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

AI Agents在软件测试中的落地实践
智能化测试新时代

王哲 | 百度

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

王哲

百度资深测试工程师

2017年毕业加入百度，先后负责百度基础架构、AI 产品等多个核心业务的质量保障工作、并负责百度技术中台群组测试环境治理等工作、目前主要负责 AI 驱动的智能化测试解决方案的设计与实施落地。

在测试智能化、自动化测试、测试工具建设以及云原生等方面有着深刻的见解和丰富的实践经验。

目录
CONTENTS

PART 01

智能化测试的重要性与挑战

软件测试智能化的重要性

1. 互联网企业精细化发展，降本增效 & 效率提升是核心任务

2. 软件复杂度上升，研发模式变革(低代码、零代码、AI生成)，传统测试模式容易成为整个研发环节的效率瓶颈

3. ChatGPT 等生成式 AI 能力高速发展，给软件测试变革带来了巨大的机遇。

测试智能化过程遇到的问题

QA 与 RD 工作的对比

- 虽然看起来**单项任务难度相对更低**，但是**涉及面更多更广**。
- 想要辅助生成部分代码&用例设计相对更简单，但是想要彻底完成某项任务其实更难。

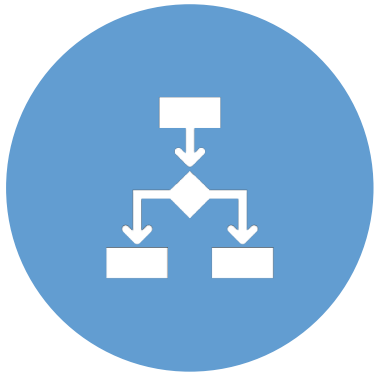
Copilot模式在测试场景的应用

- Copilot 模式在测试场景中接受度低（例如在接口测试场景，输入详细的测试步骤生成初版代码，人工进行细节调整后再进行调试执行）
- 核心原因是测试场景单项任务复杂度较低，仅做代码生成、续写或部分工作无法满足需求，不如直接参考历史用例进行复制粘贴并修改。在收益不明显的情况下，改变习惯的工作模式难度大。
- 因此，需要端到端带来足够的变化+收益才能快速应用落地

AI 现阶段彻底替代 QA 的可行性

- 由于测试场景的复杂性，现阶段让 AI 彻底替代 QA 不现实，需要人与 AI 协作才能取得更好的效果
- 协作模式可以是 **AI 完成基础版本，人来 Review 结果并通过人工指导或少量调整**达到希望结果
- 这就要求 AI 来适配人的习惯，生成的基础版本尽可能与人的期望一致

AI Agents 在测试领域的应用场景



用例设计更加准确

即更理解业务背景、测试场景覆盖更全、
生成用例设计更加符合QA书写习惯

接口自动化端到端完成测试任务

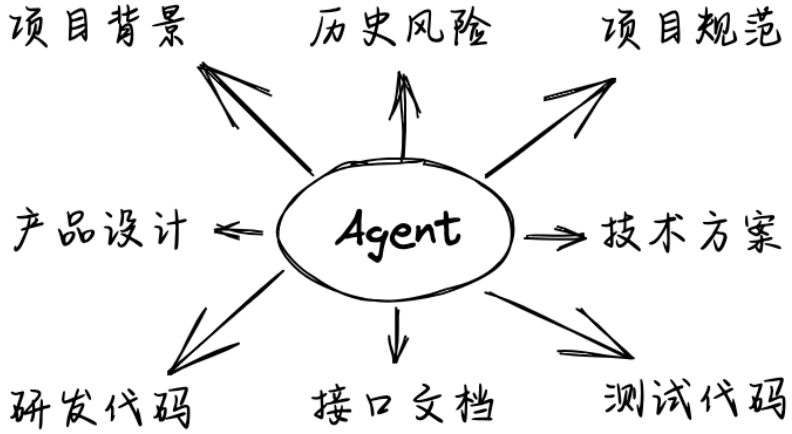
打通生成、合并、执行、修复、提交等环节

WEB UI 测试智能化

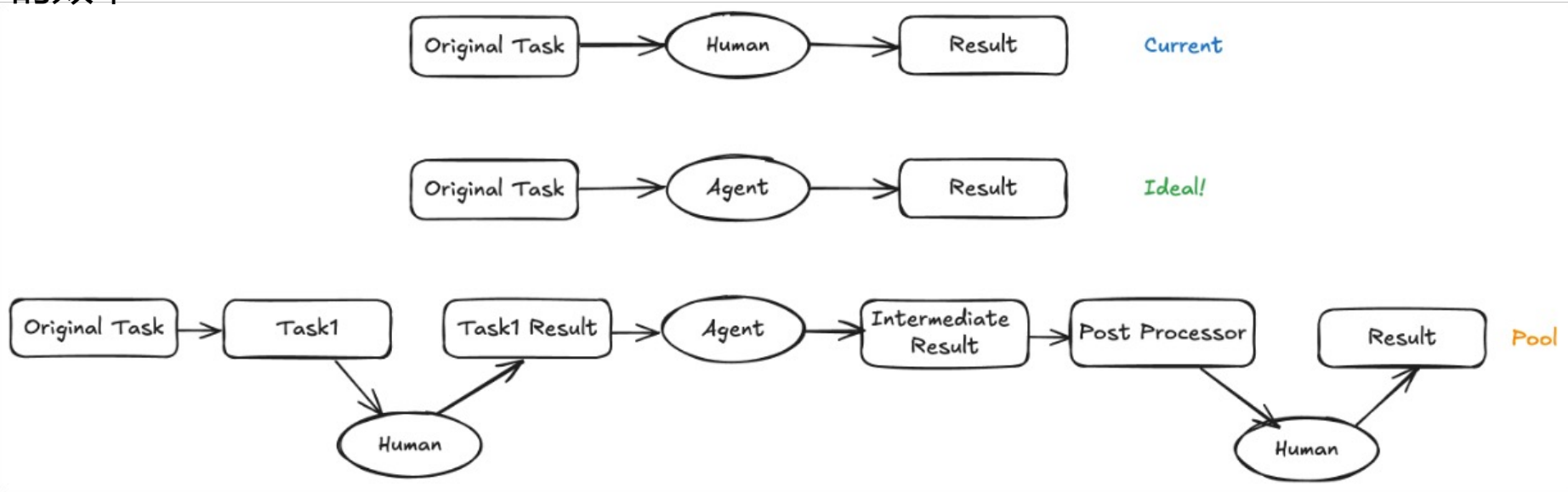
基于用例设计快速实现 WEB UI 智能化测试，
包括智能测试 & 自动化代码生成与维护

AI Agents 需要解决的核心问题

一、QA作为对整个项目全局最了解的人，如何让 AI Agent 对项目有足够了解，从而生成的内容更准



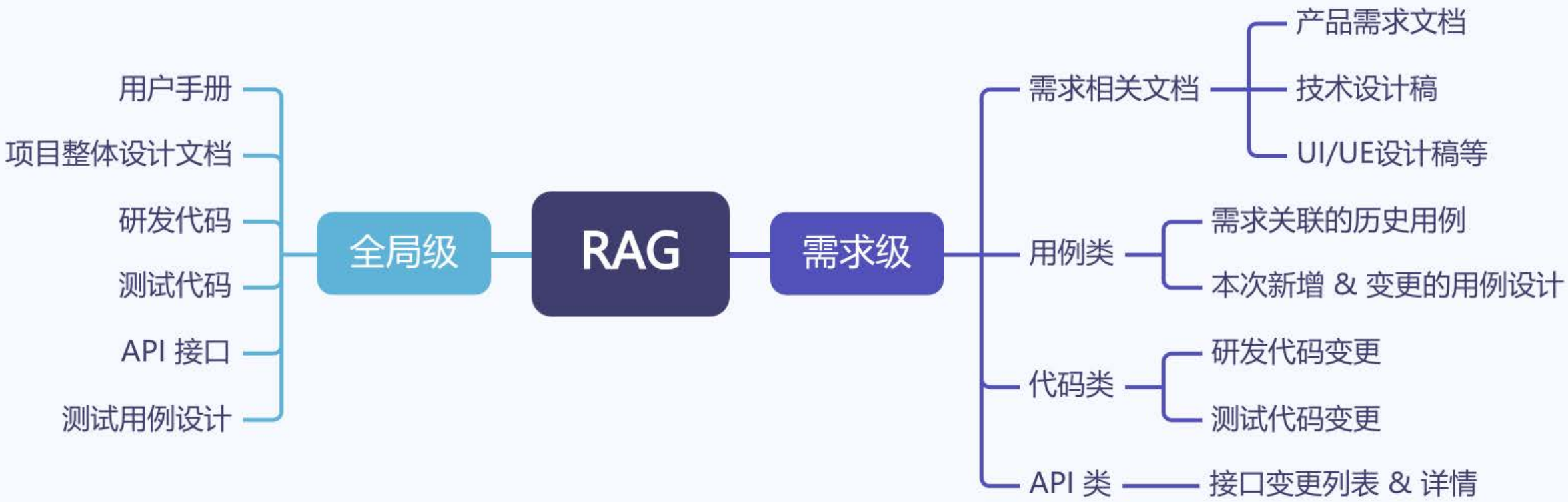
二、如何让 Agent 独立完成某类工作，避免频繁的进行任务A->任务B->任务C的任务转化，从而有效提升QA工作的效率



