面对组件化测试的探索
——智能化测试框架的打造

——“将自动化测试从回归阶段拉回正轨”

目录
CONTENTS

01 自动化的现状

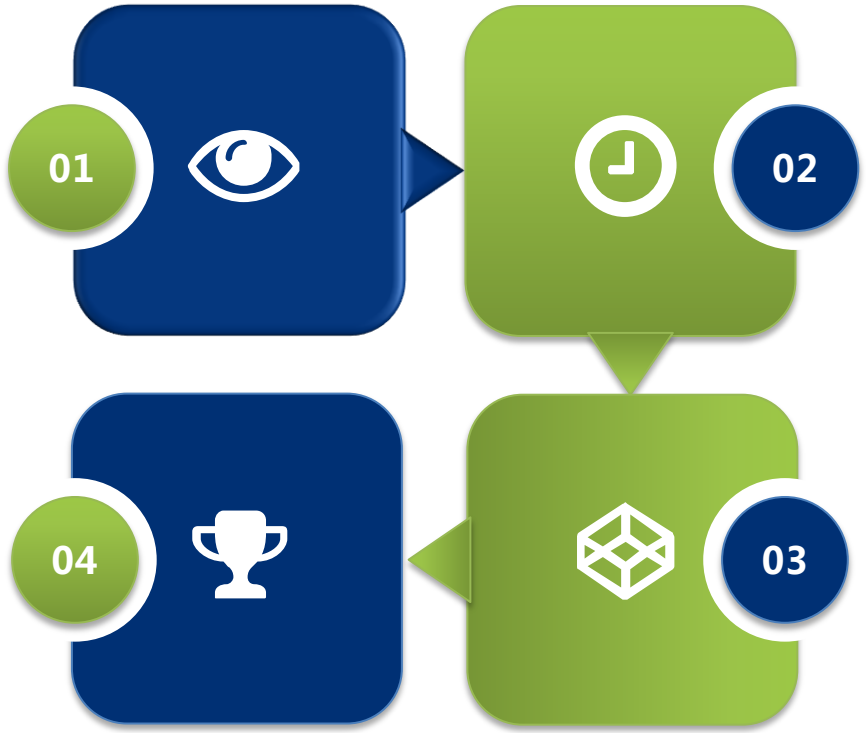
02 对测试框架的要求

03 自动的自动化框架

自动化被迫沉到了回归测试

手工测试工程师

撰写测试用例
执行测试



测试开发工程师

依据手工测试case编写脚本
撰写测试数据
调试测试脚本

测试开发工程师

运行测试脚本
回归测试

测试开发工程师

提交测试脚本
维护测试脚本

质量保证部门的困境

技术基础

代码基础薄弱

学校学习的内容几乎淡忘

工作饱和

工作任务繁重

业余时间都所剩无几

自动化太难

开始过很多次学习都无疾而终

业务测试主导

测试工具熟练

类似postman、fiddler工具

框架无法选择

TestNG、Junit等

CI/CD,DevOps各种新概念已经应接不暇

目录
CONTENTS

01 自动化的现状

02 对测试框架的要求

03 自动的自动化框架

对测试框架的要求

减少开发成本

减少框架的维护成本
减少平台的维护成本
尽量成熟工具引入二次开发，避免重复

降低测试脚本开发难度

可以快速学习快速上手
拿来即用

完全解耦

独立即可运行
集中调度即可运行
CI调度也可运行

减少测试脚本的开发成本

越少人参与测试用例开发
越少人的工作耗费在脚本撰写

随时测试

可以定时执行测试
可以按需求执行测试
可以按构建执行测试

降低测试用例维护成本

降低测试用例修复成本

需满足的个人需求

简单

学习成本低

- 简单开发语言基础
