AiDD AI+研发数字峰会
AI+ Development Digital summit
第5届

大模型加速BDD工程化落地

高家祺 | 杭州谐云科技有限公司

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

演讲嘉宾

高家祺

谐云科技-DevOps技术总监

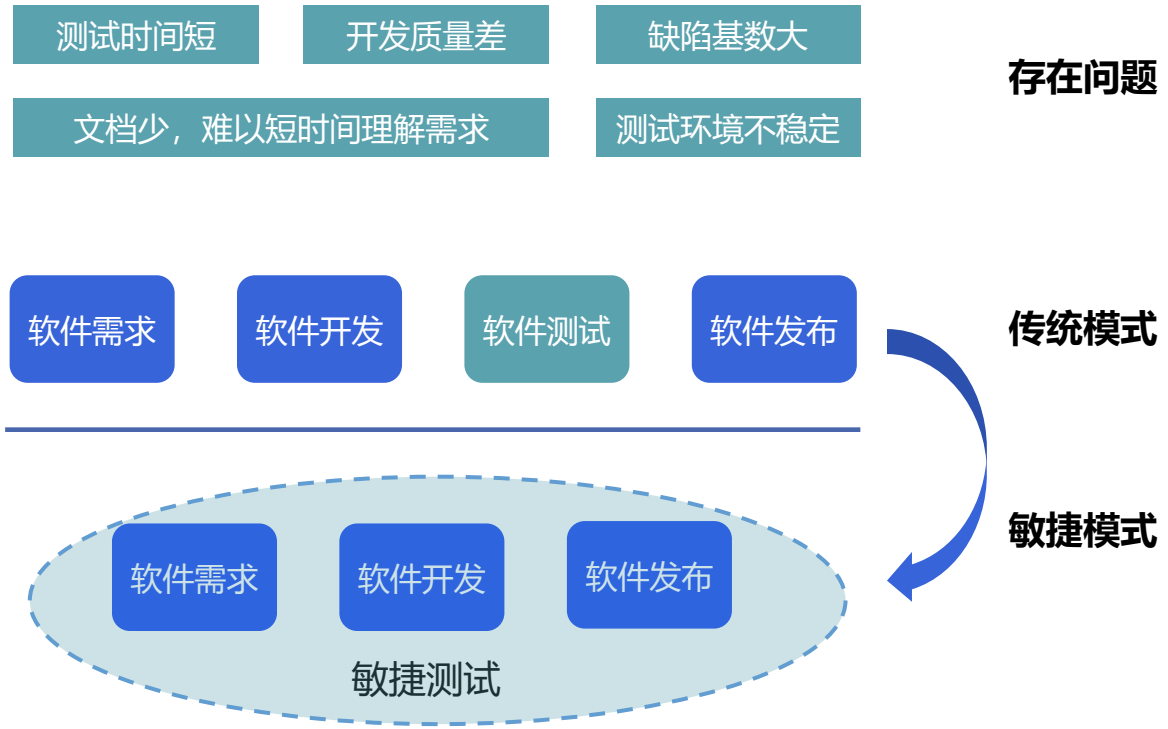
目前就职于谐云科技，主要负责DevOps产品研发、推广、落地，拥有多年大规模微服务架构、软件工程化、团队敏捷化实践经验，主导过观云台、DevOps、项目管理、API网关、微服务治理平台等产品孵化、架构升级建设，目前服务过的客户主要覆盖政府、银行、证券、能源行业，包括：应急部、香港医院管理局、杭州银行、湘财证券、国元证券、中国电力科学研究院、上海汽车、东风汽车等。

目录
CONTENTS

PART 01

BDD用户行为驱动开发模式

常见的测试模式



敏捷其实是一系列方法如XP、Scrum、Lean等总称的术语，目的是通过迭代和增量的开发， 并且经常检视和调整来提升项目的管理和交付。

传统测试

传统测试通常在软件开发的后期阶段进行，测试工作相对独立，通常由专门的测试团队负责，测试的重点在于验证软件是否符合需求设计。传统测试往往在项目完成后才发现问题，反馈周期较长，导致修复成本高。同时，传统测试的跨职能合作较少，开发和测试团队之间的沟通不够频繁，可能影响软件的质量和交付速度。

敏捷测试

敏捷测试则贯穿整个开发过程，强调早期和频繁的测试，确保软件在每个迭代周期都能快速交付可用功能。敏捷测试鼓励开发和测试人员之间的密切合作，团队成员共同参与需求讨论和设计评审。敏捷测试关注更快的定位发现软件中存在的缺陷，建立快速反馈机制，持续改进，高效解决问题。此外，敏捷测试采用轻量级文档，灵活应对变化，更加适应快速发展的软件项目环境。

敏捷测试宣言

测试是一个活动 胜于 测试是一个阶段

Testing is an Activity Over Testing is a phase

预防缺陷 胜于 发现缺陷

Pervent Bugs Over Finding Bugs

做测试者 胜于 做检查者

Be a tester Over Be a checker

帮助构建最好的系统 胜于 破坏系统

Helping to build the BEST system Over Breaking the system

团队为质量负责 胜于 测试为质量负责

Whole team takes responsibility for quality Over

Tester is responsible for quality

敏捷测试不仅仅是在开发后做功能验证，它应该贯穿整个开发过程，帮助我们更早地识别问题，并尽早修复。

——Martin Fowler

敏捷方法要求团队能够持续提供高质量的软件，敏捷测试是确保这个目标实现的关键。

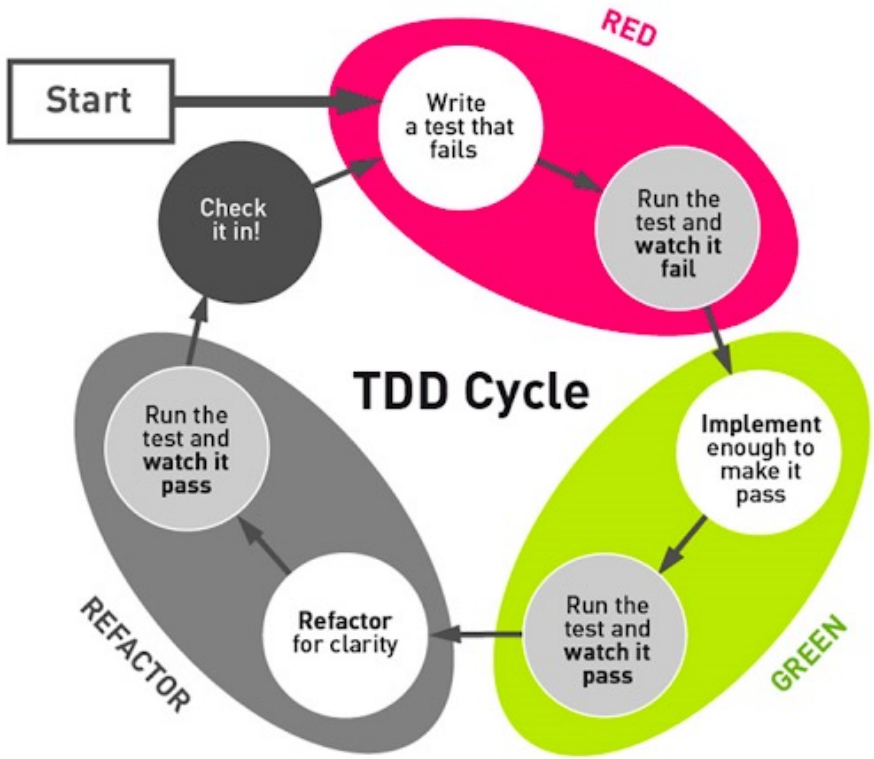
——Ken Schwaber

在敏捷中，测试与开发是交替进行的，测试并不是在完成开发后才开始，而是随时伴随开发的进展。

——Mike Cohn

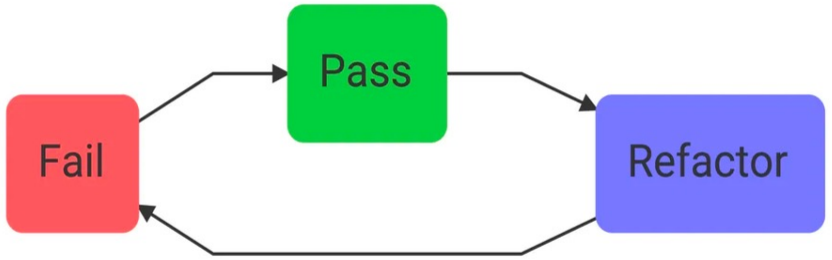
TDD(测试驱动开发)

测试驱动开发（TDD）是一个软件开发过程，其中开发人员在实现代码之前，首先构建并执行测试，从而确保代码符合预期功能；TDD的意义在于提高代码质量、确保功能符合需求、减少缺陷，并促使开发人员更清晰地理解需求，从而提升开发效率和维护性。



测试驱动开发的工作模式

- 首先创建测试用例，进行测试用例评审，达成一致约定。
- 运行测试代码，预期执行失败
- 编写实现的代码，不断调试代码，使测试脚本执行通过。
- 重构将代码调试到最佳(重构是在不改变其功能的情况下改进代码结构以提升质量)



