SHANGHAI JIAO TONG UNIVERSITY

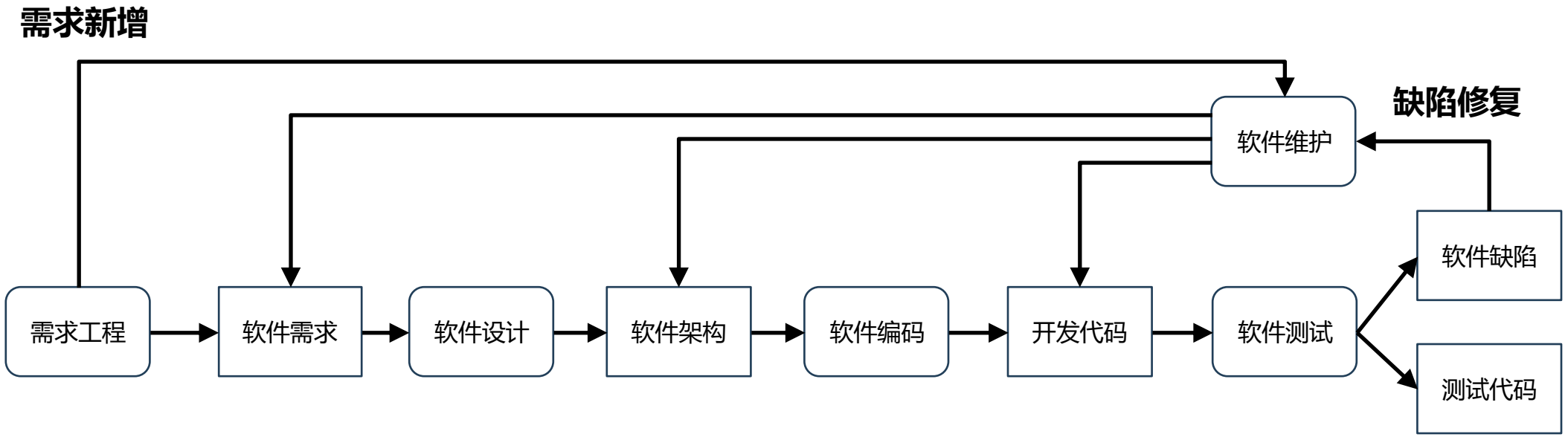
需求、测试与编码

林云

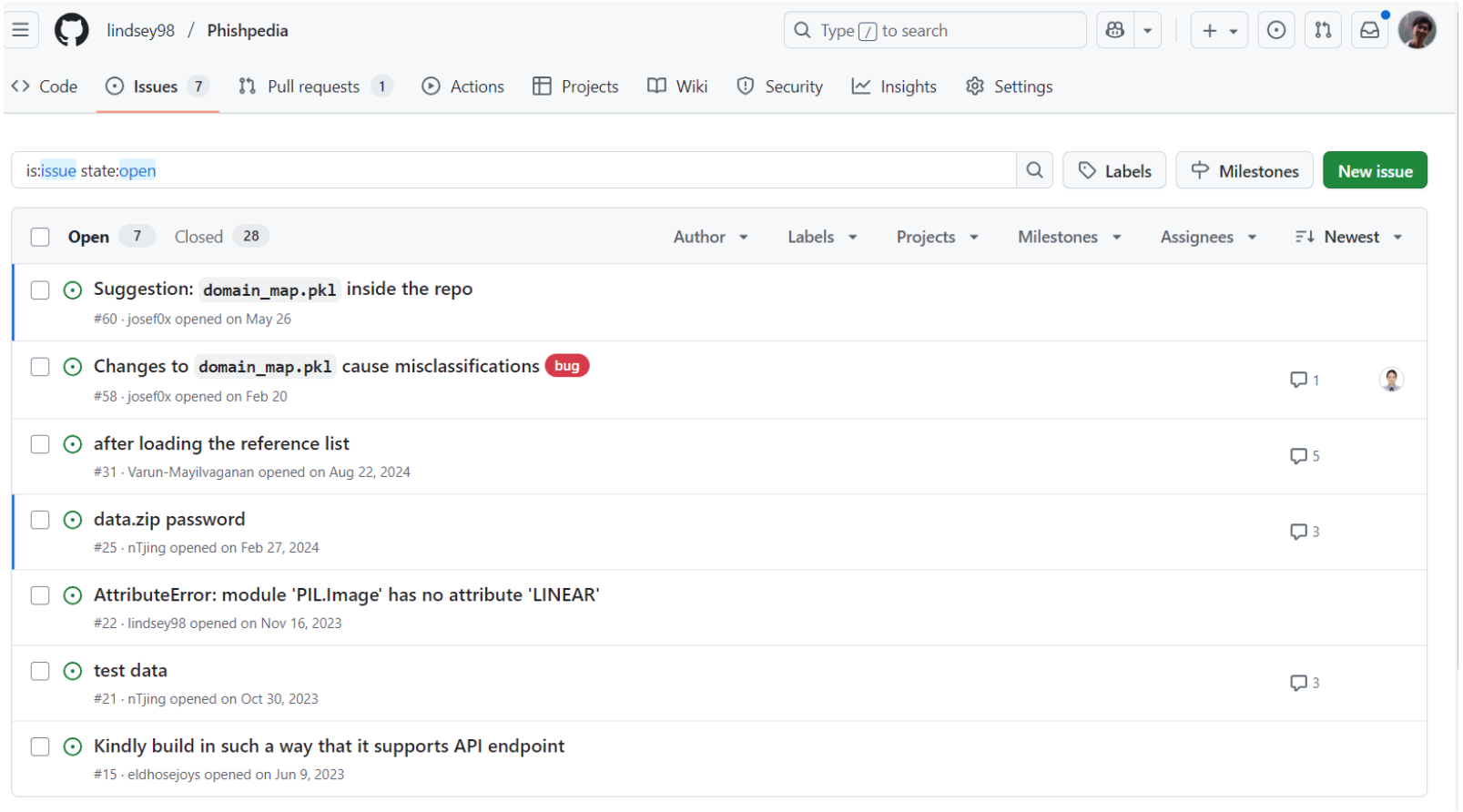
上海交通大学计算机系

lin_yun@sjtu.edu.cn

上节课回顾：软件工程的基本阶段

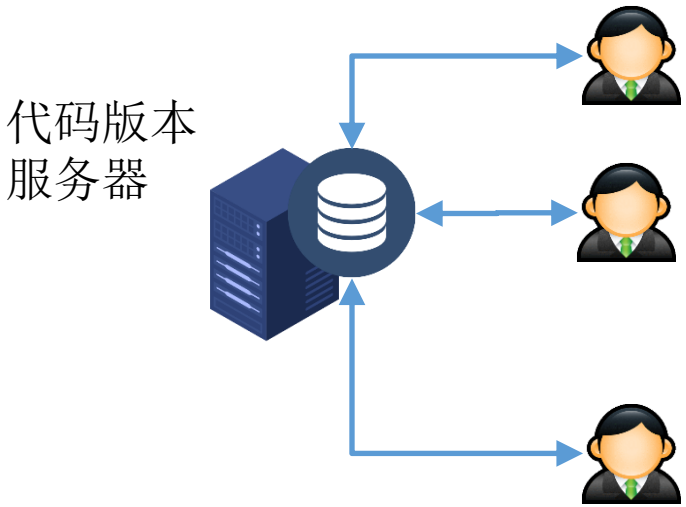


上节课回顾：问题追踪系统（issue tracking system）

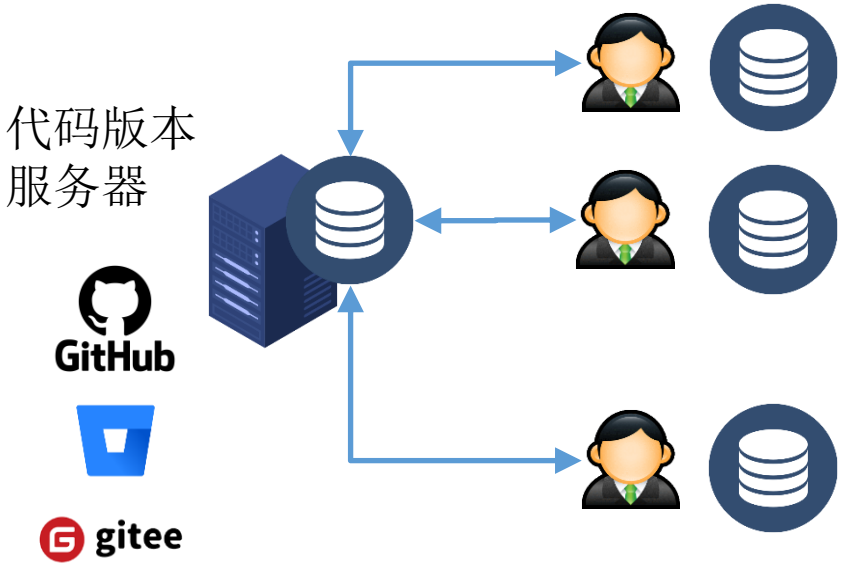


上节课回顾：版本控制系统

集中式版本控制系统



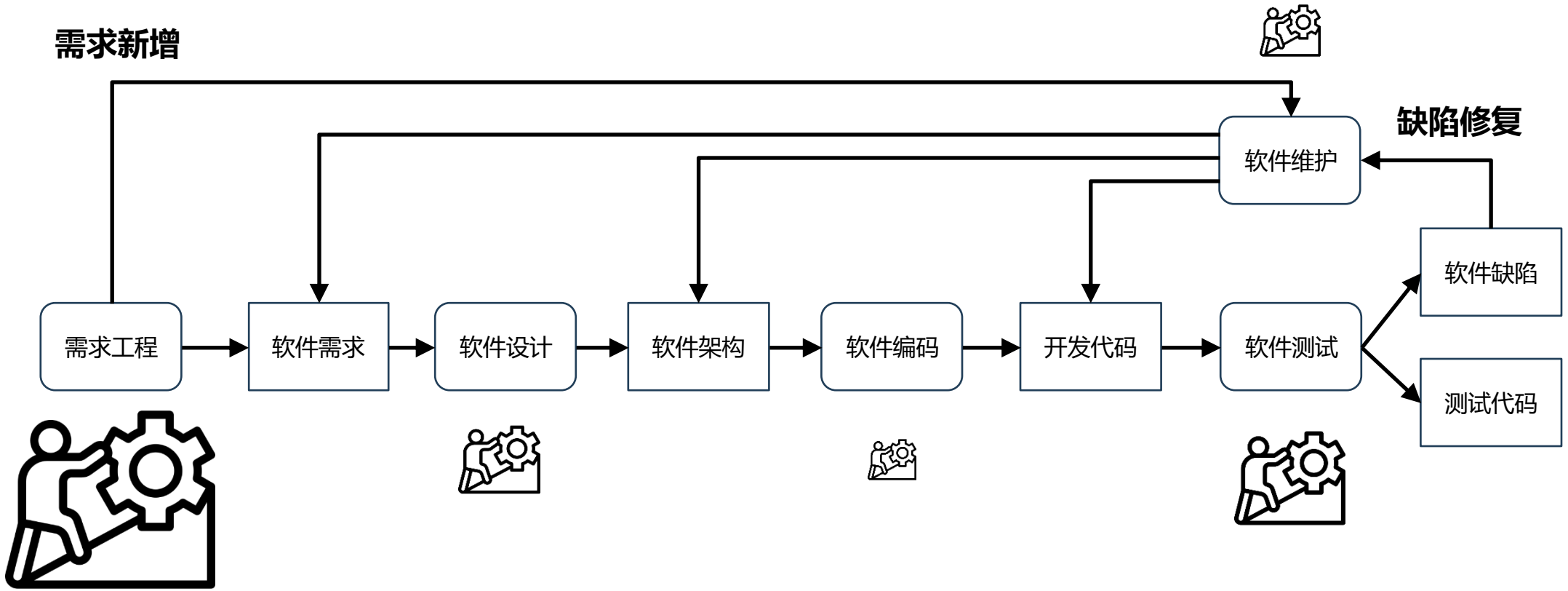
分布式版本控制系统



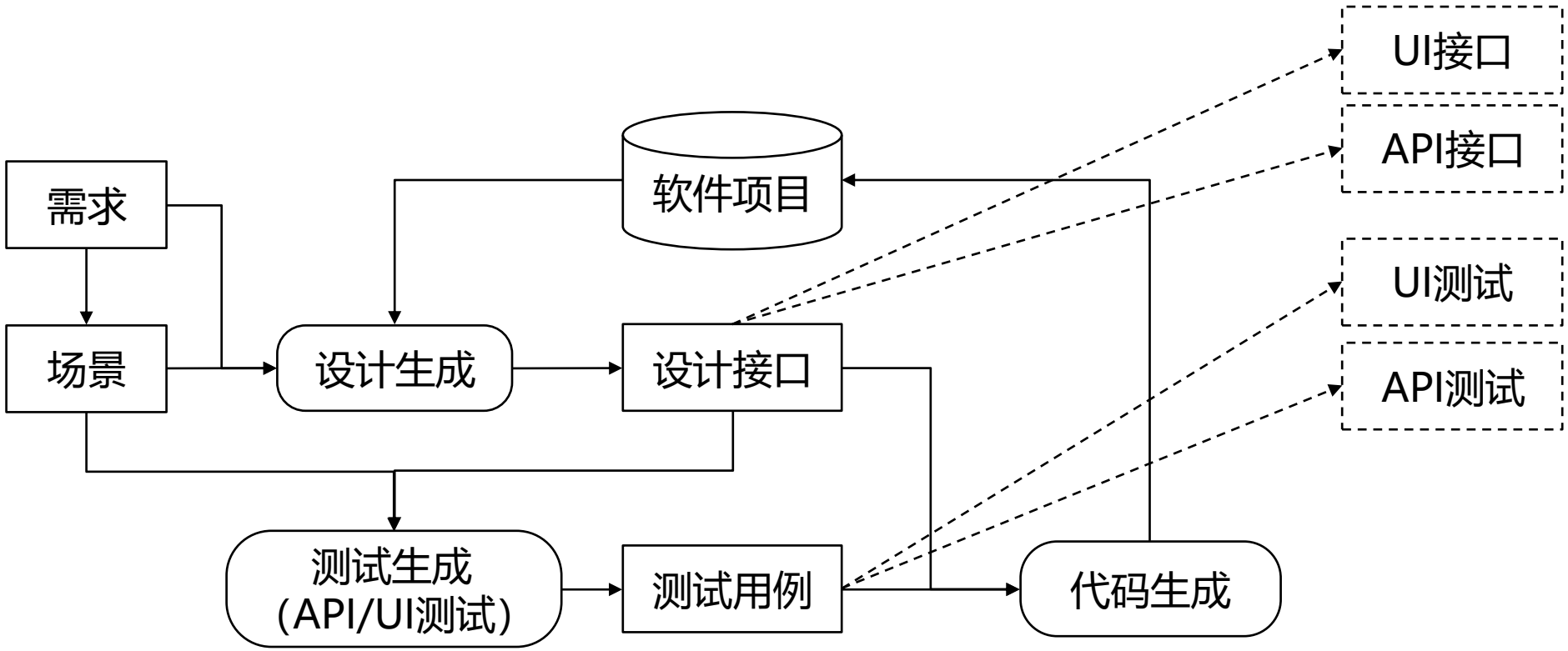
学习目标

- 理解和掌握需求、需求场景、软件设计、软件测试、和代码编写的概念
- 理解测试驱动开发的概念
- 基于AI编程来基于需求实践测试驱动的开发

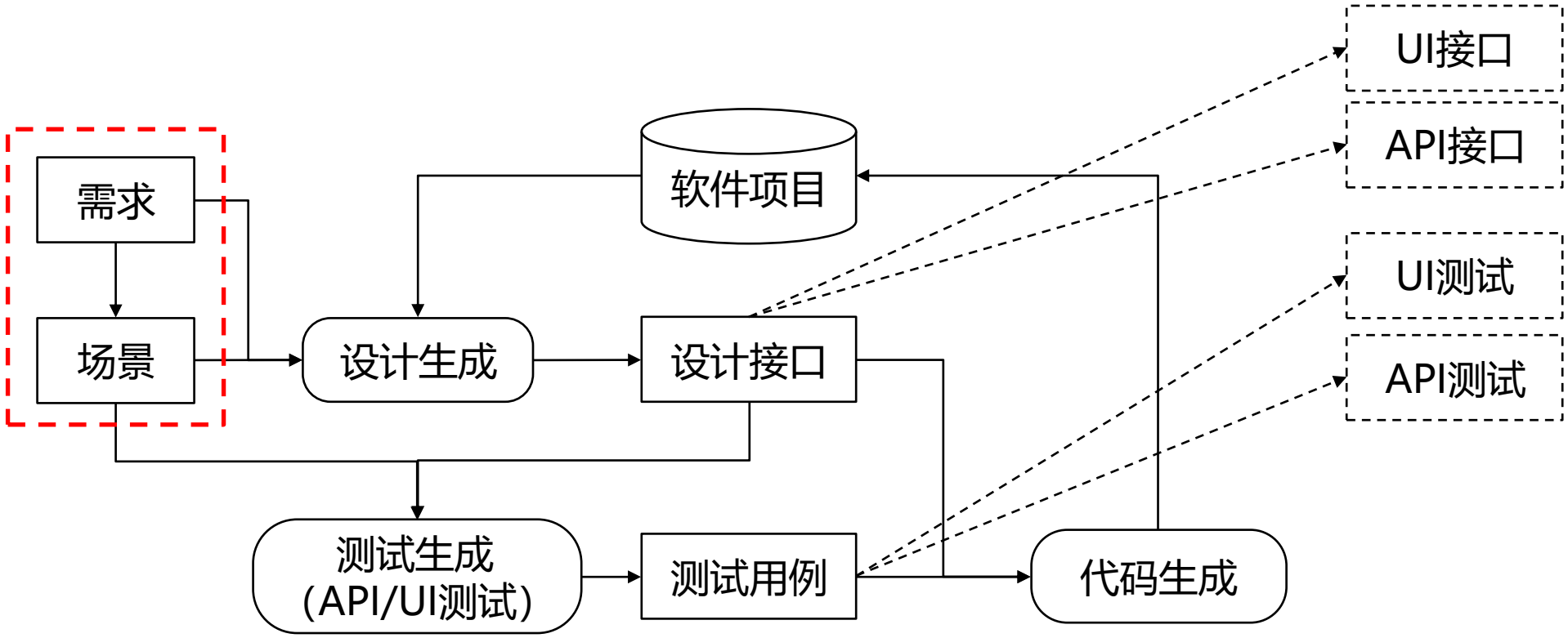
AI时代下软件工程的需求工程



测试驱动的开发与实践


UI接口
API接口
UI测试
API测试
软件项目
需求
场景
设计生成
设计接口
测试生成
(API/UI测试)
测试用例
代码生成

测试驱动的开发与实践


需求
场景
软件项目
设计生成
设计接口
测试生成
(API/UI测试)
测试用例
代码生成
UI接口
API接口
UI测试
API测试

软件需求

- 软件需求是对系统应做什么以及如何表现的说明

软件需求

- 软件需求是对系统应做什么以及如何表现的说明

一个淘宝网站的登录功能背后有哪些需求?

软件需求

- 软件需求是对系统应做什么以及如何表现的说明
  - 功能性需求（functional requirements）: 系统要做什么
  - 非功能性需求（non-functional requirement）： 系统要做的多好

功能性需求	非功能性需求
用户可以通过**手机号+密码**登录	登录响应时间 ≤ 2秒
用户可以通过**验证码**登录	
登录成功后跳转至首页并显示用户名	登录页面在Chrome/Edge/Safari下兼容

