TiD 2017
质量竞争力大会

测试基础架构最佳实践

茹炳晟（Robin Ru）
ebay中国研发中心

软件研发
SOFTWARE DEVELOPMENT

茹炳晟 （Robin Ru）

产地：上海

主要工作经历：

- ebay 中国研发中心 -- 测试基础架构技术主管
- Hewlett-Packard 惠普软件(中国)研发中心 -- 测试架构师、资深测试专家
- Alcatel-Lucent阿尔卡特朗讯(上海)研发中心 -- 测试技术主管
- Cisco 思科(中国)研发中心 -- 资深测试开发工程师

兴趣爱好：

- 户外运动爱好者
- 全球自由行爱好者
- 高级开放水域潜水员 + 高氧空气潜水员

联系方式：

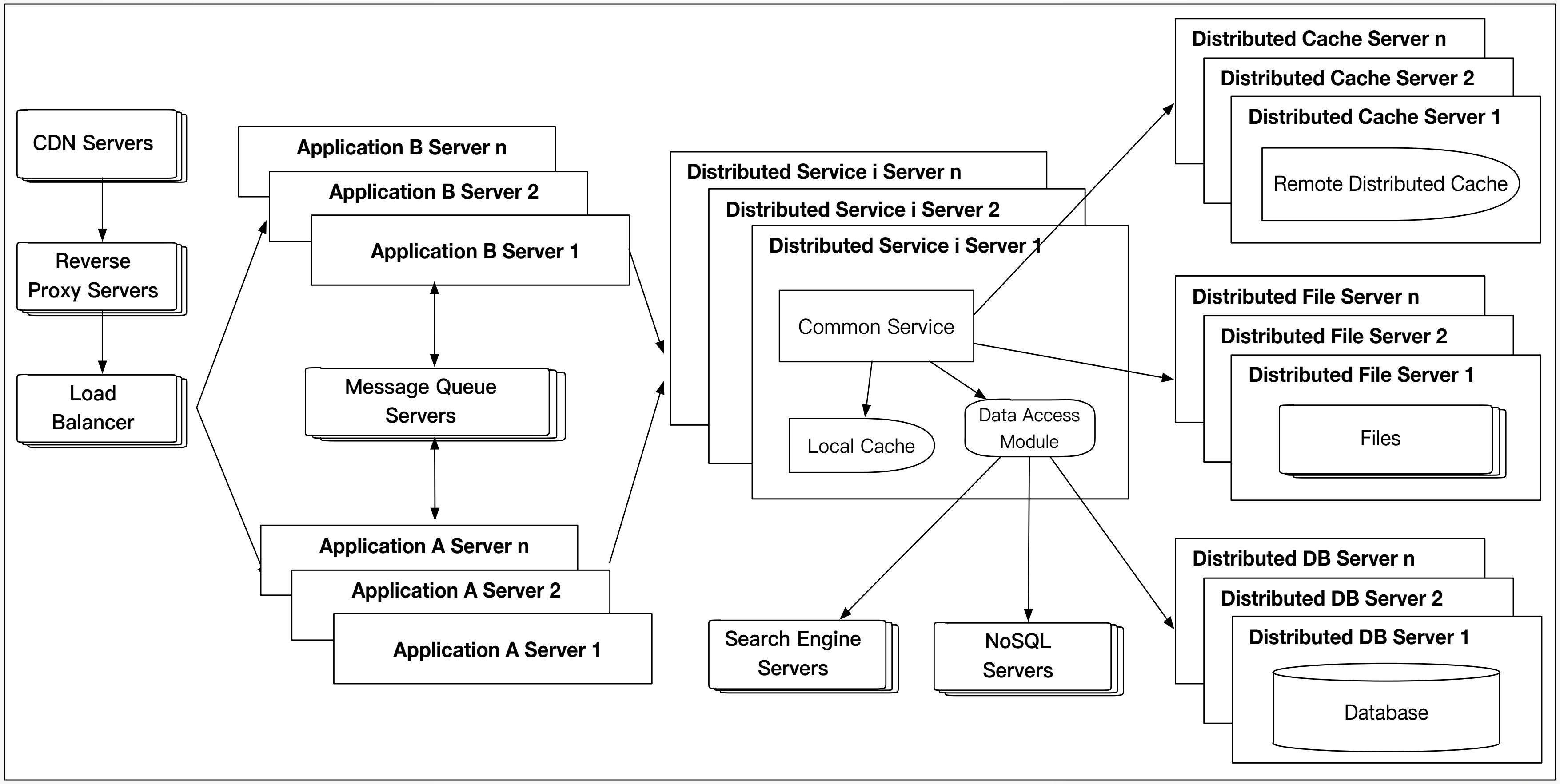
- biru@ebay.com or 微信

引言

- ❑3小时 → 讲什么 (What)
- ❑各种测试架构和测试框架 → 为什么（Why）
- ❑知其然知其所以然 → 怎么讲 （How）
- ❑站在前人的肩膀上 → 收获 （Best Practice）

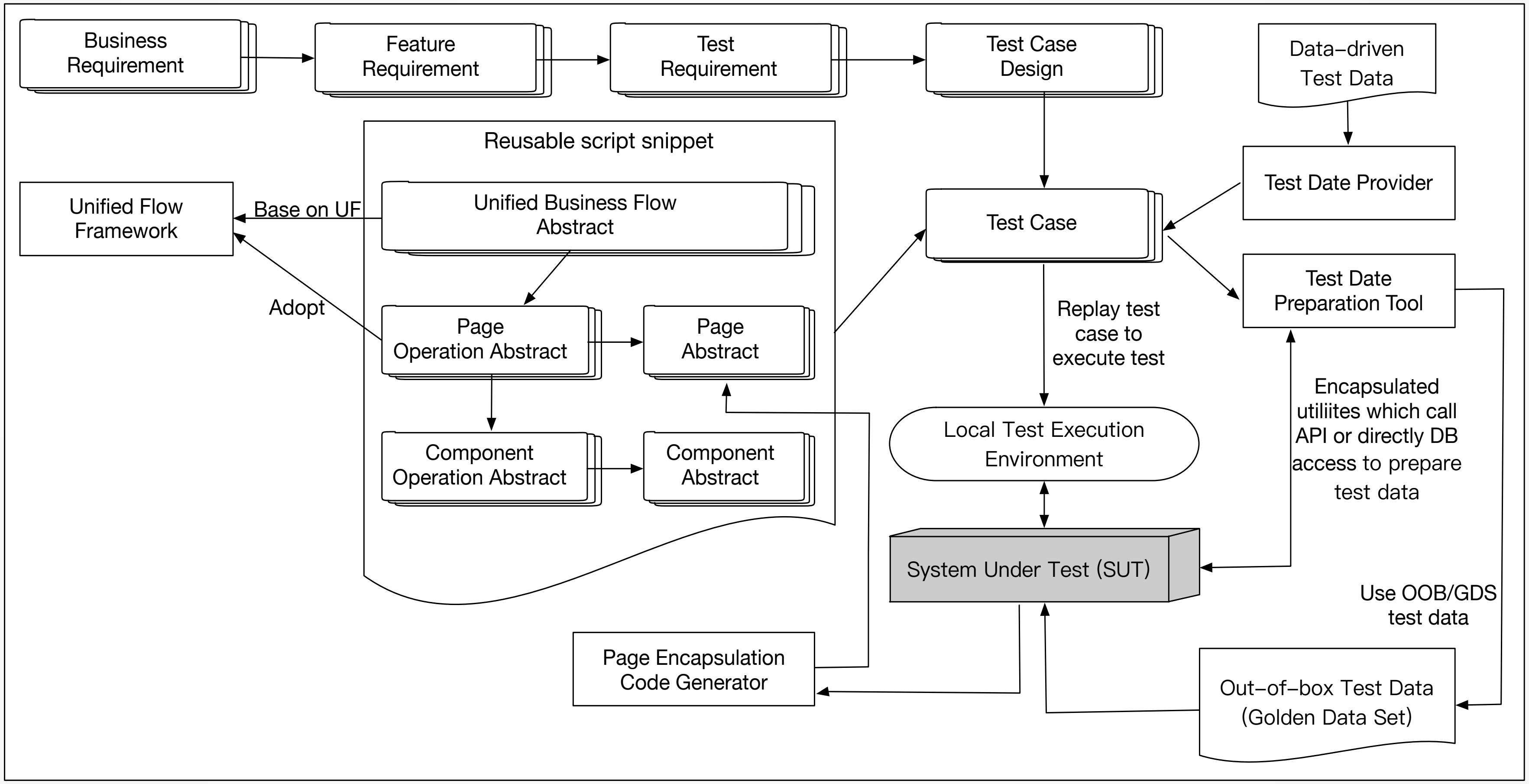
TiD2017

大型网站分布式架构



TiD2017

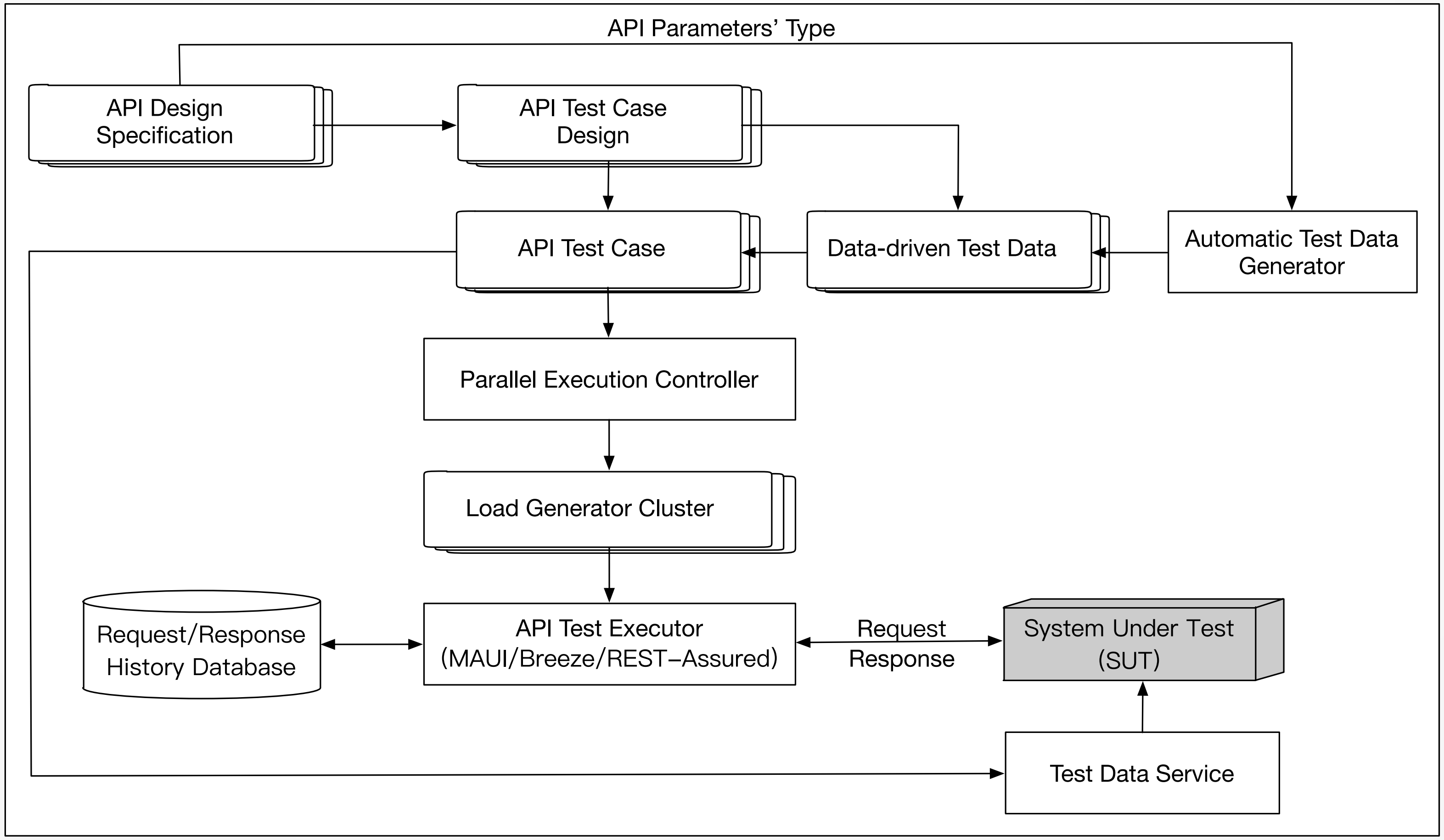
GUI Automation Test 架构



Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Data–driven Test Data
Reusable script snippet
Unified Flow Framework
Base on UF
Unified Business Flow Abstract
Test Date Provider
Test Case
Adopt
Page Operation Abstract
Page Abstract
Replay test case to execute test
Test Date Preparation Tool
Component Operation Abstract
Component Abstract
Local Test Execution Environment
Encapsulated utiliites which call API or directly DB access to prepare test data
System Under Test (SUT)
Use OOB/GDS test data
Page Encapsulation Code Generator
Out–of–box Test Data (Golden Data Set)

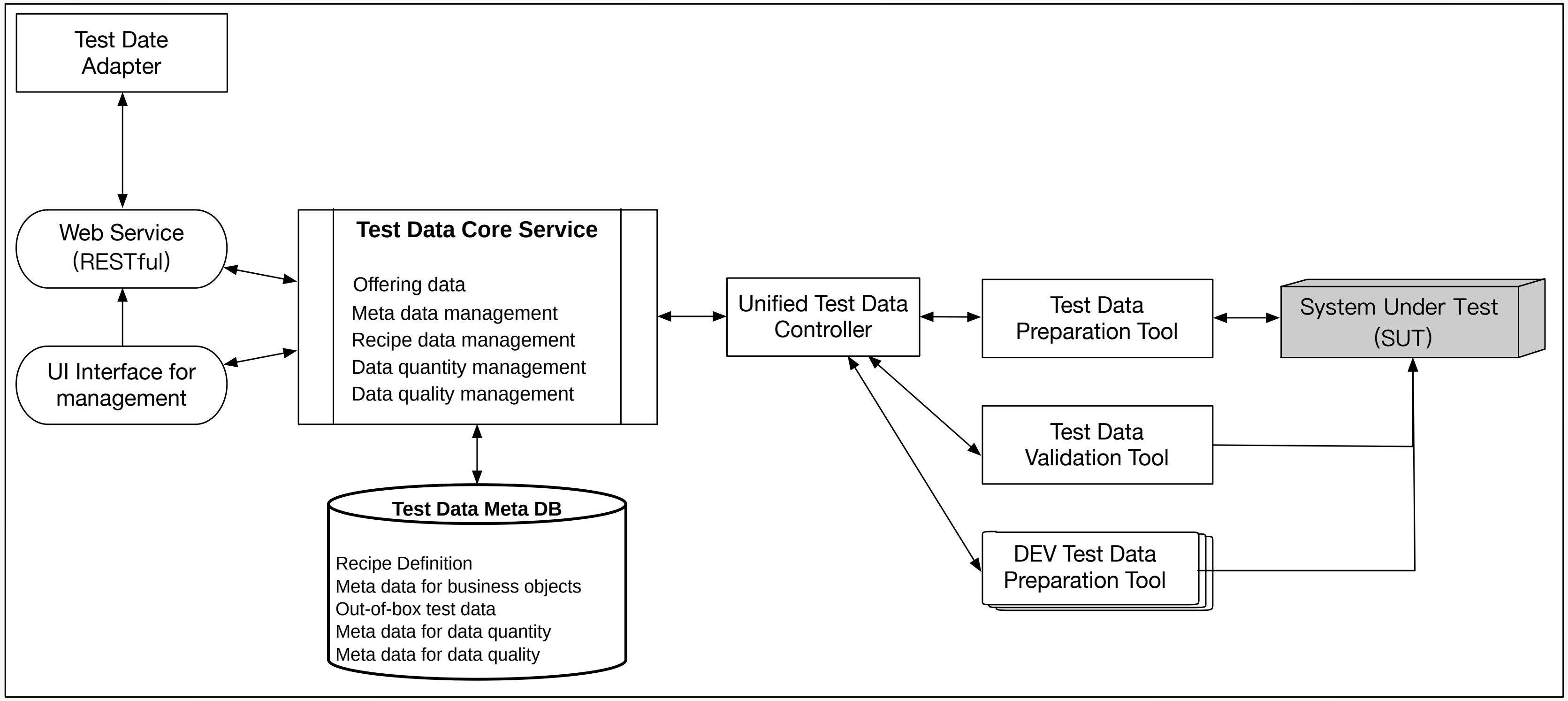
TiD2017

API Automation Test 架构



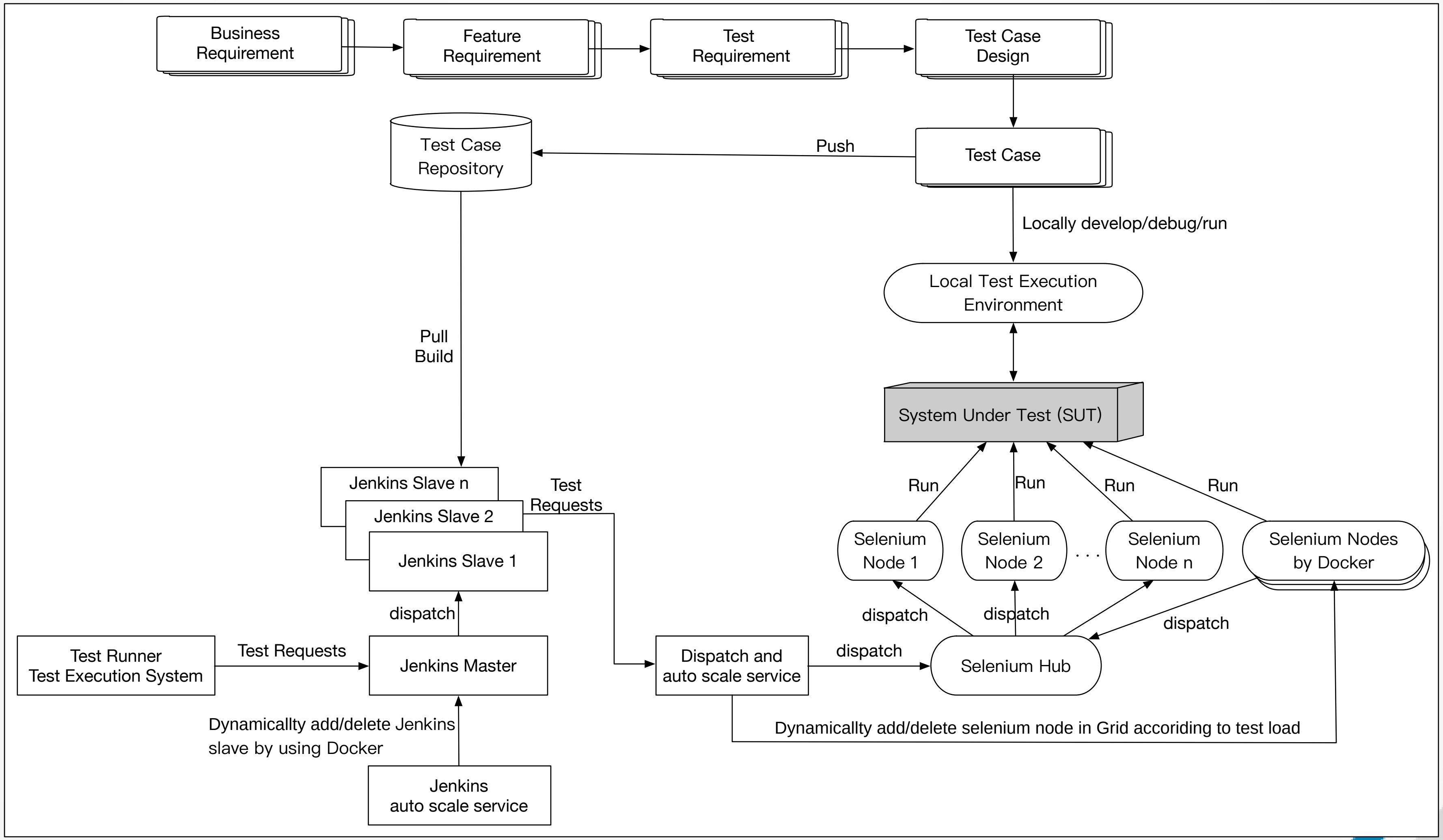
TiD2017

Test Data Platform 架构

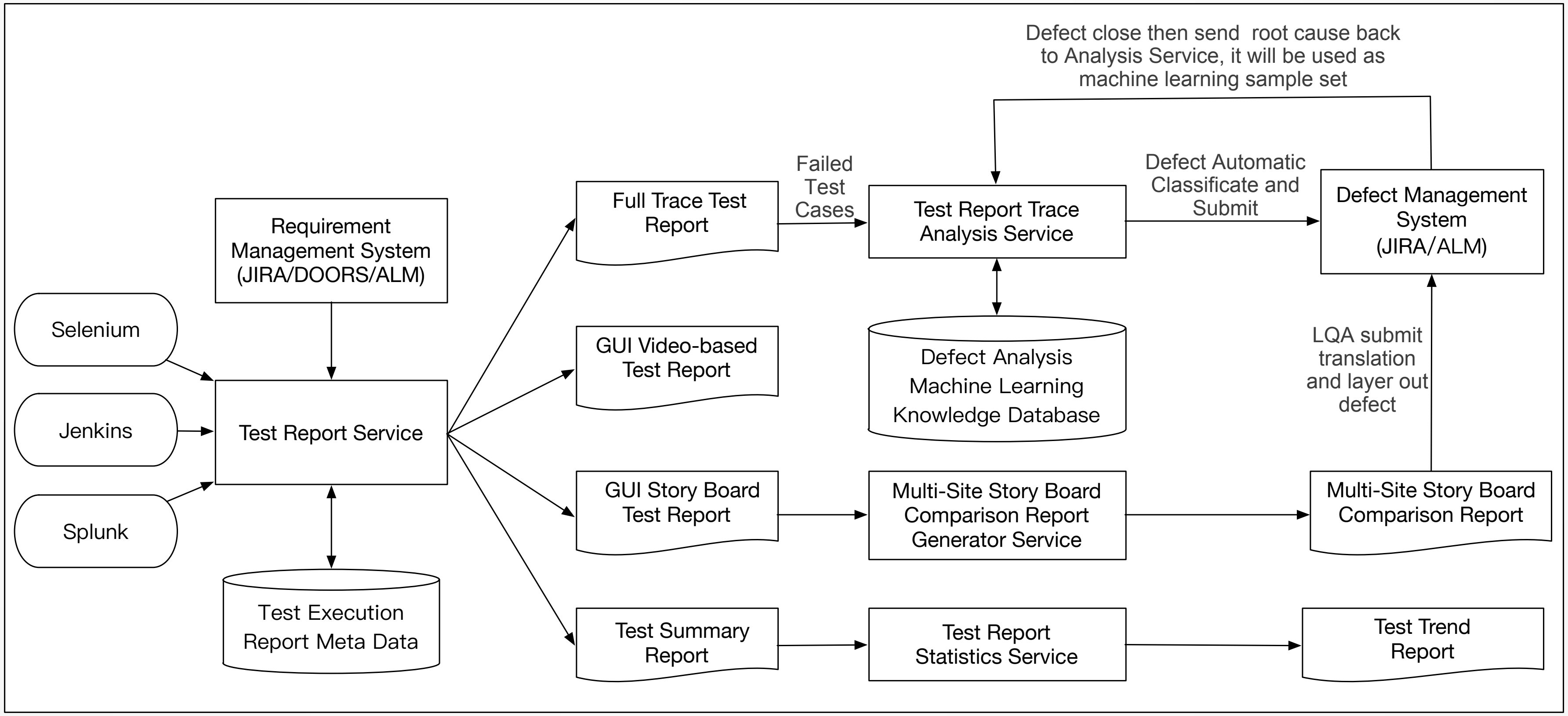


TiD2017

Test Execution Environment 架构



Test Report Platform 架构


Defect close then send root cause back to Analysis Service, it will be used as machine learning sample set
Requirement Management System (JIRA/DOORS/ALM)
Selenium
Jenkins
Splunk
Test Report Service
Test Execution Report Meta Data
Full Trace Test Report
GUI Video-based Test Report
GUI Story Board Test Report
Test Summary Report
Failed Test Cases
Test Report Trace Analysis Service
Defect Analysis Machine Learning Knowledge Database
Multi-Site Story Board Comparison Report Generator Service
Test Report Statistics Service
Defect Automatic Classificate and Submit
Defect Management System (JIRA/ALM)
LQA submit translation and layer out defect
Multi-Site Story Board Comparison Report
Test Trend Report

TiD2017

Agenda

- ❑ 测试基础架构的范畴
- ❑ 从大型网站技术架构的演变谈起
- ❑ 测试基础架构的演变
  - ➢ GUI Automation Test Framework的演变
  - ➢ Test Data Platform的演变
  - ➢ API Automation Test Framework的演变
  - ➢ Test Execution Environment的演变
  - ➢ Test Execution and Management Platform的演变
  - ➢ Test Report Platform的演变

Agenda

- ❑ 代码级单元测试的最佳实践
  - ➢ 测试架构师在代码级单元测试扮演什么样的角色
  - ➢ 代码的基本特征
  - ➢ 代码缺陷产生的原因
  - ➢ 代码错误的分类
  - ➢ 代码级测试方法的分类
  - ➢ 完备测试用例的设计
  - ➢ 代码测试覆盖率的衡量

TiD2017

Agenda

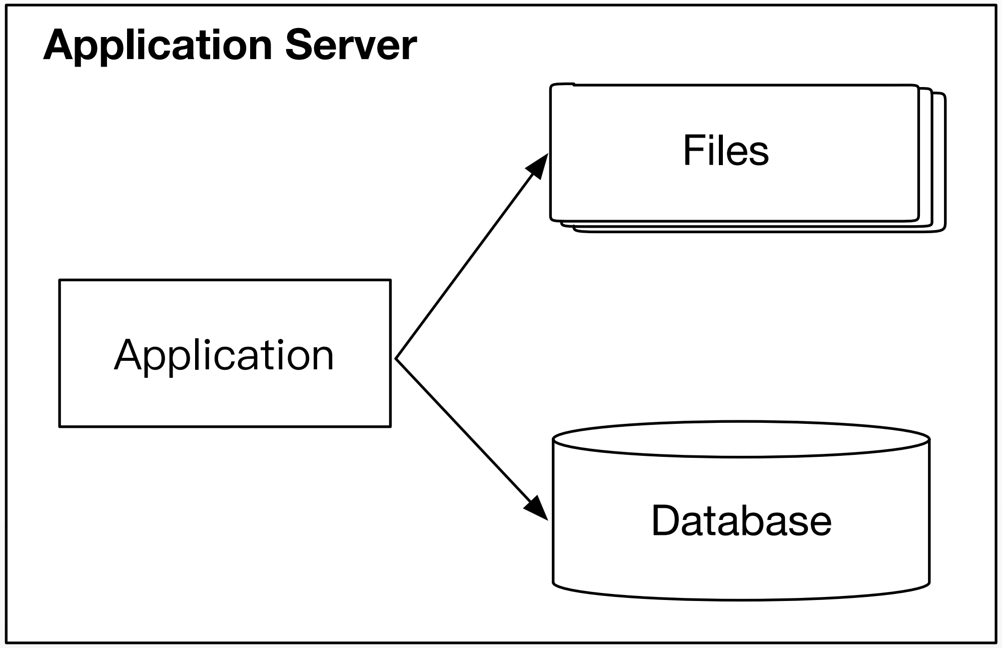
- ❑ 全球化Site自动化测试的最佳实践
  - ➢ What‘s Global Site Test ?
  - ➢ Global Site Test Challenge
  - ➢ Global Test Capability Brief Introduction and Key Value
  - ➢ Global Test Capability Design Philosophy
  - ➢ Go deeper into Global Test Capability
    - ❖ Global site test base utilities
    - ❖ Global site test data utilities
    - ❖ Global Configuration Repository
    - ❖ Unified Flow Framework
    - ❖ Multi-Site Story Board Test Report
    - ❖ Global Site Test Overall Architecture

测试基础架构的范畴

- ❑ **Automation Test Framework**
  - ➢ **Code Level Unit Test**
  - ➢ **Code Level Integration Test**
  - ➢ **GUI Level**
  - ➢ **API Level**
- ❑ **Test Execution Environment**
  - ➢ **Jenkins**
  - ➢ **Selenium Gird**
  - ➢ **Other**
- ❑ **Test Case Execution and Management Platform**
- ❑ **Test Data Platform**
- ❑ **DevOps Integration Interface**

