51CTO WOT

World Of Tech 2024

WOT全球技术创新大会

智启新纪
慧创万物

AI赋能软件研发再思考

覃 宇
Thoughtworks智能硬件数字化转型负责人

智启新纪
慧创万物

面向 AI 的软件研发再思考

AI 编码助手带来了价值，而且还在不断增长
在使用 AI 编码助手的程序员中，75%的人表示这些工具达到或超过了他们的预期；这可能就是95%的开发者已经在使用它们的原因。

野蛮生长的AI增加了企业的风险
最近的研究发现，自从引入AI助手以来，代码质量和安全性有所下降。

在编码以外的领域加速使用人工智能
只有 ~30% 的周期时间用于编码。不要仅仅关注编码和编码的速度，菜呢加速 AI 的成功。

行业趋势

- 编码助手的使用将会持续下去，而且还在快速增长
- 与现代工程实践相结合，来控制管理风险
- 实现持久价值：将 AI 辅助集成到团队流程中，在整个生命周期中将AI与工程实践和领域知识相结合，来放大价值

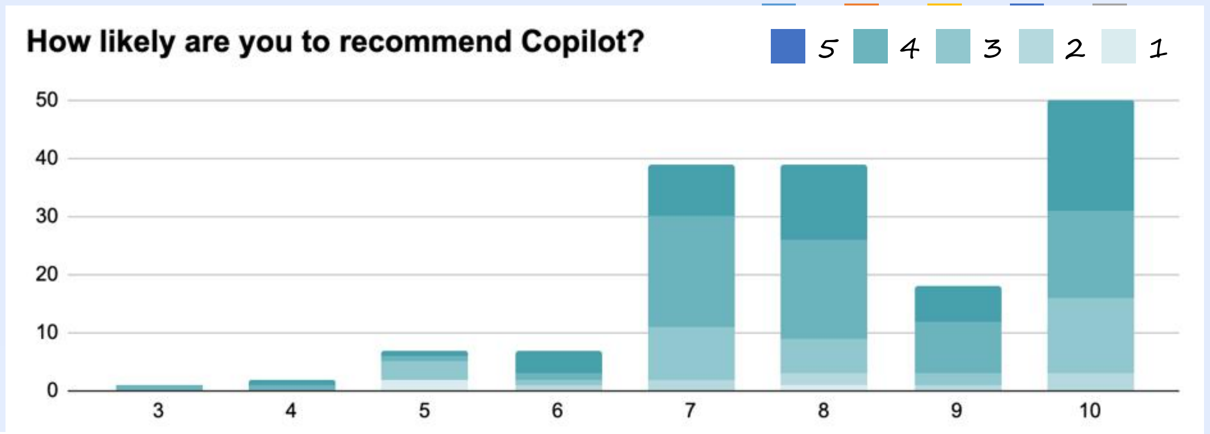
智启新纪 慧创万物

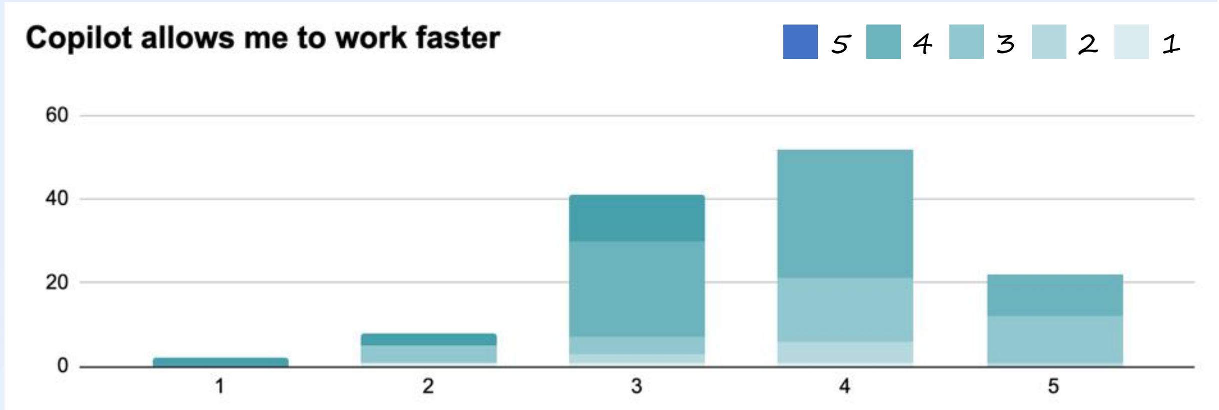
AI 辅助编码的价值已经验证

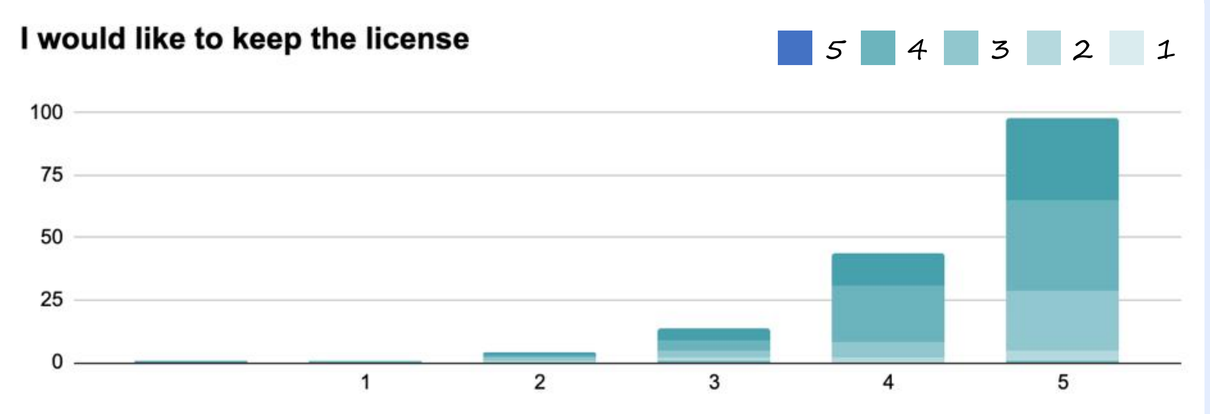
我们开发人员的体验证实，编码助手对开发人员的体验有积极影响。不同经验水平开发人员的反馈结果没有较大差别。