软件需求

- 软件需求是对系统应做什么以及如何表现的说明
  - 功能性需求（functional requirements）：系统要做什么
  - 非功能性需求（non-functional requirement）： 系统要做的多好

功能性需求	非功能性需求
用户可以通过**手机号+密码**登录	登录响应时间 ≤ 2秒
用户可以通过**验证码**登录	
登录成功后跳转至首页并显示用户名	登录页面在Chrome/Edge/Safari下兼容

还有哪些非功能性需求？

需求场景

- **需求场景（Requirement Scenario）**是对系统在特定上下文下所表现行为的结构化描述，用于说明：在某种前提条件（Given）下，当用户或系统执行某个动作（When）时，系统应表现出何种结果（Then）。

元素	说明	示例（淘宝登录
Given（前提条件）	描述系统初始状态或上下文	用户已打开淘宝登录页面
When（触发动作）	用户或系统执行的事件	用户输入手机号和密码并点击“登录”
Then（预期结果）	系统应产生的行为或输出	系统跳转到首页并显示用户名

需求场景（形式化表述）

$$S = \langle Context, Action, ExpectedOutcome \rangle$$

- 其中：
  - Context 对应 Given
  - Action 对应 When
  - ExpectedOutcome 对应 Then
    这样的形式允许AI系统将自然语言需求映射为机器可理解的逻辑条件。

场景描述练习

- “用户在登录页面输入正确的账号和密码后，系统应显示首页并欢迎用户。”

场景名称	用户成功登录淘宝
Given	
When	
Then	

场景描述练习

- “用户在登录页面输入正确的账号和密码后，系统应显示首页并欢迎用户。”

场景名称	用户成功登录淘宝
Given	用户已打开淘宝登录页面
When	用户输入正确的账号和密码并点击“登录”
Then	系统跳转到首页并显示欢迎信息

场景描述练习

- “系统应在2秒内完成登录响应。”

场景名称	用户成功登录淘宝
Given	
When	
Then	

场景描述练习

- “系统应在2秒内完成登录响应。”

场景名称	登录响应时间要求
Given	用户在正常网络环境下打开淘宝登录页
