

# 基于演进式设计的LLM辅助开发实践

张刚 | 软件工程博士，资深技术专家

# 科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



 **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**  **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

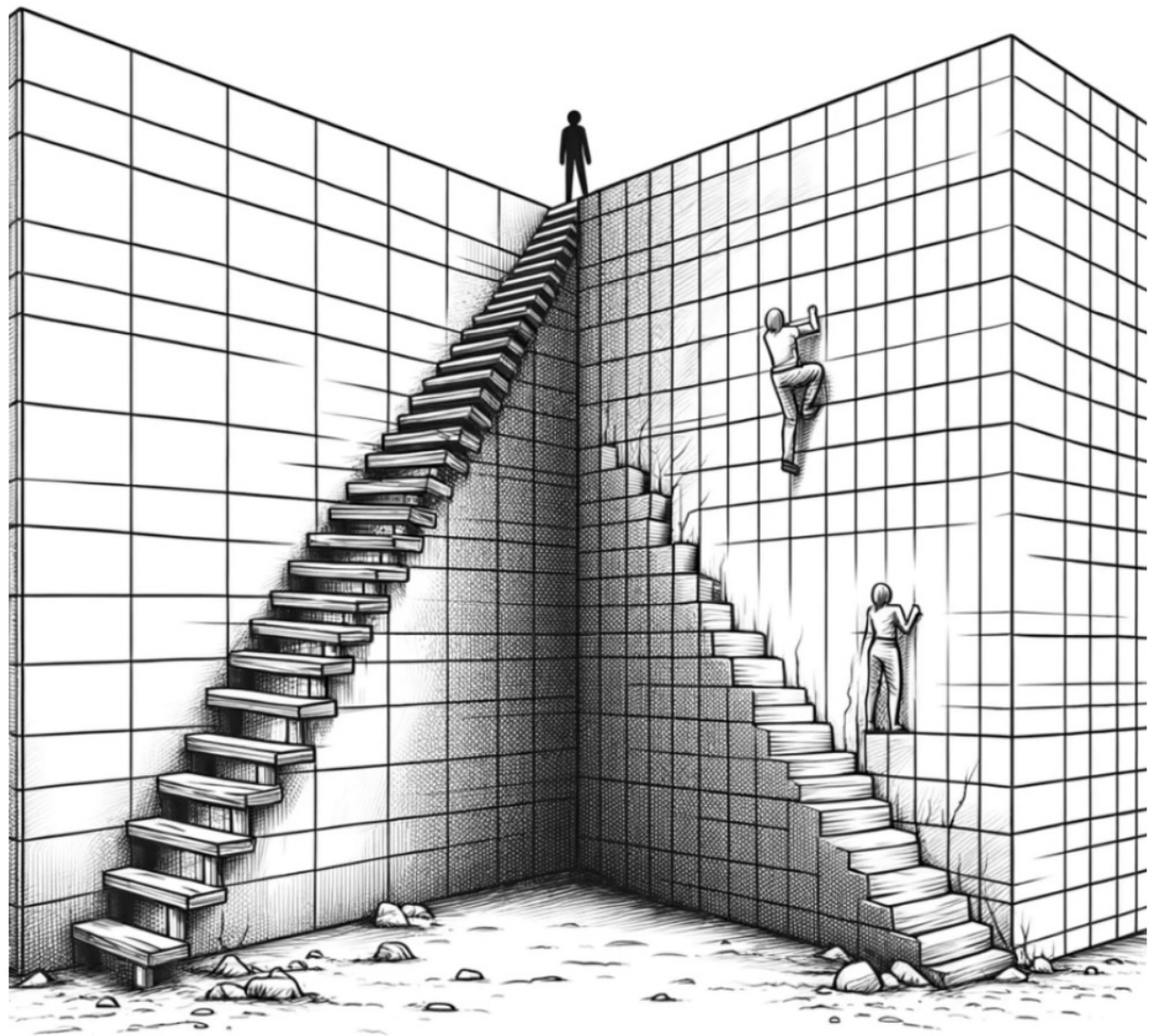


AiDD峰会详情



## ▶ 关于演进式设计

为什么LLM驱动的软件开发都特别喜欢演示“贪吃蛇”这种游戏，而不是真实的软件项目？



# ► Canvas的体验革新，其实也是“演进式设计”

📄 ✎ ChatGPT 4o with canvas ▾

React+antd+zustand完成这个设计。

🌀 ↔ Toolbar Design With Zustand

Here's a basic implementation of the toolbar using React, Ant Design, and Zustand. The toolbar lists element types that are allowed within the "diagram.class" configuration, and clicking on a toolbar button will select the corresponding element type.

Does this approach align with what you're aiming for? Feel free to let me know if you need more modifications or explanations!

🔊 📄 👍 🗨

📎 给"ChatGPT"发送消息 ↑

ChatGPT 也可能会犯错。请核查重要信息。 ?

✕ Toolbar Design With Zustand 📄 MA

```
1 import React from 'react';
2 import { Button, Tooltip } from 'antd';
3 import { useStore } from 'zustand';
4 import { create } from 'zustand';
5 import { elementConfig } from './elementConfig'; // Assume this is the provided config file
6 import { VerticalAlignMiddleOutlined } from '@ant-design/icons'; // Import any placeholder icons needed
7
8 // Define the Zustand store for managing selected element type
9 interface ToolbarState {
10   selectedElementType: string | null;
11   selectElementType: (elementType: string) => void;
12 }
13
14 const useToolbarStore = create<ToolbarState>((set) => ({
15   selectedElementType: null,
16   selectElementType: (elementType) => set({ selectedElementType: elementType }),
17 }));
18
19 // Toolbar component
20 const Toolbar: React.FC = () => {
21   const { selectedElementType, selectElementType } = useStore(useToolbarStore);
22
23   // Filter element types that are allowed in 'diagram.class'
24   const toolbarElements = Object.entries(elementConfig).filter(
25     ([key, config]) => config.allowedNodeTypes?.includes('node.class') || config.allowedNodeTypes
26   );
27
28   return (
29     <div style={{ display: 'flex', flexDirection: 'column', gap: '8px', padding: '8px' }}>
30       {toolbarElements.map(([elementType, config]) => (
31         <Tooltip title={config.displayName} key={elementType}>
32           <Button
```

🗨  
📄  
🔍  
🗨  
[>]





## 张刚

软件工程博士，资深技术专家。

著有《软件设计：从专业到卓越》、《大模型辅助软件开发》，个人公众号“演进式设计”。从未脱离技术一线，是一名超过20年码龄的老程序员。曾担任Alcatel-Lucent 固网集团 ISAM架构负责人，阿里巴巴数字供应链服务化治理负责人，P9技术专家。

主要作品：

- ISAM软件架构
- 阿里巴巴数字供应链服务化治理体系
- “共享出行”软件工程开源案例
- Depends代码依赖分析工具

目前致力于业务数字化和软件工程领域的方法论和工具的建设与完善。



## 作为本次大会唯一的3小时工作坊，我希望给大家带来什么？

- 经验分享：如何使用大模型来提升开发效率
  - 工程方法：
    - 在大模型时代，软件工程能力究竟是不重要了？还是更重要了？
    - 为什么我会说：软件开发是一个持续探索和发现的过程？
  - 未来思考：
    - 更适合开发者的未来开发形态是怎样的？
- 
- 不是关于如何利用工具和编写提示词的技巧的工作坊
  - 不涉及如何开发AI应用



# 目录

## CONTENTS

1. 背景：大模型如何改变了我的工作方式？
2. 案例研究：如何用大模型辅助，完成一个端到端的项目？
3. 方法探寻：哪些工程方法值得关注？
4. 动手实践：
  - 用演进式设计的思路来完成一个小产品
  - 挑战更复杂的功能
5. 总结和探讨

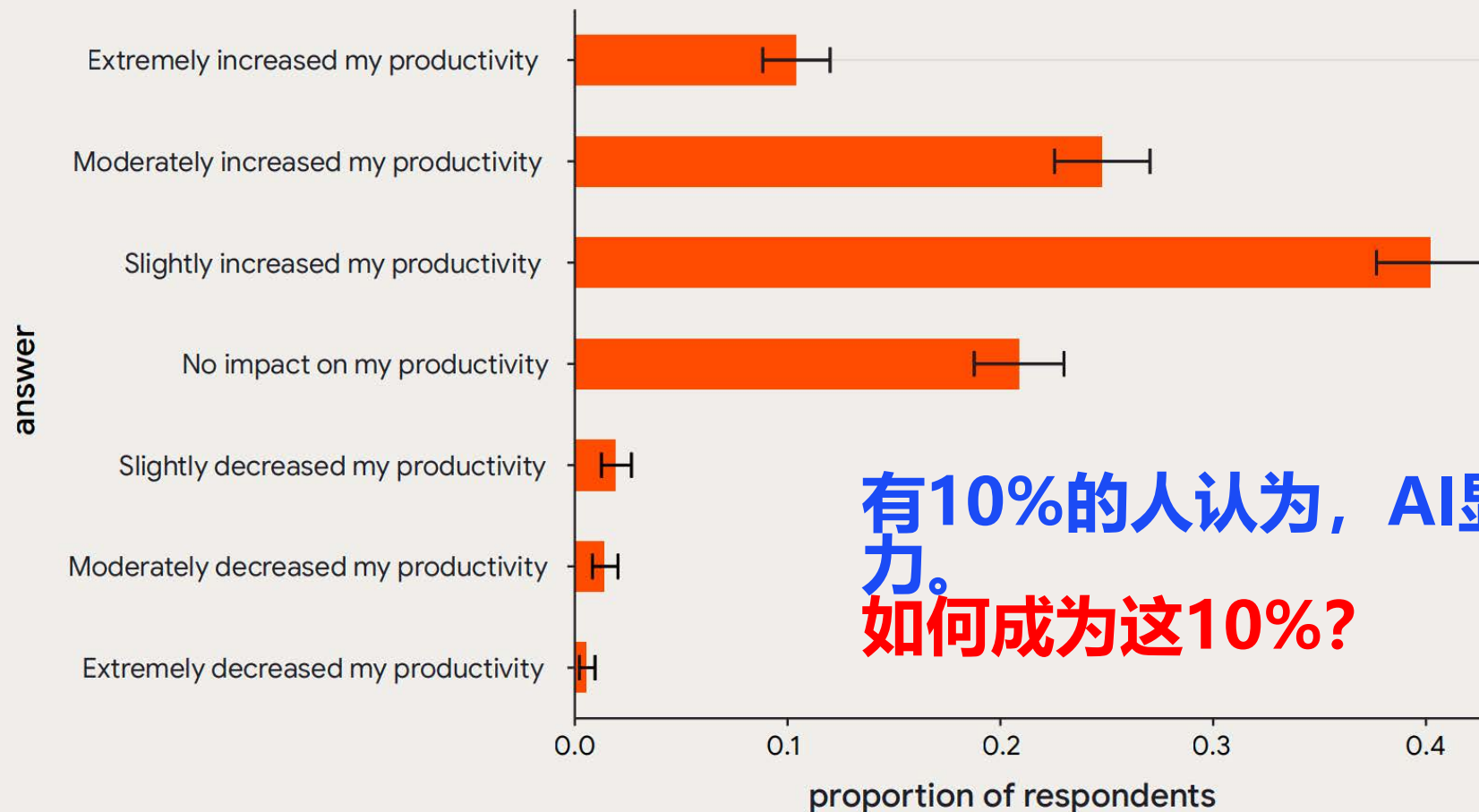
## ▶ 真的是这样吗？

“大模型来了，程序员要失业了”的说法比较少了，  
“大模型不好用”的说法，却渐渐多了起来



# ► DORA2024: AI在多大程度上提升了开发者的生产力?

## Perceptions of Productivity Changes Due to AI



有10%的人认为，AI显著提升了生产力。  
如何成为这10%?

# PART 01

## 大模型如何改变了我的工作方式?



# ▶ 更高效率，更有能力

## • 之前

- 有一个想法 → **产品经理**定义产品方案 → **开发团队**排期（等待） → 开发 → 改BUG，改BUG → 上线 → **调整新的产品方案** → →

## • 现在：

- 有一个想法 → 定义产品路线图 → 开发 → 调整想法 → 开发 → 上线 → 继续开发

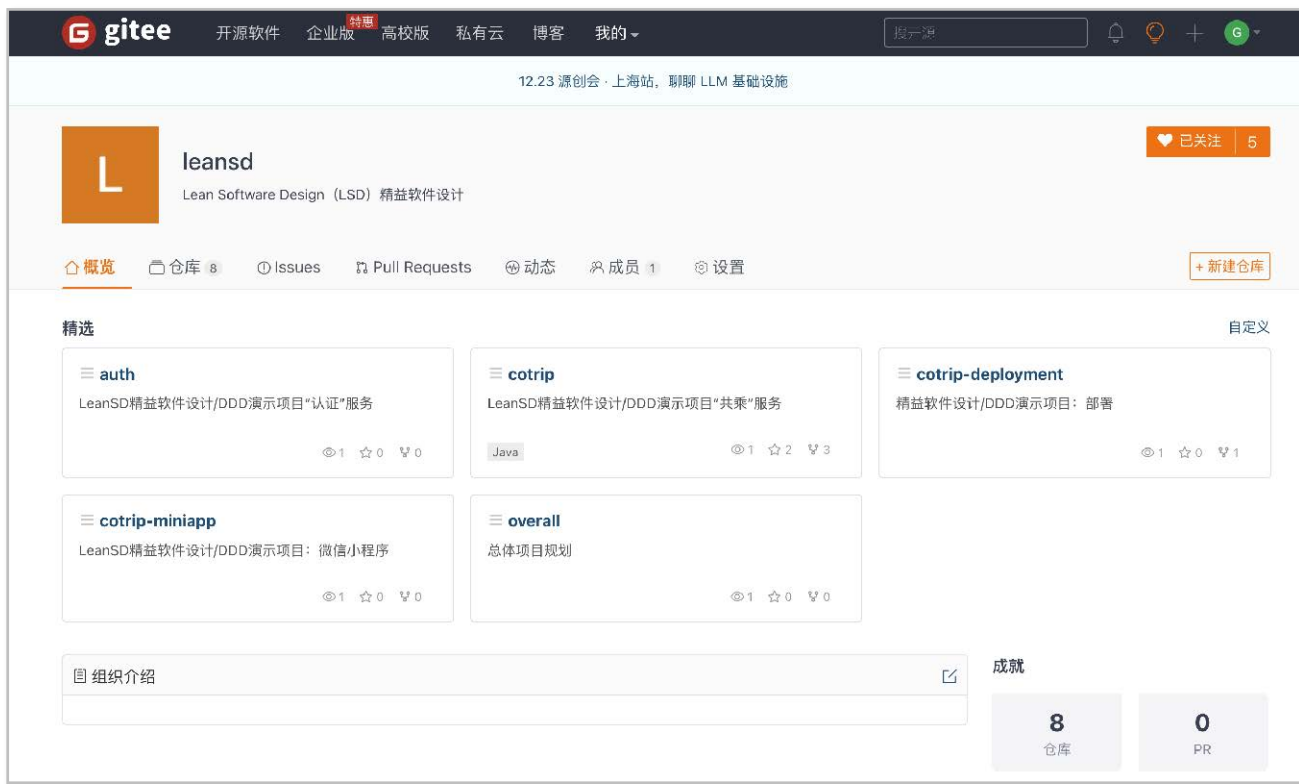
从想法到上线的周期大幅缩短  
个人更容易做到全栈  
团队规模可大幅缩小

# ▶ 用大模型辅助完成的一些作品：共享出行





# ▶ 开源代码



2023年8月-9月，使用ChatGPT 及 github copilot，完成了一个准生产环境的DDD的开源案例：共享出行。

# 案例详解



## 目录

<b>第1章 认识高效软件开发</b>	1.1 高效探索和发现 1.2 建设专业能力 1.3 用大模型提升开发效率
<b>第2章 业务规划和流程分析</b>	2.1 案例背景 2.2 精益创业和最小可行产品 2.3 共享出行的业务规划 2.4 业务流程分析 2.5 共享出行的业务流程分析
<b>第3章 分析系统需求, 澄清需求细节</b>	3.1 需求分析活动概览 3.2 大模型如何支持需求分析 3.3 管理需求全景图 3.4 在需求分析过程中沉淀领域模型 3.5 需求澄清和实例化
<b>第4章 构建初始架构</b>	4.1 架构的使命和目标 4.2 大模型如何支持架构设计 4.3 理性决策和演进式架构 4.4 共享出行的关键架构决策 4.5 从构建一个空系统开始 4.6 制定自动化测试方案
<b>第5章 实现核心域</b>	5.1 用大模型辅助开发核心域 5.2 用代码表达领域模型 5.3 发布出行计划 5.4 撮合出行计划 5.5 通过WebSocket 发送通知
<b>第6章 实现通用域: 以认证授权为例</b>	6.1 认证授权基础 6.2 Keycloak 基础 6.3 用大模型辅助开发认证授权服务 6.4 用大模型辅助开发自动化测试 6.5 用大模型辅助集成Spring 安全配置
<b>第7章 构建持续集成基础设施</b>	7.1 持续集成加速演进式设计 7.2 用大模型辅助构建持续集成流水线 7.3 用大模型支持容器化部署
<b>第8章 实现微信小程序</b>	8.1 微信小程序开发基础 8.2 规划界面原型 8.3 用大模型辅助前端开发 8.4 用大模型辅助完成权限管理
<b>第9章 持续演进</b>	9.1 业务发展要求演进式设计 9.2 上车点管理 9.3 顺风车业务

# ▶ 可以使用哪些工具?

The image displays the Cursor AI IDE interface. The main editor shows a TypeScript file named `ESNode.tsx` with code for handling node operations like `updateNodeData`, `getNode`, `deleteNode`, and `handleKeyDown`. A large "Cursor" watermark is overlaid on the code. On the right, the "CHAT" panel shows a conversation where the user asks for a simplified code example for adding drag functionality to intermediate points. The AI assistant responds with a code snippet for `SelfLoopEdge` that uses `useState` to manage a list of points and a `useCallback` for handling drag events. The interface also includes a file explorer on the left, a terminal at the bottom, and a "PROBLEMS" panel.



