从互联网工程师到 FDE

FDE 转型

不是学出来的，是做出来的

面向想转型的工程师

Shanhe

这门课带你回答一个问题

想转型 FDE，该怎么理解它、判断自己、走通路径?

01 **FDE 是什么**
新瓶装旧酒，第一性原理

02 **为什么是现在**
AI 提升研发效益，价值重心转移

03 **能力内核**
从发现问题到交付结果

04 **谁适合、谁不适合**
闭环思维 vs 流水线思维

05 **怎么转型**
思维转换 · 实操路径 · 培训模式

06 **行动指向**
你接下来可以做什么

Shanhe

FDE 不是新职业，是旧形态被重新看见

旧酒 · 本质

驻场开发
贴近业务、流程梳理
端到端交付

AI 时代

新瓶 · 时代

AI 时代重新被看见
FDE 之名，2010 年已提出
被行业重新讨论

拆开 Palantir 的标准玩法，本质就是我们在大厂早已习惯的软件研发流程。

Shanhe

Palantir vs 互联网 职能划分与映射

Palantir 官方角色	互联网对应角色
Echo · 部署策略师 沉浸工作流、出方案、盯部署	**产品与项目管理** 需求分析 + 产品原型 + 项目管理
Delta · FDE 本尊 单客户端到端交付，服务单一客户	**业务研发** 技术方案实施
Dev · 平台工程师 可复用能力，服务多个客户	**基础平台研发** 通用能力沉淀
Foundry / Ontology · 数据操作系统 数据集成 + 本体语义层	**数据底座 + 领域建模** 统一数据接入 + 业务对象建模

Shanhe

第一性原理：不是全栈，也不是 AI

× 全栈

只是技能组合，不是第一性原理

× ~~AI~~

只是工具，不是第一性原理

↓

第一性原理

贴近业务 + 流程梳理和抽象

2010 年 Palantir 就提出了 FDE，原始分工里没有全栈、也没有 AI——贴近业务才是第一性原理

Shanhe

由第一性原理推出：两件事不重要

推论一 · 手段不重要

第一性原理：帮客户解决问题、拿到结果
AI 落地只是切入点
数据治理、数字化改造、Agent搭建
都是路径，不是结果
切忌“拿着锤子找钉子”

推论二 · 组织形式不重要

按项目收费，预算内交付即可
一个人沟通高效，但能力有短板
按需求拆解、落地、基建分好工
关键是交付结果，不是交付形式

Shanhe

一个核心命题，推出三个推论

AI 大幅提升研发效率

唯一的“因”