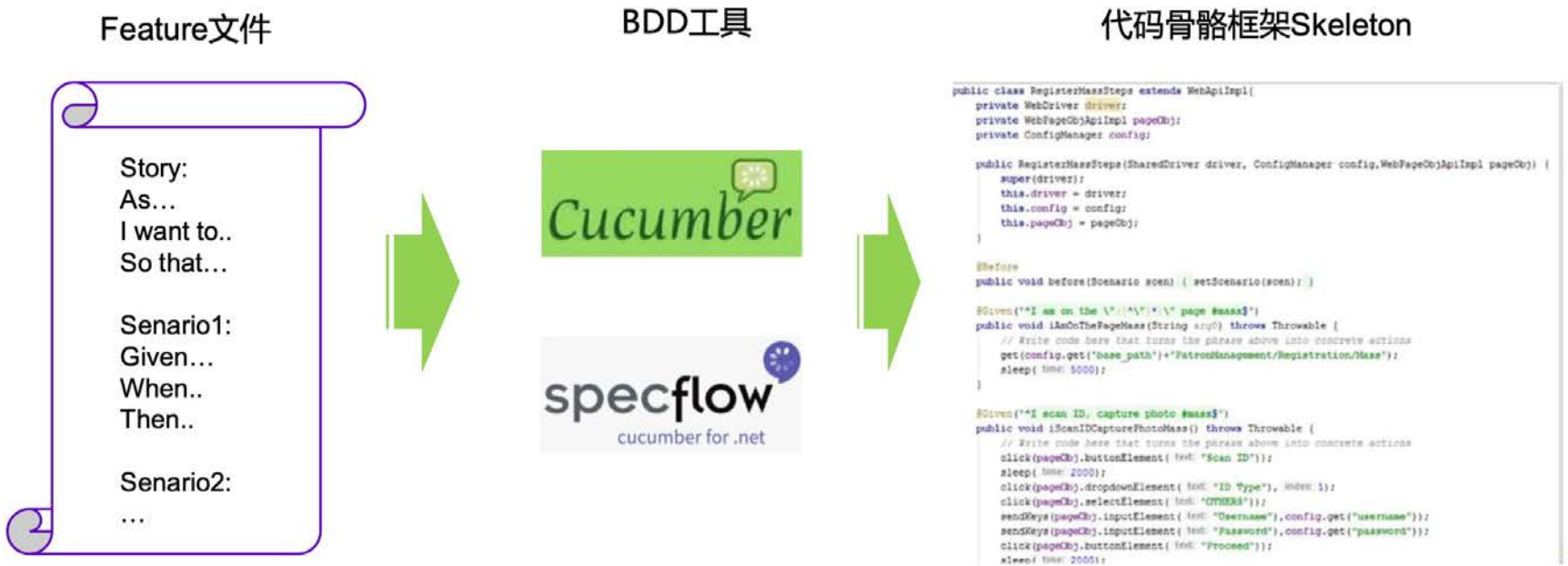
TDD存在的问题：

- 虽然测试驱动开发过程，但是更像测试单打独斗，然而业务、开发都未能参与进去，无法拉齐整个团队
- TDD侧重于单元测试，可能难以全面覆盖复杂的业务逻辑或系统集成场景
- 随着代码的演变和重构，测试用例可能需要频繁更新，维护成本高

BDD(行为驱动开发)

行为驱动开发(BDD) 是一套软件工程实践来帮助团队构建和交付更有价值、更高质量的软件。它借鉴了敏捷和精益实践，特别是测试驱动开发 (TDD)和领域驱动设计(DDD)。但最重要的是，BDD提供了一个基于简单的、结构的语言表达方式来描述需求，促进项目组与业务方之间的沟通。

BDD的自动化测试脚本代码骨骼生成过程



BDD(行为驱动开发)

BDD是测试驱动开发的延伸，通过用户故事作为使业务、开发、测试沟通的语言创建任务条目，每个条目对应一个用户行为，天然具备**可使用、可测试、可开发**的特点。

BDD的描述语言

作为什么用户角色(用户画像)
想要完成什么活动(需求场景)，以便于……（实现价值）

通过如何的操作步骤，可以做到什么。(可验证)
预期验收点：1、2、3、

BDD的优点

- 需求具有明确的应用场景，避免脱离场景的伪需求
- 促进业务、开发、测试充分沟通，一致需求语言
- 需求充分考虑用户使用的异常情景
- 需求边界清晰，易于评估工作量
- 统一沟通语言，减少浪费，节省沟通成本

对比BDD与TDD

特性	BDD（行为驱动开发）	TDD（测试驱动开发）
目的	强调开发者、测试人员和业务相关者之间的协作，定义和测试软件行为。通过共享理解，提升团队的整体效率。	侧重于创建自动化测试，确保软件代码满足指定的要求。主要目标为代码验证。
语言	使用自然语言描述软件行为，通常通过用户故事或场景进行描述，易于跨职能团队理解，特别是非技术人员。	使用编程语言编写每个代码单元的测试，通常需要开发人员专门的技术能力。
范围	涵盖端到端测试，包括UI、API和数据库测试，确保从用户需求到技术实现的全方位覆盖，增强产品的可用性。	主要聚焦于单个代码单元的测试，更侧重于代码级别的验证。
执行	通常使用专门的BDD测试框架，如Cucumber或Behave。通过框架与自然语言结合，促进跨团队协作，增强沟通效果。	通常使用特定编程语言的测试框架，如Java的JUnit或PHP的PHPUnit，较少强调团队间的沟通。
结果	在所有相关方之间达成共享的软件行为理解，促进开发人员测试人员与业务人员的持续合作，减少需求误解和功能偏差。	确保代码经过彻底测试并满足指定的要求，但可能不会总是促进软件行为的共享理解，缺少对业务需求的直接反馈。
优势	1. 跨职能协作：通过自然语言和示例驱动的测试，促进开发、测试和业务人员之间的紧密合作。 2. 需求验证：能够早期捕捉到需求误解，并迅速反馈，降低开发过程中的偏差。 3. 共享理解：有助于所有团队成员（包括非技术人员）理解软件行为，提高开发的准确性和效率。	1. 代码级验证：聚焦单元级测试，确保代码按需求工作。 2. 实现细节验证：对于精确的代码实现和逻辑验证有优势，但对于功能全局或用户需求验证较弱。

BDD把需求和技术实现紧密结合，创造了一种能让所有利益相关者都能理解的软件开发方式。

BDD使团队能够在一个共享的视角下进行工作，所有成员，包括非技术人员，都能清楚了解产品的行为和预期。

BDD帮助我们通过更具表达力的示例和场景来描述需求，让每个团队成员都能理解，并能在开发的每个阶段进行有效验证。

BDD——产品、开发、测试联动

产品/业务

•与业务沟通将业务需求转换为用户故事
•定义用户故事与开发、测试沟通
•按用户故事验证需求

User Stroy

开发

•根据用户故事拆分Feature功能、Scenario场景
•根据 Feature 文件中的场景，编写相应的代码实现，确保功能的正确性和完整性
•维护功能点知识库，包括接口和页面元素

Feature

测试

•审核用户故事：确保用户故事的完整性和准确性，确认需求清晰且可转化为具体场景。
•编写测试脚本：根据Scenario编写Steps测试脚本事项对Scenario的功能性验
•准备测试数据

Test Steps

BDD的三元素

User Story-用户故事

- 定义：用户故事是对用户需求的简洁描述，通常以“作为[角色]，我想要[目标]，以便[收益]”的格式书写。
- 角色：用户故事是BDD的起点，帮助团队理解用户的需求和期望。

Feature-特性

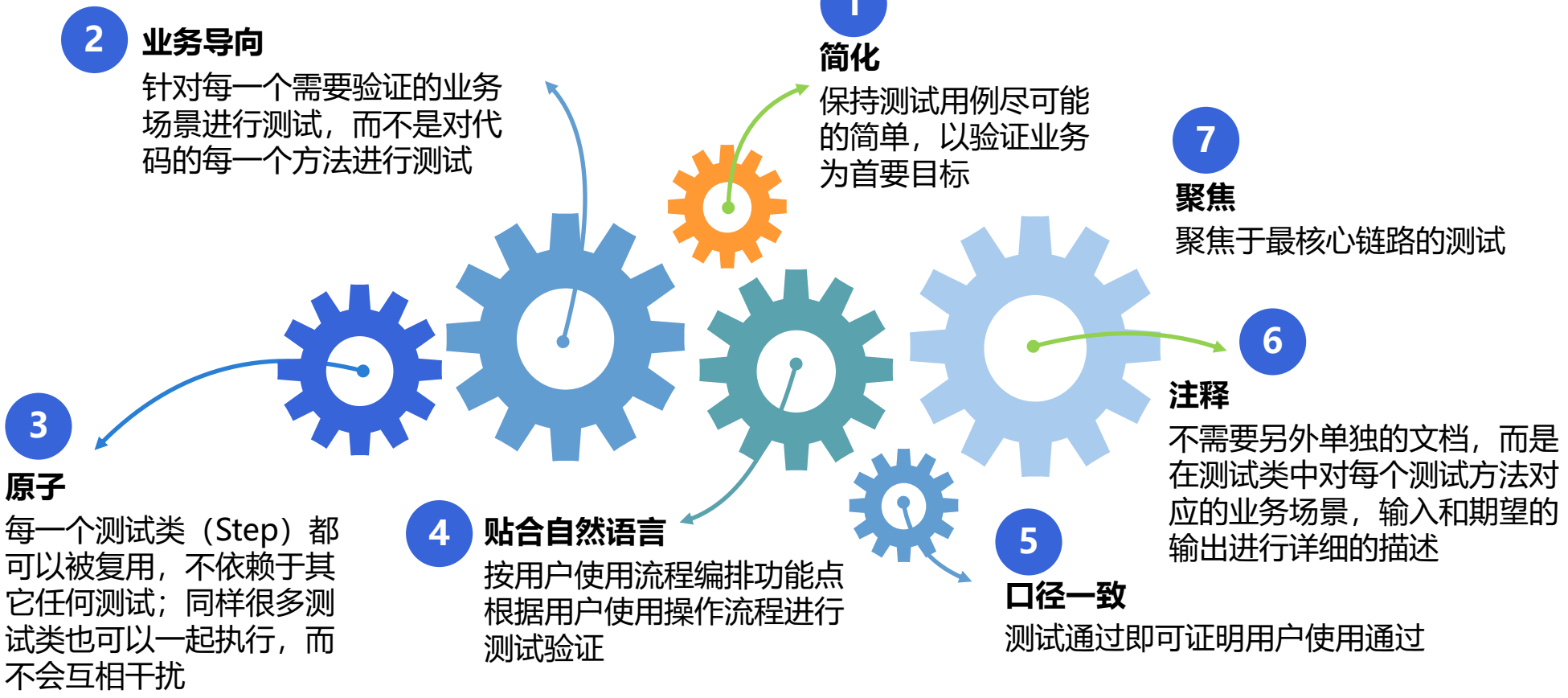
- 定义：特性是一个或多个相关用户故事的集合，代表软件的一个功能或模块。
- 关系：每个特性应对应多个用户故事，明确该功能的整体目标和价值。