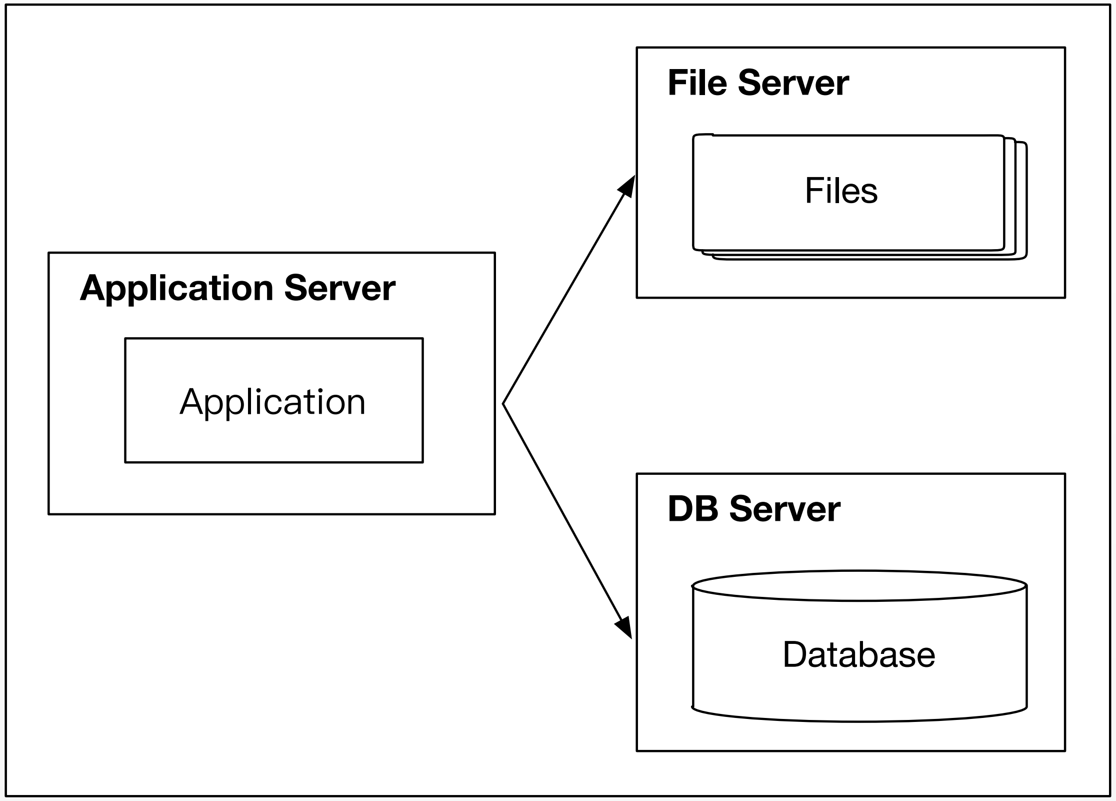
从大型网站技术架构的演变谈起

网站架构Hello World



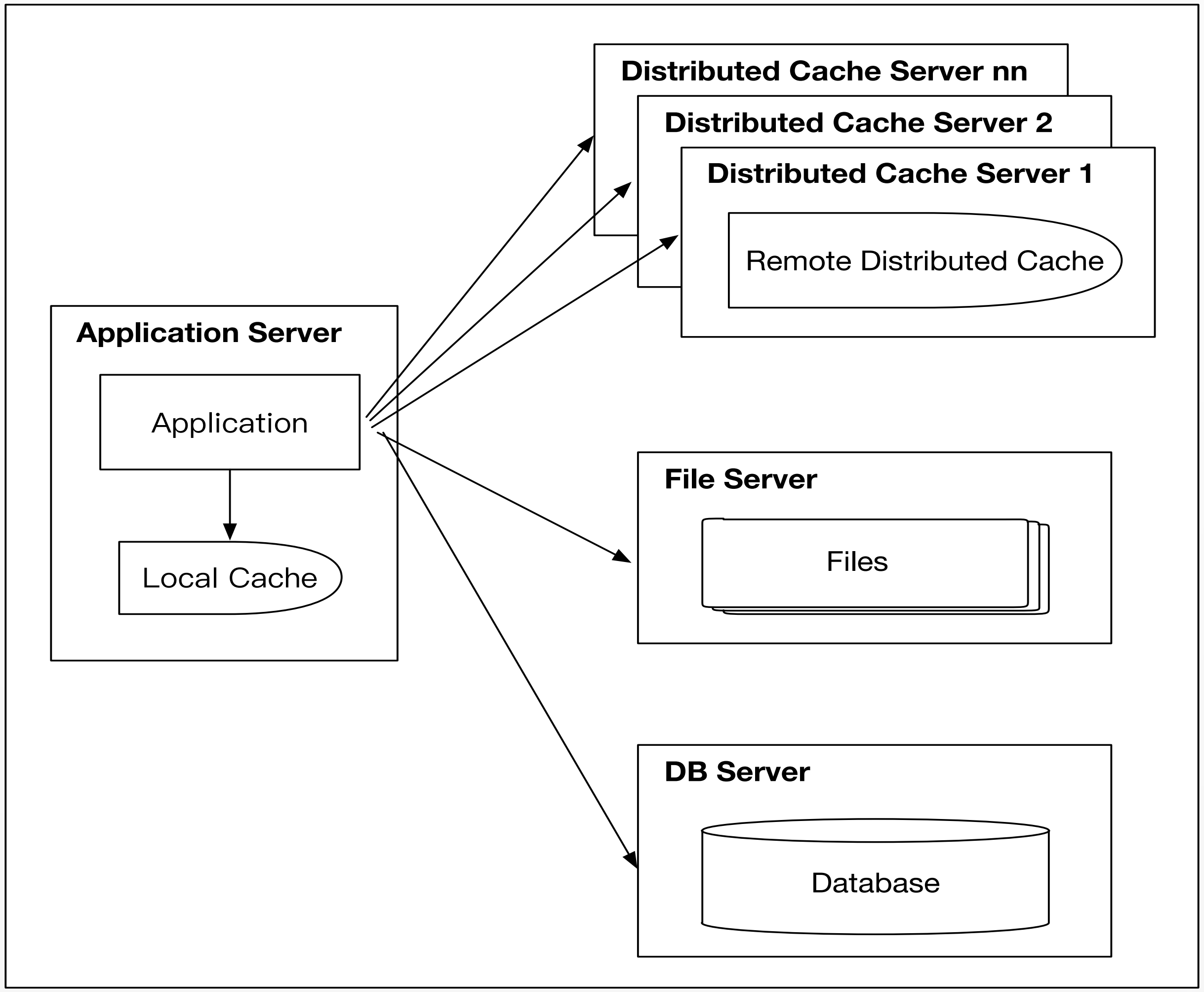
从大型网站技术架构的演变谈起

应用服务和数据服务分离



从大型网站技术架构的演变谈起

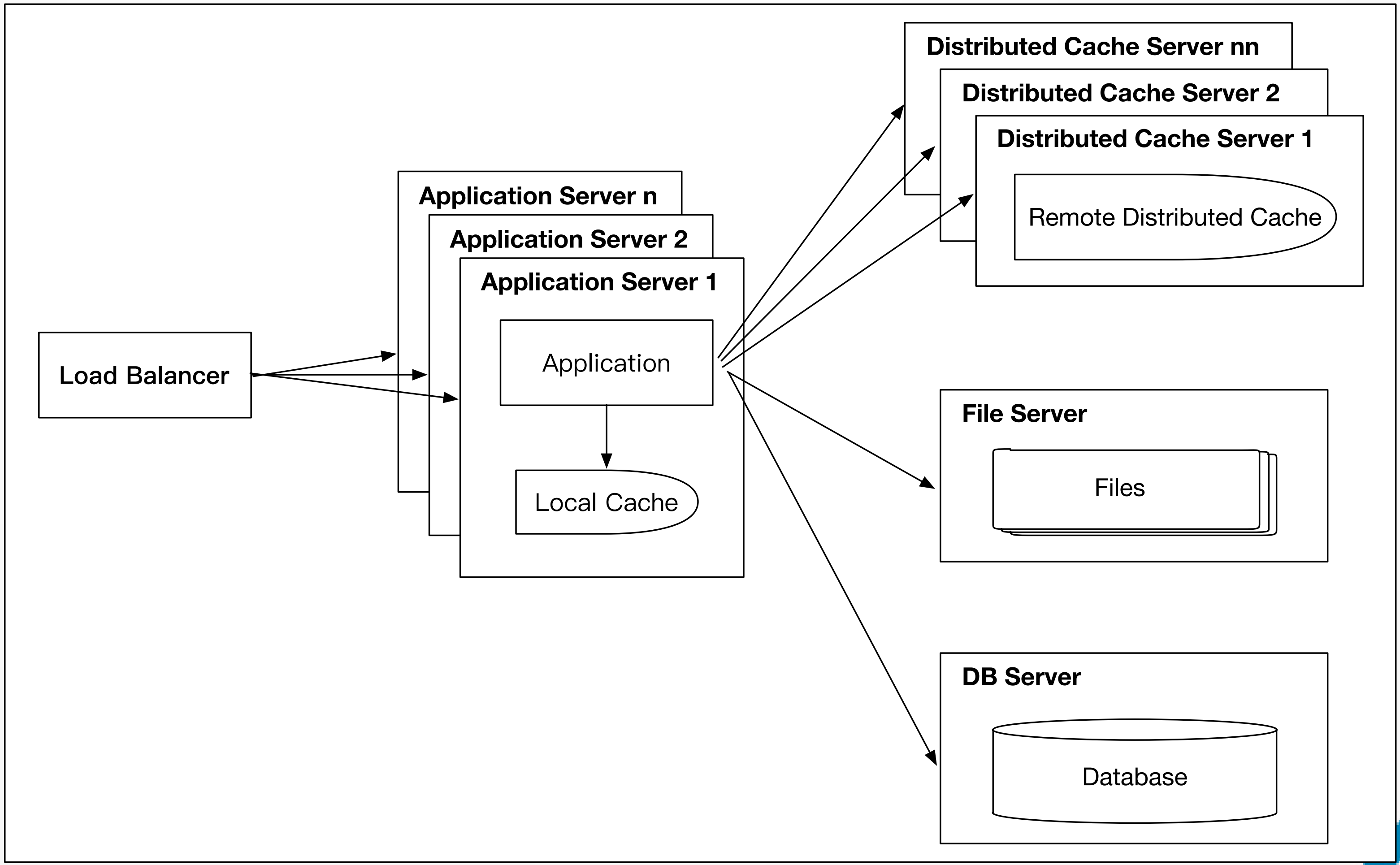
使用Cache改善网站性能



TiD2017

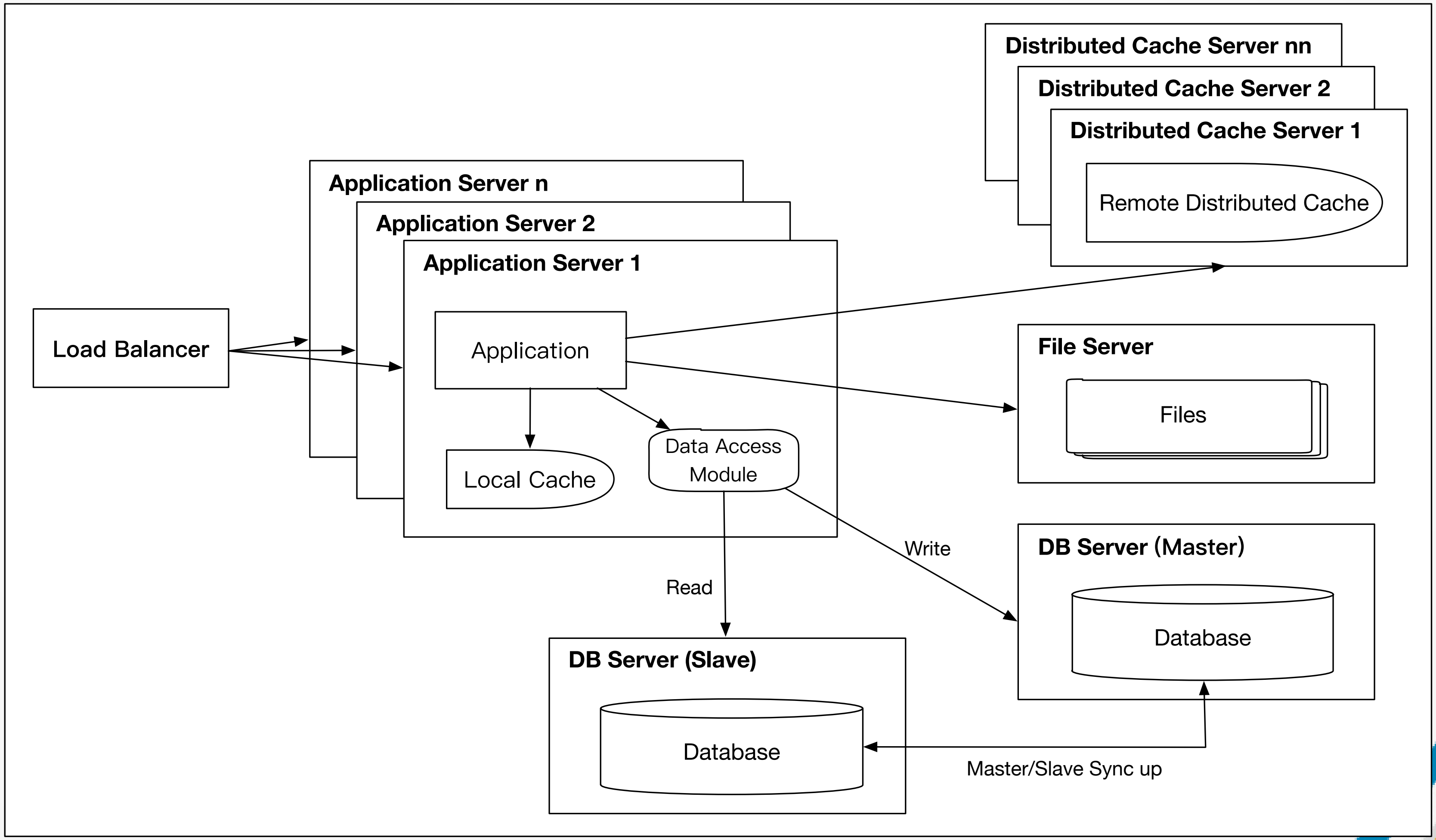
从大型网站技术架构的演变谈起

使用Application Server Cluster改善网站并发处理能力



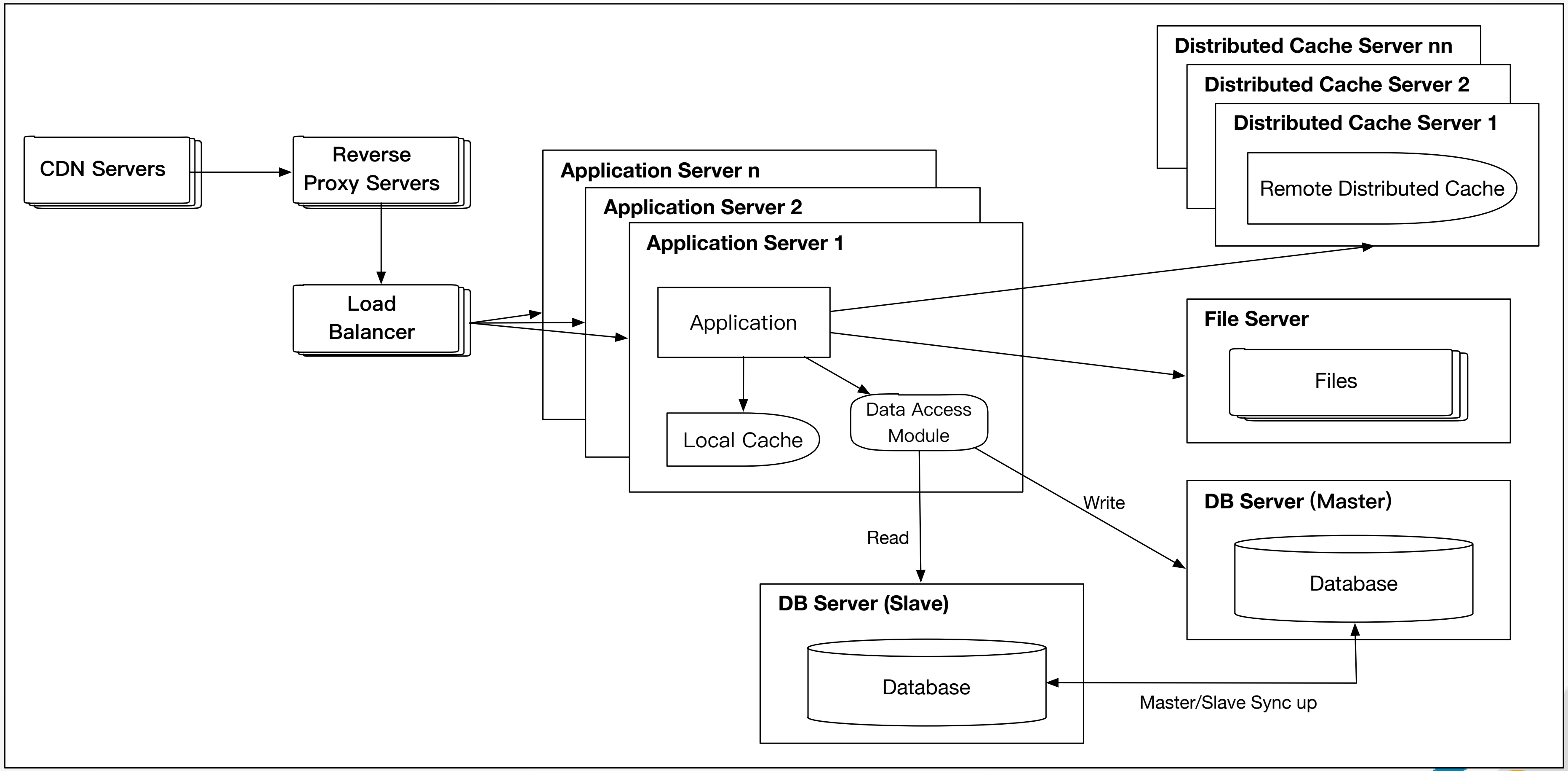
从大型网站技术架构的演变谈起

数据库读写分离



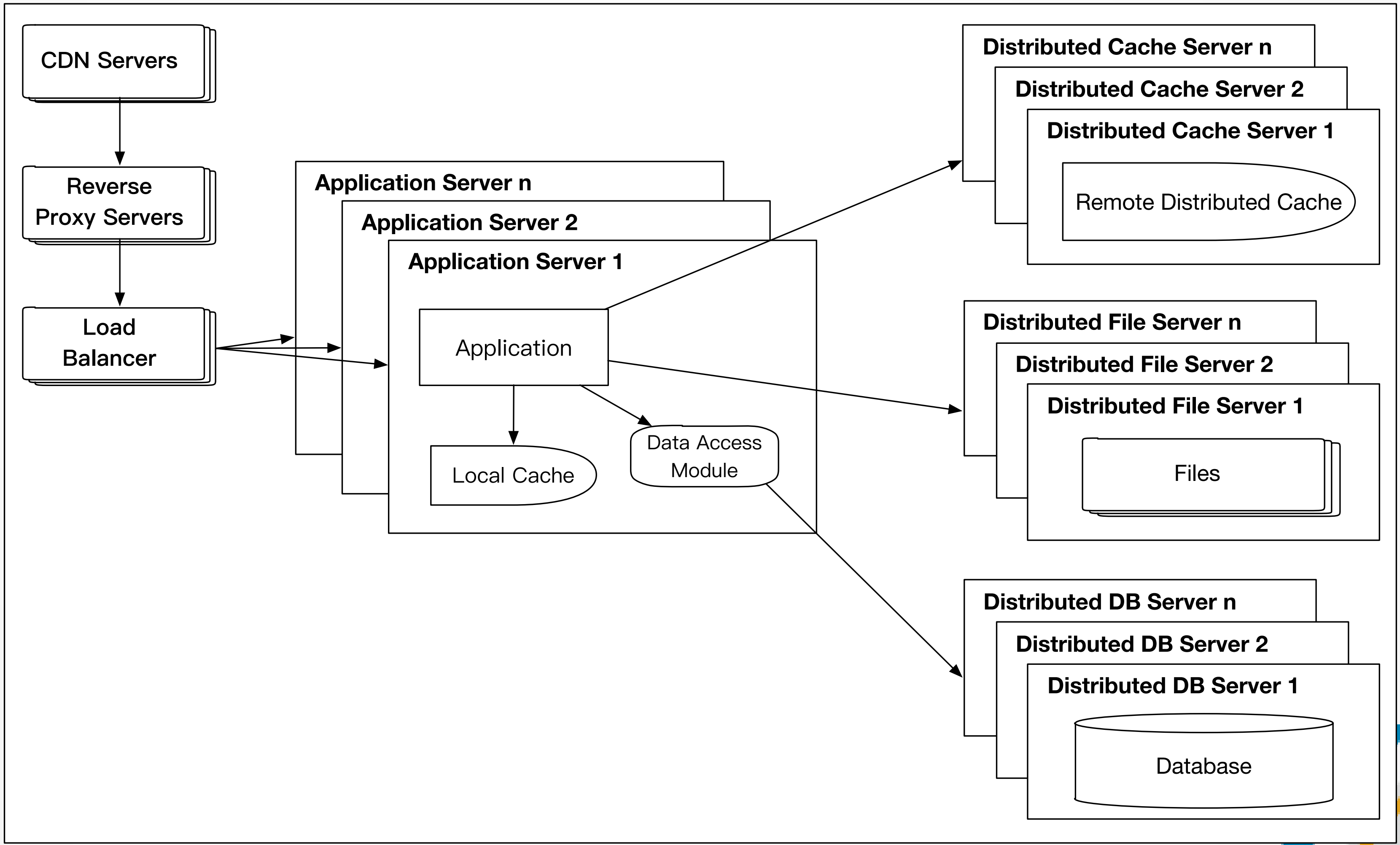
从大型网站技术架构的演变谈起

使用**Reverse Proxy**和**CDN**加速网站响应



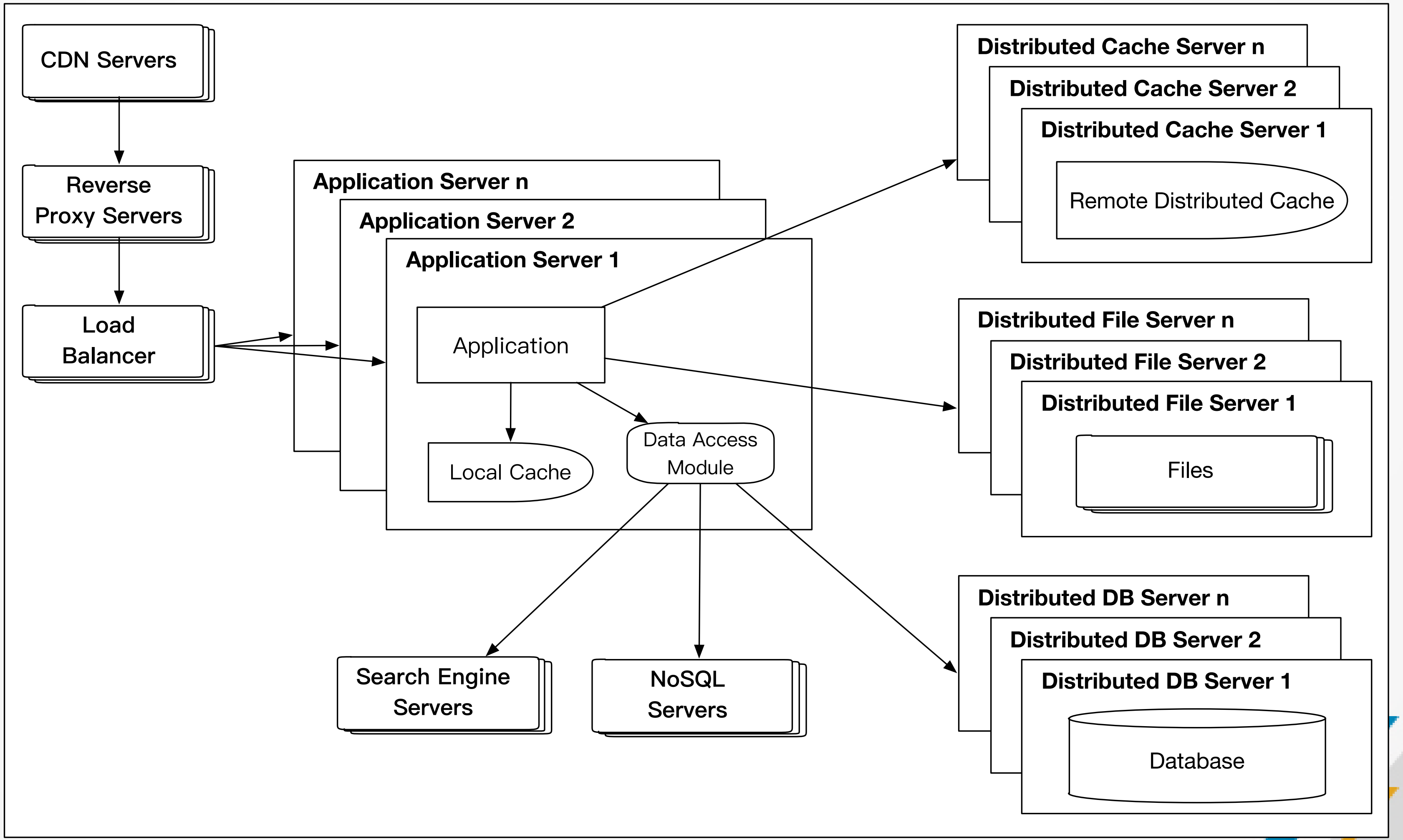
从大型网站技术架构的演变谈起

使用Distributed File Server和Distributed DB Server



从大型网站技术架构的演变谈起

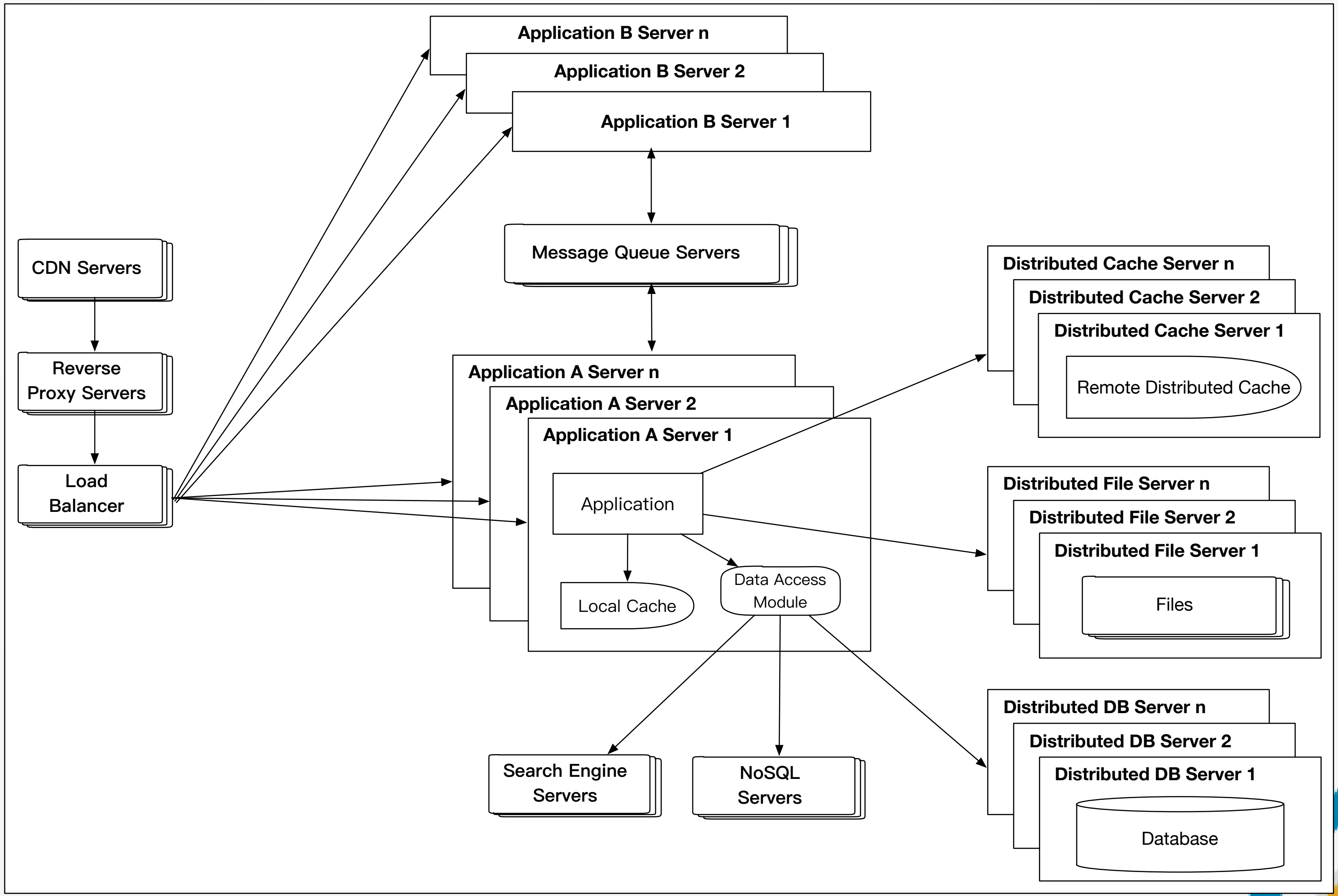
使用NoSQL和Search Engine



TiD2017

从大型网站技术架构的演变谈起

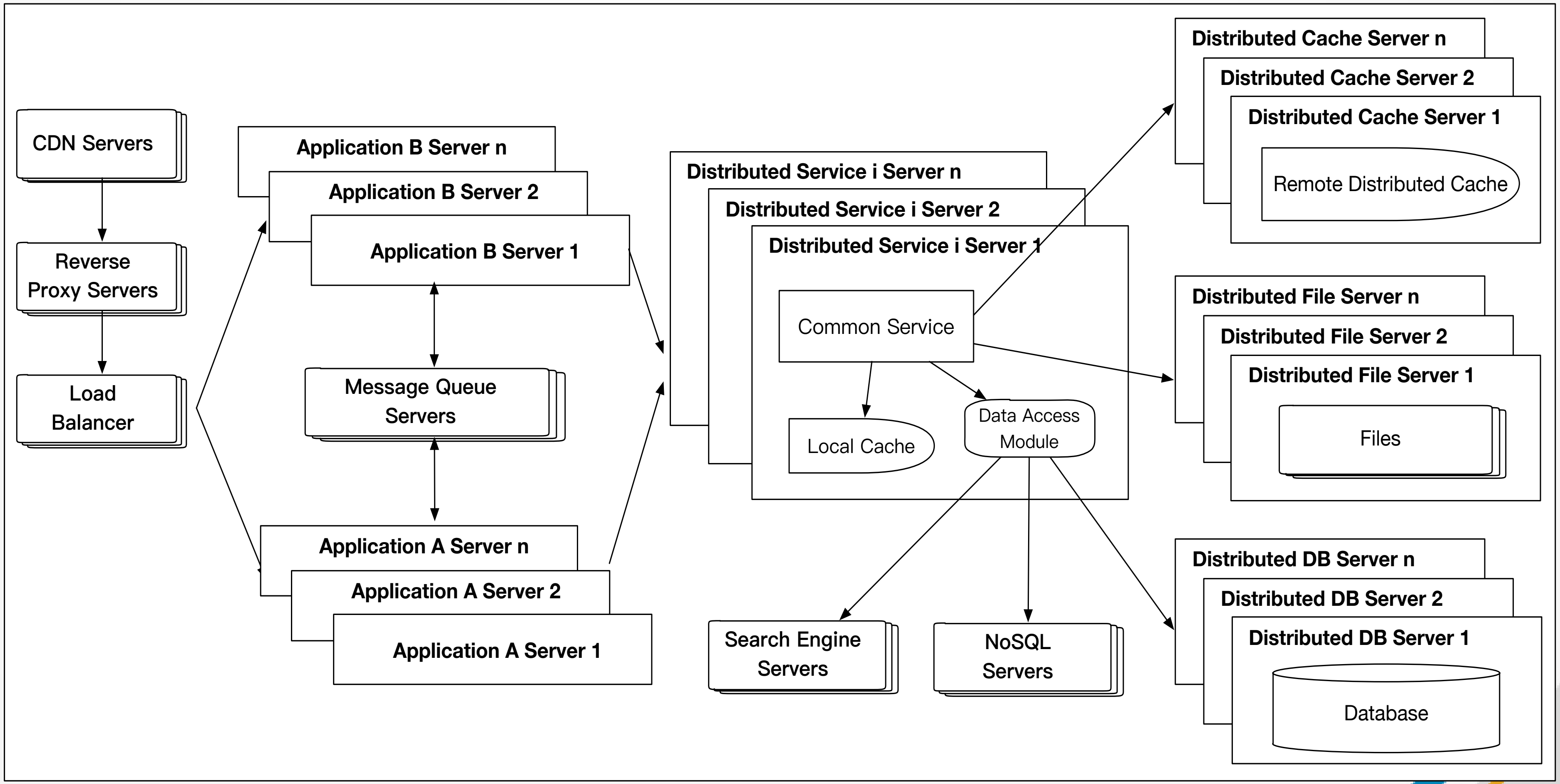
业务拆分



TiD2017

从大型网站技术架构的演变谈起

基于分布式服务架构



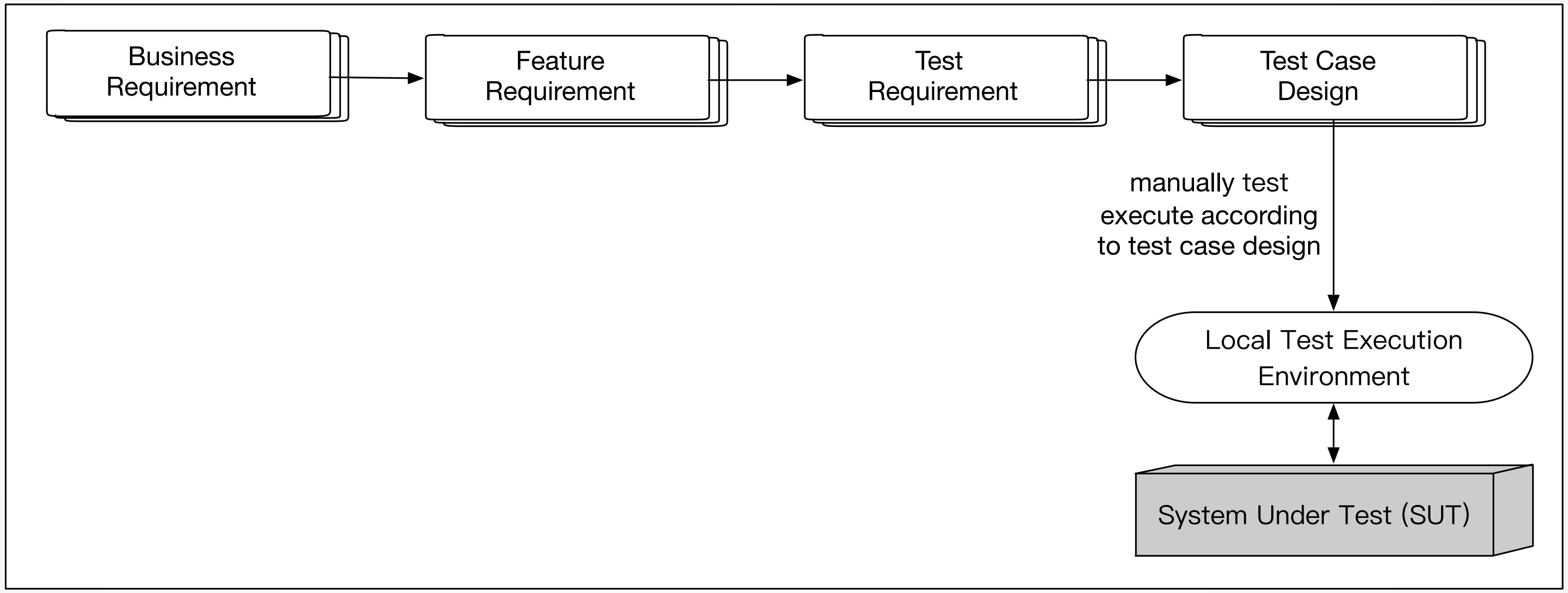
TiD2017

测试架构的演变

- ❑ GUI Automation Test Framework的演变
- ❑ Test Data Platform的演变
- ❑ API Automation Test Framework的演变
- ❑ Test Execution Environment的演变
- ❑ Test Execution and Management Platform的演变
- ❑ Test Report Platform的演变
- ❑ DevOps Integration Interface的演变

