AI

From Spec to System

从需求到代码

AI软件工厂——AI时代自动化软件开发的实践与思考

你描述需求，AI 产出整个系统 · AI-Ready® 需求建模器 × AI-Ready® 软件工厂

演讲人：杭州迅速智能CEO — 熊继斌

AI 只解决了软件工程的中间环节

需求阶段

需求澄清 · 业务建模 · 领域设计

仍依赖人的经验与协作

编码阶段

代码补全 → Agent 自主编码 → 仓库级重构

AI 工具集中爆发的环节

交付阶段

集成测试 · 部署验证 · 持续迭代

同样依赖人

我们问自己的问题

能不能让 AI 从需求阶段就介入，一直贯穿到最终的可运行系统？

两大系统：一条全自动链路

01 AI-Ready® 需求建模器

读懂需求

把模糊的自然语言，翻译为精确、无歧义、完备的结构化 DDD 领域模型。

02 AI-Ready® 软件工厂

写出系统

把领域模型拆解为开发任务，多 Agent 并行编排，产出可运行的软件系统。

01 为什么需求建模是第一步

需求到代码的鸿沟，是结构化表达问题

一个真实的故事：300 份文档，两个月的“人肉编译器”

300+

份 PRD 文档

PDF 扫描件 / Word / Visio，结构术语各异

2

名资深产品经理

4－6 周通读，2 周逐项对齐术语

2

周架构师建模

术语对齐之后才能开始领域建模

2

个月只为“理解需求”

