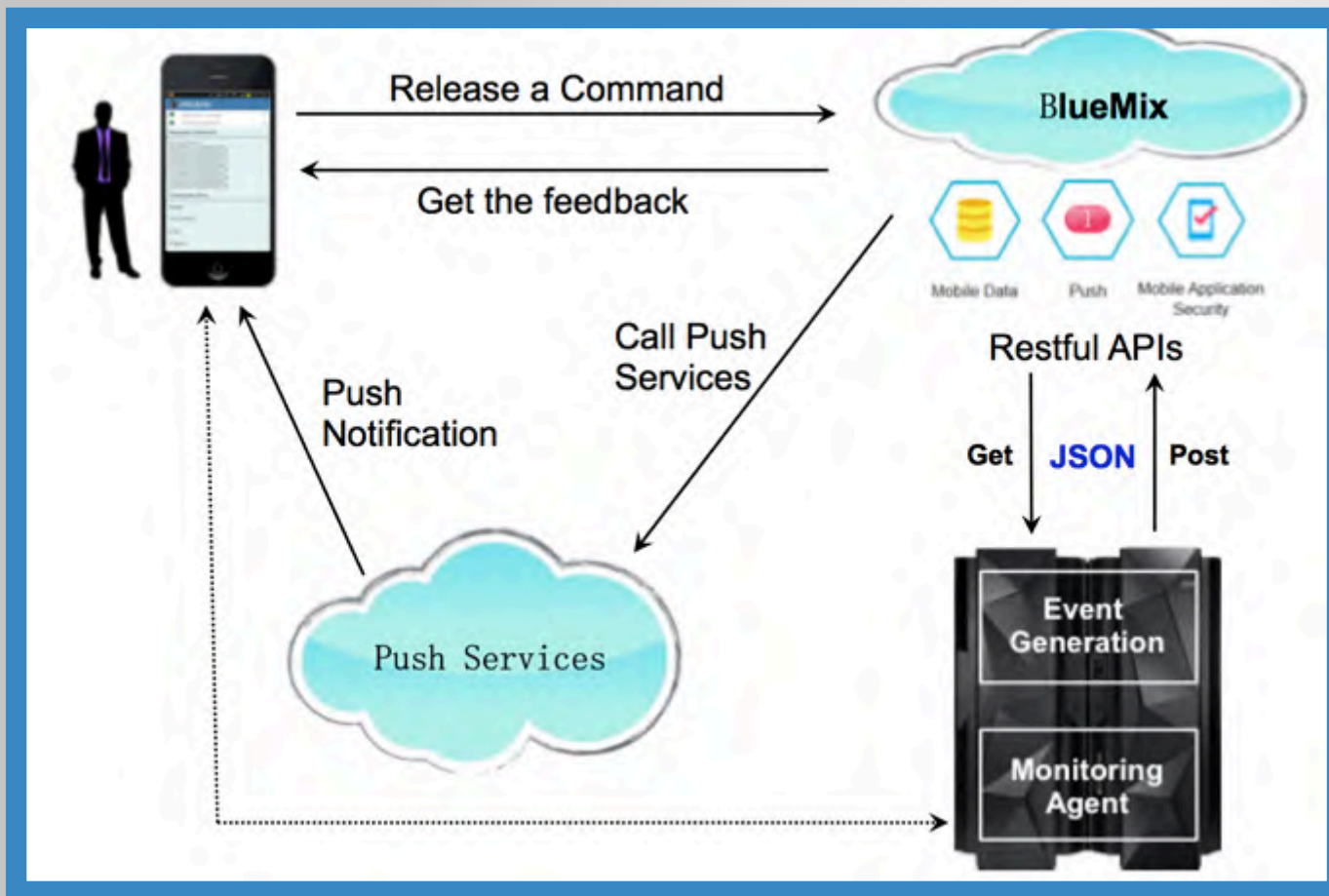


TiD2016



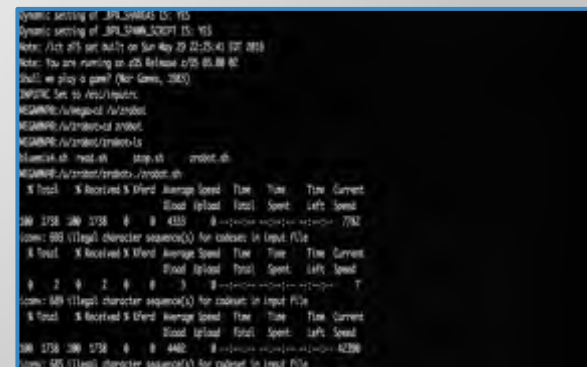
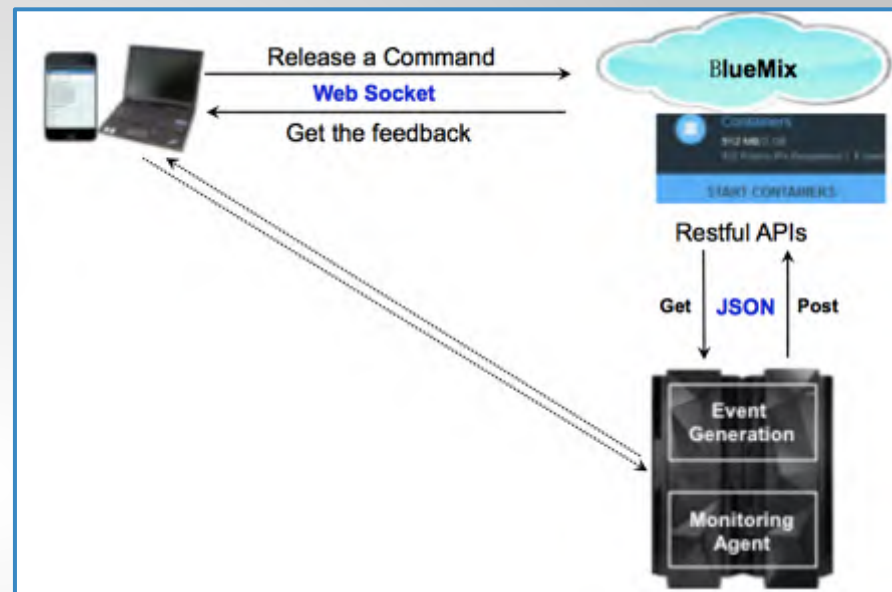


运用云计算与移动技术





- 更友好的人机对话界面
- 更直观的信息集成方式
- 执行自动化脚本，返回测试结果
- 支持多种类型的操作命令
- 实时监控测试系统状况





实践方法

- 查看log、dump、trace等记录定位出错位置
- 每个虚拟机跑一个用例，遇到错误暂停虚拟机（FDDC）
- 用例解耦以缩小调试范围
- 常见问题总结文档



工具

- 信息收集工具
- 数据查看/分析工具





测试记录

- 系统日志 (syslog)
- 系统Dump
- 操作记录 (Runlog)
- 用例执行输出 (testcase log)
- 执行结果 (failed/successful)
- 执行次数、耗时、进度
- 过程监控记录



用途

- 调试
- 审记
- 工作量与成果度量





DDL (Dynamic Dump List):

- 抽取Dump的基本信息以固定格式显示
- 集中展现所有测试人员处理过的Dump
- 比对Dump历史信息，列出之前的处理方式

关键信息

Where: 机器, 版本

When: 时间日期

Who: 测试人员、历史信息的相关人

What: 出错模块、错误返回码、系统状况

Why: 历史信息 (bug/环境问题/case问题)

抽取Dump关键信息

历史信息比对

显示类似问题

```
SV15833 16/04/08 20:00:58 NM= 2964 007F27
sym= AB/S0EC6 793 BPXINPVT BPXTXFSR BPXMIPCE REGS/0E017 FFFF
