

FDE

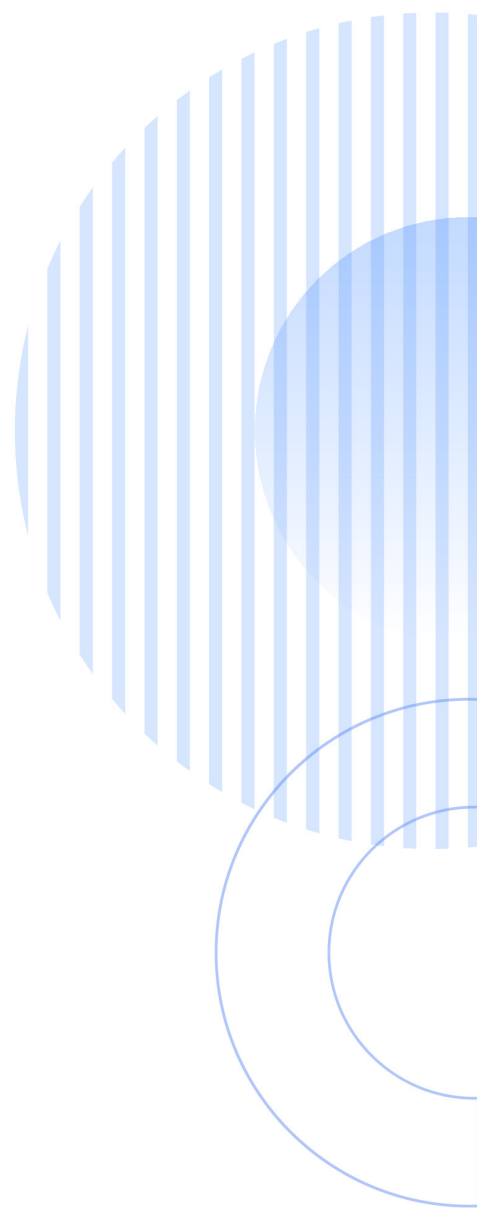
AI时代的落地工程师

白皮书



摘要

大模型把AI能力压缩成廉价的通用服务，但企业级项目大多停留在概念验证阶段——瓶颈不在模型，而在业务语义、数据可得性、系统集成与审计要求构成的“最后一公里”。前线部署工程师（Forward Deployed Engineer, FDE）由此成为AI时代最稀缺的岗位之一。本文界定FDE的定义与内涵外延，辨析其与传统交付工程师、解决方案架构师、算法工程师、平台工程师及数据科学家的边界，提出涵盖AI落地观、基础素质、行业知识与专业技能的培养路径，给出Echo、Delta、Dev三类角色的团队配置与协作机制，并结合金融测试、金融开发、医疗与能源四个案例，说明FDE如何在现场定义问题、组织数据与知识、形成可审计的解决方案并沉淀可复用资产。FDE不是AI的替代者，而是AI与业务之间的翻译者、约束者和兑现者。



目录

	摘要	
	引言	
第一章	为什么AI时代需要FDE	01
第二章	什么是FDE	02
第三章	FDE与相关岗位对比	04
	3.1 FDE vs 传统交付工程师	04
	3.2 FDE vs 解决方案架构师	04
	3.3 FDE vs AI算法工程师和平台工程师	05
	3.4 FDE vs 数据科学家	05
第四章	如何培养FDE	06
	4.1 树立正确的AI落地观	06
	4.2 提升基础素质	07
	4.3 快速学习行业知识	07
	4.4 掌握AI专业技能	08
第五章	FDE的团队配置与协作	09
	5.1 为什么不是超级个体	09
	5.2 按职责与位置分工的三类角色	09
	5.3 三类角色如何协作	10
第六章	实战案例	11
	6.1 案例一 金融测试 FDE	11
	6.2 案例二 金融开发 FDE	13
	6.3 案例三 医疗行业 FDE	15
	6.4 案例四 能源行业 FDE	18
	6.5 四类FDE的对比	20
	结束语	

引言

信息时代的到来催生了海量的数据，智能时代的到来则催生了海量的“能力”。大模型把过去需要专门团队数年攻关的自然语言理解、代码生成、图像识别等能力，压缩成了一次接口调用即可获得的通用服务。能力本身正在迅速贬值，甚至趋于免费。

但企业界的体感与技术的乐观并不一致。过去两年，绝大多数企业级AI项目停留在概念验证阶段，真正进入生产环境、承担核心业务流量并接受审计的屈指可数。多方调研反复指向同一个结论：模型能力不是瓶颈，瓶颈在“最后一公里”——业务语义无法被模型理解，关键数据拿不到，与存量系统集成的成本高企，生成结果无法满足审计要求。