推论一 · 价值重心转移

开发不再是瓶颈

贴近业务才是瓶颈

推论二 · 增量机会

启动成本高的事，现在能做了

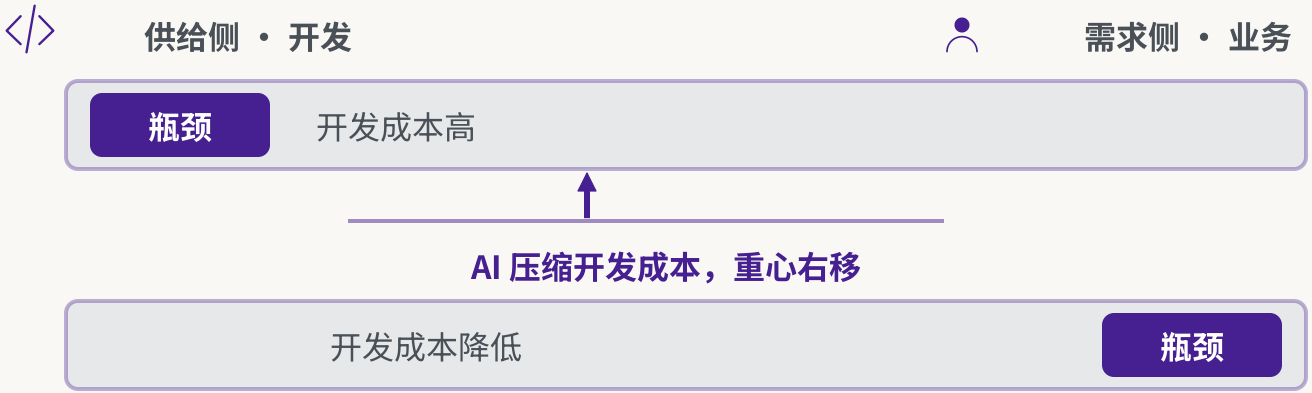
用增量思维看机会

推论三 · 敏捷迭代

几天的事缩到几分钟到半天

试错迭代变得关键

推论一：价值重心向需求侧转移



开发不再是瓶颈，贴近业务才是瓶颈——FDE的第一性原理。

Shanhe

推论二、三：机会变多，周期变短

推论二 · 增量机会

启动成本高 → 现在能做了

传统行业、零散需求
原先启动成本高，现在能做了

推论三 · 敏捷迭代

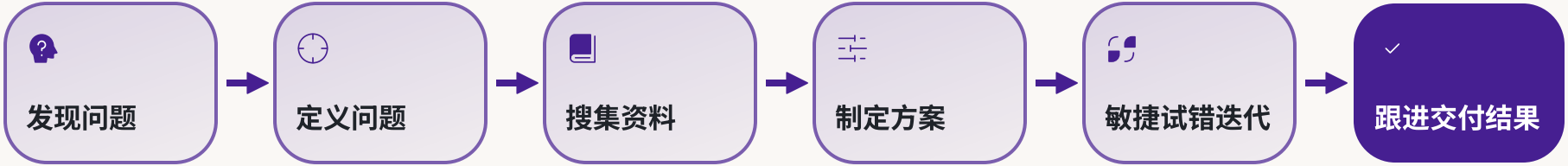
交付周期：几天 → 几分钟/半天

敏捷迭代也是同理
否则 FDE 标榜的「现场快速迭代」行不通

第三章 · 能力内核

所有能力，指向同一件事

从发现一个问题，到交付一个结果



这就是“主动发现问题、制定方案、行动并交付结果”的能力。

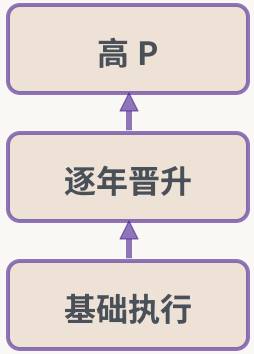
它不是靠学课程学出来的，而是在实践试错中迭代出来的。

Shanhe

这套能力，不是大厂阶梯的专利

大厂的路径假设

从小模块做起，逐年扩 scope



被体系规训，大厂**恰恰更需要转型**

现实

自主学习能力 → **实践试错中迭代**

不等教程，边做边长

强手直接**闭环一个小项目**，再逐渐做大项目

Shanhe

沟通与洞察，是贯穿其中的软素质

认真去听、认真去问，抽象出客户没说出口的事——综合起来，这是一个软素质

4 **洞悉人性**

复杂博弈时，知道何时介入、何时不该介入——该收手就收手、该闭嘴就闭嘴

3 **弦外之音**

听出话外之意，看出客户自己都没发现的需求——客户说不明白需求，是常态

2 **主动服务**

力所能及地帮用户优化体验——用户觉得“希望这么用”，你能看出其实可以更好用

1 **顺畅沟通**

一说别人就明白，一听别人说的就懂

能让人快速信任你 · 在复杂人情世故中找到最优解 · 判断什么事能帮、什么事别说话

Shanhe

闭环思维，是 FDE 的底色

为什么闭环是内核？因为有了它，你才能把上游、下游当成工作伙伴、甚至自己的分身。

对比维度	流水线思维 · 不适合	闭环思维 · 适合
责任边界（总纲） **任务思维 vs 结果思维**	对“环节/任务”负责 完成任务即止，不追效果、不衡量收益	对“最终结果”负责 跟进效果、衡量收益，最终交付结果
对上游 **问题从哪来**	只要方案 照方案执行，不追问	代入上游视角 主动理解真实问题，平等协商
对下游 **结果怎么成**	只提要求 不管实现难度	代入下游视角 体谅难度，平等协商推动

