

大模型时代, AI 辅助编程工具的技术演进



自我介绍



黄峰达 (Phodal) / 代码专家

- Thoughtworks 开源解决方案负责人, 开源架构治理平台 ArchGuard 架构师
- 华为编程语言社区 SIG-DSL 扩展核心成员
- 开坑, 开到填不完的知名开源挖坑选手: <https://github.com/phodal>, 开源有 Java 重构分析工具 Coca、Mooa 微前端框架、ArchGuard 架构治理平台等
- 著有《前端架构: 从入门到微前端》、《自己动手设计物联网》简/繁版本等书



Fklang - 架构设计 DSL

<https://github.com/feakin/fklang>



Chapi - 多语言静态代码分析引擎

<https://github.com/phodal/chapi>



ArchGuard - 开源架构治理平台 2.0

<https://github.com/archguard/archguard>



ClickPrompt - 学习与分享 prompt

<https://github.com/prompt-engineering/click-prompt>



AutoDev 开发人员的 AI 辅助编程 IDE 插件

<https://github.com/unit-mesh/auto-dev>

目录

探索大模型时代下 AI 辅助编程工具

AI 辅助编程工具从个人、团队、组织层面的变化，还有从影响开发到全生命周期管理的技术路线，包括跨 IDE 架构和差异化设计的新动向。

全面探索：从开发到全生命周期

演进路径：个体、团队、组织

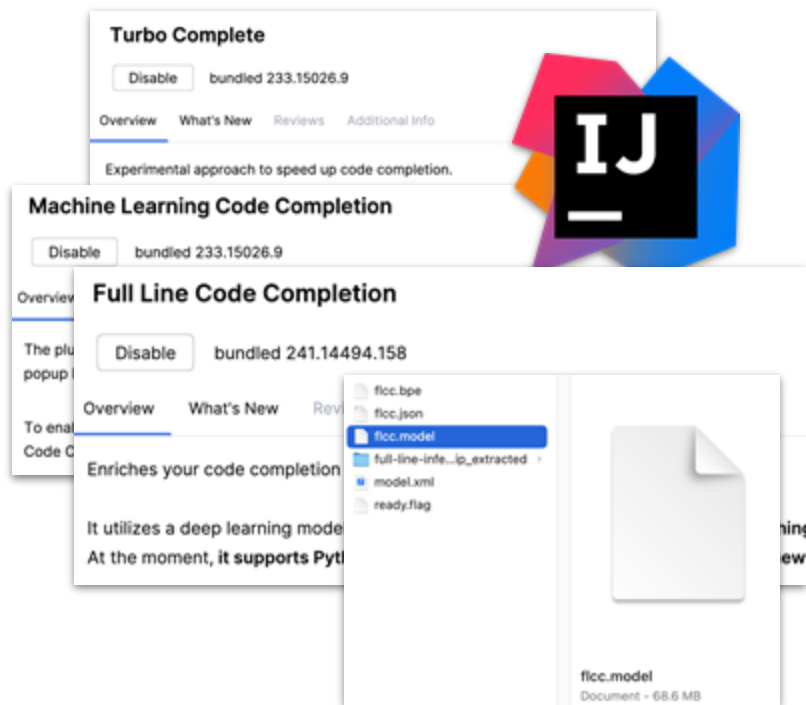
技术变化：跨 IDE 架构到差异化设计

全面探索： 从开发到全生命周期



AI 代码补全正在将成为 IDE 基础设施

如何在 SDLC 更好发挥生成式 AI 价值将成为挑战？



Copilot

AI in Editor/ IDE, Web, Voice



Advance Security

Dependabot, auto-fix...



Action

Auto Build-failed fix..



Codespace

AI Editor, Web, Voice...