测试架构的演变 - GUI Automation Test Framework

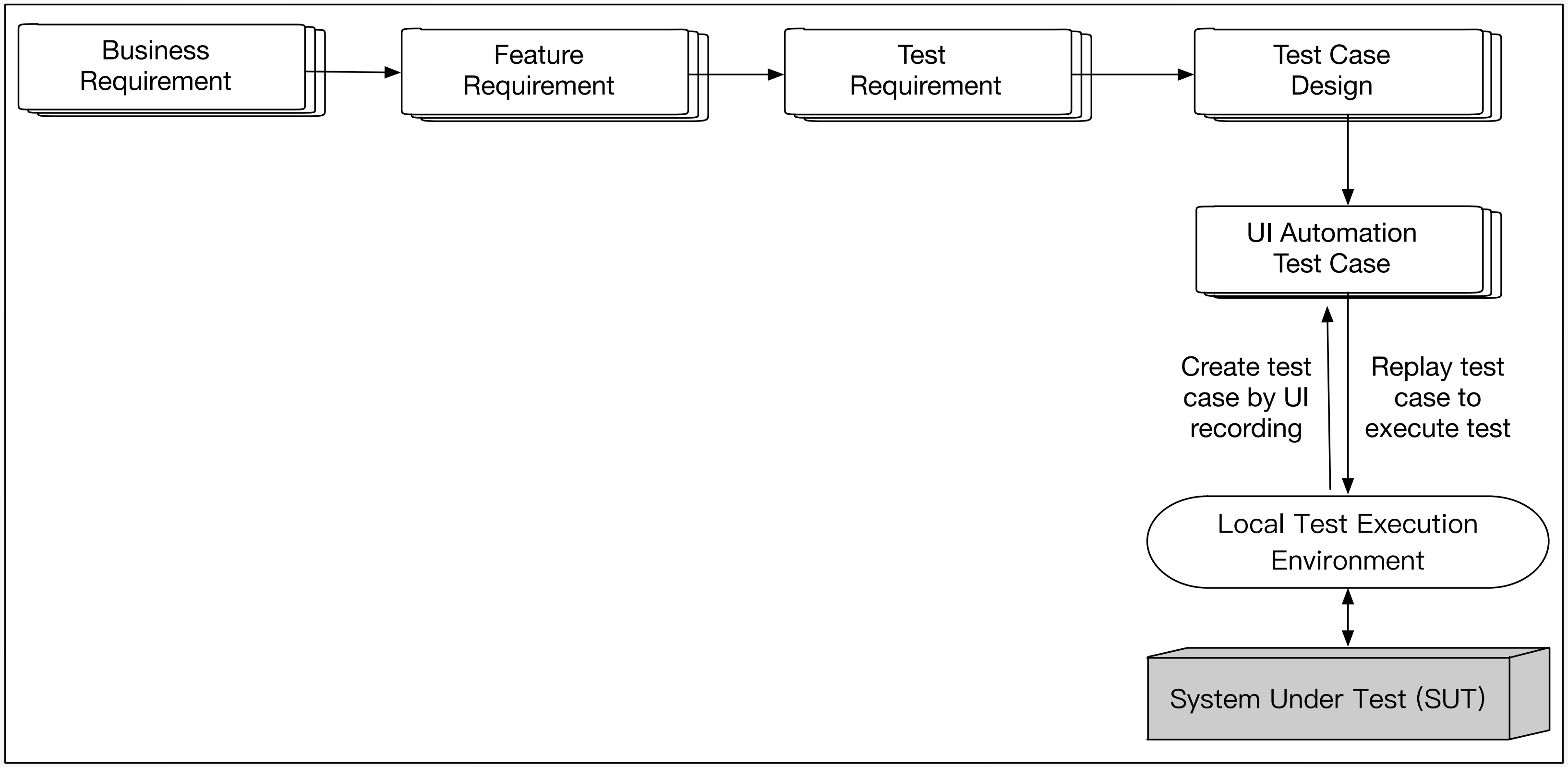
最原始的GUI测试



TiD2017

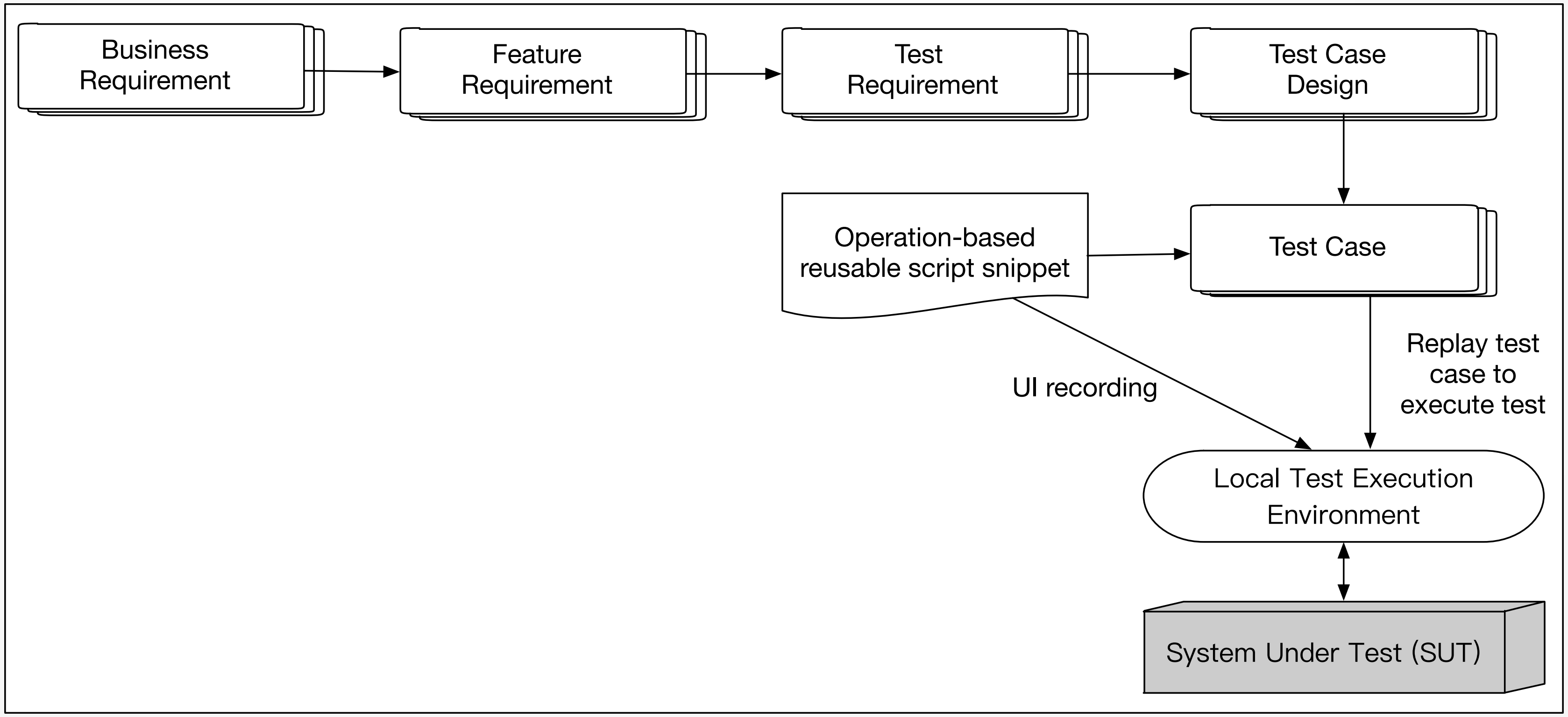
测试架构的演变 - GUI Automation Test Framework

基于录制回放的GUI自动化测试



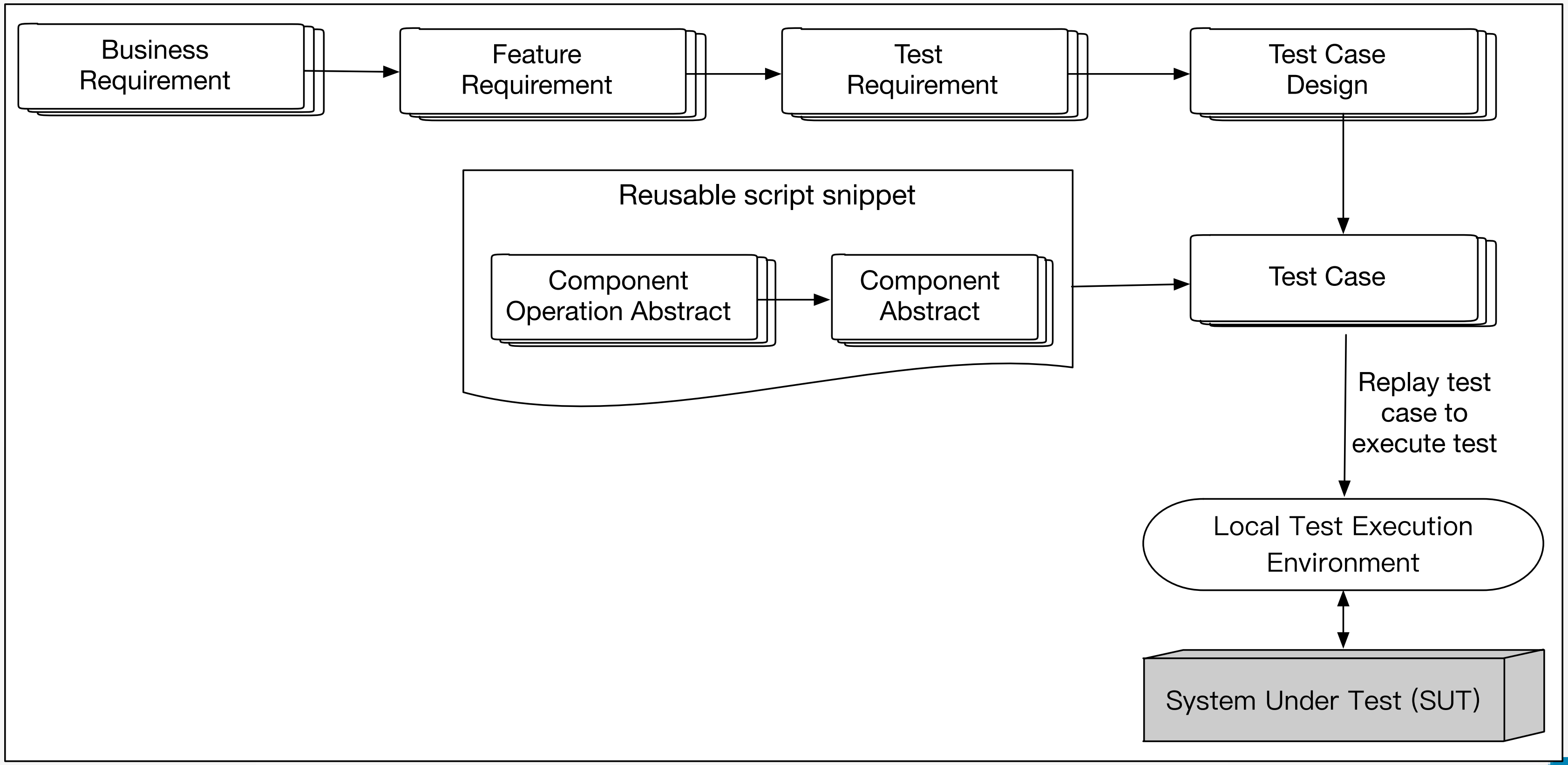
测试架构的演变 - GUI Automation Test Framework

基于可重用测试代码片段构成GUI测试用例



测试架构的演变 - GUI Automation Test Framework

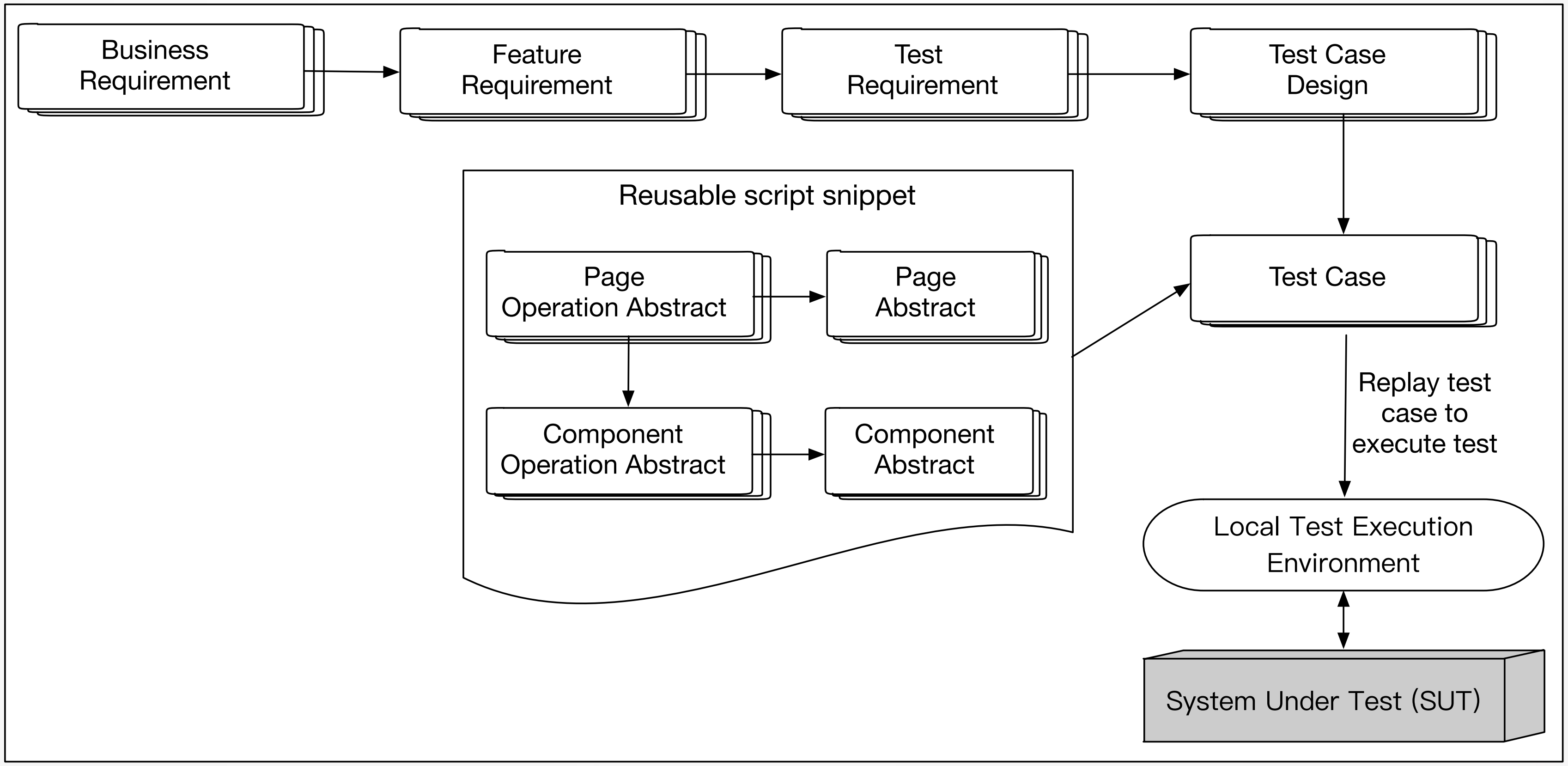
基于Component Abstract构成GUI测试用例



TiD2017

测试架构的演变 - GUI Automation Test Framework

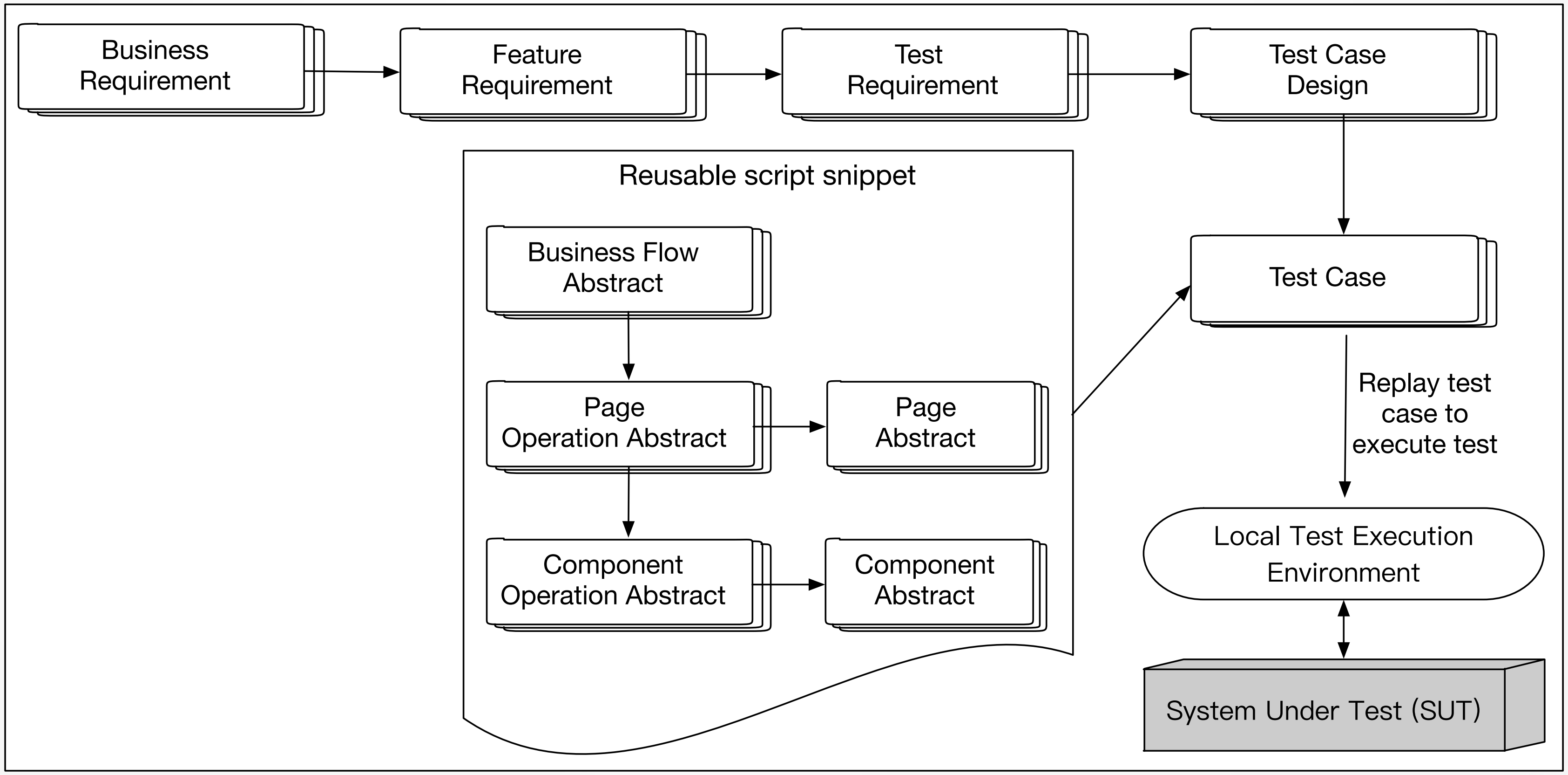
基于Page Abstract构成GUI测试用例



TiD2017

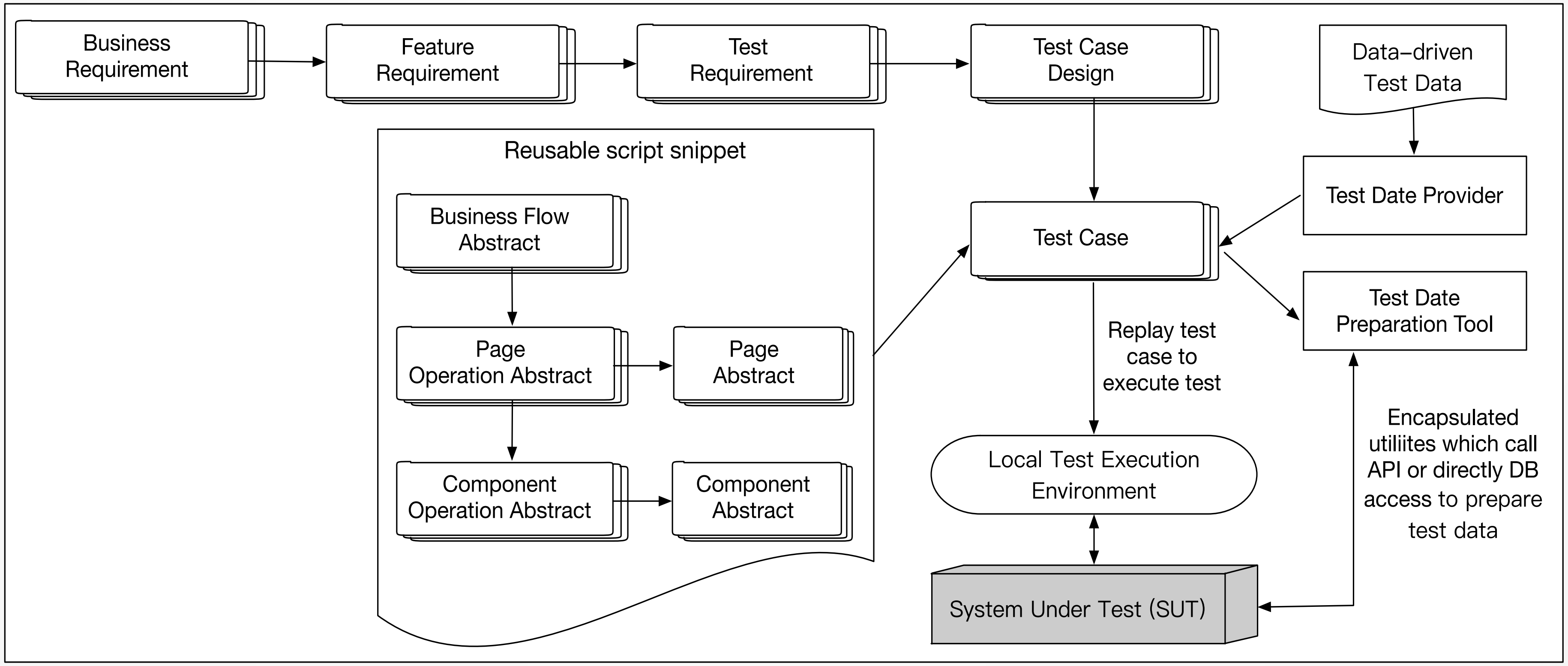
测试架构的演变 - GUI Automation Test Framework

基于Business Flow Abstract构成GUI测试用例



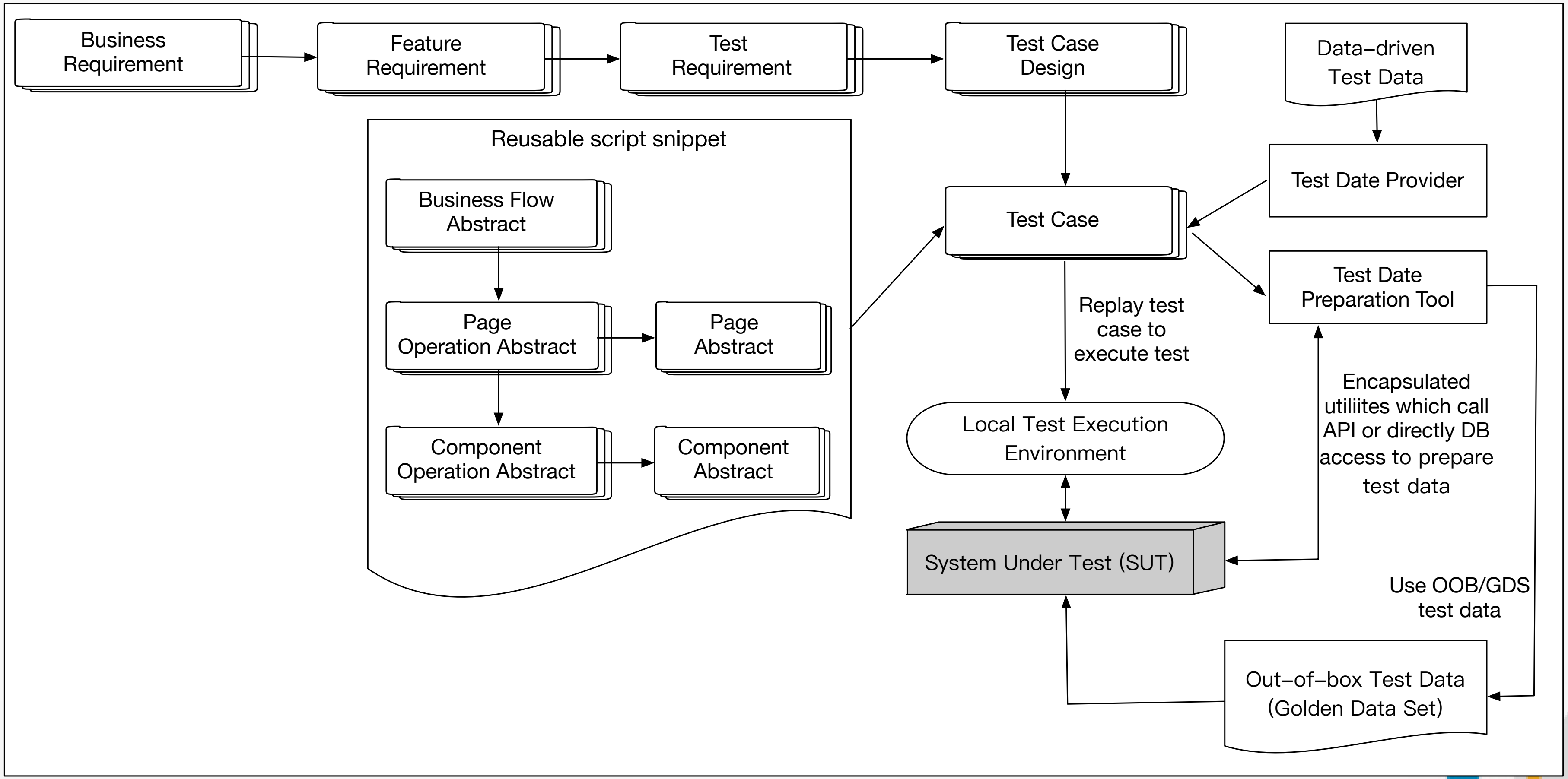
测试架构的演变 - GUI Automation Test Framework

测试数据 – 数据驱动的测试 + 测试数据准备



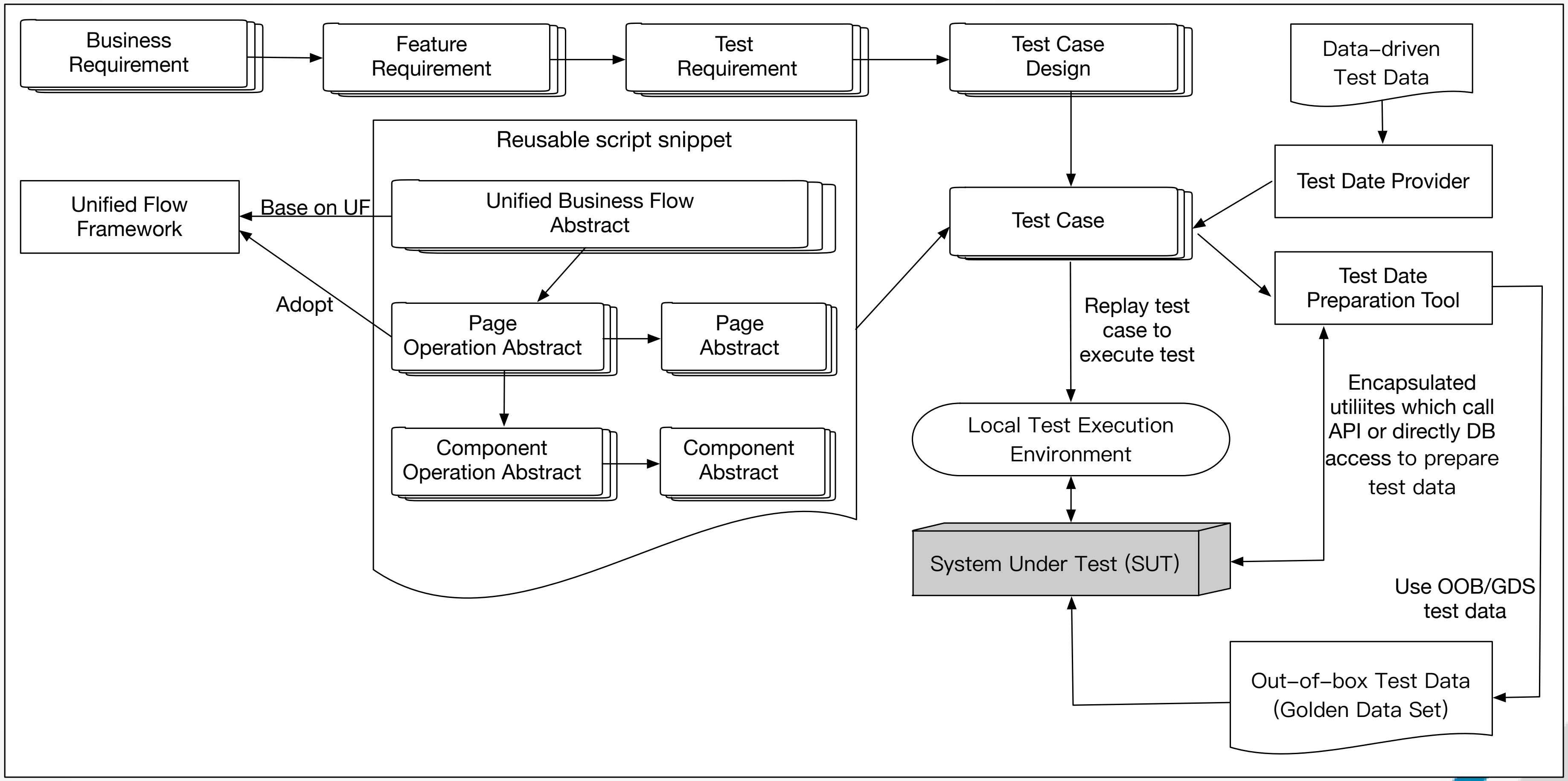
测试架构的演变 - GUI Automation Test Framework

测试数据 – 使用Out-of-box Test Data / Golden Data Set



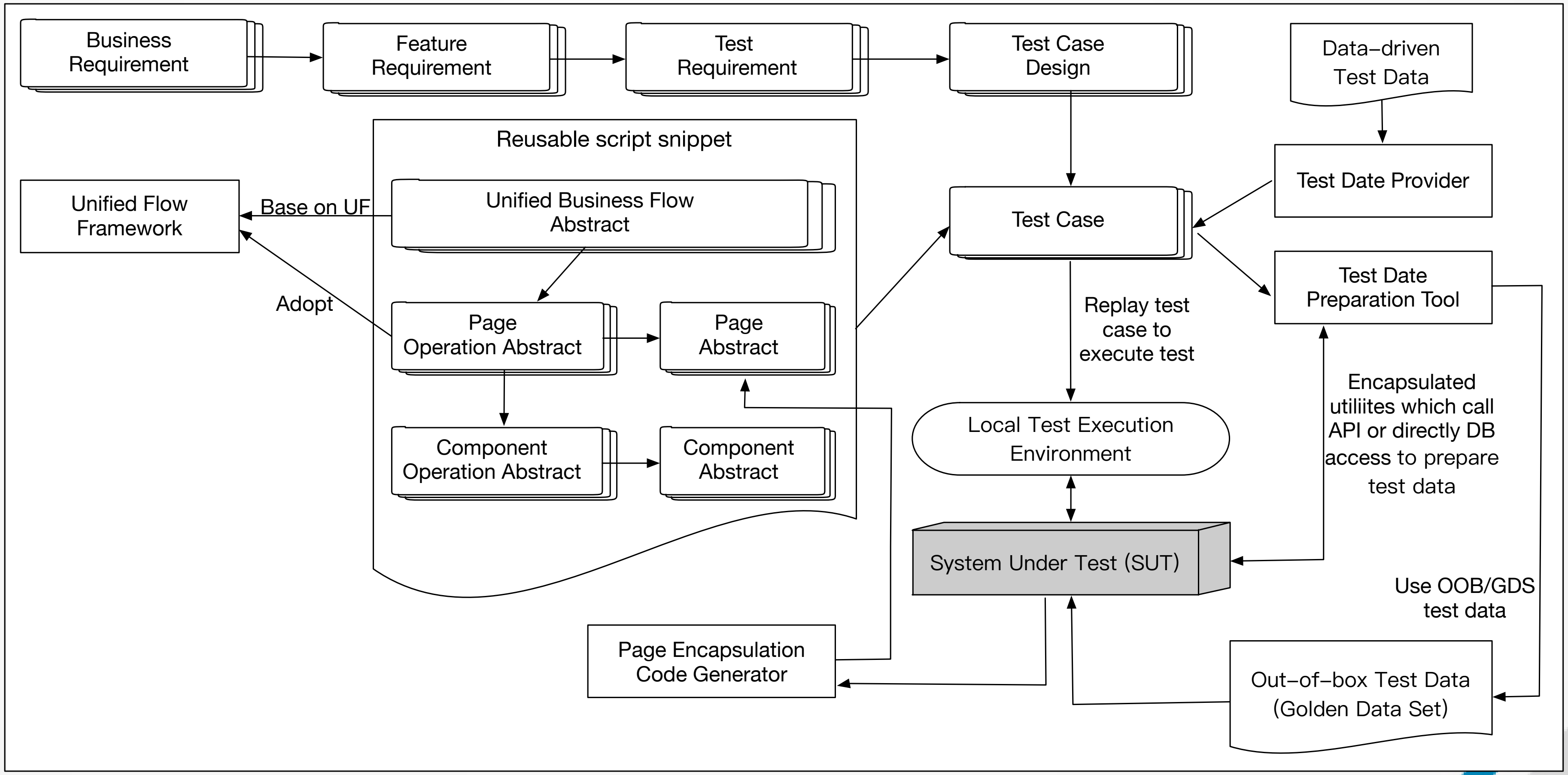
测试架构的演变 - GUI Automation Test Framework

基于Unified Flow Framework实现Flow Branch控制



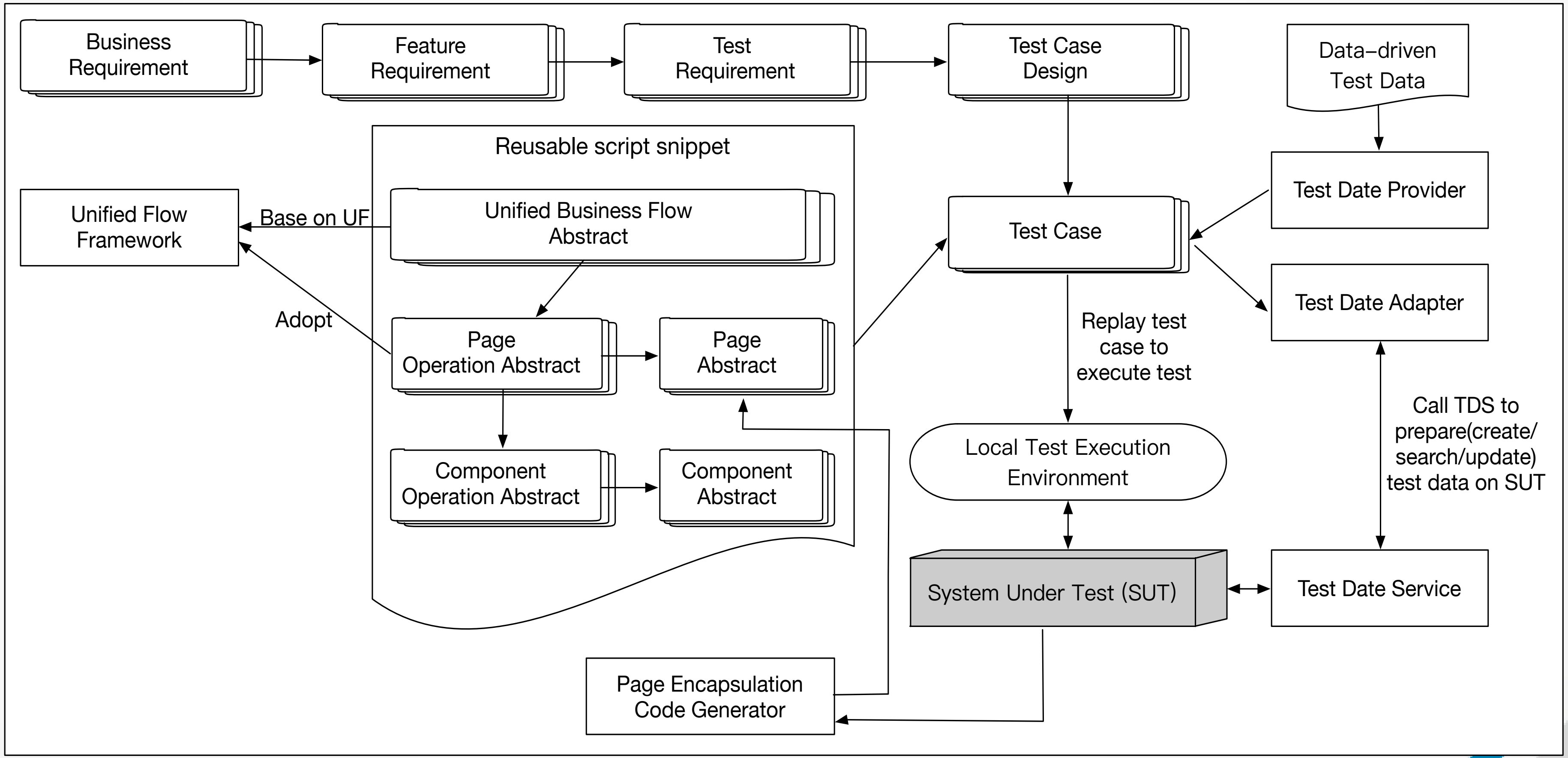
测试架构的演变 - GUI Automation Test Framework

基于Page Encapsulation Code Generator提高Page Abstract的效率


Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Data–driven Test Data
Reusable script snippet
Unified Flow Framework
Base on UF
Unified Business Flow Abstract
Adopt
Page Operation Abstract
Page Abstract
Component Operation Abstract
Component Abstract
Test Case
Test Date Provider
Test Date Preparation Tool
Replay test case to execute test
Encapsulated utiliites which call API or directly DB access to prepare test data
Local Test Execution Environment
System Under Test (SUT)
Use OOB/GDS test data
Page Encapsulation Code Generator
Out–of–box Test Data (Golden Data Set)