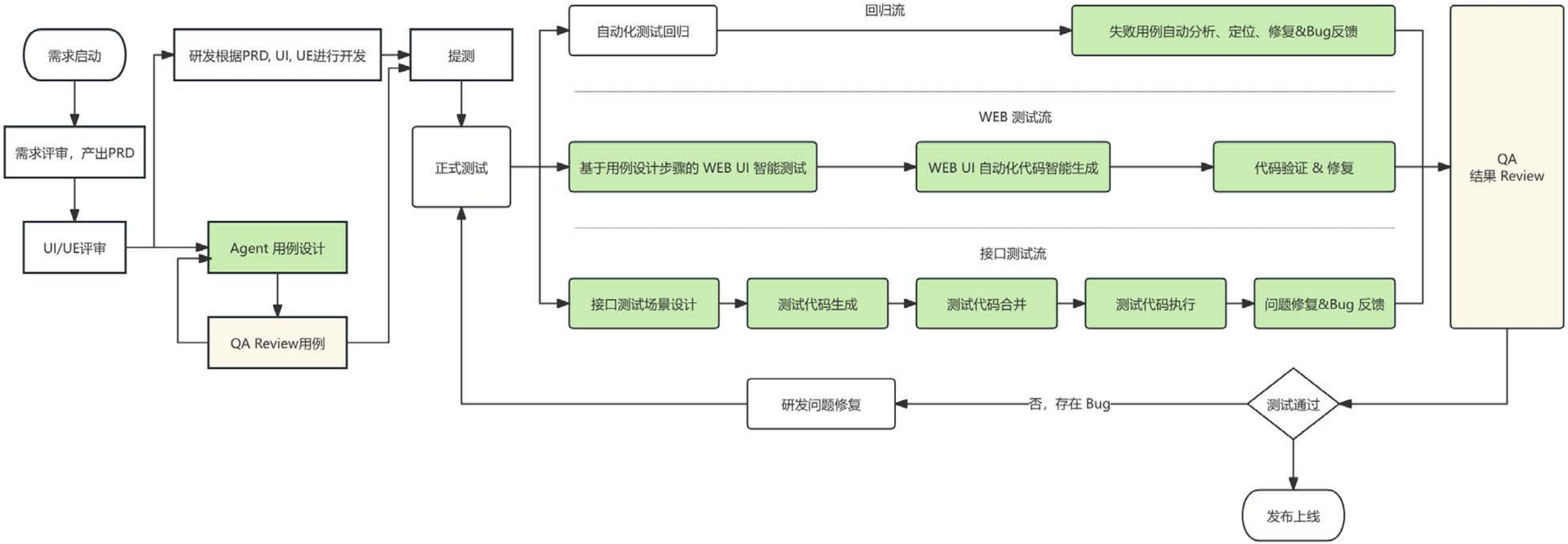
PART 02

智能化测试整体架构

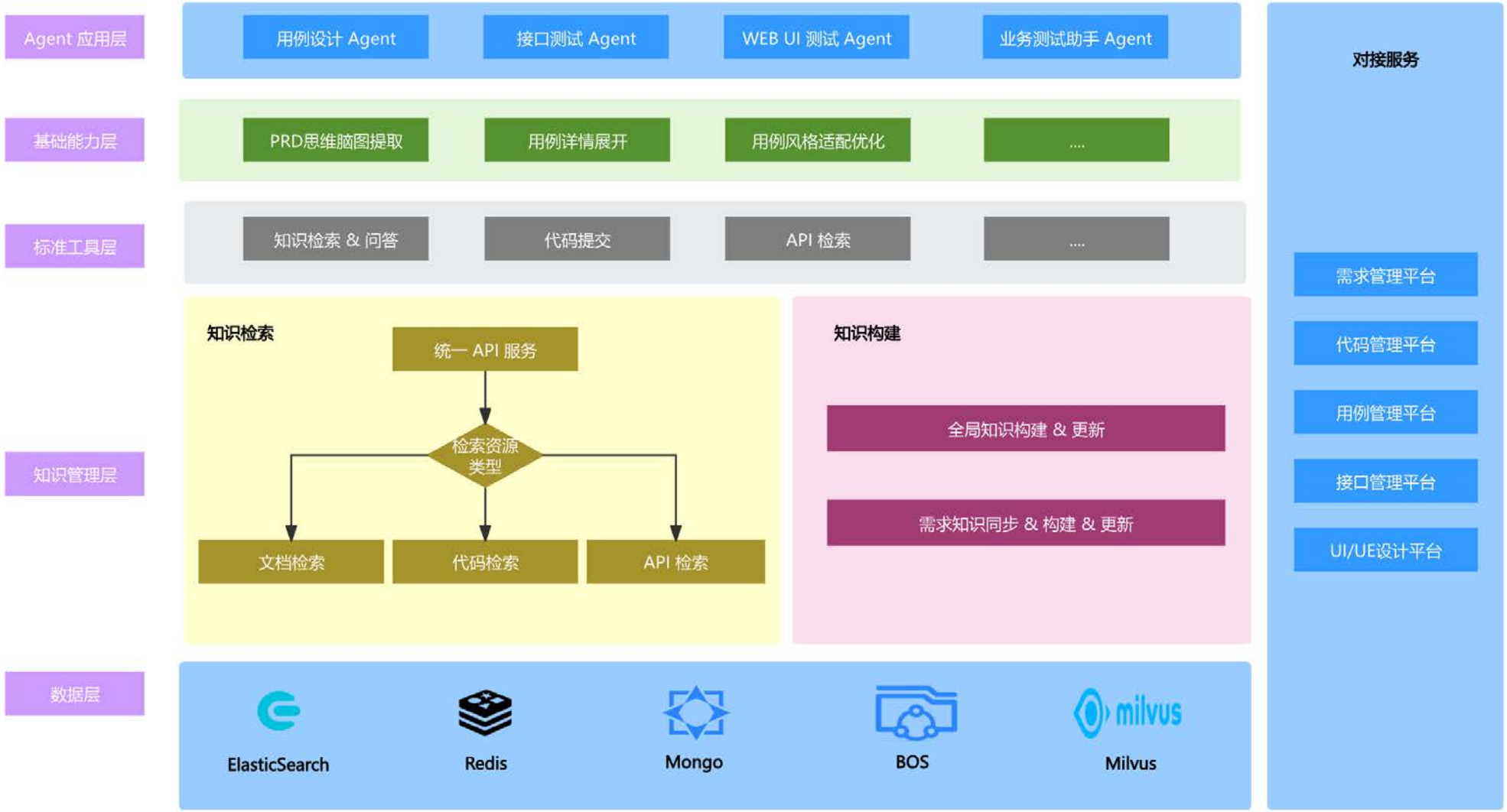
AI Agents 需要理解哪些业务知识



AI Agents加持的智能化测试工作流



智能化测试系统架构图


Agent 应用层
用例设计 Agent
接口测试 Agent
WEB UI 测试 Agent
业务测试助手 Agent
基础能力层
PRD思维脑图提取
用例详情展开
用例风格适配优化
....
标准工具层
知识检索 & 问答
代码提交
API 检索
....
知识管理层
知识检索
统一 API 服务
检索资源类型
文档检索
代码检索
API 检索
知识构建
全局知识构建 & 更新
需求知识同步 & 构建 & 更新
数据层
ElasticSearch
Redis
Mongo
BOS
Milvus
对接服务
需求管理平台
代码管理平台
用例管理平台
接口管理平台
UI/UE设计平台

PART 03

AI Agents 在测试环节的应用实践

AI Agents 需要理解业务哪些内容

入职第一周:

1. 了解系统的基本概念 & 功能范畴
2. 作为用户熟悉整个系统的功能使用
3. 了解当前系统的测试流程 & 规范
4. 熟悉测试自动化编写 & 执行规范等

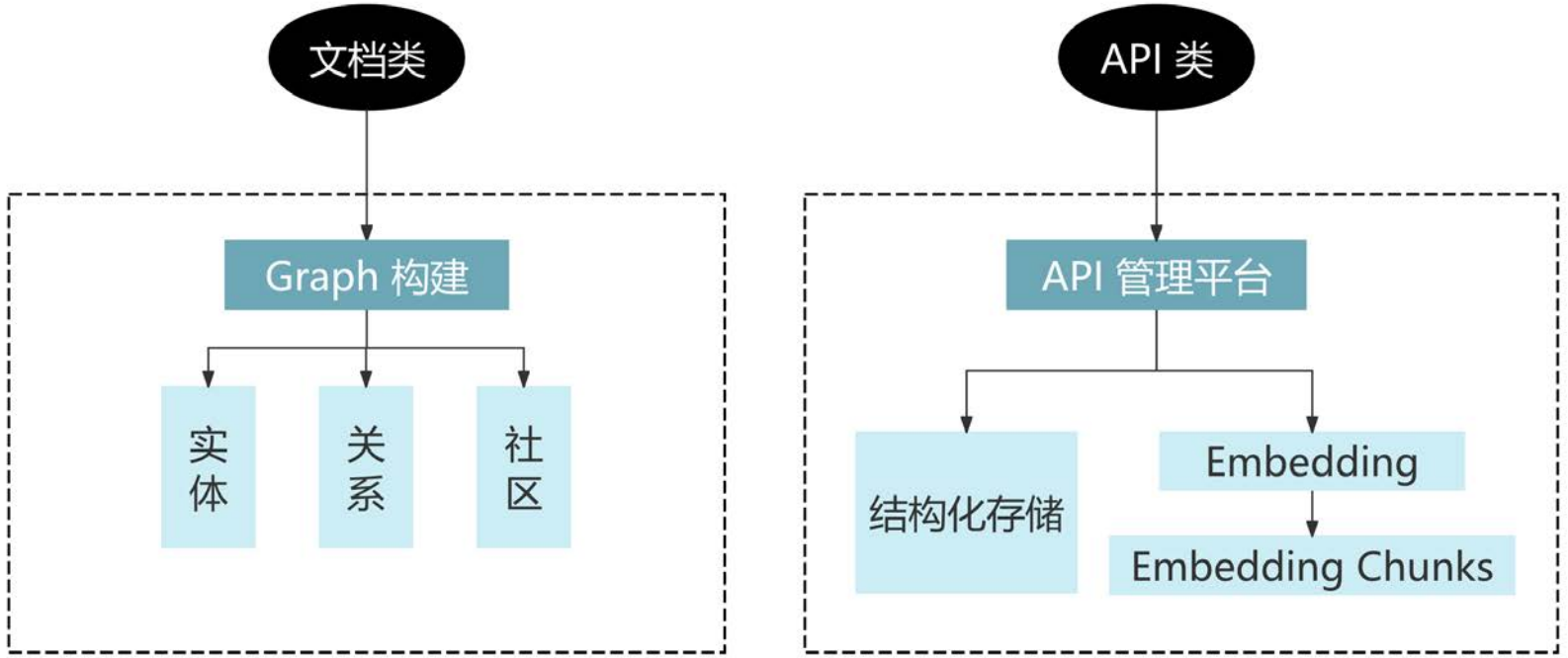
全局知识
工作正常启动的前提
在日常测试的各个环节中体现

日常迭代测试:

1. 参与需求评审，了解需求迭代功能
2. 阅读 PRD、UI/UE、技术方案等全面了解本次变更
3. 根据研发接口文档设计 & 编写接口自动化测试
4. 通过 CodeReview 了解本次变更影响面