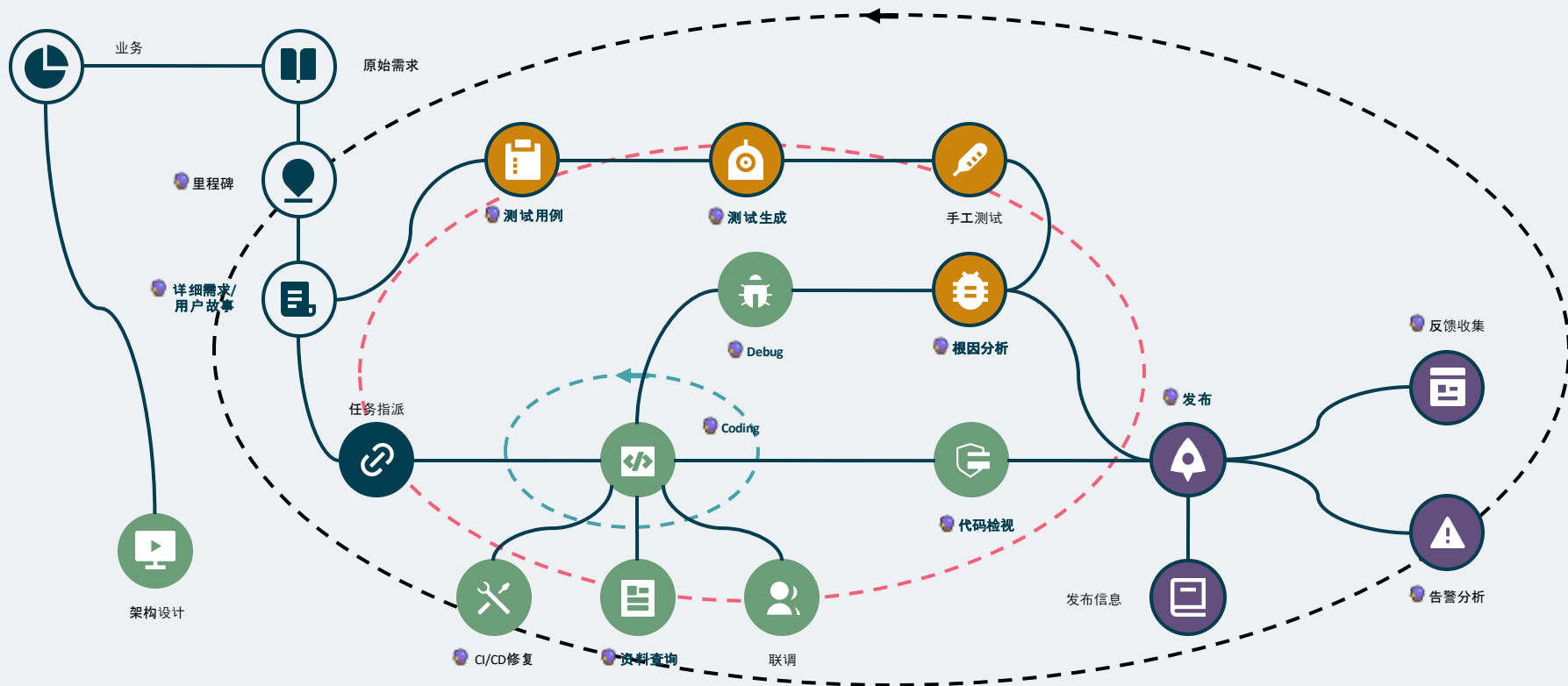


Workspace

Auto Task/Coding/Build/Deploy

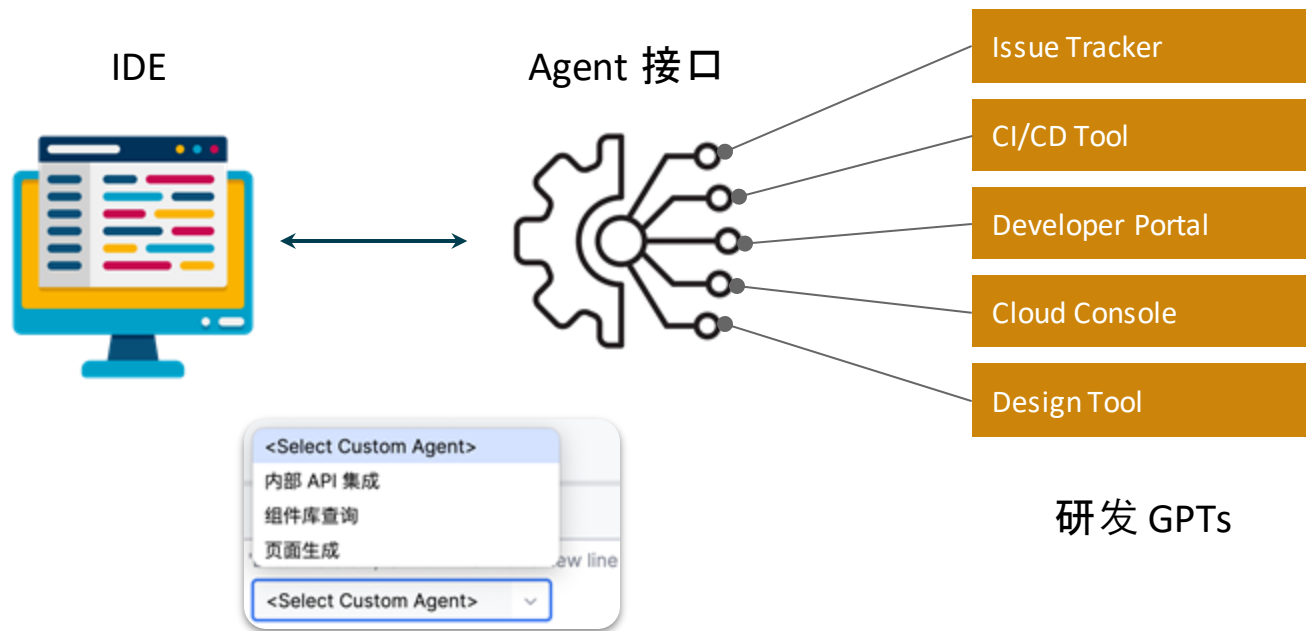
AI 辅助研发工具链建设路径

从辅助个人效率提升，到团队自定义辅助能力，再到建设组织级智能体



IDE 插件：从个人编码助手到团队辅助

放大知识，提升初级团队成员的技能/促进技能的多样性



演进路径： 个体、团队、组织



个体：自定义 IDE 的 Action

```
{
  "spec": {
    "controller": "- Use
BeanUtils.copyProperties in the Controller for
DTO to Entity conversion.",
    "service": "...",
  },
  "prompts": [
    {
      "title": "🌐🌐 Code complete",
      "autoInvoke": false,
      "matchRegex": ".*",
      "priority": 0,
      "template": "Complete
code:\n${SIMILAR_CHUNK}\n```\njava\n${METHOD}
_INPUT_OUTPUT}\n```\nmarkdown\n${SPEC_con
troller}\n```\n${SELECTION}"
    }
  ]
}
```

User

Complete code:

```
Java ①
// title
// content
//
// }
// class BlogPost {
// id
```

More lines

Markdown ①

- Use BeanUtils.copyProperties in the Controller for DTO to Entity conversion.
- Avoid using Autowired.
- Use Swagger Annotations to indicate API meanings.
- Controller methods should capture and handle business exceptions, rather than throwing system exceptions.