- 掌握测试用例设计理论
- 测试用例设计熟练
- 无需大量高深的基础

现有习惯

3年以上功能测试

- 使用过很多测试工具
- 做过很多测试项目
- 执行过很多测试用例

快速

边界的脚本编写

- 拿来即用
- 初次不影响测试项目
- 快速得到结果

目录
CONTENTS

01 自动化的现状

02 对测试框架的要求

03 自动的自动化框架

自动的自动化框架概述

工具化

- 设计参数池类（数据驱动封装）
- 检查点类（断言封装）
- 关联类（串联单接口上下文）

自动化

- 测试脚本自动生成算法
- 数据TDS

定制化

- 接口测试的IDE插件
- 模板化设计
- 内部协议模板化处理

自动的测试框架的优越性

普通RPC接口测试的入门成本

1、理解maven工程结构

2、完全理解spring

3、掌握TestNG

3、掌握Java语言基础

4、掌握面向对象思想

5、掌握注解

6、了解RPC服务的原理

VS

AAT的入门成本

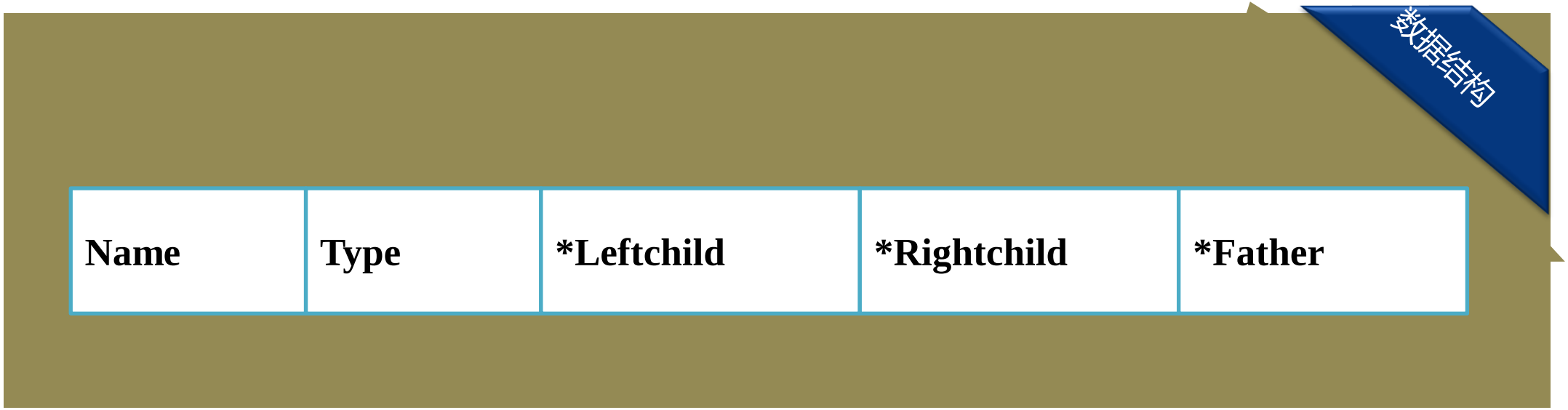
1、掌握java语言基础

2、熟练记住AAT框架的使用规范（类似工具的使用手册）

从数据结构开始

自动脚本生成的二叉树数据结构节点

在存储节点中包含了名字，类型，左孩子指针，右孩子指针和父亲指针。通过前驱二叉树的生成算法，生成一棵前驱线索二叉树。其中前驱所指为指向父亲节点的指针。



测试脚本自动生成过程

以具体接口为例简述树的生成

public String setPersion(Stirng sName,Integer iAge,HouseHold household);

