

# 从辅助到驱动：构建研发智 能体的实践与思考

演讲人：杜沛  
汽车之家 / 客户端架构师

**AiCon**  
全球人工智能开发与应用大会

# 目录

- 01 AI Coding在企业落地的挑战与思考
- 02 Coding Agent的多层记忆与能力增强设计
- 03 案例1：跨技术栈代码转换
- 04 案例2：Design to code 高UI还原度的技术突破
- 05 总结与展望

# 极客邦科技 2026 年会议规划

促进软件开发及相关领域知识与创新的传播



参会咨询



查看会议

北京

1200人

**QCon**

全球软件开发大会

会议时间：4月16-18日

- Agentic Engineering
- AgentOps
- 下一代模型架构与推理优化
- AI 原生基础设施
- 知识工程实践
- AI 安全

深圳

1000人

**AiCon**

全球人工智能开发与应用大会

会议时间：8月21-22日

- Agentic AI
- 轻量化与高效推理
- 多模态应用
- AI + IoT 场景实践
- AI 工业化落地

北京

1000人

**AiCon**

全球人工智能开发与应用大会

会议时间：12月18-19日

- 大模型架构创新
- 多模态 AI 产业融合
- 具身智能
- AI for Science
- 大模型安全

4月

6月

8月

10月

12月

**AiCon**

全球人工智能开发与应用大会

会议时间：6月26-27日

- AI Infra 系统工程
- 多 Agent 协作与实践
- 多模态融合
- 模型训练与推理创新
- 数据平台与特征服务

上海

1000人

**QCon**

全球软件开发大会

会议时间：10月22-24日

- AI Agent
- Vibe Coding
- 智能可观测
- 推理基建
- 模型攻防
- AI x 创造力

上海

1200人

# 01 AI Coding在企业落地的挑战 与思考



# 开发者的 AI Coding 核心诉求

人引领 : 通用大模型  
单指令交互

AI驱动  
• 复杂任务规划  
• 工具执行  
• 观察与修复

AI引领

1

2

3

4

5

代码续写

代码问答

助理模式

代码智能体

自适应全自主

## 深度理解业务需求

- 希望AI能够基于项目历史需求与问题经验,自动理解日志、性能等隐性技术要求

## AI代码输出一致性

- 不同开发者使用AI生成的代码保持统一的风格和模式,确保代码质量的一致性

## UI高还原度与组件复用

- AI精准理解设计稿,自动匹配并复用企业级组件库,覆盖多端场景

# 面临的挑战



## 复杂项目上下文工程

大型项目涉及多层架构、多种技术栈和复杂依赖关系,单一记忆体系难以准确把握项目全貌。AI在处理跨模块需求时,容易遗漏关键上下文,导致生成代码与实际架构脱节



## 多模态识别准度低

AI难以精准理解设计稿中的视觉细节(间距、圆角、阴影等)、交互规则(响应式布局、动画效果),生成的界面与预期存在偏差,需要大量手动调整。



## 规范的结构化注入

团队规范散落在文档、wiki、历史代码中,隐性知识难以显性化;难以整理成可理解的结构化形式。且规范本身也在持续演进,同步更新成本高



## 工程质量保障

AI在代码质量自检环节缺少明确指引,容易遗漏性能优化、安全防护、异常处理等关键检查项。自检不准确导致生成代码直接用于生产环境的风险较大。

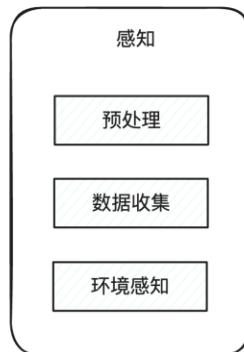
# 思考：让Agent变成‘团队成员’

- 让Agent知道“我是谁”

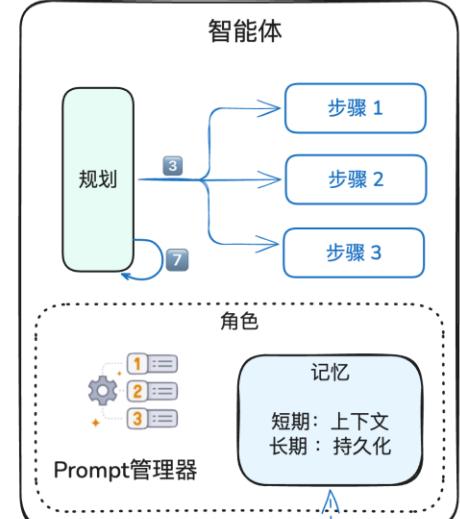
理解团队、项目和历史上下文，不再每次都失忆



① 问题输入 →



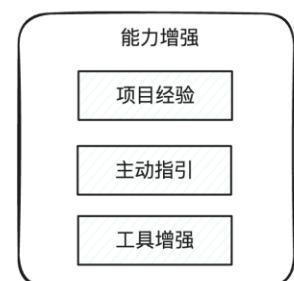
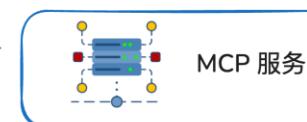
② 输入 →



⑥ 观察 ←



⑤ 执行



- 让Agent知道“我会什么”

工具是手段，技能才是目的，是真正具备处理特定领域的技能

- 让Agent知道“怎么干”

沉淀最佳实践与执行路径，按团队方式做事，而非从零试错

# 02 Coding Agent的多层次记忆与 能力增强设计

# Agent架构设计

## • 多层记忆体系

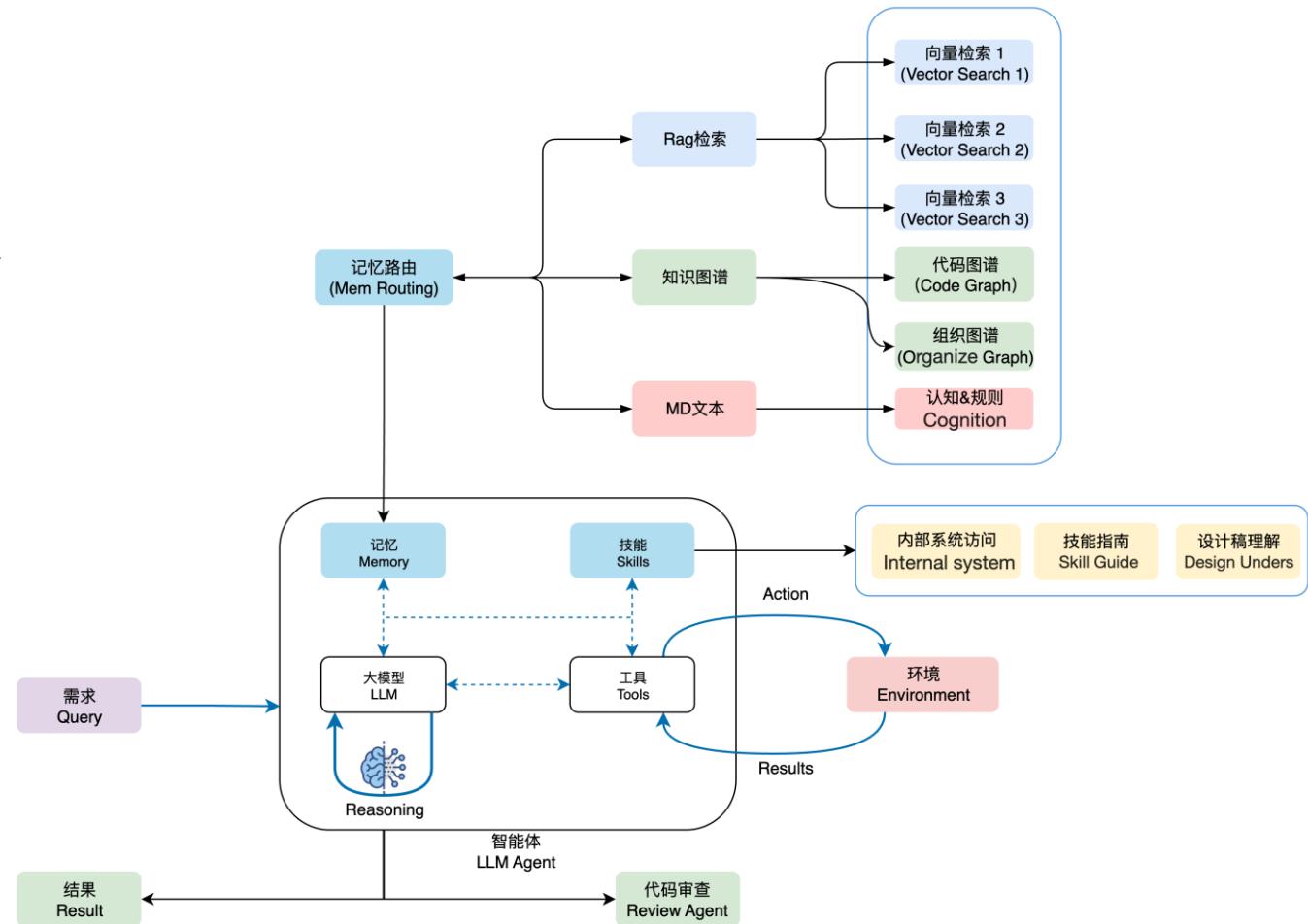
构建个人记忆、组织记忆、项目经验多层次记忆结构，让Agent既懂“我”，也懂“团队”，更懂“项目”