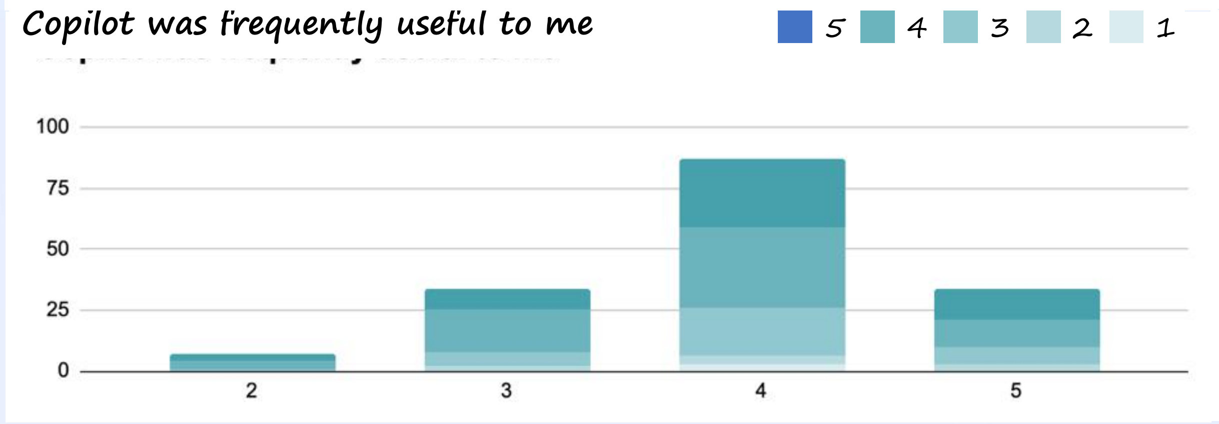
Experience level: ■ 5 ■ 4 ■ 3 ■ 2 ■ 1

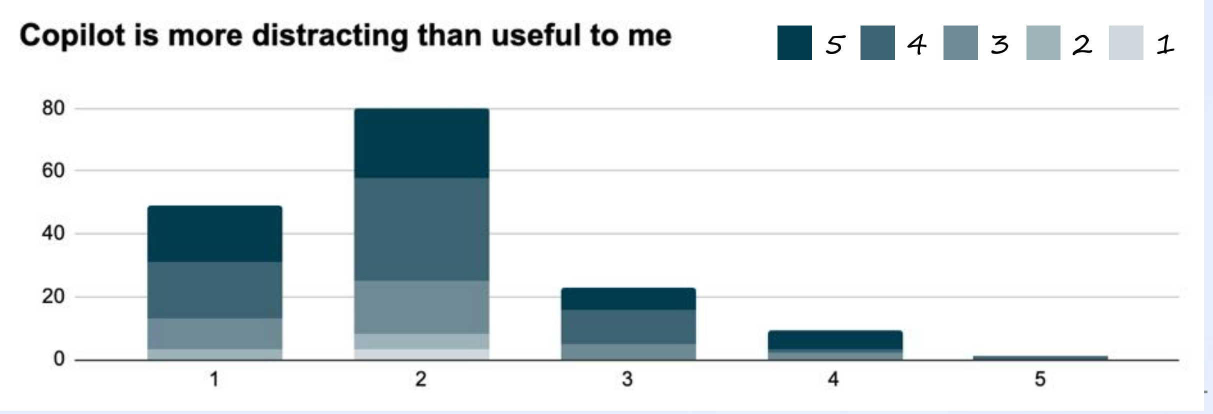
- *1-5 x-axis is a Likert scale of "Strongly Disagree (1)" to "Strongly Agree (5)"*











51CTO WOT

AI 辅助编码插件竞争白热化

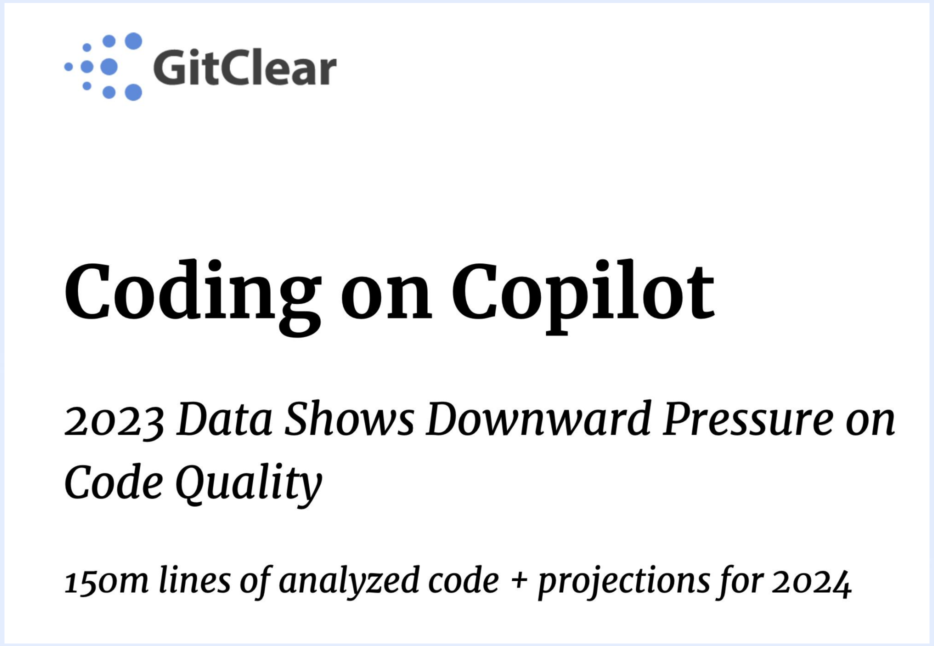
插件工具	**Github Copilot**	**Jetbrains AI**	**TONGYI Lingma**	**Baidu Comate**	**CodeArts Snap**	**Autodev**
即时补全	支持	支持	支持	支持	支持	**暂未支持**
自然语言转代码	支持	支持	支持	支持	支持	**支持**
对话聊天	支持	支持	支持	支持	支持	**支持**
代码解释	支持	支持	支持	支持	支持	**支持**
代码优化/重构	支持	支持	支持	支持	支持	**支持（结合代码扫描）**
测试生成	支持	支持	支持	支持	支持	**支持（基于调用的上下文）**
自定义模型	不支持	不支持	不支持	不支持	不支持	**支持**
自定义提示词	不支持	不支持	不支持	不支持	不支持	**支持**
代码上下文增强	未知	自动，无法定制	未知	自动索引，无法定制	未知	**支持（Context）**
业务上下文增强	不支持	不支持	不支持	不支持	不支持	**支持（AI Agent）**