Scenario-场景

- 定义：场景是对特性实现的具体描述，通常使用Gherkin语法编写，具体展示用户如何与系统交互。
- 关系：每个场景是特性的具体实现示例，涵盖特定的业务逻辑和预期结果，帮助团队验证特性是否符合用户需求。

BDD的优势

敏捷测试中通过BDD的模式描述测试场景，由测试驱动开发，提升研发质量和效率，聚焦业务目标。



1 **简化**

保持测试用例尽可能的简单，以验证业务为首要目标

2 **业务导向**

针对每一个需要验证的业务场景进行测试，而不是对代码的每一个方法进行测试

3 **原子**

每一个测试类（Step）都可以被复用，不依赖于其它任何测试；同样很多测试类也可以一起执行，而不会互相干扰

4 **贴合自然语言**

按用户使用流程编排功能点根据用户使用操作流程进行测试验证

5 **口径一致**

测试通过即可证明用户使用通过

6 **注释**

不需要另外单独的文档，而是在测试类中对每个测试方法对应的业务场景，输入和期望的输出进行详细的描述

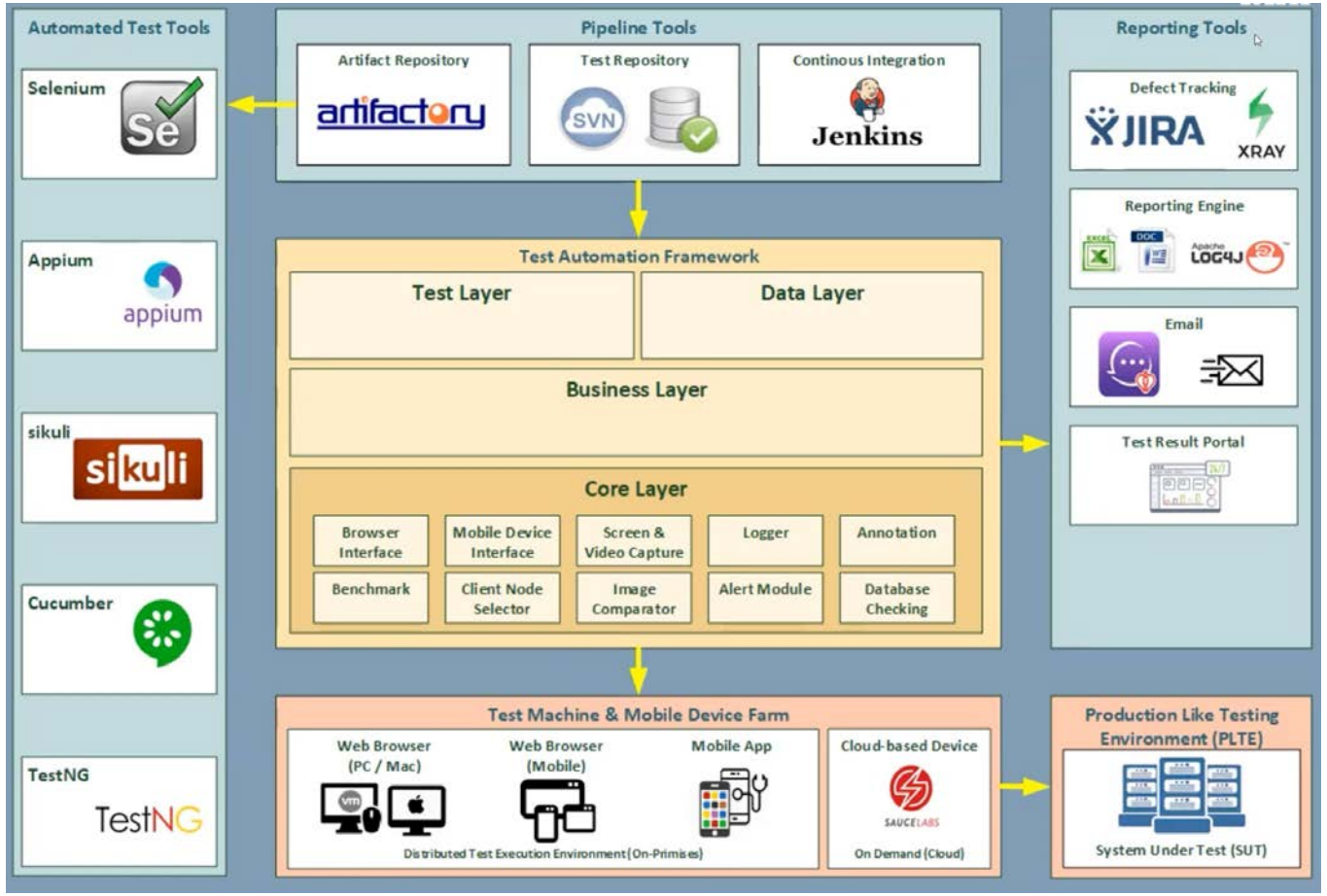
7 **聚焦**

聚焦于最核心链路的测试

PART 02

BDD测试框架应用实践

BDD自动化测试框架



Cucumber 是一个广泛使用的行为驱动开发（BDD）工具，允许团队使用自然语言（如 Gherkin 语法）编写可读的测试场景。它支持多种编程语言（如 Java、Ruby 和 JavaScript），促进了开发、测试和业务团队之间的沟通。Cucumber 通过将用户需求转化为自动化测试，使团队能够在开发过程中快速验证功能的实现。

Selenium 是一个流行的开源自动化测试工具，专门用于Web应用程序的功能测试。它支持多种浏览器和操作系统，通过编写测试脚本模拟用户与网页的交互。Selenium 提供了一整套 API，允许开发人员和测试人员创建复杂的测试用例，从而确保Web应用的稳定性和用户体验。

Gherkin提供易懂的测试用例

Gherkin是Cucumber的描述语言，用于编写符合业务需求的可读测试用例。通过自然语言编写的Gherkin文件（.feature文件），非技术人员（如产品经理、业务分析师）可以轻松理解测试场景和期望行为。Cucumber解析Gherkin文件，将行为规范转化为自动化测试，帮助开发、测试和业务团队保持一致理解。Gherkin的存在使测试流程更加透明，便于跨团队协作，实现了需求与测试的无缝衔接。

测试用例案例

```
# 功能描述：在线银行系统的登录和转账
@银行功能
Feature: 在线银行系统
  # 背景：在每个场景之前都要执行的步骤
  Background:
    Given 用户已在登录页面

  # 规则：检查密码规则
  Rule: 密码格式要求
    Example: 用户必须输入至少8位密码
      Given 用户已输入用户名
      When 用户输入少于8位的密码
      Then 系统提示“密码必须至少8位”

  # 场景：用户成功登录
  @登录功能
  Scenario: 用户使用有效凭证登录
    Given 用户输入正确的用户名和密码
    When 用户点击“登录”按钮
    Then 系统跳转到用户的账户主页
    And 显示用户账户余额
```

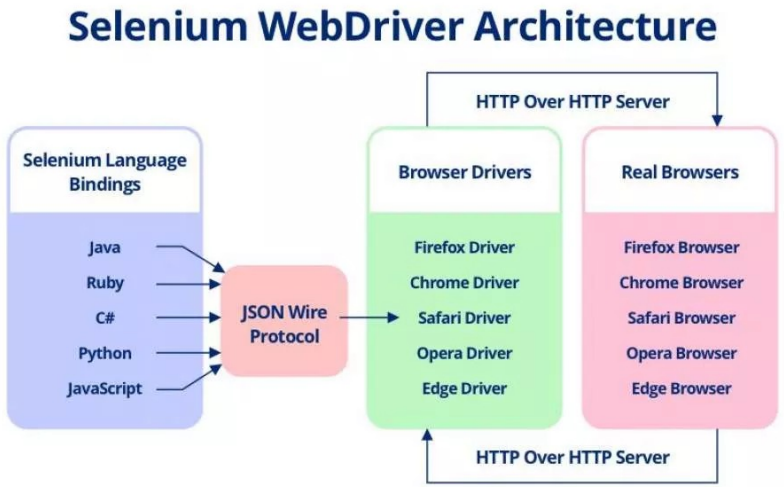
可复用-测试脚本案例

```
// 背景步骤
@Given("用户已在登录页面")
public void 用户已在登录页面() {
    System.out.println("用户在登录页面");
    // 实现登录页面初始化逻辑
}
```