## • 领域技能内置

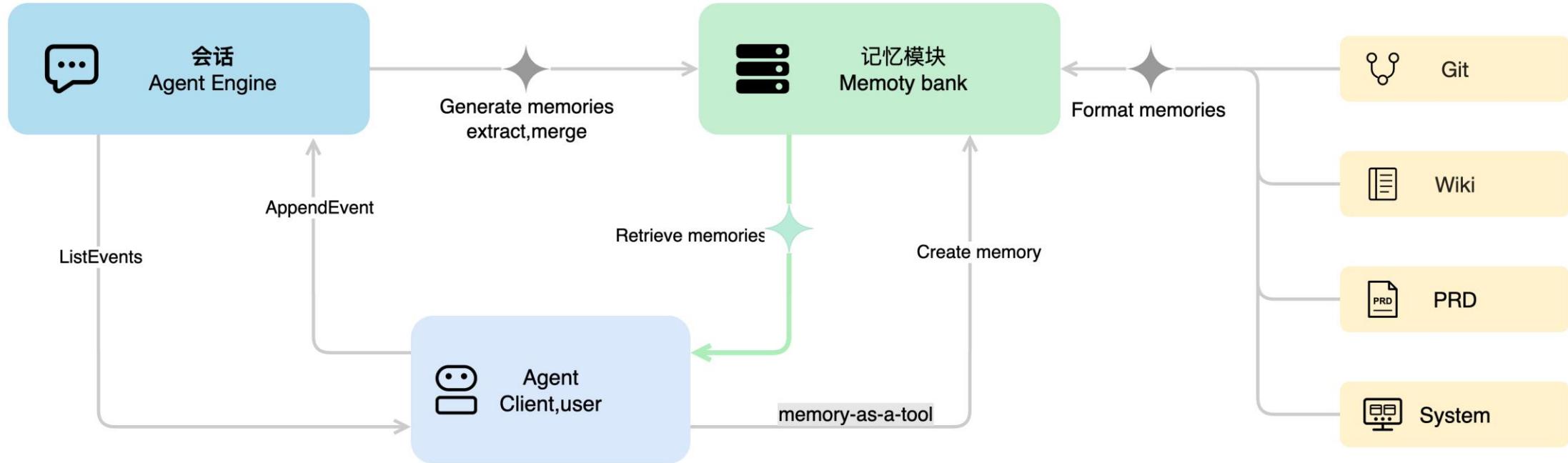
集成内部系统访问、技能指引、设计稿理解等专业技能，让智能体具备特定领域的专业能力

## • 认知规范审查

认知规范确保输出符合安全、合规和质量标准，约束智能体异常输出行为



# 记忆模块设计

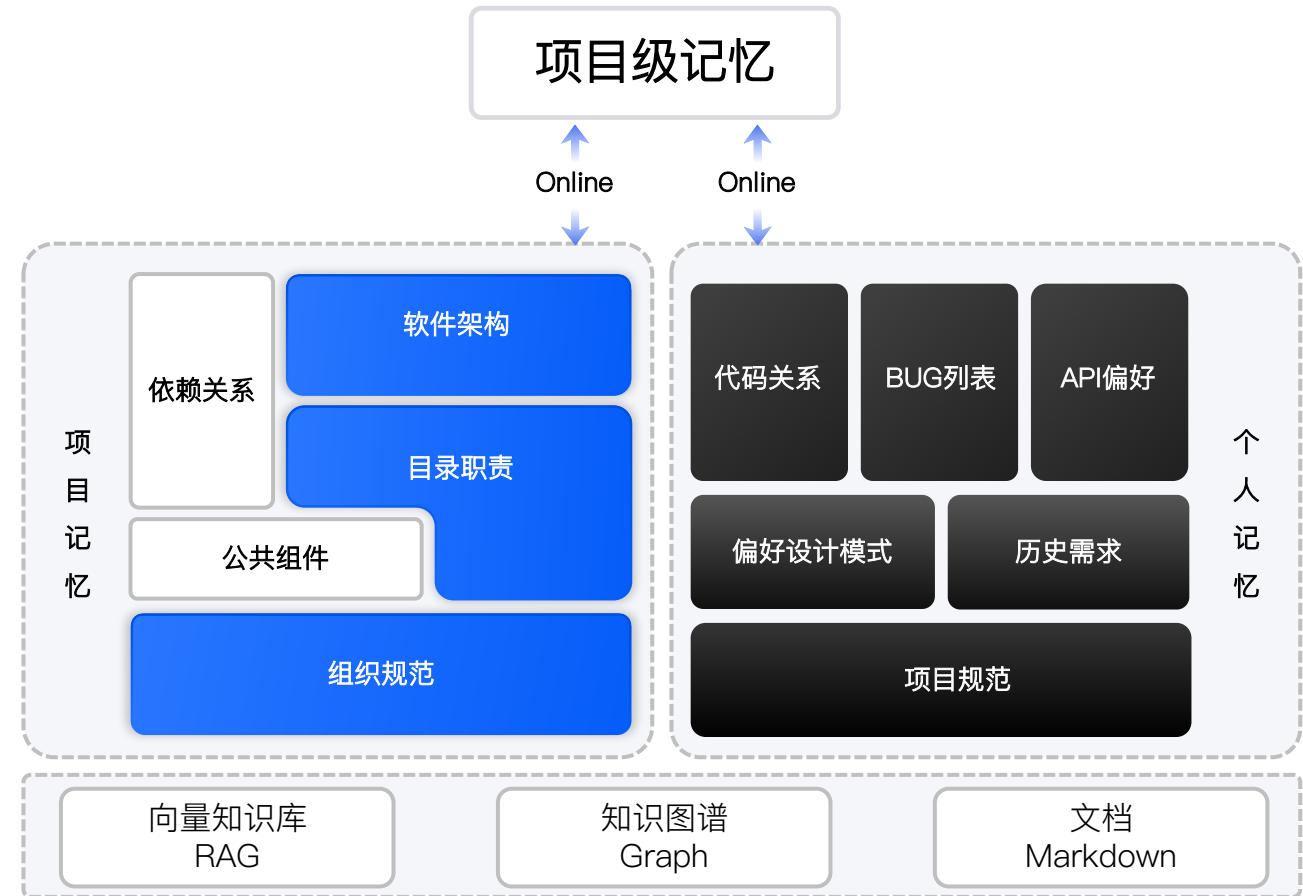


记忆来源于历史对话总结、经验灌输、文档资料、产品需求等多维输入

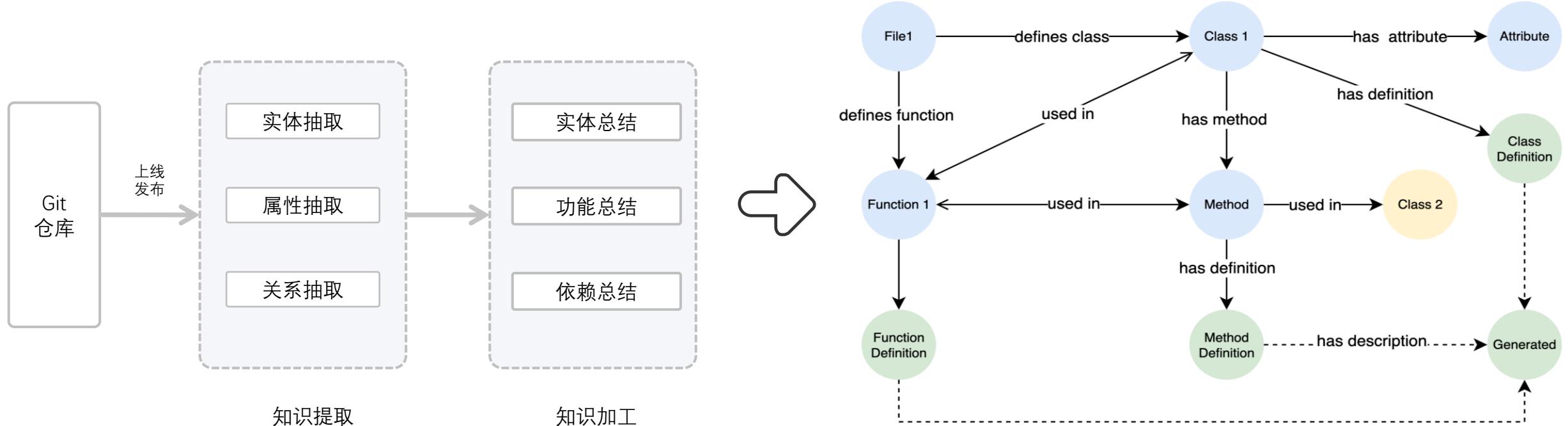
- 短期记忆：从对话上下文中动态提取和压缩关键信息
- 长期记忆：沉淀历史成功案例（项目级别），开发者组织全貌、项目细节、开发规范、公共组件等

# 项目级记忆结构

- 依赖的项目或库（包括版本号、包管理工具如npm、pip）
- 网络架构（如REST API、GraphQL、WebSocket、网关配置）
- UI布局习惯（如组件结构、网格系统、响应式设计方法）
- 样式的选择（如CSS框架、预处理器、主题变量）
- 设计模式（如MVVM、工厂模式、观察者模式在项目中的具体应用）
- 通信方式（如HTTP客户端、消息队列、RPC框架）
- 状态持久化方案（如数据库类型、ORM工具、缓存策略）
- 路由管理（如前端路由库、后端路由配置、权限控制）
- 标准样式指南（如代码格式化规则、命名约定、linting配置）
- 项目目录职责（如文件夹结构、模块划分、资产存放位置）
- 测试策略（如单元测试框架、集成测试方法、mocking工具）
- 安全实践（如认证机制、加密标准、漏洞防护）
- 项目配置细节（如构建工具、环境变量、配置文件路径）
- 项目的目录结构职责（例如/src/utils存放工具类函数）
- 使用的软件架构（如微服务、单体、事件驱动、分层架构）