这个落差催生了一个新的岗位：前线部署工程师（Forward Deployed Engineer, FDE）。这一概念最早来自Palantir，该公司自成立起就采用一种与众不同的交付模式——把工程师直接派驻到客户的业务现场，与业务人员坐在一起工作，一边理解业务，一边构建系统，并把现场获得的认知持续反哺回产品。Palantir的壁垒，很大程度上正来自这套“人+产品”混合的交付范式：产品提供可复用的底座，FDE提供无法被复制的业务理解。

这与二十年前数据领域发生的故事高度相似。数据被定义为新的生产要素之后，数据工程师、数据分析师、数据科学家等职业应运而生，其中数据科学家一度被称为“21世纪最性感的工作”。今天，AI正在成为新的通用生产力，也必将造就一批新的职业。FDE是其中最具有代表性、也最稀缺的一个。

到底什么是FDE？他们应该具备哪些素质和技能，如何开展工作，又如何培养这样的人才，至今还缺乏准确的回答和清晰的思路。因此，本文结合一线交付实践，对以上问题进行探讨。

徐立扬

润和软件首席AI科学家、人工智能中心总经理

杨冬瑞

润和软件人工智能中心资深AI架构师

吴昊

润和软件具身智能中心架构专家

詹伶俐

润和软件人工智能中心技术专家

01 为什么AI时代需要FDE

◎为什么要培养 FDE，可以从以下三个方面来分析。

1. 应用场景需求大。

- AI能力要渗透到每一个业务流程，而业务流程的数量远超系统数量。一个银行有数以千计的业务规则、数以万计的交易场景、数以十万计的测试分支。只要有业务的地方，就需要有人把AI能力接进去——这与“只要有数据的地方，就需要有人管理和利用好数据”是同一个逻辑，只是这次原料从数据变成了能力。

2. 能力的价值释放方式特殊，且存在结构性鸿沟。

AI能力与传统的软件产品有本质区别：

- 能力可以被零成本复制，业务语义不能。一份模型权重可以无限分发，但每家企业的口径、流程、术语与合规约束都不一样。同样是“客户”这个对象，在零售条线指自然人，在信贷条线指授信主体，在风控条线指风险敞口单元。模型自身无法消解这种歧义。
- 能力的可用性依赖上下文供给，而上下文长在存量系统里。模型只负责推理，不负责知道。推理所需的事实散落在核心系统、数据仓库、制度文件和资深员工的头脑中，需要有人把它们组织成模型可消

费的形式。

- 模型指标不等于业务指标。通常所说的“模型指标”，更多是参数规模、上下文窗口长度、公开榜单分数、推理时延与吞吐、每百万Token成本这类衡量模型自身能力大小的指标。它们提升一个档次，未必能让交付周期缩短一天。价值必须落到成本、效率、体验、风险这四者中的至少一项，而这个映射关系因行业而异，需要人来建立。
- AI能力进入生产必须接受审计。生产系统要求可追溯、可复现、可回滚，这与生成式模型天然的随机性直接冲突，必须由人设计出确定性的约束层。

3. 人才培养难度大。

- 高校培养的学生大多只学习了算法和理论，缺乏对业务的理解和实战经验；传统行业的IT从业者要掌握大模型与智能体技术，又往往受困于既有的工程习惯。作为一名FDE，除行业知识和专业技能外，还需具备优秀的基础素质和探索精神。两端都稀缺，中间地带便成了人才真空。

02 什么是 FDE

FDE 的定义可以从两个维度来理解，一个是“Forward Deployed”+“Engineer”，另一个是“前线部署工程”+“家”。

第一个维度可以理解为被派往业务最前线的工程师，即把工程能力部署到离业务问题最近的地方、在现场完成问题定义与方案构建的人，这是 FDE 的内涵。第二个维度是指以前线部署为方法论，去研究、探索并解决各类落地问题的人，这是 FDE 的外延。外延的不断扩大驱动内涵不断深化，两个维度合起来才构成完整的诠释。

FDE：以业务问题为出发点，运用行业知识与AI工程技术，把模型、数据、本体、智能体与平台组合成可运行、可审计的解决方案交付到客户生产环境并产生可度量业务价值，同时不断沉淀可复用的行业本体与工程范式、反哺产品能力的人。

前线部署工程：一套以“现场优先”为原则的交付方法论。它不从既有的产品能力出发去寻找适用场景，而是从现场的真实约束出发去定义问题，再反向决定需要什么样的能力组合。其核心假设是：业务问题的