以判断的速度编码!

```
commit 5b394792700d91c0d09bc99fc0011e0e4ca13234 (HEAD -> main, origin/main)
Author: gangz <gangz2009@gmail.com>
Date: Thu Oct 10 19:45:34 2024 +0800

    can align node vertically

commit ea53b9749aa1d172f2254da5f77c7d4ccffea239
Author: gangz <gangz2009@gmail.com>
Date: Thu Oct 10 19:38:16 2024 +0800

    add toolbar for Horizontally align

commit 5fd2ca68b6d67cf3a1fe439438030e7e38445256
Author: gangz <gangz2009@gmail.com>
Date: Thu Oct 10 19:30:58 2024 +0800

    can align node Horizontally
```

# ▶ 可以使用哪些工具?

📄 ✍️ ChatGPT ▾

- 🌀 ChatGPT
- 📊 Diagrams <Show Me> f...
- 🔍 探索 GPT

今天

- 事件驱动架构关键概念
- 提取SVG元素为组件

昨天

- 代码检查与改进建议 ...
- 自定义ReactFlow边实现

前 7 天

- 使用SVG文件的方法
- React TypeScript Sticky Note
- Zundo Undo Redo Implementation
- Zustand Zundo 状态管理分析

记忆已满 ⓘ

我不了解“防抖动或节流技术”，请帮我介绍



“防抖动”（Debounce）和“节流”（Throttle）是两种常用的性能优化技术，主要用于控制频繁触发的事件（如滚动、窗口调整大小、输入框输入等），以避免不必要的计算或处理。这两种技术虽然都能帮助减少函数的调用次数，但它们的应用场景和实现方式有所不同。

## 1. 防抖动（Debounce）

**定义:** 防抖动是一种在事件触发后延迟执行的技术，只有在事件停止触发后的一段时间后才执行处理函数。换句话说，如果在规定的时间内再次触发事件，则取消之前的执行，重新计时。

**应用场景:**

- 输入框搜索（如用户输入时触发搜索，减少 API 请求次数）
- 窗口调整大小（在用户停止调整大小后再进行处理）

**示例代码:**

```
javascript
```

↓

复制代码

# ▶ 可以使用哪些工具?

ChatGPT 4o with canvas ▾

React+antd+zustand完成这个设计。

🔗 Toolbar Design With Zustand

Here's a basic implementation of the toolbar using React, Ant Design, and Zustand. The toolbar lists element types that are allowed within the "diagram.class" configuration, and clicking on a toolbar button will select the corresponding element type.

Does this approach align with what you're aiming for? Feel free to let me know if you need more modifications or explanations!

🔊 📄 👍 🗨

📎 给"ChatGPT"发送消息

ChatGPT 也可能会犯错。请核查重要信息。

```
✕ Toolbar Design With Zustand MA  
1 import React from 'react';  
2 import { Button, Tooltip } from 'antd';  
3 import { useStore } from 'zustand';  
4 import { create } from 'zustand';  
5 import { elementConfig } from './elementConfig'; // Assume this is the provided config file  
6 import { VerticalAlignMiddleOutlined } from '@ant-design/icons'; // Import any placeholder icons needed  
7  
8 // Define the Zustand store for managing selected element type  
9 interface ToolbarState {  
10   selectedElementType: string | null;  
11   selectElementType: (elementType: string) => void;  
12 }  
13  
14 const useToolbarStore = create<ToolbarState>((set) => ({  
15   selectedElementType: null,  
16   selectElementType: (elementType) => set({ selectedElementType: elementType }),  
17 }));  
18  
19 // Toolbar component  
20 const Toolbar: React.FC = () => {  
21   const { selectedElementType, selectElementType } = useStore(useToolbarStore);  
22  
23   // Filter element types that are allowed in 'diagram.class'  
24   const toolbarElements = Object.entries(elementConfig).filter(  
25     ([key, config]) => config.allowedNodeTypes?.includes('node.class') || config.allowedNodeTypes  
26   );  
27  
28   return (  
29     <div style={{ display: 'flex', flexDirection: 'column', gap: '8px', padding: '8px' }}>  
30       {toolbarElements.map(([elementType, config]) => (  
31         <Tooltip title={config.displayName} key={elementType}>  
32           <Button
```





# ▶ 可以使用哪些工具? -- 国产大模型也很好用



```
</>  
);  
};
```

```
export default ESNode;
```

请你关注handleKeyDown这个方法。我发现，虽然isEditMode发生了变化，担心这个函数中的值没有发现变化。我认为这是因为TypeScript定义函数的机制导致的。请你予以修改，让代码工作正确。

## ChatGLM AI

你提到的问题可能是由于在 `useEffect` 钩子中直接使用了 `handleKeyDown` 函数，而没有将其包含在依赖项数组中。在TypeScript中，这通常不会引起问题，因为函数在整个生命周期内是稳定的。但是，如果你在 `handleKeyDown` 函数内部使用了组件状态（如 `isEditMode`），并且希望在状态变化时重新绑定事件监听器，那么确实需要确保这些状态也被包含在依赖项数组中。

在 `useEffect` 钩子中，你应该将 `handleKeyDown` 函数内部使用的所有外部变量作为依赖项，这样当这些变量变化时，`useEffect` 可以重新运行并更新事件监听器。在你的例子中，`handleKeyDown` 函数依赖于 `selected` 和 `node.modelNodeId`，但是它也隐式依赖于 `isEditMode`。

以下是修改后的代码：

+ 新建对话

输入@，召唤智能体

**有没有可能，发展到一定阶段，  
软件开发是人人能做的工作？**

## ► 我的观点

**软件开发仍然是一项专业技能。它的本质，是数字化世界的构建：**

**--- “现实世界问题在计算空间的映射”**

**但是，复杂性的挑战已经迁移：**

- 构建不再是有挑战的问题
- 定义问题和设计方案才是



## **PART 02**

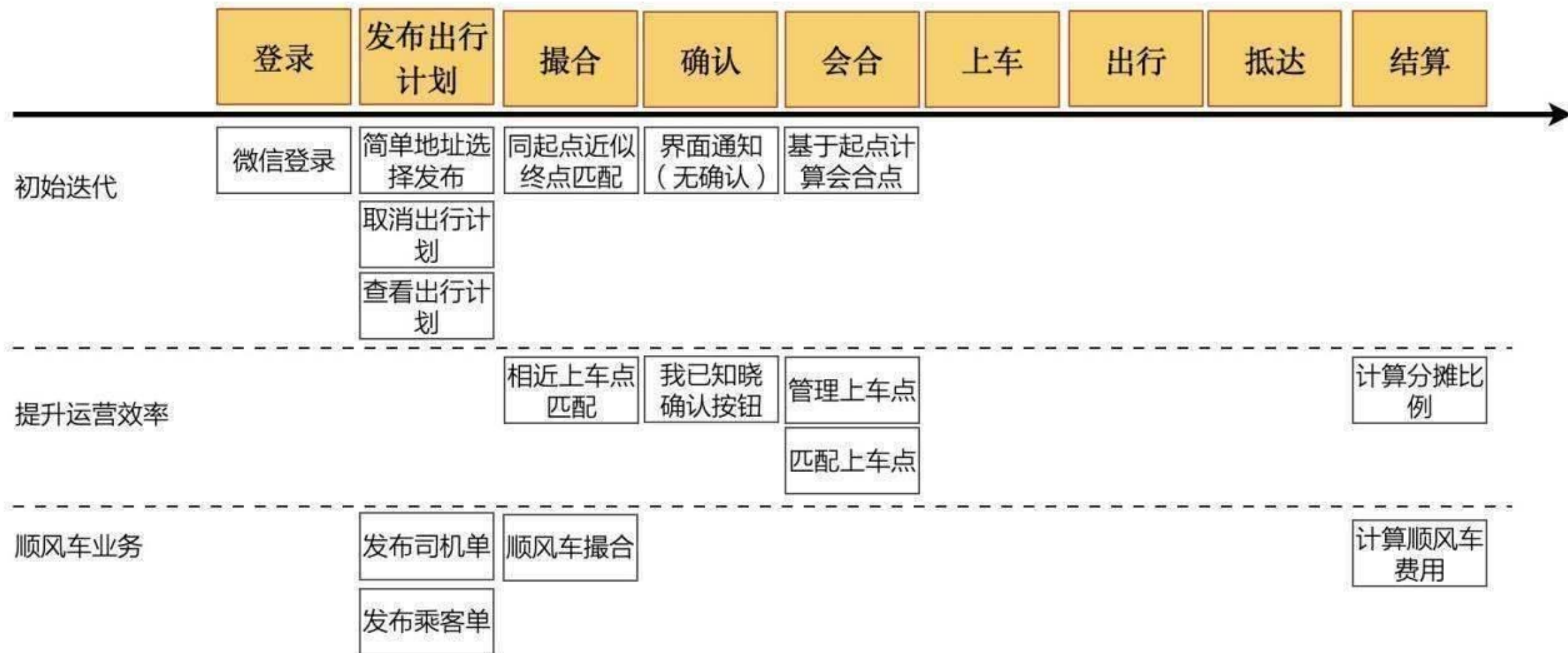
**案例研究：如何用大模型辅助，完成一个端到端的项目？**

# ▶ 共享出行



打造一个真正完整的、端到端的、可扩展的软件工程项目案例

# ▶ 定义产品路线图





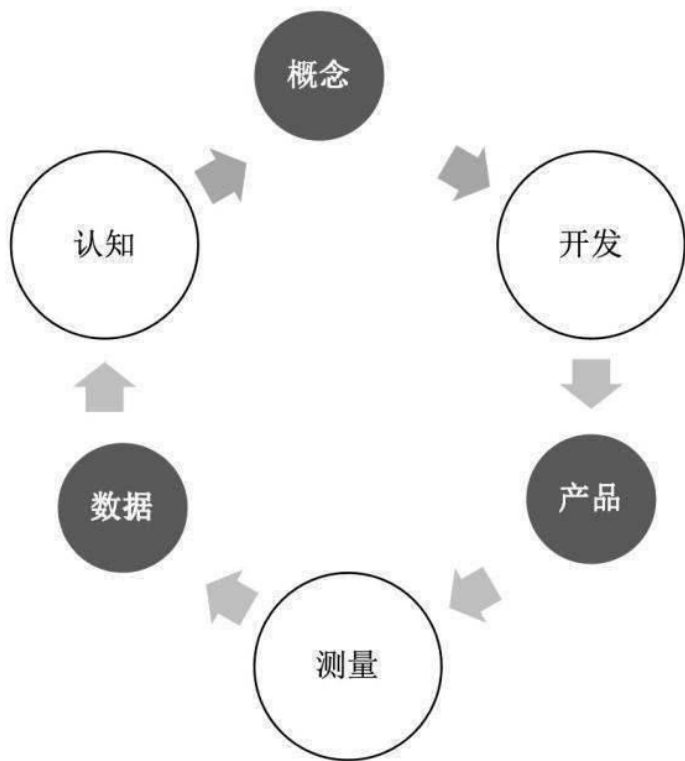
## ► 特定场景下的无车共乘

### 业务场景

在上海，春节过后火车站的人流量急剧增加，打出租车成了一项困难的任务，尤其在地铁停运后的夜间更是如此。火车站外排队的乘客队伍如一条长龙，一眼望不到头。然而，许多乘客都是一人乘车，如果恰好能有顺路的其他乘客一起，那么既能减少排队时间，提升载客效率，还能节省一部分的费用。

首个迭代的功能：用户说清起始地和目的地，以及大致的出发时间，平台撮合多个乘客，共乘一辆出租车。

## ► 为什么从“无车共乘”开始？



**创业是一个学习的过程而不是构建的过程。**

- 共享出行这个商业模式要求初期能够吸引物理位置集中的用户群。车站和机场场景符合这一要求，且推广成本较低。
- 这是一个刚需场景，用户的接受意愿较强。
- 无车共乘仅需要乘客参与，不需要司机参与，运营模式和开发都相对简单。
- 预期我们的产品将在春节前后投入运营，时间点契合，具备可行性。

# 演进式设计

是符合人类认知认知规律，  
也适合大模型协作的工作方式



# ► 共享出行核心域的初始架构

## 先从一个什么功能都没有的空系统开始

```
* Project: Maven Project
* Language: Java
* Spring Boot: 3.1.2 (选择最新稳定版本)
* Project Metadata: cotrip.demo.leandesign.cn
* Java Version: 17
* 依赖: *
  Spring Web: 它包含Spring MVC和Tomcat, 可以创建Web和REST应用。
* Spring Data JPA: 它允许你使用Spring和JPA进行数据库访问。
* Spring Boot DevTools: 它可以提供自动重启、热交换等便利的开发功能
。
* H2 Driver (用于简单的单元测试和集成测试)、MySQL Driver (用于系统测试和生产环境)。
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ExampleController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello, world!";
    }
}
```

```
tree -L 3
```

```
.
├── cotrip
│   ├── CotripApplication.java
│   ├── controller          ## 接口层
│   ├── service             ## 应用层
│   ├── domain              ## 领域层
│   └── infrastructure      ## 基础设施层
├── geo
└── site
```

```
@Entity(name = "users")
@Data
@NoArgsConstructor
@Builder
@AllArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String name;
}
```

```
import
org.springframework.data.repository.CrudRepository;
public interface UserRepository extends
    CrudRepository<User, Long> {
}
```

# ▶ 尝试开发第一个需求

## 提示

我正在开发一个共享出行的应用。其中有一个用例是：用户发布一个出行计划。出行计划包括起始地、目的地、出发时间等。请你根据用例的一般格式，撰写一个符合要求的用例。

## LLM

当然可以。根据您的需求，我将为“用户发布出行计划”编写一个概要。这个用例将涵盖上述所有要素。

**用例名称：**用户发布出行计划

**执行者：**

- 用户（系统的注册成员，寻求共享出行服务）

**场景：**

1. 用户登录共享出行应用
2. 选择发布出行计划的选项
3. 输入出行计划的详细信息，包括起始地、目的地、出发时间
4. 系统验证输入的信息
5. 用户提交出行计划
6. 系统记录并发布出行计划，供其他用户浏览
7. 用户收到出行计划发布成功的确认

**前置条件：**

- 用户已注册并能成功登录共享出行应用
- 系统正常运行，能接受并处理出行计划

**后置条件：**

- 出行计划成功发布在系统上，并对所有用户可见
- 用户的账户显示最新发布出行计划

**扩展：**

- 用户选择保存出行计划为草稿，稍后再发布
- 用户设定出行计划为私密，仅对特定用户组可见

**例外：**

1. 用户未登录：系统提示用户登录
2. 输入的信息不完整或格式错误：系统提示用户更正信息

**有没有发现什么问题？**



# ▶ 试着发现需求的模糊性

## 提示

我正在进行需求分析。我希望邀请你作为需求评审专家，帮助我审核：下面的需求有哪些地方比较模糊，需要进一步澄清？这个需求是：<发布出行计划用例的内容，此处略>

## LLM

首先，这个需求描述是相对清晰的，但仍存在一些模糊的地方，需要进一步澄清。以下是我认为需要进一步明确的部分：

### 1. 用户的个人信息：

- 除了手机号，还有哪些信息被视为“已经完善”？是否有其他关键信息必须完善？

### 2. 起始地和目的地：

- 这里应当明确起始地、目的地的格式或输入规范。例如，是选择城市、详细地址还是坐标？

### 3. 出发时间：

- 时间的精度如何设置？是到日期、小时还是分钟？是否需要考虑时区？

### 4. 系统默认乘车人数为 1：

- 如果用户更改乘车人数，是否存在上限和下限？

### 5. 未完成的出行计划的定义：

- “未完成的出行计划”这个概念需要进一步定义。例如，是说用户创建但未发布的计划？还是已发布但尚未开始的计划？

### 6. 初始迭代中的定义：

- 文中多次提到“初始迭代”，但迭代的定义和范围是什么？后续迭代中，规则会如何改变？

### 7. 出行计划状态：

- 除了“等待撮合”，出行计划是否有其他状态？状态之间的转换是如何进行的？

### 8. 发布成功的标准：

- 用户如何知道出行计划已成功发布？是否有提示信息或其他反馈机制？

考虑上述模糊之处，并与需求提出者或相关团队成员进行沟通，可以帮助需求更加清晰和完整。

**不要把大模型当成写文档的工具  
把它当成发现问题的助手**

# ► 澄清具体的需求规则

## 提示

我正在开发一个共享出行的业务。在本业务中，用户会提交自己的“出行计划”，系统会基于出行计划的匹配程度进行匹配，并创建“共乘”。其中，出行计划包括“起始地、目的地、出发时间、出行人数和状态（待撮合、已加入、已取消等。仅状态为‘待撮合’的可进行匹配）”，“共乘”则包括已经匹配的“出行计划”列表。我想到了一些策略：

1. 优先匹配已经存在的“共乘”。当已经存在的共乘不存在时，再匹配出行计划形成新的“共乘”
2. 只有出发时间段一致的才可以匹配。
3. 如果所有出行计划的出行人数超过了可提供的座位数（暂时固定为 4），不可匹配。
4. 加入同一个共乘的出行计划，任何两个计划的起始地的距离不应该超过 1 公里，和其他出行计划的目的地的距离，至少应该有一个不超过 2 公里。

请你按照由简单到复杂的顺序，整理上述共享出行的匹配策略，为它们编号，并尝试扩展更多规则。

## LLM

根据您提供的共享出行的匹配策略，我将按照由简单到复杂的顺序为它们编号，并试着进一步扩展：

1. 出发时间段匹配：只有出发时间段一致的才可以匹配。这是一个基本的规则，确保乘客都在相似的时间内出发。
  2. 状态判定：仅状态为“待撮合”的可进行匹配。避免已经匹配或取消的出行计划再次被匹配。
  3. 座位限制：如果共乘的出行人数超过了可提供的座位数（暂时固定为 4），不可匹配。这确保了车辆的座位不会被超额预订。
  4. 地点距离限制：加入同一个共乘的出行计划，任何两个计划的起始地的距离不应该超过 1 公里，和其他出行计划的目的地的距离，至少应该有一个不超过 2 公里。确保出行计划之间的地点都在合理的距离范围内，方便取乘客和送乘客。
  5. 优先匹配已有共乘：优先匹配已经存在的“共乘”。当已经存在的共乘不存在时，再匹配出行计划形成新的“共乘”。这可以优化资源利用，减少创建过多的共乘。
- 进一步扩展规则：
6. 用户评分考虑：为每个用户设定一个评分系统，优先匹配评分较高的用户。这可以确保共乘的体验更为舒适，鼓励用户维持良好的乘车行为。
  7. 时效性：如果一个出行计划在一个特定的时间内（例如 30 分钟）都没有匹配到共乘，系统可以给用户发送提醒或考虑提供其他出行建议。
  8. 容错范围：为出发时间段提供一个容错范围，例如前后 15 分钟，以增加匹配的机会。
  9. 历史匹配记录：考虑用户的历史匹配记录。如果两个用户在过去有过良好的共乘体验，那么他们再次匹配的可能性可以增加。
  10. 特殊需求匹配：允许用户设置特殊需求，如“无吸烟”，“宠物友好”等，确保只与有相同需求的用户匹配。