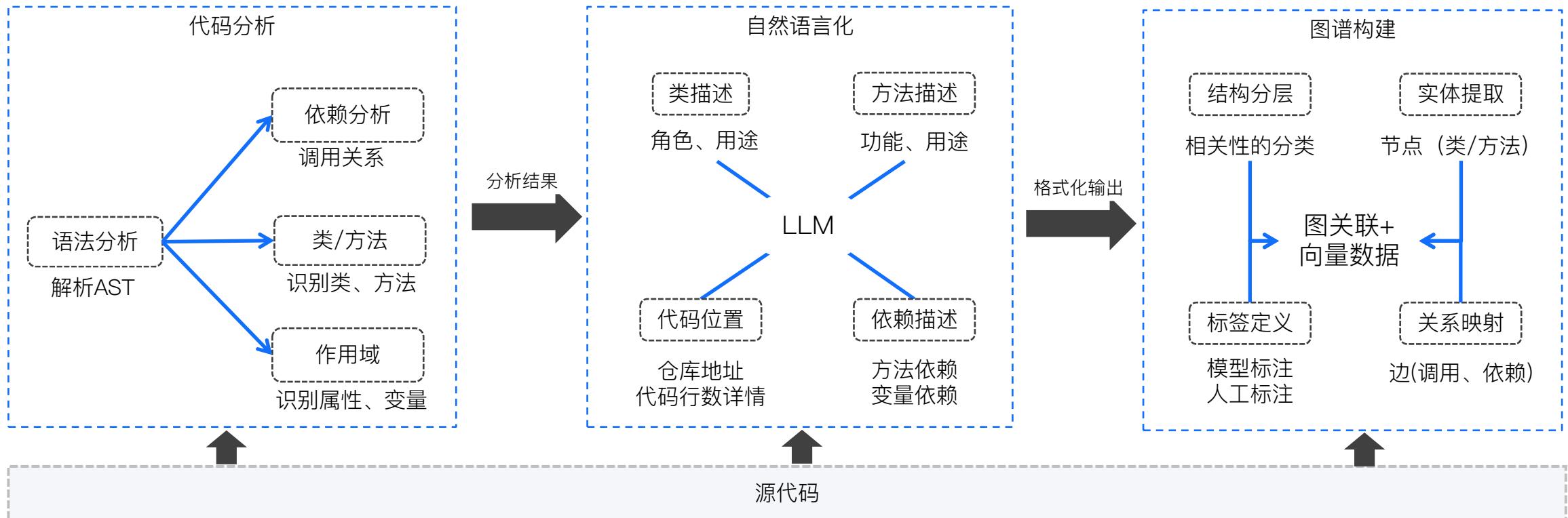


# 从代码到知识：实体-功能-依赖的结构化提取



构建结构化代码知识图谱，让大模型能够快速精准地理解代码结构和依赖关系，支撑代码生成。

# 代码图谱构建



# 记忆的持续学习

## • 知识沉淀

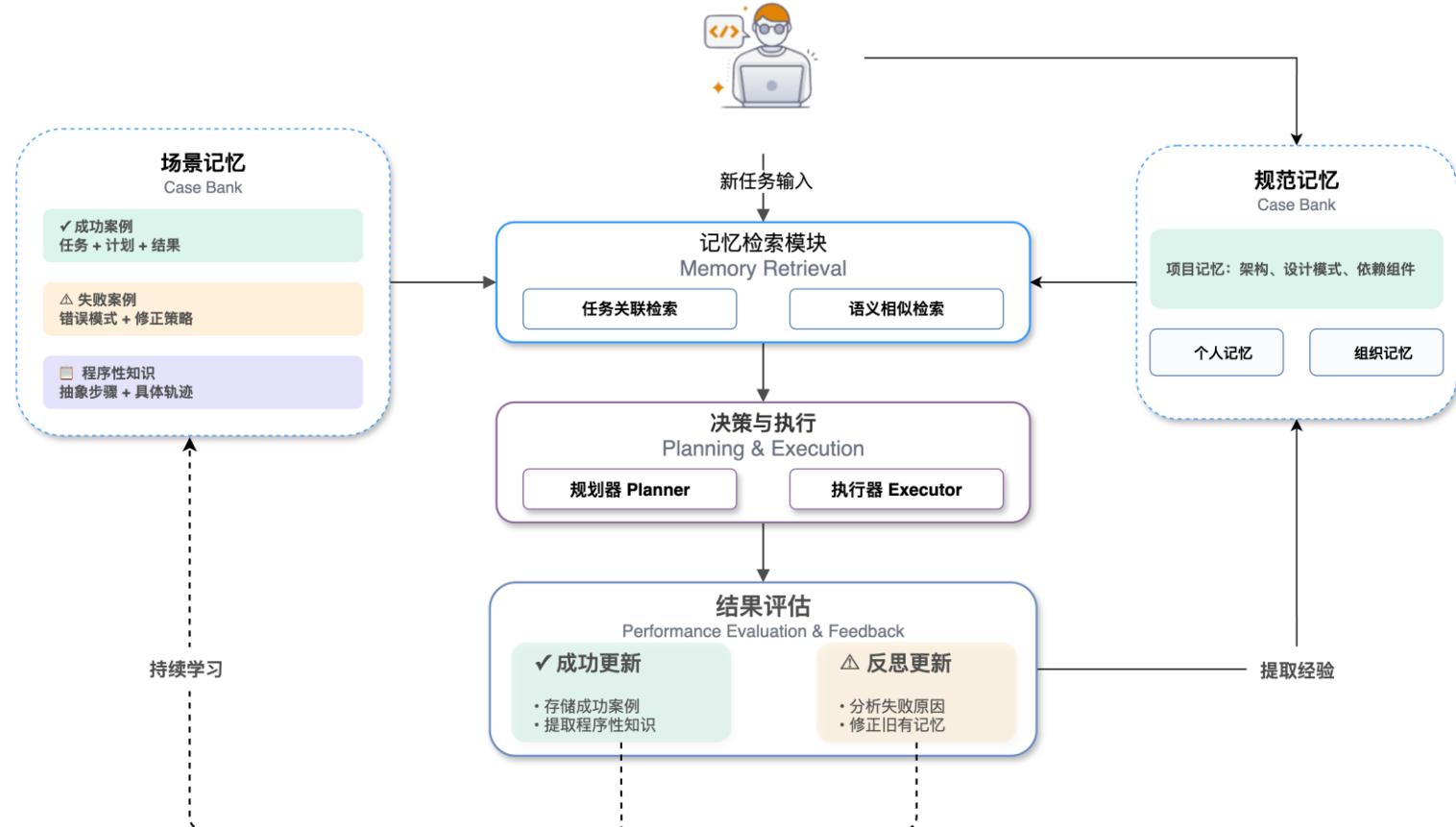
规范记忆沉淀团队标准，定义“应该怎么做”；场景记忆从任务执行中积累，提供“怎么做才对”的经验参照

## • 记忆检索

识别任务意图，按需召回对应记忆；动态组合场景经验与规范约束

## • 闭环进化

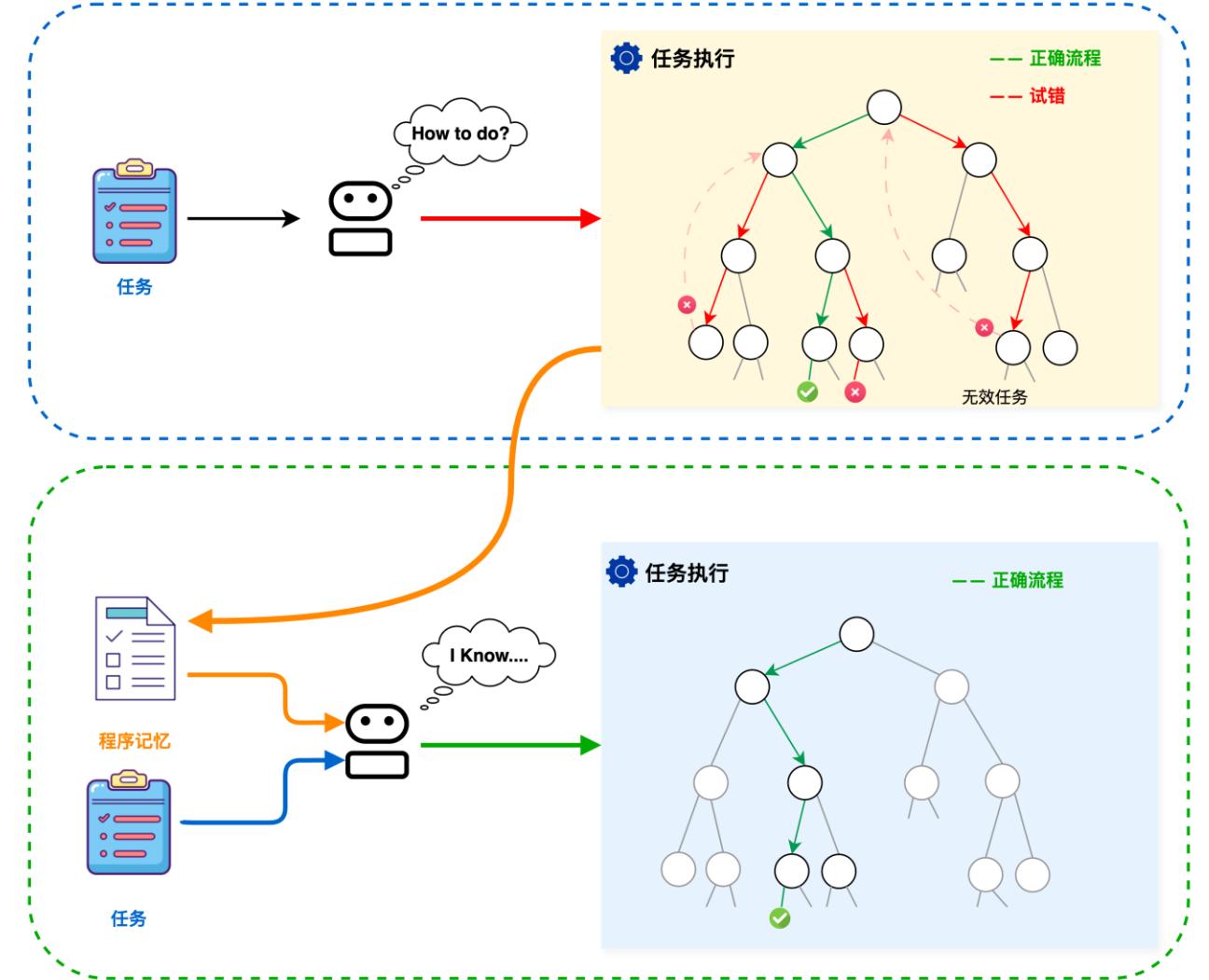
执行结果自动反哺记忆库：成功案例沉淀为场景记忆，通用规律提炼为规范记忆，持续自我优化