需求级知识
日常迭代过程中核心输入
不同的类型的知识在不同测试环节中的依赖程度有一定差异

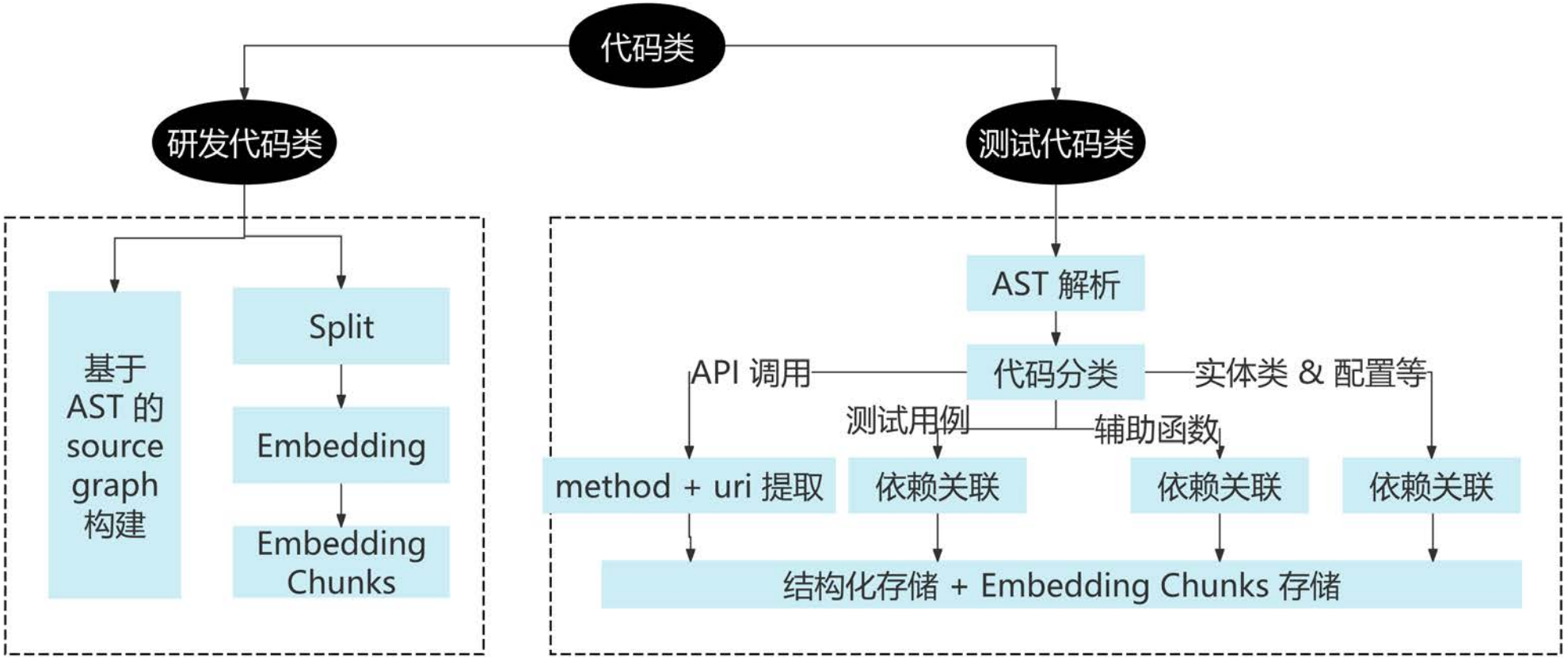
全局知识构建(文档/API)



文档类: 包含纯文本文档和图片类，其中图片类先通过 OCR 和多模态模型理解转为文字类，整体通过 GraphRAG 进行索引构建并用于后续检索增强，从而提升检索准确度

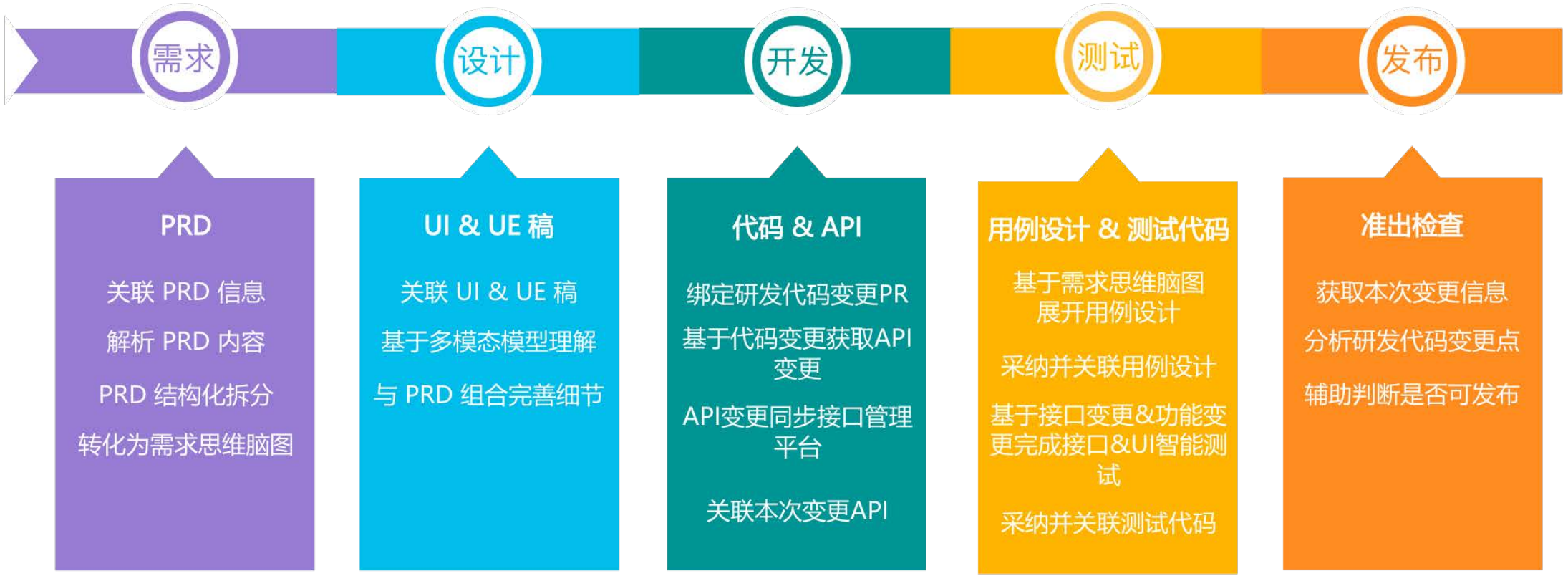
API 类: 统一在 API 管理平台纳管，底层提供结构化存储和向量化存储两种能力，支持结构化检索和语义检索

全局知识构建(代码)



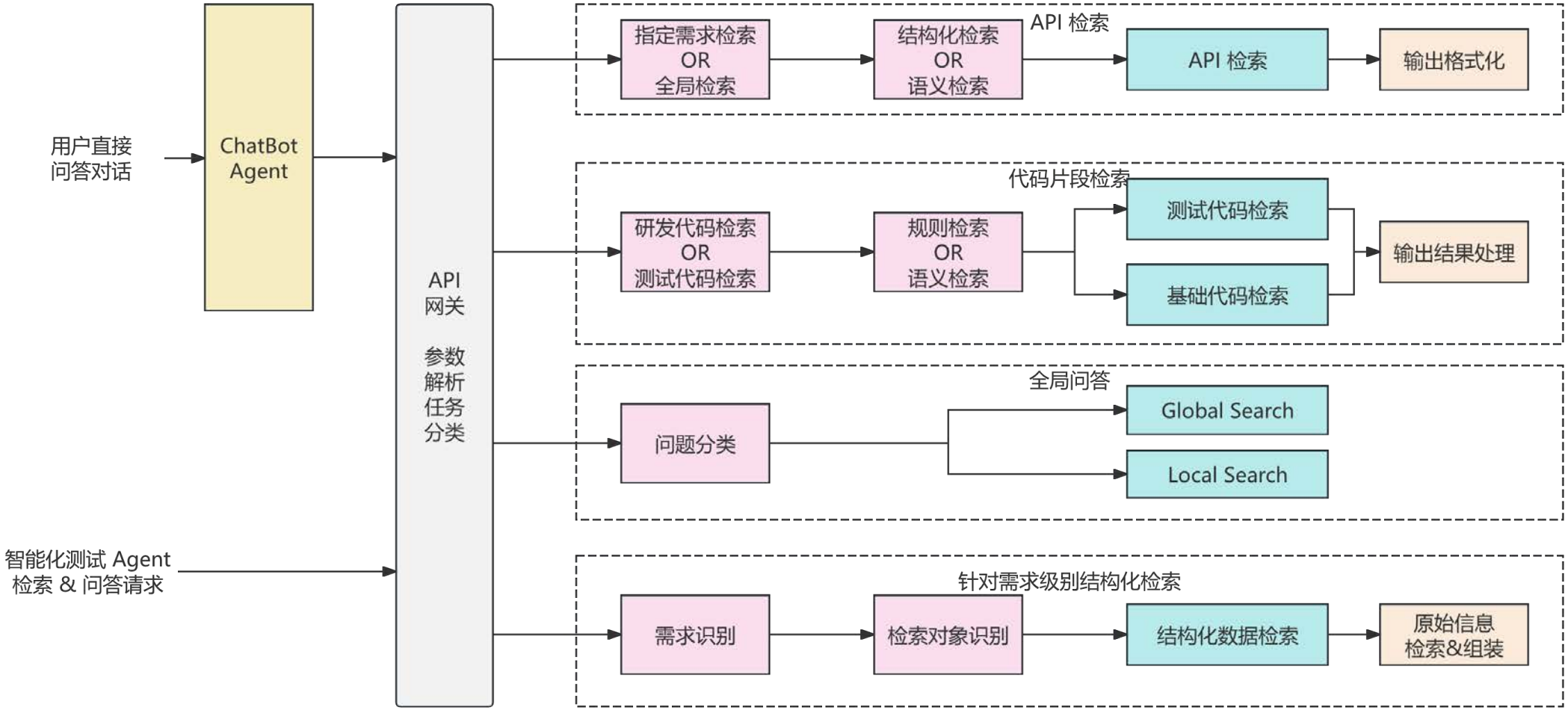
代码解析与 API 类似，代码管理平台默认支持了基于 AST 解析的 SourceGraph 检索能力和基于 embedding 的语义检索能力，同时，结合测试代码库的典型特征，我们进一步对测试代码库的代码段进行分类，并针对其中 API 调用的代码进行了结构化信息的提取。

需求级知识构建



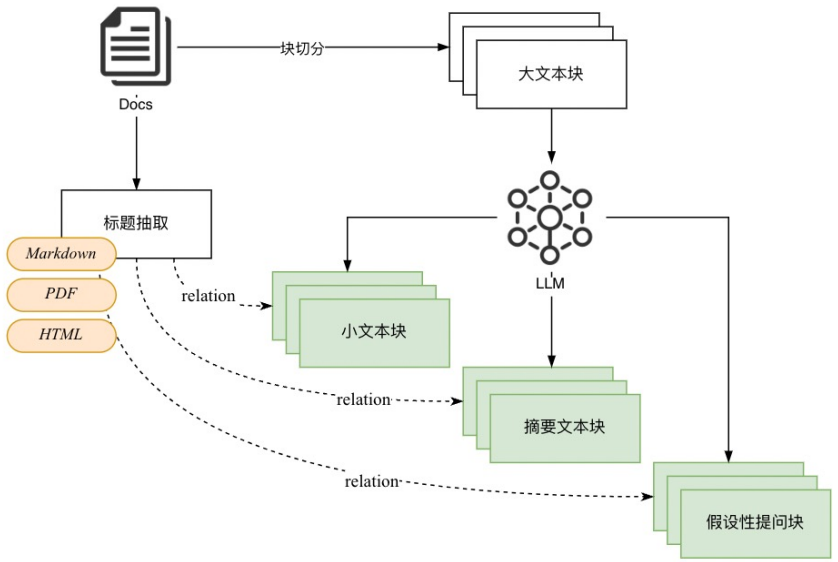
与全局知识不同，需求级知识往往是本次测试任务中各个环节的核心输入，且上下文内容有限。
因此，在数据的处理上更多的是对知识的关联绑定，便于对相关数据的规则检索；此外，针对 UI&UE 稿等图片信息，与全局知识类似，也会进行一些预处理，包括图片理解等操作；同时，针对 PRD 文档会进行一些结构化拆分，例如思维脑图转化，用于后续用例设计等环节。