ttl= COMPON=BPX,COMP ID=SCRX1,ISSUER=BPXMIPCE,MODULE=BPXTXFSR+1D34,ABEND=S0EC6,

ZD003N2V,SV15834 16/04/10 03:45:56 NM= 2964 16D3C7
sym= AB/S07B0 519 IEANUC01 HISNMT HISNMT REGS/FFFFFF FFFF
ttl= COMPON=HIS,COMPID=SCHIS,ISSUER=HISNMT

ZZ032043,SV15835 16/04/06 10:34:33 NM= 2964 0240F7
sym= AB/S00D6 027 UNKNOWN UNKNOWN DSNTFRCV REGS/0C1BC FFFF
ttl= DBX1,ABEND=0D6-00000027,U=QREPSVS,M=N,C=111,RDS=SQL,M=DSNTFRCV,PS

SV15836 16/04/10 23:10:05 NM= 2964 0B40F7
sym= AB/S00C6 006 KLV$RSS KLV$RSS REGS/0E35E 0B260
ttl= CT/ENGINE ABEND S0C6 U0000 AT AF80F6E8 (KLV$RSS+330)

SV15837 16/04/10 23:09:43 NM= 2827 04B8A6
sym= AB/S00C6 006 KLV$POST KLV$POST REGS/0B06A FFFF
ttl= CT/ENGINE ABEND S0C6 U0000 AT AEA08B02 (KLV$POST+132)
```





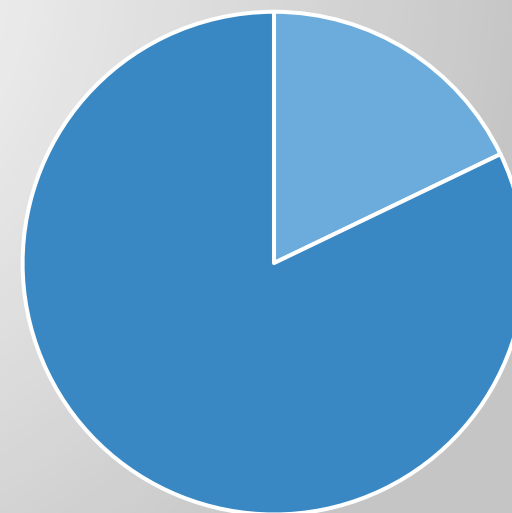
抽取Bug库的Dump关键信息，基于简单贝叶斯模型计算有效/无效dump的概率。