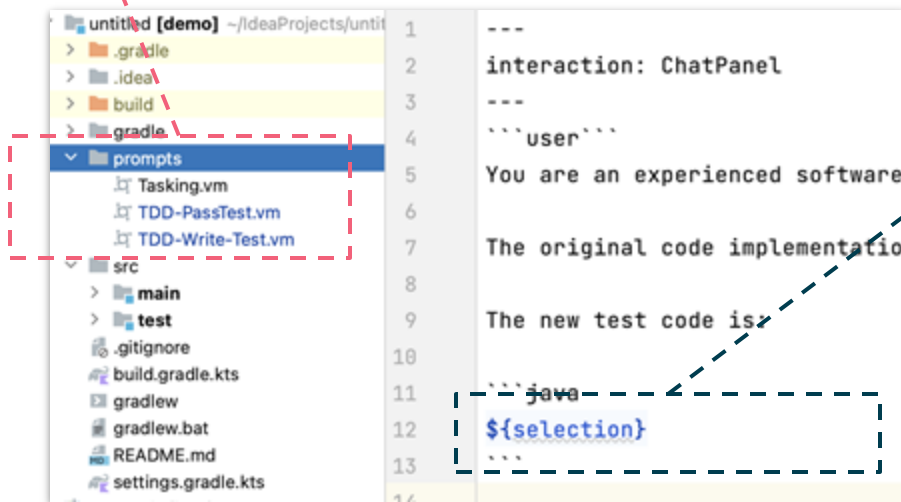
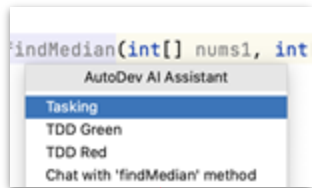
Java ①

```
@ApiOperation(value = "Create a new blog")
@PostMapping("/")
public BlogPost createBlog(@RequestBody
CreateBlogRequest request) {
```

Prompt (用户视图)

团队 AI：自定义团队的 AI Action

Prompt 即代码，在团队中共享适合于项目的 AI 辅助实践，提升整体开发效率



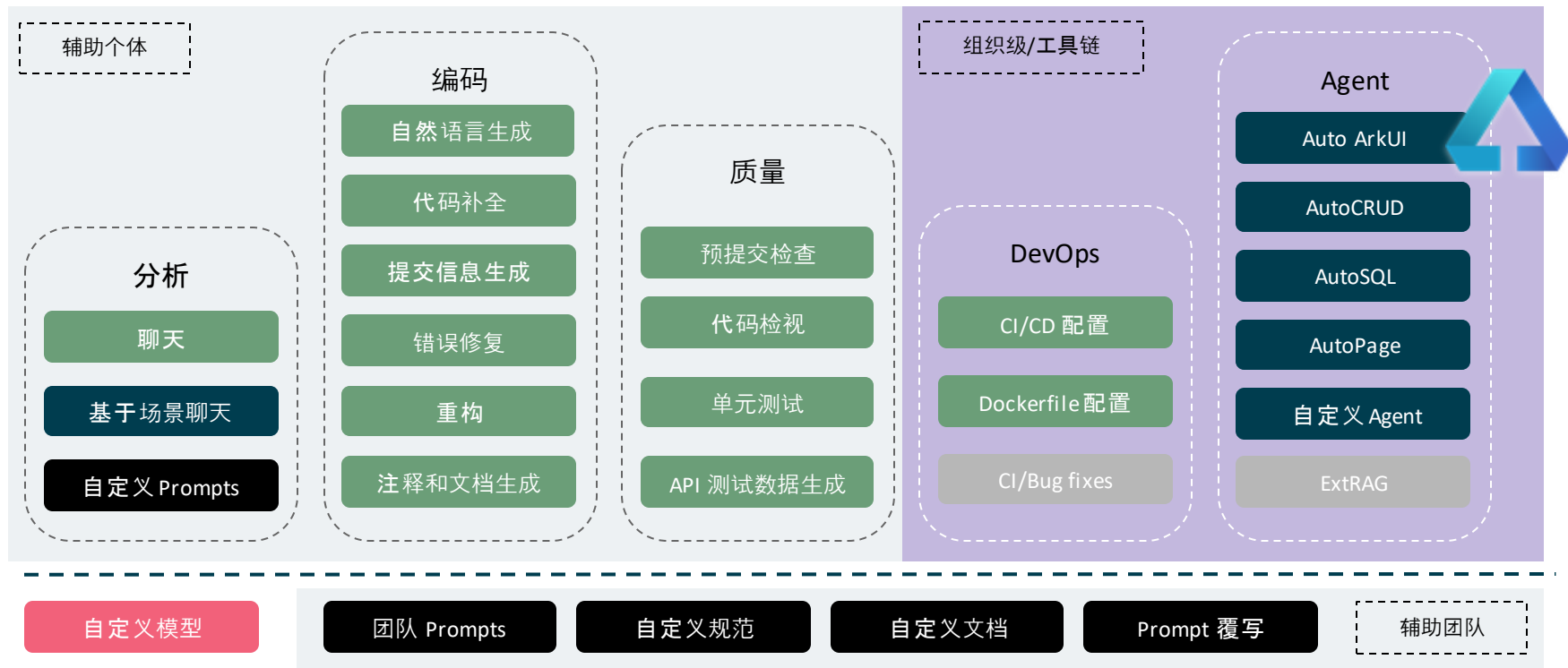
- [selection](#): 当前选定的文本。
- [commentSymbol](#): 获取当前语言的注释符号，例如：
`//, #, --, /* */`
- [beforeCursor](#): 当前光标之前的文本。
- [afterCursor](#): 当前光标之后的文本。
- [language](#): 当前文件的语言，例如：`kotlin, java, python, javascript`。
- [fileName](#): 当前文件的文件名。
- [filePath](#): 当前文件的文件路径。
- [methodName](#): 当前方法的方法名。
- [frameworkContext](#): 获取当前文件的框架上下文，例如：`spring, junit, mockito` 等。