知识检索与应用能力

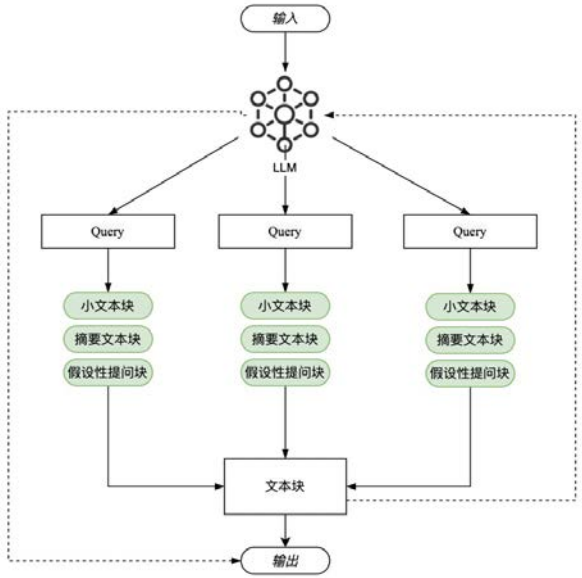


我们建设了统一的检索服务打通各类知识检索，通过统一检索&问答 API 来面向上层的各类测试场景智能体提供服务；同时，针对一些简单问答场景，也提供了开箱即用的 ChatBot，可以针对项目知识、需求迭代细节等内容提供问答服务。

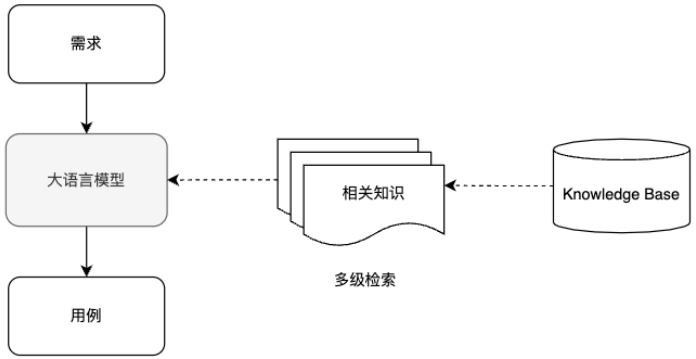
传统智能化用例设计的局限



多级检索 Chunks 构建



多级 Chunks 检索



需求 + 多级检索 Chunks 生成用例设计

局限：

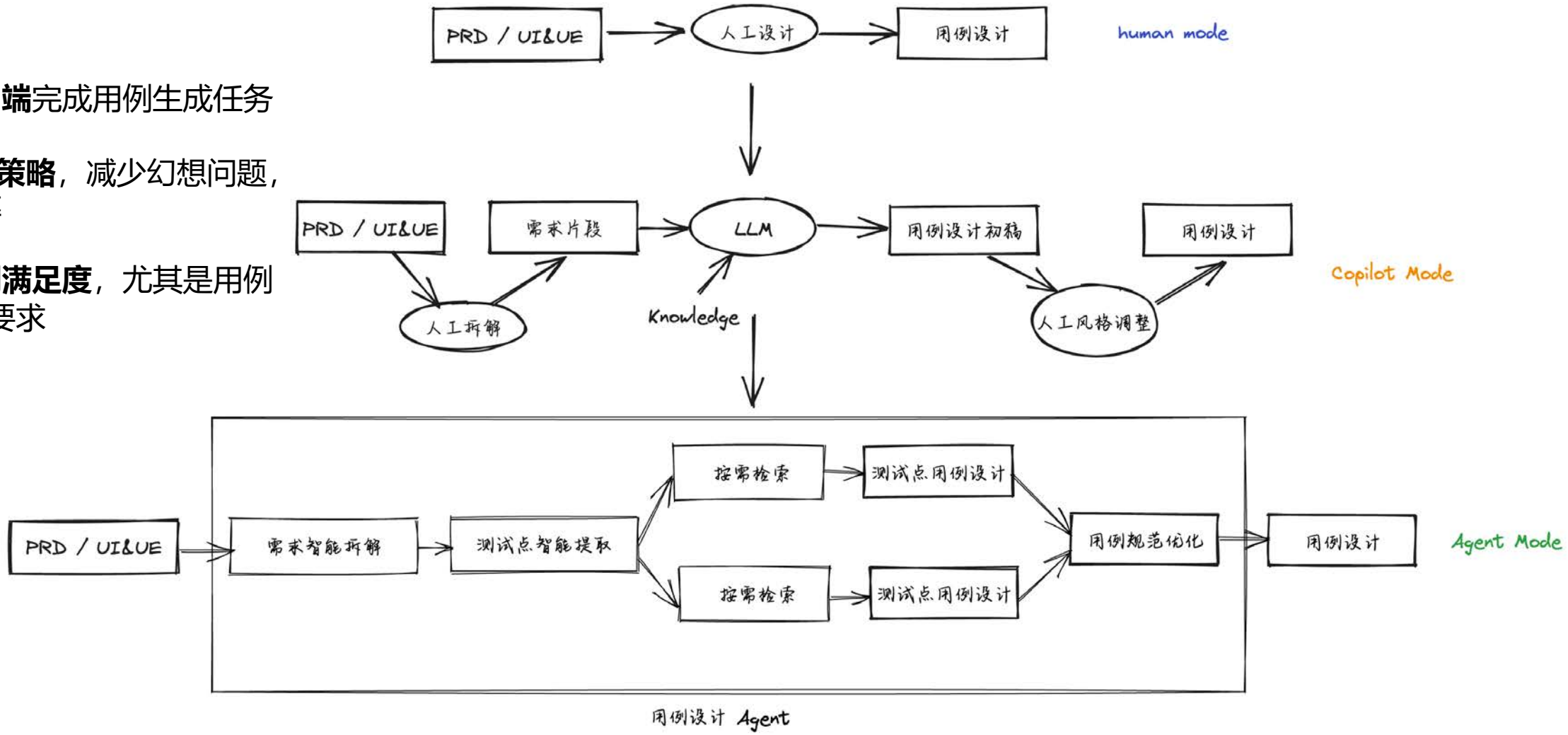
1. 多级 Chunks 检索后得到的原始片段作为模型输入的一部分，存在导致模型幻想发挥的可能；
2. 针对图片类的信息无法有效利用，导致生成用例不完整；
3. 需求 -> 用例的单步推理模式很难控制用例拆解的粒度，通常需要人工介入提前把完整需求拆分成若干个子需求；
4. 用例生成符合业务规范难以保证，绝大部分场景需要人工介入对结果进行调整；