# “学会”正确做事

## 经验指引，精准执行：

- 消除产出不确定性：从“How to do?”升级为具备主动判断力的“*I Know....*”
- 规范内化为先验：将复杂决策树简化为可预测的、规范化的最短路径
- 指引式收敛：通过结构化的指引经验，Agent 跳过不必要的探索和验证环节



02

## 案例1：跨技术栈代码转换

# ■ 跨端代码转换

## 面临的挑战

- **平台特性差异**

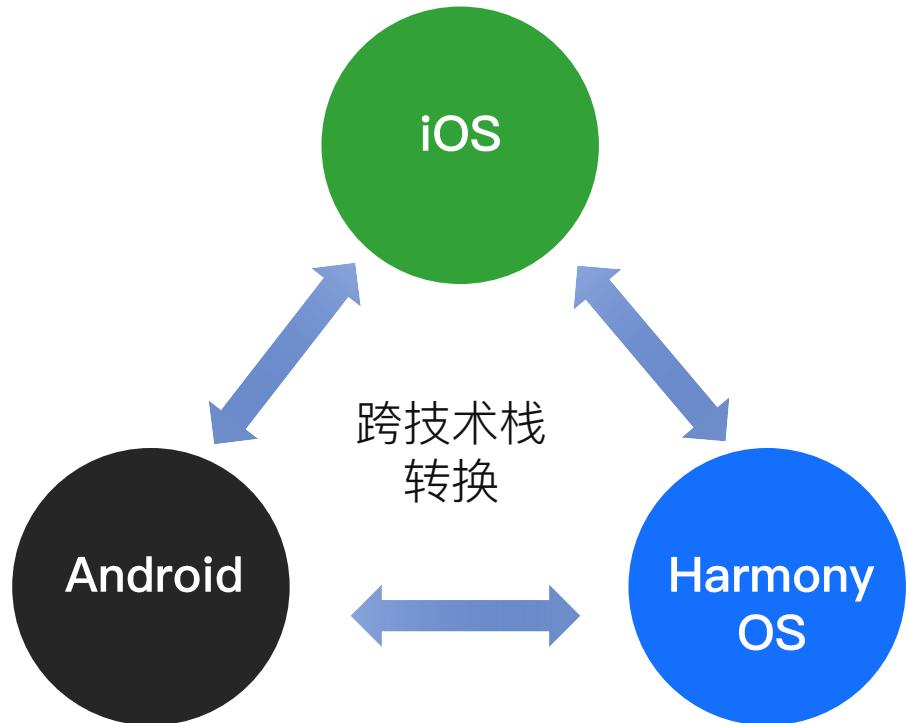
各平台生命周期、权限管理、导航模式等底层机制不同，无法直接平  
移，需针对性适配

- **API对应关系**

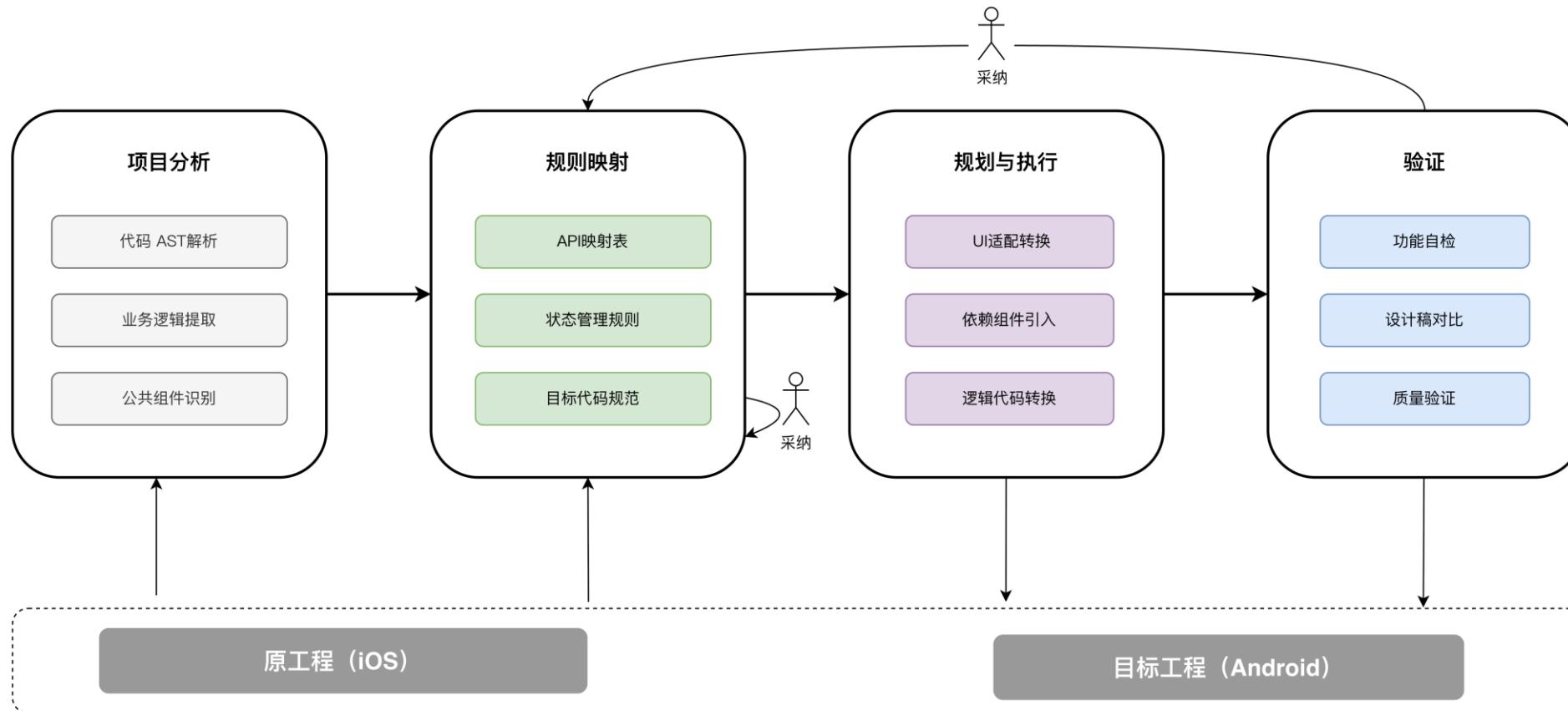
不同平台依赖的三方组件、API差异等，如果没有清晰的对应关系，  
模型不知道怎么转