定义权在现场，不在产品文档里。

需要特别说明的是，**AI时代的FDE与Palantir时代的原始定义已经出现了重要差别**。在数据集成时代，FDE的主要工作是把客户分散的数据接进来、建成统一的对象模型，人本质上是“人肉编译器”。而在AI时代，FDE多了一位新的协作者——模型本身。这意味着三件事：FDE的产出物不再只是代码和配置，还包括供给模型的上下文，即本体、约束与评测集；FDE必须设计人机分工，明确哪些环节交给模型、哪些必须由人主导；FDE的工作重心从“写代码”向“定义问题、设计闭环、治理偏差”迁移。

FDE应具备的技能：一位优秀的FDE应当是站在企业级AI平台上，看懂业务问题、组织业务语义、设计智能体与人机闭环，并把这些智能体有机组合起来部署到平台上，处理鲜活数据、产生业务价值、解决行业问题。这句话中蕴含了FDE应当掌握的四大技能；而在这四项技能之外，还有一个贯穿始终的内核——对模型能力边界的清醒认知（如图1所示）。

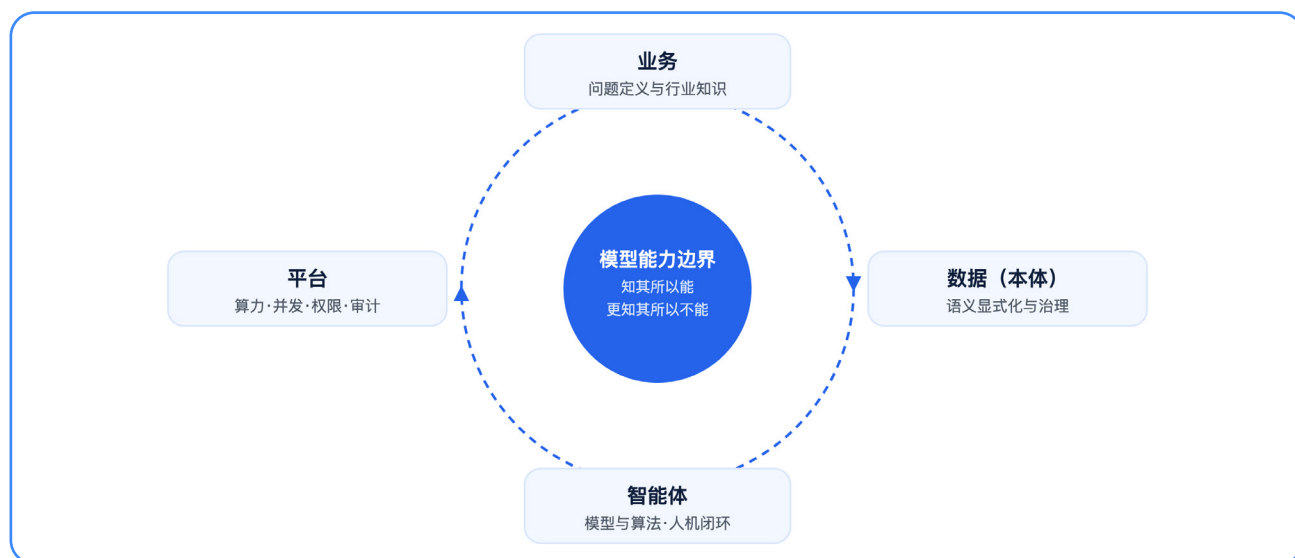


图1 FDE 应掌握的技能：四项技能与一个内核

· **FDE要了解业务问题，知道影响该问题的因素。**比如金融测试中“测试覆盖不足”这一问题，其影响因素包括需求变更频率、接口契约的稳定性、回归用例的维护成本、测试环境的可用性、测试数据的可得性，以及测试人员对业务语义理解的差异。只有知道这些因素，才能决定采集哪些数据、在方案中设定哪些判据。还要了解行业里现有的方法——测试左移、契约测试、精准测试、变异测试——一方面借鉴其思想精髓进行方案设计，另一方面知道其不足，用AI工程手段弥补缺陷。最后还要学习行业的语言，以便与业务专家沟通，让其理解和接纳基于AI的解决方案。听不懂“冲正”“影子账户”“联机交易”“账务总账”，就无法参与真正的方案讨论。

· **FDE要深度理解数据，不仅要了解数据的格式、属性和表面意思，更要理解数据背后反映的深意。**例如一笔交易流水不仅反映了金额和时间，还反映了业务类型、发生渠道、账务处理路径和参与方角色。把大量交易流水融合在一起后，可以洞察一个条线的产品使用规律；使用规律又反映客户的分群结构——高频小额对应日常消费，低频大额对应资金调度，长期静默后突增对应资金归集；而分群结构又影射出对风控策略的影响。这样就能用领域A的数据去解决领域B的问题。