用例设计Agent目标

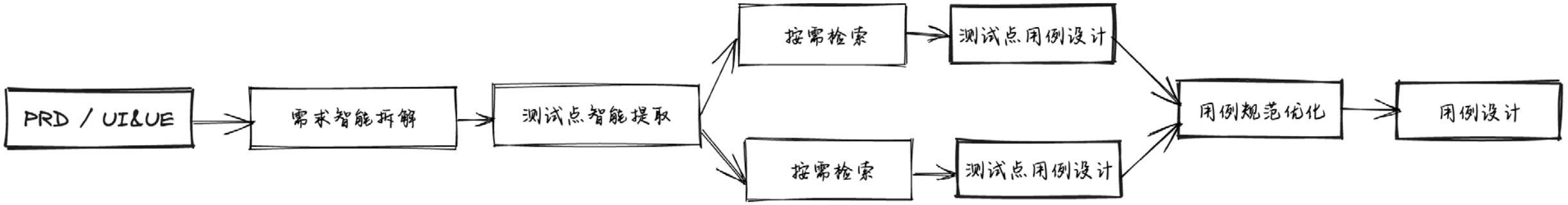
一、尽可能**端到端**完成用例生成任务

二、**优化 RAG 策略**，减少幻想问题，提升生成准确率

三、**提升业务侧满足度**，尤其是用例规范 & 风格等要求



用例设计Agent Workflow设计



需求智能拆解

将非结构化的 PRD 等文档经过模型理解，转化为结构化的思维脑图

测试点智能提取

为思维脑图中提取出一组相对独立的测试点，每个测试点后续会展开为若干个用例

按需检索

通过 Agent 模式，让 LLM 针对测试点中的模糊内容发起提问，通过问答模式实现检索增强

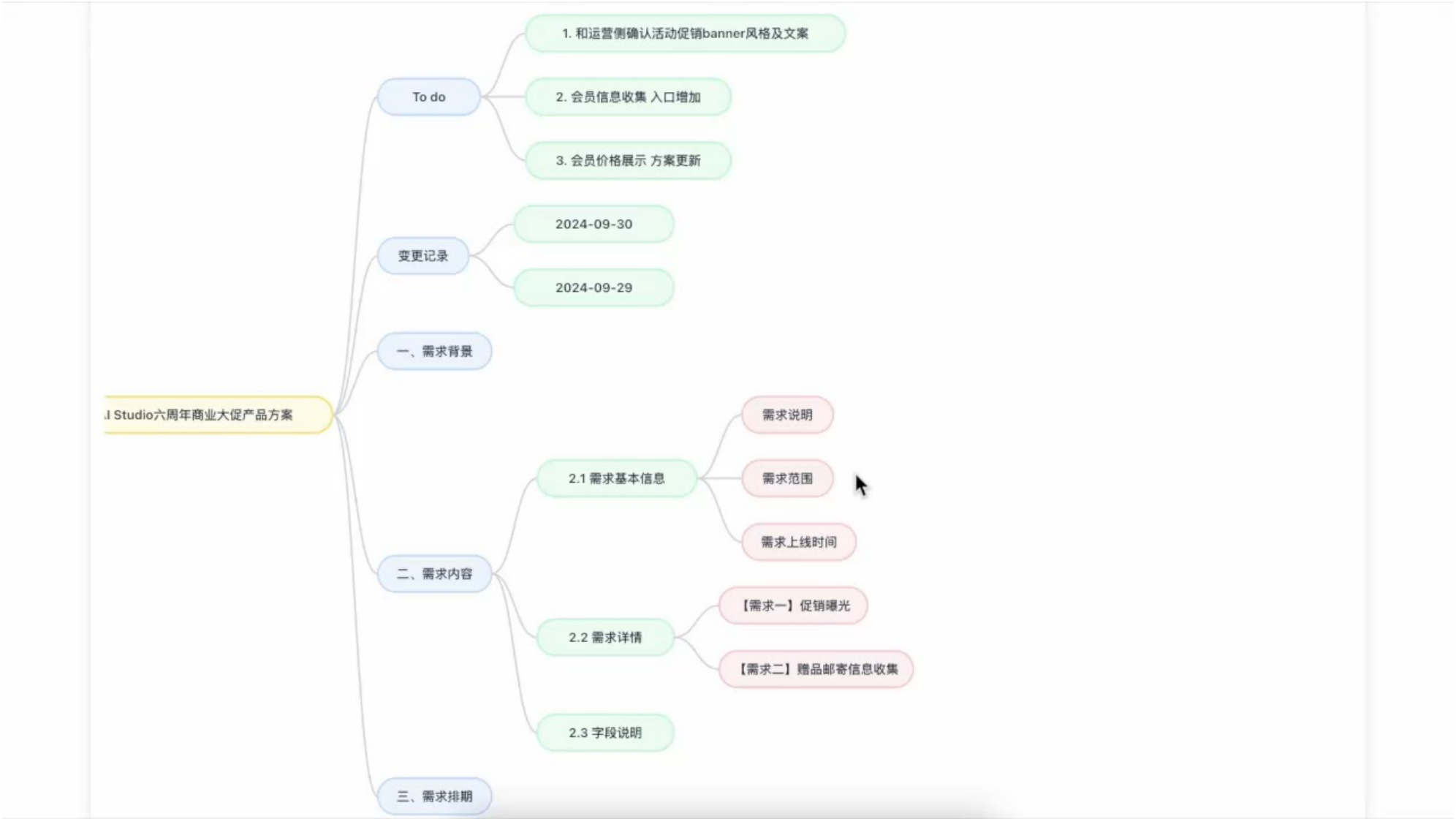
测试点用例设计

利用 LLM 能力，通过测试点及检索增强整合后的内容生成对应的测试用例

用例规范优化

将模型生成的测试用例按照各个业务线的用例编写规范进行优化&改写

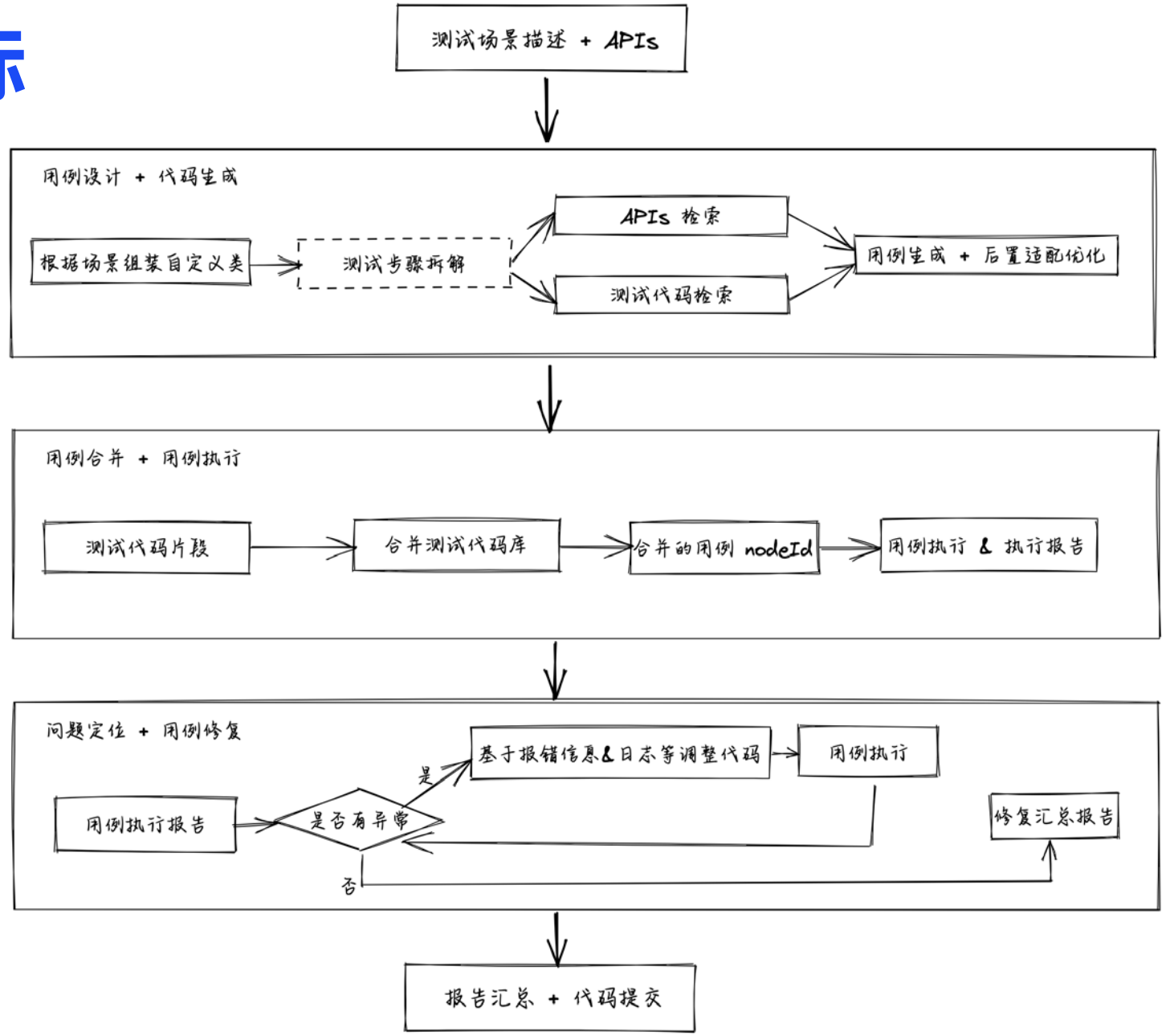
用例设计Agent效果演示



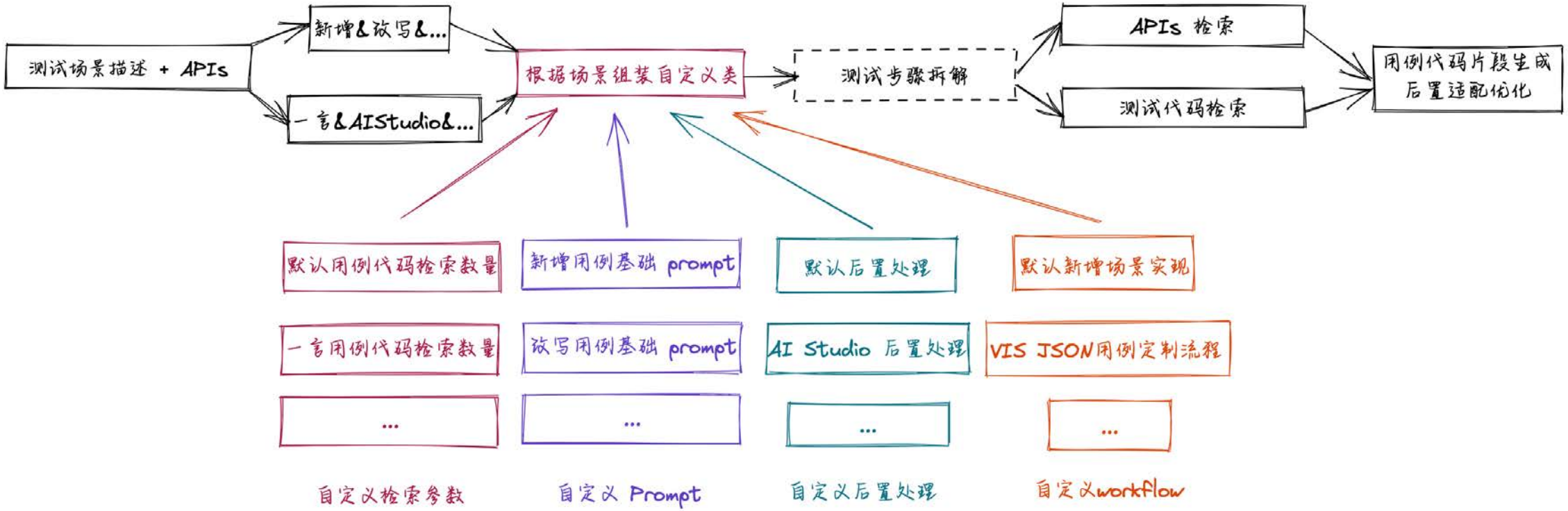
接口测试Agent目标

一、尽可能**端到端**完成接口测试任务，包括生成&合并&调试等环节。

二、在整体能力**可复用**的前提下， 灵活支持业务场景的**自定义扩展**，满足不同类型的项目需求



接口测试Agent之用例生成



一、针对输入参数识别用例编写场景(新增&改写&等)和所属业务，从而支持按需参数&实现方法加载；
二、灵活的模块化设计 + 完善的基础能力实现，业务侧既可以通过轻量级配置模式，也可以通过自定义实现模式让生成效果更match业务预期

接口测试Agent之用例合并&执行

用例合并Agent 定义(简化版)

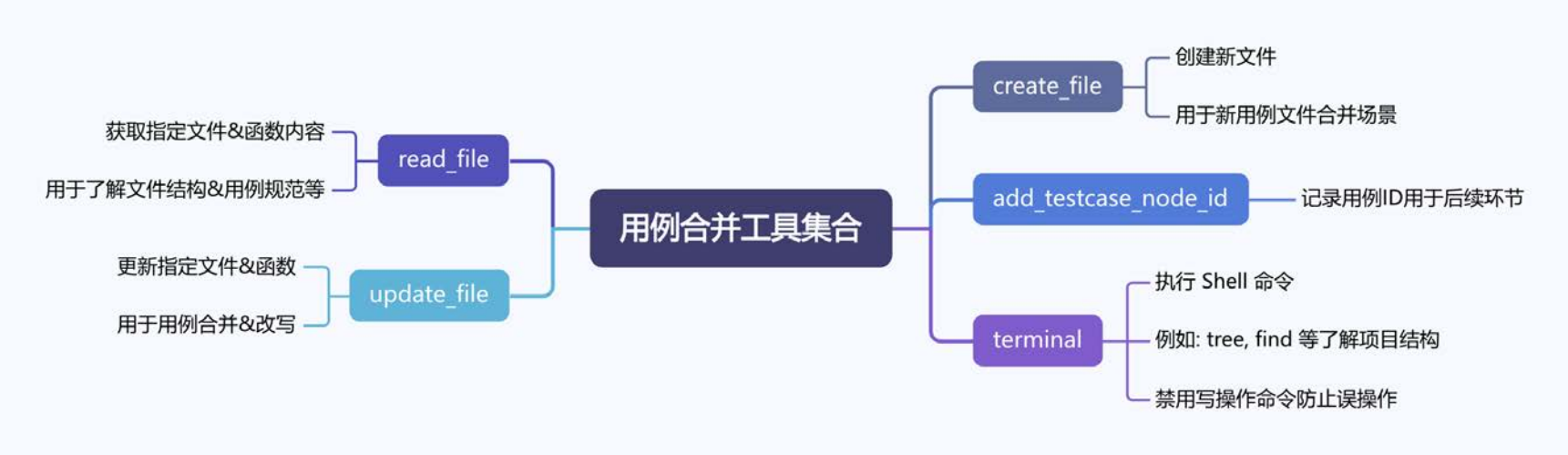
```
你非常精通 Pytest 服务端自动化测试，你和你的伙伴正在合力为一个项目编写自动化用例测试代码。
目前，你的伙伴已经针对需要测试的需求编写了一批测试用例代码片段，但是还没有整个当前的测试项目进行适配，也没有合并到当前项目中。
而你的任务就是需要需要在理解这些代码的基础上，深入理解当前的测试项目结构，并按照当前项目的编写规范和要求，将这些代码片段进行完善优化并合并到当前的项目中。
在合并的过程中，你需要注意代码的格式问题，确保代码的可读性和规范性。
同时，之前同事给出的代码也可能会有一些细节性问题(例如代码风格不一致、丢失import依赖等问题)，你需要一起根据具体情况进行调整和优化。
针对请求地址、端口等信息，你需要参考当前项目中其他类似的用例，将新的代码按照原有用例规范进行改写并合并到当前项目中。

在这个过程中你可以调用工具进行相关文件的代码检索、修改文件代码、创建一个新文件、终端命令执行等。
每当涉及到 pytest 用例的代码合入项目文件完成后，你需要调用add_testcase_node_id追加本次新增用例的 node_id 列表，以便后续的测试执行。

通常而言，在解决这个问题的通常策略如下:
step1: 首先，你需要理解当前项目的结构，了解当前项目的编写规范和要求，以便于后续的代码合并。
step2: 然后你需要对当前需要合并的代码进行理解，分析它的功能和逻辑，从而找出它应该合并到哪个文件中。
step3: 在合并到具体文件中前，你还需要参考当前文件已经的风格和规范或其他类似文件已经的风格和规范，按需调整代码。确保合并后的代码符合当前项目的编写规范和要求，例如有效的处理 import 等关系、保证调用服务地址与整个项目一致等。
step4: 最后，你需要将代码片段合并到对应的文件中，并确保代码的格式正确，依赖导入正常，确保代码的可读性和规范性。
```