- **框架适配**

UI框架组件体系不兼容，功能相似但实现机制、属性配置、事件处理  
方式都不同，只是一对一转换可能出现大量不适配问题

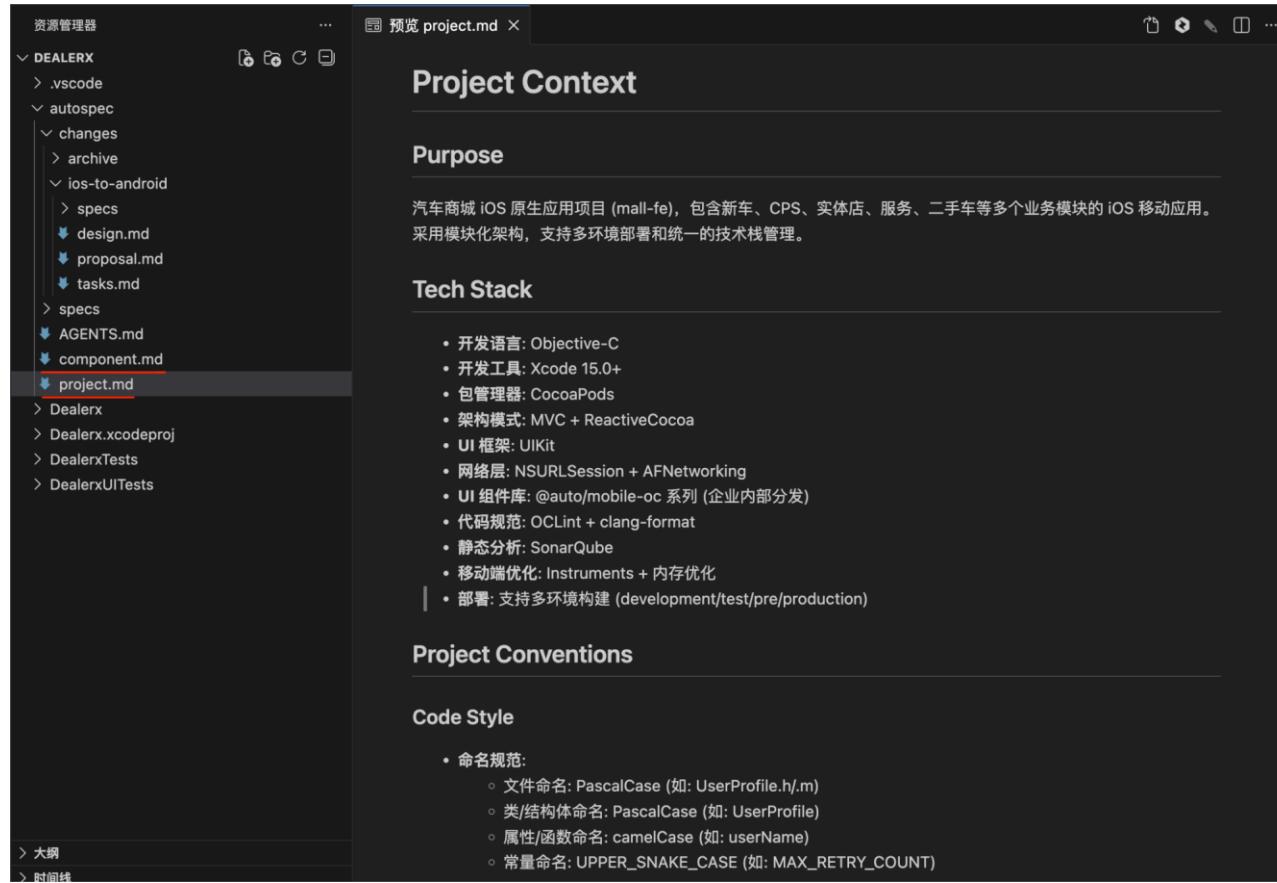


# 基于Spec的跨技术栈代码转换方案



将任务结构化为Rules/Templates/Workflow，LLM基于明确的规范执行转换，确保输出的准确性。

# 工程描述与组件映射



Project Context

### Purpose

汽车商城 iOS 原生应用项目 (mall-fe)，包含新车、CPS、实体店、服务、二手车等多个业务模块的 iOS 移动应用。采用模块化架构，支持多环境部署和统一的技术栈管理。

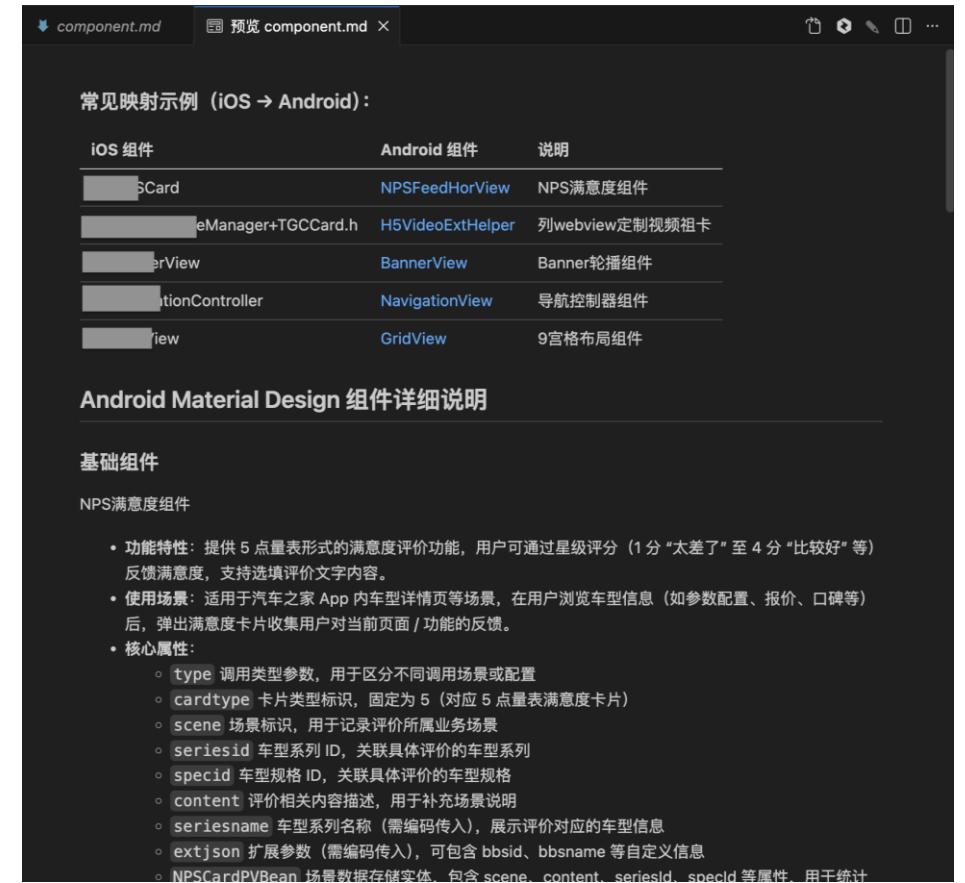
### Tech Stack

- 开发语言: Objective-C
- 开发工具: Xcode 15.0+
- 包管理器: CocoaPods
- 架构模式: MVC + ReactiveCocoa
- UI 框架: UIKit
- 网络层: NSURLConnection + AFNetworking
- UI 组件库: @auto/mobile-oc 系列 (企业内部分发)
- 代码规范: OCLint + clang-format
- 静态分析: SonarQube
- 移动端优化: Instruments + 内存优化
- 部署: 支持多环境构建 (development/test/pre/production)

### Project Conventions

#### Code Style

- 命名规范:
  - 文件命名: PascalCase (如: UserProfile.h/m)
  - 类/结构体命名: PascalCase (如: UserProfile)
  - 属性/函数命名: camelCase (如: userName)
  - 常量命名: UPPER\_SNAKE\_CASE (如: MAX\_RETRY\_COUNT)



常见映射示例 (iOS → Android):

iOS 组件	Android 组件	说明
NSCard	NPSFeedHorView	NPS满意度组件
NSManager+TGCCard.h	H5VideoExtHelper	列webview定制视频组件
NSerView	BannerView	Banner轮播组件
NavigationController	NavigationView	导航控制器组件
GridView	GridView	9宫格布局组件