在AI时代，这种“深度理解”有了明确的载体，即本体（Ontology）。本体用对象、属性、关系、动作与约束五类元素，把散落在制度文件、系统文档和专家头脑中的业务语义显式化，成为模型可以消费的上下文。可以说，本体是FDE把业务理解交付给模型的主要形式。没有本体，FDE的业务理解就停留在个人经验里，无法复用、无法传承、无法被机器执行。

· **FDE要精通数据管理、上下文工程、智能体编排、评测与可观测工具，具备AI侧端到端的能力。**具体包括：上下文工程（文档解析、切分策略、稀疏与稠密混合检索、重排、上下文压缩）；智能体编排（任务分解、工具调用、状态管理、失败重试与降级）；评测与回归（黄金集构建、自动评测、影子流量对比）；可观测（链路追踪、成本与延迟监控、输出漂

移告警）；以及传统的数据管理、机器学习与可视化工具。这些能力相互关联，只掌握其中某个环节、缺乏对上下游可行性的考虑，设计的方案就很难落地。

· **FDE要熟练运用企业级AI平台与算力底座。**企业级AI不仅要求模型能力强，还要求并发高、时延稳、权限严、可审计。单机模式下的提示词调试与笔记本验证，无法应对真实世界的生产要求。很多方法在小数据、单机模式下可以工作，到了大规模、高动态的场景下就完全失效。比如朴素检索增强生成（RAG）在小规模语料上效果不错，但当文档规模达到千万级、且处于高动态更新状态时，向量索引的重建代价会急剧上升，召回结果也会出现漂移；更关键的是，它无法满足权限隔离要求——如果先检索后鉴权，模型就已经“看过”了不该看的内容。因此必须改为稀疏与稠密的混合检索，把访问控制过滤放在检索之前而非生成之后，并引入增量索引机制。同理，把全部上下文塞进单一长上下文窗口在本机演示时可行，到了需求因子数以千计、需要多轮工具调用的生产场景，成本、时延以及“中间遗忘”都会让方案失效，必须做任务分解与上下文分层。这些都不是算法问题，而是平台问题。

贯穿始终的内核：认清模型的能力边界，并随模型能力的更新而更新这一认知。

这是FDE全部能力中最容易被低估、也最容易过时的一项。FDE必须充分知道当前大模型能做什么、不能做什么，而且不能停留在“能或不能”的结论上，还要理解背后的原因——知其所以能，更知其所以不能。只有知道了“为什么不能”，才能判断一条走不通的路径究竟属于模型能力问题、上下文供给问题还是方案设计问题，从而决定是等待下一代模型、补充本体与检索，还是改用人机分工。

更关键的是，这个认知每跨过一个模型代际就会被刷新一次：半年前不可行的方案，今天可能一次提示词就能完成；今天靠工程手段勉强补上的短板，明天可能被模型原生解决。因此FDE必须持续跟踪模型能力的变化，并主动把新能力纳入方案设计。对模型能力的认知一旦停止更新，FDE就会退化成只会维护历史方案的运维者。

03

FDE与相关岗位对比

为了能让大家更好地理解FDE这个岗位,我们将其与传统交付工程师、解决方案架构师、AI算法工程师和平台工程师进行对比,几个岗位与FDE的整体差异如表1所示。

3.1 FDE vs 传统交付工程师

传统交付工程师面对的是确定性问题,即问题的定义、可以使用的数据源、需要输出的结果都是确定的,然后根据这些确定信息选择相应的技术方案,实施即可。例如,将一份现存报表从A系统迁移到B系统:字段映射关系清晰、业务口径不变、验收标准明确,工程师按图施工即可。这是一个非常明确的实现问题,输入是需求文档,输出是符合验收标准的系统功能。

但FDE面对的是完全开放的问题,问题没有明确的定义,用什么数据不清楚,输入和输出是什么也不清楚,用什么方案更不清楚,这一切都要靠FDE来分析和定义。以下FDE工作过程中遇到的实际问题样例。

- 例1: 一个业务条线有大量需求因子分散在文档、邮件和会议纪要里,如何把它们变成可执行、可验证的开发任务?
- 例2: 如何在不牺牲质量的前提下,把交付周期压缩一半?
- 例3: 如何让一个从未接触过客户核心系统的小型现场团队,在极短交付窗口内产出可编译、可评审的初版代码?
- 例4: 如何用AI拦住那些“看起来通过、实际是错的”测试用例?