组织：AI 辅助研发组织的技术蓝图



开源辅助开发 IDE 插件 AutoDev 能力全景

大模型作为软件研发知识传递者，增强个体、团队、组织效率和体验

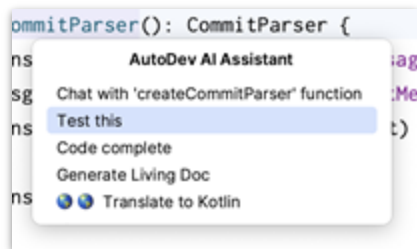


技术演进： 跨 IDE 架构到差异化设计



AutoDev 的上下文工程：精准单元测试生成示例

结合静态代码分析、依赖分析、通用测试规范，生成测试上下文，以生成精准的测试用例



Intention

(Alt + Enter)

Write unit test for following code.

You MUST use `should_xx` style for test method name.
When testing `controller`, you MUST use `MockMvc` and test API only.

You are working on a project that uses `Spring MVC`, `Spring WebFlux`, `JDBC` to build RESTful APIs.

```
// class BookMeetingRoomResponse {  
// ...  
// }  
// class BlogController {  
//   blogService  
//   + public BlogController(BlogService blogService)  
//   + @PostMapping("/blog") public BlogPost  
createBlog(CreateBlogDto blogDto)  
//   + @GetMapping("/blog") public List<BlogPost>  
getBlog()  
// }
```

```
```java  
@PostMapping("/{meetingRoomId}/book")
public ResponseEntity<BookMeetingRoomResponse>
bookMeetingRoom(@PathVariable String meetingRoomId, @RequestBody
BookMeetingRoomRequest request) {
 // 业务逻辑
 BookMeetingRoomResponse response = new BookMeetingRoomResponse();
 // 设置 response 的属性
 return new ResponseEntity<>(response, HttpStatus.CREATED);
}
```
```

Start with ``import`` syntax here:

Action 类型

语言上下文
(结合规范)