1.结合代码扫描：诸如 *Coding Guidelines*、内部编码规划扫描出来的问题，作为提示词的一部分，可有效改进重构能力，修复质量问题。

警惕 AI 生成代码的风险

GitClear收集了2020年1月至2023年12月期间撰写的1.53亿行更改代码。这是已知最大的高度结构化代码变更数据数据库，用于评估代码质量差异。

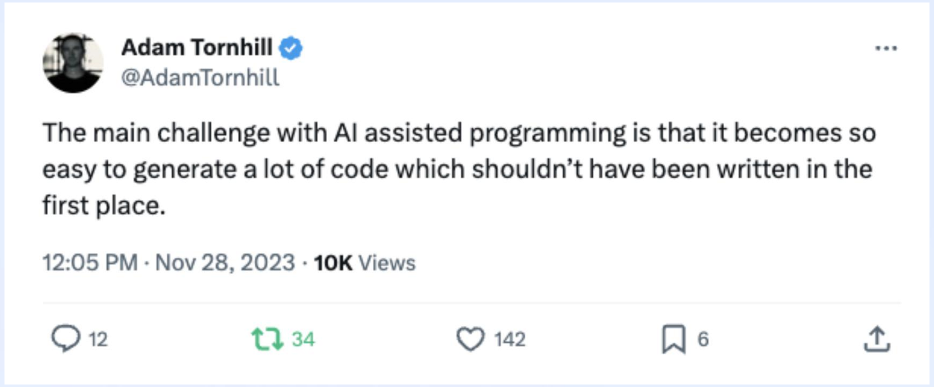
GitClear发现代码可维护性的趋势令人担忧。预计到2024年，代码流失率（即在撰写后不到两周的时间内进行恢复或再次修改的代码行百分比）将比2021年（人工智能出现之前的基线）翻一番。 “添加代码”和“复制/粘贴代码”的百分比与“更新”、“删除”和“移动”代码成比例增加。在这方面，2023年期间生成的代码更像来自流动的贡献者，容易违反代码仓库的 DRY 原则。


GitClear

Coding on Copilot

2023 Data Shows Downward Pressure on Code Quality

150m lines of analyzed code + projections for 2024


Adam Tornhill
@AdamTornhill

The main challenge with AI assisted programming is that it becomes so easy to generate a lot of code which shouldn't have been written in the first place.

12:05 PM · Nov 28, 2023 · 10K Views

12 34 142 6

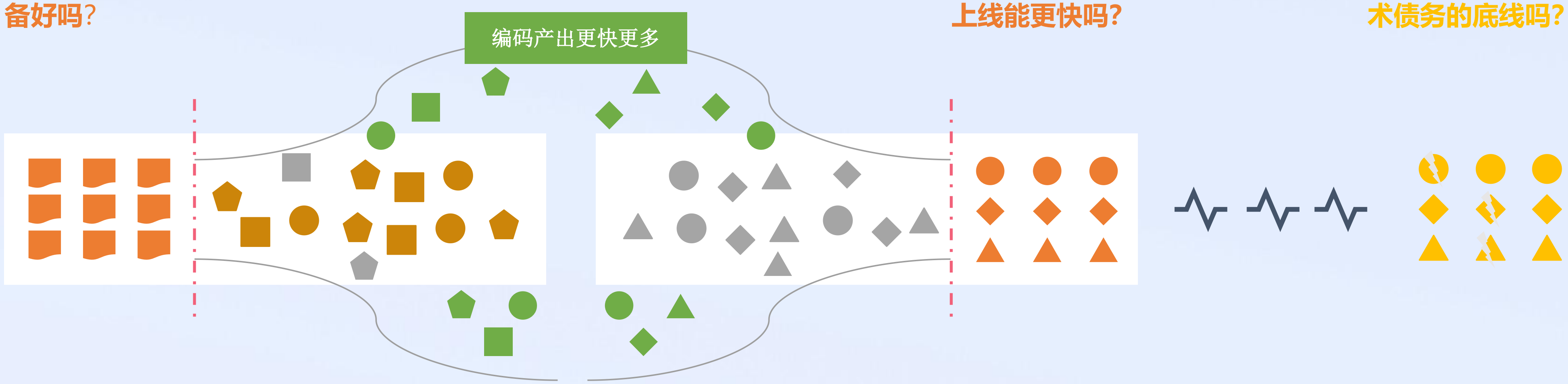
智启新纪 慧创万物

编码效率提升还暴露了其他短板

如果编码更快了，**需求能更快地准备好吗？**

如果编码更快了，**代码审查能更快吗吗？上线能更快吗？**

如果产出的代码更多了，**还能守住技术债务的底线吗？**



51CTO WOT

编码以外 AI 的使用仍不普遍

Research

- Market Research
- Competitor Analysis
- User Needs Identification
- Feasibility Study
- Technology Exploration
- Regulatory Requirements Identification
- Security Considerations
- Existing System Analysis (for modernization)
- Mobile Platform Analysis (for mobile projects)
- Data Analysis (for data projects)

Planning

- Project Scope Definition
- Communication Planning
- Project Governance
- Documentation Plan
- Mobile Platform Selection
- Resource Planning
- Budget Planning
- Risk Management
- Quality Assurance Planning
- Data Migration Plan
- Security Planning
- Project Scheduling

Analysis

- Software Architecture Analysis
- Data protection impact assessment
- Requirement Gathering
- Compliance Requirement Identification
- Requirement Analysis
- Business Process Analysis
- User Interface Analysis
- Security Analysis
- Existing System Analysis
- Data Analysis
- Vulnerability Assessment
- Data Source Analysis