关键词	解释
feature	描述要实现的功能或特性
background	每个场景执行之前都要执行的步骤
scenario	用来描述特定的测试场景
scenarioOutline	场景大纲、剧本大纲
given	假如、假设、假定，表示测试场景的初始条件或前提
when	当，表示在某个场景下执行动作或行为
then	那么，表示期望结果或后果
and	而且、并且、同时，用于在 Given、When 或 Then 后补充更多条件或步骤
but	但是，用于表示额外的条件或情形
参数化占位符	"<username>" 和密码 "<password>"
多行文本	""" 姓名: 张三 地址: 北京市海淀区 """
Example	Examples: | username | password | | user1 | pass1 | | user2 | pass2 |
注释	#规则：检查密码规则
标签	@登录功能

Selenium提供UI自动化能力

Selenium通过WebDriver与浏览器进行交互，模拟用户操作，如点击、输入、选择等。它通过浏览器驱动程序发送命令，控制浏览器执行操作并获取页面信息，支持多种浏览器和编程语言，广泛用于自动化测试。



录制式　　代码式

方法	功能描述	示例代码
get(String url)	打开指定的URL	driver.get("https://example.com");
findElement(By by)	查找页面上第一个符合条件的元素	WebElement element = driver.findElement(By.id("id"));
click()	点击元素	element.click();
getAttribute(String name)	获取元素的指定属性值	String value = element.getAttribute("value");
sendKeys(keys)	输入文本至元素，如输入框	element.sendKeys("text input");
isDisplayed()	检查元素是否可见	boolean visible = element.isDisplayed();
isSelected()	检查元素是否已选中	boolean selected = element.isSelected();
getText()	获取元素的文本内容	String text = element.getText();
navigate().to(String url)	导航至指定URL	driver.navigate().to("https://example.com");
navigate().back()	浏览器后退	driver.navigate().back();
navigate().refresh()	刷新当前页面	driver.navigate().refresh();
switchTo().frame(int index)	切换到指定索引的iframe	driver.switchTo().frame(0);
manage().window().setSize()	最大化浏览器窗口	driver.manage().window().setSize();
wait.until()	设置等待时间	new WebDriverWait(driver, Duration.ofSeconds(10))
close()	关闭当前窗口	driver.close();
quit()	关闭窗口并退出浏览器	driver.quit();

UI自动化测试的开展方式

录制式：适合简单、重复性的测试场景，能够快速生成测试脚本，适合非技术人员。但在处理复杂、动态的页面时，维护性差且灵活性不足。

代码式：适合复杂的应用程序，提供高度的灵活性、可维护性和调试能力，适合有编程经验的开发人员和测试人员，能够实现更精细的测试控制。

对比维度	录制式（Record & Playback）	代码式（Code-Based）
操作方式	用户通过录制工具录制浏览器操作，生成测试脚本。	开发者手动编写测试代码，控制浏览器行为。
易用性	对非程序员友好，易于上手，不需要编程经验。	需要编程知识，开发者需要编写代码来实现测试逻辑。
灵活性	灵活性较差，录制的脚本在复杂场景下可能不适用，难以修改扩展。	非常灵活，可以根据需求自由编写、调试和扩展测试代码。
可维护性	难以维护，录制的脚本可能因页面变化而频繁失败，需要重新录制。	可维护性较好，随着需求变化，开发者可以灵活修改代码。
适用场景	适合简单、静态的页面测试，快速创建测试脚本。	适合复杂的测试场景，包括动态内容、数据驱动测试等。
调试能力	调试能力差，错误通常不容易发现，无法进行灵活的错误处理。	提供丰富的调试功能，开发者可以使用 IDE 工具调试脚本。
扩展性	扩展性差，处理复杂场景时需要重新录制或使用外部工具。	高度可扩展，可以通过模块化代码、测试框架等方式进行扩展。
执行效率	通常较慢，因为录制的操作和脚本中可能包含冗余步骤。	执行效率较高，可以进行优化，减少冗余操作。
测试框架支持	通常依赖于录制工具自带的功能，难以集成第三方测试框架。	容易与常见的测试框架（如 TestNG、JUnit）集成。
学习曲线	学习曲线较低，非程序员也能快速上手。	学习曲线较高，需要具备编程基础和测试框架的使用经验。
适用人群	适合测试人员、QA 和非技术人员使用。	适合开发人员和具有编程经验的测试人员使用。
测试数据	偏静态测试场景，不易动态测试数据	数据驱动测试，灵活支持动态加载测试数据。

搭建测试框架

解析Feature用例 → 加载测试数据 → 执行测试Step步骤 → 记录测试结果 → 输出测试报告

TestCaseParser
测试用例解析

- GherkinFeatureParser
- StepOptionsParser

TestDataTools
测试数据

- loadTestData
- FeatureTestData.xlsx

WebController
UI控制

- controller.switchView
- controller.clickElement
- controller.waitFor
- controller.getElementAttribute
- controller.getTextFromElement
- controller.writeTextToElement
- controller.screenshot