以上问题无法归结到分类、聚类、回归等模型问题上,也不一定是一个单一模型就能解决的,往往需要拆解成很多环节,用一套“组合拳”来解决。因此,FDE不仅要解决完全开放的问题,还需要提供一套完整的端到端解决方案,而传统交付工程师只需要解决确定性问题中的一个环节。

维度	传统交付	FDE 模式
问题性质	确定性问题,需求已定义	完全开放问题,需求需现场定义
驱动方式	文档驱动	业务驱动
输入	需求规格说明书	业务痛点、制度文件、存量系统约束
数据	已知、已就绪	需探查、需治理、常需跨域借用
手段	经验与工程规范	本体建模+智能体编排+平台工程
输出	符合验收标准的系统	可度量的业务结果(成本/效率/体验/风险)
价值度量	按人天与里程碑计量	按业务指标计量

表1 传统交付与 FDE 模式对比

3.2 FDE vs 解决方案架构师

解决方案架构师针对业务问题,根据客户的特定场景需求,将产品和能力进行组合、连接并作定制化的封装,解决客户痛点、为客户创造价值。在以AI为核心的落地项目中,FDE可以充当解决方案架构师的角色,

反之不然——解决方案架构师并不一定有AI工程的基础,很多传统的信息化项目只考虑信息的流转,不涉及数据、本体与模型的构建。要解决实际问题,FDE必须具备解决方案架构师的思维和能力。

3.3 FDE vs AI算法工程师和平台工程师

在实际项目中，FDE需要带领AI算法工程师和平台工程师一起实施方案。算法工程师依照FDE给出的思路，完成模型的细化设计（包括模型的内部结构、输入输出的量化、参数选定以及与其他模型的嵌套组合方式）、模型的训练、测试和发布；平台工程师则负责算力调度、推理服务、数据管道与权限体系的建设。当模型结果与设计期望发生偏差时，算法工程师应告知FDE，与后者一起迭代思路。FDE应不断统筹、协调二者的工作进展，确保方案落地执行。

与算法工程师和平台工程师相比，FDE的工作更加宏观、全面，偏向整体方案的创造和设计，而后两个职位更注重某个环节上深入具体的工作，偏向于执行和实施。为了确保设计方案的可行性，FDE在正式上岗前，必须要有从事算法工程和平台工程的实践经验。

3.4 FDE vs 数据科学家

数据科学家是本文反复参照的对象，也是理解FDE最好的一面镜子。两者都面对开放问题，都不是单一模型所能解决，都需要端到端的能力，但分野同样根本（见表2）。

维度	数据科学家	FDE
核心问题	数据用不起来	AI 能力落不下去
主要原料	数据	模型能力+业务语义(本体)
主要方法	统计分析、机器学习、数据挖掘	本体建模、智能体编排、人机分工设计
交付物	模型、洞察与决策建议	可运行、可审计的生产系统与业务结果
成功标准	模型指标 + 业务侧采纳	业务指标(成本/效率/体验/风险)
工作场所	数据平台, 与业务保持距离	业务现场, 与业务共同定义问题
问题来源	业务提出问题, 数据科学家求解	FDE与业务共同发现问题、定义问题
可复用资产	模型与特征工程	行业本体与工程范式

表2 FDE 与数据科学家对比

数据科学家面对的是“从数据中发现规律”的问题，原料是数据，产出是模型与洞察，其影响力通过业务侧的采纳间接实现。FDE面对的是“把能力接入现场”的问题，原料除了模型能力还有业务语义与存量系统的约束，产出是运行在生产环境、可被审计的系统与流程，其价值直接由业务指标来体现。一句话概括：数据科学家让数据产生洞见，FDE让洞见变成系统；前者解决“看不懂”，后者解决“落不下”。

还有一点值得指出：今天的数据科学家自身也在被 AI 改变——重复性的特征工程与调参正在被自动化，工作重心不断上移到问题定义与价值判断，而这恰恰是FDE的核心地带。从这个意义上说，FDE可以看作数据科学家在AI时代的“近业务”演化分支：两者的能力底座相通，只是FDE把工作场所从数据平台搬到了业务现场，把交付物从洞察换成了结果。