Design

- System Design
- Software Architecture Design
- UI/UX Design
- Compliance-Driven Design
- Security Design
- API Design (for application development)
- Microservice Design (for modern applications)
- Mobile Design (for mobile applications)
- Data Pipeline Design (for data platforms)
- Risk management policy creation
- Incident response plan creation
- Infrastructure Design
- Database Design
- Privacy policy creation
- Terms of service creation
- Security policy creation
- Data retention policy creation

Development

- Refactoring
- **Coding**
- Database Management
- UI/UX Implementation
- Security Implementation
- Reverse engineering
- Data Modeling
- Data Integration
- Data Security Implementation
- Integration
- API Development
- Testing (Unit and Component)
- Data Processing Implementation
- Continuous Integration
- Database Design
- Data Migration
- Data Validation
- Database Implementation

Testing

- Alpha Testing
- Compliance Testing
- Accessibility Compliance
- Defect resolution
- System Testing
- Non-functional Testing
- User Acceptance Testing (UAT)
- Beta Testing
- Mobile App Testing (for mobile projects)
- Data Testing (for data projects)
- Accessibility Testing
- Load and Stress Testing
- Compatibility Testing
- Defect tracking
- Unit Testing
- Integration Testing
- Functional Testing
- Regression Testing
- Automation Testing
- Test data management
- Test reporting

Deployment

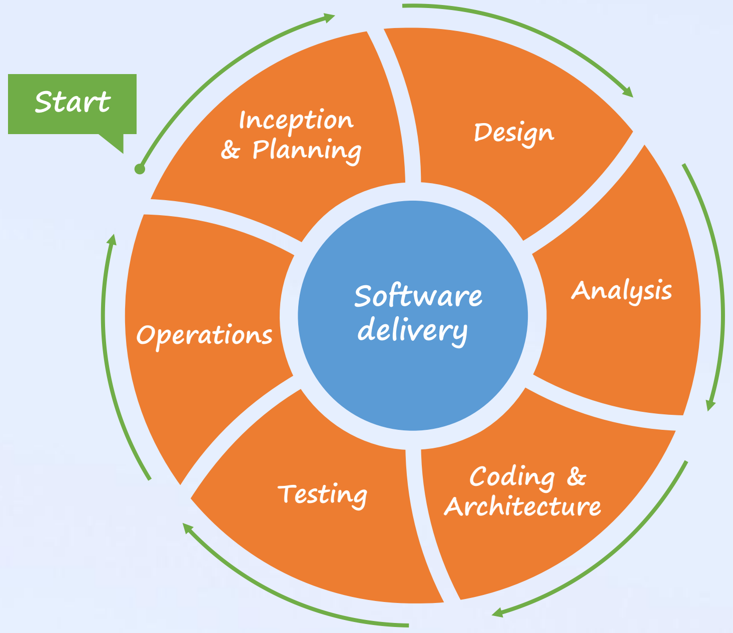
- Change Management
- Cutover Management
- Performance Tuning
- Mobile App Store Submission
- Environment Setup
- Release Preparation
- Data Migration
- Software Installation
- System Monitoring
- Load Balancing
- Compliance monitoring
- Backup and Recovery Setup
- SEO Setup

Maintenance

- Release and patch management
- System audits
- Incident Management
- Security Updates
- Performance Tuning
- Backup and Recovery
- System Documentation
- Compliance Checks
- Decommissioning
- Mobile App Updates
- Data Validation
- Security monitoring
- Performance monitoring
- Security monitoring
- Capacity planning
- Software Upgrades

51CTO WOT

编码以外 AI 仍然可以提供辅助



Access world-class software expertise:

- Expert advice: 100s of clients worldwide
- Technical accelerators: Haiven™, Code Concise, and more
- A structured and strategic method for adopting AI across your teams

Inception & Planning

- Generate solutions to strategic challenges using the "Playing to Win" framework
- Analyze potential future scenarios and their impact on your business / organization
- Rapidly generate conceptual solutions to user pain points
- Derive a detailed backlog from high level scope

Design

- Integrate AI-powered visual design tools into your workflow
- Ideate solutions with "How Might We" Design Thinking method
- Segment users into personas and prepare research questions

Analysis

- Break down epics into user stories with interactive AI chat
- Break down visual user journeys into stories through AI diagram analysis
- Create detailed user stories, including "given, when, then" acceptance criteria

Coding

- Choose the right IDE-based coding assistant, avoid pitfalls, and enable developers to master responsible code generation
- Create automatic lists of developer tasks required to implement given a stories
- Explain technical diagrams and rapidly onboard new

Architecture

- Analyze architecture diagrams and discuss strengths and weaknesses with an AI architect
- Rapidly write comprehensive architecture decision records, using AI to discover gaps in your thinking
- Perform "red team" brainstorming to uncover security threats and other failure scenarios for your application

Testing

- Generate consumer-driven contract tests for RESTful APIs
- Convert "given, when, then" scenarios into automated acceptance tests
- Rapidly generate test data, avoiding the risks of cloning production data
- Use natural language to explore test databases
- Speed time to resolution and improve code quality while gaining efficiency and reducing YoY "run" cost.

Operations

- Automate and optimize tasks and processes to speed time to resolution and improve code quality while gaining efficiency and reducing YoY "run" cost.
- Use real-time insights from AI data to inform continuous application improvements over time.