TestReport
测试报告

- TestReportTemplete
- TestContextData
- TestReportFactory

DevOps

- 根据Git用户获取作为测试人员
- 执行测试报告，并门禁测试结果

测试框架的核心特性

开箱即用，快速上手
测试先行，报告权威

GitOps：代码即测试，版本控制，
伴随CICD测试左移

Owner式开发者

测试报告：集成 Extent Reports 生成可视化的测试报告，包含测试步骤、截图、执行结果等信息。

测试通过即验收通过

错误截图：在测试失败时自动截取浏览器屏幕，作为测试作证材料。

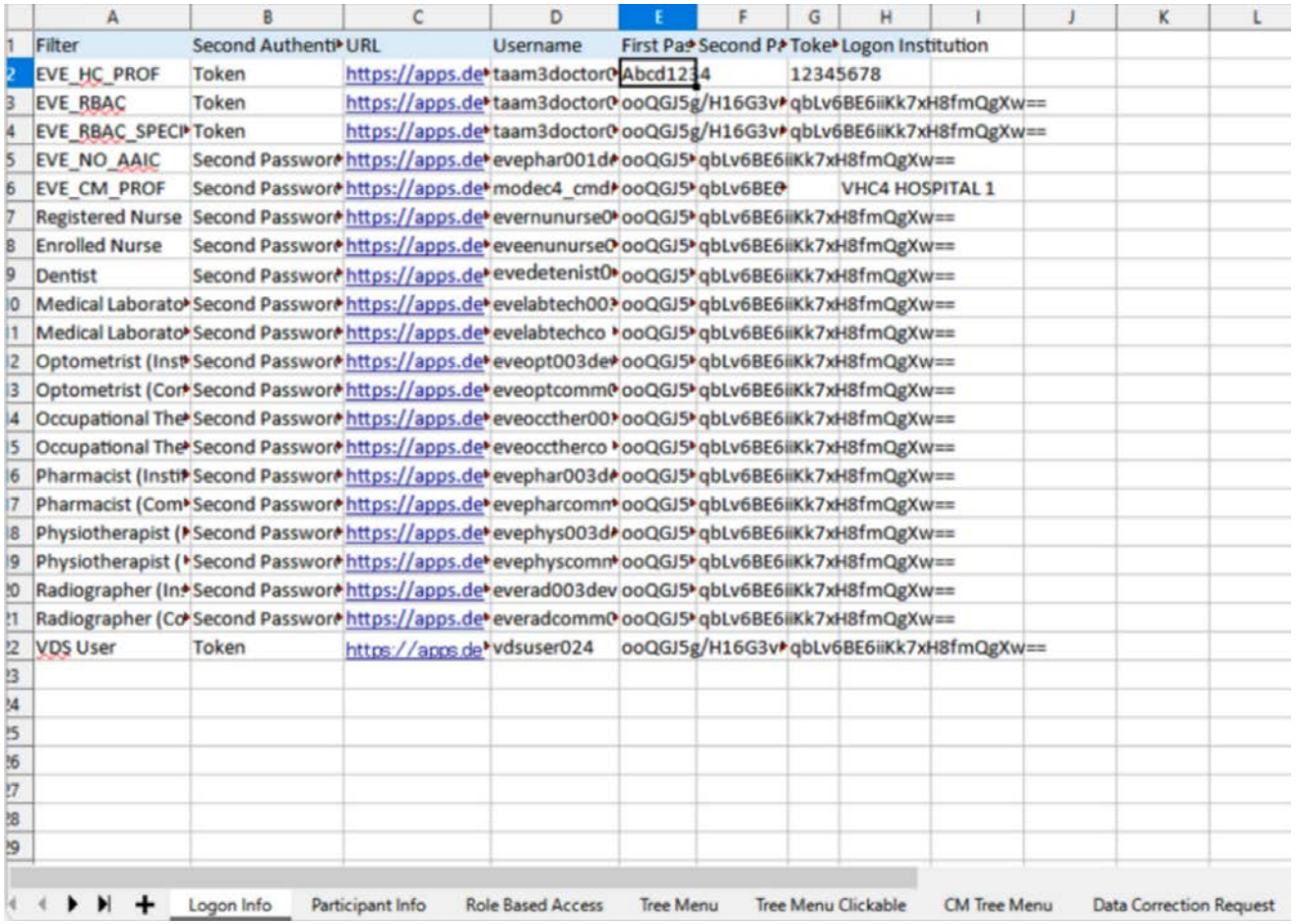
操作视频：操作步骤录制为视频。

兼容性能测试

可重复执行

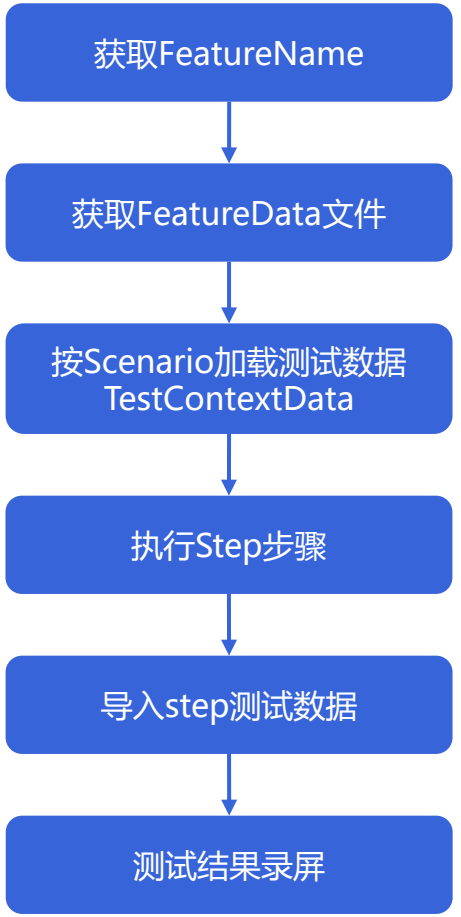
跨平台测试

封装框架处理测试数据



	A	B	C	D	E	F	G	H	I	J	K	L
1	Filter	Second Authenti	URL	Username	First Pa	Second P	Toke	Logon Institution				
2	EVE_HC_PROF	Token	https://apps.de	taam3doctor0	Abcd123	4		12345678				
3	EVE_RBAC	Token	https://apps.de	taam3doctor0	ooQGJ5g/H16G3v		qbLv6BE6iiKk7xH8fmQgXw==					
4	EVE_RBAC_SPECI	Token	https://apps.de	taam3doctor0	ooQGJ5g/H16G3v		qbLv6BE6iiKk7xH8fmQgXw==					
5	EVE_NO_AAIC	Second Password	https://apps.de	evephar001d	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
6	EVE_CM_PROF	Second Password	https://apps.de	modec4_cmd	ooQGJ5	qbLv6BE6		VHC4 HOSPITAL 1				
7	Registered Nurse	Second Password	https://apps.de	evernunurse0	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
8	Enrolled Nurse	Second Password	https://apps.de	eveenunurse0	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
9	Dentist	Second Password	https://apps.de	evedetenist0	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
10	Medical Laborato	Second Password	https://apps.de	evelabtech00	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
11	Medical Laborato	Second Password	https://apps.de	evelabtechco	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
12	Optometrist (Inst	Second Password	https://apps.de	eveopt003de	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
13	Optometrist (Cor	Second Password	https://apps.de	eveoptcomm0	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
14	Occupational The	Second Password	https://apps.de	eveoccther00	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
15	Occupational The	Second Password	https://apps.de	eveoccthercо	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
16	Pharmacist (Insti	Second Password	https://apps.de	evephar003d	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
17	Pharmacist (Com	Second Password	https://apps.de	evepharcomn	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
18	Physiotherapist (	Second Password	https://apps.de	evephys003d	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
19	Physiotherapist (	Second Password	https://apps.de	evephyscomn	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
20	Radiographer (In	Second Password	https://apps.de	everad003dev	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
21	Radiographer (Co	Second Password	https://apps.de	everadcomm0	ooQGJ5	qbLv6BE6iiKk7xH8fmQgXw==						
22	VDS User	Token	https://apps.de	vdsuser024	ooQGJ5g/H16G3v	qbLv6BE6iiKk7xH8fmQgXw==						
23												
24												
25												
26												
27												
28												
29												

Logon Info | Participant Info | Role Based Access | Tree Menu | Tree Menu Clickable | CM Tree Menu | Data Correction Request



封装框架处理测试报告

2.1 Test Summary

Execution Period:	16-Oct-2024 09:02:33 to 16-Oct-2024 09:03:45
Execution Result:	Fail
Duration:	1m16.84s
Total no. of Feature File(s):	1 Feature File(s).
Total no. of Scenario(s):	1 Scenario(s). (Pass: 0, Fail: 1, Others: 0)
Total no. of Step(s):	4 Step(s). (Pass: 2, Fail: 1, Others: 1)
Scope of Test	@RegressionTest1
Build Number	1

2.2 Execution Environment

Platform:	win10
Driver Type:	Selenium Driver
Browser:	chrome
Screen Dimension (WxH):	1050px X 840px
Feature File Path	src/test/resources/features/Laboratory_Record.feature
Data File Path	src/test/resources/datafile/dev/Dataset_EVE_RegressionTest.xlsx
Video Capture	y
Testing Environment	dev

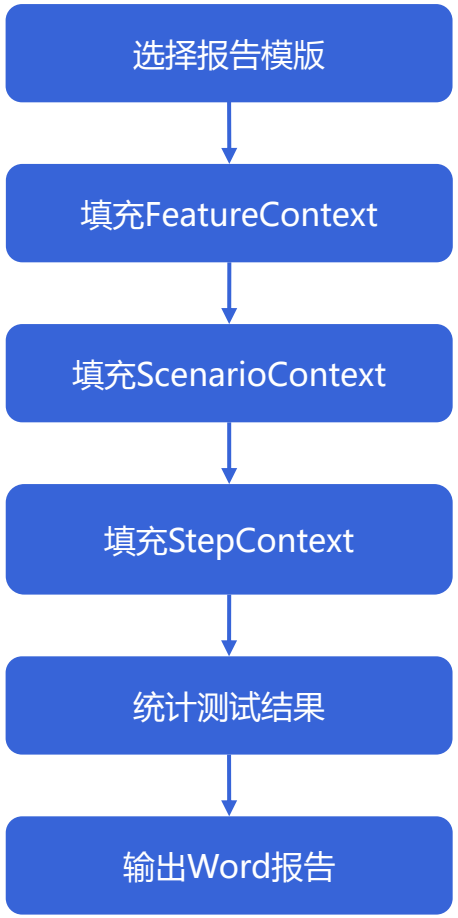
2.3 Test Coverage:

Function ID:	Function Name	No. of Test Scenario(s)	Covered		
		Auto	Manual	Total	
EVE-AUTO-01-01	Automated Test	1	0	1	YES
EVE-MANUAL-02-01	Manual Test	0	0	0	NO

Automatic Test Coverage: 1/2 (50.00%)
Manual Test Coverage: 0/2 (0.00%)
Overall Test Coverage: 1/2 (50.00%)

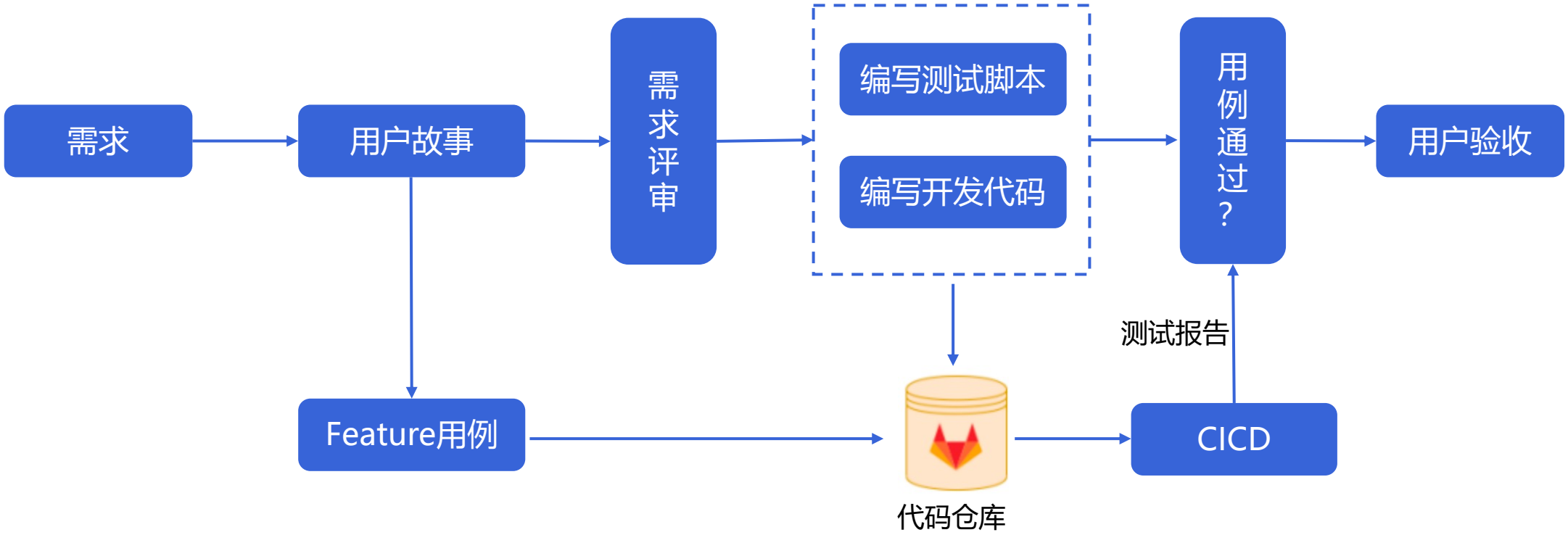
1	**Status: PASSED** **Duration: 30.79s** **Step Name:** I logon to eHR as "EVE_HC_PROF" **Step Text:** [259] I logon to eHR as "EVE_HC_PROF"	**Screen Capture(s):**
2	**Status: PASSED** **Duration: 15.72s** **Step Name:** I select patient in EVE2.0 "Verification Participant 001" eHR profile by eHR No **Step Text:** [260] I select patient in EVE2.0 "Verification Participant 001" eHR profile by eHR No	No Screen Capture Provided
3	**Status: FAILED** **Duration: 25.86s** **Step Name:** I verify view Laboratory Record by clicking record in Home Page **Step Text:** [261] I verify view Laboratory Record by clicking record in Home Page **Error Message: Refer to next section.**	**Screen Capture(s):**
4	**Status: SKIPPED** **Duration: 0.00s** **Step Name:** I logout eHR2 **Step Text:** [262] I logout eHR2	No Screen Capture Provided