任务定义&描述

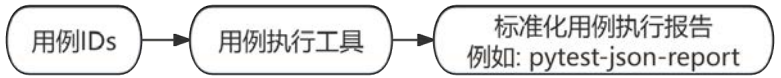
工具介绍&说明

引导的思考步骤

用例合并工具集合



用例执行



针对指定类型用例调用**标准执行工具**执行即可，并产出**标准格式报告**用于后续用例分析&修复场景。

接口测试Agent之用例问题定位&修复

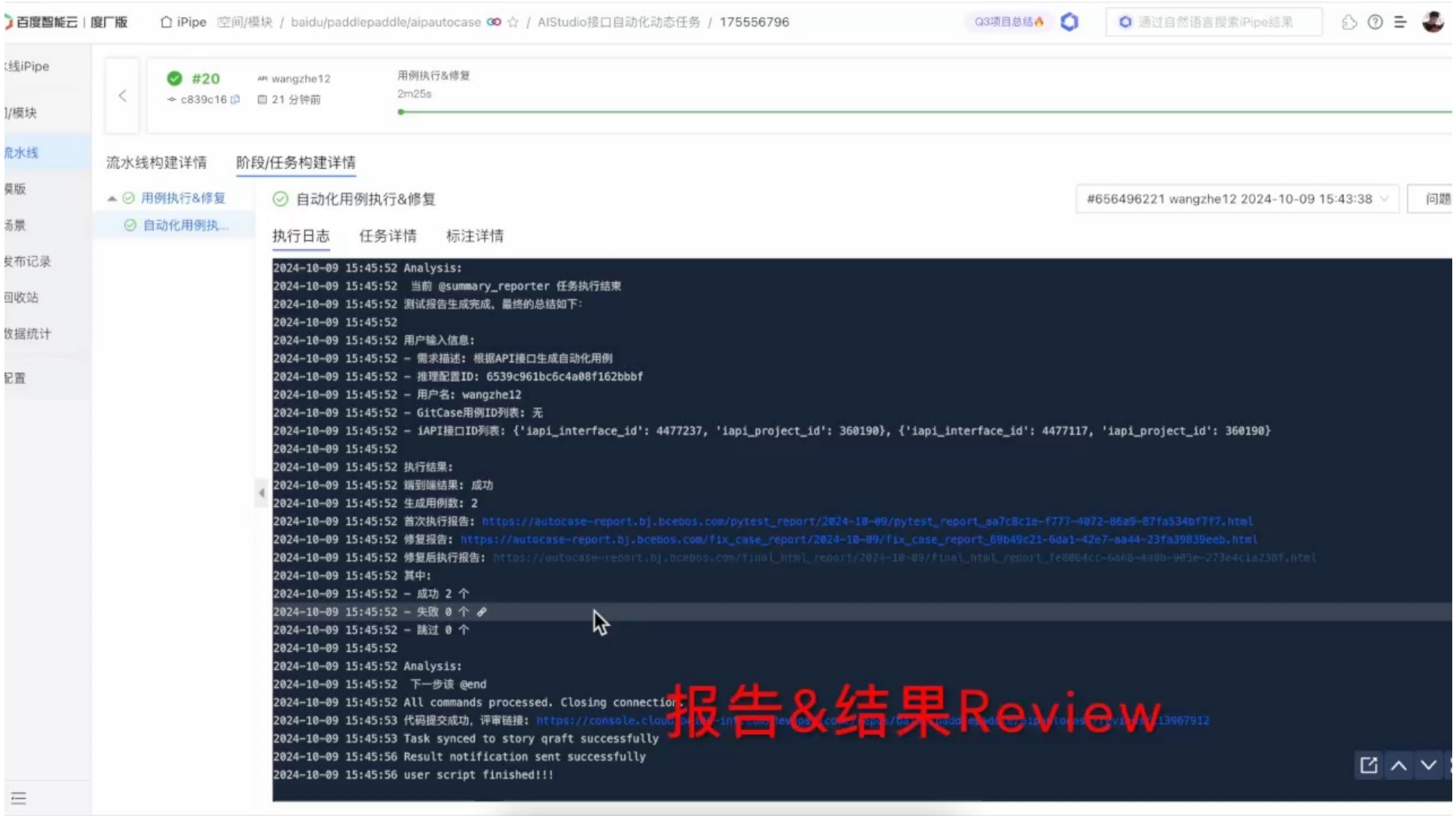
用例的定位 & 修复是日常接口自动化相关工作中的重头戏，也是涉及相对较广的工作。

测试用例异常的原因可能很多，包括:

1. 用例本身的语法问题 & 逻辑问题
2. 服务端本身的稳定性 & 功能问题
3. 服务端 API 变更，与之前不兼容，导致用例失败
4. 服务端内部有隐藏逻辑校验，测试代码没有考虑兼容等

让 Agent 能够完成任务的前提是 Agent 能够通过工具获取到我们日常问题定位需要的内容，因此，我们需要提供一系列工具来完成相关任务。

接口测试Agent效果演示



WEB UI测试典型特征 & 核心目标

WEB UI 测试典型特征:

1. 对于用户产品而言，**工作量大**，几乎所有的改动都需要从 WEB 端进行相关的验证 & 回归。
2. 自动化成本**编写 & 维护成本高**，每个操作对应的元素都需要人工设计&编写元素定位表达式，同时一次组件升级或页面优化都可能导致原有自动化代码不可用。

传统录制&回放方案弊端:

1. 传统的录制&回放方案仅解决了首次元素定位的成本，但是往往元素定位的方式并不鲁棒，导致页面稍有细微调整都会导致自动化测试代码失效，**维护成本更大**。
2. 无论是新增用例，还是原有用例失效，都需要重新进行一次录制，而录制过程其实是纯人工操作，**额外增加一系列录制成本**，随着用例的增多，维护成本不断线性增长。

WEB UI 智能化核心目标:

1. 利用大模型的能力，**智能化生成元素定位方式**（例如 ID, XPATH, CSS Selector等），并保证元素定位表达式尽可能简洁&鲁棒。
2. 新增场景基于用例设计或简要截图说明自动执行智能化测试并生成自动化测试代码，回归等原有用例调整环节自动完成用例代码修复，**全流程尽可能降低人工参与的操作成本**。

WEB UI测试Agent建设历程-可行性验证

HTML 压缩

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Test Page</title>
    <link rel="stylesheet" href="styles.css">
    <style>
        body { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h1>Welcome to the Test Page</h1>
    <svg width="100" height="100">
        <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
    </svg>
    <p>This is a paragraph.</p>

    <!-- This is a comment that should be removed -->
    <div>
        <p>Another paragraph inside a div.</p>
    </div>

    <script>
        console.log("This is a script located at the bottom of body");
    </script>
</body>
</html>
```

head 移除

svg 精简 -> <svg></svg>

注释移除

script 移除

通过 HTML 压缩和基于 LLM 的元素定位，可以实现基本的可行性验证，通过基础的验证集合验证， 75% 的元素定位完全正确，且符合专业标准，**初步证明了整体方案的可行性**。

基于 LLM 的元素定位

```
你是一个资深的 WEB UI 自动化测试工程师，你非常熟悉 Selenium 框架并精通一系列 Selenium 的元素定位和操作方法。
你需要根据测试操作以及当前页面的HTML元素状态，推断出对应操作元素的定位方式。

其中：
1. 所有的元素定位都需要确保定位到的元素是唯一的，否则会定位失败；
2. 所有的元素定位都必须是从 HTML 页面中进行分析，务必不要根据操作步骤来幻想，如果 HTML 页面中无法找到相关元素，则直接报错即可。

最终，你需要通过 JSON 格式输出当前的元素的定位方式。
其中，输出的 JSON 内容需要能够被如下 Pydantic 对象 Locator 解析：

```python
class By(Enum):
    """
    Set of supported locator strategies.
    """
    ID = "id"
    XPATH = "xpath"
    LINK_TEXT = "link text"
    PARTIAL_LINK_TEXT = "partial link text"
    NAME = "name"
    TAG_NAME = "tag name"
    CLASS_NAME = "class name"
    CSS_SELECTOR = "css selector"

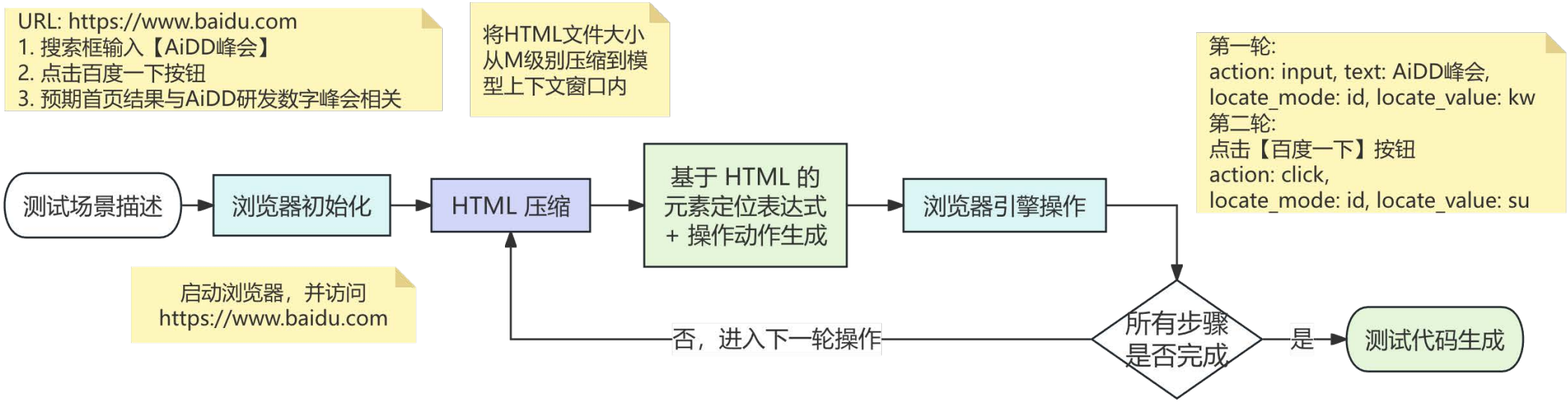
class Locator(BaseModel):
    """
    定位器
    """
    by: By = Field(..., description="定位模式， 例如 ID, CSS_SELECTOR, 等")
    value: str = Field(..., description="定位标记，例如 #username 等")
```