When	用户输入正确的账号密码并点击“登录”
Then	系统在2秒内完成验证并跳转首页

需求场景描述语言Gherkin

- **Gherkin** 是一种结构化的自然语言（Structured Natural Language），用于行为驱动开发（BDD）中描述软件系统的行为。
  它让需求文档既能被人阅读，又能被测试工具执行。

示例：

```
Feature: 用户登录淘宝

  Scenario: 成功登录
    Given 用户在登录页面
    When 用户输入正确的账号和密码并点击“登录”
    Then 系统跳转到首页并显示用户名

  Scenario: 密码错误
    Given 用户在登录页面
    When 用户输入错误的密码
    Then 系统提示“用户名或密码错误”
```

分支需求场景

- **变体需求场景（Variant Scenario）**是指：在一个功能需求下，围绕同一主目标（Main Scenario），针对不同输入条件、异常情况或交互路径而产生的分支场景（Alternative Scenarios）或异常场景（Exception Scenarios）。

主场景（用户名密码登录）

Scenario: 用户使用正确手机号和密码成功登录
Given 用户已打开淘宝登录页面
When 用户输入正确的手机号和密码并点击“登录”
Then 系统跳转到首页并显示用户名

分支场景（短信登录）

Scenario: 用户使用短信验证码登录
Given 用户已打开淘宝登录页面
And 用户选择“短信登录”选项
When 用户输入手机号并正确输入验证码
Then 系统登录成功并跳转到首页

分支需求场景

- **变体需求场景（Variant Scenario）**是指：在一个功能需求下，围绕同一主目标（Main Scenario），针对不同输入条件、异常情况或交互路径而产生的分支场景（Alternative Scenarios）或异常场景（Exception Scenarios）。

主场景（用户名密码登录）

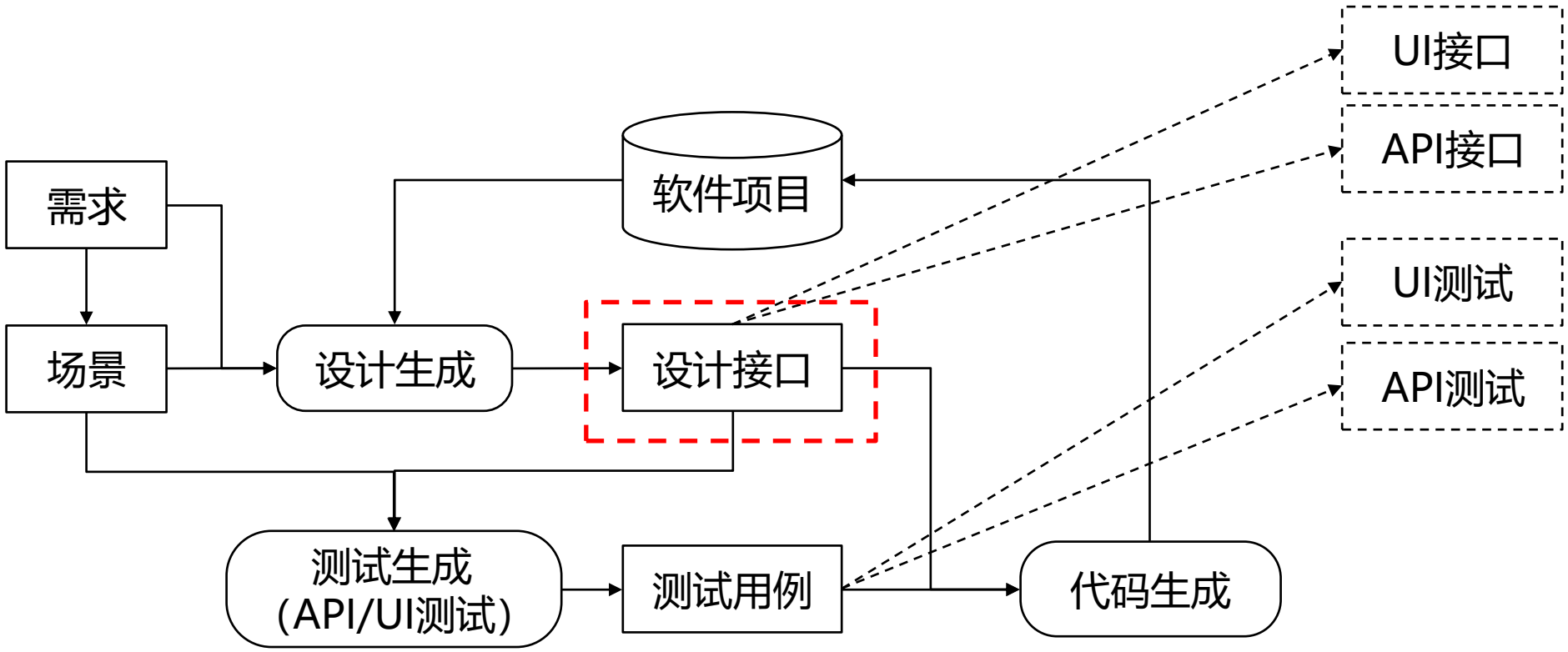
Scenario: 用户使用正确手机号和密码成功登录
Given 用户已打开淘宝登录页面
When 用户输入正确的手机号和密码并点击“登录”
Then 系统跳转到首页并显示用户名