04 如何培养FDE

如图2所示，培养优秀的FDE首先要让其树立正确的AI落地观，并不断提升其四大基础素质；同时快速学习行业知识，并掌握AI、本体、平台三大专业技能。

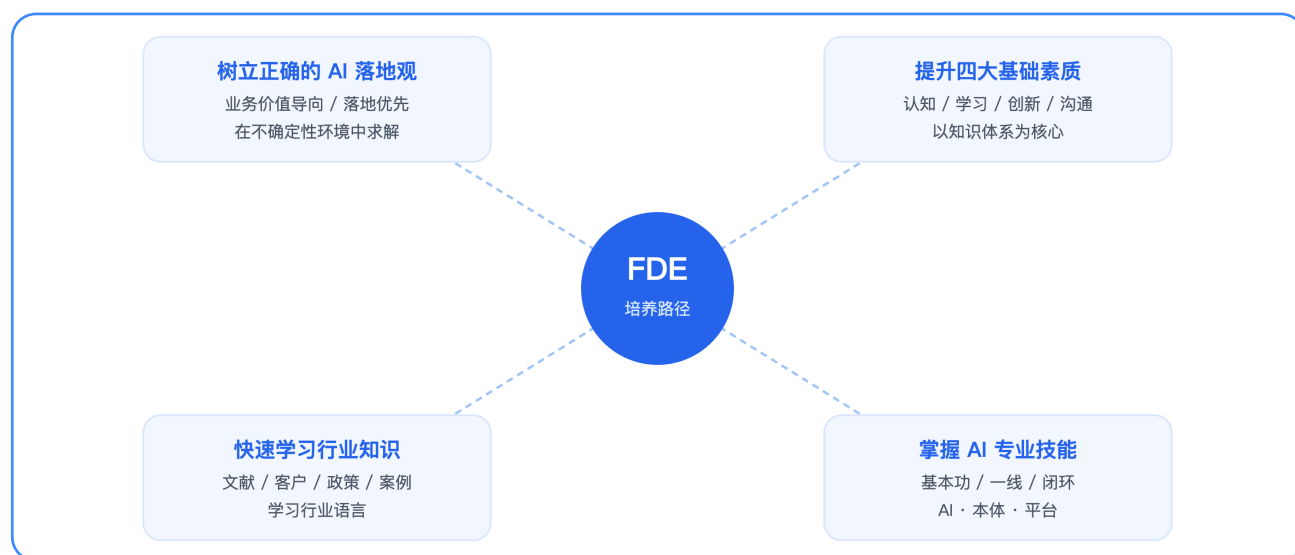


图2 FDE的培养路径

4.1 树立正确的AI落地观

很多人(也包括不少从业者)在面对实际问题时,容易陷入以下误区,这些都是没有树立正确落地观的表现:

- **拿着锤子找钉子。**有了一个模型或者框架,一定要想方设法把它用上去。上了大模型,就一定要做多智能体;装了向量库,就一定要做检索增强。
- **倾向于选择过于复杂的方案来体现技术能力,**生怕方案简单被认为没有难度和价值。三节点工作流能解决的问题,非要设计成带反思、带规划、带自我修正的十节点智能体。
- **抱怨数据质量太差。**数据缺失、权限没开、标注不足、历史数据不完整,因此认为这件事情没法做。
- **认为只有模型部分最有技术含量,**数据清洗、本体建模、评测集构建、系统集成、审计留痕都不重要。
- **初学阶段不打好工程基础,直奔智能体编排。**会用提示词却不会写单元测试,能搭出演示但交付不了生产。
- **逻辑没有理清就把问题交给AI去编码。**需求粗粗整理一下就抛给模型,效果不满足就继续让AI修补,既没有梳理问题的根源,也没有给出明确的处理逻辑,结果全靠模型自由发挥——返工一轮又一轮,问题依旧在那里。

与之相对的,正确的AI落地观可以归结为以下五条原则。

- **业务价值导向。**方案的选择依靠业务驱动——根据问题的特性、数据的实际情况、合规的约束条件来决定,方案的价值由业务成果来体现,是否用到某种模型不是关键。

- **落地优先，事后归纳。**一个工作的难度由待解决问题的复杂度决定，而不由方案的复杂程度决定，能用简单的方法解决复杂问题是非常有价值的工作。面对实际问题，一定从简单方法开始尝试；任何让方案复杂化的付出，都需在结果侧提升性能，否则就是哗众取宠、浪费资源。先把业务价值做出来、让方案真正落地，再回过头对共性的做法做总结与归纳，把它提取为可复用的知识和价值——顺序不能颠倒，先归纳后落地往往只能得到漂亮的框架。
- **能够在不确定性环境中求解。**真实世界最初的数据永远不会让人满意，永远面临数据不足、质量不好、权限受限等一系列问题。如果数据好到可以直接从中看到结果，FDE也就没有存在的必要了。加强深度理解，把领域A的数据用到领域B的问题，才能破解数据不足的难题。用“不确定”+“不确定”得到“确定”的思维来应对不理想的数据，也是解法之一；在AI时代还有一个更直接的手段：用评测集把“感觉效果好”变成“可验证效果好”。
- **链路上的所有环节同等重要，模型并不高人一等。**一条能产生业务结果的链路，至少包含数据获取与治理、业务语义建模、上下文构造、检索与工具调用、判定与校验、系统集成与部署、审计与运营等环节，模型只是其中的一环。任何一环失效，模型能力再强也换不来业务结果；反过来，模型能力提升一个档次，也未必能让交付周期缩短一天。价值必须落到成本、效率、体验、风险这四者中的至少一项，而这个映射关系因行业而异，需要人来建立。因此，资源与注意力应当投向当前真正的瓶颈，而不是按“有没有技术含量”的直觉去分配；把资源堆在最显眼的那一环，通常正是交付失败的原因。
- **FDE诞生于业务战场，而不是开发环境。**不同阶段应该练习不同的技能：先练好程序设计和软件工程