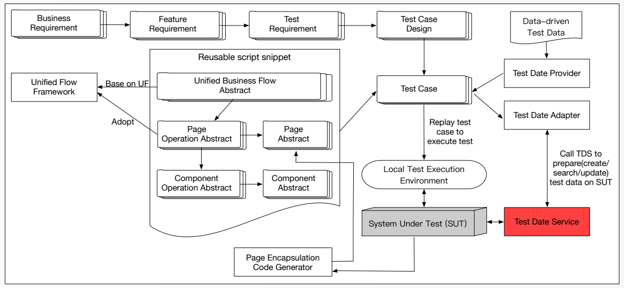
测试架构的演变 – Test Data Platform

引入Test Data Service，提供统一的测试数据准备服务

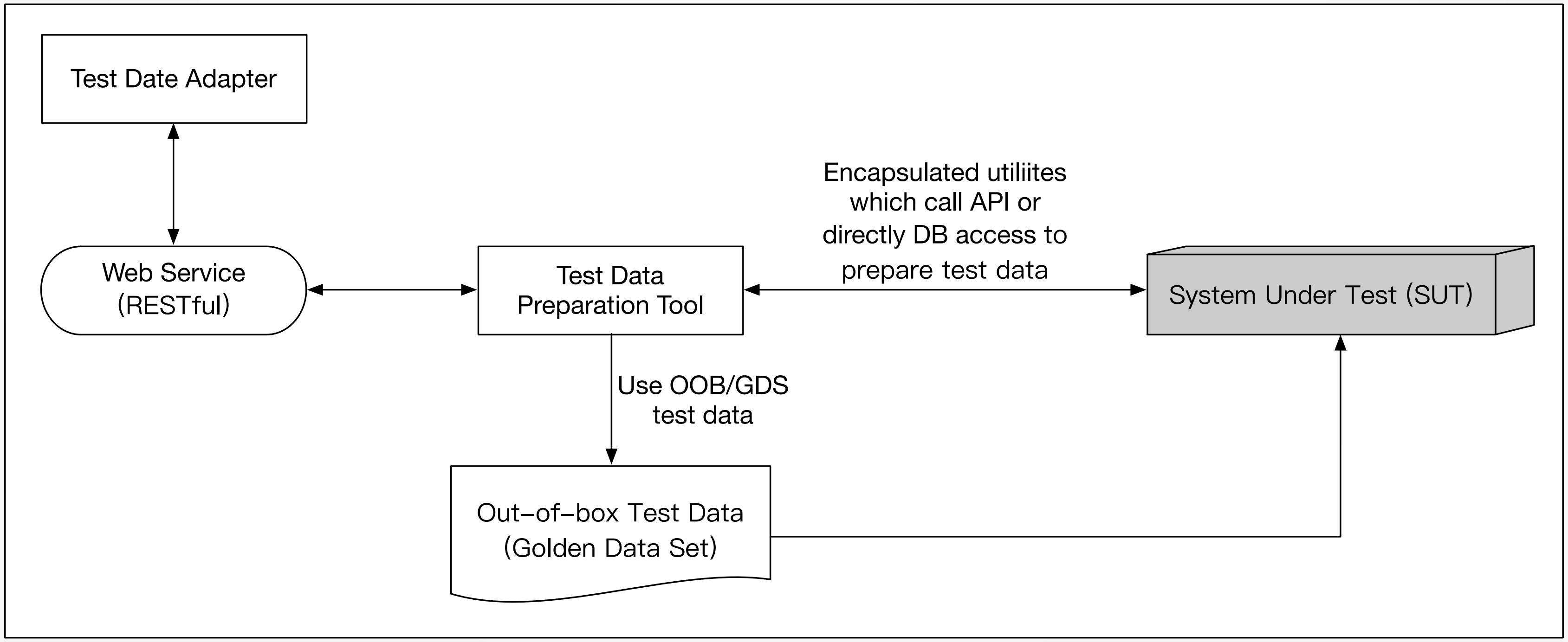


TiD2017

测试架构的演变 – Test Data Platform

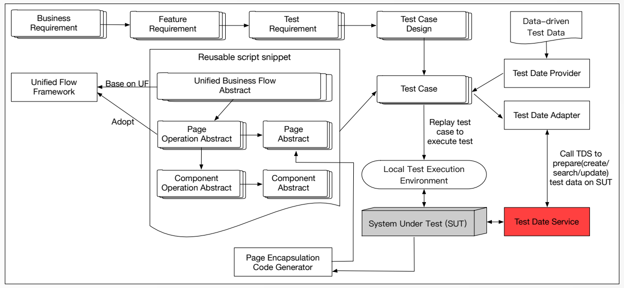


Test Data Service的雏形

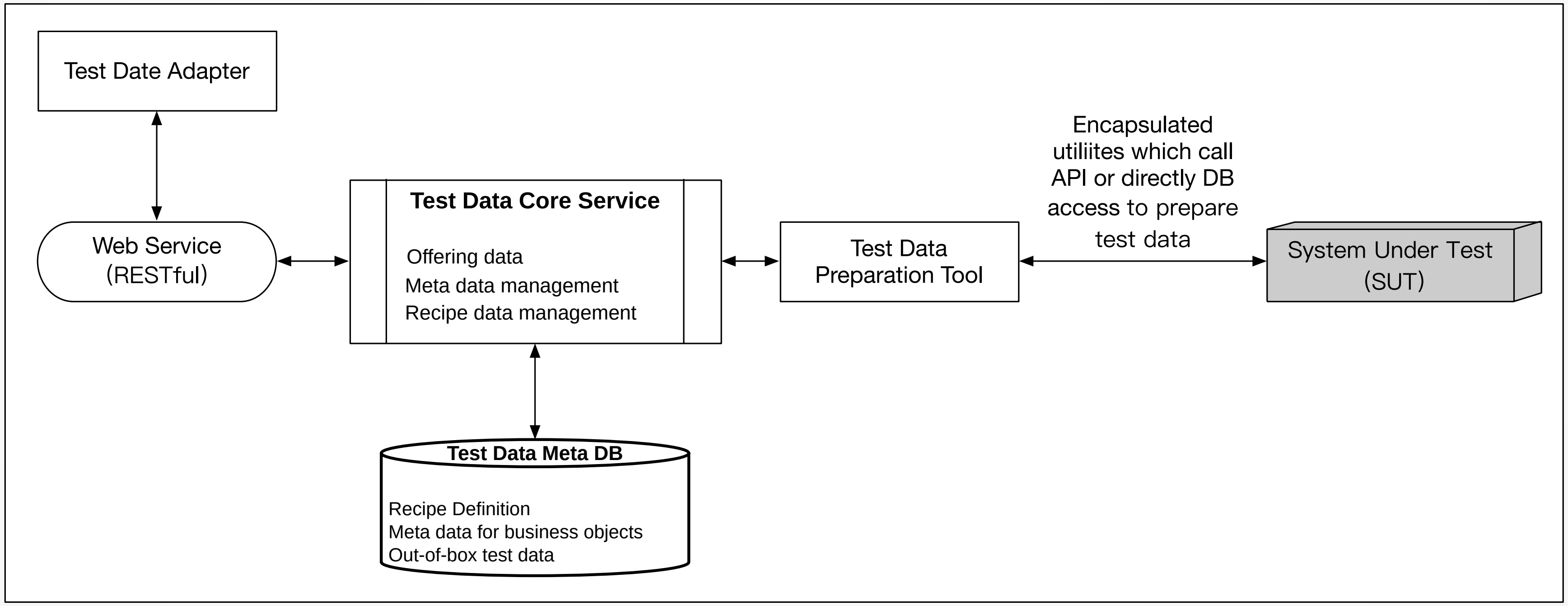

Test Date Adapter
Web Service (RESTful)
Test Data Preparation Tool
Encapsulated utiliites which call API or directly DB access to prepare test data
System Under Test (SUT)
Use OOB/GDS test data
Out–of–box Test Data (Golden Data Set)

TiD2017

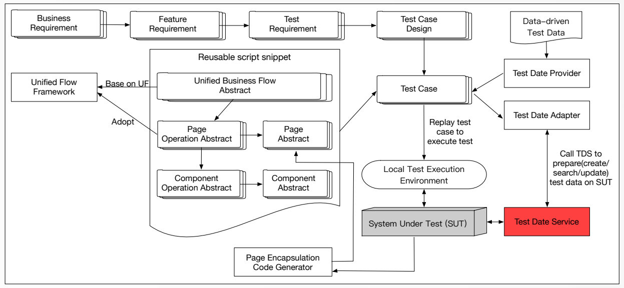
测试架构的演变 – Test Data Platform



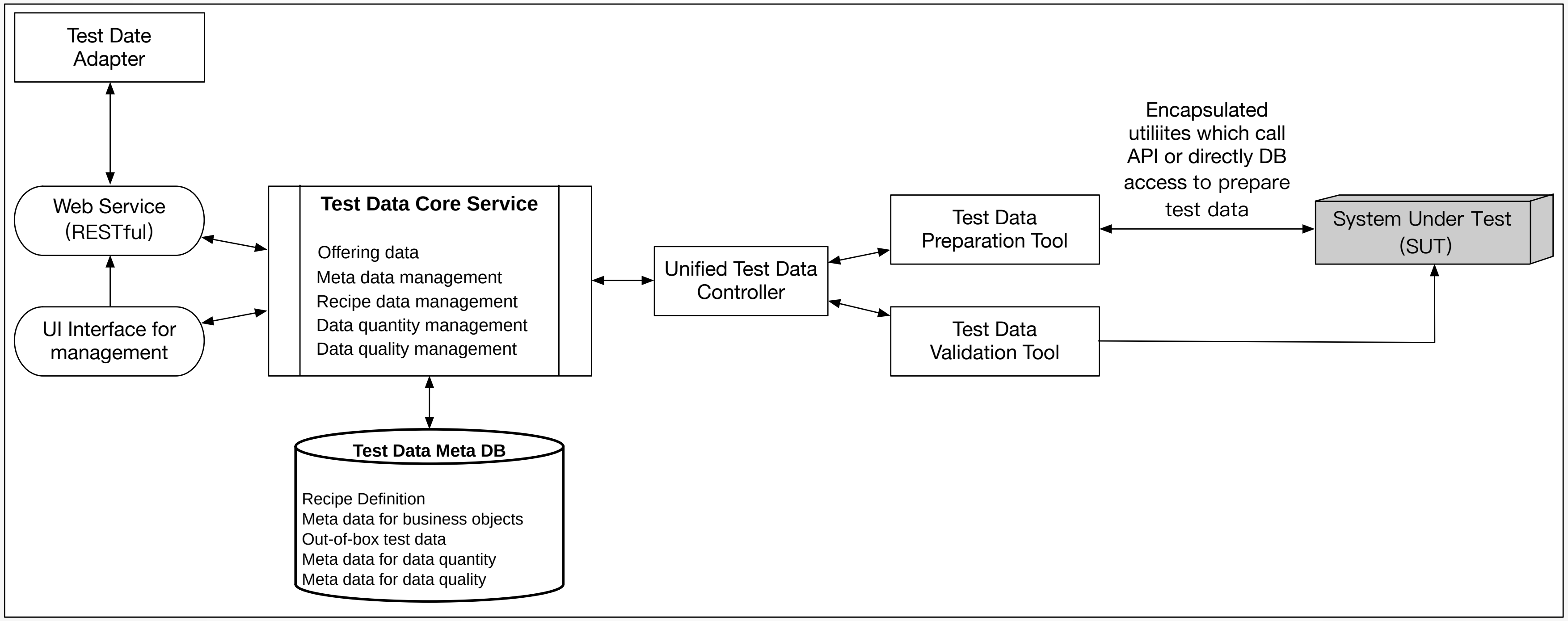
引入Test Data Core Service和Recipe



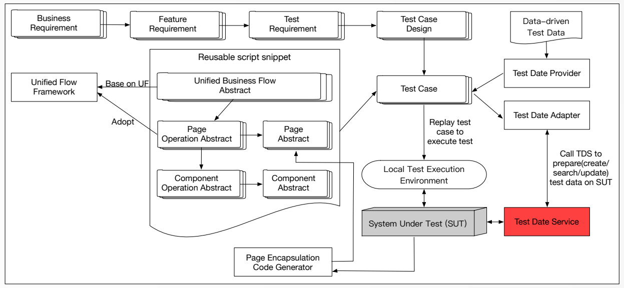
测试架构的演变 – Test Data Platform



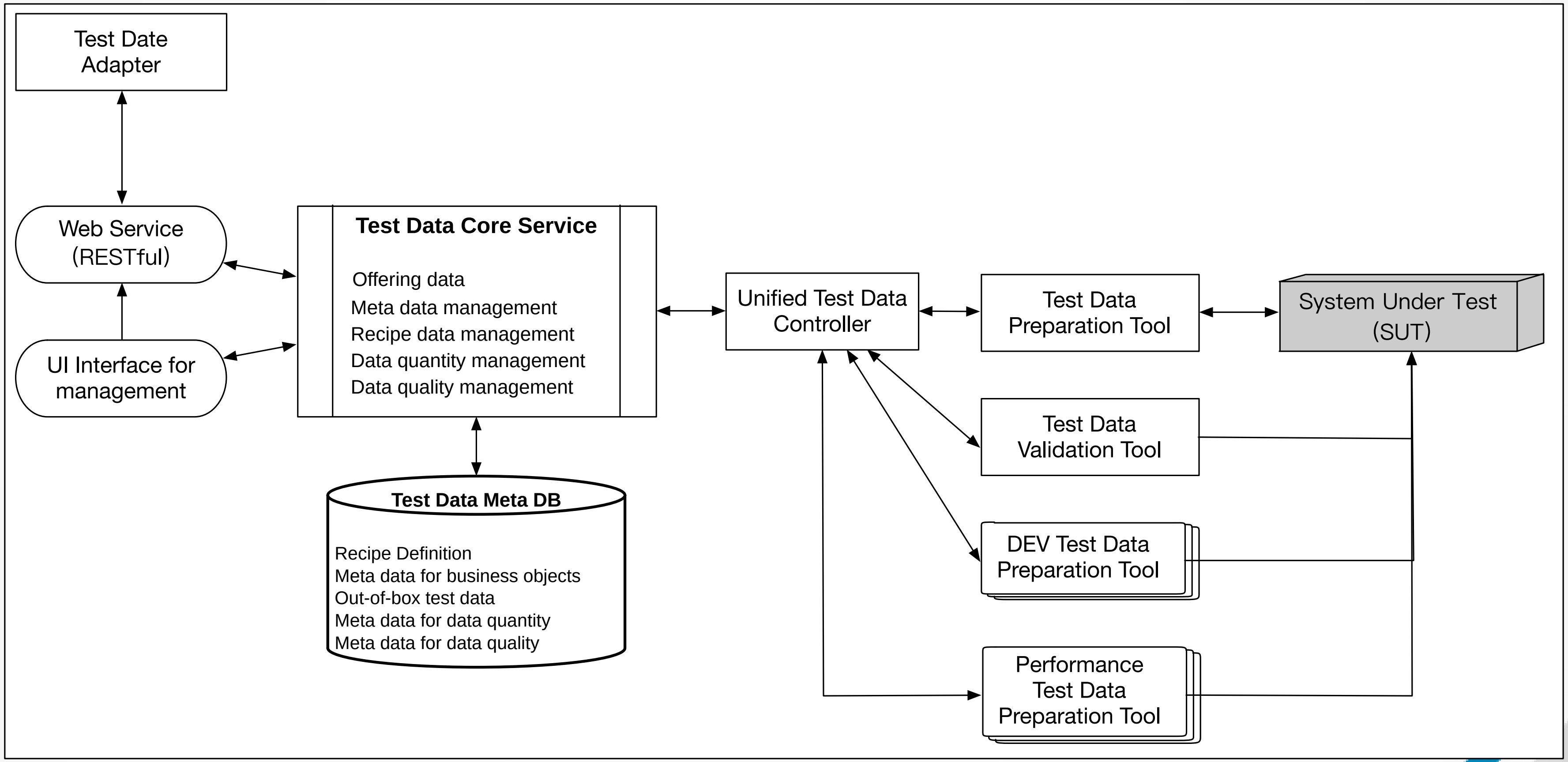
引入Data Quantity / Quality管理


Test Date Adapter
Web Service (RESTful)
UI Interface for management
Test Data Core Service
Offering data
Meta data management
Recipe data management
Data quantity management
Data quality management
Test Data Meta DB
Recipe Definition
Meta data for business objects
Out-of-box test data
Meta data for data quantity
Meta data for data quality
Unified Test Data Controller
Test Data Preparation Tool
Test Data Validation Tool
Encapsulated utiliites which call API or directly DB access to prepare test data
System Under Test (SUT)

测试架构的演变 – Test Data Platform

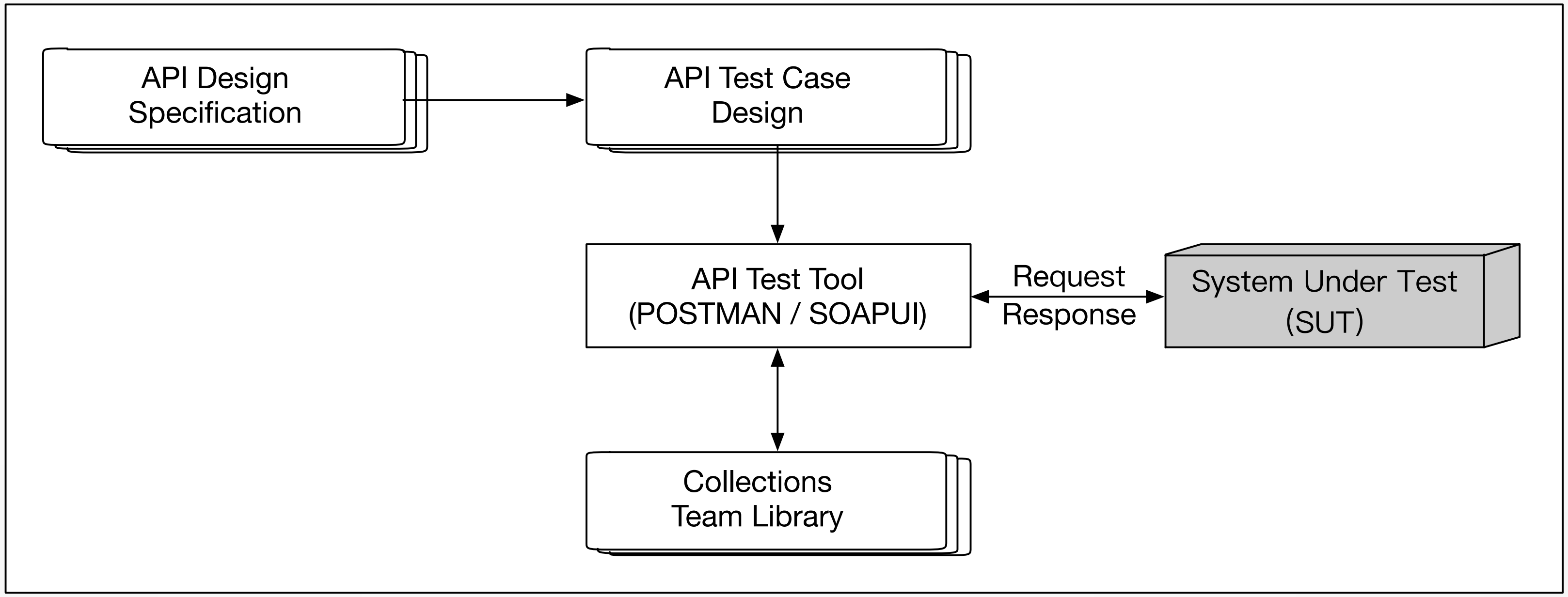


引入Unified Controller接入不同Test Data Tools


Test Date Adapter
Web Service (RESTful)
UI Interface for management
Test Data Core Service
Offering data
Meta data management
Recipe data management
Data quantity management
Data quality management
Test Data Meta DB
Recipe Definition
Meta data for business objects
Out-of-box test data
Meta data for data quantity
Meta data for data quality
Unified Test Data Controller
Test Data Preparation Tool
System Under Test (SUT)
Test Data Validation Tool
DEV Test Data Preparation Tool
Performance Test Data Preparation Tool

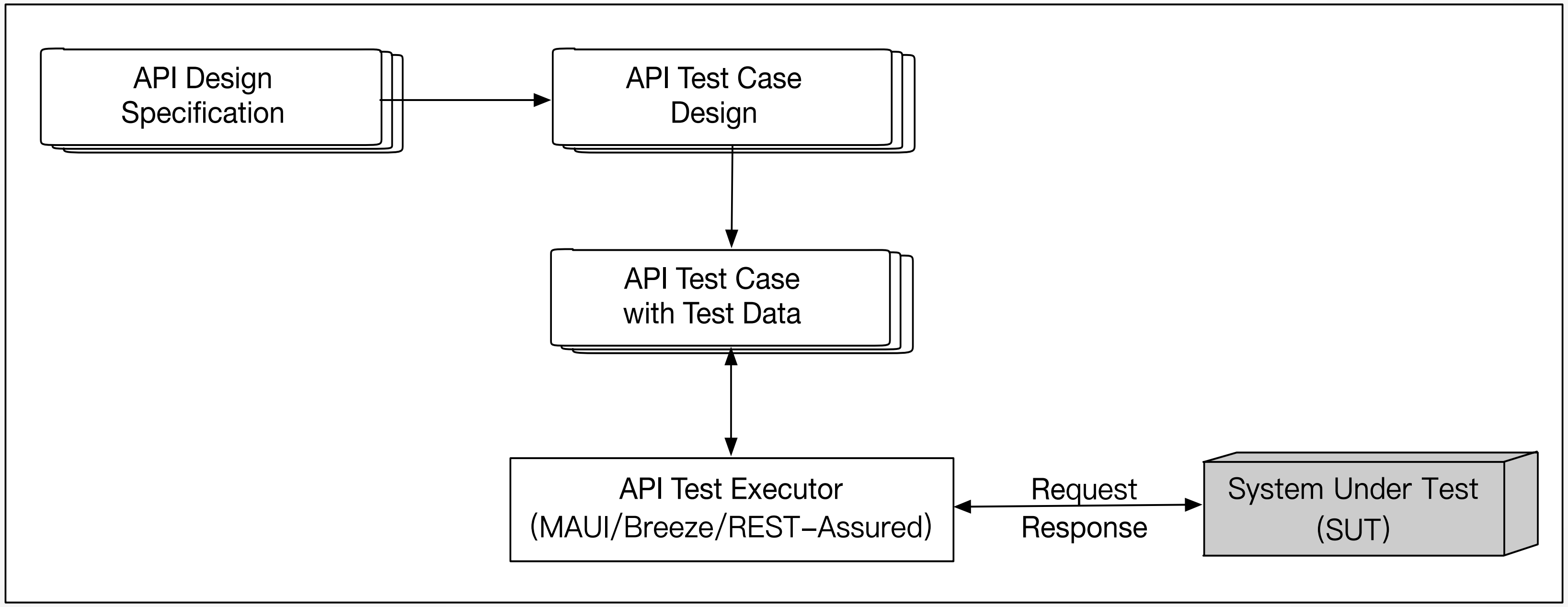
测试架构的演变 – API Automation Test Framework

最原始的API测试



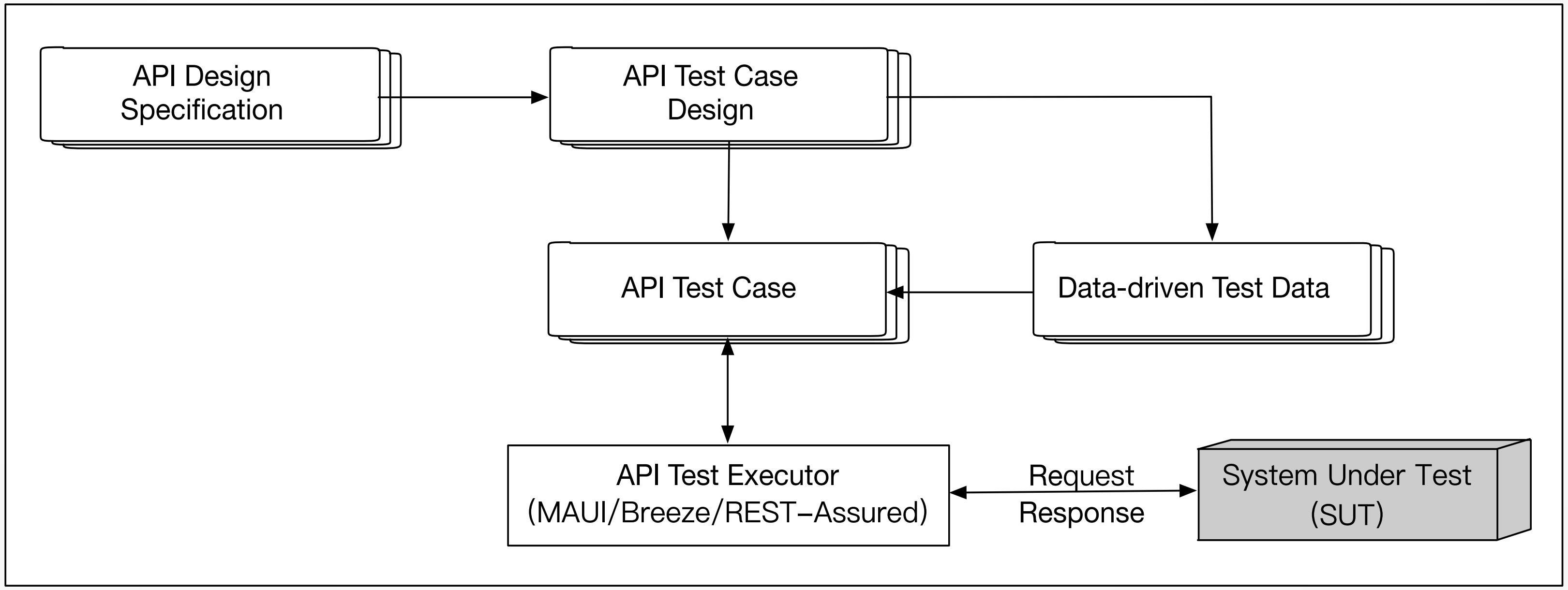
测试架构的演变 – API Automation Test Framework

引入API Test Executor实现Code-based API自动化测试



测试架构的演变 – API Automation Test Framework

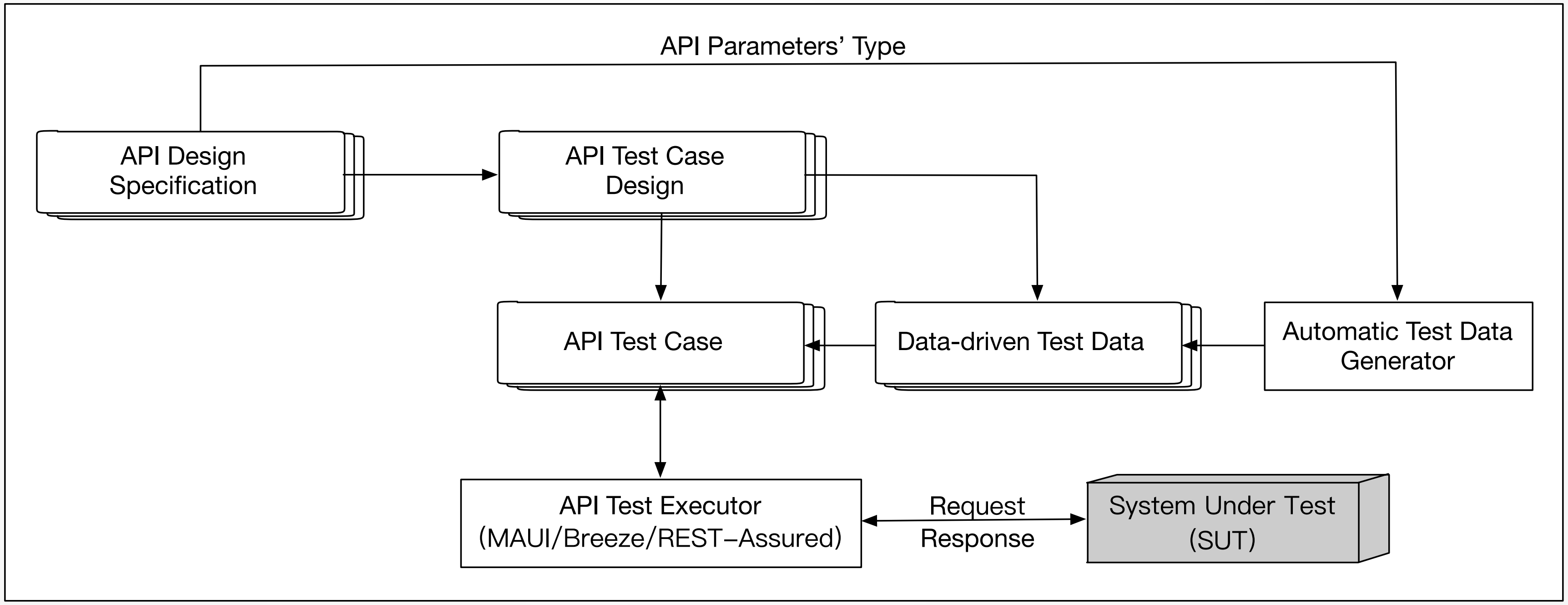
Test Case和Test Data分离实现Data-Driven Test



TiD2017

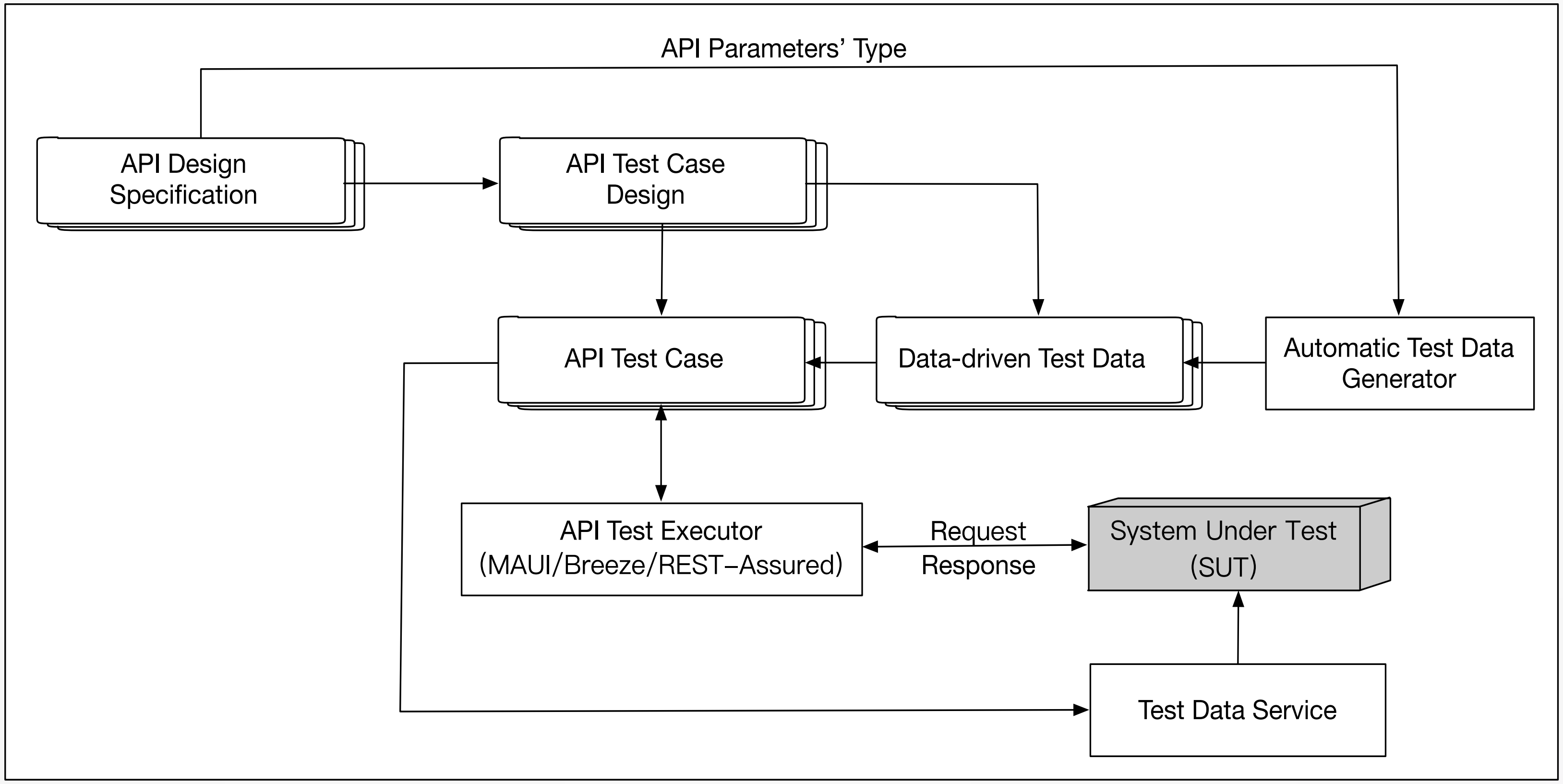
测试架构的演变 – API Automation Test Framework

引入Data-Driven Test Data的自动生成



测试架构的演变 – API Automation Test Framework

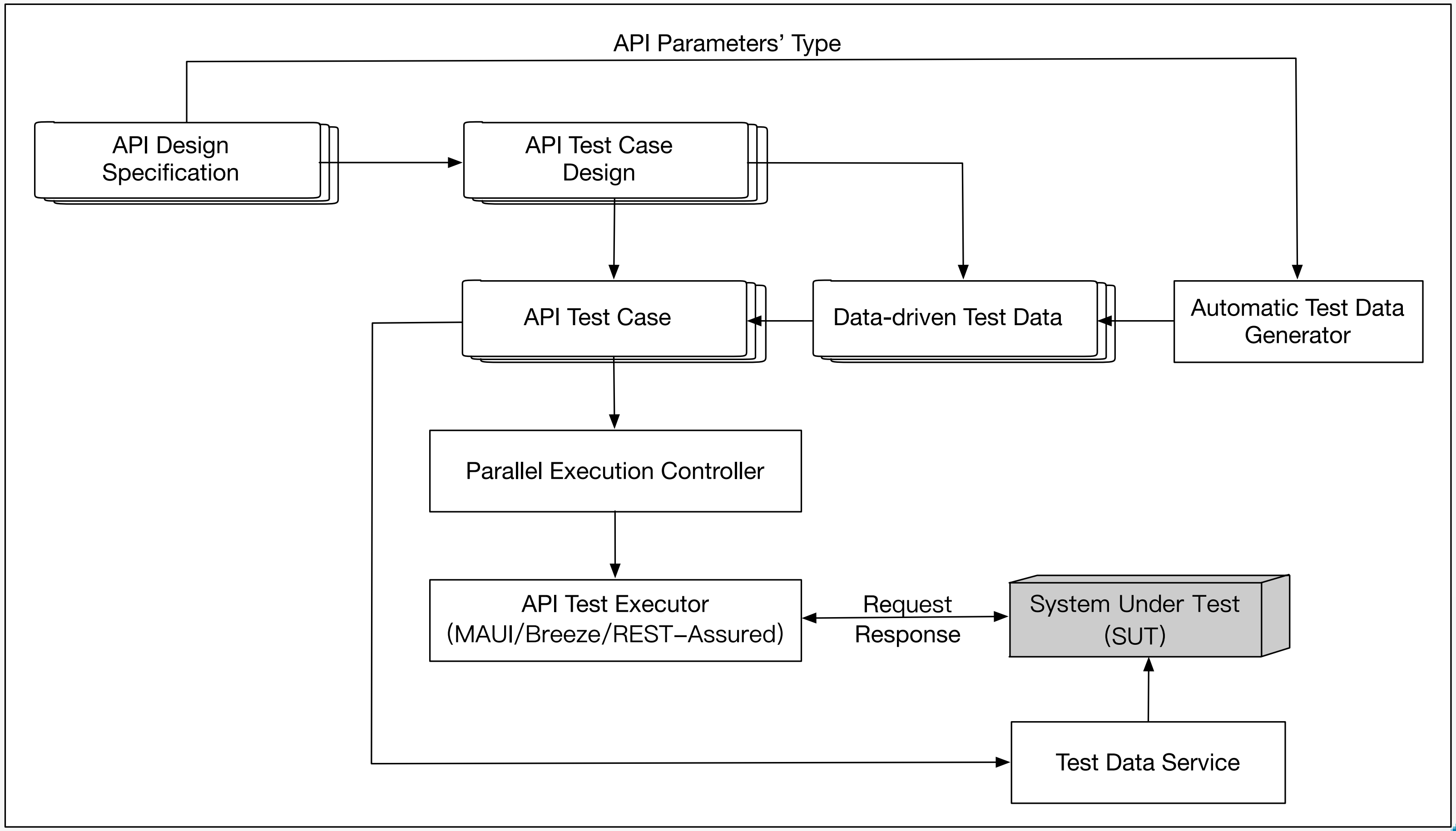
引入Test Data Service生成测试数据



TiD2017

测试架构的演变 – API Automation Test Framework

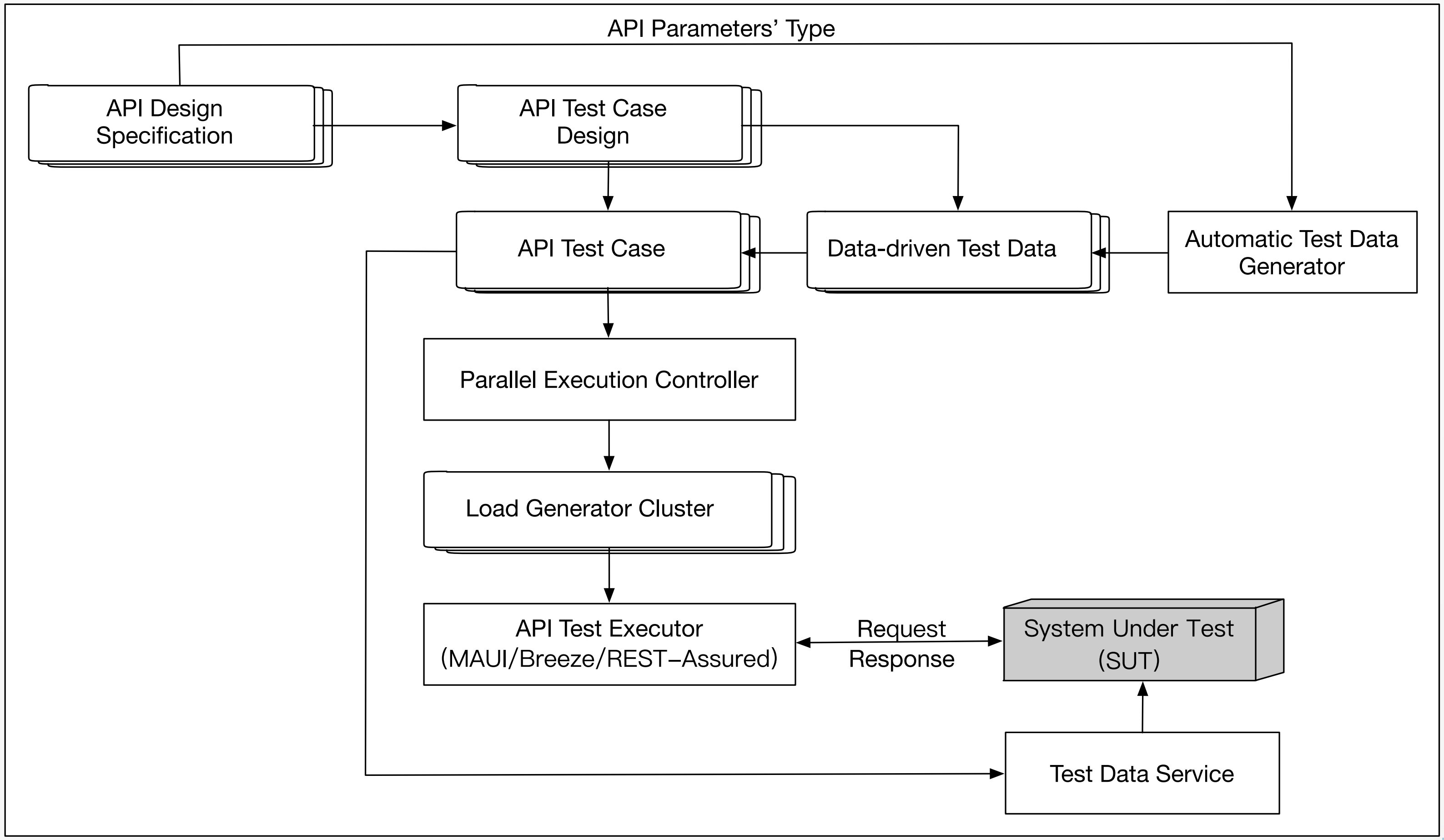
引入Parallel Execution Controller实现API并发测试