其中户口类HouseHold的字段（类成员）部分如下：

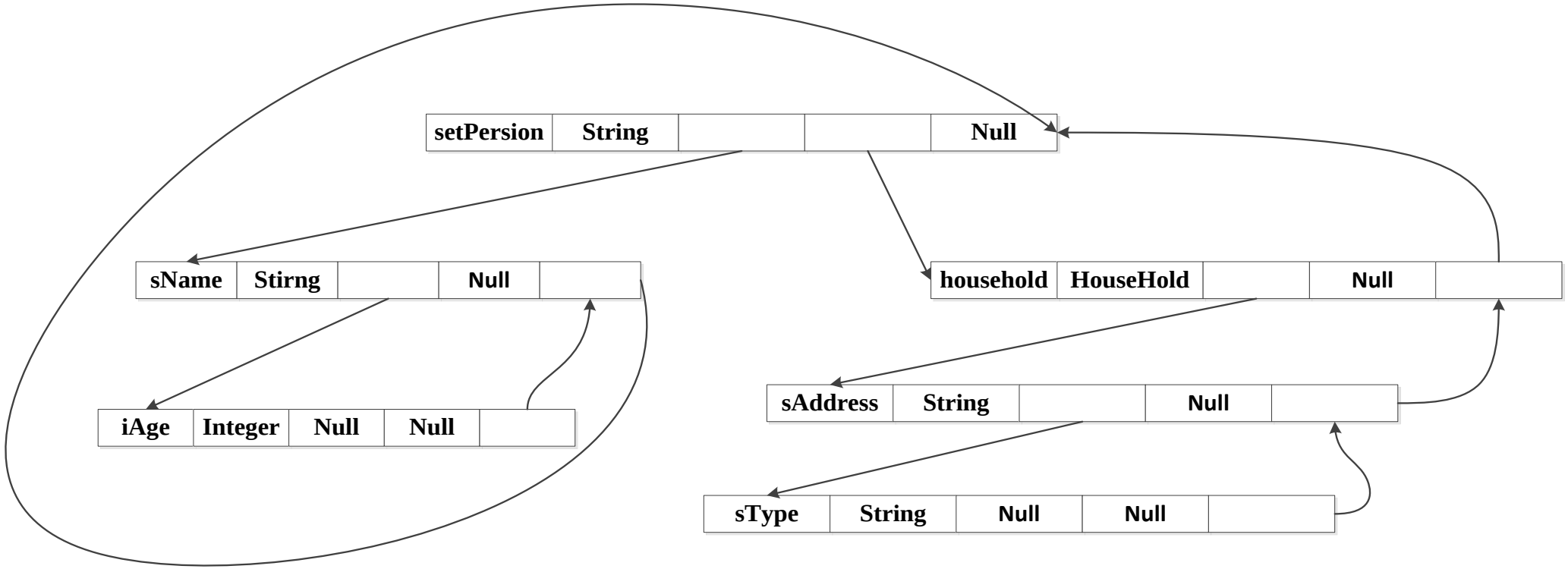
Public class HouseHold{

public String sAddress；//户口地址

public String sType;//户口属性（农业，非农业）

……

}



完成一个小目标

```java
public class TestSetPersion extends BaseTest{
    private static final Logger log = Logger.getLogger(TestSetPersion.class);

    @Resource
    PassengerJSFService jsfPassengerJSFService;

    @Test(dataProvider = "defaultMapDataProvider", dataProviderClass = DefaultMapDataProvider.class)
    @ParamFile(path = "/PassengerJSFServiceParam.xlsx", sheetName = "setPersion")
    public void testSetPersion(Map<String,String> excelData){
        TestParamPool testParamPool = new TestParamPool(excelData);
        HouseHold arg0 = new HouseHold();

        Integet arg1 = testParamPool.getInt("arg1")
        String arg2 = testParamPool.getString("arg2")
        String sAddress = testParamPool.getString("sAddress");
        String sType = testParamPool.getString("sType");


        arg0.setAddress(sAddress);
        arg0.setType(sType);

        Result response = jsfPassengerJSFService.setPersion(arg2,arg1,arg0);
        // add some log and assert

    }
}
```

travelbox / api / train / TrainPassengerJSFServiceTest / TestAddPassenger.json

TestRemoveContact.java × AutoGenDemo.java × src\···\TestAddPassenger.java × Te

arg1

```
1
2   [{"arg0":{"sAddress":"str","sType":"str"},"arg1":"int","arg2":"str"}]
```

TestDataService

1 模糊

参考模糊测试数据规则
定义不同类型的模糊数据mock
包含：

- **远程执行注入规则**
- **HTTP投注如规则**
- **文件注入规则**
- **通用模板注入规则**
- **随机字符规则**
- **SQL注入规则**
- **XSS注入规则**