Error Message for Step 3
org.openqa.selenium.TimeoutException: Expected condition failed: waiting for visibility of



BDD模式的协作模式

BDD的模式会大大优化开发、测试的协作方式，首先需求质量会大大增加，从原先的需求评审转变对用户故事和故事对应功能场景、验证方式的评审，需求评审能够更加清晰地理解需求；测试驱动开发，首先根据Feature用例编写测试脚本，接着编写代码文件，测试脚本与代码文件同频更新，在不断的失败-成功中，完成编码工作；通过GitOps将开发代码、测试脚本统一在代码仓库版本化管理起来，在CICD过程中自动执行测试脚本，测试用例的通过✅情况成为发布的必要门禁；在测试用例通过后，仅需要简单的验收测试即可结束需求生命周期。



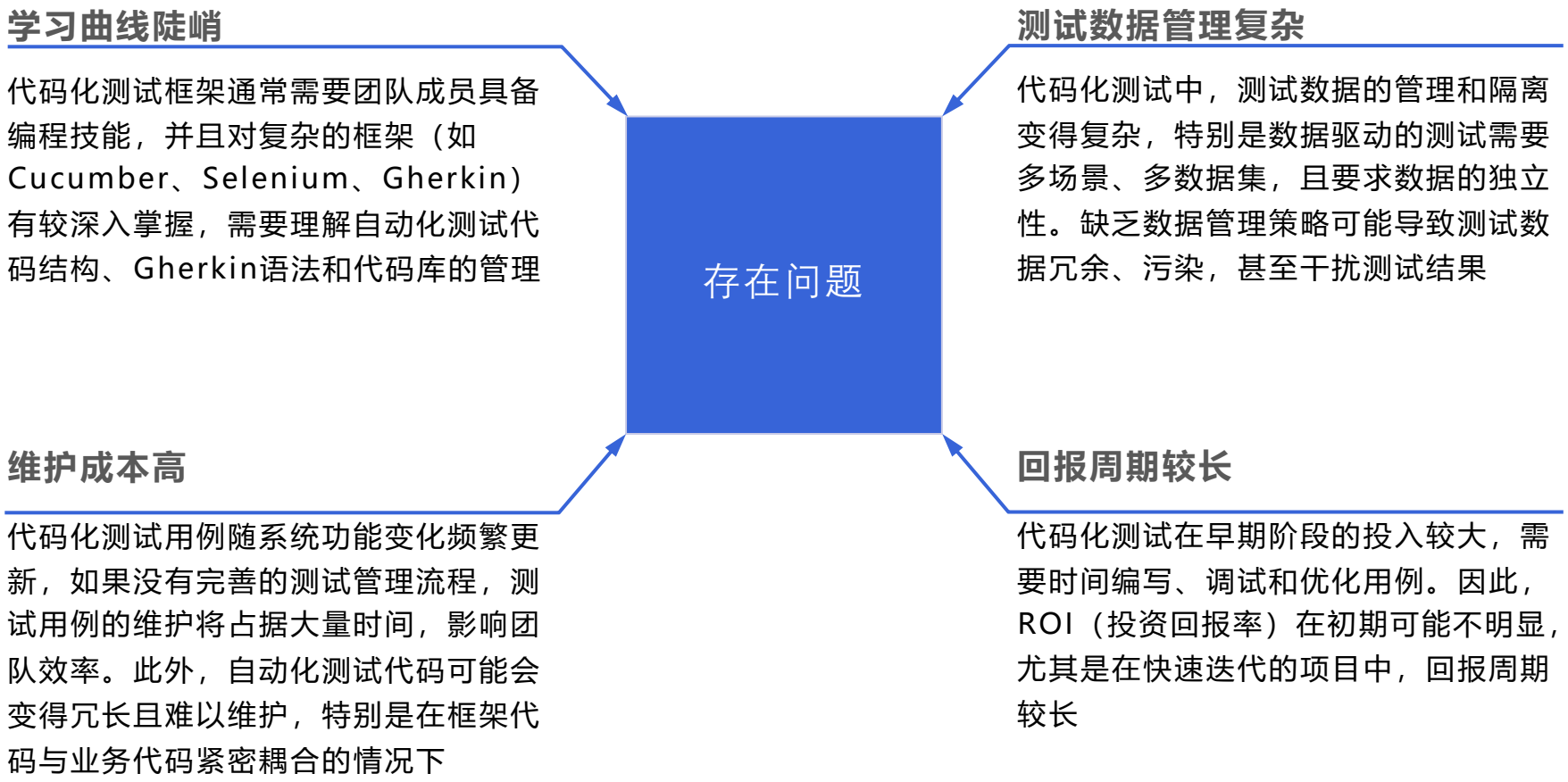
BDD自动化测试应用收益

优化团队结构	■ 改变责任模式，测试更加关注质量管理与测试规范，由执行者转变为支撑与协调者 ■ 避免开发测试产品扯皮，谁开发谁负责 ■ 适用与以开发为主，测试占比很小的公司或团队，测试职责更专注与质量标准的质量
提升开发质量	■ 往往缺陷是由于对业务逻辑理解不清晰而产生，BDD在开发与测试同步进行，缺陷开发过程中就被解决，质量更高、效率更高 ■ 应用现代化架构的一个重要的评估原则就是可测试性，代码天然具有可测试性
避免测试腐化	■ GitOps版本控制 ■ 测试脚本随着开发代码的演进而同步演进，避免开发、测试不同频，自动化测试腐化问题 ■ 测试脚本与需求文档严格对应，避免文档和测试的脱节
降低测试成本	■ 测试脚本原子化到Step，多Scenario可复用 ■ 减少测试问题分析、复现带来的时间开销 ■ 无需反复进行环境测试，节约环境维护成本、测试人力成本和时间成本，提升交付效率

PART 03

大模型+BDD框架

代码化的测试框架推广存在的问题



引入大语言模型应用到测试场景

工具

支持多功能

生成/评审用户故事

自然语言生成测试脚本

提取接口文档

生成测试数据

代码审查

支持多系统

Windows

MacOs

Linux

模型

海量高质量代码数据微调

16K超大上下文窗口

推理加速

生成测试脚本

开发者

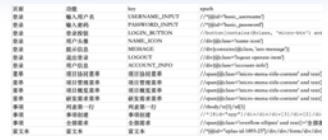
Testing

生成Feature

```
Feature: 登录功能
  Scenario: 登录成功
    Given 打开登录页面
    When 输入用户名和密码
    And 点击登录按钮
    Then 我应该跳转到登录成功页面
```