TiD2017

测试架构的演变 – API Automation Test Framework

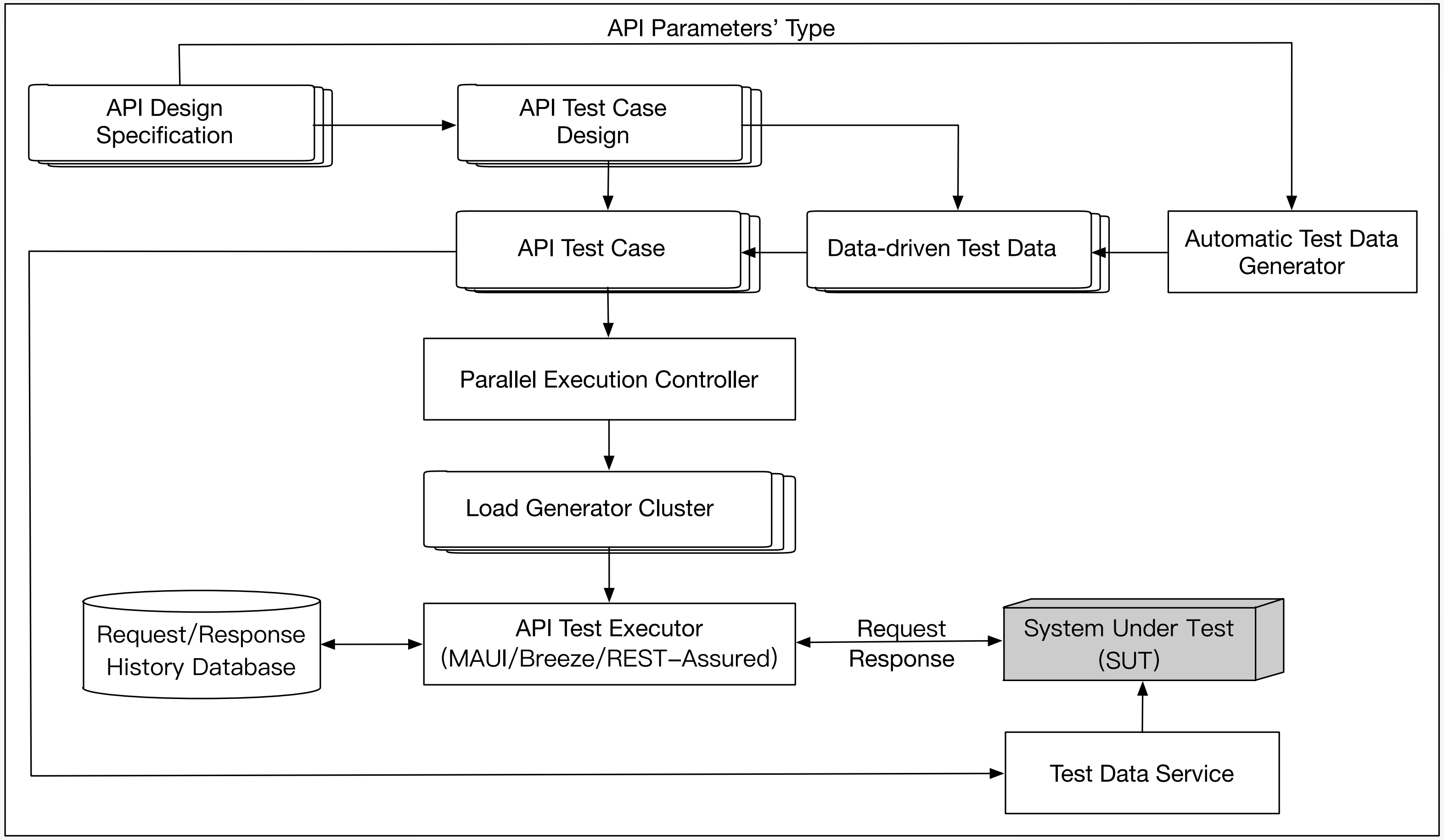
引入Load Generator Cluster实现API高并发和压力测试



TiD2017

测试架构的演变 – API Automation Test Framework

引入2R History Database实现API Diff Identification

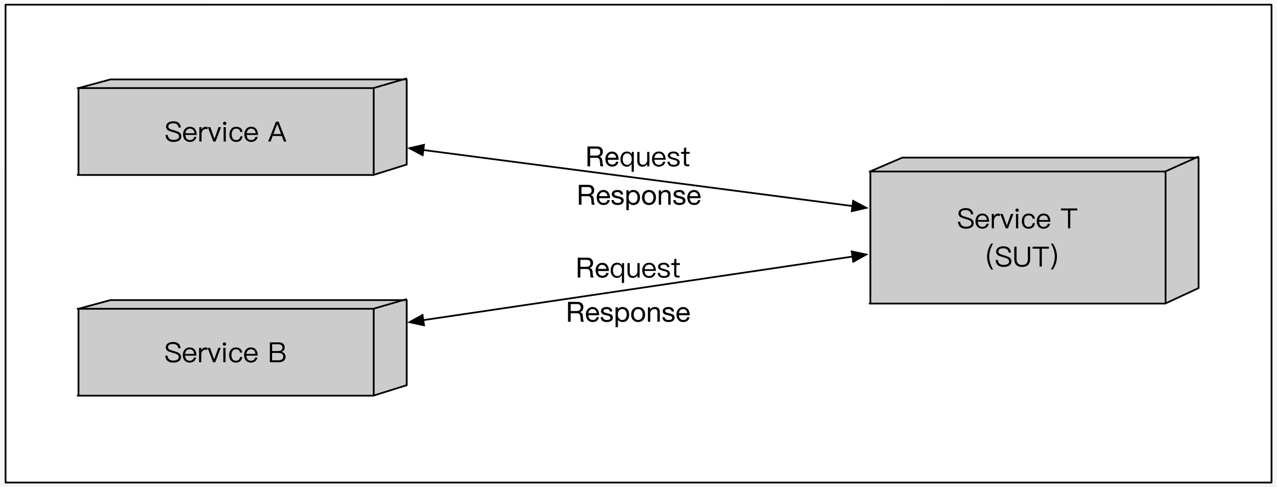


TiD2017

测试架构的演变 – API Automation Test Framework

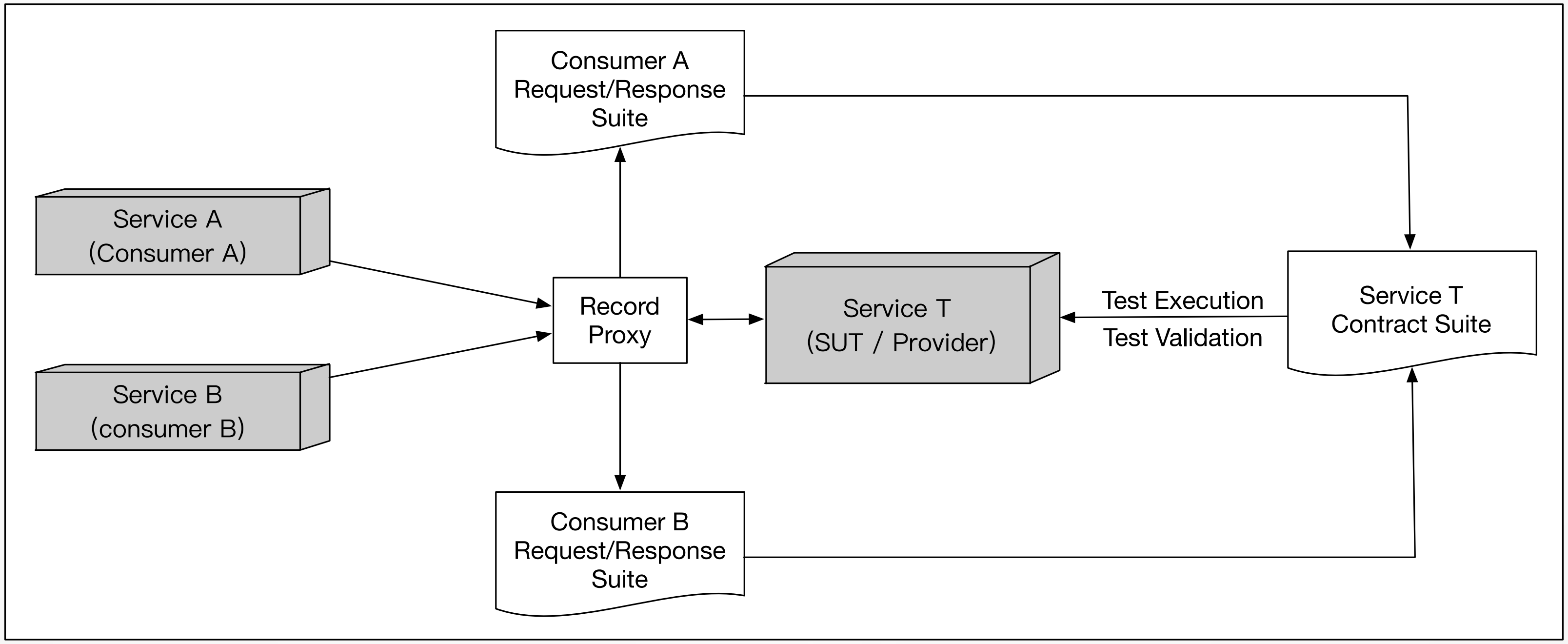
微服务架构下的API测试挑战

- ❑ API的种类数量多
- ❑ API测试高覆盖率的代价大
- ❑ 微服务架构之间的耦合
- ❑ 第三方服务的耦合



测试架构的演变 – API Automation Test Framework

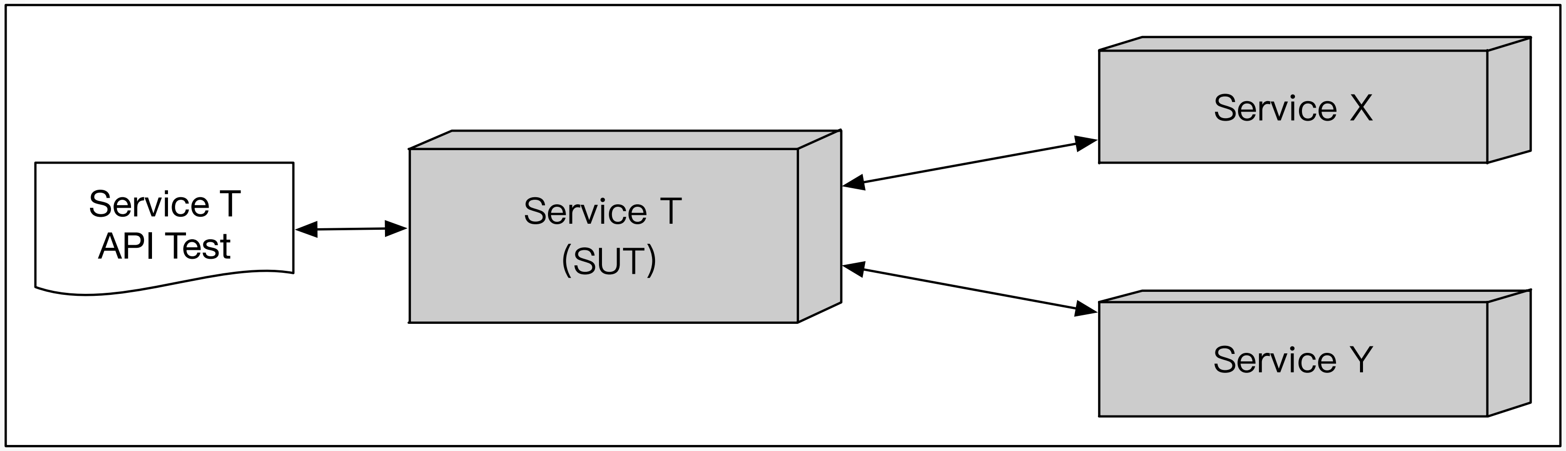
基于消费者契约的API测试 – 基于契约的测试与验证



TiD2017

测试架构的演变 – API Automation Test Framework

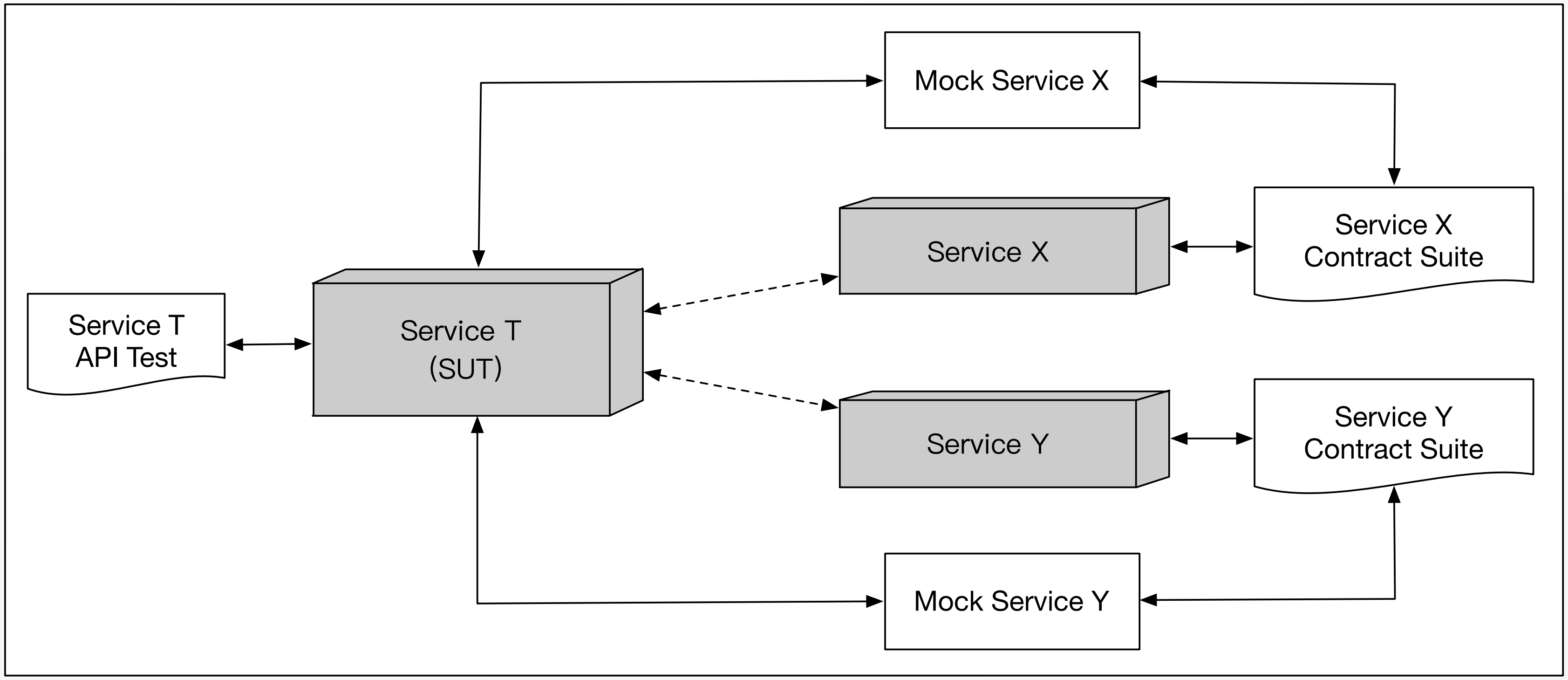
基于消费者契约的API测试 – 基于契约的Mock Service实现API依赖解耦