### Android Material Design 组件详细说明

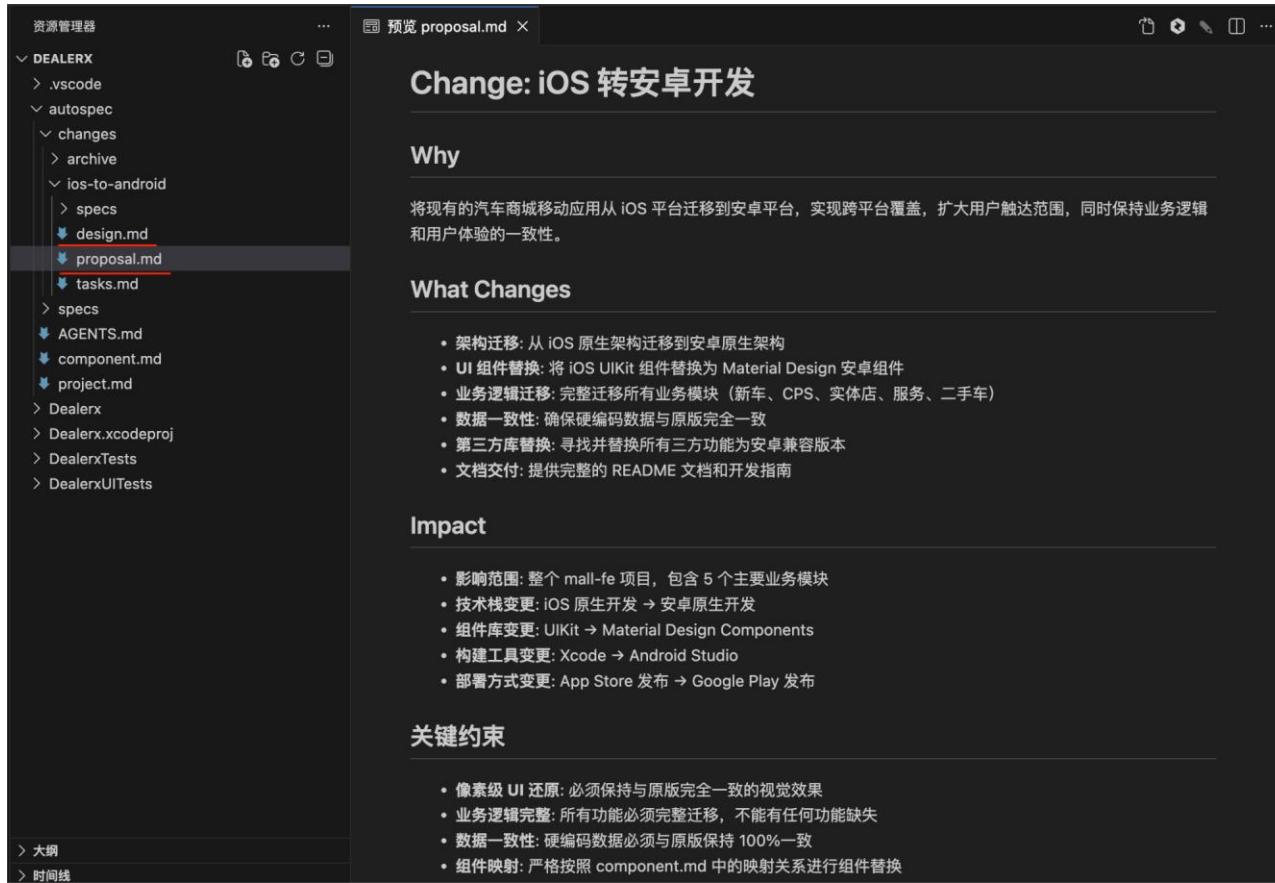
#### 基础组件

NPS满意度组件

- 功能特性: 提供 5 点量表形式的满意度评价功能，用户可通过星级评分 (1 分 “太差了” 至 4 分 “比较好” 等) 反馈满意度，支持选填评价文字内容。
- 使用场景: 适用于汽车之家 App 内车型详情页等场景，在用户浏览车型信息 (如参数配置、报价、口碑等) 后，弹出满意度卡片收集用户对当前页面 / 功能的反馈。
- 核心属性:
  - type 调用类型参数，用于区分不同调用场景或配置
  - cardtype 卡片类型标识，固定为 5 (对应 5 点量表满意度卡片)
  - scene 场景标识，用于记录评价所属业务场景
  - seriesid 车型系列 ID，关联具体评价的车型系列
  - specid 车型规格 ID，关联具体评价的车型规格
  - content 评价相关内容描述，用于补充场景说明
  - seriesname 车型系列名称 (需编码传入)，展示评价对应的车型信息
  - extjson 扩展参数 (需编码传入)，可包含 bbsid、bbsname 等自定义信息
  - NPSCardPVBean 场景数据存储实体，包含 scene, content, seriesId, specId 等属性，用于统计

解析源工程技术栈 (project.md)，建立 iOS → Android 组件映射表(component.md)。

# 提案与架构设计



资源管理器

- DEALERX
- .vscode
- autospec
- changes
- archive
- ios-to-android
- specs
- design.md
- proposal.md
- tasks.md
- AGENTS.md
- component.md
- project.md
- Dealerx
- Dealerx.xcodeproj
- DealerxTests
- DealerxUITests

预览 proposal.md ×

## Change: iOS 转安卓开发

### Why

将现有的汽车商城移动应用从 iOS 平台迁移到安卓平台，实现跨平台覆盖，扩大用户触达范围，同时保持业务逻辑和用户体验的一致性。

### What Changes

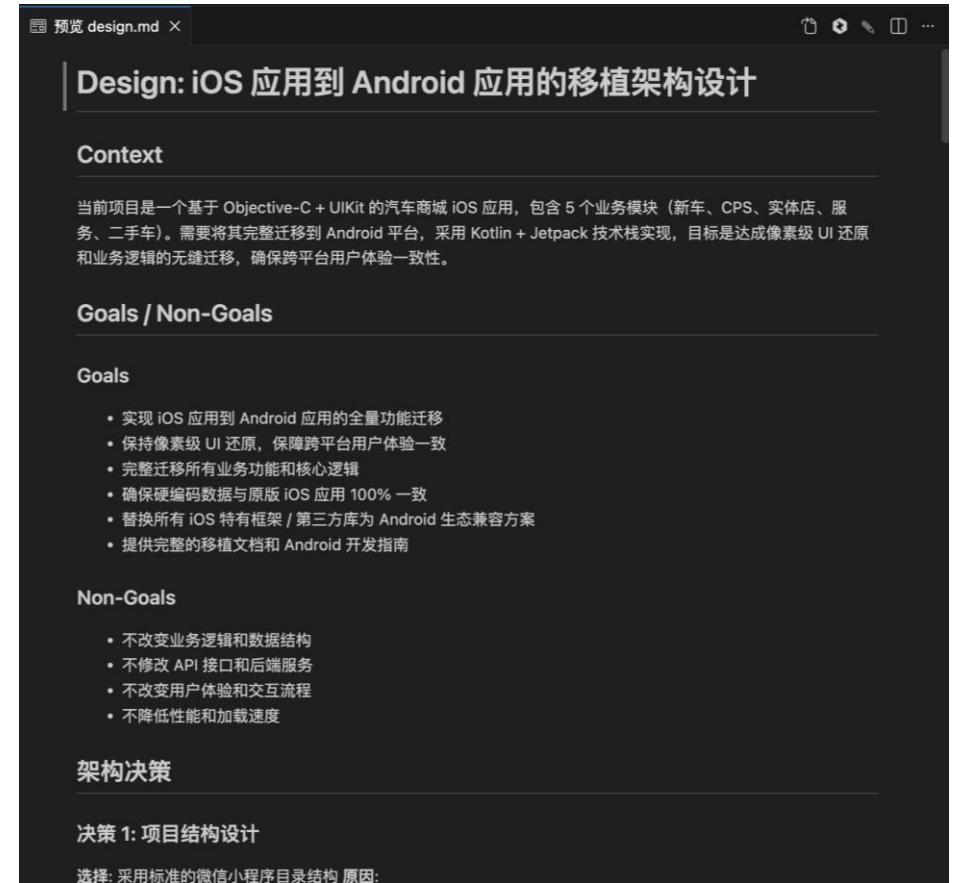
- 架构迁移: 从 iOS 原生架构迁移到安卓原生架构
- UI 组件替换: 将 iOS UIKit 组件替换为 Material Design 安卓组件
- 业务逻辑迁移: 完整迁移所有业务模块（新车、CPS、实体店、服务、二手车）
- 数据一致性: 确保硬编码数据与原版完全一致
- 第三方库替换: 寻找并替换所有三方功能为安卓兼容版本
- 文档交付: 提供完整的 README 文档和开发指南

### Impact

- 影响范围: 整个 mall-fe 项目，包含 5 个主要业务模块
- 技术栈变更: iOS 原生开发 → 安卓原生开发
- 组件库变更: UIKit → Material Design Components
- 构建工具变更: Xcode → Android Studio
- 部署方式变更: App Store 发布 → Google Play 发布

### 关键约束

- 像素级 UI 还原: 必须保持与原版完全一致的视觉效果
- 业务逻辑完整: 所有功能必须完整迁移，不能有任何功能缺失
- 数据一致性: 硬编码数据必须与原版保持 100% 一致
- 组件映射: 严格按照 component.md 中的映射关系进行组件替换



预览 design.md ×

## Design: iOS 应用到 Android 应用的移植架构设计

### Context

当前项目是一个基于 Objective-C + UIKit 的汽车商城 iOS 应用，包含 5 个业务模块（新车、CPS、实体店、服务、二手车）。需要将其完整迁移到 Android 平台，采用 Kotlin + Jetpack 技术栈实现，目标是达成像素级 UI 还原和业务逻辑的无缝迁移，确保跨平台用户体验一致性。

### Goals / Non-Goals

#### Goals

- 实现 iOS 应用到 Android 应用的全量功能迁移
- 保持像素级 UI 还原，保障跨平台用户体验一致
- 完整迁移所有业务功能和核心逻辑
- 确保硬编码数据与原版 iOS 应用 100% 一致
- 替换所有 iOS 特有框架 / 第三方库为 Android 生态兼容方案
- 提供完整的移植文档和 Android 开发指南

#### Non-Goals

- 不改变业务逻辑和数据结构
- 不修改 API 接口和后端服务
- 不改变用户体验和交互流程
- 不降低性能和加载速度

### 架构决策

#### 决策 1: 项目结构设计

选择: 采用标准的微信小程序目录结构 原因:

基于工程分析自动生成 proposal.md (迁移方案与约束) 和 design.md (架构决策与技术选型)

# 任务拆解与执行计划

- 有序编排

任务按执行顺序排列，前置任务完成后才推进后续任务

- 复用优先

优先完成组件层和通用能力，业务模块直接复用

- 粒度可控

每个任务明确输入输出，可独立执行、可单独验证

- 可追踪

任务状态清晰，便于进度跟踪、代码审查和质量验收

The screenshot shows a code editor interface with a sidebar and a main content area. The sidebar on the left is titled '资源管理器' (File Explorer) and shows a tree view of a project structure. The 'tasks.md' file is selected in the tree. The main content area has a title 'Tasks: iOS 转安卓开发' and is divided into numbered sections:

- 1. 项目初始化和基础架构搭建
  - [] 1.1 创建 Android 项目结构
  - [] 1.2 配置 Android 基础框架和目录结构
  - [] 1.3 集成 Material Design 组件库
  - [] 1.4 配置开发环境和构建工具 (Android Studio)
  - [] 1.5 建立项目基础配置文件
- 2. 核心组件库迁移
  - [] 2.1 分析现有 iOS UIKit 组件使用情况
  - [] 2.2 建立 iOS 组件到 Android Material Design 的映射关系
  - [] 2.3 创建自定义组件封装层，兼容原有 API
  - [] 2.4 迁移 Layout、PageLoading、Empty 等基础组件
  - [] 2.5 迁移业务通用组件
- 3. 路由和状态管理迁移
  - [] 3.1 分析现有 iOS 导航控制器配置
  - [] 3.2 设计 Android Activity 和 Fragment 路由结构
  - [] 3.3 实现页面跳转和参数传递机制 (Intent)
  - [] 3.4 迁移状态管理逻辑 (iOS State 到 Android ViewModel)
  - [] 3.5 实现 Activity 和 Fragment 生命周期管理
- 4. 业务模块迁移 - 新车模块 (new-car)
  - [] 4.1 分析 new-car 模块的页面结构和功能
  - [] 4.2 new-car 模块的样式如图：[图片链接](#)
  - [] 4.3 迁移车源订单详情页 (car-order-detail)
  - [] 4.4 迁移店铺详情页 (shop-details)
  - [] 4.5 迁移支付成功页 (payment-success)