两三个高薪人力，慢、贵、不可复现

鸿沟不在写代码，而在需求的结构化表达与多方对齐，传统上由一群资深人员靠开会和手工标注完成的“人肉编译器”。

—— 需求建模器，就是用 AI 替代这层翻译。

原子三要素

1

实体 Entity

有独立身份和生命周期的对象。

访客 · 通行证 · 审批记录

2

关系 Relation

实体之间的语义连接。

访客申请通行证 · 审批关联申请

3

事件 Event

驱动状态变更的业务动作。

提交申请 · 审批通过 · 放弃重提

这三个概念是领域驱动设计（DDD）的基石——我们做的，是让 AI 从自然语言中自动识别和提取它们。

三段式建模流水线

① 事件风暴 Event Storming

通读全部文档，提取所有“已发生的业务事实”，不预设架构边界。

② 四色建模 Four-Color

时刻-时段 / 角色 / 描述 / 参与方-地点-物品，构建实体关系与属性框架。

③ 限界上下文 Bounded Context

划分独立领域模块——申请域 / 审批域 / 通行证域 / 核验域。

需求建模器全流程

中间增设“事件追溯”一步检查因果链断点，建模准确率提升约 30%。

六子 Agent 协作架构

SubAgent1

文档解析

多模态输入理解，转为统一结构化表示。

SubAgent2

事件识别

提取业务事件，建立时间线与因果链。

SubAgent3

实体建模

识别实体与属性，建立四色模型。

SubAgent4

聚合设计

聚合成聚合根，定义业务规则与不变量。

SubAgent5

领域划分

识别限界上下文，定义接口与依赖。

SubAgent6

原型生成

基于业务场景自动生成可交互原型。

就像一支乐队：子 Agent 是乐手（专长），主 Agent 是指挥（调度），Meta-Spec 就是乐谱。

02 上下文管理

AI 建模的“内存”——对抗上下文断裂

三个关键机制

01

需求版本管理

需求变更时对比版本差异，只对受影响局部重新建模，增量传播到下游。

02

上下文预算

给每个子 Agent 分配“注意力配额”，适度信息屏蔽，反而减少幻觉、提升质量。

03

Meta-Spec

建模的规格语言：用结构化规范替代提示词工程，可验证、可追踪、可复现。

Meta-Spec：规格驱动，数据说话

传统：提示词工程

告诉 AI “你应该怎么做”。
不可验证、不可追踪、不可复现；
模型一升级，效果就变。

VS

Meta-Spec：结构化 Schema

告诉 AI “你要产出什么”。
事件字段数、关系图格式、上下文接口，
全部以严格 Schema 约束。

Schema 匹配度

跨 Agent 一致性

产物完整性

为什么是 DDD 建模

领域驱动设计（Domain-Driven Design）

先把业务领域本身建清楚，再谈技术实现——天然区分领域逻辑与技术细节。

A

结构化、可枚举、可验证

每类产物都有明确定义与约束，AI 在确定骨架里填充，而非自由发挥。

B

与代码结构一一对应

正因这种对应，软件工厂才能把模型机械地拆解成开发任务。

建模产物 → 软件资产

领域事件	→	状态变更接口 / API
实体 / 值对象	→	数据模型 / 表结构
聚合根	→	业务规则 / 事务边界
限界上下文	→	独立开发模块 / 微服务

DDD 不只是一种建模风格——它是连接业务语言与代码结构的那座桥。

03 软件工厂

从 DDD 模型到可运行代码的自动化

模型拆解：从领域模型到任务工作流

拆解产物（DeepSeek-V4-Pro）

- 按限界上下文划分的开发模块
- 实现任务卡片（Issue）+ 工时、优先级、标签
- 任务依赖关系 + Mermaid 依赖拓扑图

BLOCKERS 机制

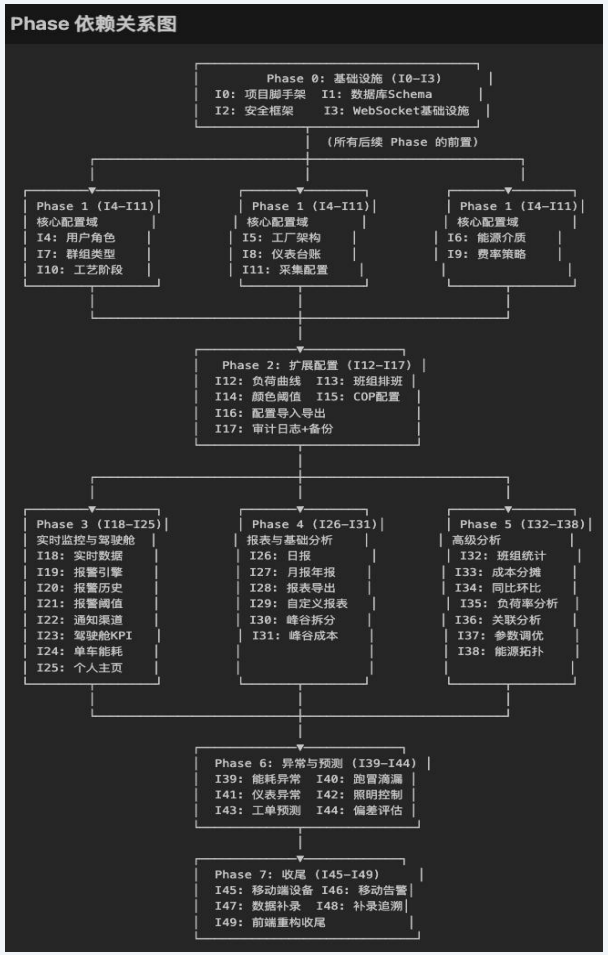
ReadyPlane 限制一个 Issue 只能有一个父节点。

我们在描述中嵌入 <!-- BLOCKERS:I0,I2,B1 --> 注释，由编排引擎解析，建立真正的多源阻塞关系。

```mermaid
gantt
    title 50 Issues 并行执行甘特图 (3-5人团队)
    dateFormat  YYYY-MM-DD
    axisFormat  W%W

    section Phase 0 基础设施
    I0 脚手架          :p00, 2026-06-22, 2d
    I1 数据库          :p01, after p00, 3d
    I2 安全框架        :p02, after p01, 3d
    I3 WebSocket       :p03, after p00, 2d

    section Phase 1 核心配置
    I4 用户角色        :p10, after p02, 2d
    I5 工厂架构        :p11, after p01, 2d
    I6 能源介质        :p12, after p01, 1d
    I7 群组类型        :p13, after p01, 1d
    I8 仪表台账        :p14, after p11 p12, 3d
    I9 费率策略        :p15, after p12, 2d
    I10 工艺阶段       :p16, after p11, 1d
    I11 采集监控       :p17, after p14, 3d

    section Phase 2 扩展配置
    I12 负荷曲线       :p20, after p12 p16, 2d
    I13 班组排班       :p21, after p01, 1d
    I14 颜色阈值       :p22, after p01, 1d
    I15 COP配置        :p23, after p01, 1d
    I16 导入导出       :p24, after p14 p15 p20 p21, 2d
    I17 审计备份       :p25, after p10, 2d

    section Phase 3 实时监控
    I18 实时数据       :p30, after p14 p03, 3d
    I19 报警引擎       :p31, after p30, 3d
    I20 报警历史       :p32, after p31, 2d
    I21 报警阈值       :p33, after p31, 2d
    I22 通知渠道       :p34, after p33, 2d
    I23 驾驶舱KPI      :p35, after p30, 3d
    I24 单车能耗       :p36, after p30, 2d
    I25 个人主页       :p37, after p30, 2d

    section Phase 4 报表
    I26 日报           :p40, after p30, 2d
    I27 月报年报       :p41, after p30, 2d
    I28 报表导出       :p42, after p40 p41, 2d
    I29 自定义报表     :p43, after p40, 2d
    I30 分时拆分       :p44, after p30 p15, 2d
    I31 峰谷成本       :p45, after p44, 2d

    section Phase 5 高级分析
    I32 班组统计       :p50, after p30 p21, 2d
    I33 成本分摊       :p51, after p50, 2d
    I34 同比环比       :p52, after p30, 2d
    I35 负荷率         :p53, after p30 p20, 2d
    I36 关联分析       :p54, after p30, 2d
    I37 参数调优       :p55, after p54, 2d
    I38 能源拓扑       :p56, after p30, 3d

    section Phase 6 异常预测
    I39 能耗异常       :p60, after p30 p22 p16, 2d
    I40 跑冒滴漏       :p61, after p30 p22, 2d
    I41 仪表异常       :p62, after p30 p22, 2d
    I42 照明控制       :p63, after p14, 1d
    I43 工单预测       :p64, after p30, 2d
    I44 偏差评估       :p65, after p64, 2d

    section Phase 7 收尾
    I45 移动设备       :p70, after p30, 2d
    I46 移动告警       :p71, after p31 p70, 1d
    I47 数据补录       :p72, after p14, 2d
    I48 补录追溯       :p73, after p72, 1d
    I49 前端收尾       :p74, after p35 p37 p40 p56 p70 p71 p72, 3d
```