TiD2017

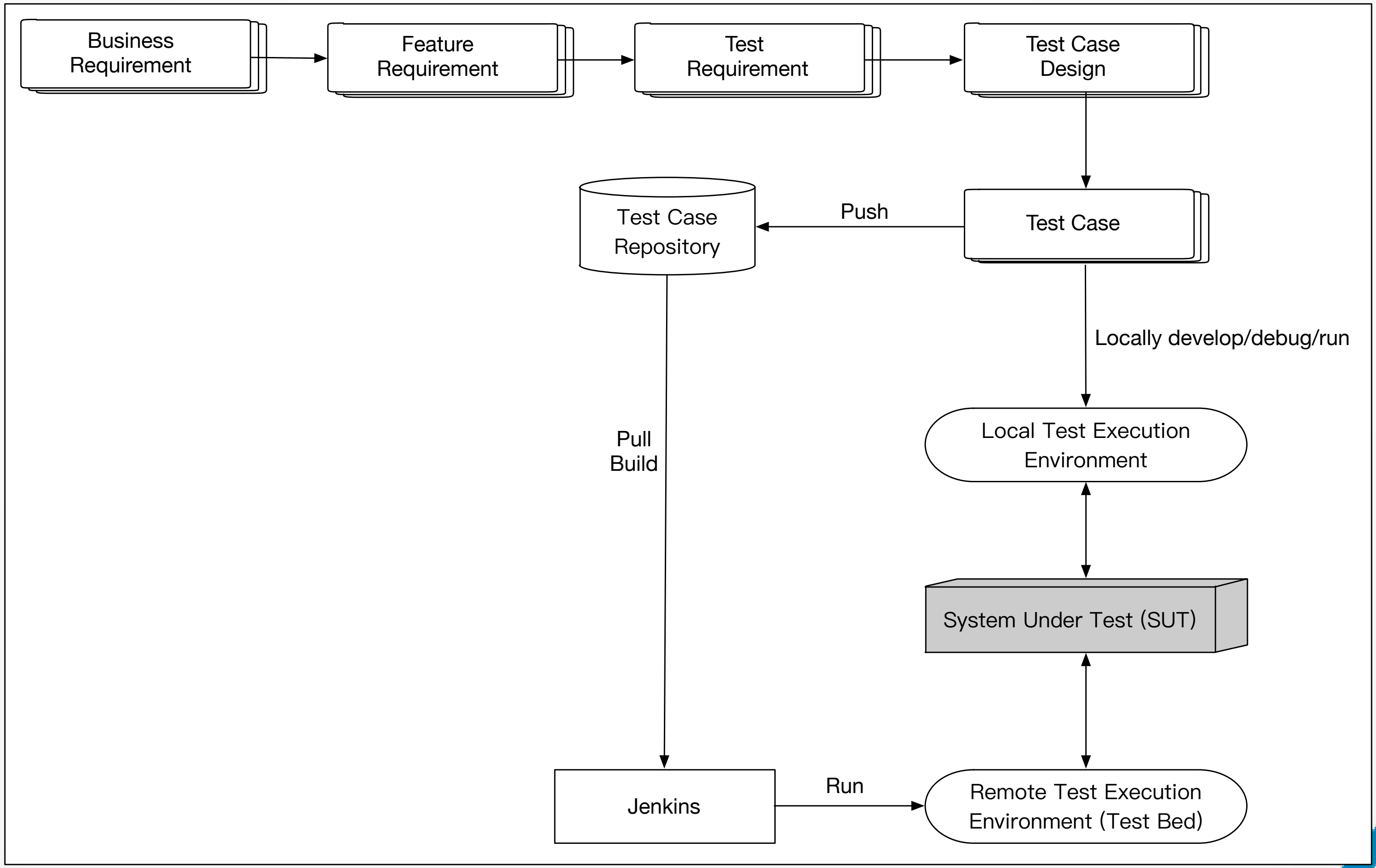
测试架构的演变 – API Automation Test Framework

基于消费者契约的API测试 – 基于契约的Mock Service实现API依赖解耦



测试架构的演变 - Test Execution Environment

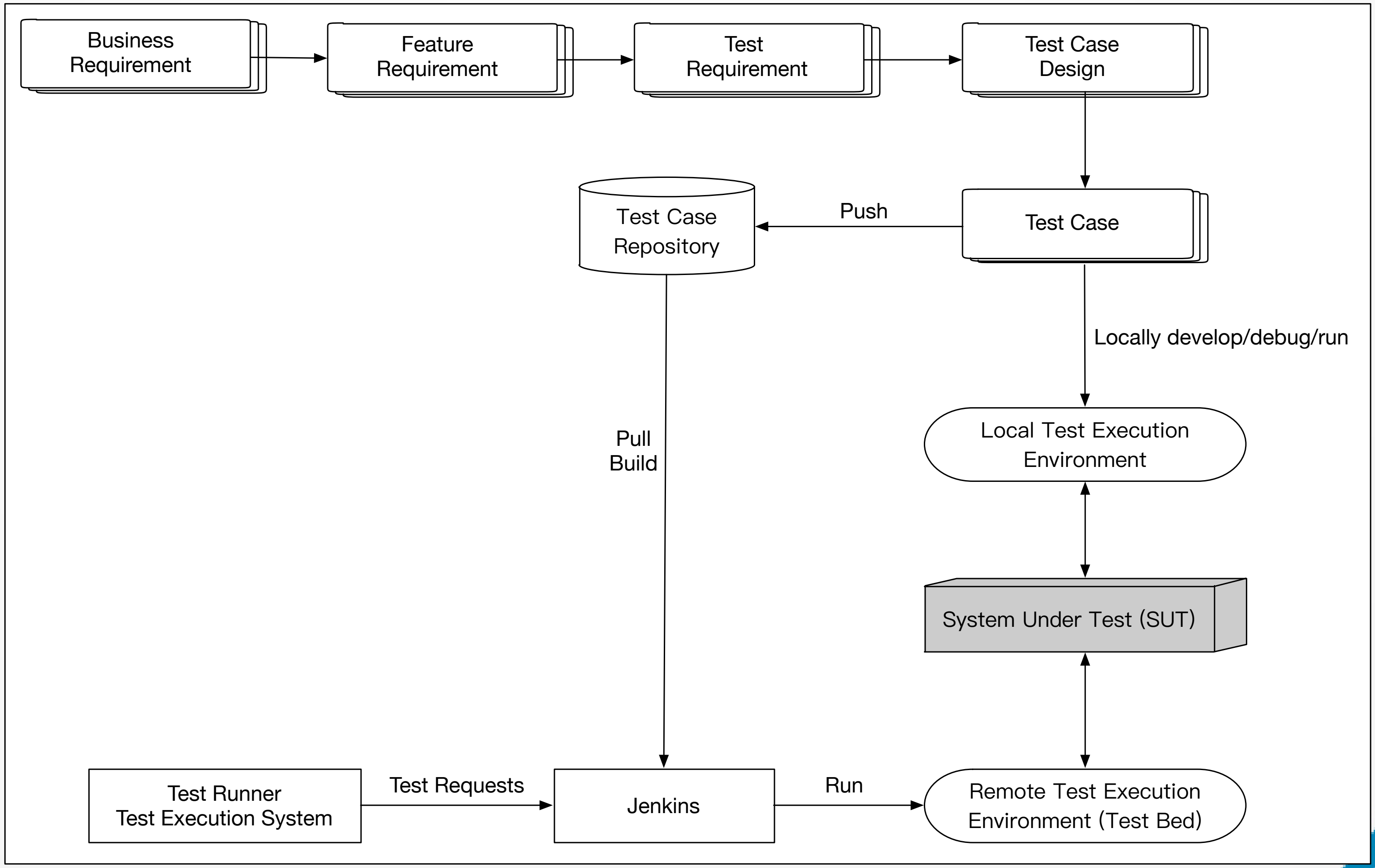
基于Jenkins触发测试执行



TiD2017

测试架构的演变 - Test Execution Environment

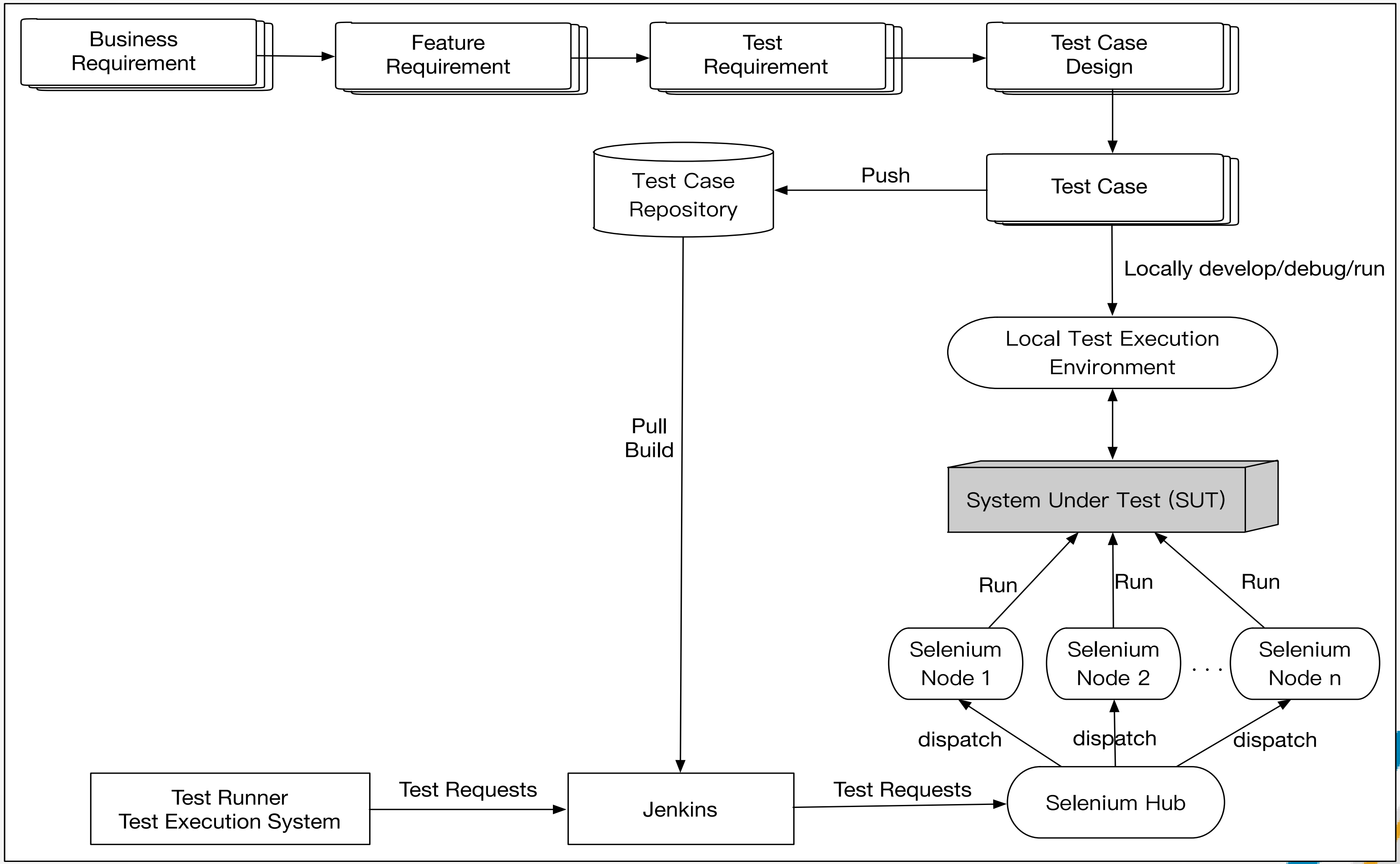
基于Test Runner / Test Execution System



TiD2017

测试架构的演变 - Test Execution Environment

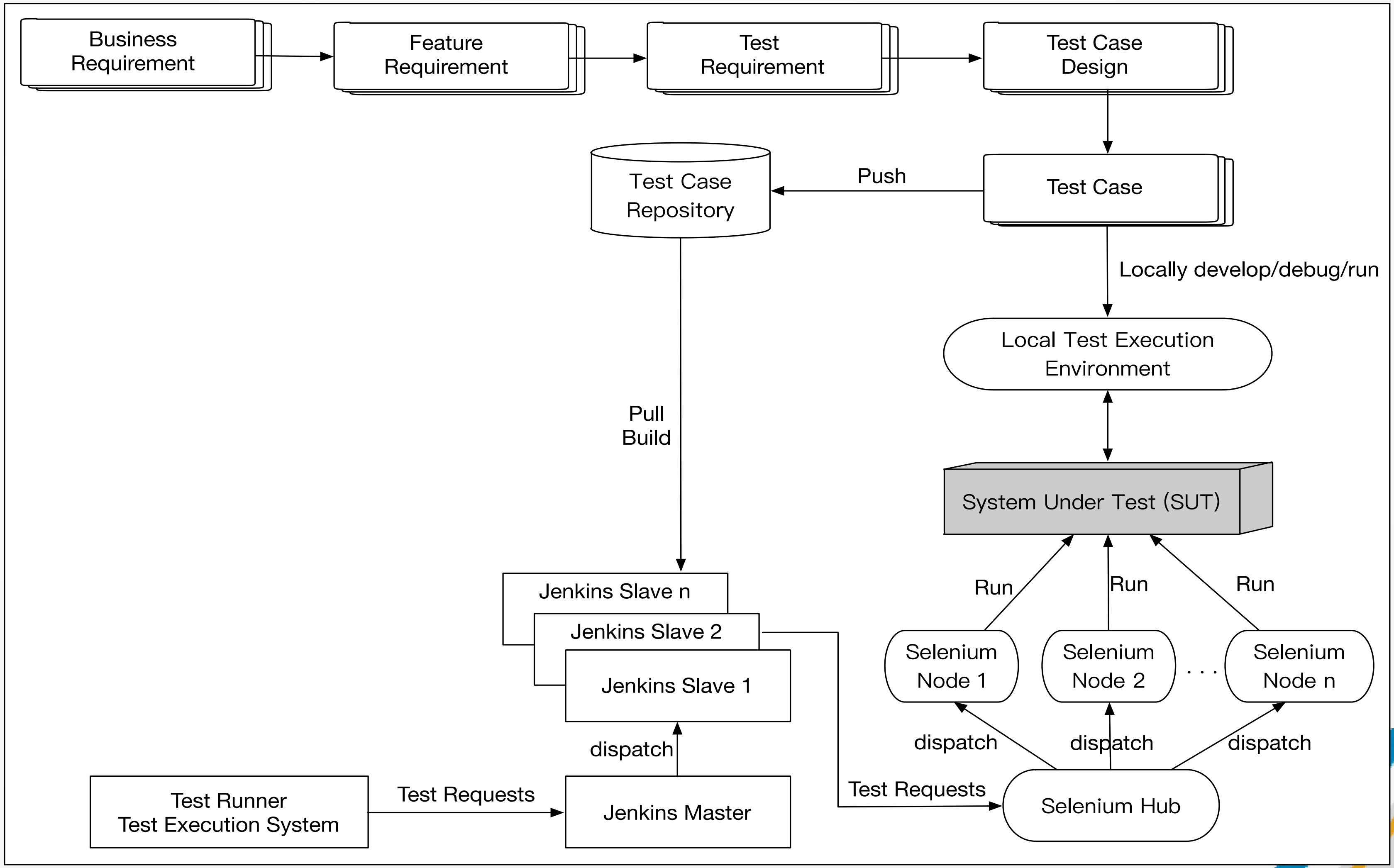
基于Selenium Grid提高测试并行执行能力



TiD2017

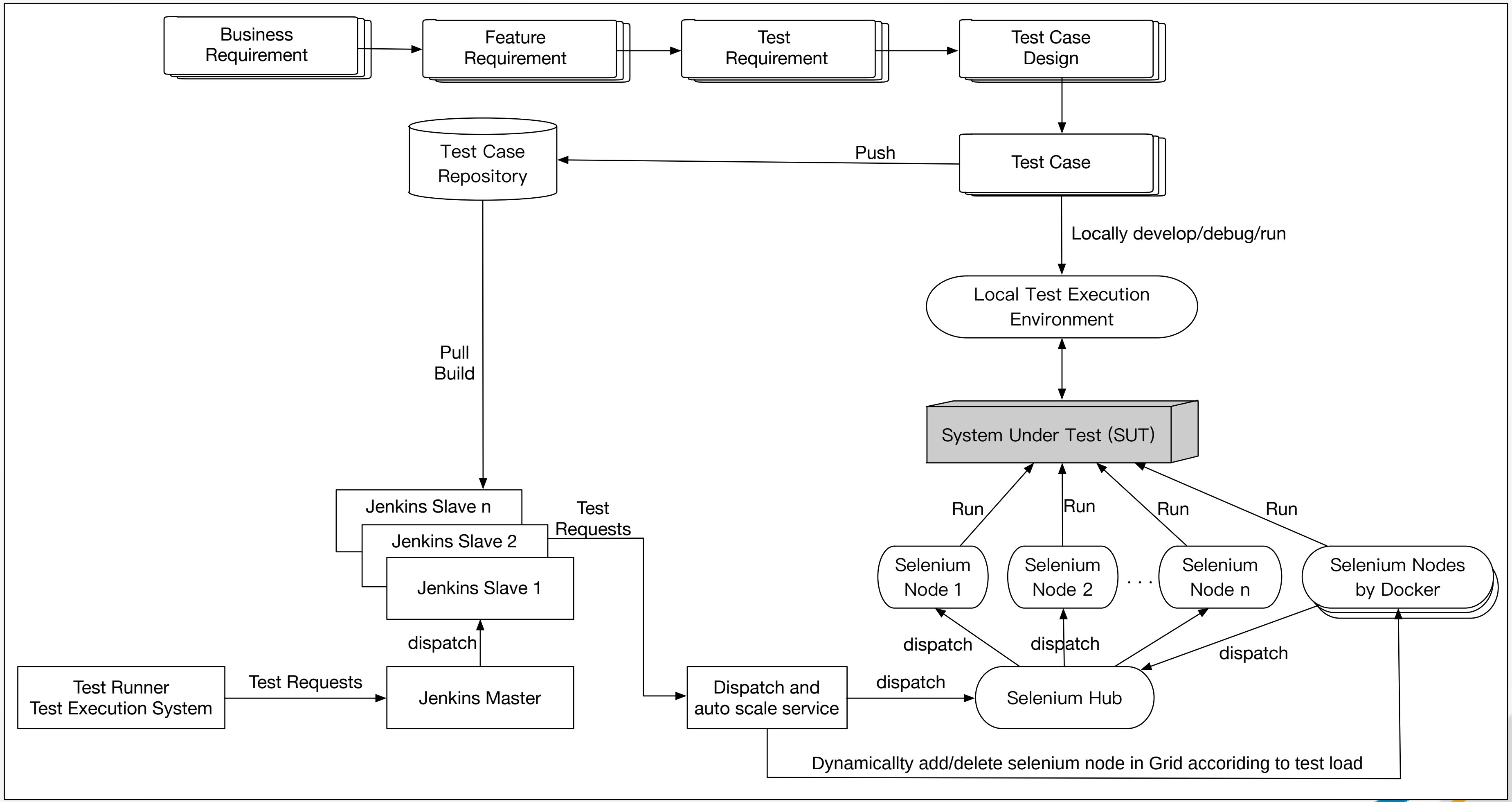
测试架构的演变 - Test Execution Environment

基于Jenkins Cluster提高测试并行执行能力



测试架构的演变 - Test Execution Environment

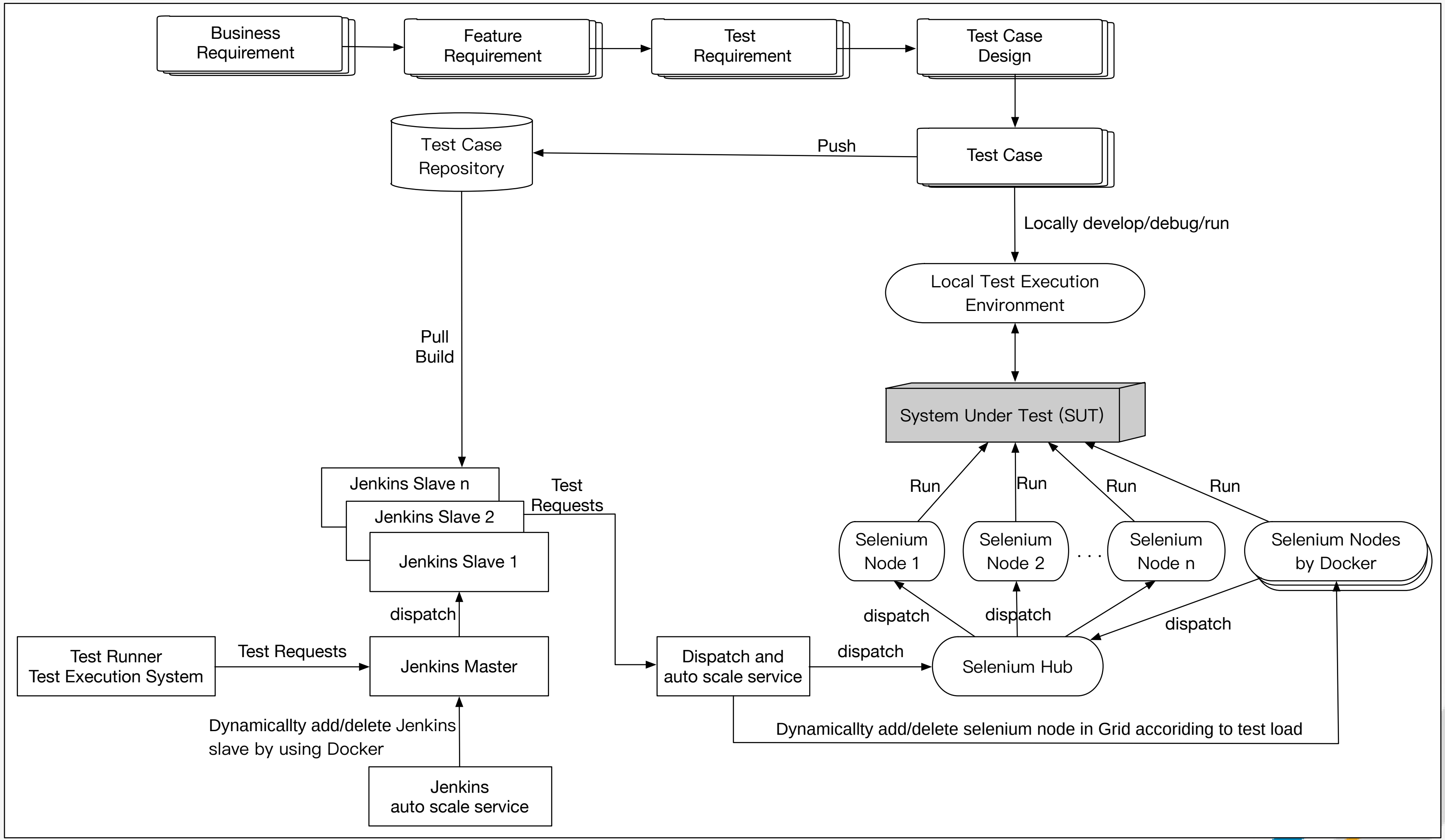
基于测试负载，用Docker实现Selenium Grid的动态扩展与收缩



TiD2017

测试架构的演变 - Test Execution Environment

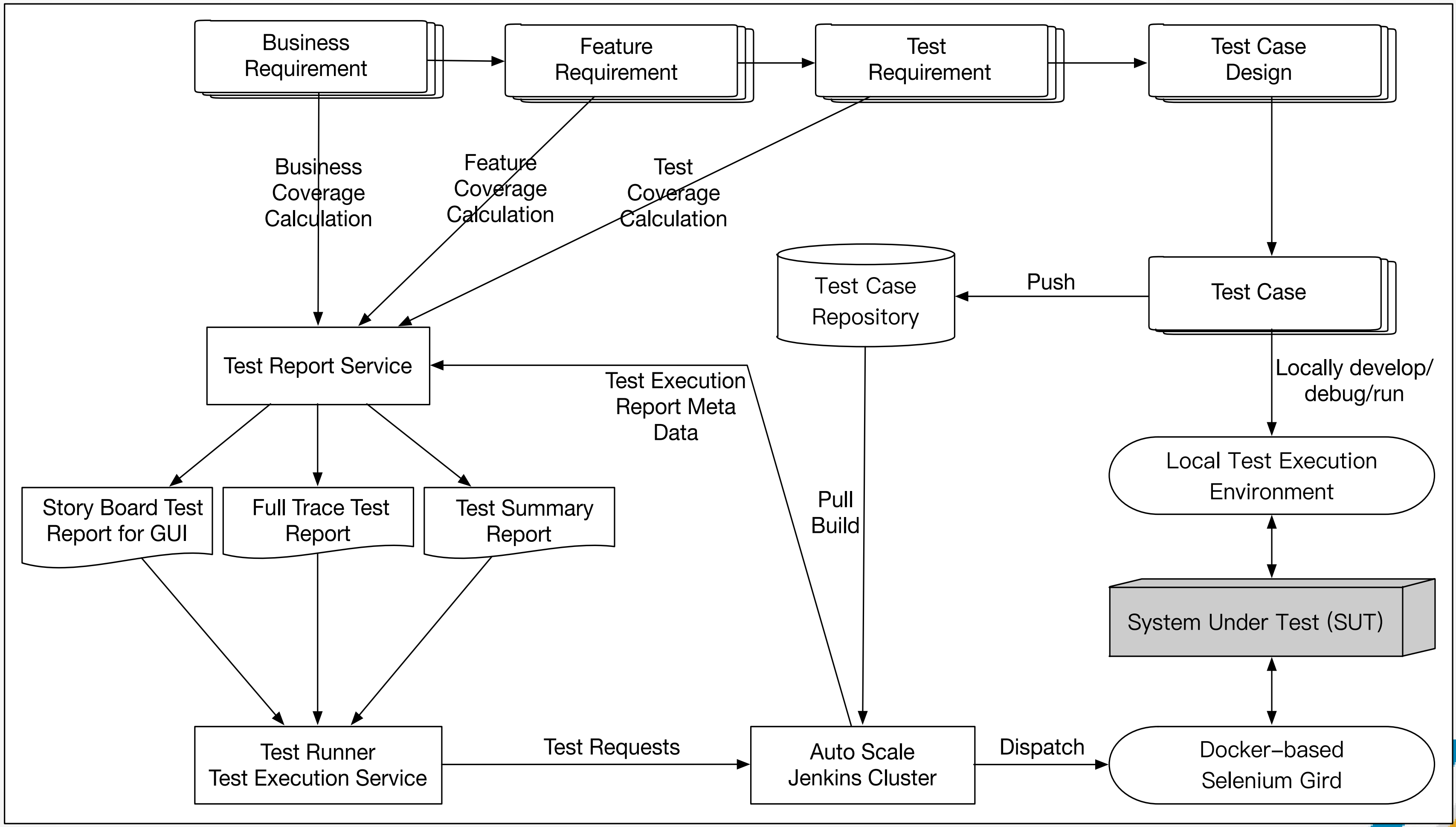
基友Docker实现Jenkins Cluster的动态扩展与收缩



TiD2017

测试架构的演变 - Test Execution and Management Platform

引入Test Report Service生成各种测试报告



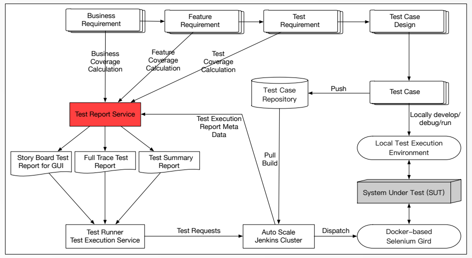
Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Business Coverage Calculation
Feature Coverage Calculation
Test Coverage Calculation
Test Case Repository
Push
Test Case
Test Report Service
Test Execution Report Meta Data
Locally develop/ debug/run
Local Test Execution Environment
Story Board Test Report for GUI
Full Trace Test Report
Test Summary Report
Pull Build
System Under Test (SUT)
Test Runner Test Execution Service
Test Requests
Auto Scale Jenkins Cluster
Dispatch
Docker–based Selenium Gird

测试架构的演变 - Test Execution and Management Platform

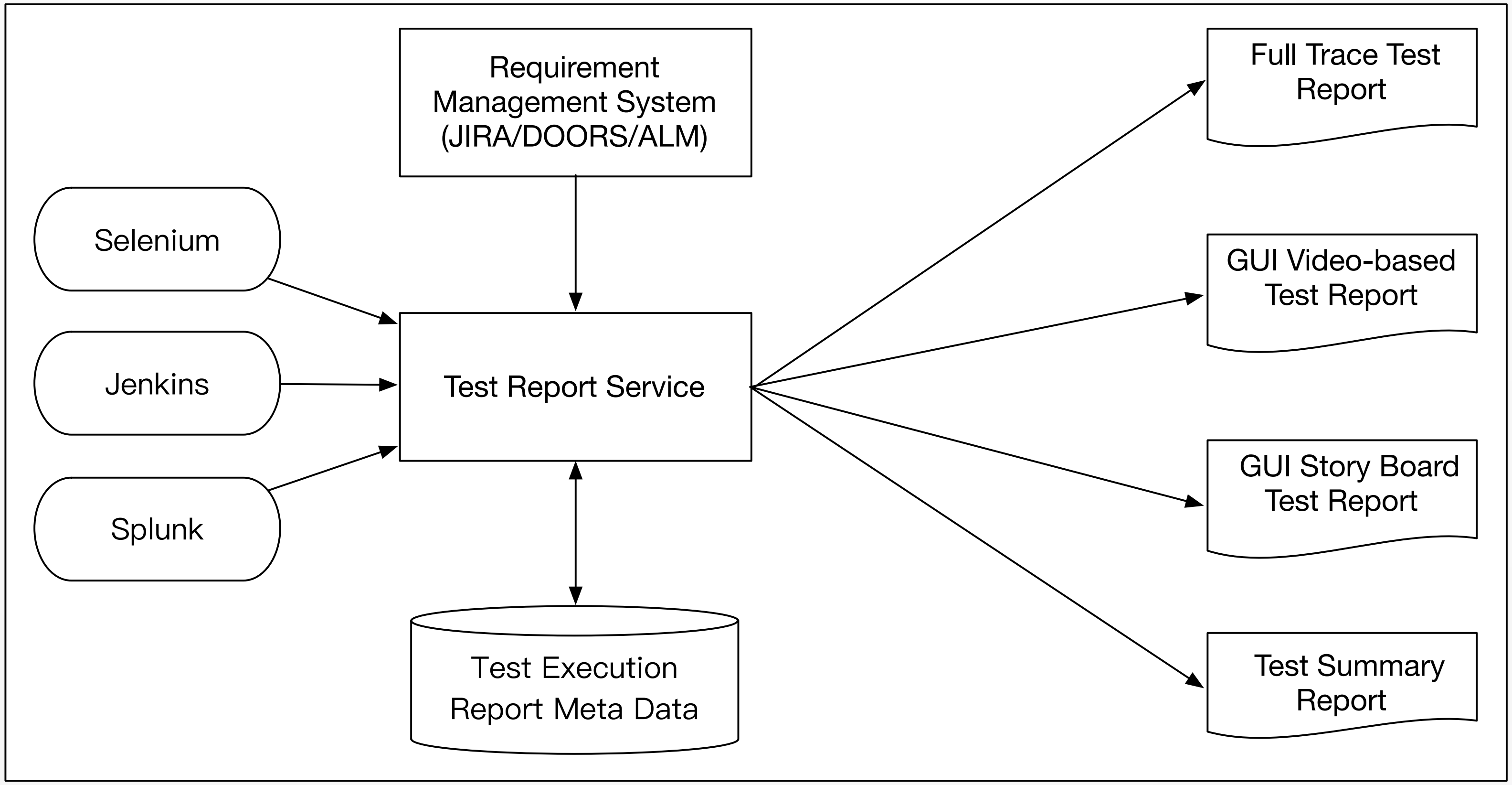
Test Report Service生成的各种测试报告

- ❑ **Full Trace Test Report for DEV/QE**
- ❑ **GUI Video-based Test Report for PM/PO**
- ❑ **Story Board Test Report for PM/PO**
- ❑ **Multi-Site Story Board Comparison Test Report for LQA**
- ❑ **Test Summary Report for management team**
- ❑ **Test Trend Report for management team**

测试架构的演变 - Test Report Platform

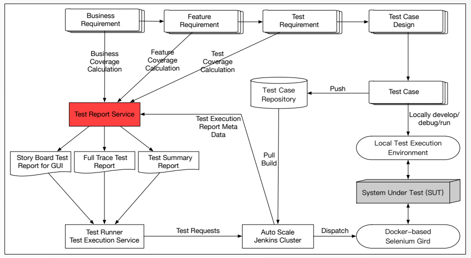

Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Business Coverage Calculation
Feature Coverage Calculation
Test Coverage Calculation
Test Case Repository
Push
Test Case
Test Report Service
Test Execution Report Meta Data
Locally develop/debug/run
Local Test Execution Environment
Story Board Test Report for GUI
Full Trace Test Report
Test Summary Report
Pull Build
System Under Test (SUT)
Test Runner Test Execution Service
Test Requests
Auto Scale Jenkins Cluster
Dispatch
Docker–based Selenium Gird

典型的Test Report Platform架构

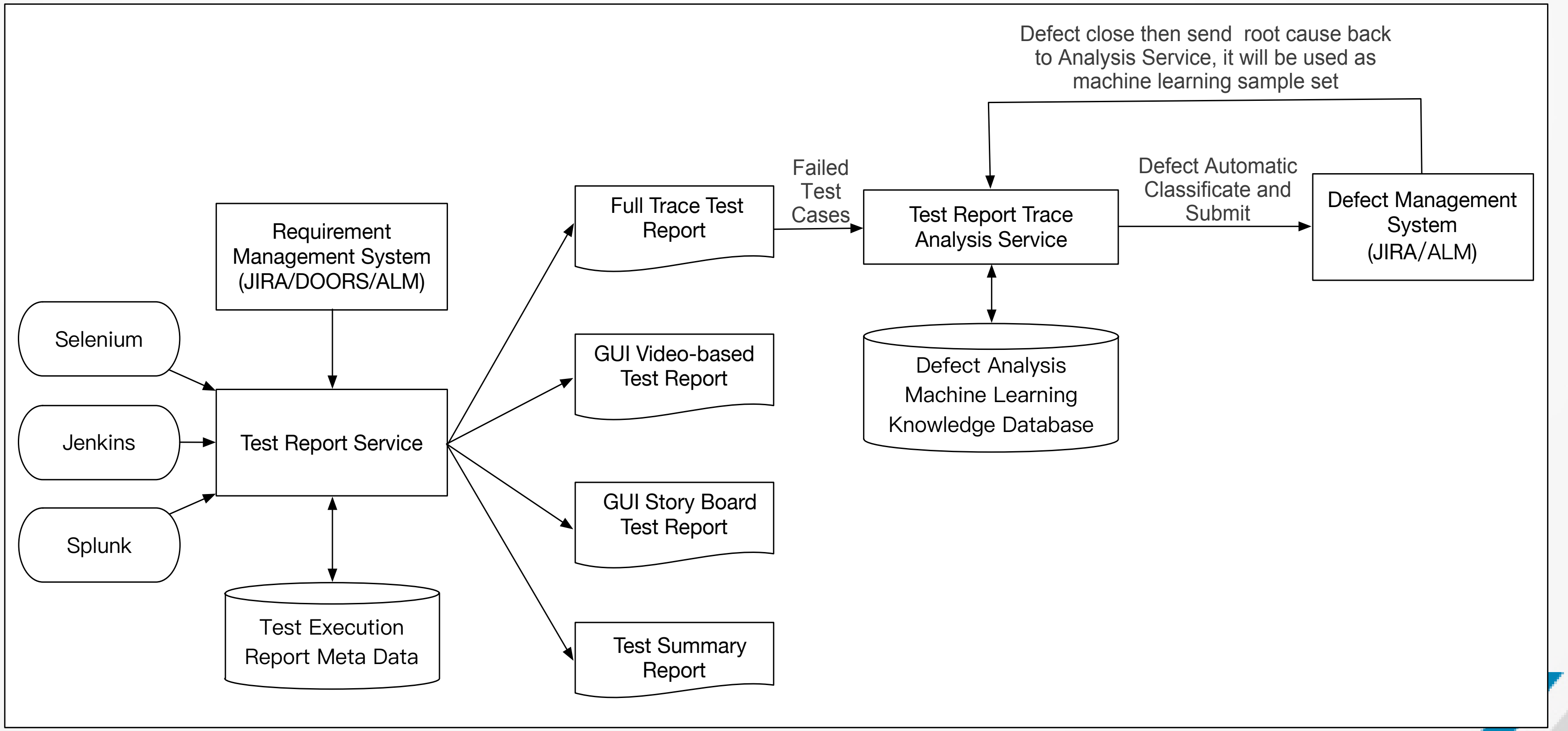

Requirement Management System (JIRA/DOORS/ALM)
Selenium
Jenkins
Splunk
Test Report Service
Test Execution Report Meta Data
Full Trace Test Report
GUI Video-based Test Report
GUI Story Board Test Report
Test Summary Report

TiD2017

测试架构的演变 - Test Report Platform

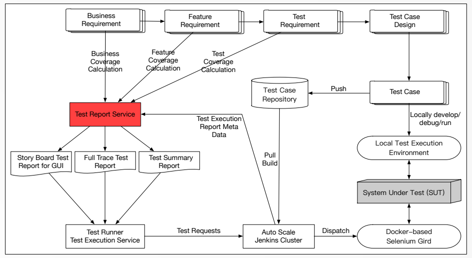

Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Business Coverage Calculation
Feature Coverage Calculation
Test Coverage Calculation
Test Case Repository
Push
Test Case
Test Report Service
Test Execution Report Meta Data
Locally develop/debug/run
Local Test Execution Environment
Story Board Test Report for GUI
Full Trace Test Report
Test Summary Report
Pull Build
System Under Test (SUT)
Test Runner Test Execution Service
Test Requests
Auto Scale Jenkins Cluster
Dispatch
Docker-based Selenium Gird

引入Test Analysis Service提高Defect分类效率

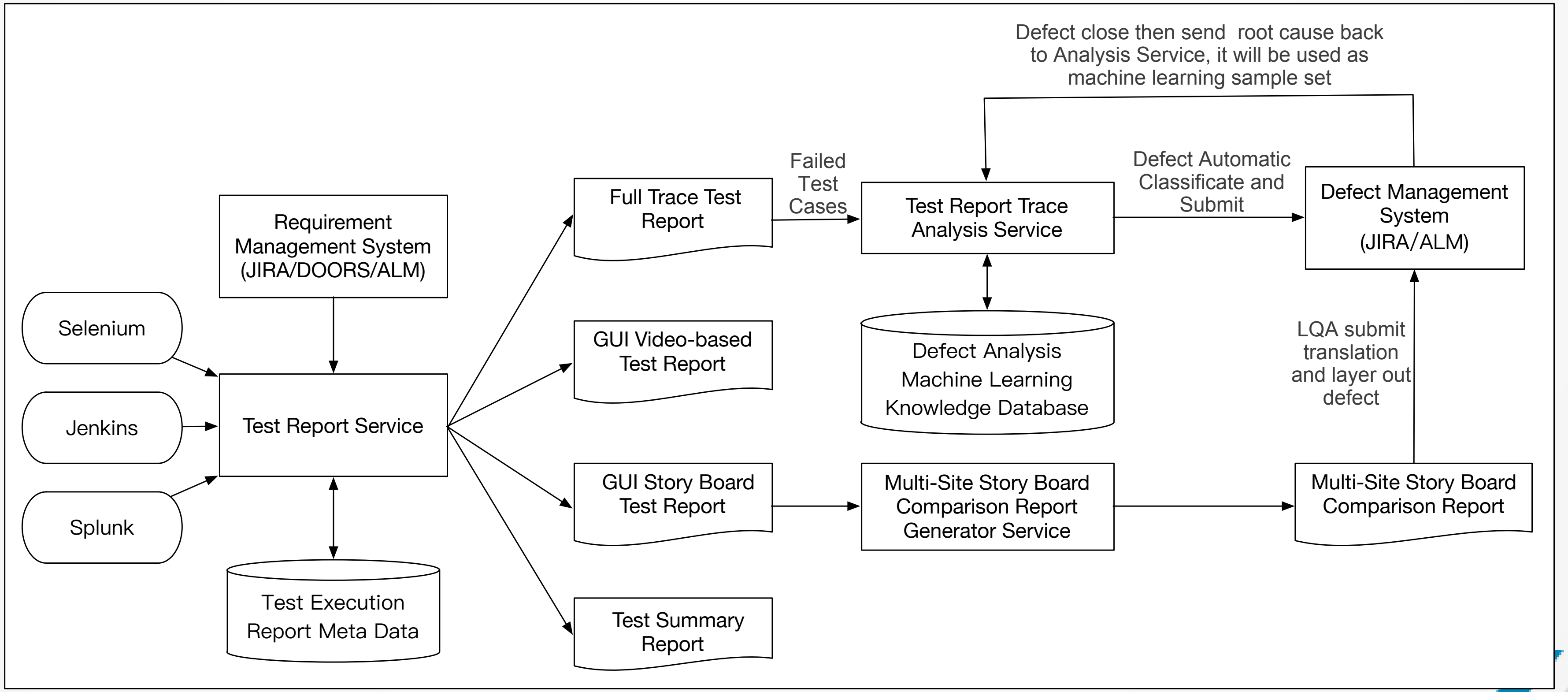

Defect close then send root cause back to Analysis Service, it will be used as machine learning sample set
Failed Test Cases
Defect Automatic Classificate and Submit
Full Trace Test Report
Test Report Trace Analysis Service
Defect Management System (JIRA/ALM)
Requirement Management System (JIRA/DOORS/ALM)
Defect Analysis Machine Learning Knowledge Database
Selenium
GUI Video-based Test Report
Jenkins
Test Report Service
GUI Story Board Test Report
Splunk
Test Execution Report Meta Data
Test Summary Report

TiD2017

测试架构的演变 - Test Report Platform

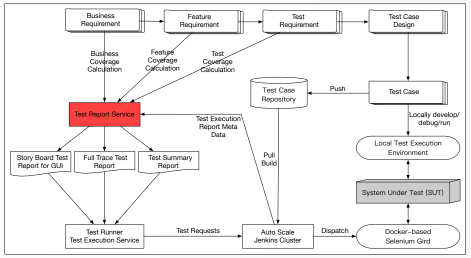

Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Business Coverage Calculation
Feature Coverage Calculation
Test Coverage Calculation
Test Case Repository
Push
Test Case
Test Report Service
Test Execution Report Meta Data
Locally develop/ debug/run
Local Test Execution Environment
Story Board Test Report for GUI
Full Trace Test Report
Test Summary Report
Pull Build
System Under Test (SUT)
Test Runner Test Execution Service
Test Requests
Auto Scale Jenkins Cluster
Dispatch
Docker–based Selenium Gird

引入Multi-Site Comparison Report提高LQA 测试效率

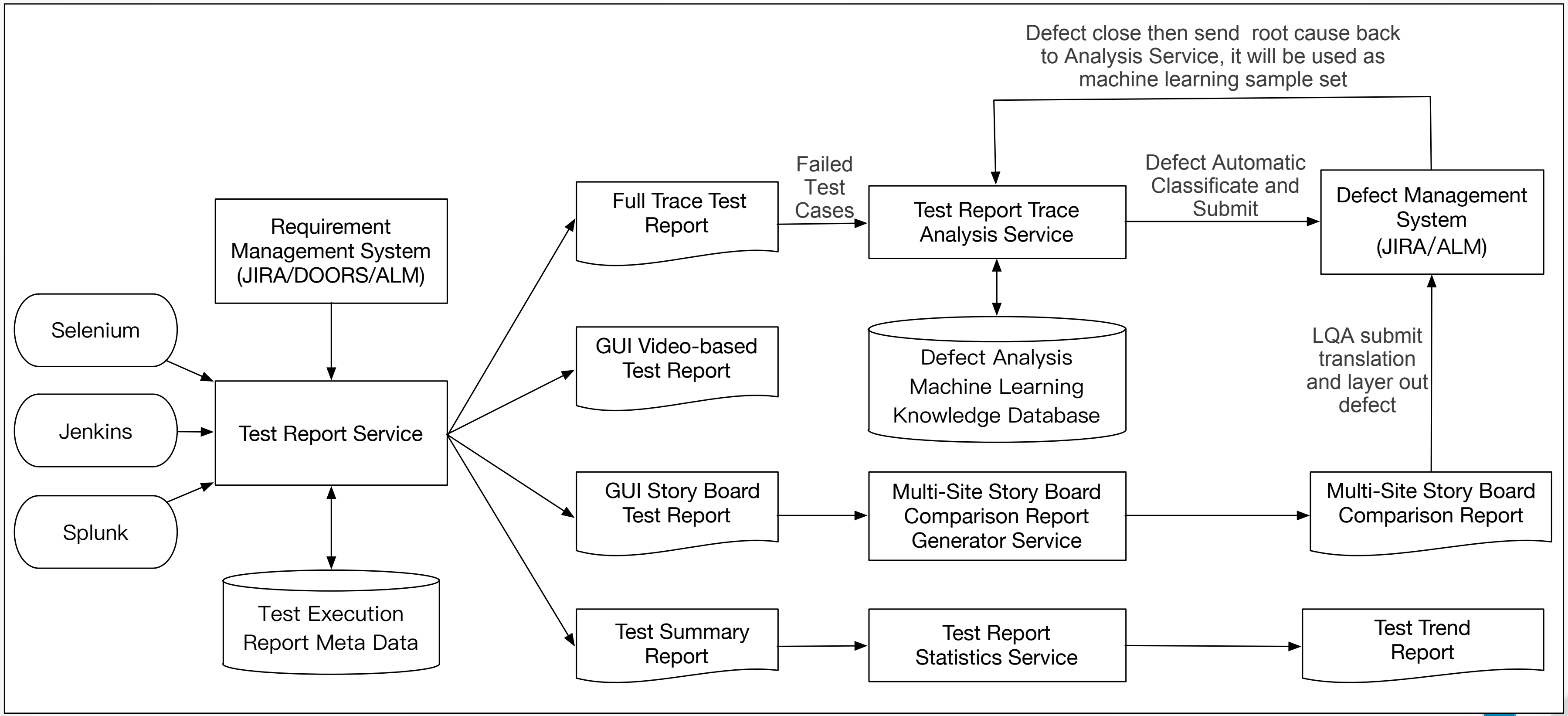

Defect close then send root cause back to Analysis Service, it will be used as machine learning sample set
Failed Test Cases
Defect Automatic Classificate and Submit
Full Trace Test Report
Test Report Trace Analysis Service
Defect Management System (JIRA/ALM)
Requirement Management System (JIRA/DOORS/ALM)
Selenium
GUI Video-based Test Report
Defect Analysis Machine Learning Knowledge Database
LQA submit translation and layer out defect
Jenkins
Test Report Service
Splunk
GUI Story Board Test Report
Multi-Site Story Board Comparison Report Generator Service
Multi-Site Story Board Comparison Report
Test Execution Report Meta Data
Test Summary Report

TiD2017

测试架构的演变 - Test Report Platform


Business Requirement
Feature Requirement
Test Requirement
Test Case Design
Business Coverage Calculation
Feature Coverage Calculation
Test Coverage Calculation
Test Case Repository
Push
Test Case
Test Report Service
Test Execution Report Meta Data
Locally develop/ debug/run
Local Test Execution Environment
Story Board Test Report for GUI
Full Trace Test Report
Test Summary Report
Pull Build
System Under Test (SUT)
Test Runner Test Execution Service
Test Requests
Auto Scale Jenkins Cluster
Dispatch
Docker-based Selenium Gird

引入Test Statistics Service


Defect close then send root cause back to Analysis Service, it will be used as machine learning sample set
Failed Test Cases
Defect Automatic Classificate and Submit
Full Trace Test Report
Test Report Trace Analysis Service
Defect Management System (JIRA/ALM)
Requirement Management System (JIRA/DOORS/ALM)
Selenium
GUI Video-based Test Report
Defect Analysis Machine Learning Knowledge Database
LQA submit translation and layer out defect
Jenkins
Test Report Service
Splunk
GUI Story Board Test Report
Multi-Site Story Board Comparison Report Generator Service
Multi-Site Story Board Comparison Report
Test Execution Report Meta Data
Test Summary Report
Test Report Statistics Service
Test Trend Report

代码级单元测试的最佳实践

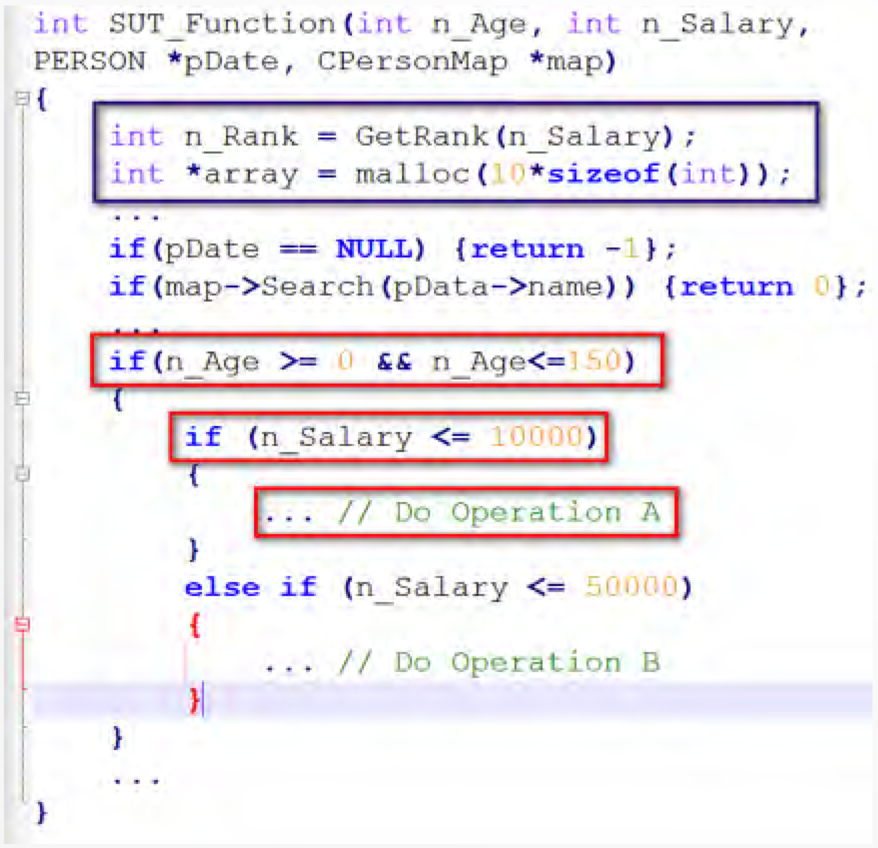
测试架构师在代码级单元测试扮演什么样的角色？

- 单元测试框架选型（**JUnit，TestNG，Spock，etc**）
- **Mock / Stub** 框架选型
- 静态扫描的规则定义与优化
- 单元测试的**CI/CD**集成
- 系统性方法排除开发的思维惯性
- 基于代码级错误分析，帮助提高开发人员代码质量意识