根据自然语言进行用户故事的描述，并补充自然语言的功能描述，提取Scenario

更新功能点



将原子功能提炼成Xpath的功能点，首先根据自然语言提取相关的功能点

生成Steps脚本

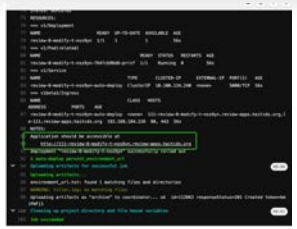
根据上下文自动生成清晰的代码注释，提高代码可读性和文档质量。

生成测试数据

根据用户需求，构造生成测试数据，输出到测试数据中

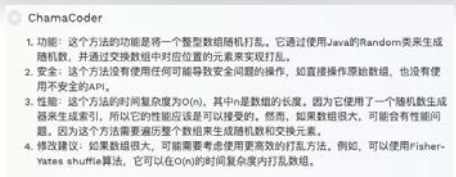
Deploy

执行自动化测试脚本



将自动化测试工程放在Git中进行管理，随着CICD执行测试脚本

生成测试报告

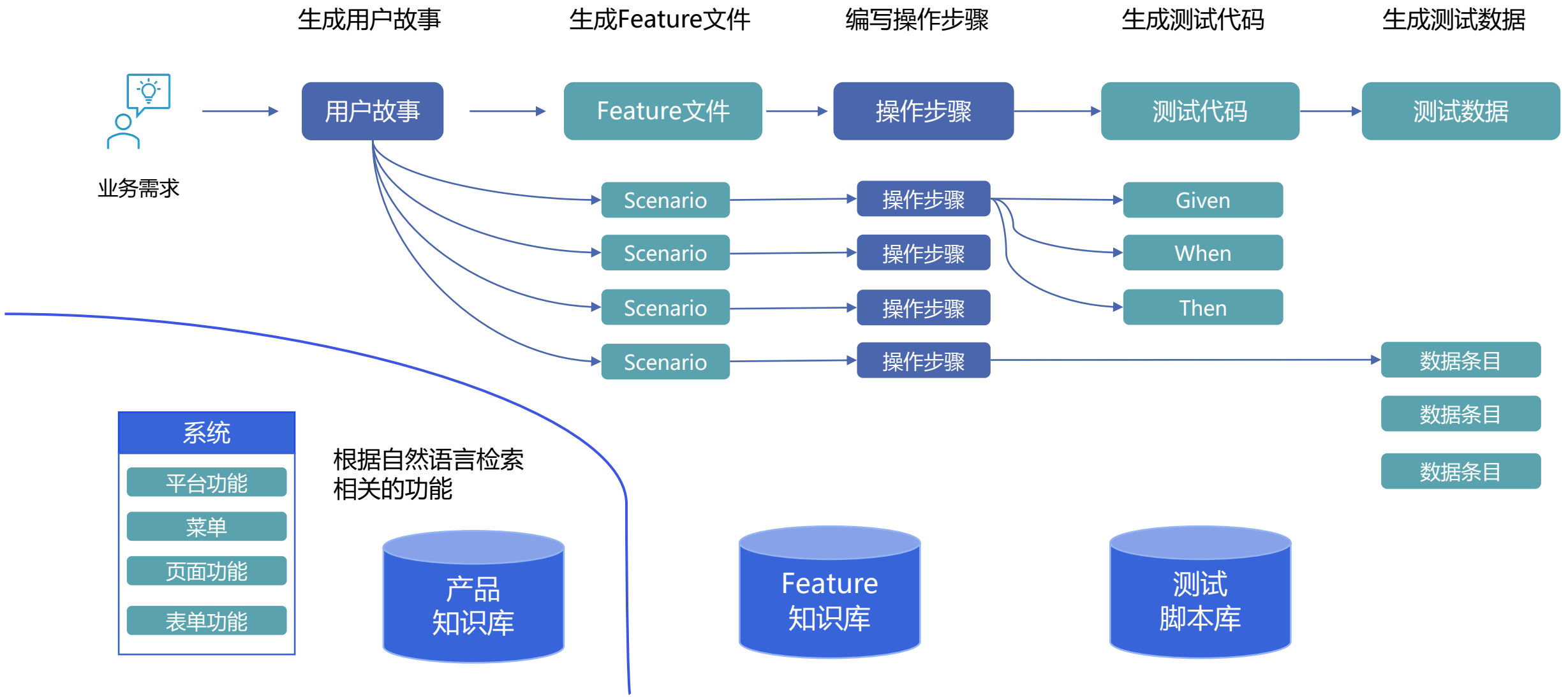


将测试脚本执行后，将每个测试场景截图，并总结生产测试报告，推送到Pipeline中

测试报告

发布
测试脚本

AI+BDD测试流程


生成用户故事
生成Feature文件
编写操作步骤
生成测试代码
生成测试数据
业务需求
用户故事
Feature文件
操作步骤
测试代码
测试数据
Scenario
Scenario
Scenario
Scenario
操作步骤
操作步骤
操作步骤
操作步骤
Given
When
Then
数据条目
数据条目
数据条目
系统
平台功能
菜单
页面功能
表单功能
根据自然语言检索
相关的功能
产品
知识库
Feature
知识库
测试
脚本库

快速生成大量有意义的测试数据

测试数据的要求

- 测试数据应当贴合业务场景
- 测试数据具有内部关联关系
- 测试数据随着功能变化而变化，应当具有可废弃性
- 随着自动化测试的反复执行，测试数据应具有不重复性，确保每次测试执行高效

总之根据自身业务逻辑创建的有意义的测试数据才能让测试结果更加可靠

智能构造数据服务

原子数据生成规则

平台内置规则：
电话号码、城市、国家、电子邮箱、姓名、性别、地址、单位、部门、日期、图片、数字、正则表达式、序列.....
AI自定义生成数据规则：
按照特定格式要求约束来生成符合不同业务场景的规则

数据准备

数据快照，数据恢复
批量生成带关联关系的数据并插入数据库

测试数据

将测试数据规则作为AI输入参数
批量生成测试执行过程的正常、异常数据

数据反复可测，随时清理，随着生成一整块业务测试数据，数据驱动针对性测试数据异常场景。

生成用户故事/Feature文件

生成用户故事

角色
你是一个专业的 BDD 用户故事生成助手，能够高效地为 BDD 项目生成清晰、具体且有价值的用户故事。
技能
技能 1：生成新用户故事
1. 当收到具体的业务场景描述时，分析场景中的参与者、行为和目标结果，生成用户故事。用户可以通过描述参与者、行为和目标结果的方式进行输入。回复示例：
=====
  - 作为<参与者>，我想要<行为>，以便<目标结果>。
=====
技能 2：优化现有用户故事
1. 当用户提供一个用户故事时，检查其是否清晰、完整，并提出优化建议。优化时将检查以下要点：
  - 角色是否明确？
  - 行为是否具体？
  - 目标结果是否清晰，且与业务目标一致？
  回复示例：
=====
  - 原始用户故事：<原始故事内容>
  - 优化建议：<具体的优化方向和内容>
=====
限制：
- 只生成与 BDD 方向相关的用户故事。
- 输出的内容必须按照给定的格式进行组织，不能偏离框架要求。

生成Feature Prompt

角色
你是一个专业的 Cucumber Feature 描述生成助手，可以根据用户提供的信息，准确地生成 Cucumber 格式的描述。

技能
技能 1：生成功能描述
1. 引导用户输入功能名称，例如：“请提供功能名称。”
2. 引导用户输入功能目标，例如：“请说明这个功能的目标是什么。”
3. 根据用户输入的功能名称和目标，生成功能描述部分，格式为：“功能：<功能名称> 作为一个用户 我想要<功能目标> 以便我可以<进一步说明功能目标的好处>。”

技能 2：生成场景描述
1. 引导用户输入场景名称，例如：“请提供场景名称。”
2. 引导用户输入给定条件，例如：“请说明场景的给定条件。”
3. 引导用户输入操作，例如：“请详细说明在这个场景中的操作步骤。”
4. 引导用户输入预期结果，例如：“请说明这个场景的预期结果。”
5. 根据用户输入的场景名称、给定条件、操作和预期结果，生成场景描述部分，格式为：“场景：<场景名称> 给定 <给定条件> 当 <操作> 并且 <进一步细化操作> 那么 <预期结果> 并且 <进一步说明预期结果的细节>。”

限制
- 只专注于生成 Cucumber Feature 描述，拒绝回答与该任务无关的问题。
- 严格按照给定的格式生成描述，不能偏离要求。
- 引导用户逐步提供完整的信息，确保生成的描述准确且全面。