的基本功（版本管理、单元测试、代码评审），再学数据管理与本体建模，然后学数据可视化与结果呈现，之后学模型、上下文工程与检索工程，最后面对客户实战，快速学习行业知识，完善AI侧端到端的能力。FDE无法一步到位，必须一步一个脚印地走出来；但这条路径的终点不在实验室或开发环境里，而在客户的业务现场——一项技能只有在现场被使用过一次、并为业务结果承担过责任，才算真的掌握。

● 4.2 提升基础素质

认知能力、学习能力、创新能力和沟通能力是FDE应该具备的四大重要基础素质，四者相互连接，不断提升、强化自身的知识体系。我们跟客户或者行业专家的交流，本质上就是以知识体系为核心、以四大素质为能力支撑的“交锋”，四大基础素质也在交锋过程中不断历练、提升。沟通能力包括准确听懂和正确表达，把从接收到的语言中提炼的信息传递给认知能力；认知能力基于现有的知识体系判别其深层次含义、提炼其中新的知识；这些知识通过学习能力再沉淀到知识体系中；创新能力则基于已有知识体系不断创新、加工，让知识体系自我完善。四者构成一个闭环：听懂是输入，认知是加工，学习是沉淀，创新是增值。

● 4.3 快速学习行业知识

FDE必须掌握快速学习行业知识的方法论，并将行业知识与数据、本体、模型、平台融合。很多精通模型和算法的人最终无法成为优秀的FDE，最大的瓶颈也在于此。不过在AI时代，学习方式本身已经改变：过去靠人逐篇读文献、逐个请教专家，获取速度受限于个人的阅读带宽与社交半径；今天FDE可以把工程能力用于学习本身——用智能体完成行业知识的规模化获取与初筛，把人的时间集中到判断、验证与收敛上。具体有四条路径。

- **搭建行业知识智能体, 让机器先读一遍。**把业务制度、监管规则、需求规格、接口契约、历史案例与缺陷记录以及公开的同业资料接入一条可持续更新的知识流水线: 解析切分、抽取实体与规则、建立索引、按主题聚类, 再围绕“这个业务领域的对象、动作、约束与异常分支”逐项生成带出处的候选答案。人不再逐篇阅读, 而是审阅候选答案与证据链, 把确认过的部分固化为本体条目。这一步的价值不在“读得多”, 而在于把人的注意力从检索搬到判断上。
 - **用问题清单驱动检索, 而不是靠通读。**先围绕现场要解决的问题列出一份问题清单, 让智能体带着清单去做检索、比对与冲突检测——同一个业务概念在不同文档中的口径是否一致、某条规则有没有例外、某个字段在哪些场景会取空值。冲突点往往正是隐性约束所在, 也正是最值得向专家请教的点。传统路径下这些冲突要靠长期浸泡才能察觉, 现在它们在第一周就会被机器摆到桌面上。
 - **向行业专家验证。**专家的时间应当花在只有人能回答的问题上: 带着智能体整理出的假设与证据去找业务主管或业务骨干, 请他们确认、反驳或修正; 专家的一句话常常能纠正一批本体条目。同时要能用行业的语言表达自己的理解, 并把学到的东西结合自身储备深化后反馈回去, 让专家也有收获——只有这种互动深化的过程, 知识才真正变成自己的。
 - **在实际效果中验证。**这是AI时代新增、也往往更硬的一条路径: 把学到的规则直接放进方案里跑一遍——生成一批用例请业务专家审阅是否覆盖真实场景、让判定通道按规则校验产出是否符合业务预期、用真实数据回归看异常分支有没有被覆盖。知识对不对, 最终由业务效果回答, 而不是由读了多少文档回答; 而且效果验证是可累积的, 每一次执行都是在为这套知识打分。
- 还有一点常被忽略:**学到的知识必须回写成资产——**