这些规则和策略应该能帮助您更好地为用户提供优质的共享出行服务。当然，随着业务的发展，还可以根据实际需要不断地优化和调整这些规则。



# ▶ 用测试用例来表达业务规则

## 提示

我正在开发一个共享出行的业务。在本业务中，用户会提交自己的“出行计划”，系统会基于出行计划的匹配程度进行匹配，并创建“共乘”。

其中，出行计划包括“起始地、目的地、出发时间段、所需座位数和状态（待撮合、已加入、已取消等。仅状态为‘待撮合’的可进行匹配）”，“共乘”则包括已经匹配的“出行计划”列表。

我们已经定义了如下的业务规则：

1.出发时间段匹配：只有出发时间段一致的才可以匹配。这可以确保乘客都在相似的时间内出发。

2.状态判定：仅状态为‘待撮合’的可进行匹配。避免已经匹配或取消的出行计划再次被匹配。

所有规则的触发场景是：在已经存在（或不存在）出行计划（或共乘）的情况下，新创建了一个出行计划，则对该出行计划和已有的出行计划（或共乘）进行匹配。

请你基于上述规则和触发场景，采用实例化需求的方法，为每条规则编写测试用例。

测试用例的要求如下：

1.需要从正、反方向为每个规则设计用例，而且可能需要多个用例。

2.用正交、控制变量的方法设计用例数据，确保除了关注的变量之外，给出的用例不影响到规则的达成。例如，当你考虑时间段匹配时，可以把出行计划的对应的起始地都设成地标1，目的地都设成地标2，这样就可以消除由于距离导致的匹配问题。

3.采用 Given-When-Then 的形式输出。每个规则都需要列出出行计划和共乘的具体数据。上述要求非常重要，请在完成案例之后先自我检查，确保你的输出符合上述要求

## LLM

基于您的描述，我将为每条规则编写测试用例：

### 1.出发时间段匹配

正向用例：

Given

- 出行计划 A: 起始地-地标 1, 目的地-地标 2, 时间段-08:00-09:00, 座位数-2, 状态-待撮合
- 出行计划 B: 起始地-地标 1, 目的地-地标 2, 时间段-08:00-09:00, 座位数-2, 状态-待撮合

When

- 发布出行计划 B

Then

- 出行计划 A 和 B 匹配成为一个共乘

反向用例：

Given

- 出行计划 A: 起始地-地标 1, 目的地-地标 2, 时间段-08:00-09:00, 座位数-2, 状态-待撮合
- 出行计划 B: 起始地-地标 1, 目的地-地标 2, 时间段-09:00-10:00, 座位数-2, 状态-待撮合

When

- 发布出行计划 B

Then

- 出行计划 A 和 B 无法匹配成共乘

### 2.状态判定

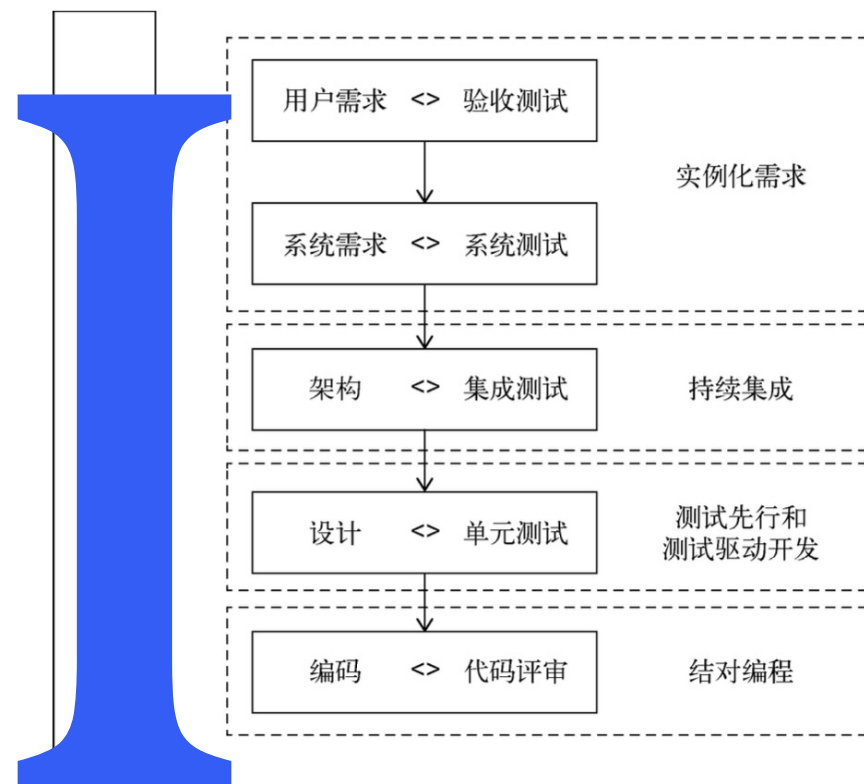
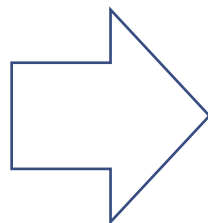
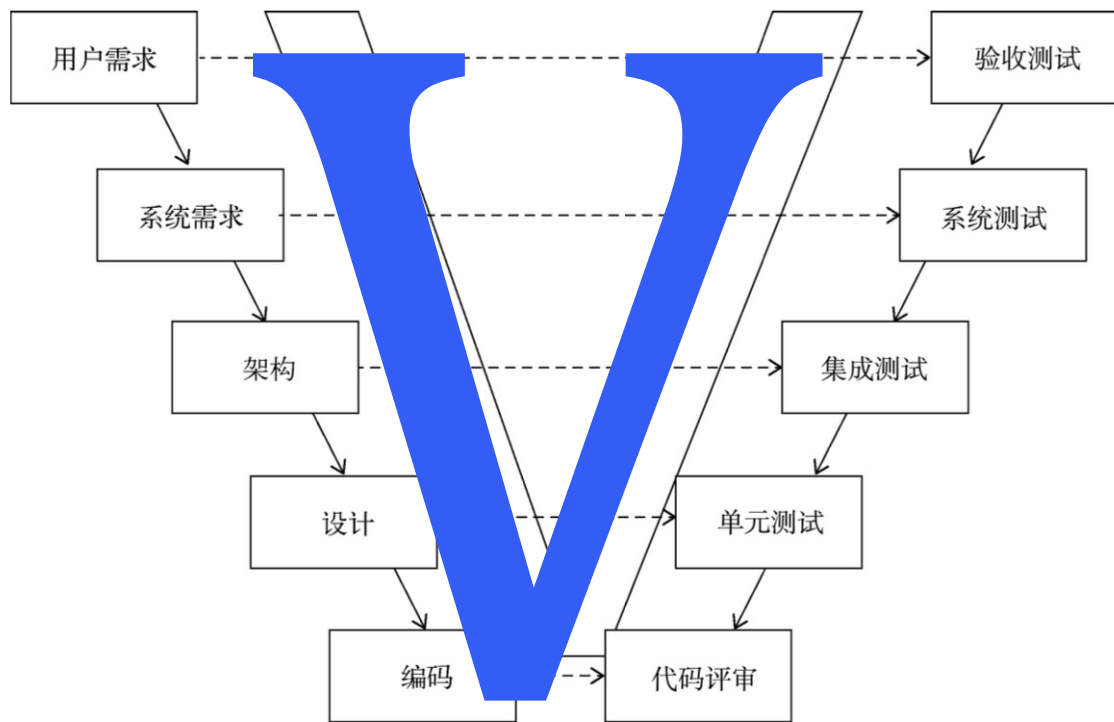
正向用例（同前）

反向用例：

Given

- 出行计划 A: 起始地-地标 1, 目的地-地标 2, 时间段-08:00-09:00, 座位数-2, 状态-已加入
- 出行计划 B: 起始地-地标 1, 目的地-地标 2, 时间段-08:00-09:00, 座位数-2, 状态-待撮合

# ► 测试先行：从V模型到I模型



**语言模型是不精确的  
但是， 契约是精确的**





**语言模型是不精确的**

**我们可以采用受限的自然语言：领域模型**

# ► 补充内容：把关键概念录入大模型，提升交互效率

记忆

×

Is developing a ride-sharing application using Domain-Driven Design (DDD) methodology, Spring Boot, and Java 17.

Ride-sharing application's domain model includes the following components:

- TripPlan**: Serves as an aggregate root and includes the following data structures:
  - PlanSpecification**: A value object containing the starting location, destination, planned departure time, and required number of seats. The 'starting location' and 'destination' are Embeddable objects that include latitude and longitude.
  - TimeSpan**: An Embeddable object representing the planned departure time, which includes the following data:
    - Start time
    - End time
  - Status**
- Cotrip**: Another aggregate root that holds a list of TripPlanIds and a status, representing the result of ride-sharing matching and the progress of the shared trip.



# ► 补充内容：把关键概念录入大模型，提升交互效率

请记住，我在开发一个共享出行的应用。我使用DDD的方法，使用Spring Boot+Java17。

记忆已更新

好的，已记住！如果在开发过程中有任何问题或需要进一步的建议，随时告诉我。

请记住，我的共享出行应用的领域模型如下：

## 出行计划 (TripPlan)

它是聚合根，数据结构如下：

\*出行信息 (PlanSpecification)

\*状态 (Status)

其中，出行信息 (PlanSpecification) 是一个值对象，包括如下数据：

\*起始地

\*目的地

\*计划出发时间

\*所需座位数

其中“起始地”和“目的地”是Embeddable类型的对象，包括如下数据：

\*经度

\*纬度

## 共乘 (cotrip)

它是另外一个聚合根。它持有TripPlanId的列表和状态，代表了共享出行撮合后的结果和共享出行的进程。

记忆已更新

已记住您的领域模型。如果在设计或实现过程中有任何疑问，随时告诉我，我会尽力提供帮助。

由 Xnip 截图

请生成TripPlan的领域层代码

以下是 TripPlan 的领域层代码，用于您的共享出行应用程序。此代码基于您的DDD框架和Spring Boot的使用，并结合Lombok进行简化：

```
java 复制代码

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.time.LocalDateTime;
import java.util.UUID;

@Getter
@Setter
@NoArgsConstructor
public class TripPlan extends AggregateRoot {

    private String tripPlanId;
    private PlanSpecification planSpecification;
    private TimeSpan timeSpan;
    private TripStatus status;

    public TripPlan(PlanSpecification planSpecification, TimeSpan timeSpan) {
        this.tripPlanId = UUID.randomUUID().toString();
        this.planSpecification = planSpecification;
        this.timeSpan = timeSpan;
        this.status = TripStatus.PENDING;
    }
}
```

# 生成自动化测试和产品代码

## 提示

我们已经有了 TripPlan 的定义。现在，我希望你以测试先行的方式，实现如下需求：

用户给出行程的起始地、目的地、计划出发时间和所需座位数，发布行程，行程发布成功后，应该触发 TripPlanCreatedEvent，数据库中可以到新发布的 TripPlan。

测试要求如下：

- 本测试的被测对象是 TripPlanService；
- 由于这个场景中写入数据库是一个关键诉求，所以请使用 SpringJUnitRunner；
- 测试完成后需要清空数据库（请使用 DirtyContext）。

输入数据应该是一个 DTO 对象，如果测试需要的对象不存在，请你直接创建它们。

## LLM

现在，我们可以编写一个 JUnit 测试用例来测试这个需求：

```
当 @RunWith(SpringRunner.class)
场景, @SpringBootTest
确保 @DirtyContext(classMode = DirtyContext.ClassMode.AFTER_EACH_TEST_METHOD)
保存 public class TripPlanServiceTest {
    首 @Autowired
    private TripPlanService tripPlanService;
    @Autowired
    private TripPlanRepository tripPlanRepository;

public @MockBean
    private ApplicationEventPublisher eventPublisher;

    }
    @Test
    // public void testCreateTripPlan() {
    //     // 创建测试数据
    @Data @NoArg @AllAr public void testCreateTripPlan() {
        // 创建测试数据
        LocationDTO departureLocation = new LocationDTO(31.240084, 121.501868);
        LocationDTO arrivalLocation = new LocationDTO(31.232862, 121.472768);
        TripPlanDTO tripPlanDTO = new TripPlanDTO(departureLocation, arrivalLocation, LocalDateTime.now().plusHours(2), 30);
        //
        // 模拟事件发布
        doNothing().when(eventPublisher).publishEvent(any(TripPlanCreatedEvent.class));
        // 调用服务方法
        TripPlan tripPlan = tripPlanService.createTripPlan(tripPlanDTO);
        // 验证数据库保存
        Optional<TripPlan> retrievedTripPlan = tripPlanRepository.findById(tripPlan.getId());
        assertTrue(retrievedTripPlan.isPresent());
        assertEquals(tripPlan.getId(), retrievedTripPlan.get().getId());
        // 验证事件发布
        verify(eventPublisher).publishEvent(any(TripPlanCreatedEvent.class));
    }
}

这个测试用例首先创建了起始和目的地的 Location 对象，并将其保存到数据库中。然后，它使用这些地点创建了一个 TripPlanDTO 对象，并调用 TripPlanService 的 createTripPlan 方法。最后，它验证了新的 TripPlan 对象已经保存到数据库中，并且 TripPlanCreatedEvent 已经发布。

注意，这里我们使用了 @MockBean 来模拟 ApplicationEventPublisher，这样我们可以轻松地验证是否触发了事件。同时，我们使用了 @DirtyContext 注解来确保测试之后的数据库清理。这样，每个测试都会在干净的环境中运行，确保测试之间没有副作用。

这个测试示例覆盖了你提到的主要需求，并提供了一个坚实的基础，可以根据你的具体需求和代码结构进行扩展和调整。
```

# ▶ 开发RESTful API

提示:

我有一个服务层接口如下: <TripPlanService 的代码, 略>

请为 TripPlanService 创建一个 REST API, 我希望路径是/trip-plan。

LLM:

下面是使用 Spring Boot 构建的基于上述服务接口的 REST API 的示例:

```
@RestController
@RequestMapping("/trip-plan")
public class TripPlanController {

    private final TripPlanService tripPlanService;

    @Autowired
    public TripPlanController(TripPlanService tripPlanService) {
        this.tripPlanService = tripPlanService;
    }

    @PostMapping
    public ResponseEntity<TripPlan> createTripPlan(@RequestBody TripPlanDTO tri
pPlanDTO, @RequestParam("user-id") String creatorId) {
        TripPlan createdTripPlan = tripPlanService.createTripPlan(tripPlanDTO,
new UserId(creatorId));
        return new ResponseEntity<>(createdTripPlan, HttpStatus.CREATED);
    }
}
```



# ▶ 开发小程序前端



# 开发小程序前端

## 提示

现在让我们继续实现 plan 组件。这个组件的要求如下：

1. 纵向包括两个区域。
2. 上方区域描述出行计划的信息。包括 3 行。
3. 第一行，是一个表示时间的图标，加一个时间段的滚动选择框（时间段数据来自 data 的，是起始 hh:mm-终止 hh:mm 的数组）表示出发时间。再加一个带有加减号的数字组件，表示乘车人数。
4. 第二行，是一个蓝色圆点小图标，加一个文字显示组件，表示起始地。
5. 第三行，是一个绿色圆点小图标，加一个文字显示组件，表示目的地。
6. 下方区域是一个红色的“确定”按钮。

```
LLM
    flex-direction: column;
    justify-content: space-around;
}
为了
(WCSS
.time-and-people, .departure, .destination {
  display: flex;
  align-items: center;
}
WXML (p
<view c
<!--
<view
  width: 20px; /* 图标的宽度 */
  height: 20px; /* 图标的高度 */
  margin-right: 8px; /* 与其他元素的间隔 */
</view>
<!--
<view
  .time-picker {
    flex-grow: 1;
  }
</view>
<!--
<view
  .number-selector {
    display: flex;
    align-items: center;
  }
  .minus, .plus {
    padding: 5px;
  }
  border: 1px solid #ccc; /* 加减号的边框 */
  user-select: none; /* 防止选中文本 */
</view>
<!--
<view
  .number {
    margin: 0 10px;
  }
</view>
</view>
.confirm-button {
  background-color: red;
  color: white;
  border: none;
  padding: 10px;
  border-radius: 5px;
}
这里
```

destination 是上方区域的三行内容。confirm-button 是下方的按钮。

```
WCSS (plan.wxss)
.plan-container {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  padding: 10px;
  background-color: #fff;
}
.plan-info {
  display: flex;
```

# ▶ 重构

## 提示

可否帮我把“带有加减号的数字选择器”封装为一个组件？

## LLM

```
count: {
  type: Number,
  value: 1
}
},
methods: {
  // 减号方法
  decrement: function() {
    if (this.data.count > 1) { // 如果人数大于 1, 则减少
      this.setData({
        count: this.data.count - 1
      });
      this.triggerEvent('change', {count: this.data.count});
    }
  },
  // 加号方法
  increment: function() {
    this.setData({
      count: this.data.count + 1
    });
    this.triggerEvent('change', {count: this.data.count});
  }
}
})
```

这个 JavaScript 文件为组件提供了增减数字的功能，并且通过自定义事件 `change` 将数字变化通知给父组件。

请注意，你需要在小程序的 `components` 目录下创建相应的文件，并在使用此组件的页面的 `.json` 配置文件中注册这个新组件。例如：

```
{
  "usingComponents": {
    "number-selector": "/components/number-selector/number-selector"
  }
}
```