2024 智启新纪 慧创万物

51CTO WOT

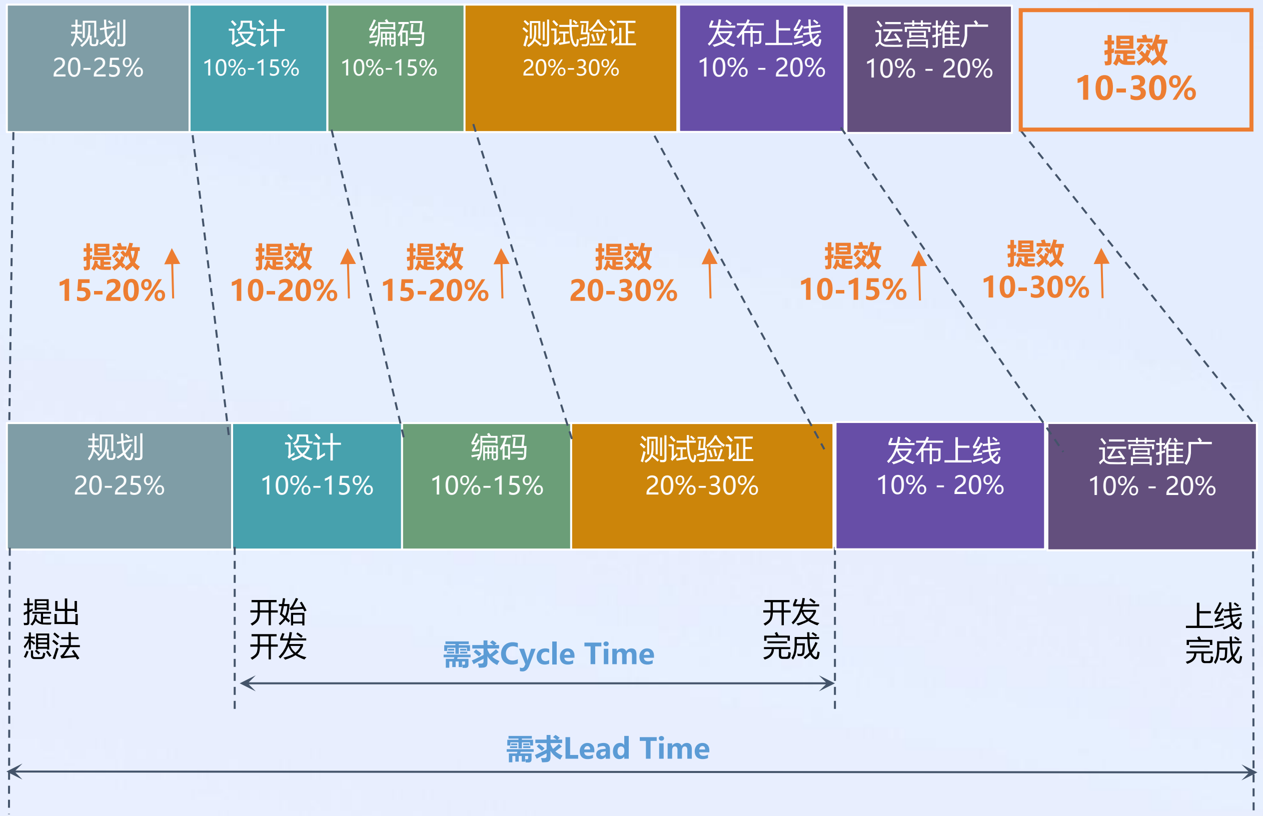
扎实的工程实践仍然很重要，甚至变得更重要

- AI 会增强人的技能，但不会取代人。
- AI 不会区分代码的好坏或流程的好坏。不要用 AI 增强原本就不应该做的事情。
- 软件交付仍将是一项团队协作工作，AI 需要提升团队的整体能力，而不仅仅是个人的吞吐量。
- AI 增强需要转变观念。需要将人工智能视为人而不是软件：大型语言模型不像其他软件那样可靠、可预测和可解释。
- 知识质量仍然很重要：大语言模型和人一样，错误的或可读性差的文档和代码一样会让大语言模型感到困惑。

AI 带来了新的工具，这些工具以及工作方式和正确的团队组织结构，对于缩短产品上市时间、提高质量和持续保持团队士气至关重要。

扎实的工程实践和高质量的知识才能加速 AI 提效

Thoughtworks研究预测：通过AI来增强各个环节，让AI替代各个环节中个角色的重复、繁琐性工作任务，并通过AI降低信息获取、任务切换、认知负载等协作摩擦，预测团队效能可能提升**10-30%。**

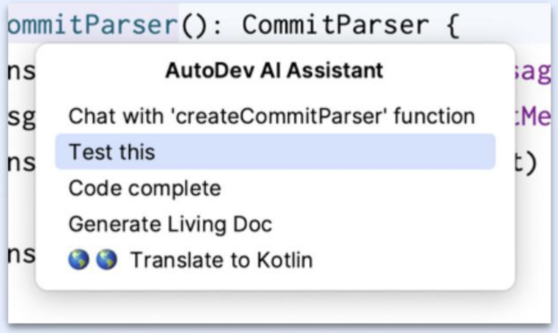


以X组织为例，按照当前各环节在整个leadtime占比，预测整体Leadtime乐观情况下可缩短28.5%；悲观情况下可以缩短12.5%

阶段	当前Lead Time占比	预计环节提效	预估对组织整体效能影响
规划	20%	15-20%	**3-4%**
设计	15%	10-30%	**1.5-4.5%**
编码	15%	15-30%	**2-4.5%**
测试验证	20%	20-40%	**4-8%**
发布上线	10%	10-15%	**1-1.5%**
推广运营	20%	10-30%	**2-6%**
合计	Lead Time缩短 **12.5% ~ 28.5%**		