2 数据实体

定义各种内部数据实体：

Person
Sku
Address
ExpressDilivery
Order
Payment
Base（random）

3 智能

通过拦截器获取生产数据（输入和输出）
脱敏后存入大数据平台
通过引入聚类分析算法标记数据
通过参数语义将标记数据拆分成元数据入智能数据mock仓库
通过智能数据mock对外提供服务

1、在第一次将被测接口加入测试平台时，测试开发工程师需要对改接口的入参做语义标记
2、无法分析语义的参数会使用对应的随机数据服务
3、特殊需求的时候，需要人工配置调用模糊数据mock

参数数据重组策略

精简策略

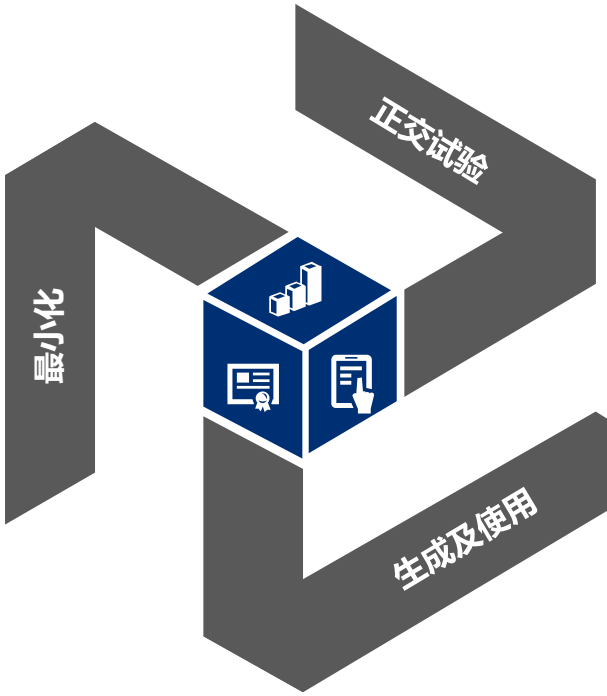
去重后按顺序组合

将每一个参数内部进行去重

取出全部参数第一行作为第一个条case

以此类推

如果遇见参数A的水平大于B，那么B再次循环



最大覆盖策略

去重后正交

将每一个参数内部进行去重

调用正交试验设计测试用例方法生成测试用例

默认策略

生成及使用

不做任何处理

数据契约

Test Data Service Interactive language

Test Data Service Interactive 是数据契约的达成载体和实现方式。