本质上，上游和下游的事，你都能做——所以你才能平视两端。

Shanhe

另一项内核：应对不确定性的能力

闭环必然穿越不确定地带——靠的不是“敢不敢”，而是三件具体能力

01 · 试错迭代

方案与结果之间永远有差距，第一版不可能完美。
不能拿到完整方案就无脑执行——边做边试、打磨修正，直到达成目标。

02 · 应变能力

贴近客户，必然引入方案没覆盖的新输入。
要敏捷地把新输入融合进方案，而不是固守原案。

03 · 取舍能力

不能大而全把所有需求都做进来，否则维护就是灾难。
要能做出正确的取舍和判断。

试错、应变、取舍——合起来，才撑得起一条闭环。

Shanhe

不同职能，各自的转型切入点

原职能	巨婴行为（流水线）	贴近 FDE 的特质（闭环）	转型建议
产品	出方案，扔给研发 “怎么实现我不管” “我的需求不能改”	挖痛点、给方案 能顺畅地和研发沟通	Echo 方向 做需求场景的整理和抽象 力所能及地学习技术基础知识
业务研发	只翻译产品文档 不追问客户原始问题 “PRD就是这么写的” “PRD没写这个”	多问一句—— “原始场景是什么？” “客户是怎么用的？” “达成结果了吗？”	Delta方向 技术全栈化 能够自己从原始需求出方案 从项目中沉淀领域知识和技术基建
基础研发	只沉迷技术框架和原理 容易技术自嗨	真的关心： “技术被用在了什么地方？”	Dev方向 关注原始需求场景和抽象过程 最好跟着Echo跑现场

核心逻辑不变：完成软件系统的全链路交付；AI = 学习助力 + 实施提速 → 敏捷迭代变关键

Shanhe

学一堆课，不等于转型 FDE

预加载 · 误区

- Git 与命令行
- 编程语言基础
- 数据库
- 前端框架
- 后端服务
- 全栈开发
- DevOps 部署
- AI 通识
- 机器学习基础
- Agent 开发

需要先学一堆课程，才能开始干?

懒加载 · 正解

真的积极主动解决过一个问题

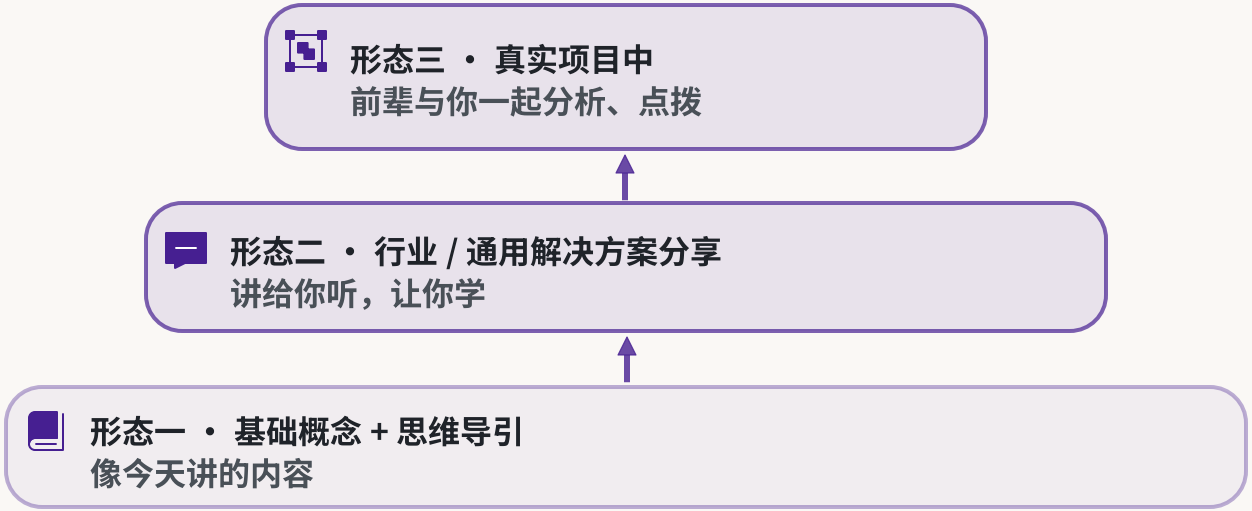
遇到什么问题，马上学什么
不是不学左边的内容，而是按需补

评估标准： 看你如何分析问题 → 拆解问题 → 定义目标收益 → 制定方案 → 实施纠错 → 交付结果

Shanhe

FDE 培训，不做“喂到嘴里的”

实操靠资深的人带教——陪跑、拆解，让你自己悟。培训分三种形态：



如果你需要“喂到嘴里的”课程——那可能恰恰说明，你不适合这行。

Shanhe

下一步，你能做什么？

01

照镜子

对照第四章，判断自己是闭环思维，还是流水线思维

02

选路径

按背景选入口：Echo / Delta / Dev

03

主动动手

不靠课程，先解决一个自己或身边的真实问题

04

找人带

在真实项目中，让资深前辈陪你分析、点拨

别急着学，先动手解决一个问题

FDE 不是新职业，是已有形态在 AI 时代被重新看见。
用增量思维、闭环能力、全链路交付——
做那个主动发现问题、交付结果的人。