分支场景（短信登录）

Scenario: 用户使用短信验证码登录
Given 用户已打开淘宝登录页面
And 用户选择“短信登录”选项
When 用户输入手机号并正确输入验证码
Then 系统登录成功并跳转到首页

请写一下还有哪些分支场景？

测试驱动的开发与实践


需求
场景
设计生成
软件项目
设计接口
测试生成
(API/UI测试)
测试用例
代码生成
UI接口
API接口
UI测试
API测试

软件接口

- **接口（Software Interface）**是指软件系统中不同模块或系统之间进行交互、通信和数据交换的契约（Contract）。它定义了“别人可以怎么用我”，而不暴露“我是怎么做的”。

网站系统的MVC架构

软件接口

- **接口（Software Interface）**是指软件系统中不同模块或系统之间进行交互、通信和数据交换的契约（Contract）。它定义了“别人可以怎么用我”，而不暴露“我是怎么做的”。

select * from student where student.age > 18

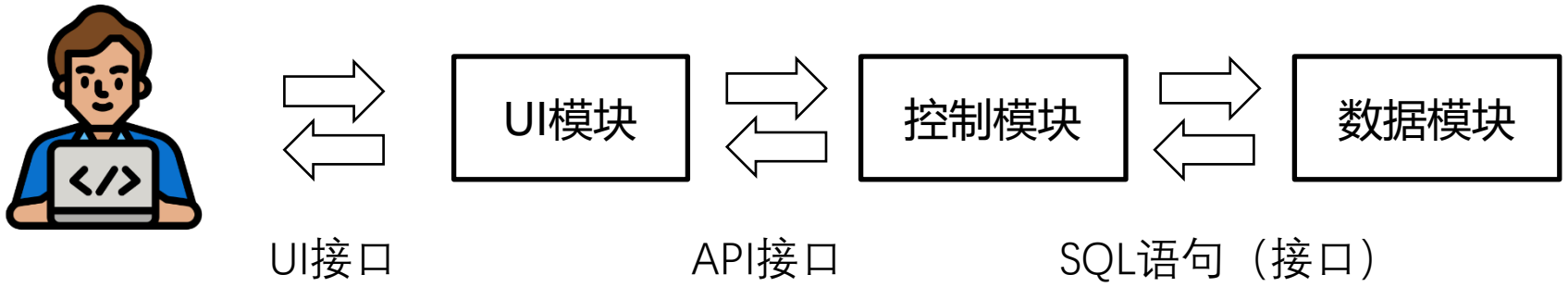
软件接口

- **接口（Software Interface）**是指软件系统中不同模块或系统之间进行交互、通信和数据交换的契约（Contract）。它定义了“别人可以怎么用我”，而不暴露“我是怎么做的”。

我们平时还有哪些软件接口的例子？

软件接口

- **接口（Software Interface）**是指软件系统中不同模块或系统之间进行交互、通信和数据交换的契约（Contract）。它定义了“别人可以怎么用我”，而不暴露“我是怎么做的”。
  - UI 接口（User Interface）
  - API 接口（Application Programming Interface）



UI接口：可视化交互界面

- 典型组成：
  - 输入控件（文本框、下拉框、按钮）
  - 输出控件（提示信息、结果区域）
- 设计目标：
  - 让用户理解并完成任务，如：“手机号输入框”、“登录按钮”、“验证码提示”
- 约束属性：
  - 字段长度、格式校验（例如手机号必须为11位）
  - 可用性（响应时间、错误提示）

API接口：系统调用的“机器级契约”

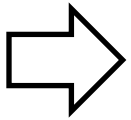
名称	用户登录接口
URL	POST /api/login
请求参数	{ "phone": "string", "password": "string" }
返回格式	JSON
响应示例	{ "status": "success", "token": "xxxxxx" }
调用约束	每个手机号一分钟内最多请求3次

从需求到接口

需求场景

Feature: 用户登录接口

Scenario: 用户使用正确的手机号和密码登录成功
Given 用户已打开登录页面
When 用户输入正确的手机号和密码并点击“登录”
Then 系统返回状态 "success" 并生成 token



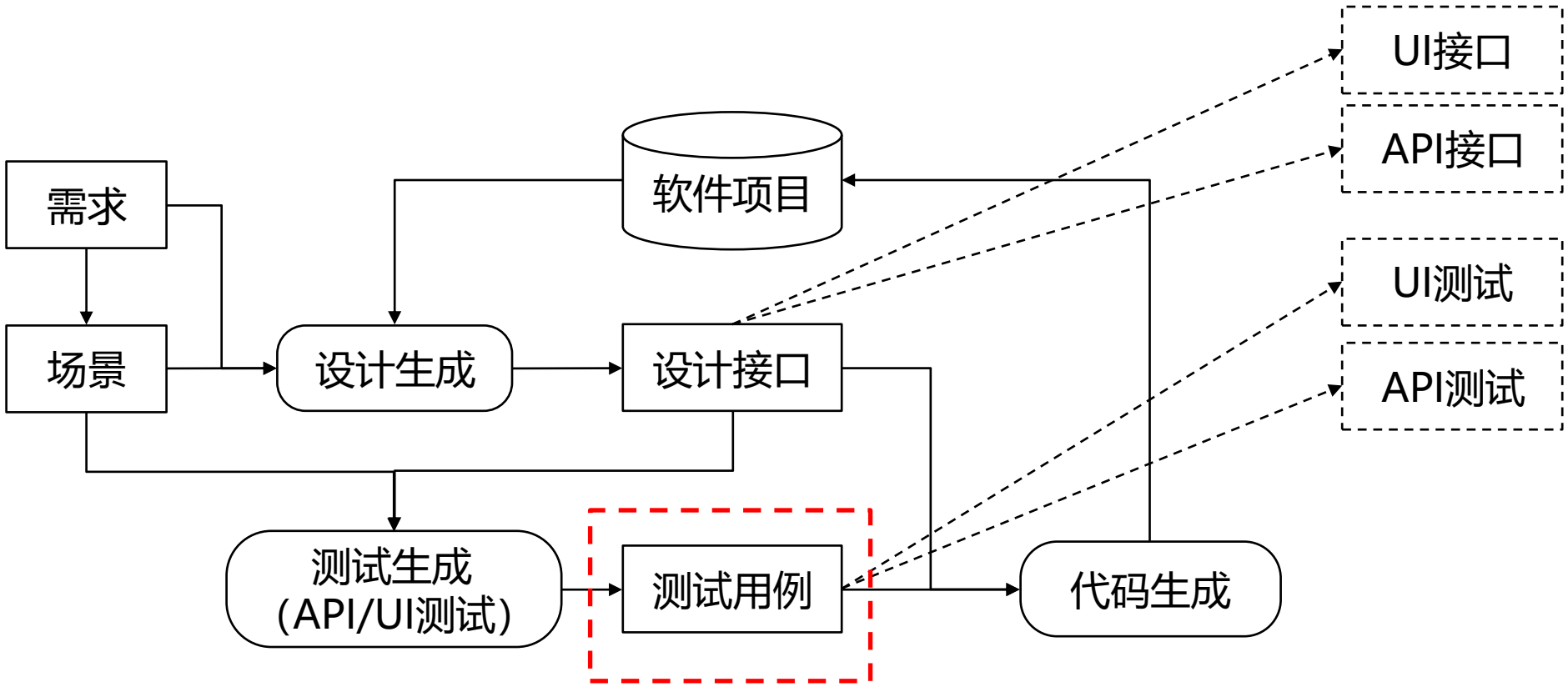
接口设计

名称	用户登录接口
URL	POST /api/login
请求参数	{ "phone": "string", "password": "string" }
返回格式	JSON
响应示例	{ "status": "success", "token": "xxxxxx" }
调用约束	每个手机号一分钟内最多请求3次

动手实践:

- 请大家打开canvas，下载requirement.md，其中包含第一个登录需求;
- 下载 课程实践流程.pdf，执行 “Task1: 从零到一，处理第一个需求” 中的第一步，从需求到接口;

测试驱动的开发与实践


需求
场景
软件项目
设计生成
设计接口
测试生成
(API/UI测试)
测试用例
代码生成
UI接口
API接口
UI测试
API测试

测试用例

- **测试用例（Test Case）**是指：为了验证一个系统或组件是否满足特定需求，而设计的一组输入、执行条件和预期结果。它是需求的可执行验证单元。

元素	说明	示例
用例编号（ID）	测试用例唯一标识	TC-LOGIN-001
用例名称	测试的目标行为	用户成功登录淘宝
前置条件（Precondition）	系统或环境状态	用户已打开登录页面
输入数据（Input）	测试所用数据	手机号=13800001111，密码=abc123
执行步骤（Steps）	操作序列	点击“登录”按钮
预期结果（Expected Result）	系统应返回的状态或输出	跳转首页并显示用户名