> **符号说明**：`⚡` = 实时数据核心节点 (I18)，是 Phase 3-7 的关键枢纽

mermaid



任务依赖关系

三引擎调度体系

Planner 任务定义层

基于开源二开的任务看板。Issue 直写 PostgreSQL 批量插入，含文件路径、函数签名、验收标准、阻塞标签。

Orchestrator 编排调度层

Elixir 引擎，每 5 秒轮询，按优先级与阻塞排序，生成 WORKFLOW.md 启动 Agent。

Coder 代码执行层

无状态原子执行单元，隔离工作空间内读写代码、跑测试、git push，完成即退出。

maybe_unblock_downstream：Agent 完成 Issue 后，Orchestrator自动检查并解锁下游任务，让多 Agent 并行开发真正成为可能。

端到端实战：校园访客系统全链路跑通

1. 需求文档输入
2. 建模器生成 DDD 模型
3. 拆解任务卡
4. 批量写入 ReadyPlane
5. ReadySymphony 自动调度
6. 浏览器可访问可运行系统

13	4	20	41
业务事件	限界上下文	Issue 任务卡	条阻塞关系

人工参与的环节只有两个：初始需求的描述和最终结果的验收，中间全部自动化。

04 全链路的最后一公里

从跑通到工业级可用

正在推进的三个方向

01

增量建模 · DocMap

持续建模：文档任何变更都触发受影响局部的重建。DocMap 把数百份文档建成层级索引网络，快速定位影响范围。

02

DDD 建模质量优化

建模是质量杠杆点。插入“事件追溯”检查因果链完整性（准确率 +30%）；“Skill 重构”固化每个子 Agent 的输入契约与输出规范，消除产物漂移。

03

Canonical Business Story

把用户故事升级为 CBS，补全前置 / 后置条件、业务规则、异常路径、验收测试点——由输出要求反推 Schema。

05 从工具到范式

软件生产方式的结构性变化

钱在哪？AI 产业链的三个层级

LAYER 1

资源层 · Token 生产

芯片、算力中心、电力。

NVIDIA 已验证

LAYER 2

模型层 · 造部件

解决理解与生成的能力问题。

Anthropic / OpenAI 已验证

LAYER 3

Agent 层 · 造机器

用模型驱动能自主完成任务的智能体。

应用层下半场 我们在这里

单个 Agent 护城河很浅——单个是作坊，把一组 Agent 协作成自动化工厂才是壁垒。
OpenAI AGI 五步（对话 · 推理 · Agent · 创新者 · 组织者），我们处于第三到第四步的过渡区。

需求驱动完整软件工程

不替代开发者，而是重新定义工作方式

从“写代码”到“描述需求、验证结果、持续迭代”——创造力被释放到更高层次的业务理解与系统设计。

Token 经济的下半场：不卖 Token，卖生产线

谢谢大家！