技术栈上下文

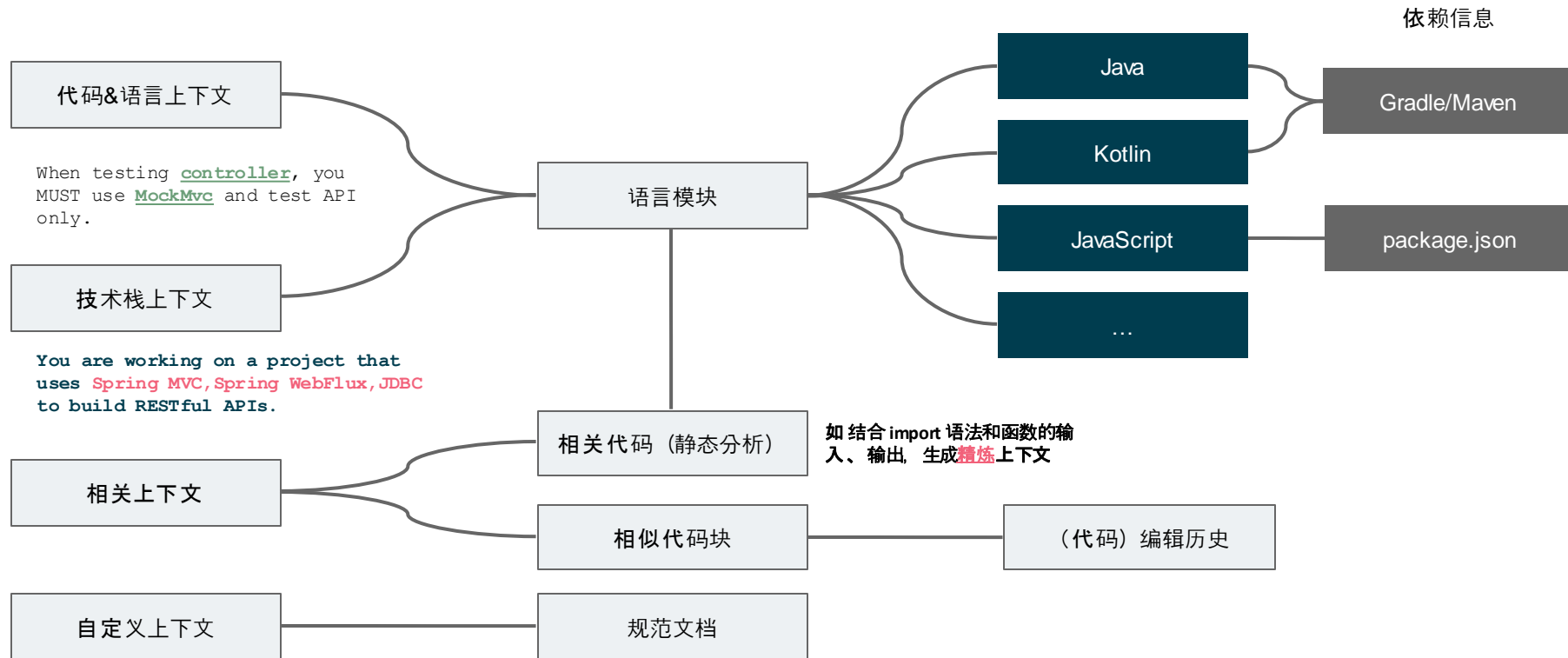
相关上下文
(ClassProvider)

代码
(PsiElement)