代码测试脚本

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class LoginServiceTest {

    @Test
    void testLoginSuccess(){
        // 假设有一个待测试的类LoginService
        LoginService service = new LoginService();

        // 输入测试数据
        String phone = "13800001111";
        String password = "abc123";

        // 调用被测方法
        String result = service.login(phone, password);

        // 验证预期结果
        assertEquals("success", result);
    }
}
```

测试输入

接口调用

测试预期（test oracle）

前沿阅读：测试生成即编辑生成

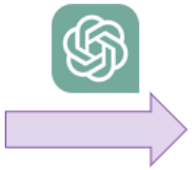
 # Objective:
Tests creating an embedded server with a custom thread pool and checks if the server starts correctly.

 # Expected Results:
1. Jetty server factory creates a server with the specified thread pool exactly once.
2. The server is initialized with the provided configurations and launched at port 6758 successfully.
3. No additional interactions with Jetty server factory occur beyond initial creation.

需求描述

```
public EmbeddedServer create(Routes rm, StaticFilesConfiguration conf, ExceptionMapper em
    , boolean han) {
  MatcherFilter filter = new MatcherFilter(rm, conf, em, han);
  filter.init(null);

  JettyHandler handler = new JettyHandler(filter);
  handler.getSessionCookieConfig().setHttpOnly(httpOnly);
  return new EmbeddedJettyServer(serverFactory, handler).withThreadPool(threadPool);
}
```

被测代码



```
- public void create_withoutHttpOnly(){
+ public void create_withThreadPool(){
+    QueuedThreadPool pool = new QueuedThreadPool();
     JettyServerFactory jFactory = mock(...);
     StaticFilesConfiguration conf = mock(...);
     ExceptionMapper map = mock(...);
     Routes routes = mock(...);

-    Server server = new Server();
+    Server server = new Server(pool);

-    when(jFactory.create(100, 10, 10000)).thenReturn(server);
+    when(jFactory.create(pool)).thenReturn(server);

-    EmbeddedJettyFactory eFactory = new EmbeddedJettyFactory(jFactory).withHttpOnly(false);
+    EmbeddedJettyFactory eFactory = new EmbeddedJettyFactory(jFactory).withThreadPool(pool);

     embeddedServer = eFactory.create(routes, conf, map, false);
     embeddedServer.trustForwardHeaders(true);

-    embeddedServer.ignite("localhost", 6759, null, 100, 10, 10000);
+    embeddedServer.ignite("localhost", 6758, null, 0, 0, 0);

-    assertFalse(server.getHandler().getSessionCookieConfig().isHttpOnly());
+    verify(jFactory, times(1)).create(pool);
+    verifyNoMoreInteractions(jFactory);
```

Qi, Binhang, Yun Lin, Xinyi Weng, Yuhuan Huang, Chenyan Liu, Hailong Sun, and Jin Song Dong. "Intention-Driven Generation of Project-Specific Test Cases." *arXiv preprint arXiv:2507.20619* (2025).

GUI测试

- **GUI测试（Graphical User Interface Testing）**是指对应用程序的图形化界面（按钮、输入框、菜单、窗口等）进行测试，以确保用户在操作界面时，系统表现出符合预期的行为。

步骤	操作	预期结果
1	打开登录页面	页面加载完成，显示输入框与按钮
2	在手机号输入框输入“13800001111”	输入框显示正确内容
3	在密码输入框输入“abc123”	密码以掩码形式显示
4	点击“登录”按钮	页面跳转到首页并显示用户名
5	若输入错误密码	页面显示“用户名或密码错误”提示

GUI测试

步骤	操作	预期结果
1	打开登录页面	页面加载完成，显示输入框与按钮
2	在手机号输入框输入“13800001111”	输入框显示正确内容
3	在密码输入框输入“abc123”	密码以掩码形式显示
4	点击“登录”按钮	页面跳转到首页并显示用户名
5	若输入错误密码	页面显示“用户名或密码错误”提示

GUI测试脚本

```
WebDriver driver = new ChromeDriver();
driver.get("https://taobao.com/login");

driver.findElement(By.id("phone")).sendKeys("13800001111");
driver.findElement(By.id("password")).sendKeys("abc123");
driver.findElement(By.id("loginButton")).click();

String title = driver.getTitle();
assertEquals("淘宝首页", title);
driver.quit();
```

GUI测试脚本

```
WebDriver driver = new ChromeDriver();
driver.get("https://taobao.com/login");

driver.findElement(By.id("phone")).sendKeys("13800001111");
driver.findElement(By.id("password")).sendKeys("abc123");
driver.findElement(By.id("loginButton")).click();