这样，你就可以在任何页面上通过 `<number-selector></number-selector>` 来使用这个数字选择器组件了，并且可以监听 `change` 事件来响应数字的变化。

```
number-selector.js
Component({
  properties: {
    // 可以从外部传入的属性
```



# 设计品味非常重要

**在开发阶段**

**如果你精通设计，大模型可承担90%以上的工作**

# ▶ 编写部署资源文件

## 提示

请为我的一个 Spring boot 项目创建一个 Dockerfile。我的项目的信息如下：

1. 基于 spring boot3
2. 运行在 Java17
3. 项目名称叫 cotrip

你可以参考如下的 maven 信息：

```
<groupId>demo.leansd.cn</groupId>
<artifactId>cotrip</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

## LLM

基于您提供的信息，以下是为您的 Spring Boot 项目创建的 Dockerfile：

```
# 使用官方的 Java 17 基础镜像
FROM openjdk:17-jdk-slim

# 设置工作目录
WORKDIR /app

# 将 Maven 构建的 jar 文件复制到容器中
COPY target/cotrip-0.0.1-SNAPSHOT.jar /app/cotrip.jar

# 暴露 8080 端口
EXPOSE 8080

# 设置容器启动时的命令
CMD ["java", "-jar", "cotrip.jar"]
```

请按照以下步骤使用此 Dockerfile：

1. 确保您已经在项目根目录下运行了 `mvn clean package`，这样 **Maven** 会在 `target` 目录下生成一个名为 `cotrip-0.0.1-SNAPSHOT.jar` 的可执行 `jar` 文件。
2. 将上述 `Dockerfile` 内容保存到项目的根目录，并命名为 `Dockerfile`。
3. 在项目的根目录下运行以下命令，构建 **Docker** 镜像：

```
docker build -t demo.leansd.cn/cotrip:0.0.1-SNAPSHOT .
```

4. 运行容器：

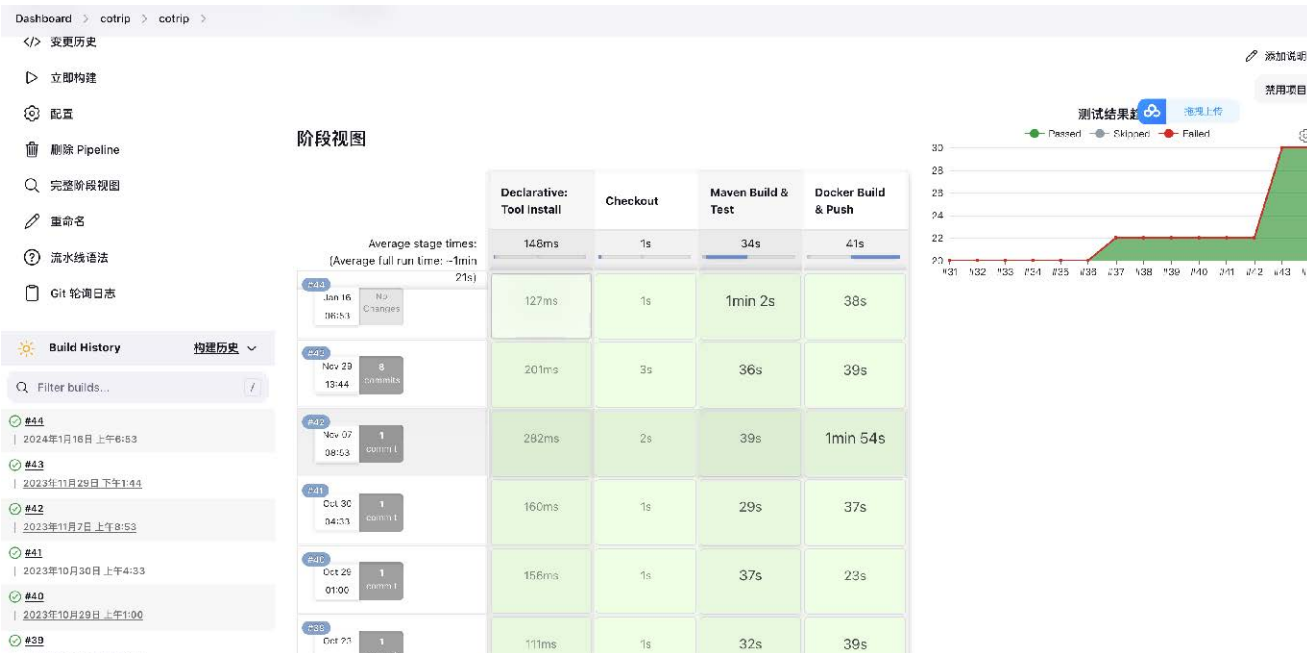
```
docker run -p 8080:8080 demo.leansd.cn/cotrip:0.0.1-SNAPSHOT
```

现在，您的 **Spring Boot** 应用应该在 **Docker** 容器中运行，并监听 `8080` 端口。您可以通过浏览器或其他工具访问它。

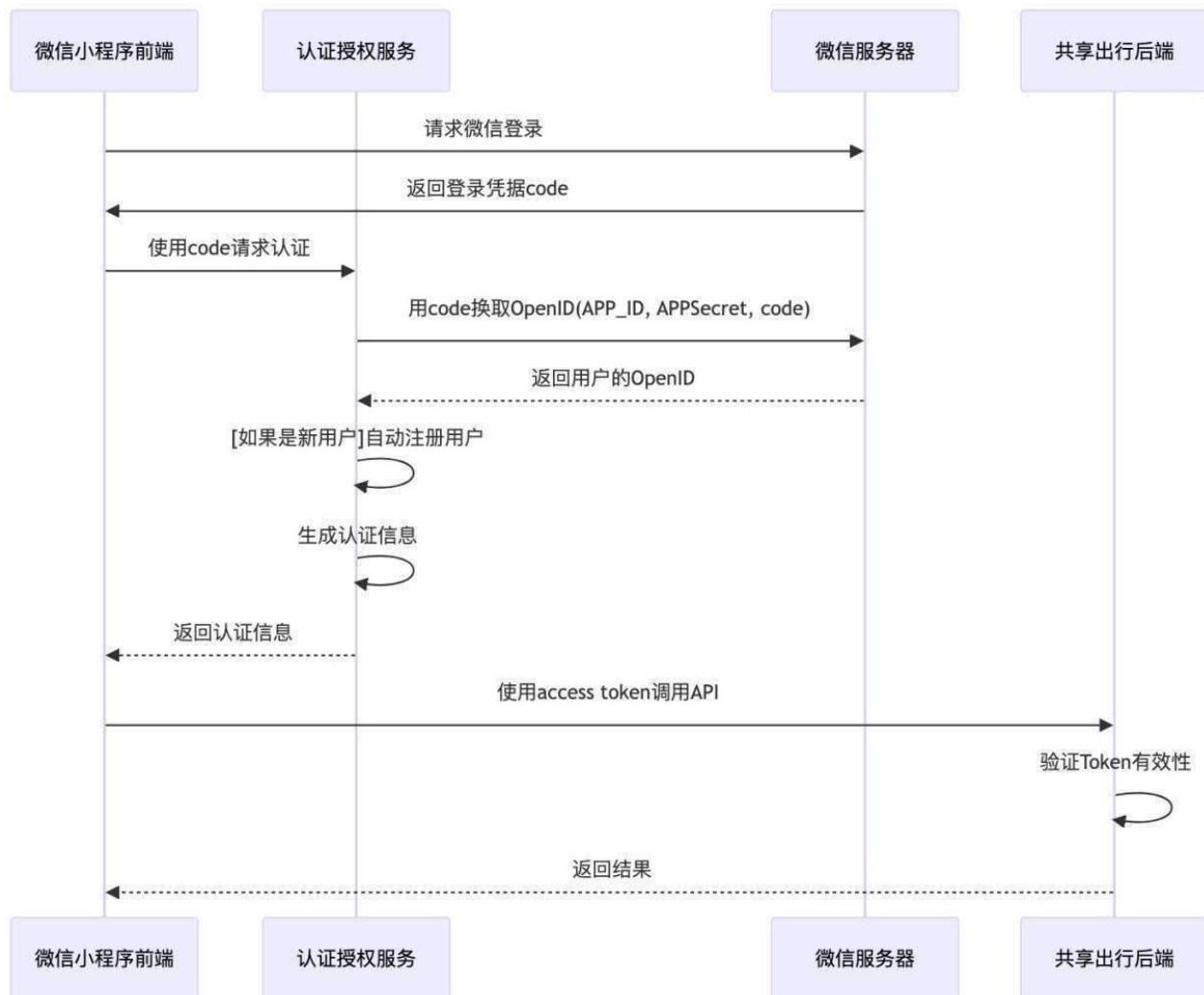


# 设计持续集成流水线

```
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = 'registry.cn-hangzhou.aliyuncs.com/leands/auth:latest'
    KUBECONFIG_CREDENTIAL_ID = 'kubeconf_server'
  }
  stages {
    stage('Checkout') {
      steps {
        checkout([
          $class: 'GitSCM',
          branches: [[name: 'main']],
          doGenerateSubmoduleConfigurations: false,
          extensions: [],
          submoduleCfg: [],
          userRemoteConfigs: [[
            url: 'https://gitee.com/leands/auth.git'
          ]]
        ])
      }
    }
    stage('Unit Test') {
      steps {
        sh 'npm install'
        sh 'npm run test'
        junit '**/test-results/jest/*.xml'
      }
    }
    stage('Docker Build & Push') {
      when {
        expression { currentBuild.resultIsBetterOrEqualTo('SUCCESS') }
      }
      steps {
        script {
          docker.build("${DOCKER_IMAGE}")
          docker.withRegistry('https://registry.cn-hangzhou.aliyuncs.com/eme
rgentdesign/leands', 'aliyun-docker-registry-credentials') {
            docker.image("${DOCKER_IMAGE}").push()
          }
        }
      }
    }
  }
}
```



# 实现用户登录



# 实现用户登录

## 提示

我们将使用 Node.js+express 实现一个用于微信小程序注册的后端服务。这个后端服务将依赖 Keycloak 来完成用户管理和 token 管理等任务。如果是新用户（openid 在 Keycloak 中不存在），需要自动在 Keycloak 中创建一个新用户。请基于这个策略，编写 token 生成和用户管理的服务端代码。

## LLM

如果您打算使用 Keycloak 来管理用户并为微信小程序生成令牌，您需要执行以下步骤：

1. 从微信小程序获取 code。

```
const KeycloakAdmin = require('keycloak-admin');

const clientId = 'admin-cli';

let users = await KeycloakAdmin.users.find({ search: openid });

if (users.length === 0) {
  // 创建新用户
  await KeycloakAdmin.users.create({
    username: openid,
    enabled: true,
    credentials: [{ type: 'password', value: 'a-strong-password' }] // 为用户设置一个密码或其他认证方式
  });
}

// 从 Keycloak 获取 token (这可能需要你根据自己的 Keycloak 设置进行调整)
const tokenResponse = await axios.post(`${KeycloakConfig.baseUrl}/realms/${KeycloakConfig.realmName}/protocol/openid-connect/token`, {
  username: openid,
  password: 'a-strong-password',
  client_id: 'your-client-id', // 你的客户端 ID
  client_secret: 'your-client-secret', // 你的客户端密钥
  grant_type: 'password'
});

const token = tokenResponse.data.access_token;
res.json({ token });
} catch (error) {
  console.error('错误: ', error);
  res.status(500).send('登录失败');
}
});
app.listen(3000, () => {
  console.log('Server started on http://localhost:3000');
});

const response = await axios.get(`https://api.weixin.qq.com/sns/jscode2session?appid=${APP_ID}&secret=${APP_SECRET}&js_code=${code}&grant_type=authorization_code`);
const openid = response.data.openid;