TiD2017

代码级单元测试的最佳实践

代码的基本特征


```
int SUT_Function(int n_Age, int n_Salary,
PERSON *pDate, CPersonMap *map)
{
    int n_Rank = GetRank(n_Salary);
    int *array = malloc(10*sizeof(int));
    ...
    if(pDate == NULL) {return -1};
    if(map->Search(pData->name)) {return 0};
    ...
    if(n_Age >= 0 && n_Age<=150)
    {
        if (n_Salary <= 10000)
        {
            ... // Do Operation A
        }
        else if (n_Salary <= 50000)
        {
            ... // Do Operation B
        }
    }
    ...
}
```

对数据进行分类处理

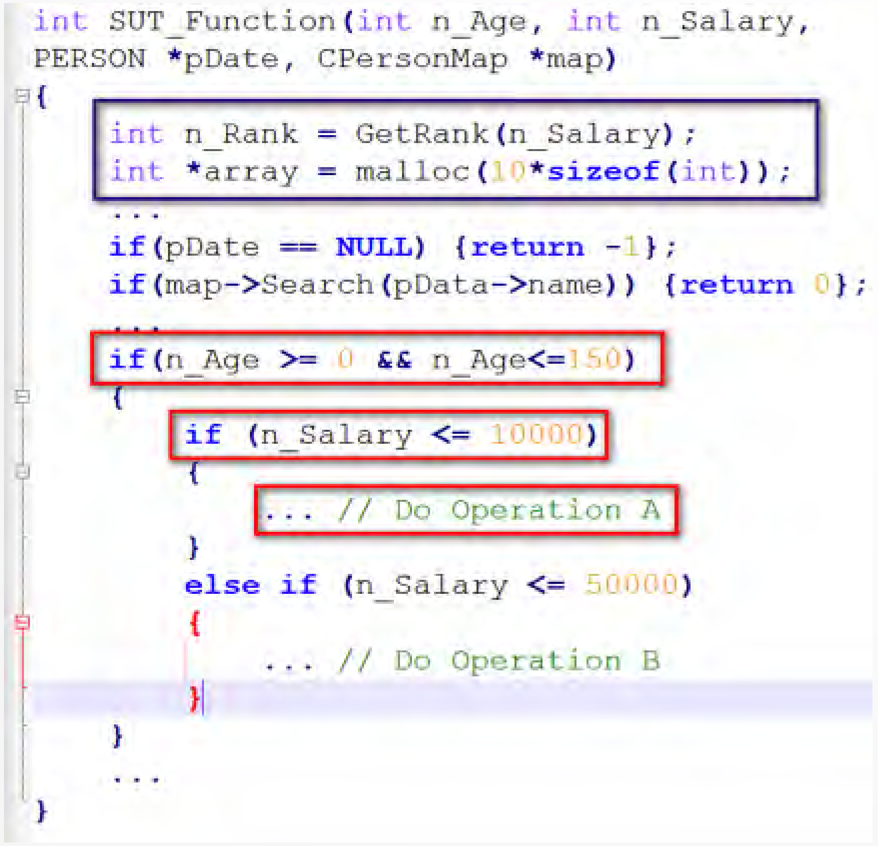
每一次条件判定，就是一次分类处理

嵌套的条件和循环，也是分类处理

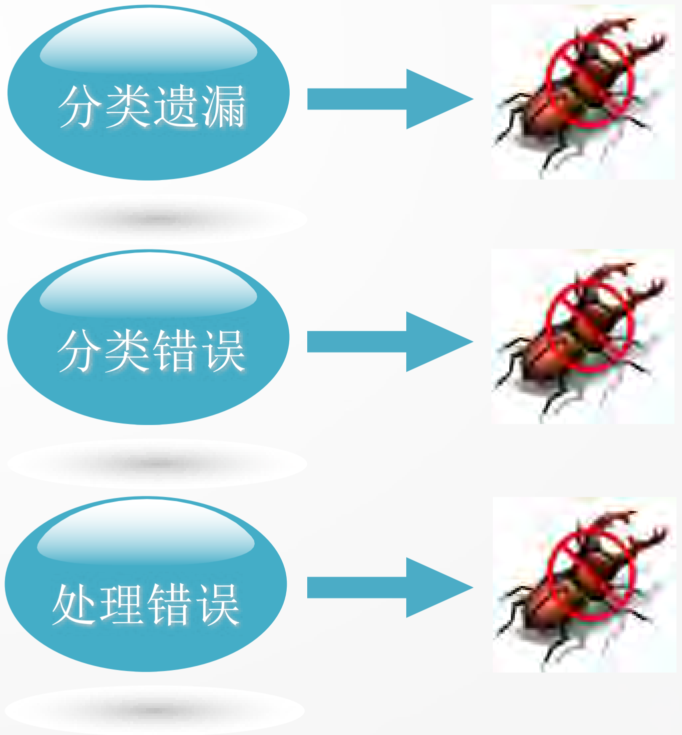
对子函数的调用获得内部输入

代码级单元测试的最佳实践

代码缺陷产生的原因



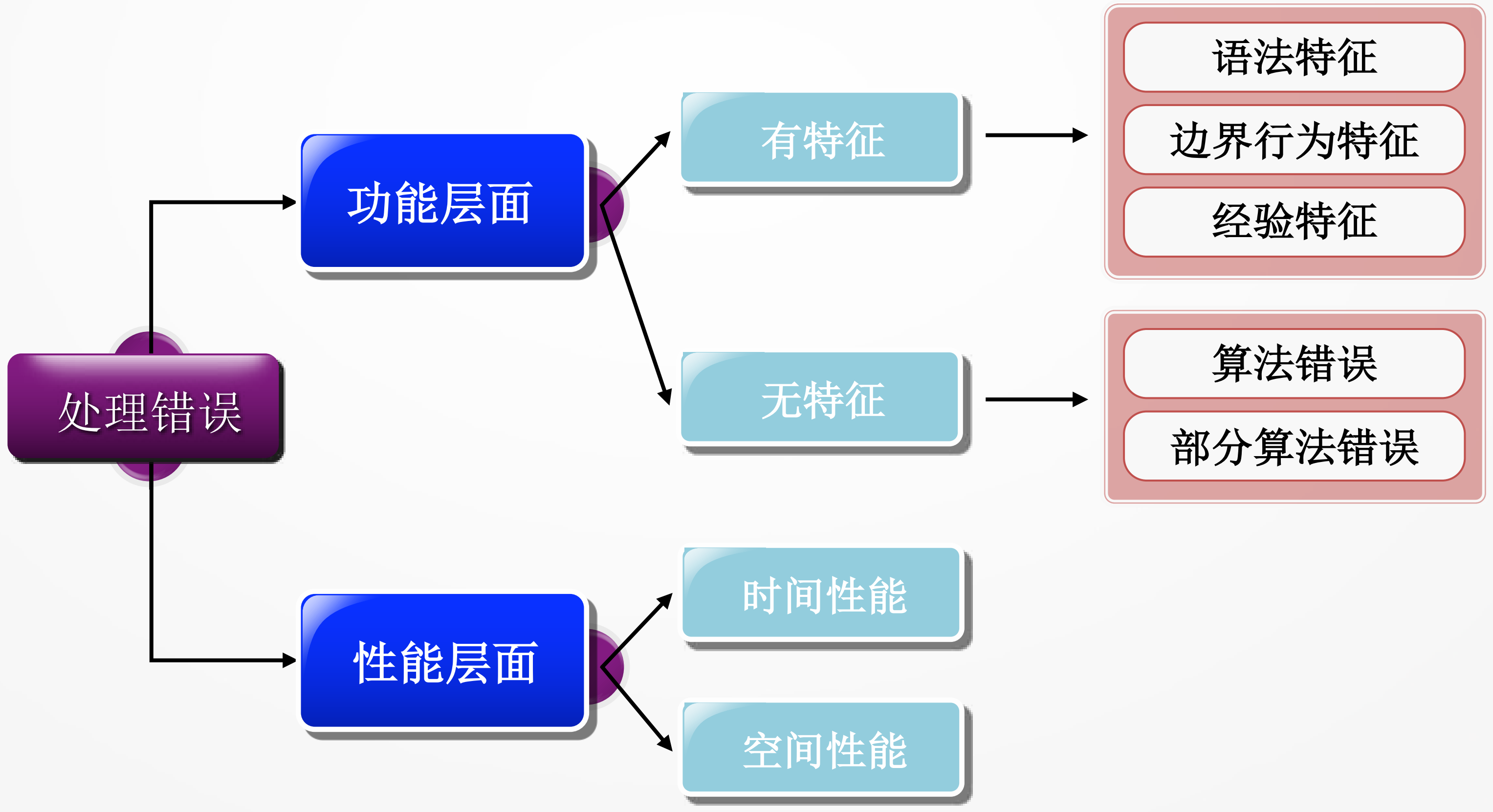
```
int SUT_Function(int n_Age, int n_Salary,
PERSON *pDate, CPersonMap *map)
{
    int n_Rank = GetRank(n_Salary);
    int *array = malloc(10*sizeof(int));
    ...
    if(pDate == NULL) {return -1};
    if(map->Search(pData->name)) {return 0};
    ...
    if(n_Age >= 0 && n_Age<=150)
    {
        if (n_Salary <= 10000)
        {
            ... // Do Operation A
        }
        else if (n_Salary <= 50000)
        {
            ... // Do Operation B
        }
    }
    ...
}
```



TiD2017

代码级单元测试的最佳实践

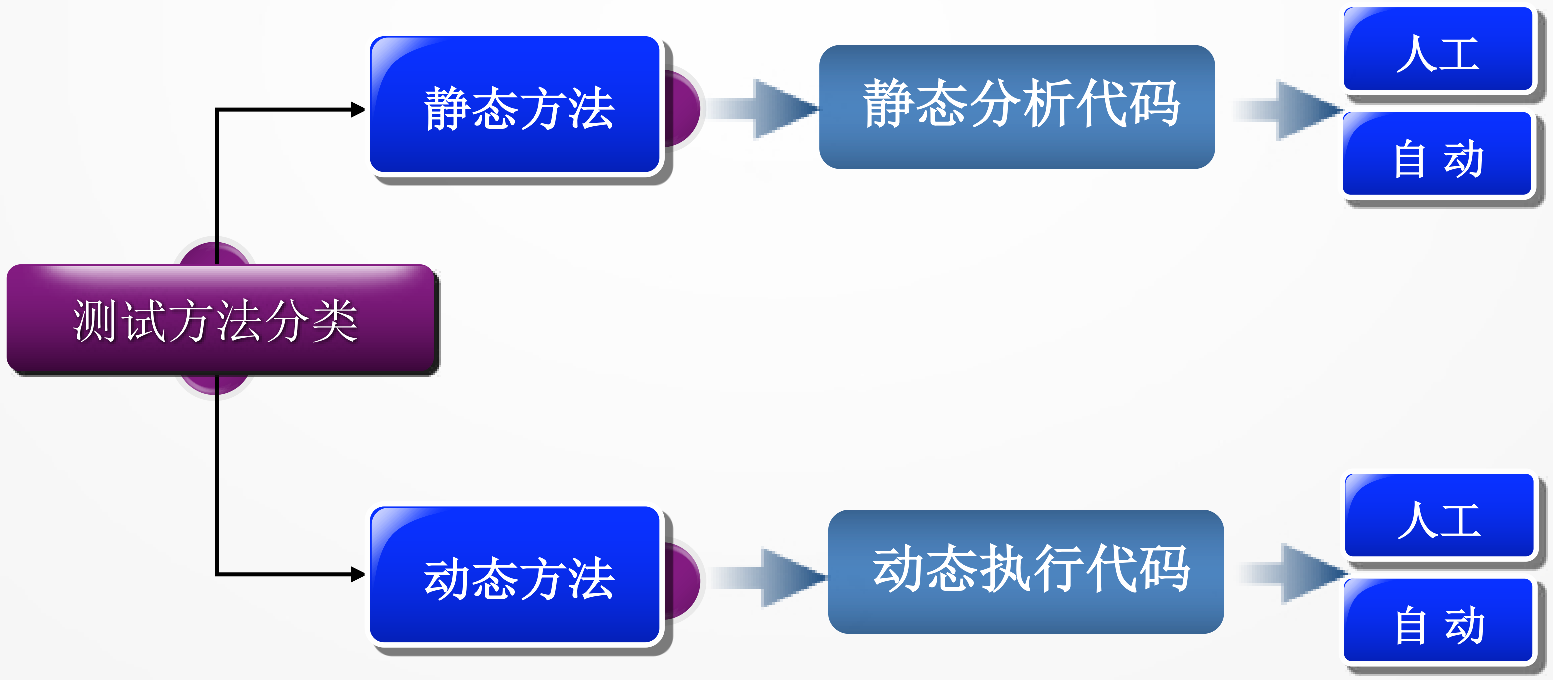
代码错误的分类



TiD2017

代码级单元测试的最佳实践

代码级测试方法的分类



TiD2017

代码级单元测试的最佳实践

代码级测试方法的分类

人工静态方法

自动静态方法

人工动态方法

自动动态方法

人工静态方法： 通过人工阅读代码来查找代码中潜在的各种错误，通常采用开发人员代码走查、结对编程、同行评审等手段。显然效率低下，效果完全取决于个人的能力。

自动静态方法： 不实际执行代码，通过扫描代码来发现语法错误，潜在语义错误和部分动态错误。这类可被发现的错误通常具有某些特征。目前广泛采用，并用于代码风格检查。

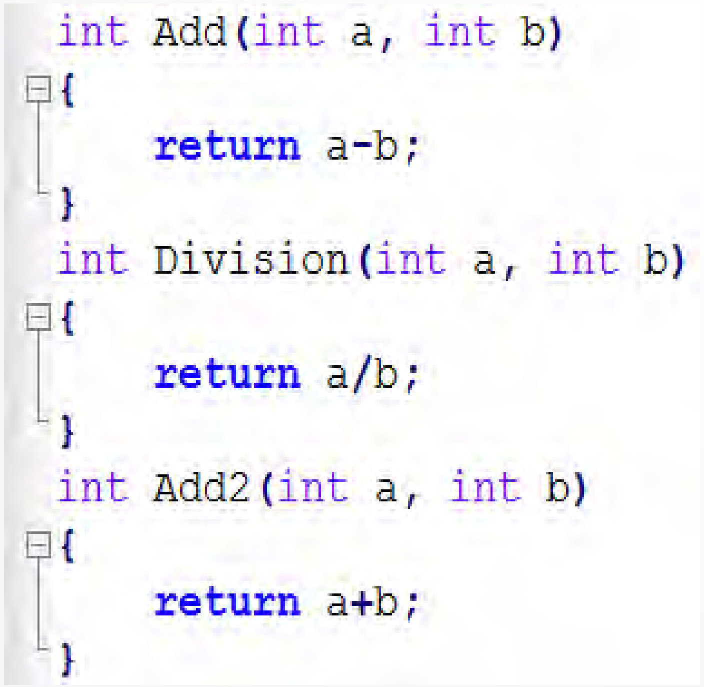
人工动态方法： 设计代码的输入和预期的正确输出的集合，执行代码，并判断实际输出是否符合预期。传统意义上的单元测试和代码级集成测试。

自动动态方法： 基于代码自动生成边界测试用例并执行来捕捉潜在的异常、崩溃和超时。只能发现行为特征错误，因为工具不可能自动了解代码所要实现的功能。

TiD2017

代码级单元测试的最佳实践

代码级测试方法示例



```
int Add(int a, int b)
{
    return a-b;
}
int Division(int a, int b)
{
    return a/b;
}
int Add2(int a, int b)
{
    return a+b;
}
```

人工静态方法	能发现
自动静态方法	没有语法错误和潜在动态错误，无法发现问题
人工动态方法	通过设定输入，执行程序并判断输出与预期的差异，可以发现问题。但是或许只是部分问题
自动动态方法	可能发现，自动采用边界测试用例来执行代码，捕捉潜在的异常、崩溃和超时

代码级单元测试的最佳实践

完备测试用例的设计

什么是好的测试用例?

- 发现了缺陷的用例就是好的用例

用例发现了缺陷是好用例，缺陷修复后，同样的用例难道就不是好用例了吗？

- 发现缺陷可能性大的用例就是好用例

如何评估用例发现缺陷的可能性？

- 发现至今未被发现的缺陷的用例就是好用例

如何评估是否还存在未被发现的缺陷？

代码级单元测试的最佳实践

完备测试用例的设计

好的测试用例 一定是一个完备的集合，能够覆盖所有等价类以及各种边界值

渔网捕鱼的例子

- 如果把被测试代码视为一个池塘，Defect视为池塘中的鱼，建立用例集的过程就像是在编织渔网
- 好的用例集就是一张能够覆盖整个池塘的渔网，只要池塘里有鱼，网就一定能把鱼捞上来。
- 如果网是完整的且合格的，那么捞不到鱼，就能证明池塘中没有鱼。
- 渔网的好坏与是池塘中否有鱼无关

代码级单元测试的最佳实践

完备测试用例的设计

“好”的测试用例集合

等价类划分的准确性

对于每个等价类都能保证只要其中一个输入测试通过，其他输入也一定测试通过

等价类集合的完备性

所有等价类都已经正确识别并且测试通过

边界值/条件处理的正确性

所有可能的边界值和边界条件都已经正确识别并且测试通过

做到了以上三点，就可以肯定测试是充分且完备的，既做到了完整的输入覆盖

TiD2017

代码级单元测试的最佳实践

代码测试覆盖率的衡量

通常采用白盒测试覆盖率来度量代码测试的充分性和完整性

覆盖率=（至少被执行了一次的条目数）/条目总数 ＊ 100%

其中条目可以是语句，条件，判定，路径等，分别对应不同种类的代码覆盖率

- 评估代码测试的充分性和完整性
- 单纯统计覆盖率意义不大，关键是帮助找出遗漏用例，有针对性的补充用例
- 识别不可达的“僵尸”代码

- 基于已存在的代码，在惯性思维作用下，与开发人员自行测试的局限性一致
- 测试成本随着覆盖率的提高而增加，越往后越难

代码级单元测试的最佳实践

探讨覆盖率的局限性

有了很高的白色覆盖率，是否就高枕无忧了呐？

比如：某项目的代码达到了MC/DC覆盖率100%

极端的例子：一个被测函数，里面只有一行代码，衡量代码质量的所有覆盖率指标都是100%

- 白盒覆盖是基于现有代码的，不能发现“未考虑某些输入”形成的缺陷。
- 满足白盒覆盖率测试的要求，并不意味整个测试已完成，而只能说明测试对象已不需要基于此技术再进行额外的测试
- “未考虑得得某些输入”通常是边界输入，会导致有特征的错误（崩溃，异常等），自动动态善于捕捉此类错误
- 不要一味追求高覆盖率，覆盖率仅仅是手段，透过现象看本质

全球化Site自动化测试的挑战

- What‘s Global Site Test ?
- Global Site Test Challenge
- Global Test Capability Key Value
- Global Test Capability Brief Introduction
- Global Test Capability Design Philosophy

全球化Site自动化测试的挑战

- Go deeper into Global Test Capability
  - ➢ Global site test base utilities
  - ➢ Global site test data utilities
  - ➢ Global Configuration Repository
  - ➢ Unified Flow Framework
  - ➢ Multi-Site Story Board Test Report
- Global Site Test Overall Architecture

What‘s Global Site Test?

www.XXX.com

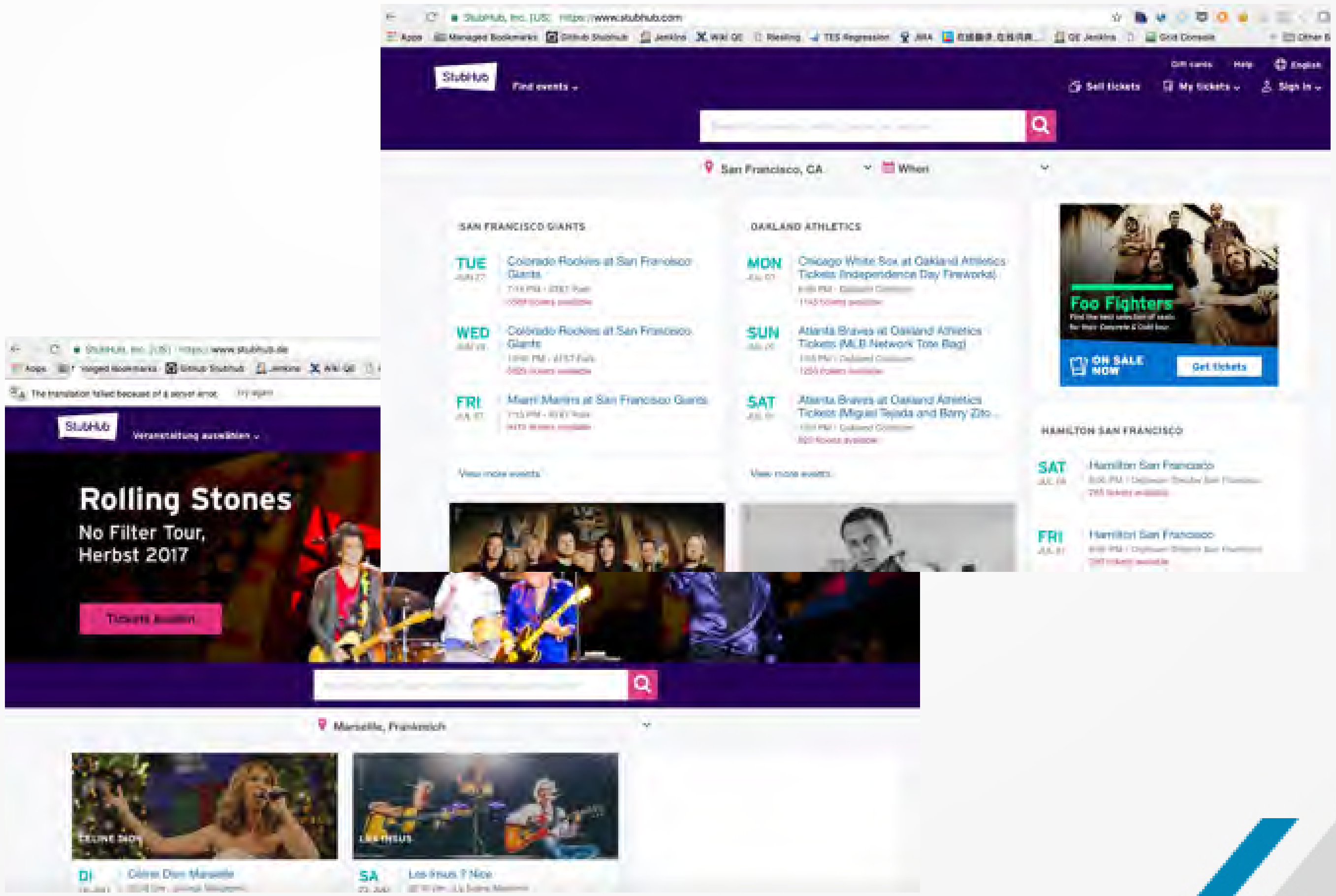
www.XXX.mx

www.XXX.uk

www.XXX.de

www.XXX.fr

www.XXX.cn



Global Site Test Challenge

- Hardcode tech debt (site, feature toggle) in test cases
- Not support different site switch during CBT test in framework level
- Not support global test data preparation
- Test report not design for LQA, need manually effort to do screenshot
- Not support validation on Currency, Date, Number and Price in different site with IBR rules
- The limitation of test execution capability

What’s our solution?

Global Test Platform

Global Test Platform Key Value

- Scalability for test case
- Highly reusable test components (Page, Flow) to combine E2E test
- Easy to achieve high test coverage
- Support CBT in test framework level
- Flexible test validation to support global feature
- Everything is configurable instead of hardcode
- Unified Flow Framework to handle flow branch
- Designed test report for global test

Global Test Platform Brief Introduction

Desktop Website
support different browser (Chrome, Firefox, Safari, IE, etc.)

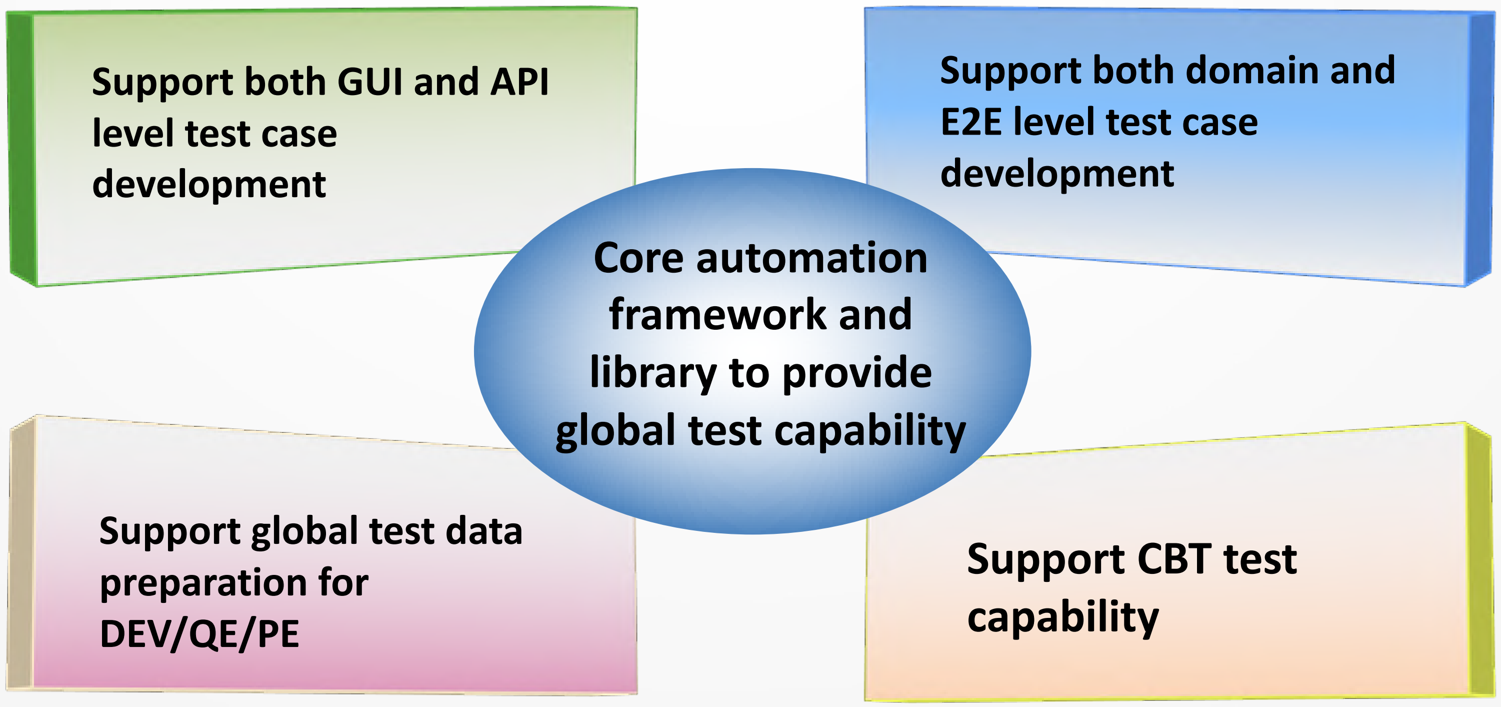
Mobile Website
support different mobile browser with same code as desktop

Base on QE-Infrastructure and inherit all core test capability

Mobile Native Application
Support both iOS and Android

Web Service
support both RESTFul and SOAP styles

Global Test Platform Brief Introduction



Core automation framework and library to provide global test capability

Support both GUI and API level test case development

Support both domain and E2E level test case development

Support global test data preparation for DEV/QE/PE

Support CBT test capability

TiD2017

Global Test Capability Design Philosophy

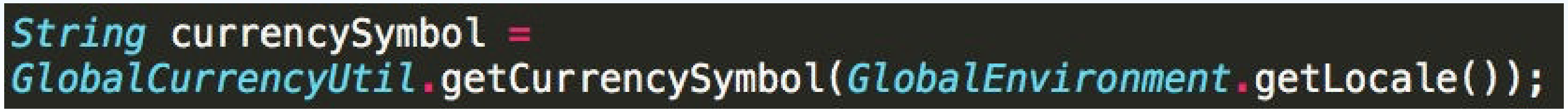
- Scalable for support new site launch test
- Configurable instead of hardcode
- Design for reusable of global test artifacts
- Easy understanding test report for linguist QA

TiD2017

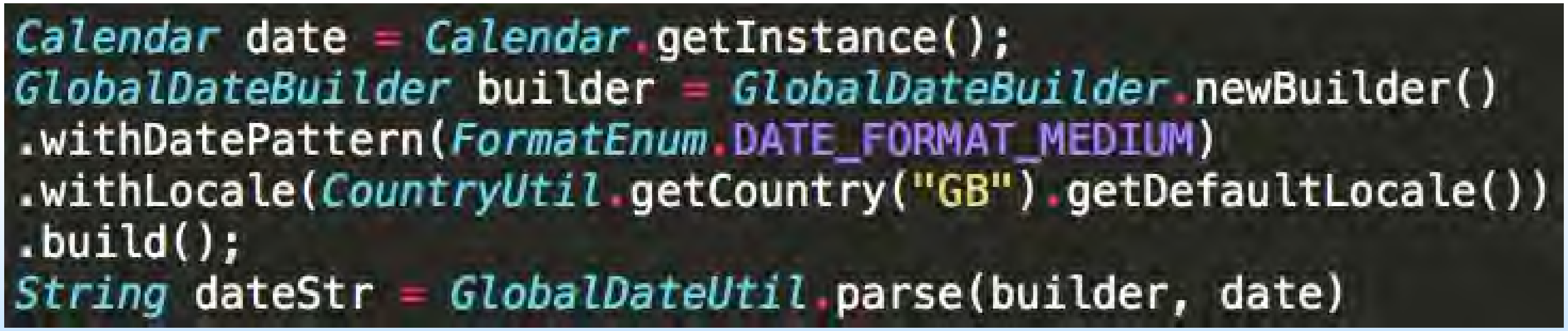
Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

- ❑ **Format Util – DateUtil, CurrencyUtil, PriceUtil, NumberUtil**
- ❑ **Environment Util – Global ENV Support Util**



```java
String currencySymbol =
GlobalCurrencyUtil.getCurrencySymbol(GlobalEnvironment.getLocale());
```



```java
Calendar date = Calendar.getInstance();
GlobalDateBuilder builder = GlobalDateBuilder.newBuilder()
.withDatePattern(FormatEnum.DATE_FORMAT_MEDIUM)
.withLocale(CountryUtil.getCountry("GB").getDefaultLocale())
.build();
String dateStr = GlobalDateUtil.parse(builder, date)
```

TiD2017

Go deeper into Global Test Capability

- Global Test Base Utilities
- Global Test Data Utilities
- Global Configuration Repository
- Unified Flow Framework
- Multi-Site Story Board Test Report

❑ **Global Base Util – SiteUtil, CountryUtil, StoreUtil**

```
int domainId = SiteUtil.getSite("UK").getDomainId();
String defaultCountry = SiteUtil.getSite("UK").getDefaultCountry();
```

```
String defaultSite = CountryUtil.getCountry("US").getDefaultSite();
List<Locale> supportedLocales = CountryUtil.getCountry("GB").getSupportedLocales();
```

❑ **CBT Util – Change site in same test case**

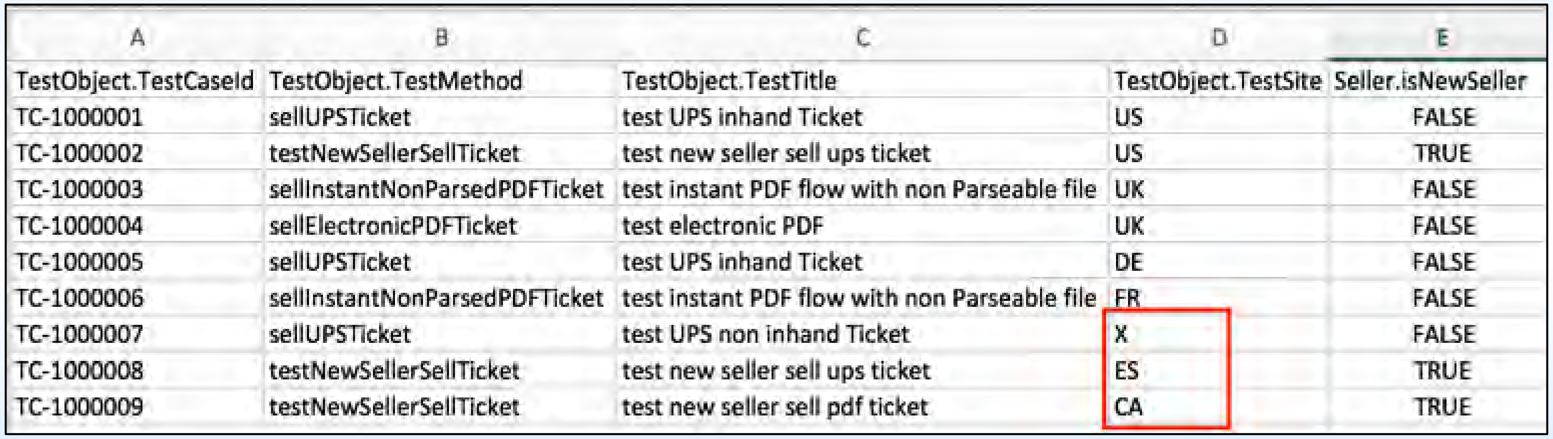
```
CBTUtil.switchSiteIfNeeded(sellerCountryCode);
```