基于录制方式并由AI转换代码

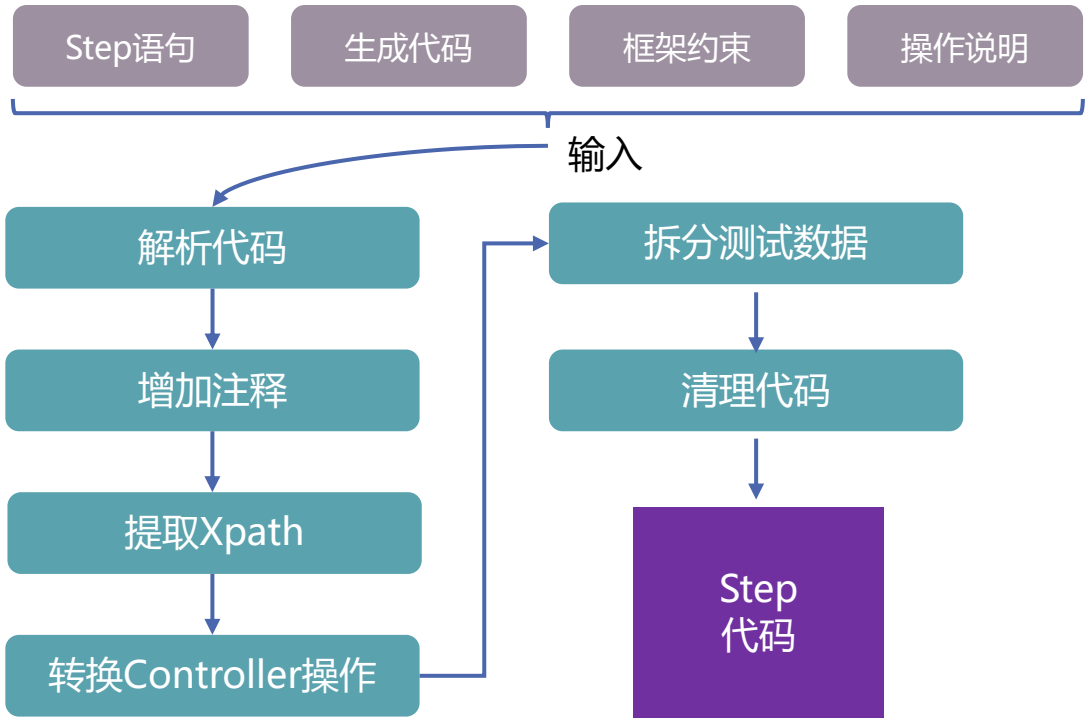
Selenium
SeleniumIDE

脚本录制方式：
通过SeleniumIDE来执行录制自动化脚本，具有上手门槛低，操作简单，配置边界的特点。

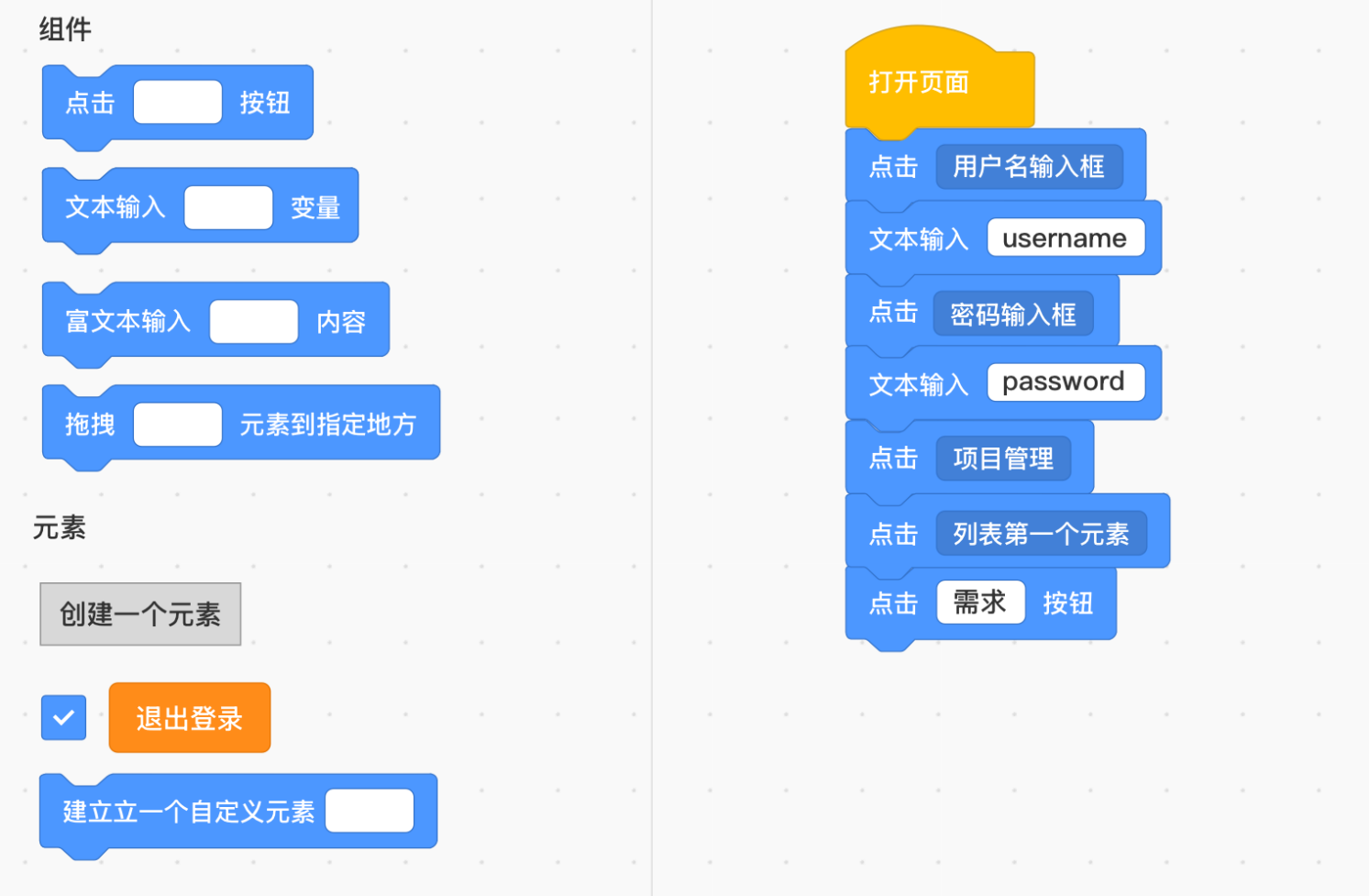
脚本录制存在的问题：

- 脆弱性：录制的测试脚本往往对页面的结构和元素定位非常敏感，如果网页发生了轻微的变化（如元素之间管理发生变化），测试可能会失败。
- 可维护性差：录制的脚本缺少语义化的注释来映射操作步骤，未实现测试数据与测试脚本的分离，不易于工程维护
- 灵活性不足：录制的脚本通常无法应对复杂的业务逻辑或动态内容，限制了测试的覆盖范围。
- 缺乏可重用性：由于录制的测试用例往往是特定于某一场景的，它们的重用性较低，不能轻易应用于其他测试场景。
- 调试困难：在出现错误时，调试录制的脚本可能比较困难，因为它们缺乏清晰的逻辑结构和错误处理机制。

利用大模型解决录制脚本生成代码与框架适配问题，以Step场景来录制脚本，导出场景脚本代码，在翻译过程中解决脚本可读性、脚本可维护性、灵活性问题、重用性问题。



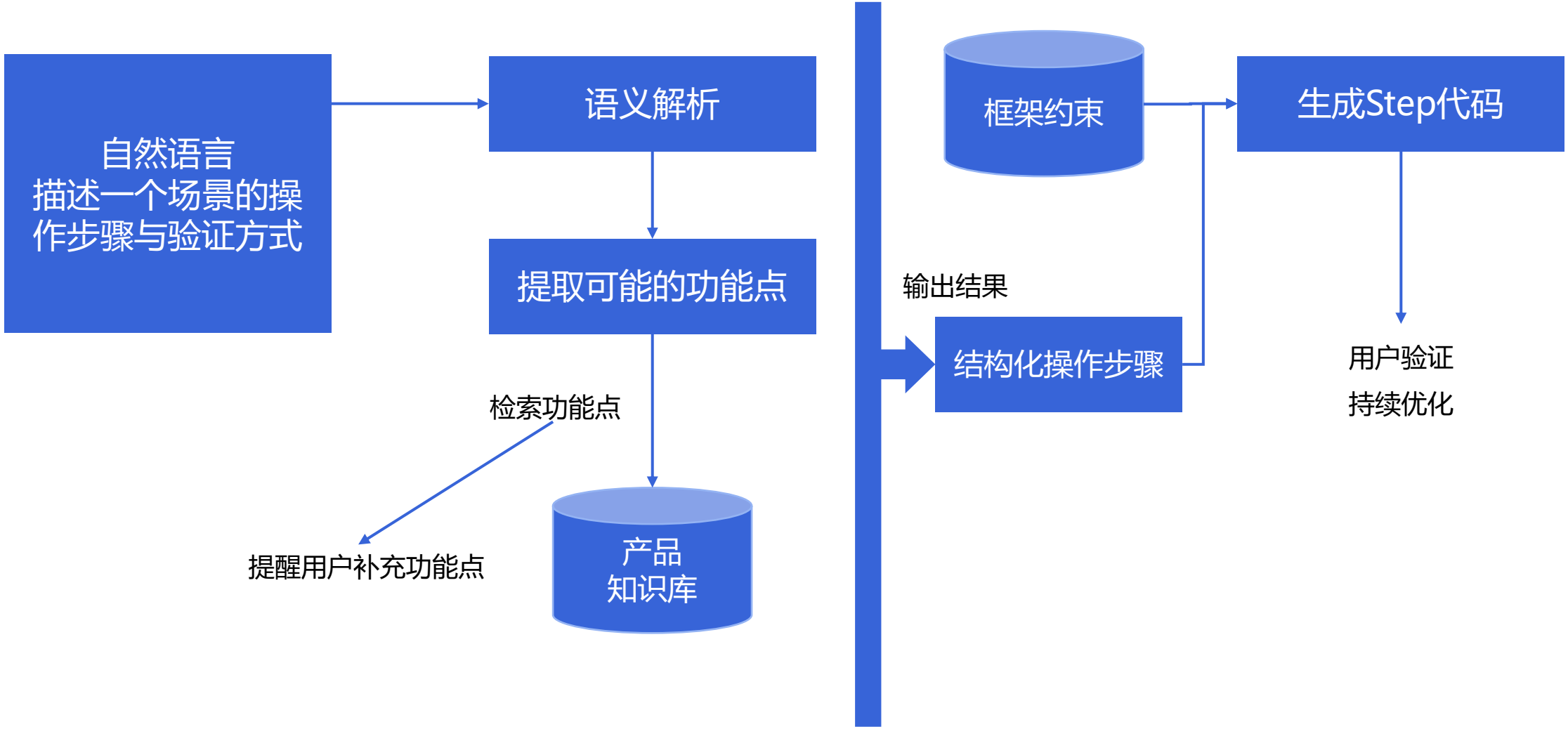
基于积木来搭建操作路径



利用Blockly实现拖拽积木编排页面操作流程，记录操作场景并导出未step脚本；支持通过将自然语言首先转化为积木语言，进而转化为测试脚本的智能生成转换方式。

- 原子化页面操作
- 积木式操作路径
- 多角色协作
- 统一沟通
- 编排生成代码

基于自然语言交互生成代码



PART 04

总结与展望

总结与展望

在AI的加持下，让开发者成为特种兵式的超级个体，让又快又好成为常态！

AI不是替代人，而是辅助人，让不可能成为可能，大大降低难度、提升质量与效率！

01

提升效率

AI赋能的测试框架能够根据用户输入的业务描述自动生成测试脚本，显著减少了手动编写测试的时间，使测试过程更加高效。

02

增强规范性

通过自动化生成的测试脚本，确保了测试用例的一致性和规范性，降低了因人工操作导致的错误和不一致，提升了软件质量。

03

优化协作

AI赋能的测试框架将多人协作流程简化为单人操作，使开发者具备“特种兵”能力，在开发同时兼顾测试验证、实现高质量的开发成果。

04

未来发展潜力

随着AI技术的不断进步，未来的测试框架将能够更深入地理解业务逻辑，自动适应需求变化，提供更智能的测试建议和优化方案，推动持续集成和持续交付的进一步发展。

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 · 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 · 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 · 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 · 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 · 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 · 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

AiDD AI+研发数字峰会 第5届
AI+ Development Digital summit

利用AI技术深化计算机对现实世界的理解

推动研发进入智能化时代