String title = driver.getTitle();
assertEquals("淘宝首页", title);
driver.quit();
```

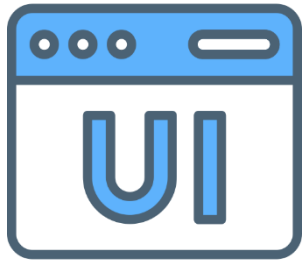
这种做法有什么弊端?

GUI智能体方案


7:22
100%
持仓详情-策略分析
持仓收益率
目标收益率
暂无数据
买入/定投点
卖出点
当日同时存在买和卖的情况
关联指数
回测详情
中证内地低碳经济主题指数 000977
估值PB
2.37
当前回撤
0.53
估值水平
低估
回撤风险
极低
定投计划
每两周周一14:30定投1000.00元
监控中
每两周周五14:30定投10000股
监控中
常见问题
风险提示
买入
卖出
终止

GUI智能体：静态需求到动态操作的映射智能体

界面操作
描述&目标



界面截图

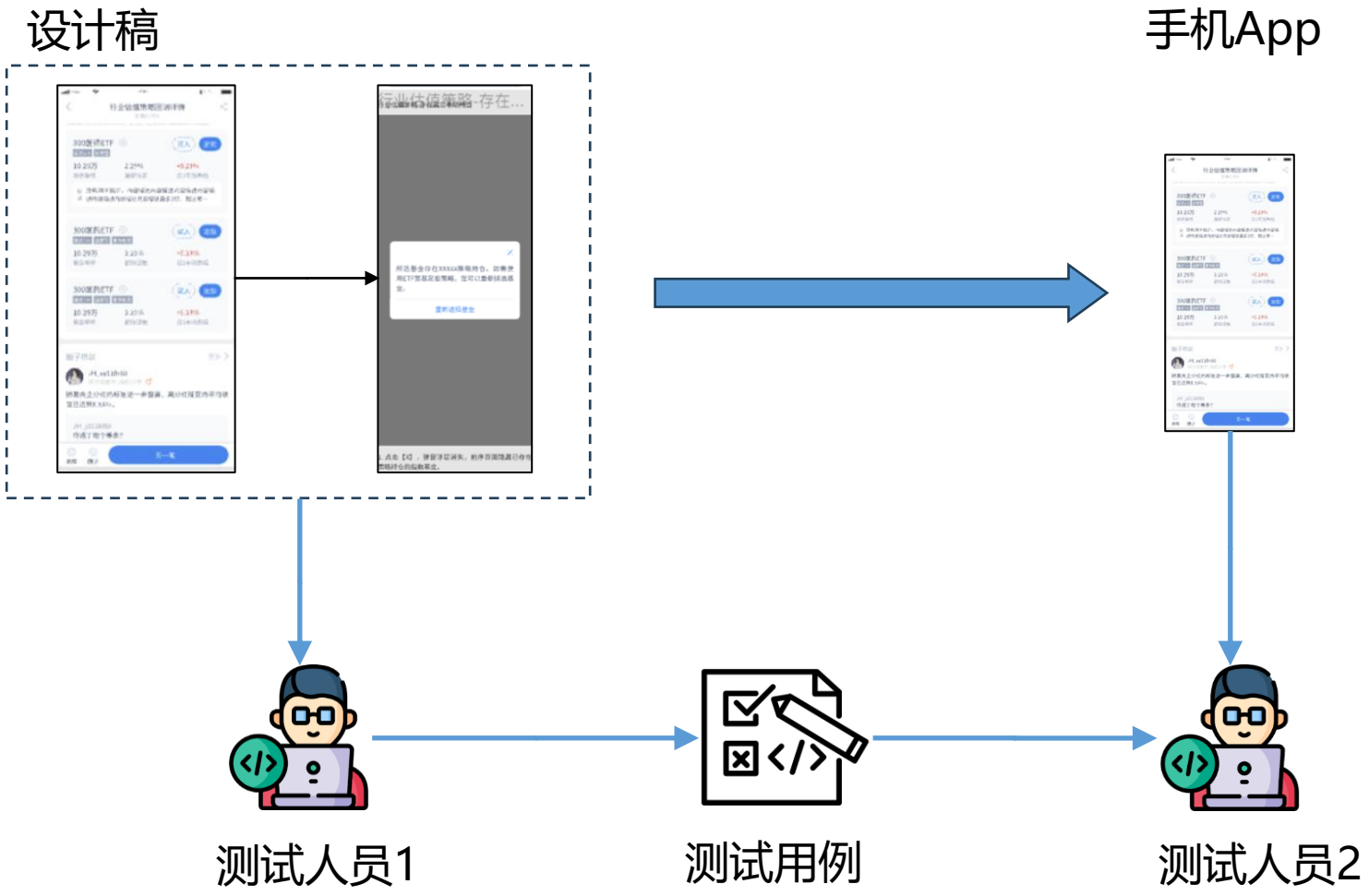
GUI智能体

界面操作序列

步骤1：在坐标（x1,y1）上做出点击操作

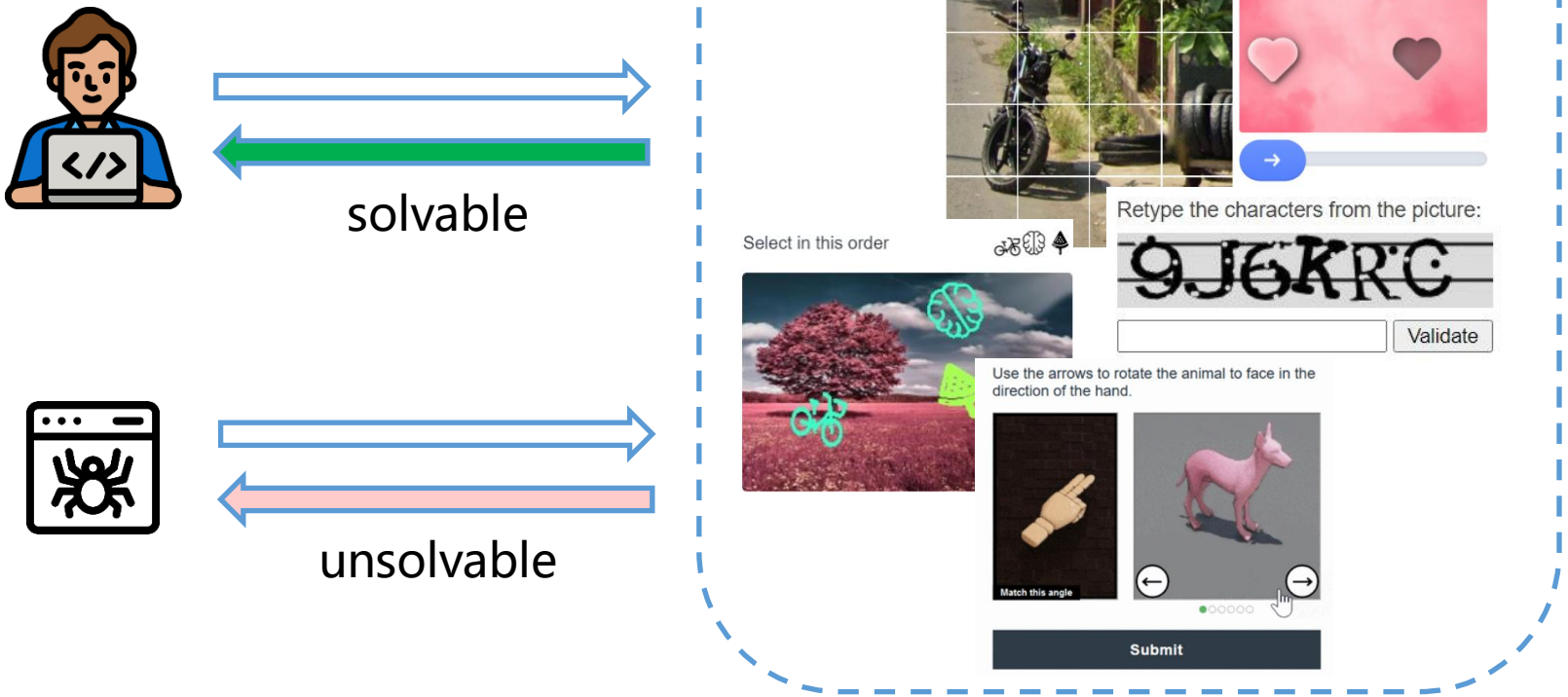
步骤2：在坐标（x2,y2）上做出拖拽操作

…… ……



相关文献阅读

Ruofan Liu[#], Xiwen Teoh[#], Yun Lin[*], Guijie Chen, Ruofei Ren, Denys Poshyvanyk, Jin Song Dong. GUIPilot: A Consistency-based Mobile GUI Testing Approach for Detecting Application-specific Bugs. (**ISSTA 2025**)



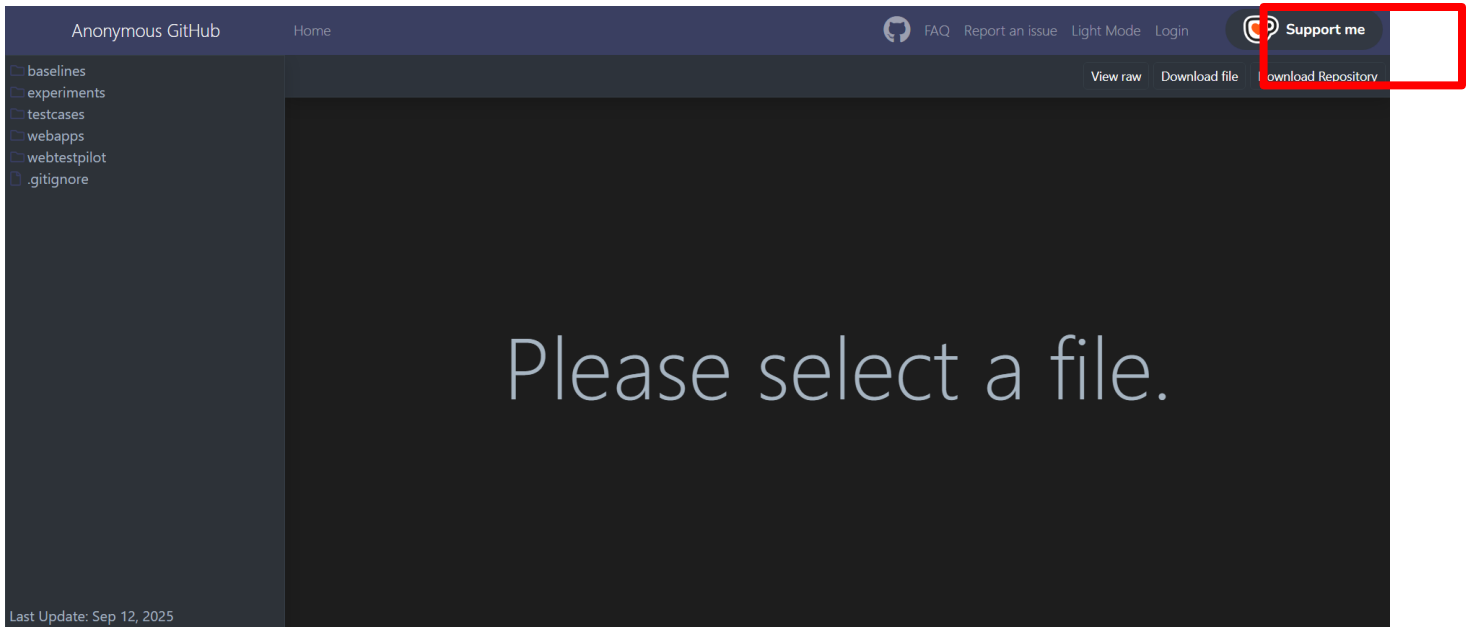
相关文献阅读

Xiwen Teoh, Yun Lin, Siqi Li, Ruofan Liu, Avi Sollomoni, Yaniv Harel, Jin Song Dong. Are CAPTCHAs Still Bot-hard? Generalized Visual CAPTCHA Solving with Agentic Vision Language Model. (**USENIX Security 2025**)

课后推荐操作：利用GUI智能体来完成操作脚本

安装步骤

- 前往 https://anonymous.4open.science/r/WebTestPilot-E701
- 点击左上角 “Download Repository”



安装步骤

- unzip WebTestPilot-E701.zip
- cd WebTestPilot-E701/webtestpilot
- rm uv.lock
- uv sync
- 在 /webtestpilot/baml_src/clients.baml
  和 /webtestpilot/src/config.yaml 里配置环境
- uv run baml-cli generate

```
2025-10-12T22:50:23.424 [BAML INFO] Generating clients with CLI version: 0.205.0
2025-10-12T22:50:23.452 [BAML INFO] Wrote 13 files to src\baml_client
2025-10-12T22:50:23.453 [BAML INFO] Generated 1 baml_client: ../src\baml_client
```

使用步骤

- 在 /webtestpilot 下创建一个新的 python 脚本
- 执行并开始测试