TiD2017

Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

❑ Site X Support Util – Data-driven for new site launch



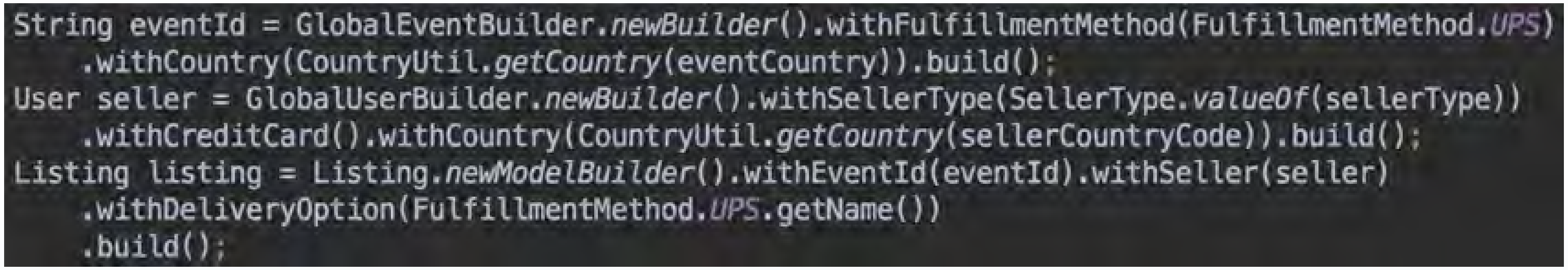
A	B	C	D	E
TestObject.TestCaseId	TestObject.TestMethod	TestObject.TestTitle	TestObject.TestSite	Seller.isNewSeller
TC-1000001	sellUPSTicket	test UPS inhand Ticket	US	FALSE
TC-1000002	testNewSellerSellTicket	test new seller sell ups ticket	US	TRUE
TC-1000003	sellInstantNonParsedPDFTicket	test instant PDF flow with non Parseable file	UK	FALSE
TC-1000004	sellElectronicPDFTicket	test electronic PDF	UK	FALSE
TC-1000005	sellUPSTicket	test UPS inhand Ticket	DE	FALSE
TC-1000006	sellInstantNonParsedPDFTicket	test instant PDF flow with non Parseable file	FR	FALSE
TC-1000007	sellUPSTicket	test UPS non inhand Ticket	X	FALSE
TC-1000008	testNewSellerSellTicket	test new seller sell ups ticket	ES	TRUE
TC-1000009	testNewSellerSellTicket	test new seller sell pdf ticket	CA	TRUE

TiD2017

Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

- ❑ **Using Global Test Data Utilities to support test data preparation for DEV/QE/PE**



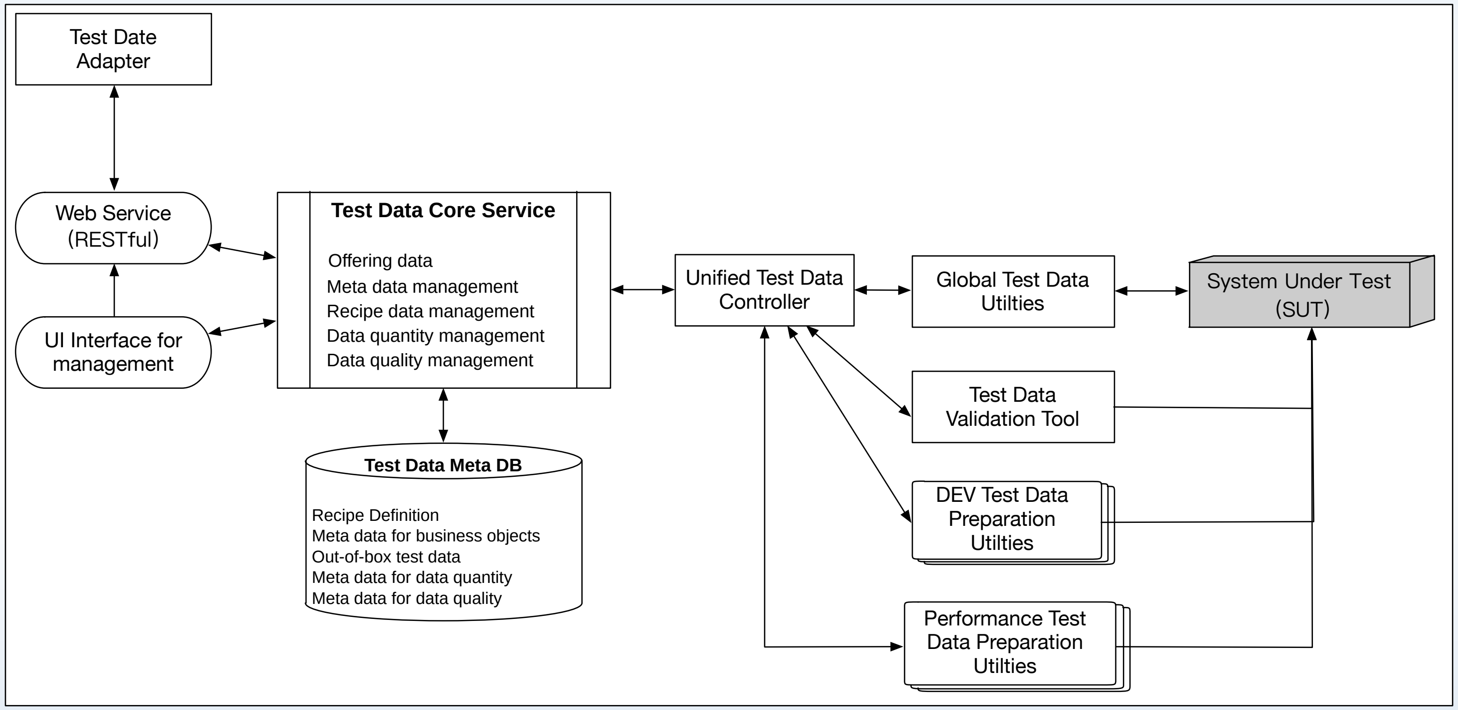
```java
String eventId = GlobalEventBuilder.newBuilder().withFulfillmentMethod(FulfillmentMethod.UPS)
    .withCountry(CountryUtil.getCountry(eventCountry)).build();
User seller = GlobalUserBuilder.newBuilder().withSellerType(SellerType.valueOf(sellerType))
    .withCreditCard().withCountry(CountryUtil.getCountry(sellerCountryCode)).build();
Listing listing = Listing.newModelBuilder().withEventId(eventId).withSeller(seller)
    .withDeliveryOption(FulfillmentMethod.UPS.getName())
    .build();
```

TiD2017

Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

❑ **Build Test Data Service base on Global Test Data Utilities to support test data preparation**


Test Date Adapter
Web Service (RESTful)
UI Interface for management
Test Data Core Service
Offering data
Meta data management
Recipe data management
Data quantity management
Data quality management
Test Data Meta DB
Recipe Definition
Meta data for business objects
Out-of-box test data
Meta data for data quantity
Meta data for data quality
Unified Test Data Controller
Global Test Data Utilties
System Under Test (SUT)
Test Data Validation Tool
DEV Test Data Preparation Utilties
Performance Test Data Preparation Utilties

TiD2017

Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

Global Registry is used to manage user signal based properties and store global/local configuration, which groups certain features together to allow users to easily turn on and off features in a particular dimension

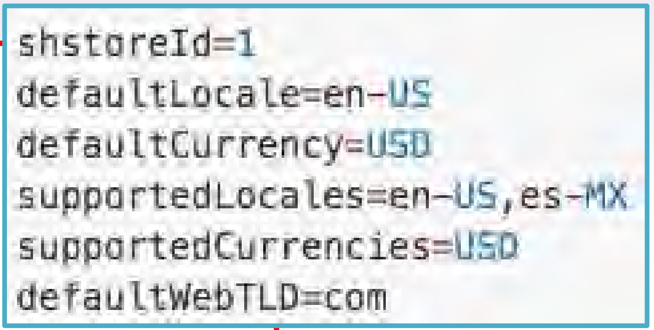
- ❑ **Decouple configure (feature toggle) from test code**
- ❑ **Multi site test with same test code**

TiD2017

Go deeper into Global Test Capability

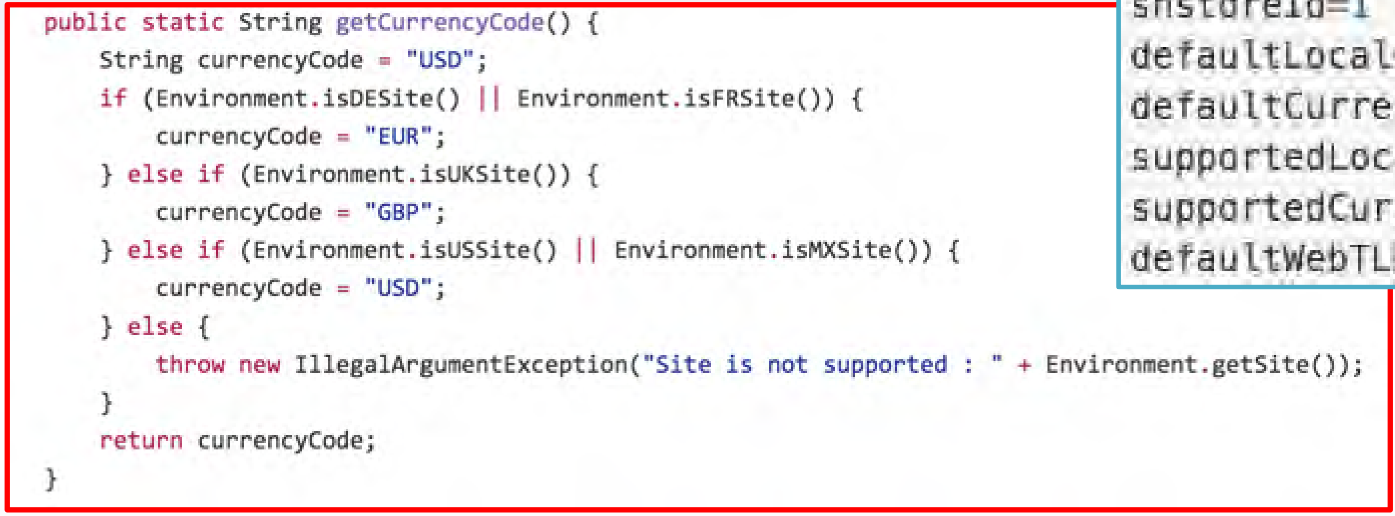
Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

Global Configuration Repository



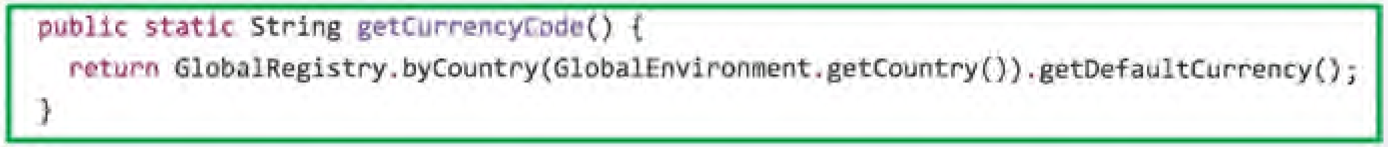
```
shstoreId=1
defaultLocale=en-US
defaultCurrency=USD
supportedLocales=en-US,es-MX
supportedCurrencies=USD
defaultWebTLD=com
```

Before



```
public static String getCurrencyCode() {
    String currencyCode = "USD";
    if (Environment.isDESite() || Environment.isFRSite()) {
        currencyCode = "EUR";
    } else if (Environment.isUKSite()) {
        currencyCode = "GBP";
    } else if (Environment.isUSSite() || Environment.isMXSite()) {
        currencyCode = "USD";
    } else {
        throw new IllegalArgumentException("Site is not supported : " + Environment.getSite());
    }
    return currencyCode;
}
```

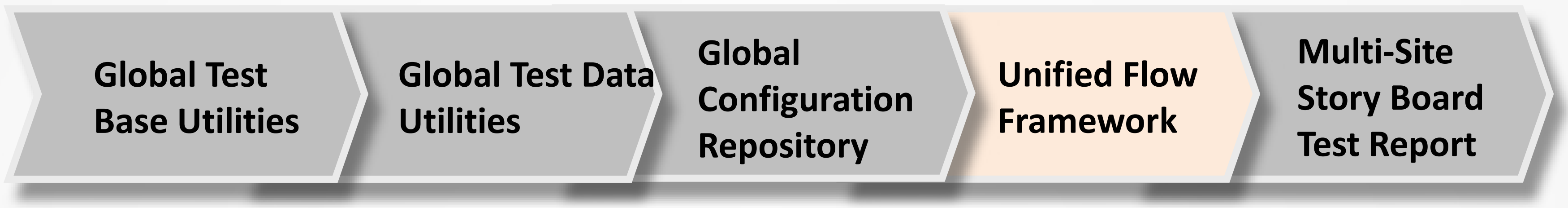
After



```
public static String getCurrencyCode() {
  return GlobalRegistry.byCountry(GlobalEnvironment.getCountry()).getDefaultCurrency();
}
```

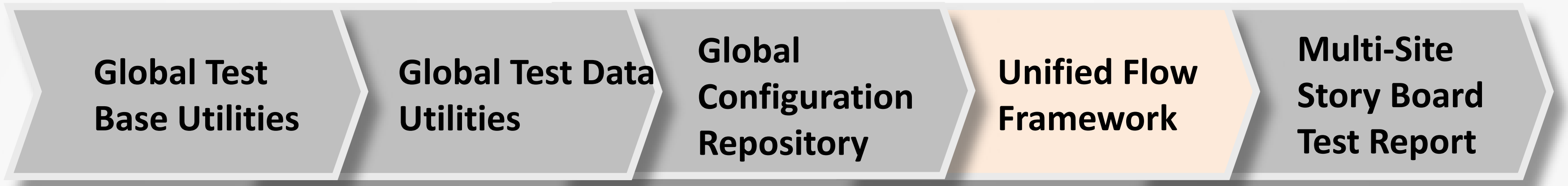
TiD2017

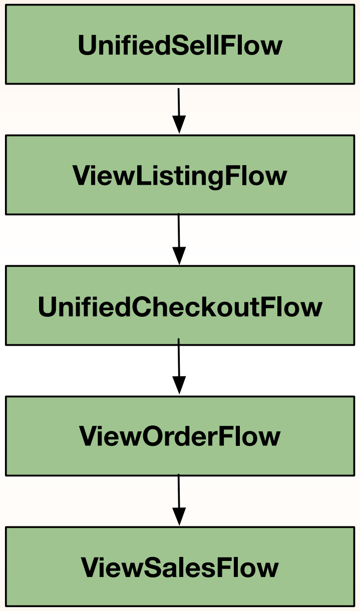
Go deeper into Global Test Capability

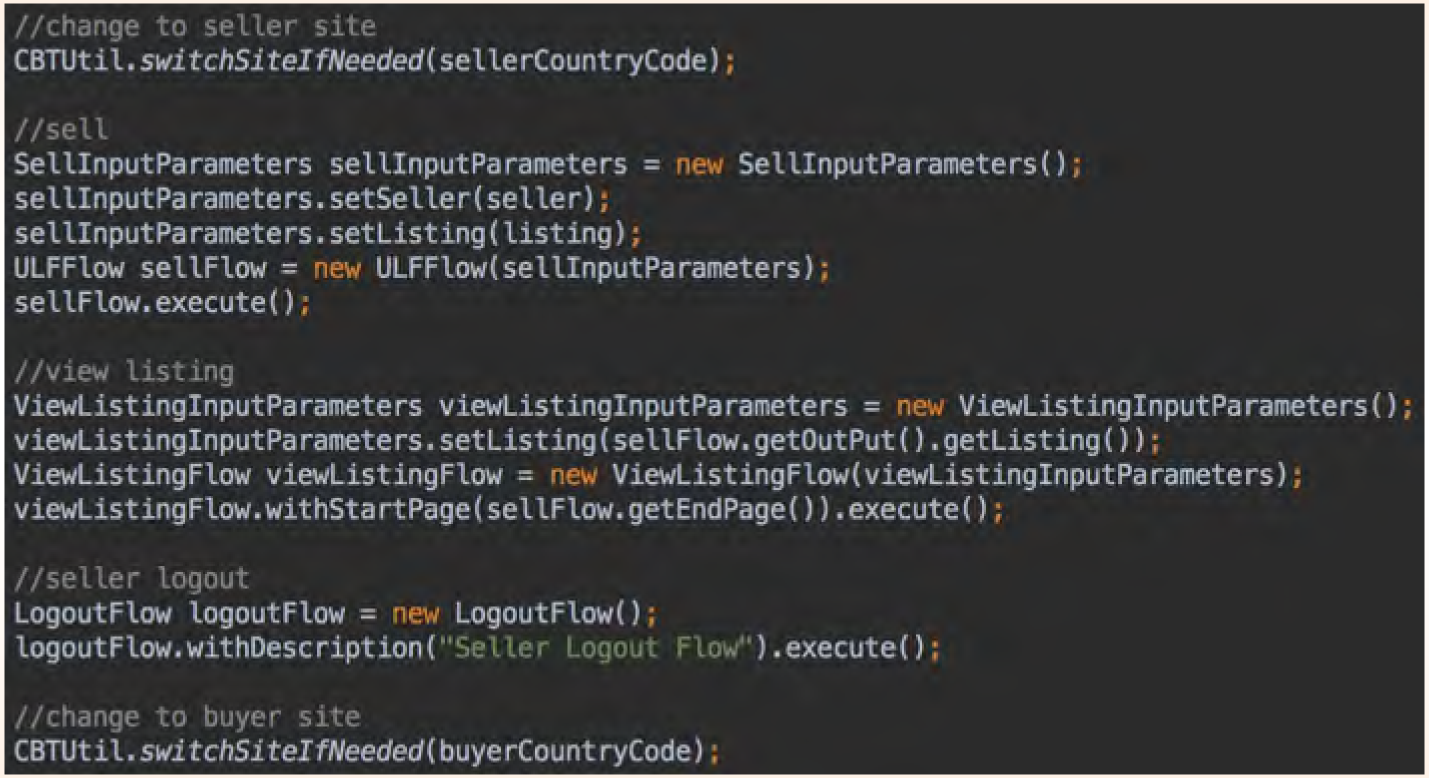

Global Test Base Utilities
Global Test Data Utilities
Global Configuration Repository
Unified Flow Framework
Multi-Site Story Board Test Report

- ❑ One flow to combine all flow branches
- ❑ Reusable Page-Flow module between domain and E2E
- ❑ Support multi entry/exit page for same flow
- ❑ Support the switch of validation ON/OFF in flow level
- ❑ Support lambda expression and chain code style
- ❑ Support customized operations before and after each Page method

Go deeper into Global Test Capability


Global Test Base Utilities
Global Test Data Utilities
Global Configuration Repository
Unified Flow Framework
Multi-Site Story Board Test Report



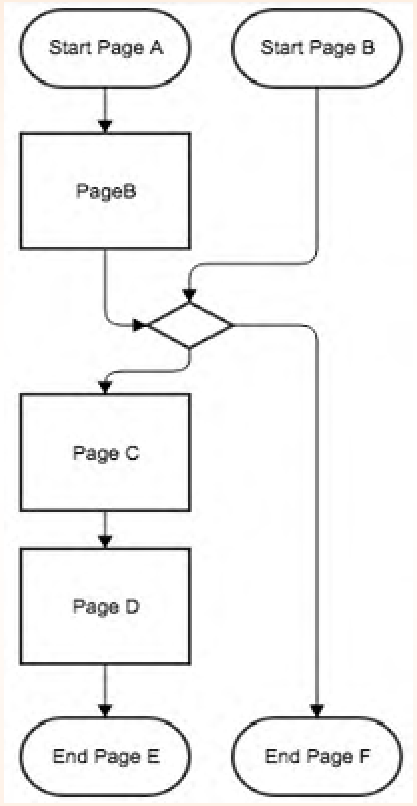


```
//change to seller site
CBTUtil.switchSiteIfNeeded(sellerCountryCode);

//sell
SellInputParameters sellInputParameters = new SellInputParameters();
sellInputParameters.setSeller(seller);
sellInputParameters.setListing(listing);
ULFFlow sellFlow = new ULFFlow(sellInputParameters);
sellFlow.execute();

//view listing
ViewListingInputParameters viewListingInputParameters = new ViewListingInputParameters();
viewListingInputParameters.setListing(sellFlow.getOutPut().getListing());
ViewListingFlow viewListingFlow = new ViewListingFlow(viewListingInputParameters);
viewListingFlow.withStartPage(sellFlow.getEndPage()).execute();

//seller logout
LogoutFlow logoutFlow = new LogoutFlow();
logoutFlow.withDescription("Seller Logout Flow").execute();

//change to buyer site
CBTUtil.switchSiteIfNeeded(buyerCountryCode);
```

TiD2017

Go deeper into Global Test Capability

- Global Test Base Utilities
- Global Test Data Utilities
- Global Configuration Repository
- Unified Flow Framework
- Multi-Site Story Board Test Report



TiD2017

Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report

Design Purpose

- ❑ Provide easy understanding test report for all stakeholders
- ❑ Improve efficiency for Linguist QA

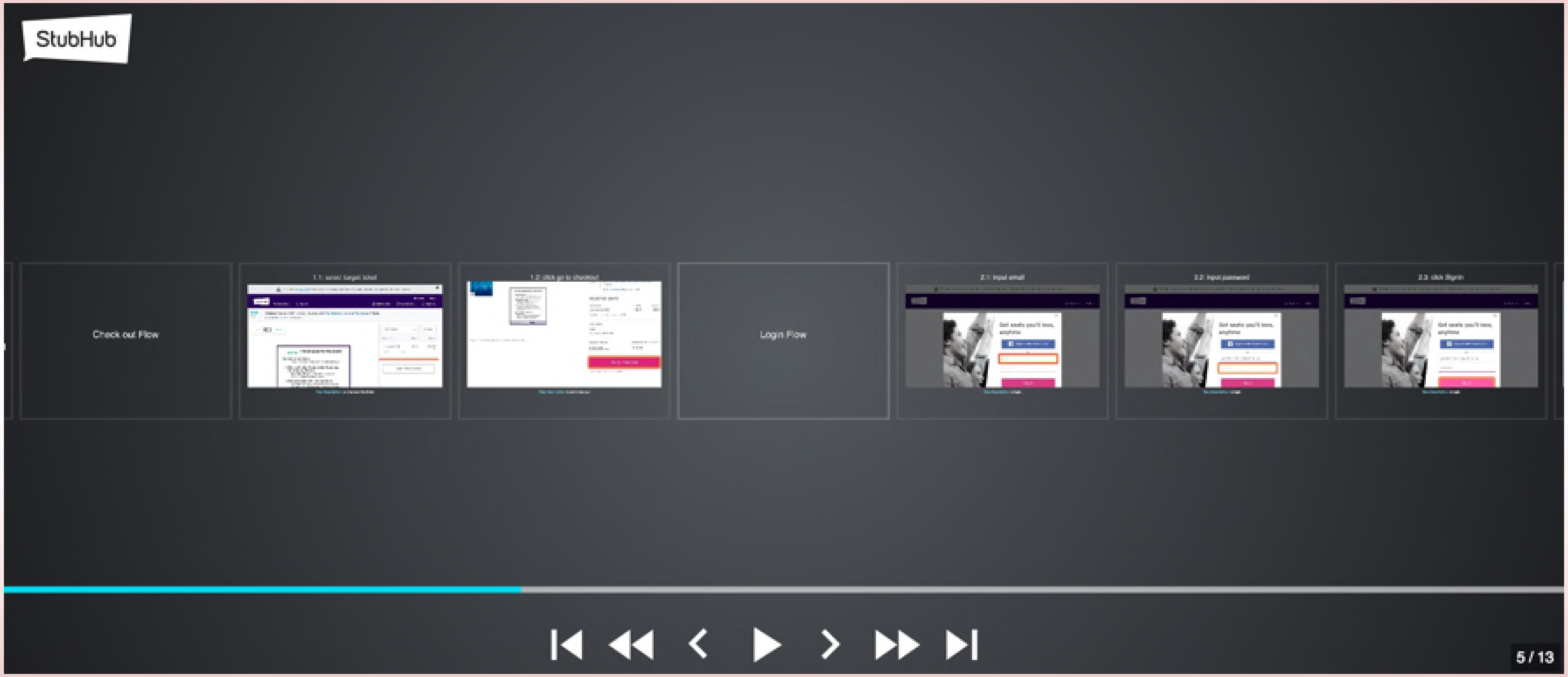
Major Feature

- ❑ Easy to go though the business flow which is made up with a list of screenshots with progress bar
- ❑ Highlight the element the current action focus on
- ❑ Show flow name, action name and description if you use Unified Flow

TiD2017

Go deeper into Global Test Capability

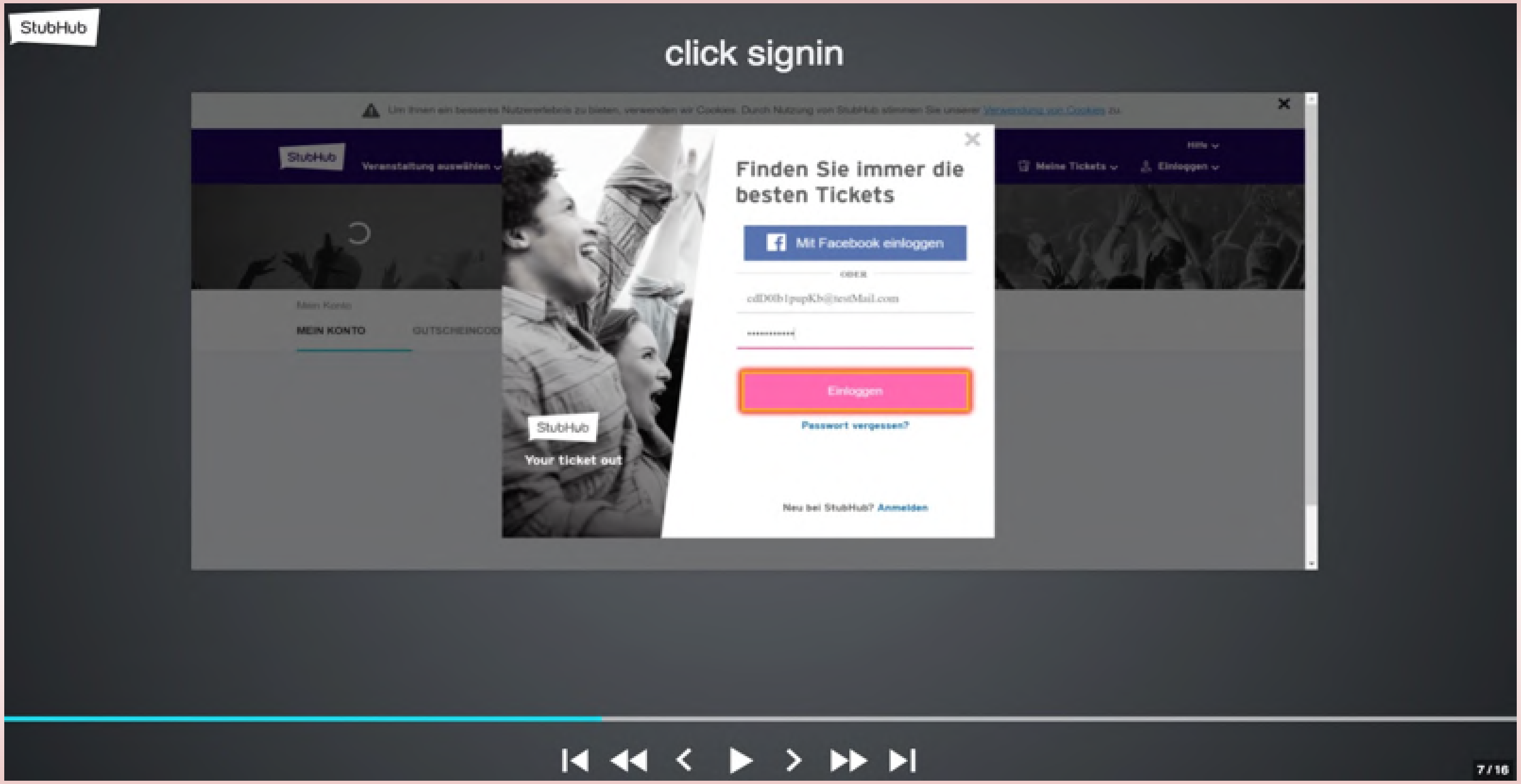
Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → **Multi-Site Story Board Test Report**



TiD2017

Go deeper into Global Test Capability

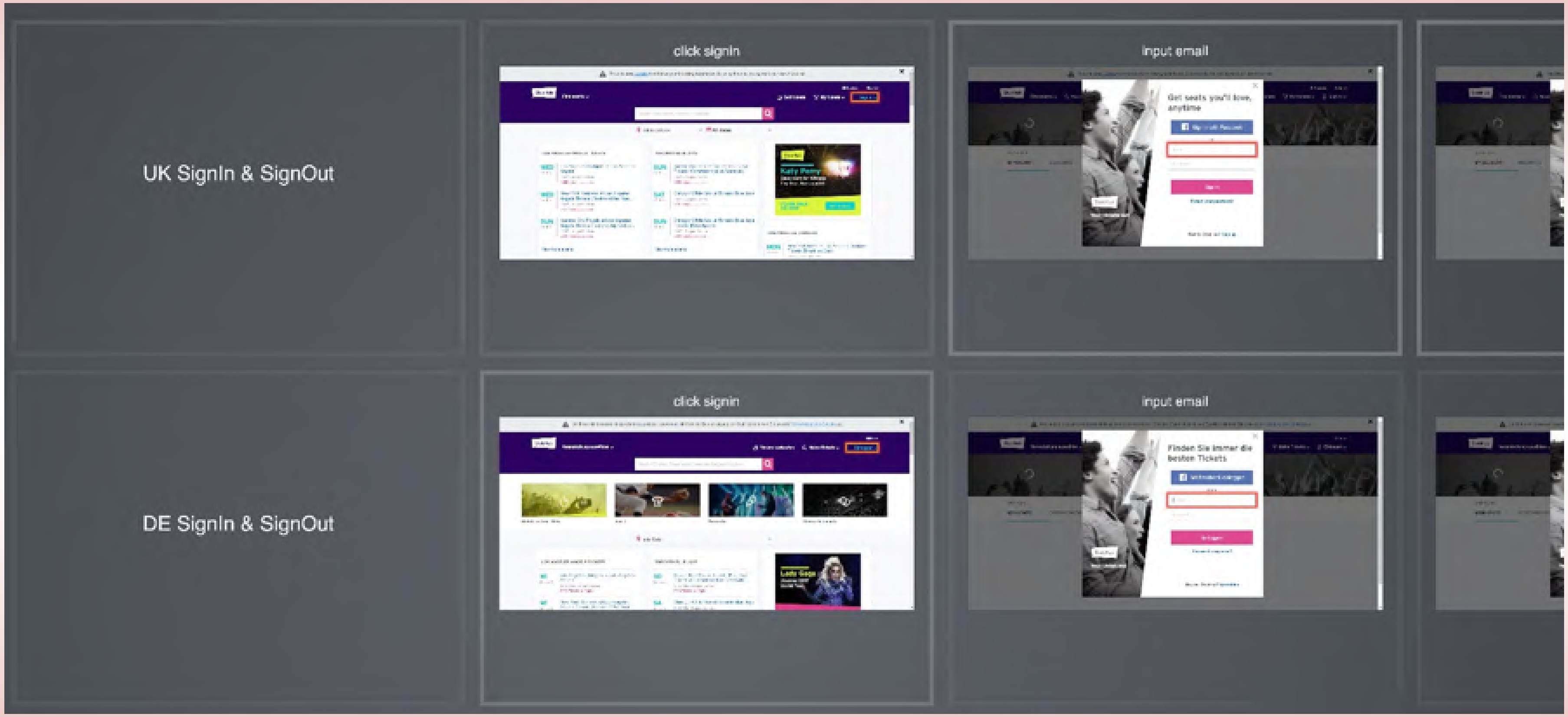
Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → **Multi-Site Story Board Test Report**



TiD2017

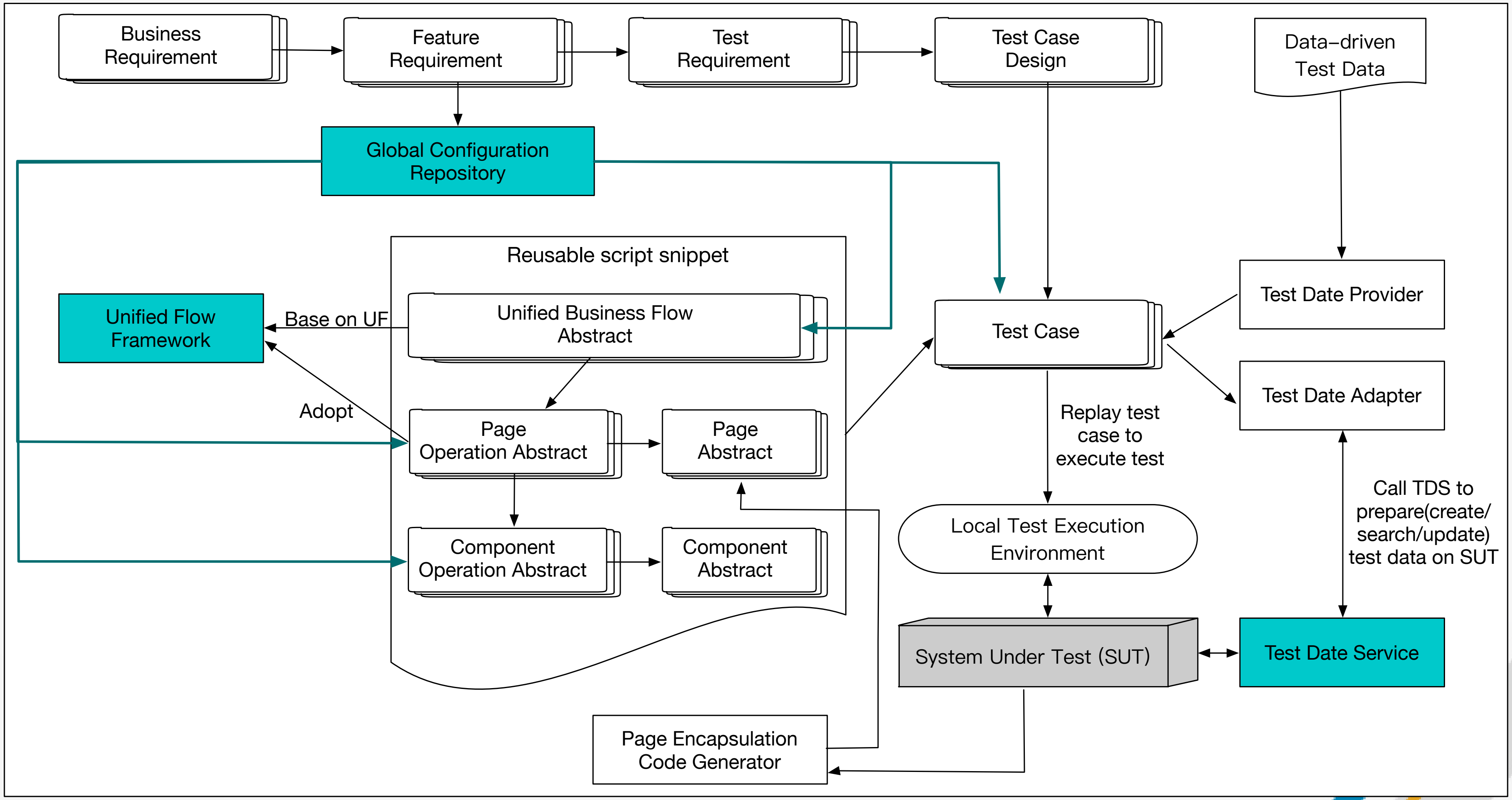
Go deeper into Global Test Capability

Global Test Base Utilities → Global Test Data Utilities → Global Configuration Repository → Unified Flow Framework → Multi-Site Story Board Test Report



TiD2017

Test Architecture - GUI Automation



TiD2017

Q & A