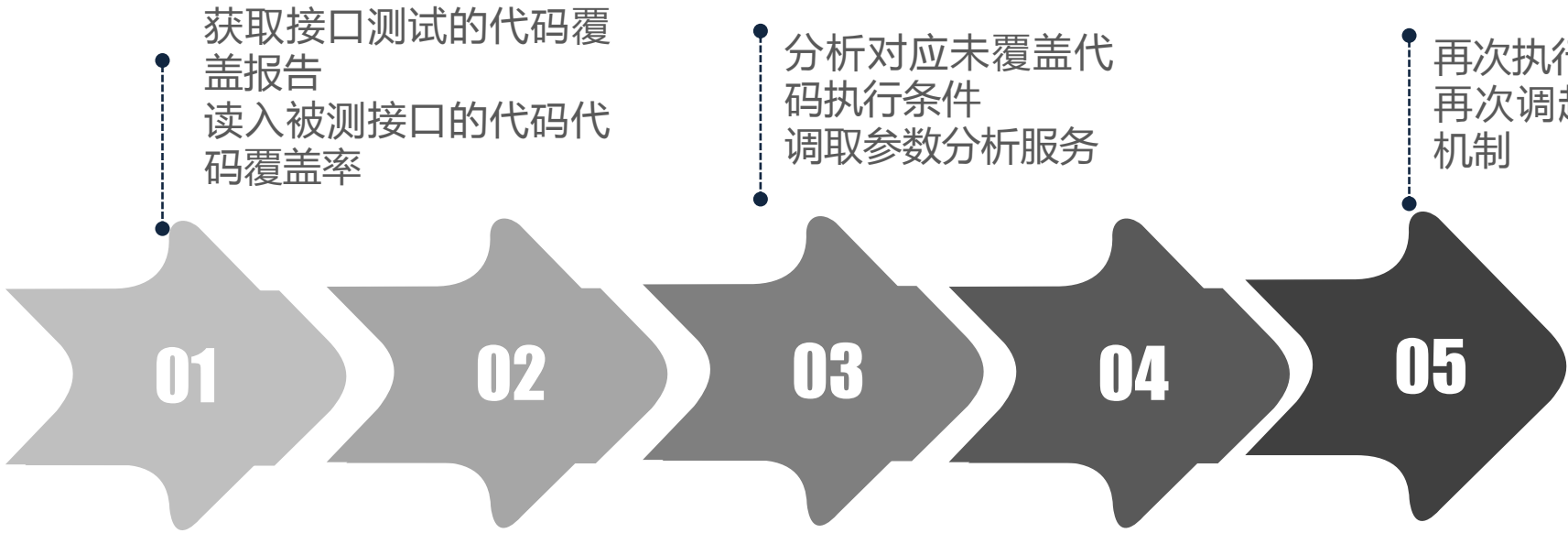
• 有约束查询

- person.name,person.last_name,address.city where person(name=crisschan,last_name=chan) and helfmoon(hsize=10)
- 是依据传入的Meta Data Provider的名字和要获取的属性同时依据条件产生对应的数据。其中HelfMoon关键字是对一次查询的全部数据的约束
- [["person.name", "person.last_name", "address.city"], [["crisschan", "chan", "怀化市"], ["crisschan", "chan", "张家界市"], ["crisschan", "chan", "新余市"], ["crisschan", "chan", "长春市"], ["crisschan", "chan", "漯河市 "], ["crisschan", "chan", "银川市"], ["crisschan", "chan", "云浮市 "], ["crisschan", "chan", "扬州市"], ["crisschan", "chan", "丹东市"], ["crisschan", "chan", "茂名市"]]]