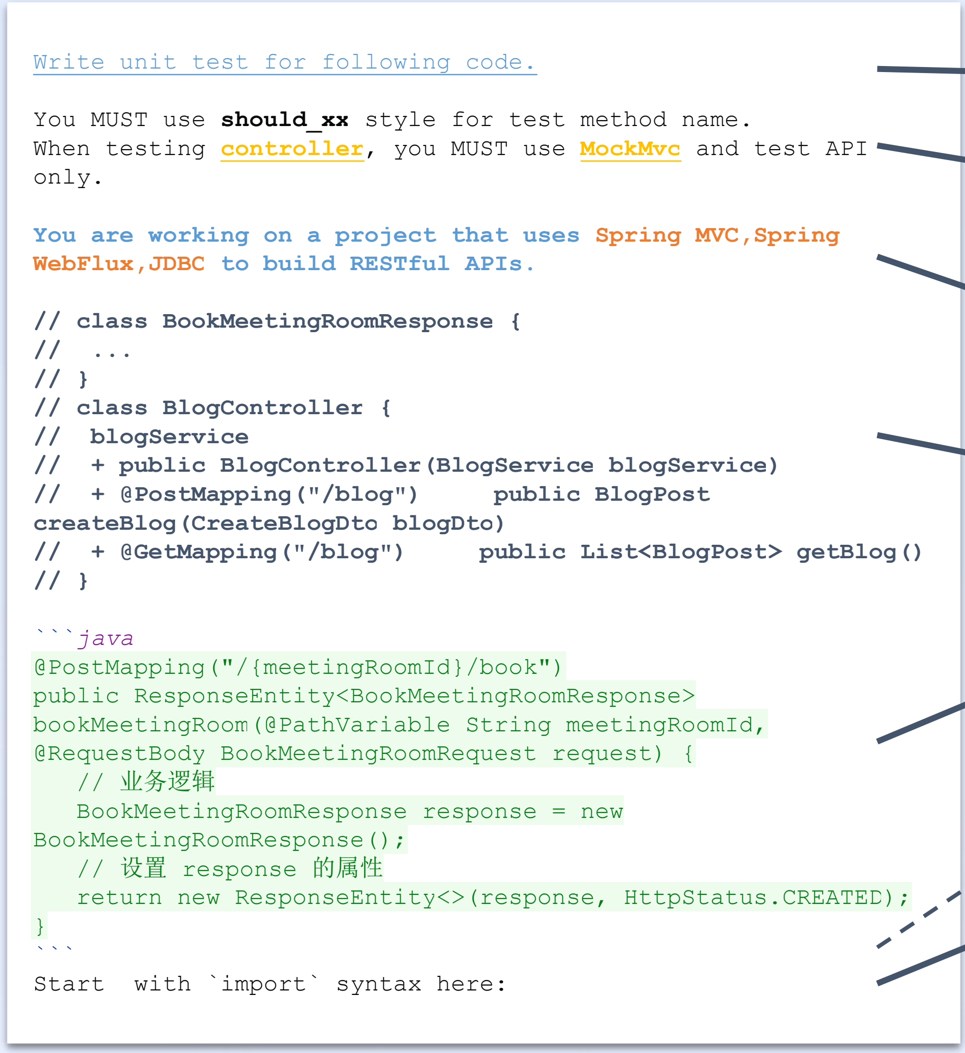
51CTO WOT

在编码场景中加速 AI 提效 —— AutoDev



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)

业务上下文
分层/单测模板

引导词

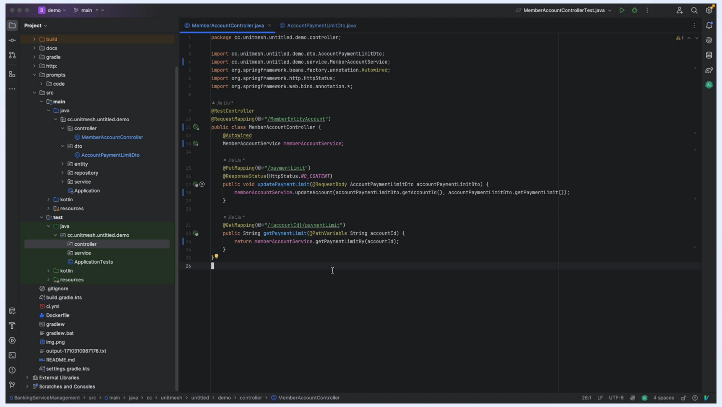
智启新纪 慧创万物 2024

在编码场景中加速 AI 提效 —— AutoDev

不做任何提示词定制

生成测试代码：**38行**
“幻觉”：**4处**
生成评判：

1. 正确使用 MockMVC
2. **未能正确准备测试数据**
3. **未能调用正确的URL**



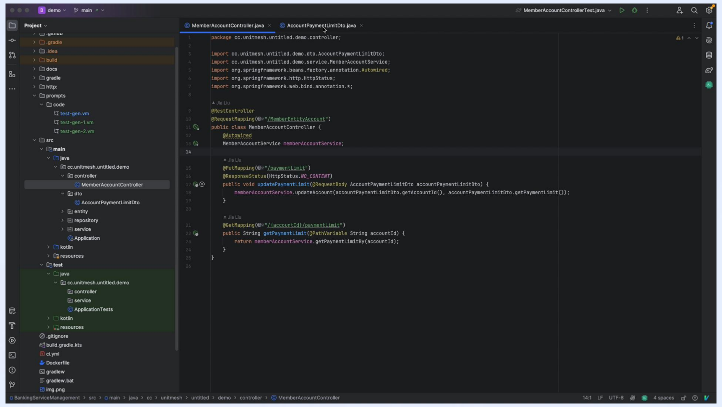
MemberAccountController.java　AccountPaymentLimitDto.java

```java
package cc.unitmesh.untitled.demo.controller;

import cc.unitmesh.untitled.demo.dto.AccountPaymentLimitDto;
import cc.unitmesh.untitled.demo.service.MemberAccountService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/MemberEntityAccount")
public class MemberAccountController {
    @Autowired
    MemberAccountService memberAccountService;

    @PutMapping("/paymentLimit")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updatePaymentLimit(@RequestBody AccountPaymentLimitDto accountPaymentLimitDto) {
        memberAccountService.updateAccount(accountPaymentLimitDto.getAccountId(), accountPaymentLimitDto.getPaymentLimit());
    }

    @GetMapping("/{accountId}/paymentLimit")
    public String getPaymentLimit(@PathVariable String accountId) {
        return memberAccountService.getPaymentLimitBy(accountId);
    }
}
```