DEMO

```
-----Bayes Classifier-----
Please input the abendCode :
500C4
Please input the reasonCode :
03B
Please input the issuer:
IXGR1REC
=====Result=====
The probability of valid is 82.070000%.
The probability of invalid is 17.930000% .
----Result : the RootCause is Valid
=====
-----Bayes Classifier-----
```

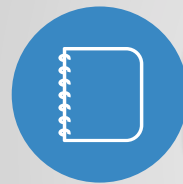
关键信息

<u>Abend Code</u> :	错误码
<u>Reason Code</u> :	错误原因
<u>Issuer</u> :	出错模块
<u>Machine</u> :	测试机器名
<u>Root Cause</u> :	是否为Bug



Invalid
Valid

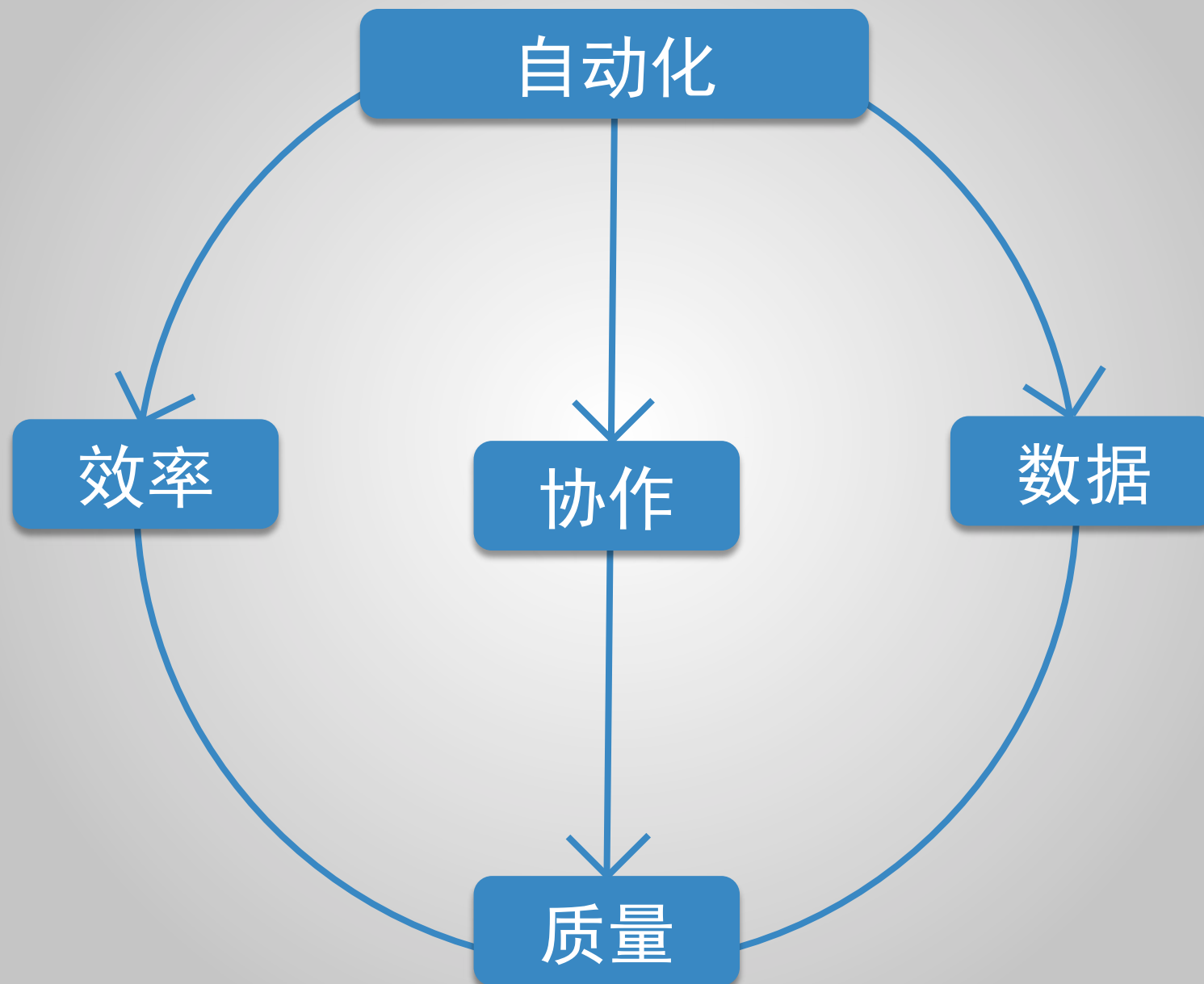




五、总结







谢谢