// 使用 openid 检查用户是否存在
await KeycloakAdmin.auth({
  username: 'admin-username', // 替换为你的 Keycloak 管理员账户
  password: 'admin-password', // 替换为你的 Keycloak 管理员密码
  grantType: 'password',
```



**在通用领域**

**大模型不仅可承担大量的工作**

**还可有效弥补开发者在通用领域的认知缺失**

# ▶ “共享出行” 案例研究小结

定义产品方案及路线图

初始迭代

初始架构

部署方案

持续集成

核心域

认证域

微信小程序

迭代增量1,2

增加上车点管理

支持顺风车业务

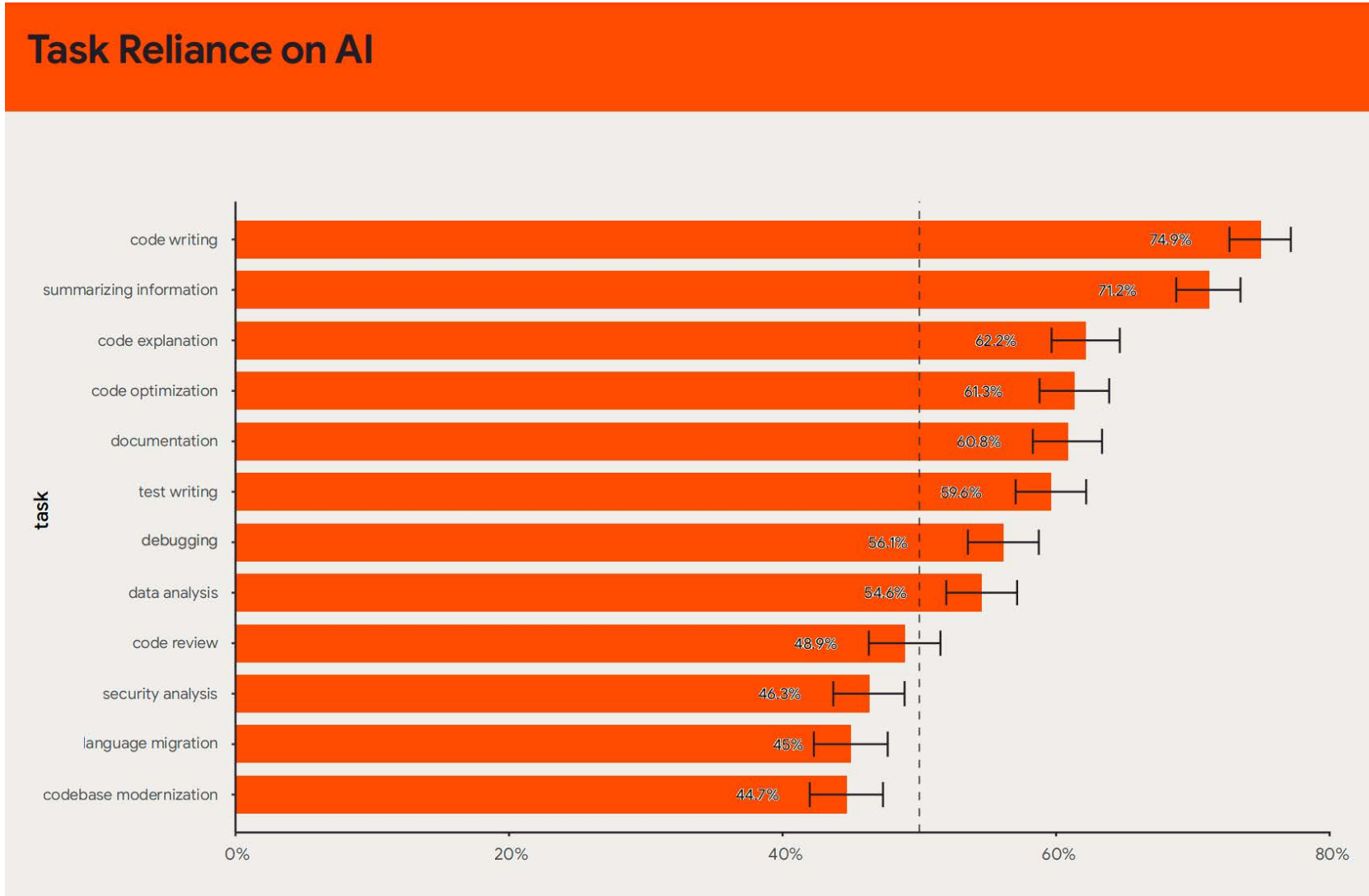
## ► 共享出行案例小结

表 1.1 可使用大模型辅助的部分开发活动

开发环节	可使用大模型辅助的部分开发活动
需求分析	产品方案设计、需求场景和业务规则分析、需求澄清、需求评审等
架构设计	架构分解、架构策略定义、架构选型等
编 码	生成整块代码、代码智能补全、代码评审、生成代码注释、按要求重构代码等
测 试	生成测试用例、编写自动化测试等
问题定位和解决	日志分析、错误信息诊断、修复建议等



# 对比一下DORA2024的开发者现状调查



## ▶ 值得思考的

**为什么我一直想做，LLM大爆发之前为什么没能做出来？**

**为什么是我更容易做出来，而不是非专业开发者更容易做出来？**

# ▶ 专业能力的重要性，不降反增

L leansd 已关注 | 1

🏠 概览 📦 仓库 6 📄 Issues 3 🔗 Pull Requests 📡 动态 👤 成员 1 ⚙️ 设置 + 新建仓库

快速搜索...

状态: 正常 ▾ 类型: 全部 ▾ 语言: 全部 ▾ 排序: 最近更新 ▾

仓库语言

- cotrip-miniapp** 👁️ 1 🐛 0  
最近更新: 16分钟前 [设置](#)
- cotrip-deployment** 👁️ 1 🐛 0  
The deployment repository of cotrip demo project  
最近更新: 17分钟前 [设置](#)
- keycloak** 👁️ 1 🐛 0  
最近更新: 1天前 [设置](#)
- auth** 👁️ 1 🐛 0  
最近更新: 5天前 [设置](#)
- cotrip** </> Java 👁️ 1 🐛 0  
最近更新: 5天前 [设置](#)

TypeScript	🟢	26%
JavaScript	🟢	17%
Dockerfile	🟢	17%
Shell	🟢	17%
Java	🟢	1%
CS	🟢	1%
HT	🟢	1%





# PART 03

在大模型时代，哪些专业能力  
更值得关注？

# ▶ 软件开发究竟是在做什么？

**软件开发的本质不是编码，而是探索 and 发现**

当然，编码并非不重要，而是更重要的价值

# ► 为什么软件开发中的探索和发现如此重要？

需求

业务目标是什么？

首先从哪个业务开始探索？

用小程序还是独立APP？

一次共乘是2单更好还是多单更好？

如何判断两个人是“最匹配”？

设计&实现

如何规划子领域？

API如何设计？

采用什么技术栈？

如何保证各个粒度的可测性？

哪些自己做？ 哪些使用第三方服务？

如何分解模块和职责？

发布和运维

如何制定发布策略？

如何保证安全性？

如何规划资源和制定伸缩策略？

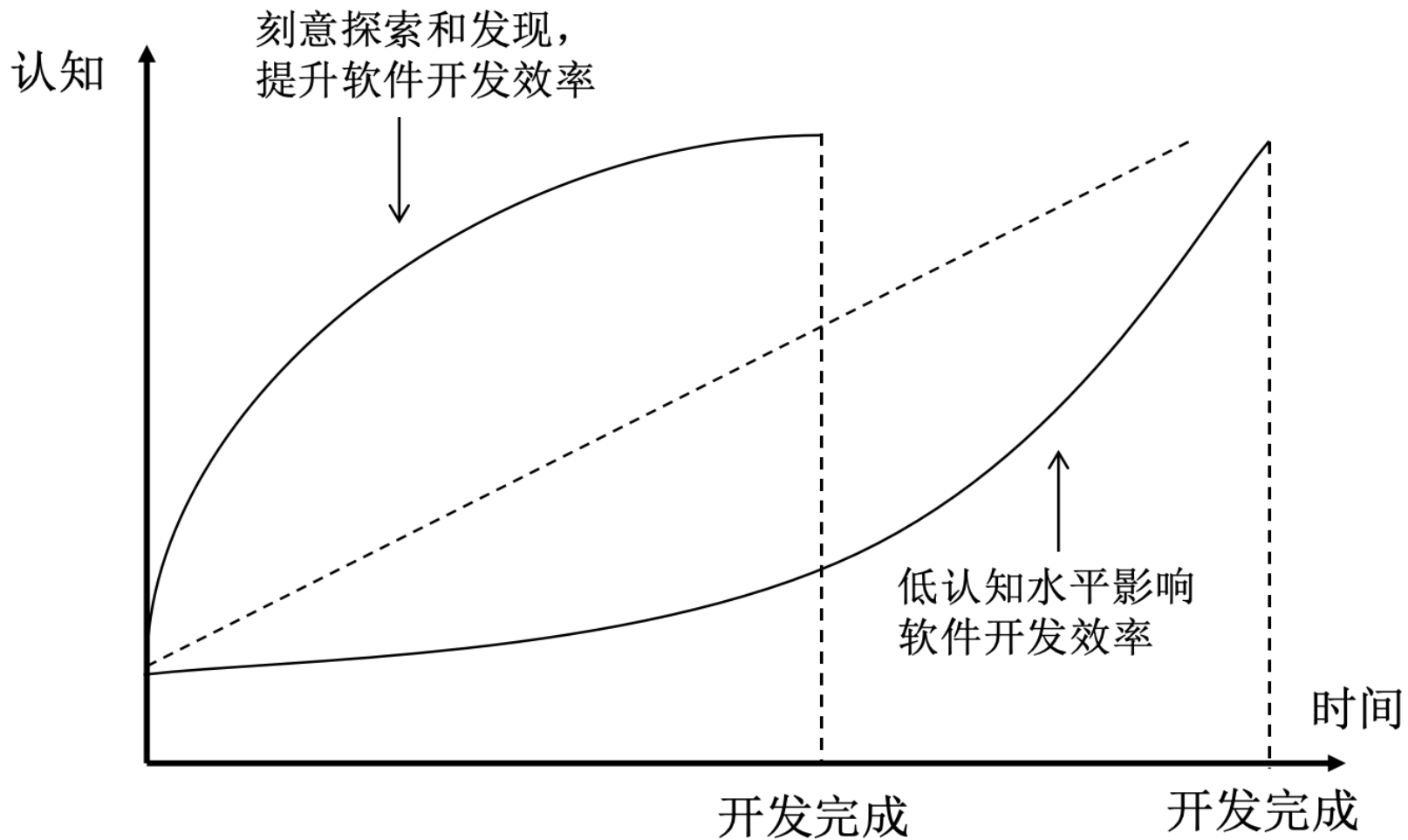
选择什么系统  
完成持续集成和发布？

如何管理配置数据？

验证



# ► 软件开发是探索和发现的过程



要解决的问题是什么？

采用什么解决方案？

实际把方案建造出来

# ▶ 弄明白该做什么，是人类的责任，而不是大模型的责任

除了你，没有其他人知道：

为什么要做这个系统，系统应该被做成什么样子

# ▶ 大模型可以做方案设计，前提是你能把问题交代清楚

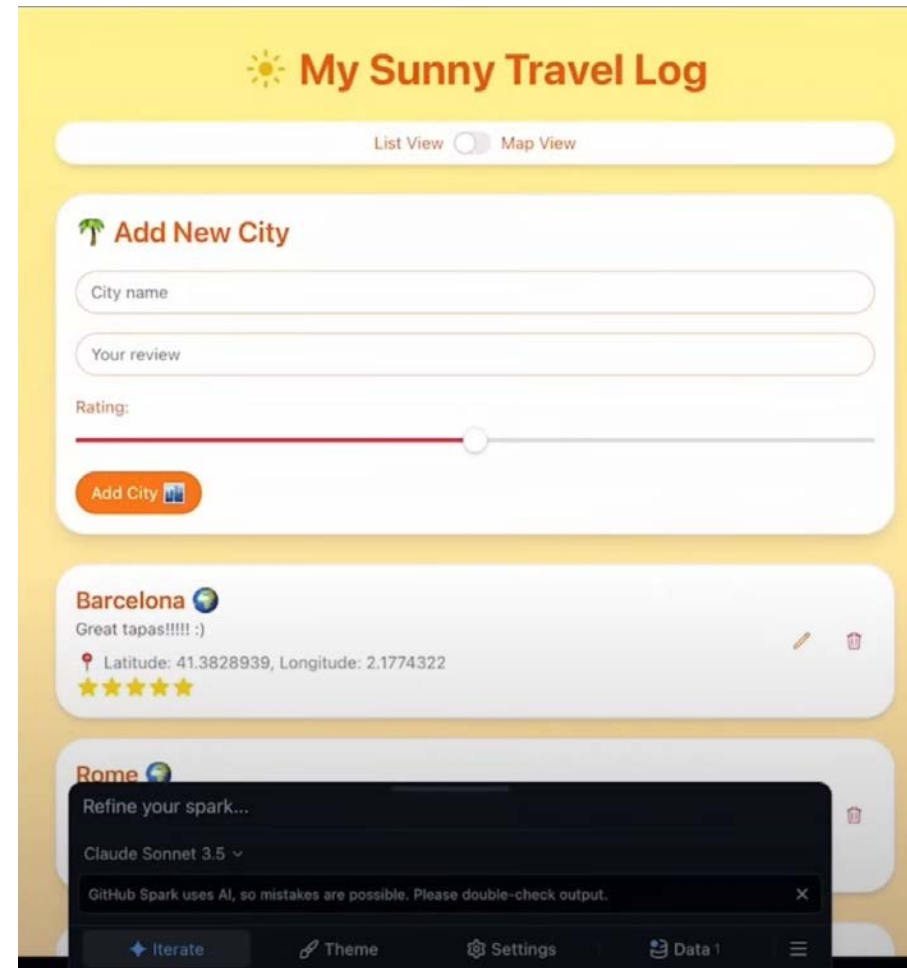
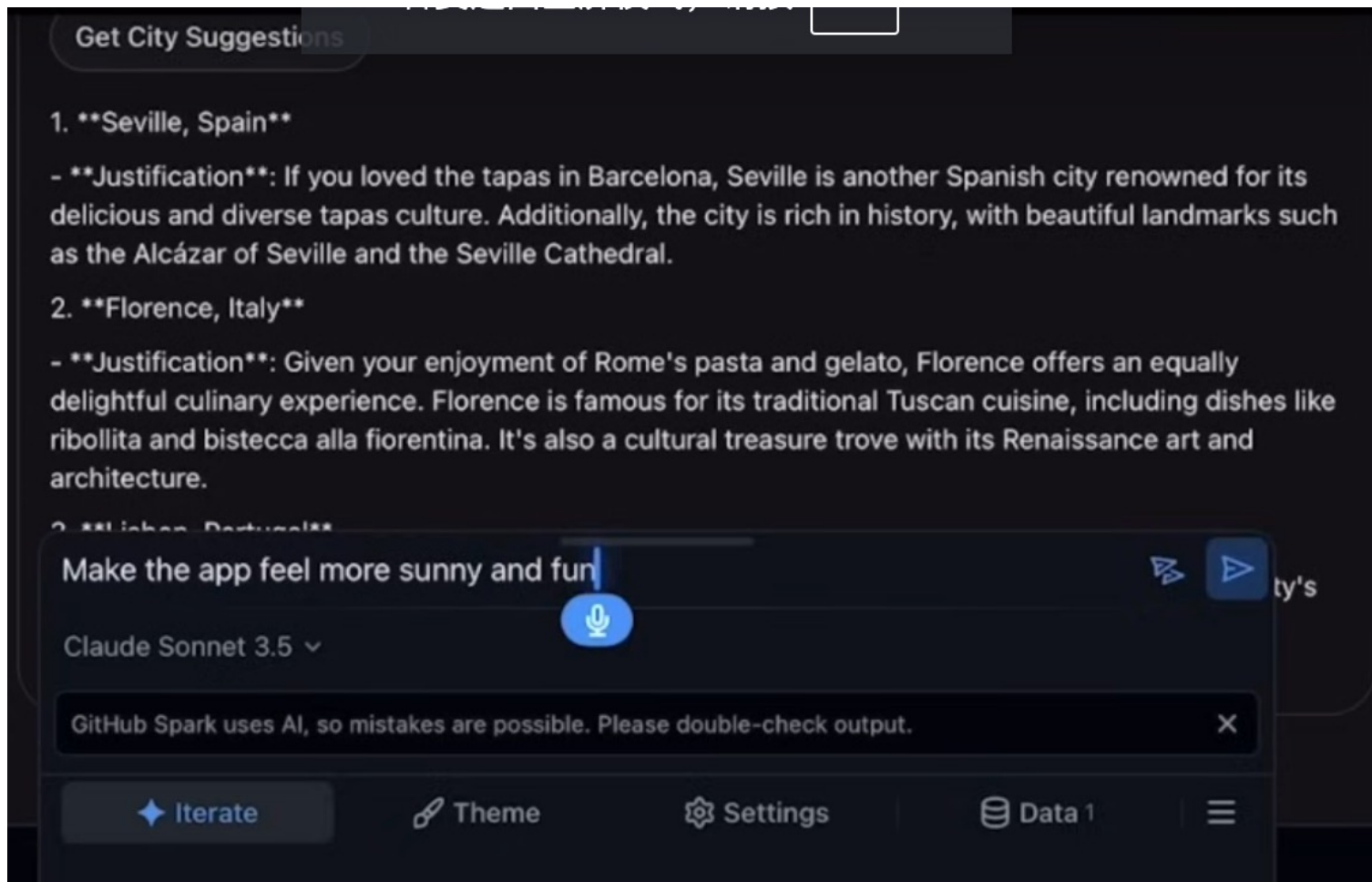
不是因为大模型的能力，而是把事情说清楚的能力。

即使大模型的上下文已经非常长，你也很难保证需求的完备性和上下文交代的清晰程度，而且互联网上本来就充斥了彼此冲突的信息

把开放性的大问题拆分为封闭性的小问题



# ▶ 让我们顺便关注一下最新的Github spark



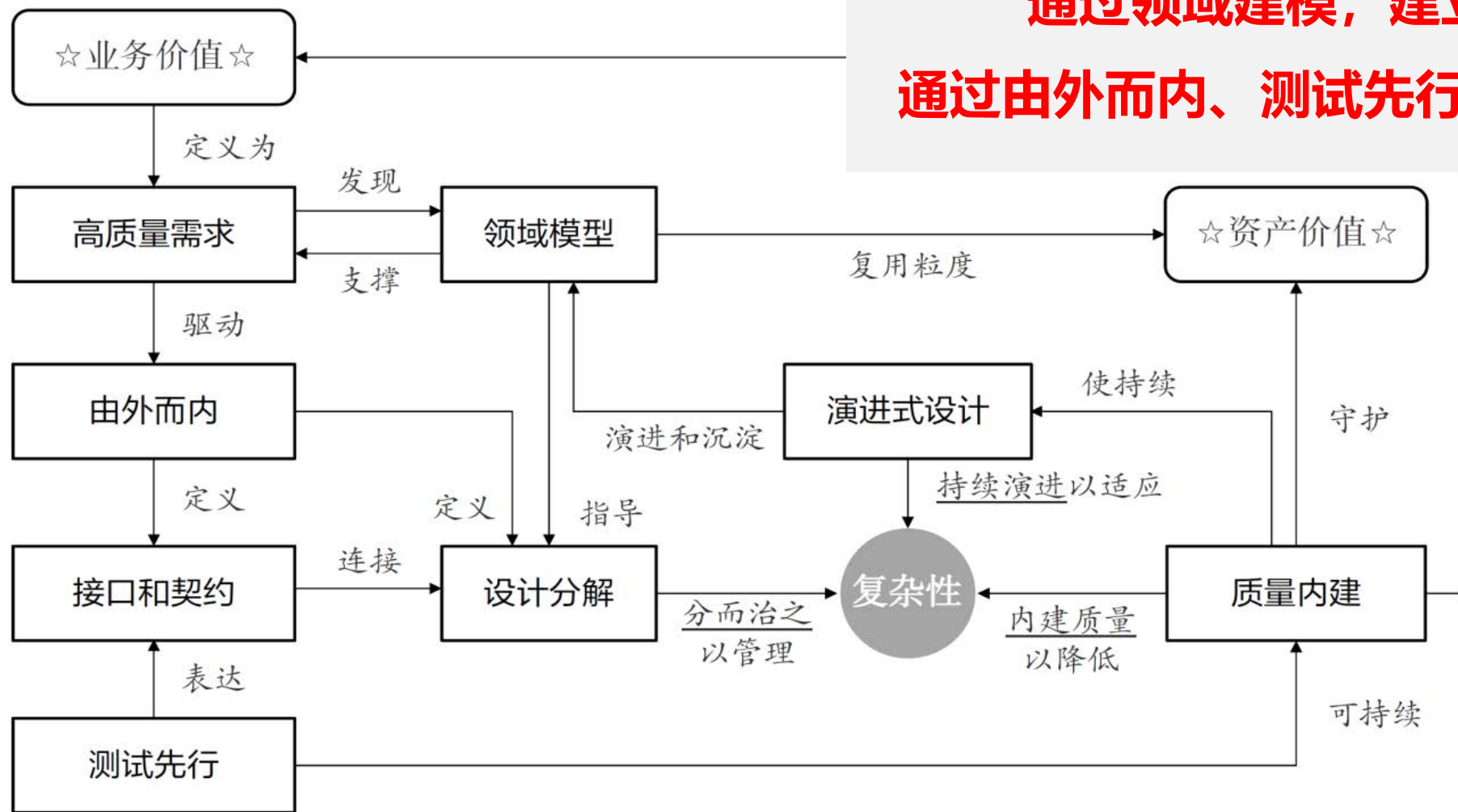
<https://githubnext.com/projects/github-spark>

# 精益软件设计方法框架

通过演进式设计，约束问题解决的范围

通过领域建模，建立概念共识

通过由外而内、测试先行，提升表达质量



# ▶ 演进式设计

解决一个大问题很  
难,

但是,

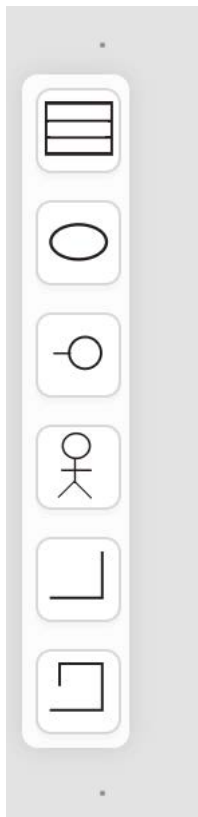
分开解决很多小问题

很容易

fix: event publish/save sequence; simplify aggregate; improve: event listener of trip-plan-created-event	gangz	2023/8/22, 20:34
add @transactional based event publisher	gangz	2023/8/22, 17:37
add exception handler for trip plan not found / driven GlobalExceptionHandler controller advice	gangz	2023/8/22, 15:53
add verify for cotrip matching by http interface	gangz	2023/8/22, 13:13
update trip plan's cotrip id after matched	gangz	2023/8/22, 12:31
add session extraction from header in dev env and add integration spring test	gangz	2023/8/22, 11:03
add more test for cotrip matching	gangz	2023/8/22, 10:14
fix design: make cotrip matching depends on tripPlanService	gangz	2023/8/22, 09:51
demo: add a unit test try to show existing bug in system/ and discussion on design strategy	gangz	2023/8/22, 09:36
add rest controller and integrate with mysql db	gangz	2023/8/22, 08:51
refactory: package definition	gangz	2023/8/21, 12:33
refactory: package structures: and add self-defined HaversineGeoService	gangz	2023/8/21, 12:28
add matching rule for departure time and start point	gangz	2023/8/21, 11:49
add matching rule for seat limitation	gangz	2023/8/20, 11:01
refactory: DTO really necessary? remove PlanSpecificationDTO, TimeSpanDTO and LocationDTO	gangz	2023/8/20, 10:08
adjust cotrip matching service contract: depends on tripplan created event instead depends on query o	gangz	2023/8/20, 00:11
refactory: make CoTripMatchingServiceTest from spring unit test to normal junit test	gangz	2023/8/18, 09:12
refactory cotrip/trip plan test and add first cotrip SBE example	gangz	2023/8/17, 19:23
create cotrip matching service with 1st test case	gangz	2023/8/15, 18:05
add trip plan status	gangz	2023/8/15, 11:09
can get session data at rest controller	gangz	2023/8/9, 08:43
add required seat to trip plan and rename routespecification	gangz	2023/8/8, 23:27
add equals/hashcode to domain event and generic id	gangz	2023/8/8, 22:34
add DDD base classes; adjust TripPlan structure: add routeSpecification and UserId	gangz	2023/8/8, 22:12
can create trip plan	gangz	2023/8/8, 16:07
split dev and integration configuration, and disable security check in dev env	gangz	2023/8/8, 10:12
enable spring security with oauth2	gangz	2023/7/30, 23:02
add spring jpa and enable database access	gangz	2023/7/29, 17:26
a blank project with hello world restful api	gangz	2023/7/29, 16:32
Initial commit	gangz*	2023/7/29, 10:34



# ▶ 示例：通过规划演进，降低大模型理解问题和人类提问的难度