```
from playwright.sync_api import sync_playwright
from baml_client.types import Step
from src.main import WebTestPilot

test_steps = [Step(condition="…", action="…", expectation="…"), …]
   with sync_playwright() as p:
      browser = p.chromium.launch(headless=True)
      context = browser.new_context()
      context.tracing.start(screenshots=True, snapshots=True, sources=True)
      page = context.new_page()
      page.goto(…)

      config = Config.load("config.yaml")
      session = Session(page, config)
      WebTestPilot.run(session, test_steps)

      context.tracing.stop(path="trace.zip")
      context.close()
      browser.close()
```

使用步骤

导入工具

定义测试步骤

访问目标网站

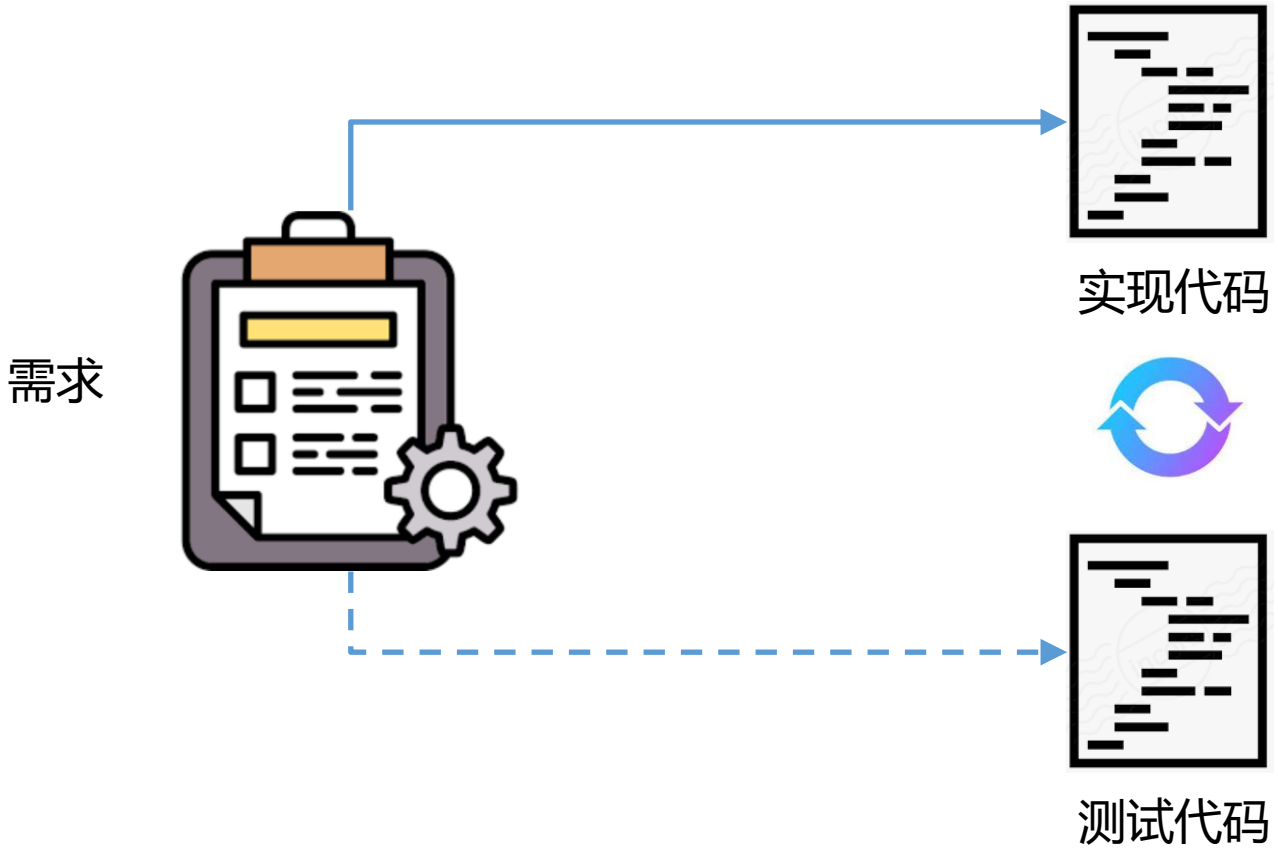
开始测试

```
from playwright.sync_api import sync_playwright
from baml_client.types import Step
from src.main import WebTestPilot

test_steps = [Step(condition="…", action="…", expectation="…"), …]
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        context = browser.new_context()
        context.tracing.start(screenshots=True, snapshots=True, sources=True)
        page = context.new_page()
        page.goto(…)

        config = Config.load("config.yaml")
        session = Session(page, config)
        WebTestPilot.run(session, test_steps)