当前需要进行的操作如下：
{action_desciption}

当前页面压缩后的 HTML 结构如下：
{html_content}

现在，你需要根据当前页面的元素状态，推断出对应元素的定位方式，并通过 JSON 格式进行输出结果。
```

WEB UI测试Agent建设历程-初版架构



在初步可行性验证完成后，我们希望快速搭建一个 Demo 系统来跑通一个完整的流程：即输入完整的测试步骤描述，通过AI Agents 来自动完成整个浏览器的操作，并在操作完成后基于操作的步骤生成自动化 WEB UI 测试代码。

涉及到的模块:

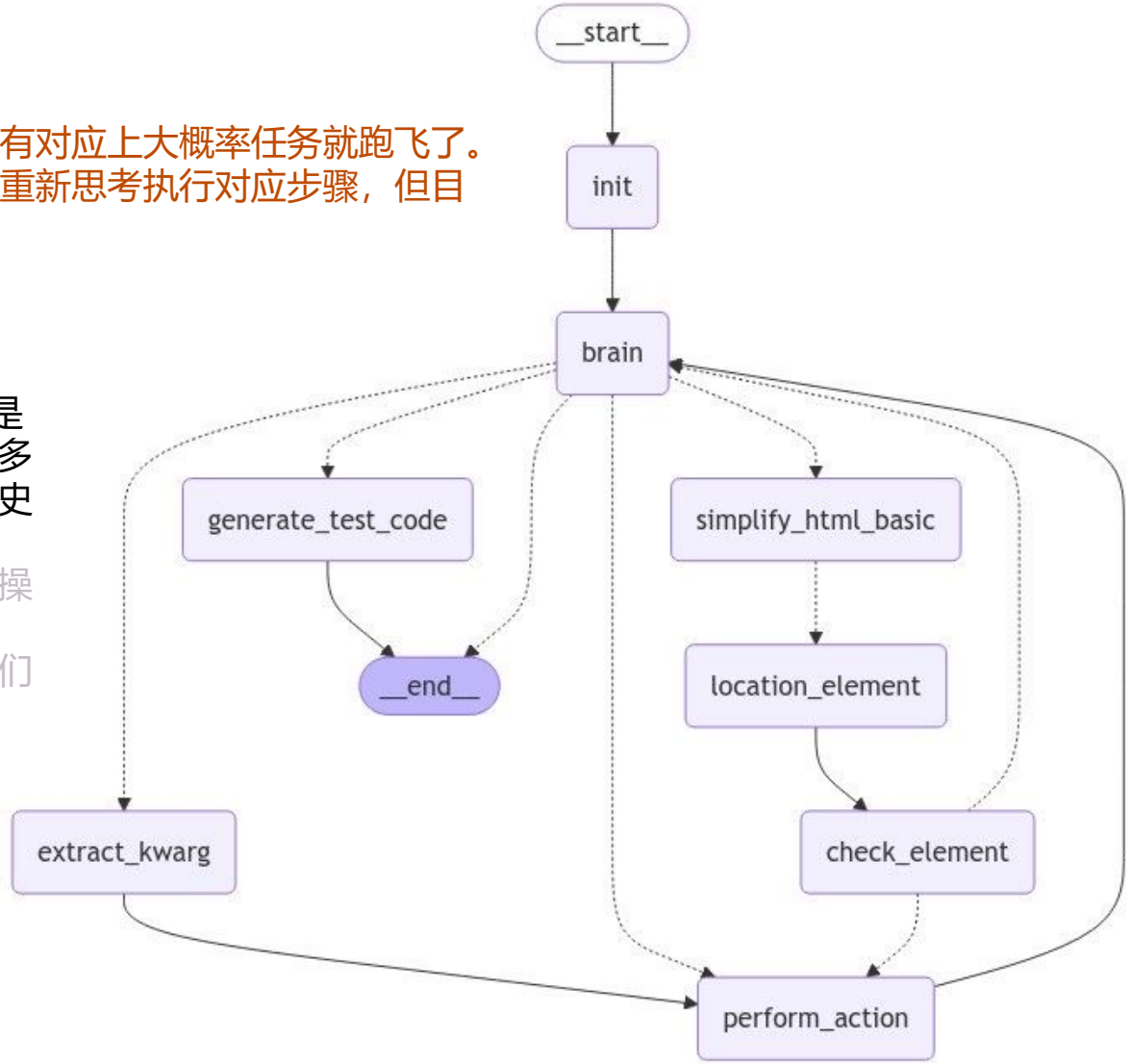
1. HTML 压缩模块: 参考可行性验证，负责 HTML 压缩
2. 元素定位: 参考可行性验证，负责找出对应的操作元素的元素定位表达式
3. 浏览器执行模块: 以 Selenium 为基础，提供了一套通用的浏览器操作接口并实现浏览器操作
4. 测试代码生成模块: 简单的 prompt 推理，传递测试场景和浏览器执行的操作动作历史即可

WEB UI测试Agent建设历程-基于准确率提升的架构优化

问题&挑战: 输入的操作步骤要求过高，实际落地困难
1. 每个用户描述的操作步骤需要严格对应浏览器的操作步骤，任何一步没有对应上大概率任务就跑飞了。
2. 对于操作失败的场景性兼容不够，任何一步没有达到预期效果时，应该重新思考执行对应步骤，但目前会自动跳过，导致大概率后续的动作全部失败了。

brain
像人在 WEB 交互一样，在当前的浏览器状态下，下一步需要进行的操作是什么，并通过简单的文本描述输出即可。而通过在 brain 中整体放入一个多轮对话场景中，可以在 brain 推理的环节中更加准确的理解全局任务、历史动作和当前操作的关系等。
在 brain 推理时，需要从当前浏览器页面的真实状态出发来判断下一步的操作以及最近一步操作的结果，仅仅通过 HTML 文件其实是几乎做不到的，更加直观的方式就是通过浏览器截图。在 brain 的每一轮新的对话中，我们会传当前浏览器最新状态的页面的截图。

location_element
单纯仅负责一件事，就是找出 brain 拆解出的操作对象的元素定位方式即可。



WEB UI测试Agent建设历程-工程优化应用

问题&挑战:
1. 成本问题: HTML压缩效率有限，每轮元素定位的输入token约为30-80K，成本过高

2. 效果问题:元素定位成功率不够高（约90%），元素定位的XPATH或CSS Selector有时冗余，影响优雅度和鲁棒性，页面结构调整后，定位容易受影响

核心区域: 针对当前核心区域向父标签连找三级，并保留当前三级父标签下的至多五级子元素，以及从根节点到三代父标签的路径。

HTML 二次压缩

操作对象:【姓名】的输入框

```
<!DOCTYPE html>
<html>
<head>
    <title>示例页面</title>
    <style>
        /* 一些样式 */
    </style>
    <script>
        // 一些脚本
    </script>
</head>
<body>
    <header>
        <h1>欢迎来到示例页面</h1>
        <nav>
            <ul>
                <li><a href="#home">首页</a></li>
                <li><a href="#about">关于我们</a></li>
                <li><a href="#contact">联系我们</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section>
            <h2>请填写以下表格</h2>
            <form id="contact-form">
                <div class="form-group">
                    <label for="name">姓名: </label>        关键区域提取
                    <input type="text" id="name" name="name">
                </div>
                <div class="form-group">
                    <label for="email">电子邮件: </label>
                    <input type="email" id="email" name="email">
                </div>
                <div class="form-group">
                    <label for="message">信息: </label>
                    <textarea id="message" name="message"></textarea>
                </div>
                <button type="submit">提交</button>
            </form>
        </section>                                  核心区域保留
    </main>
    <footer>
        <p>版权所有 © 2023 示例公司</p>
    </footer>
</body>
</html>
```

XPATH 自动优化策略

模型输出的元素定位表达式

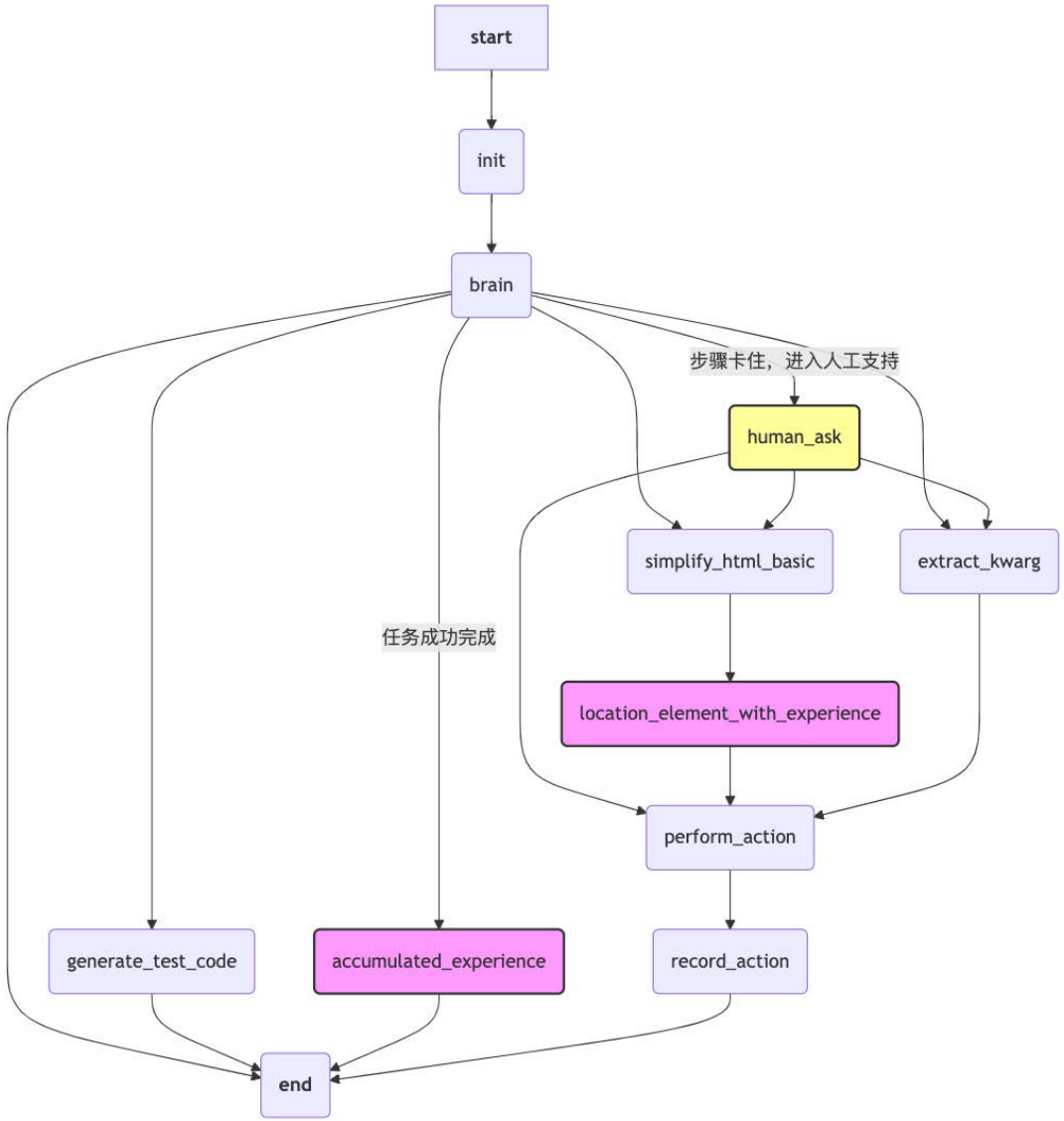
```
//div[@class='container']//form[@id='login-form']//input[@type='text' and @name='username' and @placeholder='Enter username']
```

Step1: 将 XPATH 表达式拆解了 N 个 AND 条件组合，例如:
1. 元素是input标签
2. 元素的type属性为'text'
3. 元素的name属性为'username'
4. 元素的placeholder属性为'Enter username'
5. 元素位于id为'login-form'的form标签内
6. 元素位于class为'container'的div标签内

Step2: 逐个删除 AND 条件，如果删除某个条件后，元素定位不唯一，则保留该条件，直到所有条件全部简化验证完成。