增加最佳实践和关联代码检索后

生成测试代码：**50行**
“幻觉”：**1处**
生成评判：

1. 正确使用 MockMVC
2. **正确使用已有函数准备测试数据**
3. **正确调用待测URL**



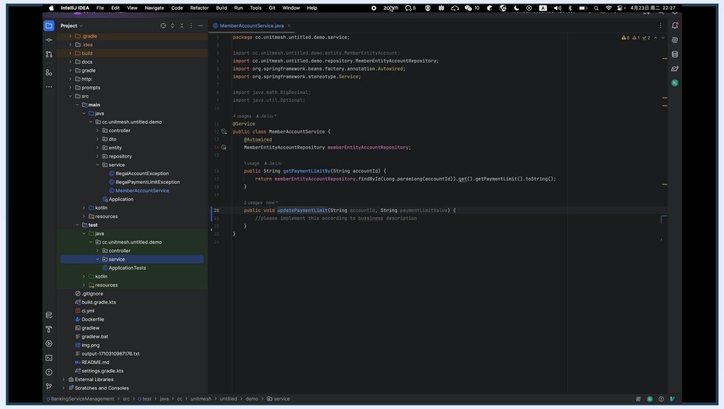
MemberAccountController.java　AccountPaymentLimitDto.java

```java
package cc.unitmesh.untitled.demo.controller;

import cc.unitmesh.untitled.demo.dto.AccountPaymentLimitDto;
import cc.unitmesh.untitled.demo.service.MemberAccountService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/MemberEntityAccount")
public class MemberAccountController {
    @Autowired
    MemberAccountService memberAccountService;

    @PutMapping("/paymentLimit")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updatePaymentLimit(@RequestBody AccountPaymentLimitDto accountPaymentLimitDto) {
        memberAccountService.updateAccount(accountPaymentLimitDto.getAccountId(), accountPaymentLimitDto.getPaymentLimit());
    }

    @GetMapping("/{accountId}/paymentLimit")
    public String getPaymentLimit(@PathVariable String accountId) {
        return memberAccountService.getPaymentLimitBy(accountId);
    }
}
```

增加业务知识后

生成测试代码：**70行**
“幻觉”：**1 处**
生成代评判：

1. 正确使用Mocito
2. **生成多条符合需求的测试case**
3. **测试case划分合理、描述符合业务用语**



MemberAccountService.java

```java
package cc.unitmesh.untitled.demo.service;

import cc.unitmesh.untitled.demo.entity.MemberEntityAccount;
import cc.unitmesh.untitled.demo.repository.MemberEntityAccountRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.math.BigDecimal;
import java.util.Optional;

@Service
public class MemberAccountService {
    @Autowired
    MemberEntityAccountRepository memberEntityAccountRepository;

    public String getPaymentLimitBy(String accountId) {
        return memberEntityAccountRepository.findById(Long.parseLong(accountId)).get().getPaymentLimit().toString();
    }

    public void updatePaymentLimit(String accountId, String paymentLimitValue) {
        //please implement this according to bussiness description
    }
}
```

DEMO：生成基于 SpringBoot 框架实现的服务 API 集成测试，选中目标测试方法执行生成测试...

智启新纪
慧创万物

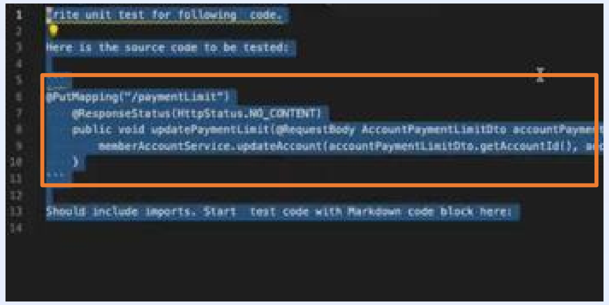
在编码场景中加速 AI 提效 —— AutoDev

不做任何提示词定制

生成测试代码：38行
“幻觉”：4处

最后发给 LLM 的提示词

仅包含选中的被测方法（sourceCode）



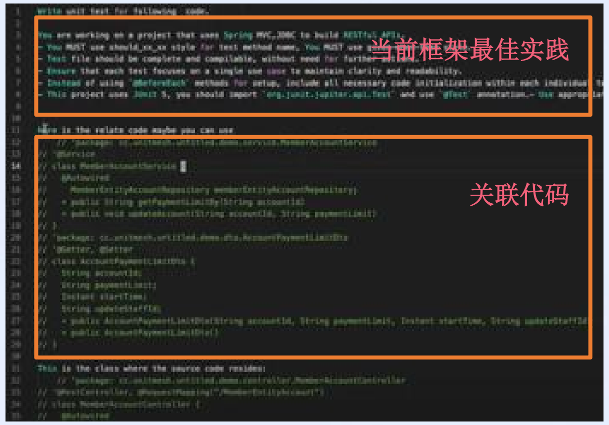
```
Write unit test for following  code.

Here is the source code to be tested:

```
@PutMapping("/paymentLimit")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void updatePaymentLimit(@RequestBody AccountPaymentLimitDto accountPayme
        memberAccountService.updateAccount(accountPaymentLimitDto.getAccountId(), ac
    }
```

Should include imports. Start test code with Markdown code block here:
```

增加最佳实践和关联代码检索后

生成测试代码：50行
“幻觉”：1处

最后发给 LLM 的提示词

1. 包含基于当前使用的 SpringBoot 框架的集成测试最佳实践指导
2. 基于SpringBoot 框架分析出被测方法的关联代码（Service 和 DTO）
3. 被测方法所在类的上下文（经过”压缩”）