```
const AssoIcon: React.FC<AssoIconProps> = ({
  size = 20,
  color = '#ffff00', // 默认填充颜色
  strokeColor = '#241c1c', // 默认描边颜色
  ...props
}) => (
  <svg
    xmlns="http://www.w3.org/2000/svg"
    width={size}
    height={size}
    viewBox="0 0 48 48"
    {...props}
  >
    <g id="layer1">
      <path
        d="M 5.6949153,37.423729 H 39.254237 V 7.7288136"
        style={{
          fill: 'none',
          fillOpacity: 1,
          stroke: strokeColor,
          strokeWidth: 1.69,
          strokeLinecap: 'butt',
          strokeLinejoin: 'miter',
          strokeMiterlimit: 4,
          strokeDasharray: 'none',
          strokeDashoffset: 0,
          strokeOpacity: 1,
        }}
      />
    </g>
  </svg>
);
```

```
const SelfAssoIcon: React.FC<SelfAssoIconProps> = ({
  size = 20, // 默认宽度和高度为20
  color = '#ffff00', // 默认填充颜色为#ffff00
  strokeColor = '#241c1c', // 默认描边颜色为#241c1c
  ...props // 允许传递其他 SVG 属性
}) => (
  <svg
    xmlns="http://www.w3.org/2000/svg"
    width={size}
    height={size}
    viewBox="0 0 48 48"
    {...props}
  >
    <g id="layer1">
      <path
        d="M 5.6949153,37.423729 H 39.254237 V 7.7288136 H 12 V 22"
        style={{
          fill: 'none',
          fillOpacity: 1,
          stroke: strokeColor,
          strokeWidth: 1.69,
          strokeLinecap: 'butt',
          strokeLinejoin: 'miter',
          strokeMiterlimit: 4,
          strokeDasharray: 'none',
          strokeDashoffset: 0,
          strokeOpacity: 1,
        }}
      />
    </g>
  </svg>
);

export default SelfAssoIcon;
```

# 失败的尝试



我有一个如下的React组件，它是一个图标。我希望能把其中的绘图元素提取出来作为元素，例如circle，path等，这样我就可以用样式和配置，生成不同的图标，请指导我如何做。下面是我的：

```
const ActorIcon = ({
  size = 16, // 控制 SVG 的宽度和高度
  color = '#ffffff', // 控制椭圆和路径的填充颜色
  strokeColor = '#241c1c', // 控制椭圆和路径的描边颜色
  ...props // 允许传递其他属性，如 className、onClick 等
}) => (
  <svg
    xmlns="http://www.w3.org/2000/svg"
    width={size}
    height={size}
    viewBox="0 0 48 48"
    {...props}
  >
```

和我想的不一样



你可以将 `ellipse` 和 `path` 元素提取成独立的可重用组件，并在主组件中传入必要的样式和配置。这可以帮助你使用同样的基础元素生成不同的图标。

下面是一个示例，展示如何将 `circle` 和 `path` 提取为独立组件，并通过 `props` 来传递样式：

## 1. 提取 Shape 元素为独立组件

```
jsx
复制代码

// Ellipse Component
const Ellipse = ({ cx, cy, rx, ry, fill, stroke, strokeWidth }) => (
  <ellipse
    cx={cx}
    cy={cy}
    rx={rx}
    ry={ry}
    style={{
      fill,
      stroke,
      strokeWidth,
      strokeLinecap: 'butt',
      strokeLinejoin: 'miter',
      strokeMiterlimit: 4,
      strokeDasharray: 'none',
    }}
  >
```

# 一步步来

由 Xnip 截图

我没有说明白，让我们从一个稍微简单的例子开始。我有如下的组件：

```
interface SelfAssoIconProps extends
React.SVGProps<SVGSVGElement> {
  size?: number; // 控制 SVG 的宽度和高度
  color?: string; // 控制便笺纸主体的填充颜色
  strokeColor?: string; // 控制便笺纸的描边颜色
}
```

```
const SelfAssolcon: React.FC<SelfAssolconProps> = ({
  size = 20, // 默认宽度和高度为20
  color = '#ffff00', // 默认填充颜色为#ffff00
  strokeColor = '#241c1c', // 默认描边颜色为#241c1c
  ...props // 允许传递其他 SVG 属性
}) => {
  <svg
    xmlns="http://www.w3.org/2000/svg"
    width={size}
    height={size}
    viewBox="0 0 48 48"
    {...props}
  >
    <g id="layer1">
      <path
        d="M 5.6949153,37.423729 H 39.254237 V 7.7288136 H 12 V
22"
        style={{
          fill: 'none',
          fillOpacity: 1,
          stroke: strokeColor,
          strokeWidth: 1.69,
          strokeLinecap: 'butt',
```

```
strokeLinejoin: 'miter',
strokeMiterlimit: 4,
strokeDasharray: 'none',
strokeDashoffset: 0,
strokeOpacity: 1,
}}
/>
</g>
</svg>
);
```

export default SelfAssolcon;，我还有另外一个组件： import React from 'react';

```
interface AssolconProps extends
React.SVGProps<SVGSVGElement> {
  size?: number; // 控制 SVG 的宽度和高度
  color?: string; // 控制便笺纸主体的填充颜色
  strokeColor?: string; // 控制便笺纸的描边颜色
}
```

```
const Assolcon: React.FC<AssolconProps> = ({
  size = 20, // 默认宽度和高度为20
  color = '#ffff00', // 默认填充颜色为#ffff00
  strokeColor = '#241c1c', // 默认描边颜色为#241c1c
  ...props // 允许传递其他 SVG 属性
}) => {
  <svg
    xmlns="http://www.w3.org/2000/svg"
    width={size}
    height={size}
    viewBox="0 0 48 48"
    {...props}
  >
```

```
<g id="layer1">
  <path
    d="M 5.6949153,37.423729 H 39.254237 V 7.7288136"
    style={{
      fill: 'none',
      fillOpacity: 1,
      stroke: strokeColor,
      strokeWidth: 1.69,
      strokeLinecap: 'butt',
      strokeLinejoin: 'miter',
      strokeMiterlimit: 4,
      strokeDasharray: 'none',
      strokeDashoffset: 0,
      strokeOpacity: 1,
    }}
  />
</g>
</svg>
);
```

export default Assolcon; 你可以看到，它们除了路径之外，其他都是一样的。我如何定义一个通用组件，只是从外部传入不同的Path数据，就可以创建两个特定的组件？



# 一步步来

你可以创建一个通用的 `Icon` 组件，它接受 `pathData` 作为属性，然后在这个通用组件中渲染传入的路径。这样，你可以根据需要创建不同的图标，只需传入不同的路径数据。

以下是如何实现这个通用组件的步骤：

## 1. 创建通用的 `Icon` 组件

```
jsx 复制代码  
  
import React from 'react';  
  
interface IconProps extends React.SVGProps<SVGSVGElement> {  
  size?: number; // 控制 SVG 的宽度和高度  
  color?: string; // 控制便笺纸主体的填充颜色  
  strokeColor?: string; // 控制便笺纸的描边颜色  
  pathData: string; // 传入的路径数据  
}  
  
const Icon: React.FC<IconProps> = ({  
  size = 20,  
  color = '#ffff00',
```

# 一步步来

很好，现在请扩展这个组件，让它可以接受一个Path数组

要扩展这个组件以接受一个 `pathData` 数组，从而能够绘制多个路径，可以将 `pathData` 改为数组并在 `Icon` 组件中遍历这个数组，动态渲染多个 `<path>` 元素。

## 1. 修改通用的 `Icon` 组件来支持 `pathData` 数组

我们将 `pathData` 改为一个数组，并使用 `map()` 函数遍历数组来渲染多个路径。

jsx

复制代码

```
import React from 'react';

interface IconProps extends React.SVGProps<SVGSVGElement> {
  size?: number; // 控制 SVG 的宽度和高度
  color?: string; // 控制便笺纸主体的填充颜色
  strokeColor?: string; // 控制便笺纸的描边颜色
  pathData: string[]; // 传入的路径数据数组
}
```

```
const Icon: React.FC<IconProps> = ({
  size = 20,
  color = '#ffff00',
  strokeColor = '#2196f3',
```

```
<g id="layer1">
  {pathData.map((d, index) => (
    <path
      key={index}
      d={d}
      style={{
        fill: 'none',
        fillOpacity: 1,
        stroke: strokeColor,
        strokeWidth: 1.69,
        strokeLinecap: 'butt',
        strokeLinejoin: 'miter',
        strokeMiterlimit: 4,
        strokeDasharray: 'none',
        strokeDashoffset: 0,
        strokeOpacity: 1,
      }}
    />
  ))}
</g>
```

# 一步步来

由 Xnip 截图

好，现在请参照Path数组的情况，生成一个组件，让它不仅仅可以接受Path，还可以接受Circle等组件。我猜测你应该在每个数据前面加入一个类型标识来解决这个问题。

你猜测得对，为了让组件不仅接受 `path`，还可以接受其他 SVG 元素如 `circle`、`rect` 等，我们可以在传递数据时，使用一个类型标识符来告诉组件应该渲染什么类型的元素。然后根据这个标识符选择相应的 SVG 元素进行渲染。

我们可以将输入的数据设计为一个对象数组，每个对象包含一个 `type` 字段来标识元素类型，另外的字段则根据类型不同来传递对应的属性。

## 1. 修改通用的 `Icon` 组件来支持多种 SVG 元素

我们将 `pathData` 改为一个通用的 `svgElements` 数组，使用 `map()` 方法根据元素的 `type` 来动态渲染不同的 SVG 元素。

```
tsx 复制代码

import React from 'react';

// 定义每个 SVG 元素对象的类型
interface SVGElementProps {
  type: 'path' | 'circle' | 'rect'; // 可以扩展更多类型
  props: React.SVGProps<SVGPathElement | SVGCircleElement | SVGRectElement>; // 各
}

interface IconProps extends React.SVGProps<SVGSVGElement> {
  size?: number; // 控制 SVG 的宽度和高度
  color?: string; // 控制便笺纸主体的填充颜色
  strokeColor?: string; // 控制便笺纸的描边颜色
  svgElements: SVGElementProps[]; // SVG 元素的数组
}
```

```
<g id="layer1">
  {svgElements.map((element, index) => {
    const { type, props: elementProps } = element;
    switch (type) {
      case 'path':
        return <path key={index} {...elementProps} />;
      case 'circle':
        return <circle key={index} {...elementProps} />;
      case 'rect':
        return <rect key={index} {...elementProps} />;
      // 可以根据需要扩展其他 SVG 元素
      default:
        return null;
    }
  })}
</g>
```



# ▶ 用领域模型支持概念表达

## 提示

我使用 Spring Boot、JPA、Lombok、基于 DDD 的四层架构开发应用。我使用的 Java 的版本是 Java 17，JPA 相关的定义请使用 jakarta.persistence。现在我有一个需求，请帮助我编写领域层代码。

这是一个共享出行领域的应用。现在需要实现的是“出行计划 (TripPlan)”，它是聚合根，数据结构如下：

- 出行信息 (PlanSpecification)
- 状态 (Status)

其中，出行信息 (PlanSpecification) 是一个值对象，包括如下数据：

- 起始地
- 目的地
- 计划出发时间
- 所需座位数<sup>1</sup>

“起始地”和“目的地”是 Embeddable 类型的对象，包括如下数据：

- 经度
- 纬度

“计划出发时间” (TimeSpan) 是 Embeddable 类型的对象，包括如下数据：

- 开始时间
- 结束时间

## 提示

我正在开发一个共享出行的业务。在本业务中，用户会提交自己的“出行计划”，系统会基于出行计划的匹配程度进行匹配，并创建“共乘”。

其中，出行计划包括“起始地、目的地、出发时间段、所需座位数和状态（待撮合、已加入、已取消等。仅状态为‘待撮合’的可进行匹配）”，“共乘”则包括已经匹配的“出行计划”列表。

我们已经定义了如下的业务规则：

1. 出发时间段匹配：只有出发时间段一致的才可以匹配。这可以确保乘客都在相似的时间内出发。

2. 状态判定：仅状态为‘待撮合’的可进行匹配。避免已经匹配或取消的出行计划再次被匹配。

所有规则的触发场景是：在已经存在（或不存在）出行计划（或共乘）的情况下，新创建了一个出行计划，则对该出行计划和已有的出行计划（或共乘）进行匹配。

请你基于上述规则和触发场景，采用实例化需求的方法，为每条规则编写测试用例。

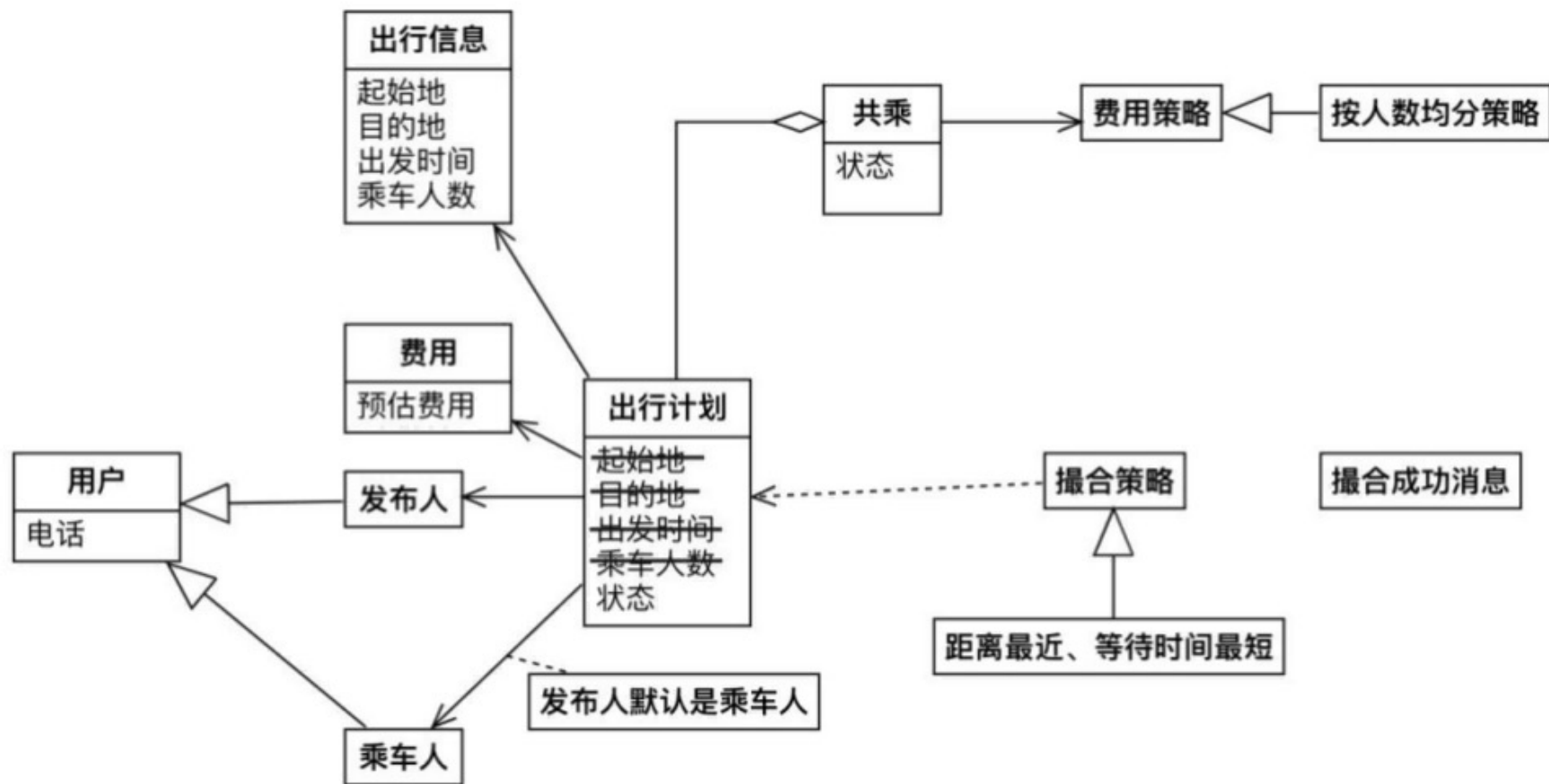
测试用例的要求如下：

1. 需要从正、反方向为每个规则设计用例，而且可能需要多个用例。

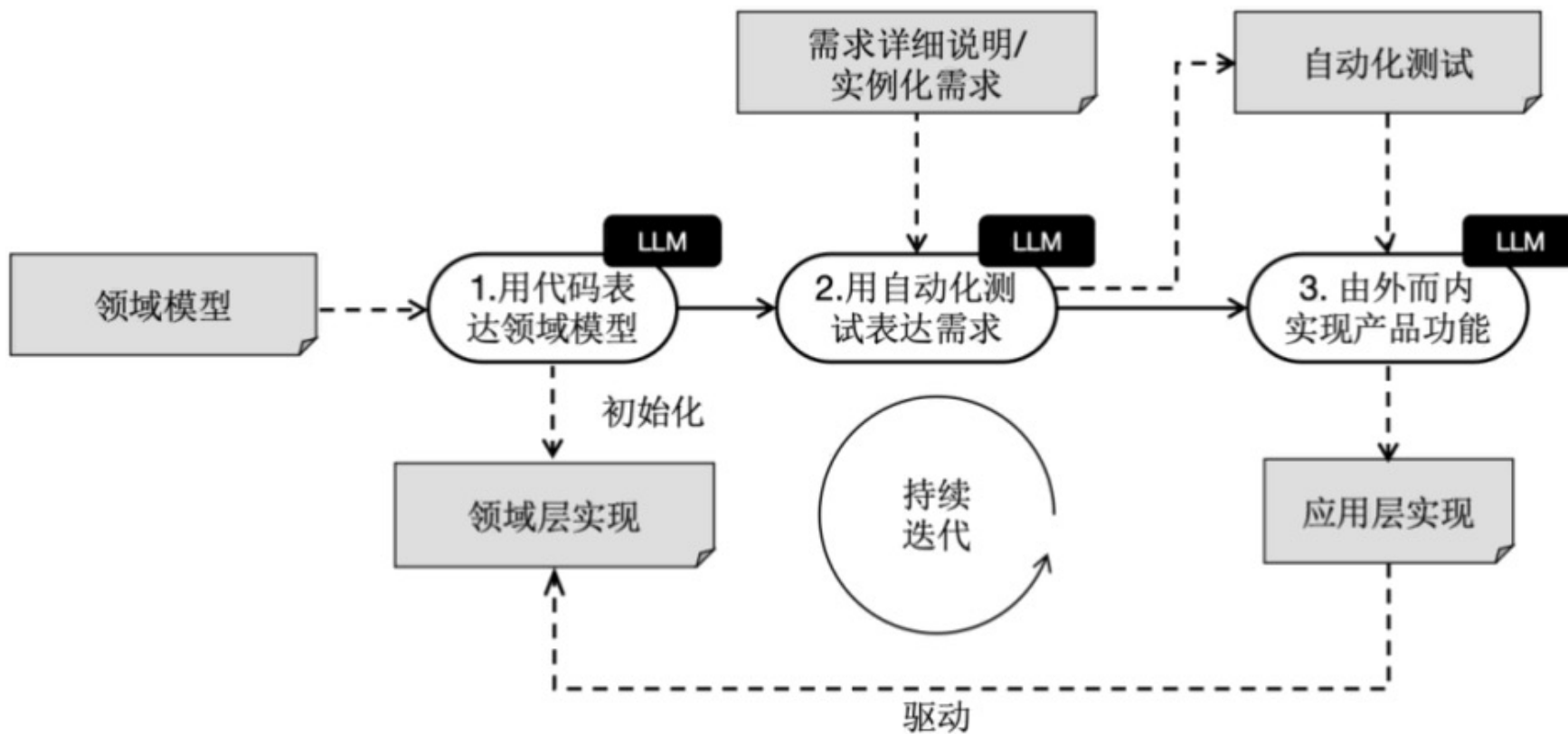
2. 用正交、控制变量的方法设计用例数据，确保除了关注的变量之外，给出的用例不影响到规则的达成。例如，当你考虑时间段匹配时，可以把出行计划的对应的起始地都设成地标 1，目的地都设成地标 2，这样就可以消除由于距离导致的匹配问题。

3. 采用 Given-When-Then 的形式输出。每个规则都需要列出出行计划和共乘的具体数据。上述要求非常重要，请你在完成案例之后先自我检查，确保你的输出符合上述要求

# 领域模型



# ▶ 先把模型层写好，有助于高质量的后续开发





# ▶ 很好的利用好由外而内，可以更精准地表达需求

## 实现出行时间匹配规则

首先我们来实现 `departureTimeNotMatch`。时间匹配是一个独立的类：`TimeSpanMatcher`，然后把具体如何匹配时间的问题委托给它