```
//form[@id='login-form']//input[@name='username']
```

WEB UI测试Agent建设历程-人机交互范式



问题&挑战: 在一系列优化后，仍然还有一些 corner case 无法正常处理，需要引入人工交互指引或者历史经验引导。

典型场景: 例如 svg, img 等图片&图标，单独依赖 HTML 内容找到对应操作元素的难度通常很大。

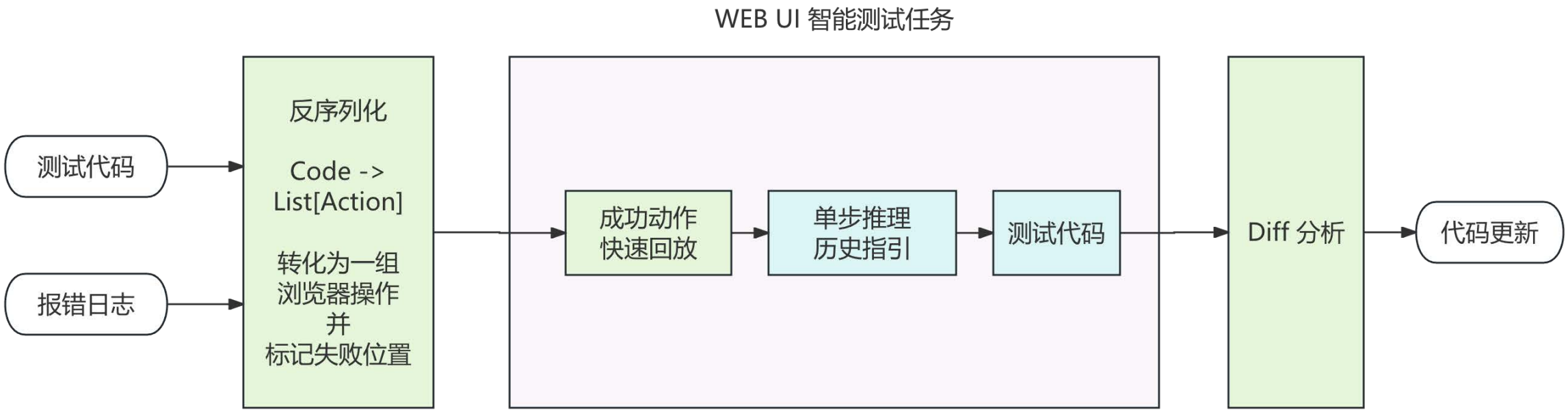
优化一：在尝试操作失败后，不直接终止，而是进入人工支持

在之前的流程中，一旦测试场景中任意一步异常，会导致整个流程终止。如果希望调整描述步骤等信息重试，需要重新开始，使用体验和效率都很差。通过加入人工支持环节，可以在遇到疑难场景后，人工进行指导来进行后续操作，从而通过简单的人工交互让整体流程跑通，大幅提升体验。同时，在人工支持的场景中，既可以简单通过文字描述下一步的操作场景来纠正步骤规划问题，也可以通过直接给出元素定位方式来解决元素定位失败的问题，具体以实际的出错原因进行指导。

优化二：引入历史经验，在元素定位时优先参考历史经验

在整个测试任务执行成功后，自动将其中涉及到的一系列元素操作的元素定位表达式进行持久化保存，在后续元素定位时，如果涉及到的页面 URL 中有历史元素定位方式，则优先从历史经验加载。一方面可以提升整体准确率，也可以降低一定的 token 消耗(历史经验的token长度通常远小于压缩后的HTML内容的token长度)

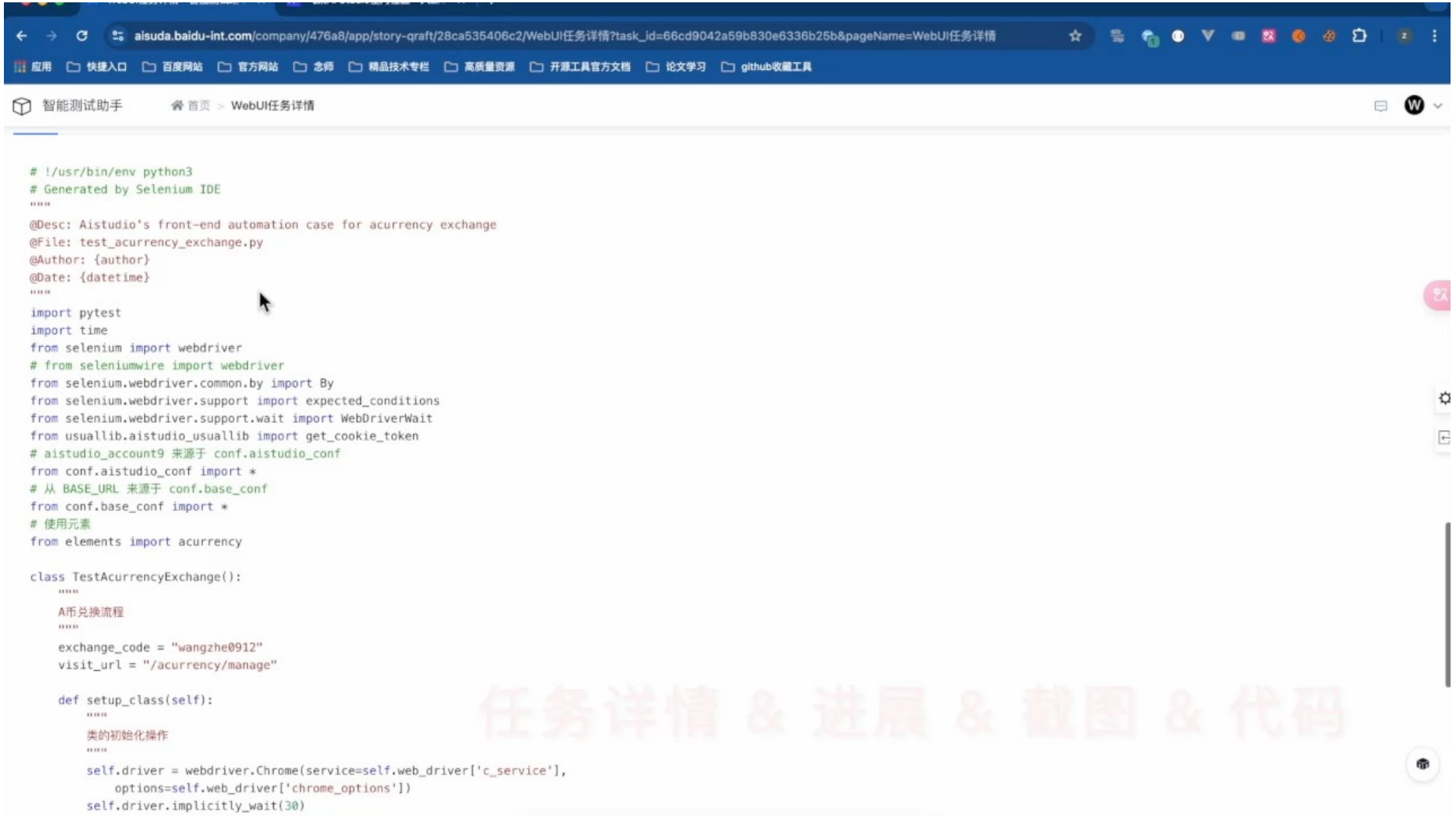
WEB UI测试Agent建设历程-自动修复能力



WEB UI 自动化用例的维护成本一贯在 WEB UI 自动化中占比很高，智能化能力需要覆盖 WEB UI 自动化用例的维护&更新。自动修复能力还是以 WEB UI 智能生成能力为核心，在这个基础上增加了测试代码到操作动作的转化、正常动作的快速回放能力以及后置的代码diff分析与更新能力。

此外，为了避免频繁转化，我们也提供了一套基于 Action 格式的数据驱动模式的 WEB UI 自动化框架，可以无缝实现智能执行+修复，适合 WEB UI 自动化从零开始的新业务。

WEB UI测试Agent效果演示



PART 04

总结与展望

总结

1. AI Agents 给测试环节带来了新的可能，它一定会颠覆传统的测试流程，形成 10x 倍提效；

2. AI Agents 在落地过程中，需要解决好两类问题，一是让 Agent 足够理解业务、打通业务私域知识；二是结合业务实际情况，让 Agents 能尽可能端到端完成任务，从而实现模式变更和效率提升

3. AI Agents 在很多环节都有充分的想象空间，包括但不限于用例设计、接口测试、WEB 测试等

4. AI Agents 等智能化解决方案在落地过程中，要循序渐进，从小规模试点开始，看到收益和变革后再逐步扩大应用范围；

5. 加强团队的 AI 原生应用思维对于智能化测试的落地有着重要的影响。

展望

1. 强化多模态数据理解能力
2. 代码 & 文档 & 设计图等各类知识一体化
3. 智能知识迭代，时刻理解项目最新进展&功能

1. 模块化组合，个性化定制，快速搭建符合业务流程的 AI Agents 生态
2. 更精准的效果、更独立的处理能力，进一步简化测试工作流程。

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