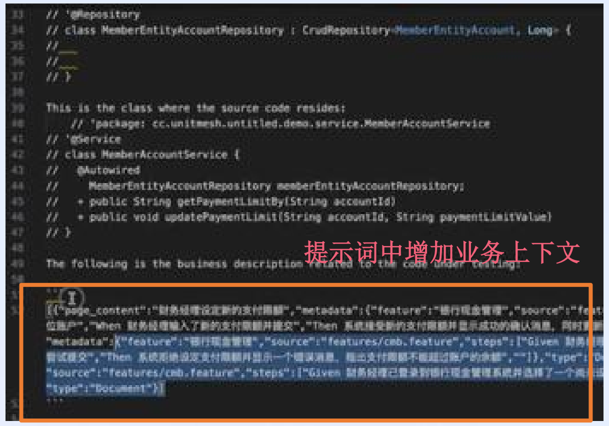
```
Write unit test for following  code.

You are working on a project that uses Spring MVC,JDBC to build RESTful APIs.
- You MUST use should_xx_xx style for test method name, You MUST use 
- Test file should be complete and compilable, without need for further
- Ensure that each test focuses on a single use case to maintain clarity and readability.
- Instead of using `@BeforeEach` methods for setup, include all necessary code initialization within each individual te
- This project uses JUnit 5, you should import `org.junit.jupiter.api.Test` and use `@Test` annotation.- Use appropria

Here is the relate code maybe you can use
    // 'package: cc.unitmesh.untitled.demo.service.MemberAccountService
// '@Service
// class MemberAccountService {
//   @Autowired
//     MemberEntityAccountRepository memberEntityAccountRepository;
//   + public String getPaymentLimitBy(String accountId)
//   + public void updateAccount(String accountId, String paymentLimit)
// }
// 'package: cc.unitmesh.untitled.demo.dto.AccountPaymentLimitDto
// '@Getter, @Setter
// class AccountPaymentLimitDto {
//   String accountId;
//   String paymentLimit;
//   Instant startTime;
//   String updateStaffId;
//   + public AccountPaymentLimitDto(String accountId, String paymentLimit, Instant startTime, String updateStaffId
//   + public AccountPaymentLimitDto()
// }

This is the class where the source code resides:
    // 'package: cc.unitmesh.untitled.demo.controller.MemberAccountController
// '@RestController, @RequestMapping("/MemberEntityAccount")
// class MemberAccountController {
//   @Autowired
```

增加业务知识后

生成测试代码：70行
“幻觉”：1 处

最后发给 LLM 的提示词

1. 包含基于当前使用的 SpringBoot 框架的集成测试最佳实践指导
2. 基于SpringBoot 框架分析出被测方法的关联代码（Service 和 DTO）
3. 被测方法所在类的上下文（经过”压缩”）
4. 与被测代码强相关的业务上下文



```
// '@Repository
// class MemberEntityAccountRepository : CrudRepository<MemberEntityAccount, Long> {
//
//
// }

This is the class where the source code resides:
    // 'package: cc.unitmesh.untitled.demo.service.MemberAccountService
// '@Service
// class MemberAccountService {
//   @Autowired
//     MemberEntityAccountRepository memberEntityAccountRepository;
//   + public String getPaymentLimitBy(String accountId)
//   + public void updatePaymentLimit(String accountId, String paymentLimitValue)
// }

The following is the business description related to the code under testing:

```
[{"page_content":"[illegible]","metadata":{"feature":"[illegible]","source":"feat
[illegible]","When [illegible]","Then [illegible]
"metadata":{"feature":"[illegible]","source":"features/cmb.feature","steps":["Given [illegible]
[illegible]","Then [illegible]",""]},"type":"Do
"source":"features/cmb.feature","steps":["Given [illegible]
"type":"Document"}]
```
```

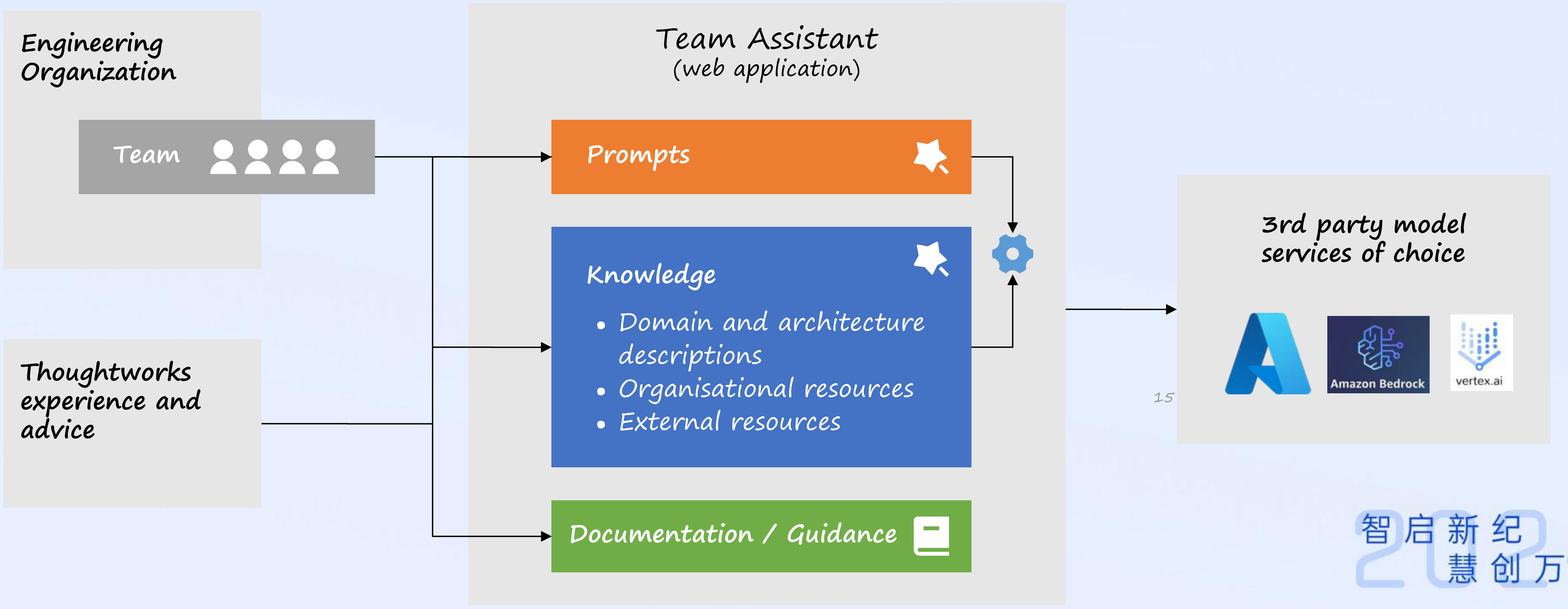
在软件交付团队和过程中加速 AI 提效 —— Haiven™

Knowledge Amplification

Thoughtworks recommended practices "just in time"

Low cost of experimentation

Generative AI and prompt upskilling



在软件交付团队和过程中加速 AI 提效 —— Haiven™

智启新纪
慧创万物

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

https://github.com/unit-mesh/auto-dev-vscode

https://github.com/tw-haiven/haiven

51CTO WOT

谢谢观看

THANKS

智启新纪
慧创万物
2024