04

## 案例2：Design to code 高 UI还原度的技术突破

# Design to code 高还原度的关键要素

## 像素级的还原度

精确还原间距、字号、颜色等视觉细节，确保像素级精准

准确生成并实现卡片中的交互信息(如左右滑动、展开收起等)

01

## UI组件复用

自动识别重复的UI 模块并抽离为独立组件

已有组件库中的组件,优先复用而非重复创建

02

## 布局适配能力

支持响应式布局，适配不同屏幕尺寸与分辨率

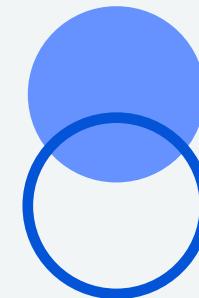
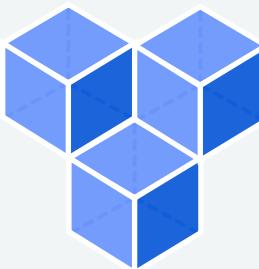
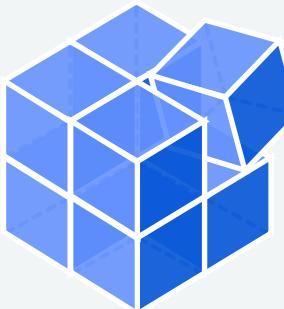
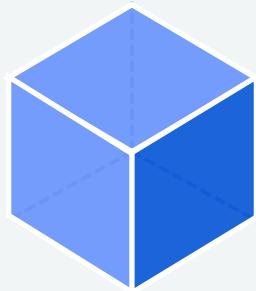
生成结构清晰的弹性代码，满足多端部署要求

03

# Design to code 的工程化路径

## 核心概念: “切”

把复杂页面简单化, 把简单问题做精准



结构化分析

组件拆分生成

页面拼装

验证纠错

四大原则

1 原子化拆分

2 组件复用

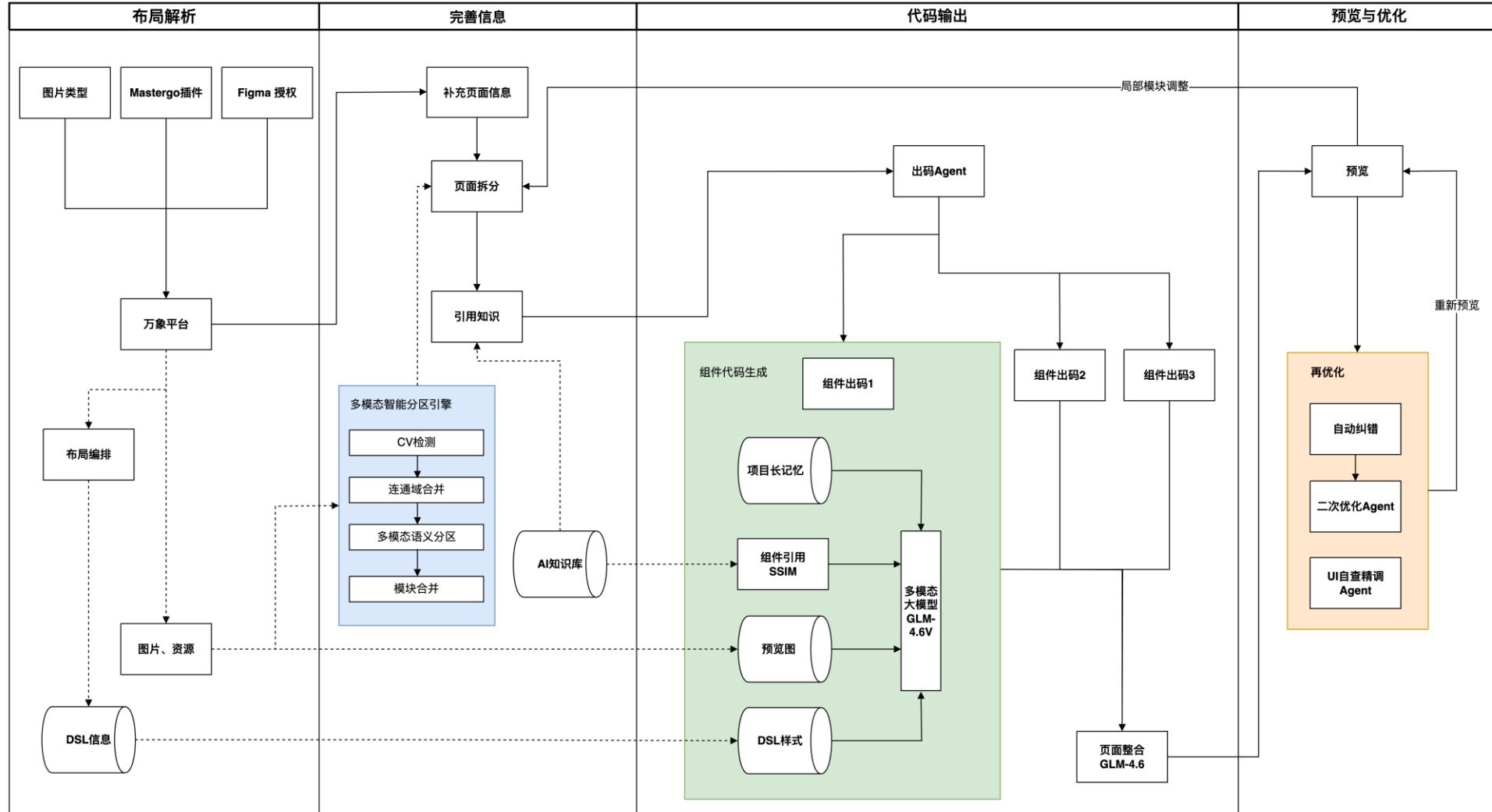
3 渐进式增强

4 边界清晰化

准则

方法

# D2C: 多模态分析+DSL解析



# 拆分示例

## 自动拆分卡片

将页面拆分为多个卡片区域，用户可再进行拖拽修正

## 自动识别组件

基于结构相似度(SSIM)查找到引用的组件

## 卡片要求描述

可以描述组件中的特殊交互与隐形要求

当前共有 6 个组件需要完善信息

一键生成所有组件说明

组件详细信息 (6个)

组件 1  已完成

组件 2  已完成

AI 分析组件

组件说明

- \*\*CarInfoCard\*\* - 汽车信息卡片，包含汽车图片展示区、型号信息和价格优惠信息
- \*\*CarImageGallery\*\* - 汽车图片展示区，包含外观和内饰两张图片及对应颜色标签
- \*\*CarPricingInfo\*\* - 汽车价格信息组，包含新车价、厂商指导价和优惠信息

标题是可以配置@Watermark 按照这个规范

已引用的知识库项目:

Watermark

组件 3  已完成

上一步  开始出码

# AI驱动的UI迭代优化

- 视觉差异自动检测，像素级精准对比

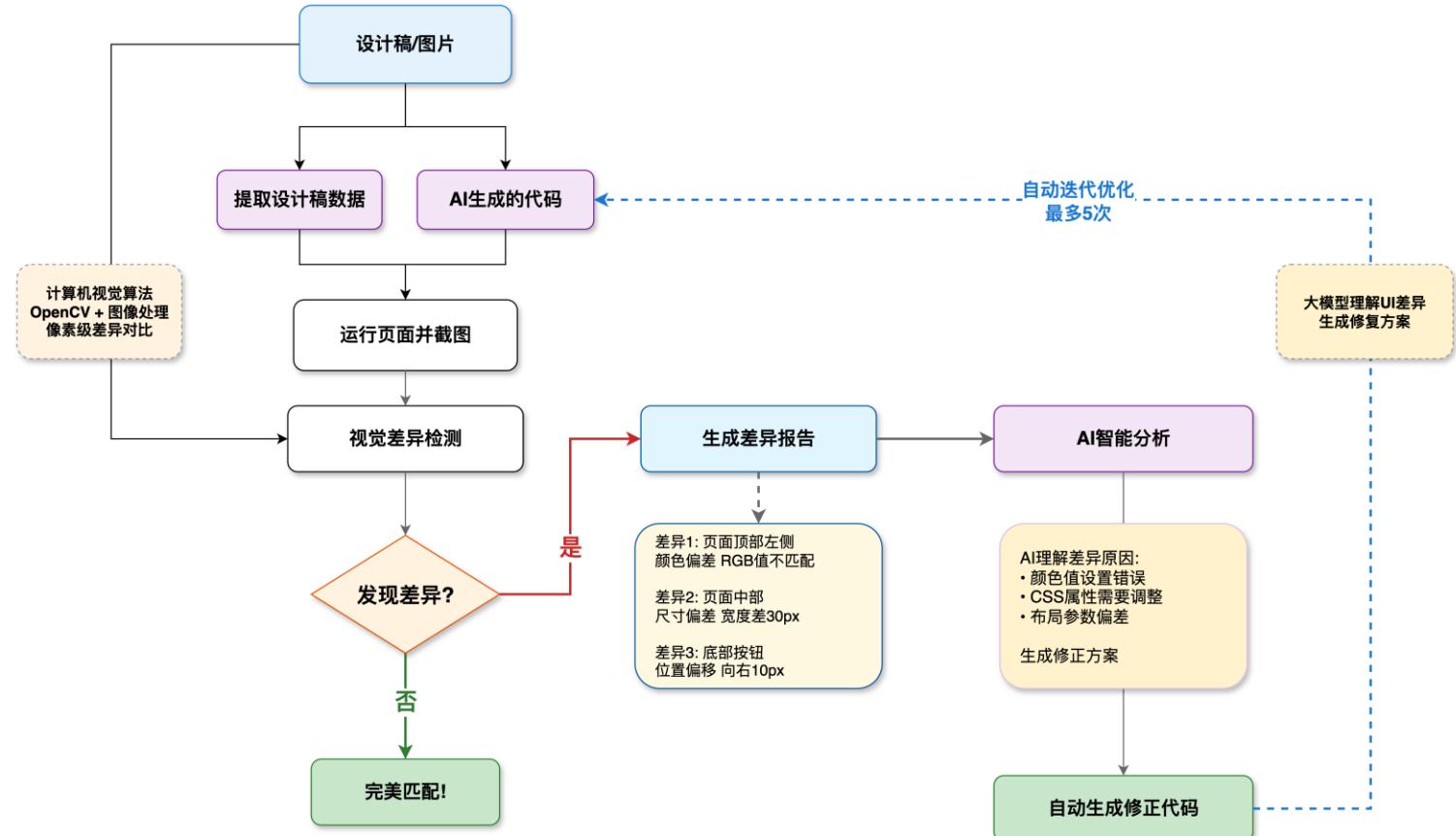
统一设计稿和运行截图数据，基于OpenCV(热力图)+OCR，自动标注出颜色、字号、布局差异等