```
private boolean departureTimeNotMatch(TripPlan existPlan, TripPlanDTO newPlan) {
    return !new
    TimeSpanMatcher().match(existPlan.getPlanSpecification().getPlannedDepartureTime(),
        newPlan.getPlanSpecification().getPlannedDepartureTime());
}
```

接下来让我们继续把工作交给LLM：

### 提示

我有如下的代码：

```
public class TimeSpan {
    LocalDateTime start;
    LocalDateTime end;
}

private boolean departureTimeNotMatch(TripPlan existPlan, TripPlanDTO newPlan) {
    return !new
    TimeSpanMatcher().match(existPlan.getPlanSpecification().getPlannedDepartureTime(),
        newPlan.getPlanSpecification().getPlannedDepartureTime());
}

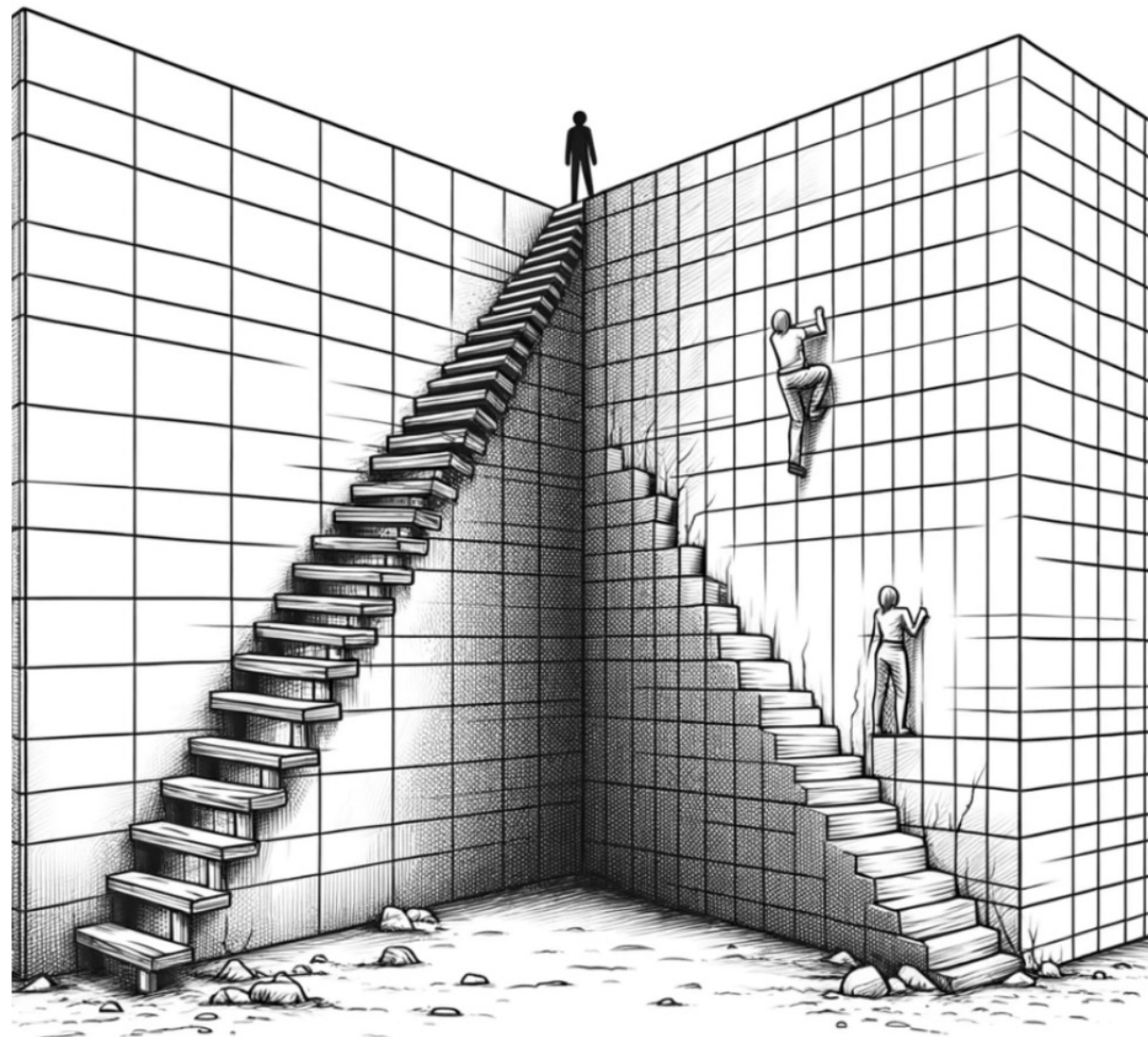
public class PlanSpecification {
    public PlanSpecification(Location departureLocation, Location arrivalLocation, TimeSpan
    plannedDepartureTime) {
        this(departureLocation, arrivalLocation, plannedDepartureTime, 1);
    }
}
```

请基于以上代码，编写TimeSpanMatcher类和对应的单元测试。

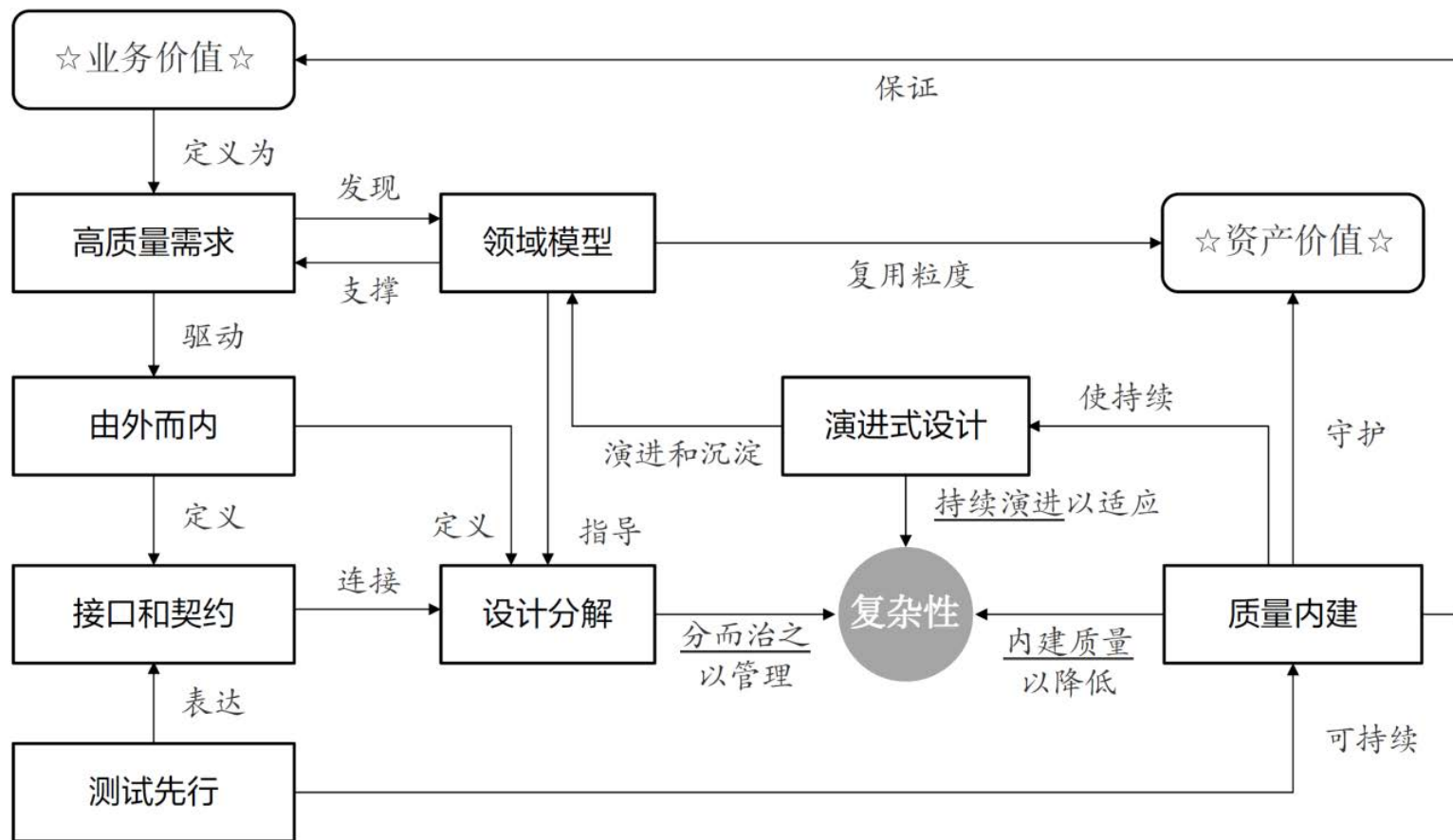
# ► 为什么我如此推崇演进式设计

软件开发的复杂性，  
会让一次性解决所有  
问题变得非常困难

但是，如果能解决一个  
小问题，就为解决  
更大的问题建立了更  
好的基础



# 精益软件设计方法框架：AI时代的开发人员专业技能





# PART 04

## 一起动手

# ▶ 我准备了3个案例...

案例1: 演进式的方式设计俄罗斯方块游戏 (难度系数 2.0, 有趣程度 2.0)

案例2: 实现自动玩俄罗斯方块的能力 (难度系数4.0, 有趣程度 4.0)

案例3: 在“共享出行”的基础上增加顺风车的基于路径的匹配, 或者增加共享巴士功能 (难度系数3.0~4.0, 有趣程度1.5)

# ▶ 环境检查

## 1. 有至少1个大模型辅助开发工具

- ChatGPT
- Claude
- 讯飞星火、通义千问、文心一言、智谱清言等
- 有Cursor或同类工具更好，没有也关系不大

## 2. 有git和IDE

- 及时保存成功的进展
- 可以下载示例项目

## 3. 编程语言

案例1可以使用任何编程语言，案例3只能使用Java，案例2处于二者之间



## ▶ 确保可持续演进

无论选择什么项目，  
请首先确保有一个可运行的系统，**并保证它始终可运行**

# ▶ 步子可以迈多大?

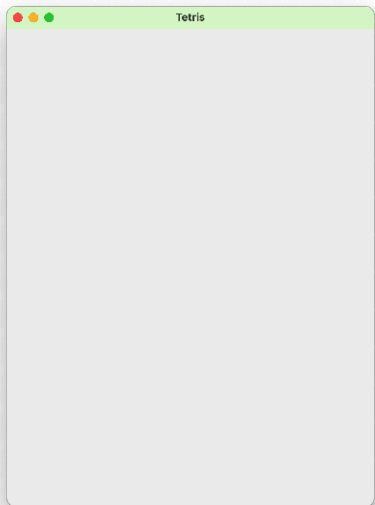
人类的能力和大模型的能力的交集

# ▶ 请坚持契约先行、测试先行

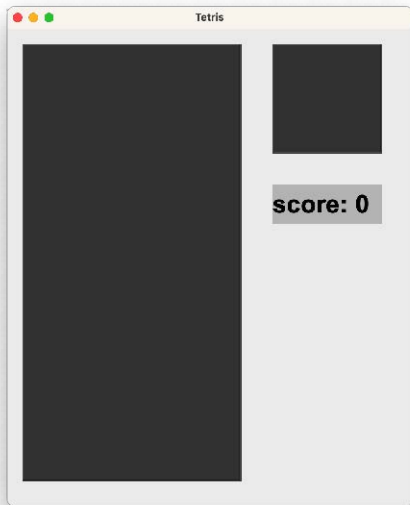
**清晰地描述要做的工作，如果可能，做到测试先行**



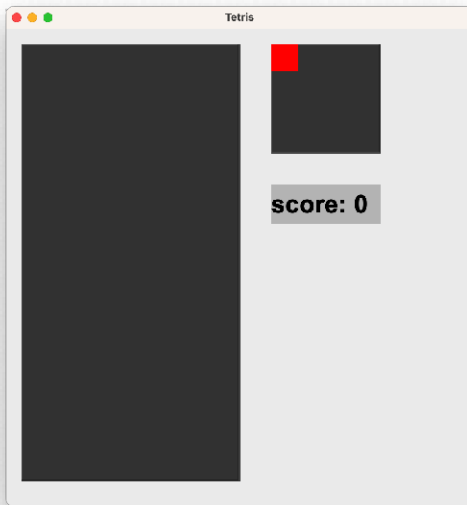
# ▶ 案例1: 俄罗斯方块游戏



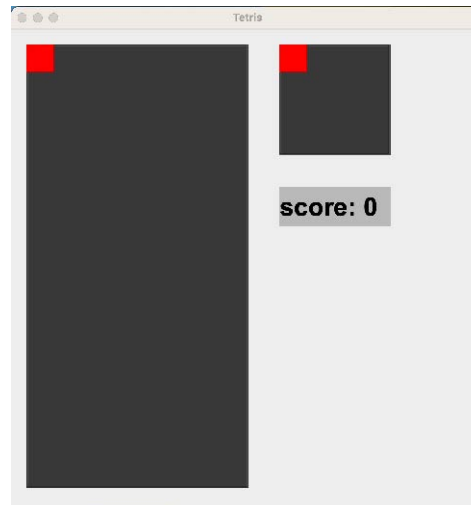
空白游戏



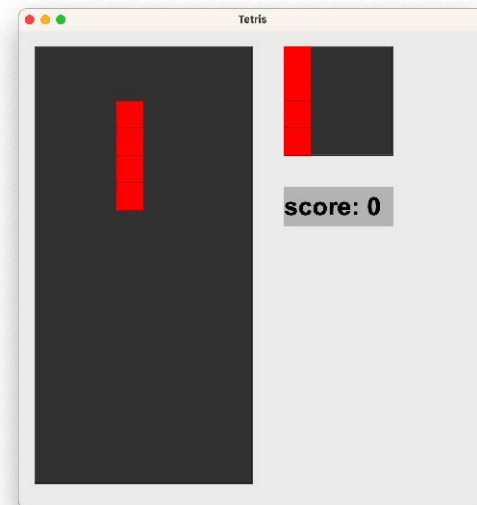
基本布局



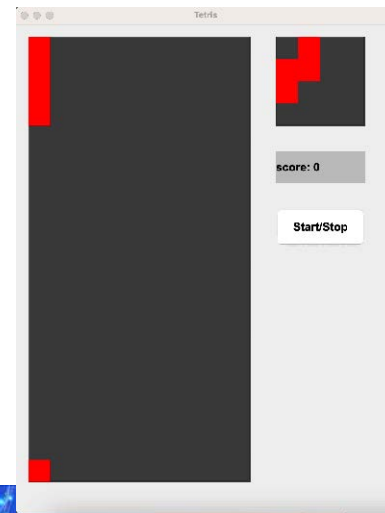
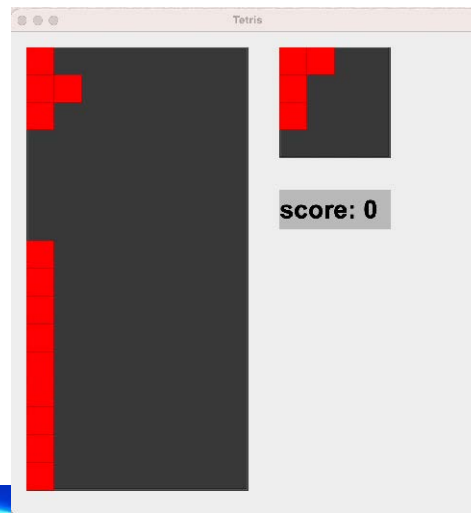
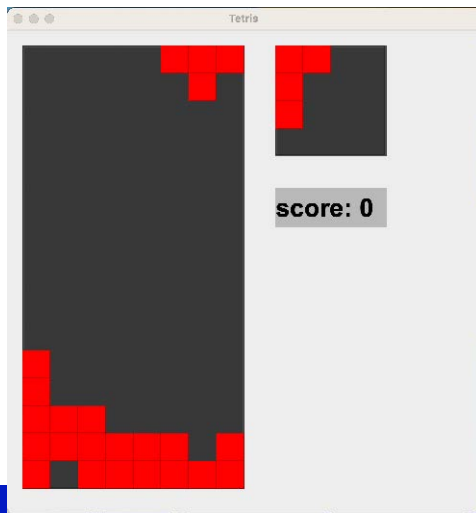
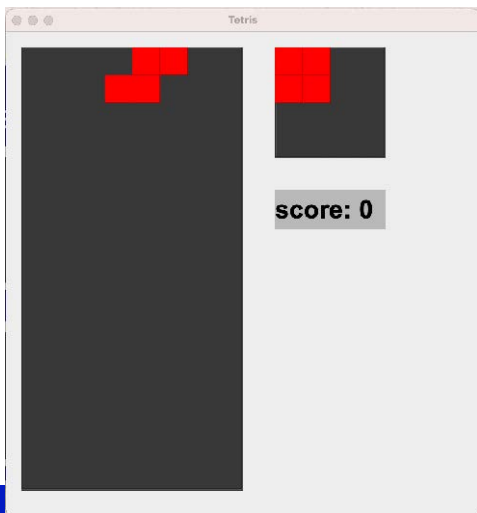
第一个“方块”



可以移动



真正的方块