引导词

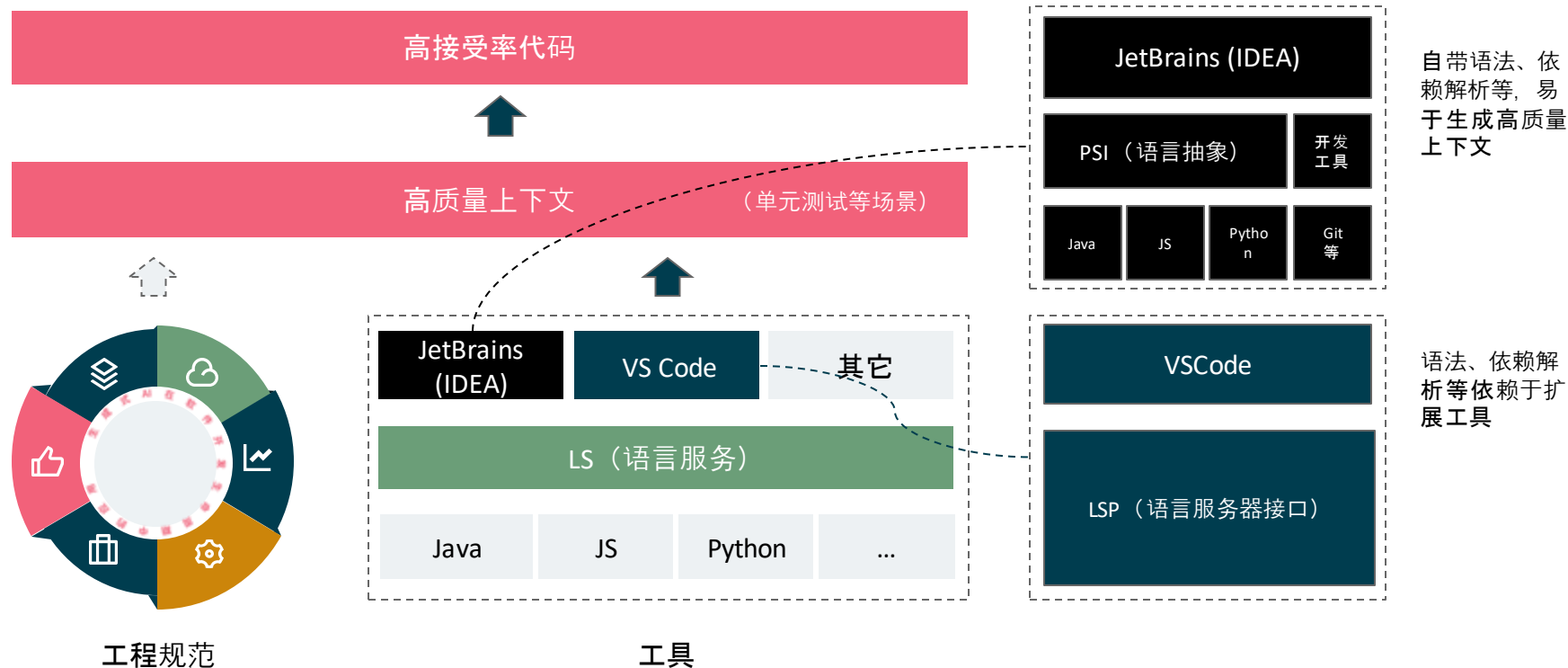
AutoDev 上下文架构示例

围绕更准确的上下文生成，构建 IDE 插件架构



AI 辅助插件技术架构演进曲线

不同 IDE/编辑器在编程语言解析能力差别大，影响上下文生成质量，需要靠架构补齐差异



One more thing...

AutoDev 的智能体语言：DevIns

最近在挖的坑：构建 **人类-AI-IDE** 的沟通语言（DSL）

The screenshot displays the DevIns IDE interface. On the left, a file named `lookup-endpoint.devins` is open. The code in the editor is a DSL for a CRUD expert. It includes a prompt: "你是一个 CRUD 专家, 你的任务是选择最适合用户需求的 Controller:", a symbol definition: `/symbol:cc.unitmesh.untitled.demo.controller`, and a list of requirements: "以下是用户的需求: - 用户需要一个接口, 可以查询所有的用户信息". A Git context menu is open, showing a list of tasks such as "chore(controller): add bank account management", "chore: fix for items", "chore: init first version test template design", "chore: remove template", "chore: update controller", "chore: update for config", "chore: update for datas", "chore: update for ddocs", "chore: update ignore", "chore: update job design for AutoDev", "Create README.md", and "delete unused files". On the right, the "Run" panel shows the "模板编译结果" (Template compilation result), which is the same DSL code. Below that, the "执行结果" (Execution result) section is empty, indicated by a dashed line.

智能体

语法上下文

Git 上下文

模板编译结果

执行结果

感谢您的聆听

黄峰达

代码专家

fdhuang@thoughtworks.com

(PS : 欢迎来 PR)