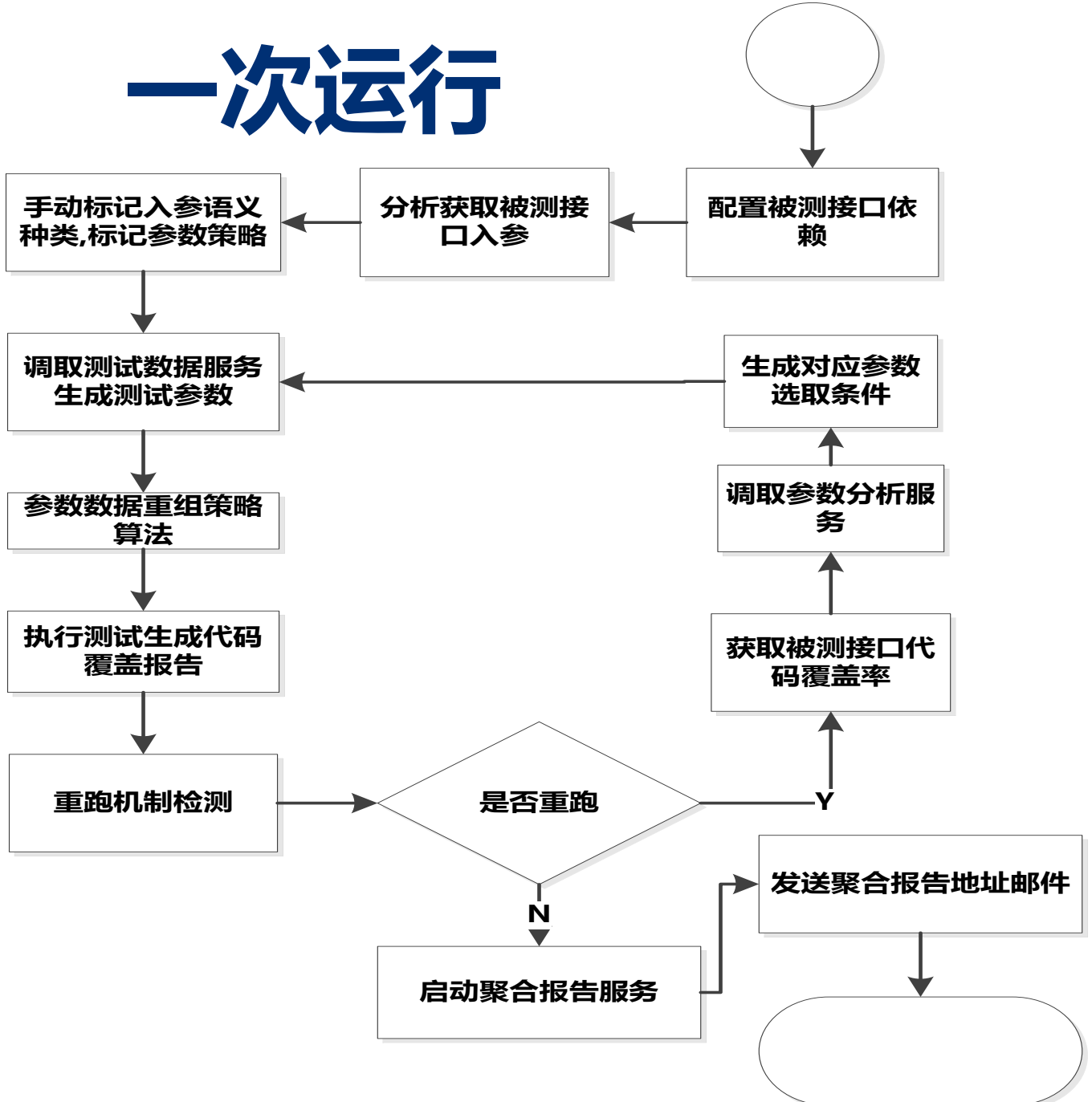
• 无约束查询

- person.name,person.last_name,address.city
- 随机产生一条记录
- [[“person.name”，“person.last_name”，“address.city”], [[“王磊”，“王”，“天津市"]]

自动重跑机制


获取接口测试的代码覆盖报告
读入被测接口的代码代码覆盖率
01
提取的覆盖率小于预设阈值
读入代码覆盖报告中代码覆盖详细内容
02
分析对应未覆盖代码执行条件
调取参数分析服务
03
生产数据契约，调取TDS服务填补对应数据
04
再次执行测试
再次调起自动重跑机制
05

一次运行


配置被测接口依赖
分析获取被测接口入参
手动标记入参语义种类,标记参数策略
调取测试数据服务生成测试参数
参数数据重组策略算法
执行测试生成代码覆盖报告
重跑机制检测
是否重跑
Y
N
获取被测接口代码覆盖率
调取参数分析服务
生成对应参数选取条件
启动聚合报告服务
发送聚合报告地址邮件

引入开源完成第一层

- EvoSuite： 是由Sheffield等大学联合开发的一种开源工具，用于自动生成测试用例集，生成的测试用例均符合Junit的标准，可直接在Junit中运行。得到了Google和Yourkit的支持。
- EvoSuite基于搜索的（最大化覆盖率或异常）生产测试用例。

初见

EvoSuite完成先验测试

自动generate
mvn：test

脚本自动生成

生成测试脚本java
生成参数json

CI系统提供调用api

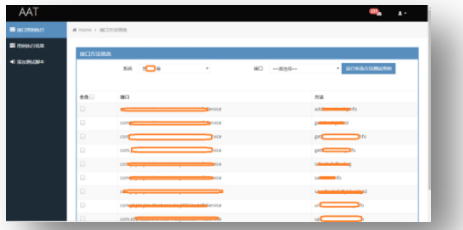
按构建调取
按周期调取
按需调取

测试数据服务

TDS
数据契约

自动重跑机制

是否启动重拍



测试运行

多系统、多接口测试脚本执行
手动执行入口
智能化接口测试入口

THANK YOU!