```
commit c0791e36434384b7ef8c894cef7a5a7e31a64116
    create elements of game(main board, preview board, score board)
commit f02223ed9379332d54770bd7e2275f431a6d2df9
    can display block in cell panel
commit 58c5e8bde6211454961319922917d81d5e074486
    game start: create active block
commit fac9afb62ddb382906a66d82fed10c56ec57593
    can accept key event: press enter to start game
commit 30a5972b1e2e9b58a5693bbe7c646cd7f028c39e
    can move down block in controller
commit 03c18f85934438ae3851a439e4a919a42f6b86b5
    add move left and move right implementation
commit 4652a270c5d2b2043ab572daeb08910ae20f506a
    can produce a bar
commit 37e2d0cd4b0d6564fdce6ec9afee40e8d63f8feb
    refactor: split block and shape concept
commit 8ed0ca4326868e776ed2a663087d429b59d3e709
    can produce bar z
commit c3e0f00fa47b8580e37d9ed10964305583e4bb9b
    implementation of shape I rotation
commit 759ac1465f223353a16e568f79fe125a91c7968c
    make block factory select shape randomly
```

👉基本布局

👉第一个“方块”

👉响应键盘

👉向下移动

👉左右移动

👉真正的方块

👉重构

👉更多方块

👉随机方块

```
commit 28b66a1fbf1b338bc68939f6b7f31371582d7033
    can turn active block
commit d5e49769581283d9a87a0b0f047d22922501c575
    implement Collision Detector
commit 24b693154cb80d50ff509c49ab471103a17e27902
    implement Block.join()
commit a2adbe9823a9b6dedf681496290c0301472410a
    can detect left and right wall
commit dc750503cc3fdd945dad4551224a0d5cc412537ea
    can eliminate whole row
commit 63d6161edff8681f6b9c9ba6961fc9a120d129
    can compress shape after elimination
commit e4711b7b65dd3fddd300f8f859f44fa2a645a025
    can notify game over
commit fdce55120bd7f08561358887118aee350f3705b
    add score keeper
commit e056ce553133f0fd7f87c3a73425768cc0583dc
    add timers
commit d7e50c1c63f185f514ef633ccba0ef45eb07e91f
    add start/stop button
```

👉可翻转

👉碰撞检测

👉底部沉积

👉左右碰撞

👉消除整行

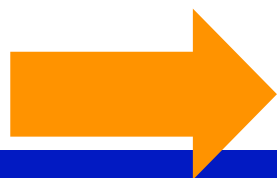
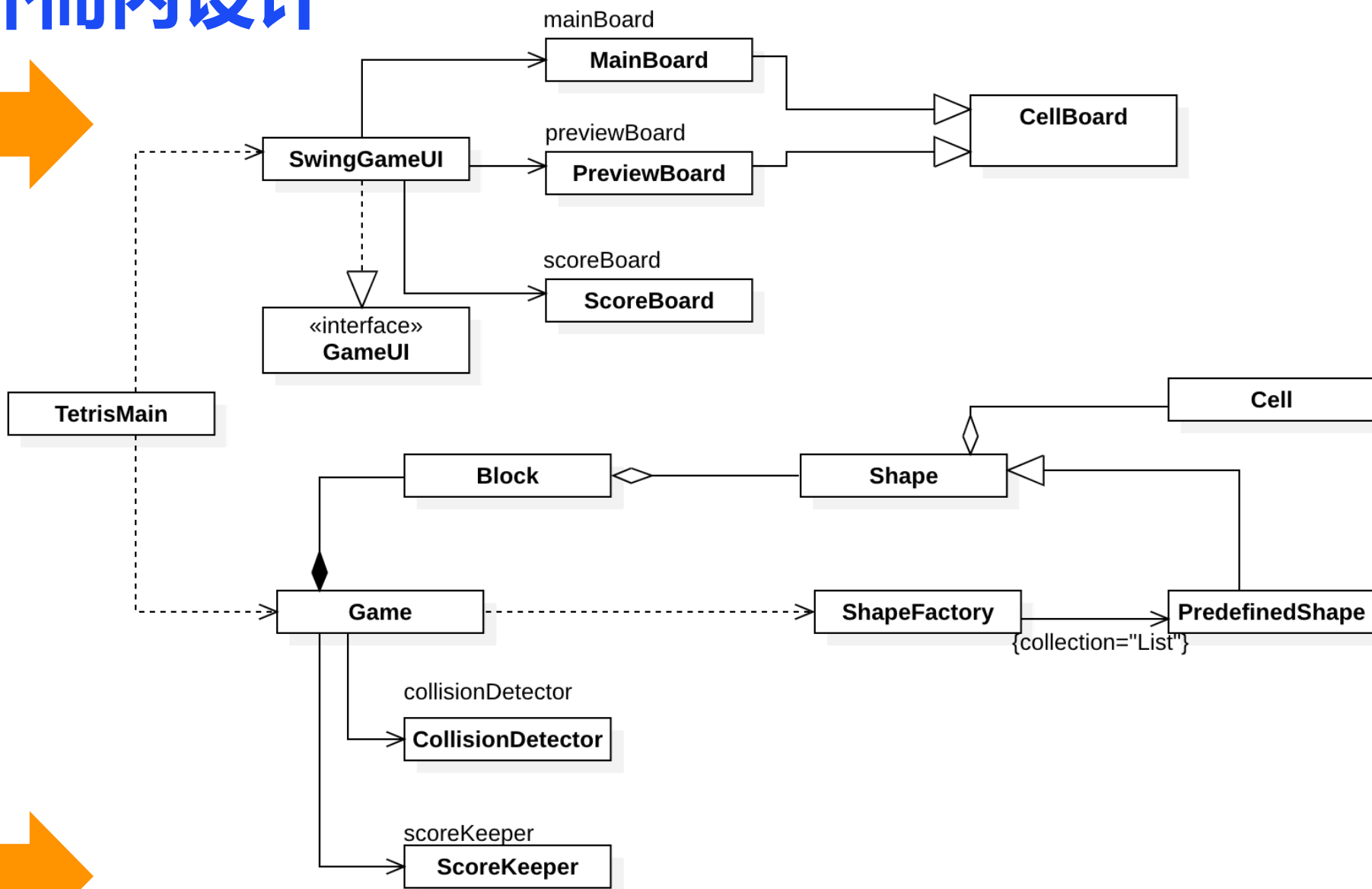
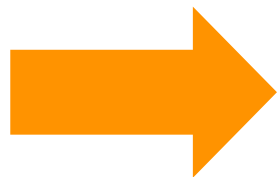
👉游戏结束

👉增加计分

👉增加定时器

👉增加控制按钮

# ▶ 尽量由外而内设计





## ▶ 步骤

1. 选择一个大模型辅助开发工具，  
(借助大模型或不借助大模型均可) 从一个空系统开始——验证环境、测试框架等
2. 规划大致的演进路径，尝试第一个功能  
(路径不同，演进的难度不同。可以尝试利用大模型做规划，不过人的判断仍然是重要的)
3. 如果涉及到复杂的部分（例如算法），可以先用自动化测试来驱动这部分的实现，先完成模块级开发，然后把这部分内容嵌进去

**请留意记录过程中的心得和问题，练习结束后进行分享**

## ▶ 参考资料

<https://gitee.com/gangz2009/tetris-by-chat-gpt>

### ChatGPT结对编程实录：提升生产力，还是被代替？

原创 张刚 演进式设计 2023年03月22日 20:27 美国

本文是一次编程实录。不过，未来的编程大概率不是这样的。AIGC<sup>Q</sup>和业务人员直接对话，而不是经过程序这层“翻译”，是更大概率的事件：

用户：我需要上线一个网站。需求是....

ChatGPT: 这是你的网站，你可以预览一下看看是不是你需要的。

用户：我希望做一些改进...

## ▶ 案例2: 使用遗传算法编写一个自动玩俄罗斯游戏的机器人

先来看一下期待的最终结果



## ▶ 演进路径指导（注意：每一步都要可工作）

1. 首先，要有一个虚拟的玩家，可以操纵游戏
2. 要有一个规划器，读取游戏的状态，并且为玩家生成规划  
(请思考：需要几个参数？)
3. 实现特征计算模块（5个特征，用于计算适应度）
4. 基于特征，通过硬编码的加权值，选取最优策略
5. 实现遗传算法，计算加权值
  - \* 创建种群
  - \* 实现交叉和变异
  - \* 在每一代之间优胜劣汰

**提醒：请充分利用大模型“解决未知的未知”的能力！**

# ▶ 从这个地方开始

<https://gitee.com/gangz2009/tetris/>

G the 'AI player' only move block right 2 columns and turn it

0e2ebe4



gangz 提交于 2年前

G add an AI player without any behavior

d668256



gangz 提交于 2年前

G add start/stop button

d7e50c1



gangz 提交于 2年前

G enlarge the window of game

74b7517



gangz 提交于 2年前



# ▶ 文档参考

写一个AI俄罗斯方块，然后看着它玩...

原创 张刚 演进式设计 2022年09月03日 10:33 上海

我很喜欢俄罗斯方块这个游戏。俄罗斯方块这么古老，为什么还有生命力呢？

和许多游戏一样，俄罗斯方块总以玩家的失败告终。生存在即将死去的恐惧中，不努力就失败，成功消除则可以获得奖赏（得分）。在这样的游戏中，丝毫不敢懈怠，是不是像极了人生？

有一天我突然想到（其实别人早就想到啦），现在AI这么火，为啥不写一个俄罗斯方块的机器人，让它自己玩呢？AI肯定比人强，把方块码得整整齐齐的，还不容易死亡，只需要看着它玩，感觉不是很好吗？

就怎么干了。先给朋友们看看效果：

## ▶ 案例3: 共享出行的演进

在当前共享出行的基础上，引入顺风车的路径规划功能和共享巴士功能



## ▶ 开发指导-顺风车（注意：每一步都要可工作）

1. 首先要理解现在的领域模型和关键概念（阅读代码的domain包的内容）
2. 分析：当前顺风车的路径规划只能支持哪种简单的情况？
3. 你能写一个测试案例，来表达一个更复杂的匹配吗？  
为了能实现这个匹配，你需要考虑哪些能力？  
(提示：如果不熟悉该领域，可以使用AI帮助进行方案设计)
4. 试着使用AI编写这部分测试和代码，并且每一步都可执行
5. 进一步思考如何进行性能优化

## ▶ 开发指导-共享巴士 (注意: 每一步都要可工作)

1. 首先要理解现在的领域模型和关键概念 (阅读代码的domain包的内容)
2. 分析: 如果要加入共享巴士, 领域模型会发生什么变化?
3. 你能写一个测试案例, 来表达共享巴士这个业务吗?
4. 试着使用AI编写这部分测试和代码, 并且每一步都可执行
5. 进一步思考如何进行未来的功能优化

# ▶ 从这个地方开始

<https://gitee.com/leansd/cotrip>

## ▶ 请在练习过程中留意如下的问题

你的开发过程保持了始终可演进吗？

你利用大模型做了哪些事情？

你认为专业技能还重要吗？

你期待怎样的辅助开发工具？



# **PART 05**

## **分享和总结**

## ▶ 练习过程分享

开发过程保持了始终可演进吗？

利用大模型做了哪些事情？

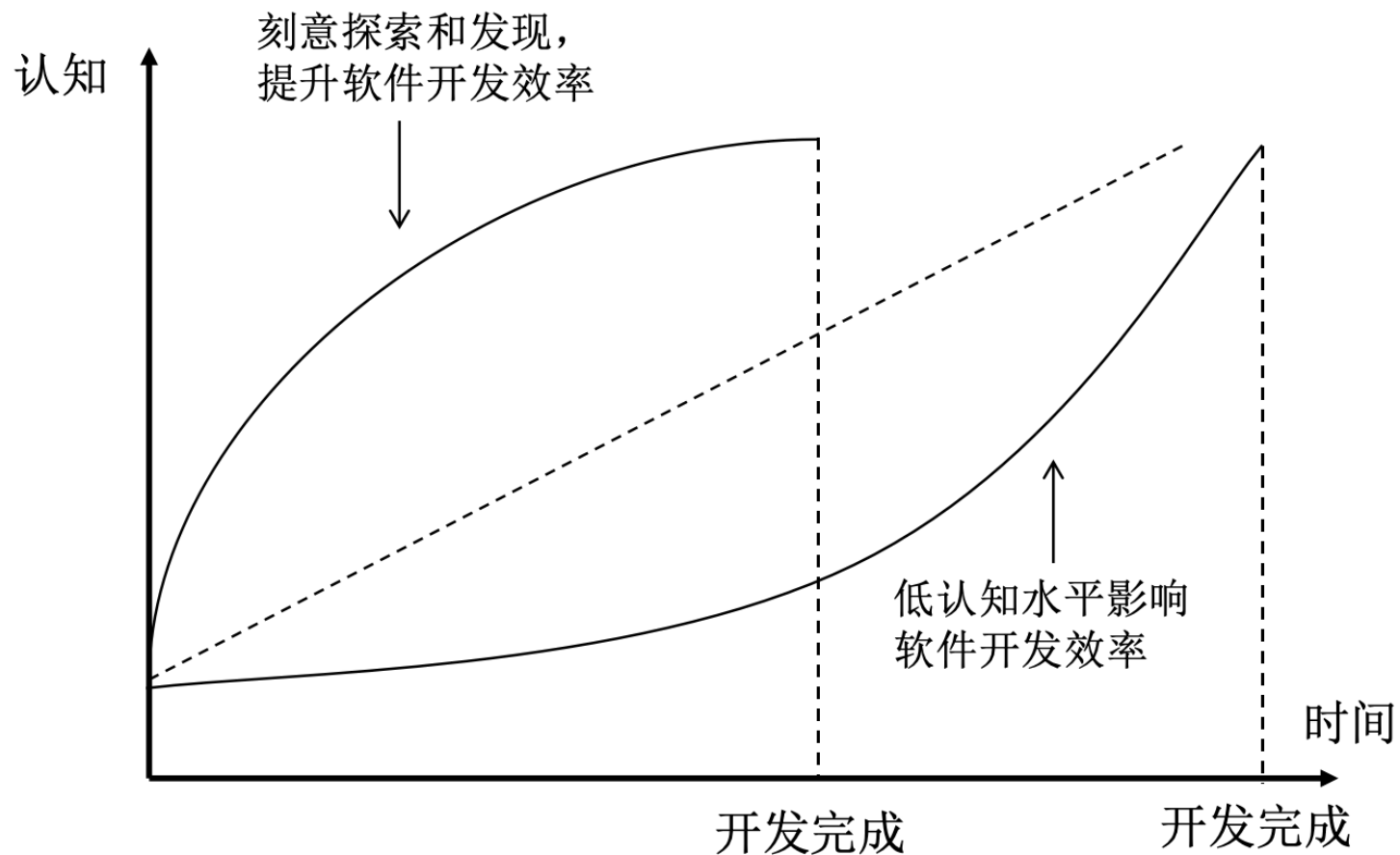
专业技能还重要吗？

期待怎样的辅助开发工具？

# ▶ 更广泛地利用LLM，更高效地开发软件

- LLM作为工程师的第二大脑
  - 各种端到端的开发工具逐渐涌现
- 人类的认知水平和GPT协同的效果正相关，强者愈强

# ► 软件研发的本质是“探索 and 发现”



要解决的问题是什么？

采用什么解决方案？

实际把方案建造出来



# ▶ 大模型时代，软件开发回归本质

- 本质：软件开发是“探索 and 发现”的过程
- 大模型加速了建造速度，提升了“探索 and 发现”的效率
- 它把程序员从繁琐的“编码工作”中解放出来，提升了思考的抽象层次和反馈的速度

# ▶ 新全栈工程师，成为更卓越的开发者

- 每个工程师对价值负责
- 更少任务协同、更多价值协同
- 未来的开发者需要做的事情是：理解软件的复杂性，并能够将一个大的问题、挑战或新特性分解成小问题。作为开发人员，要能够管理解决大型复杂问题的大型复杂系统。而且，你需要能够将它们分解为小的构建块

—— Thomas Dohmke (GitHub CEO)

# 科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



 **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**  **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



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