        context.tracing.stop(path="trace.zip")
        context.close()
        browser.close()
```

思考：GUI脚本测试和GUI智能体测试之间的优劣？

这种做法有什么弊端？

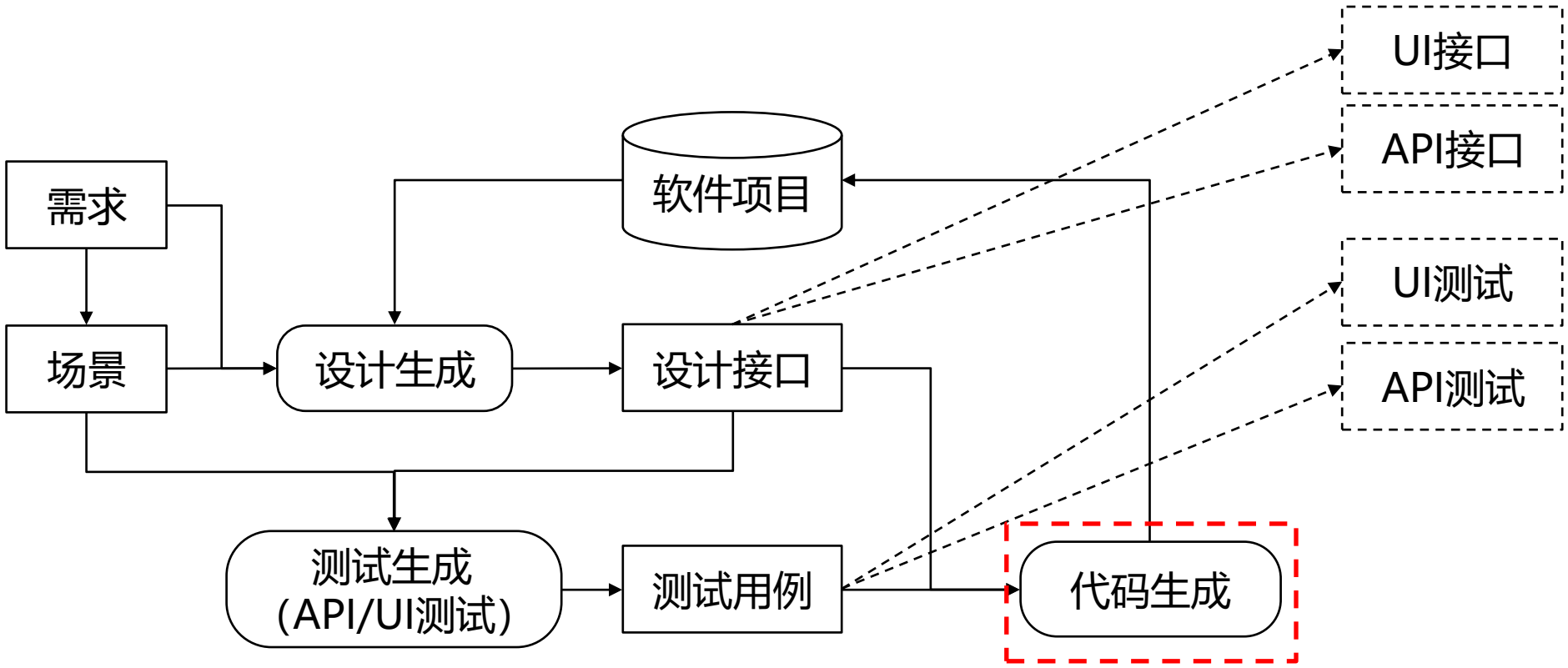
AI时代的测试趋势：信息交叉验证



动手操作

- 请下载canvas中的代码，利用Trae IDE来根据需求生成测试
- 参考 课程实践流程.pdf中 **Task 1** 中的第二步，生成测试

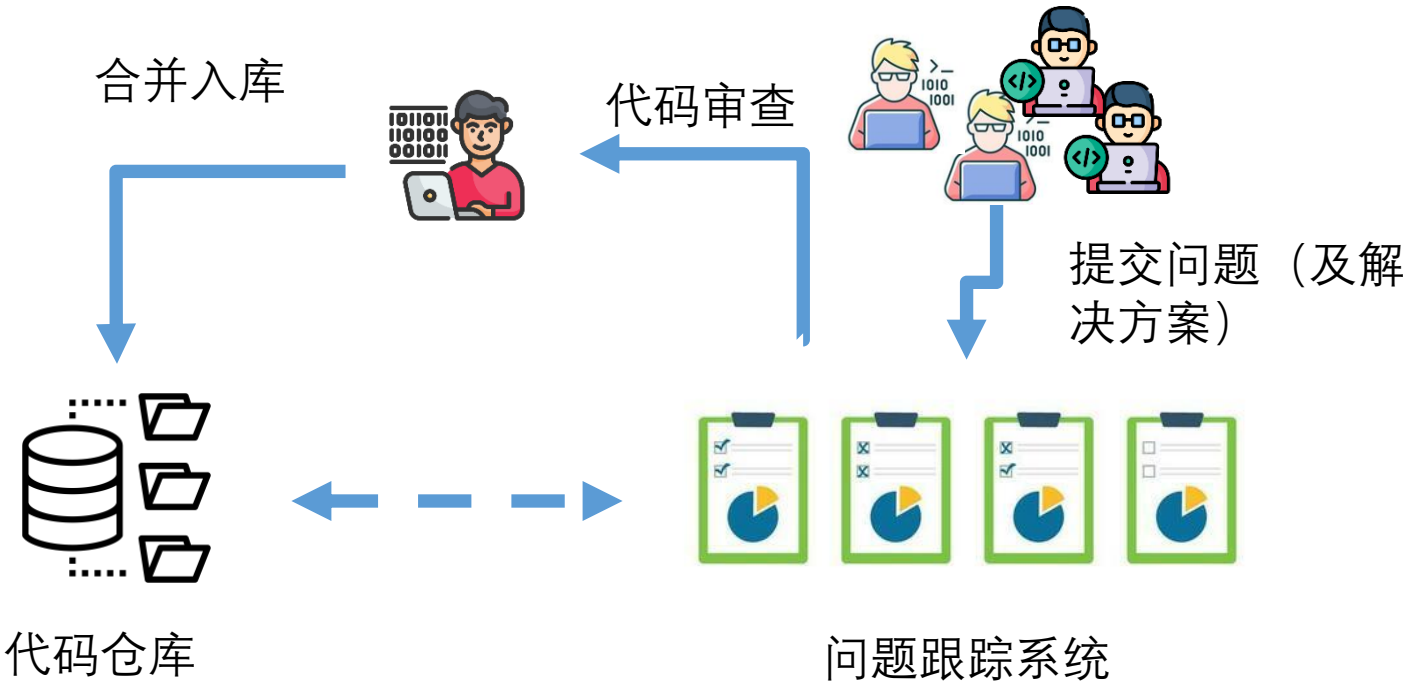
测试驱动的开发与实践


UI接口
API接口
UI测试
API测试
软件项目
需求
场景
设计生成
设计接口
测试生成
(API/UI测试)
测试用例
代码生成

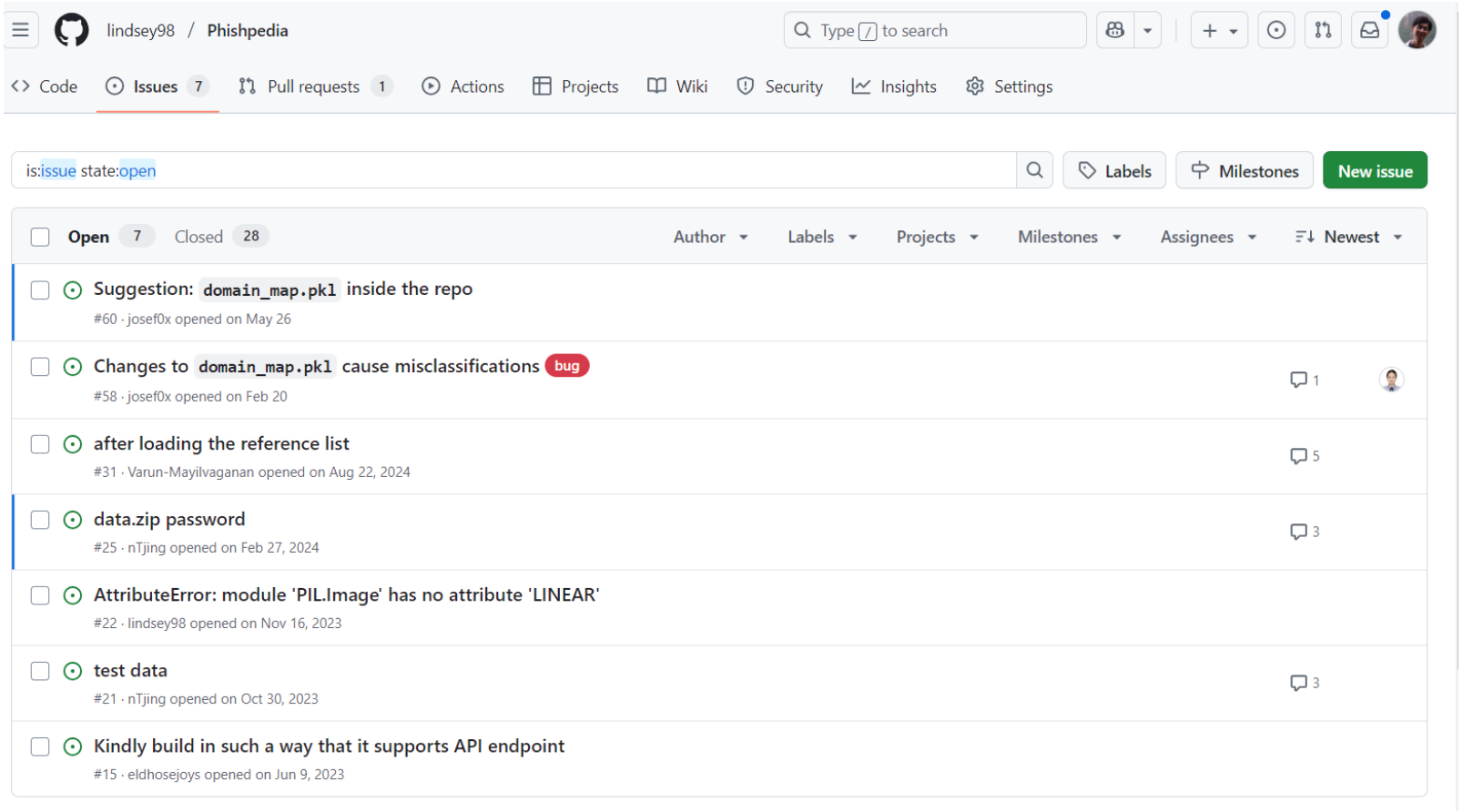
动手操作

- 请下载canvas中的代码，利用Trae IDE来生成代码
- 参考 课程实践流程.pdf中 **Task 1** 中的第三步，生成代码

Github使用

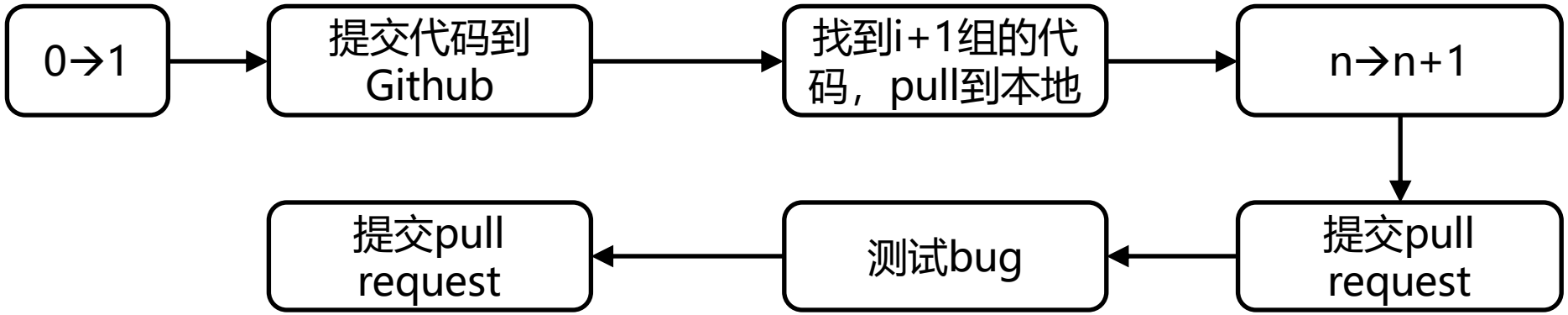


问题追踪系统（issue tracking system）



实践环节

- 从0到1：从需求到代码的端到端生成
- 从n到n+1: 增量需求添加
- 测试环节：发现项目中的bug
- 参考canvas上 课程实践流程.pdf 中的Task2(上传代码)与Task3(跨组协作与增量开发)



总结

- 理解和掌握需求、需求场景、软件设计、软件测试、和代码编写的概念
- 理解测试驱动开发的概念
- 基于AI编程来基于需求实践测试驱动的开发