- AI智能分析差异，自动生成修复方案

深度解析差异原因，输出针对性的代码优化建议

- 迭代优化闭环，持续提升还原精度

自动迭代修复直至达标，确保生成代码与设计稿高度一致



# D2C演示



该截图展示了万象编码平台的D2C功能。顶部是浏览器界面，显示网址为wanxiang.corpautohome.com/d2c/smartcode。左侧是左侧栏，包含“新聊天”、“历史聊天”、“快速上手”和“生成效果对比”等选项。右侧上方是“你想要构建什么？”的输入框，下方有“快速入门”按钮，通过设计稿生成代码。下方展示了“智能 UI 代码生成”功能，说明“设计稿、图片皆可智能出码”，并展示了从设计稿到生成代码的流程。底部有“意见反馈”和“关于我们”链接。

万象支持图片、Figma、MasterGo多种格式

# 05 总结与展望

# ■ 总结：理解AI能力边界，扬长补短

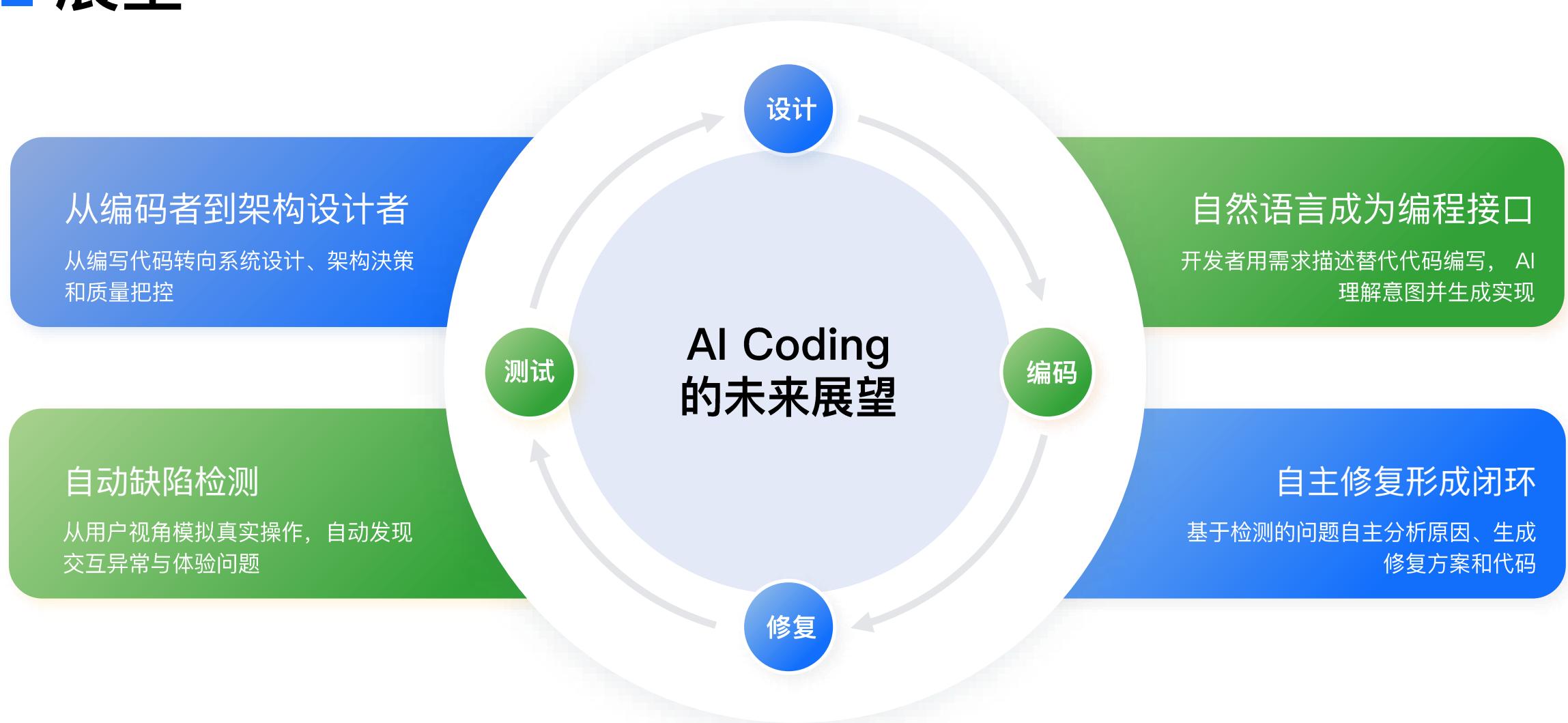
## 补齐短板

- 长token遗忘  
上下文分段与记忆管理
- 视觉理解的局限性  
复杂UI结构简化与拆解
- 业务理解缺失  
业务DSL与知识图谱

## 发挥长板

- 强大的代码生成能力  
专注AI擅长的代码编写
- 逻辑推理与决策  
智能分析与架构设计
- 跨技术栈能力  
快速适配多种技术场景

# 展望



# 极客邦科技 2026 年会议规划

促进软件开发及相关领域知识与创新的传播



参会咨询



查看会议

北京

1200人

**QCon**

全球软件开发大会

会议时间：4月16-18日

- Agentic Engineering
- AgentOps
- 下一代模型架构与推理优化
- AI 原生基础设施
- 知识工程实践
- AI 安全

深圳

1000人

**AiCon**

全球人工智能开发与应用大会

会议时间：8月21-22日

- Agentic AI
- 轻量化与高效推理
- 多模态应用
- AI + IoT 场景实践
- AI 工业化落地

北京

1000人

**AiCon**

全球人工智能开发与应用大会

会议时间：12月18-19日

- 大模型架构创新
- 多模态 AI 产业融合
- 具身智能
- AI for Science
- 大模型安全

4月

6月

8月

10月

12月

**AiCon**

全球人工智能开发与应用大会

会议时间：6月26-27日

- AI Infra 系统工程
- 多 Agent 协作与实践
- 多模态融合
- 模型训练与推理创新
- 数据平台与特征服务

上海

1000人

**QCon**

全球软件开发大会

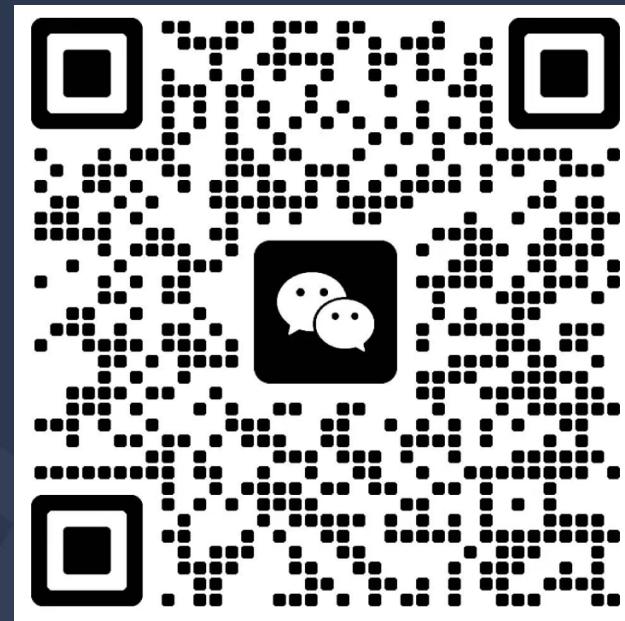
会议时间：10月22-24日

- AI Agent
- Vibe Coding
- 智能可观测
- 推理基建
- 模型攻防
- AI x 创造力

上海

1200人

# Q&A



# THANKS

探索 AI 应用边界

Explore the limits of AI applications

AiCon

全球人工智能开发与应用大会