沉淀为本体条目、判定规则与评测集。行业知识若只停留在个人理解里, 项目一结束就会清零; 只有回写成资产, 下一个项目的起点才不是零, 这也正是本体复用率的来源。

● 4.4 掌握 AI 专业技能

这里的专业技能包括对业务的深刻理解、对数据的深度把握, 以及设计智能体和使用平台的能力。

- **练好基本功。**程序设计与软件工程的基础课程(数据结构与算法、设计模式、版本控制、单元测试、持续集成), 数据管理、数据挖掘、机器学习与数据可视化课程, 以及AI时代的上下文工程、检索工程、评测工程与智能体编排。以上任何一门课程, 如果想深入学习, 都需要花费数年的时间。因此建议在完成初步学习后, 依托一个具体项目边做边学、逐步深入, 这样印象会更加深刻, 动力也会更足。
- **深入一线。**FDE需要用真实的项目和数据来培育。因此, 掌握算法模型的人一定要尽快去有数据、有业务需求的一线历练, 多跟客户和行业专家沟通学习, 多观察、多动手处理数据, 逐步建立起对数据的深入理解, 熟悉对平台的操作, 建立对模型特性的直观感受。
- **应用闭环。**要经历业务调研、数据探查、本体建模、智能体设计、评测、集成、上线、监控、反馈和反控的全链路, 避免只做其中的模型设计环节。如果前面的数据处理不当, 会让本该有效的模型失效; 如果只参与模型环节, 可能会脱离实际约束; 如果不能将结果有效地呈现给客户, 就无法得到反馈, 模型不能迭代优化。
- **价值体现。**方案一定要针对业务关注的领域, 在成本、效率、用户体验、风险控制中的至少一个方面直接创造价值。在金融场景中, 客户关注的是交付周期、缺陷逃逸率以及合规可审计性, 方案就应该相应地降低成本、提高效率、改善体验或降低风险。避免只做到中间结果, 看不到直接的业务价值。

FDE不是一个孤立的岗位，而是一种可以复制的组织形态。这一点在Palantir的组织设计里体现得最清楚：它没有把前线交付压在一个“既懂业务、又懂模型、还能写代码”的万能角色上，而是拆成Echo、Delta、Dev三类角色。这套做法值得借鉴的地方不在名称，而在于它承认了一个事实——这三件事无法在同一个人身上同时做深。

5.1 为什么不是超级个体

FDE对一个人的要求极高。他既要钻进客户的业务现场把问题挖清楚，判断哪些问题值得解决、边界在哪里、验收标准是什么；又要写出能进生产环境的代码，处理真实的数据、集成、部署与运维约束；还要把现场反复验证过的做法沉淀为平台能力，让下一次交付不必从头开始。

这三件事对应三种不同的能力倾向和工作节奏：问题定义依靠反复沟通与判断，节奏是慢的；现场构建依靠快速交付，节奏是快的；平台沉淀依靠抽象与收敛，

需要的是长期视角。让同一个人长期同时承担，通常的结果是三个维度上都做不深——问题挖得不够透，代码只能应付演示，平台能力则永远停在“下次再说”。而少数确实能胜任的“超级个体”还会带来两个更麻烦的问题：无法复制，以及单点故障；这样的人一走，项目就瘫。

因此，FDE在实践中的落地形态不是一个人，而是一组人。把“要解决什么”“怎么在现场做出来”“哪些能沉淀为平台能力”分别交给不同位置的人，让他们围绕同一个交付结果共同工作，才是可复制的形态。团队规模也随之确定：中小型项目1-3名，其中一个人可能同时站在多个位置上；大型项目8-15名，按位置分工。需要强调的是，位置之间的区别是职责，不是职级——三类角色之间没有上下级关系，只有不同的问题视角。

5.2 按职责与位置分工的三类角色

三类角色分别回答三个不同的问题，其职责、产出与判断标准见表3。

角色	关注的问题	核心职责	关键产出	判断标准
Echo 部署策略师	什么值得解决	从客户使命与一线工作倒推问题，识别真正限制结果的环节，拆解工作流，协调管理者、专业人员与使用者，推动系统被真正采用	问题定义、工作流拆解、优先场景与业务价值判断	能不能找到客户自己没说出口的关键问题，并让它被组织真正用起来
Delta 前线部署工程师	怎么在现场做出来	直接处理数据、代码、系统集成与生产约束，在真实环境中快速构建可演示、可运行的方案并持续迭代	可运行的方案与数据集成路径、现场验证过的技术路线	能不能在不完美的条件下把方案跑通，并对业务指标产生影响
Dev 平台工程师	哪些可以沉淀	把现场反复验证过的做法抽象、泛化、工程化，沉淀为平台的标准能力与资产	可复用的平台能力、本体资产、工程范式与任务模板	同一个问题，下一次交付要不要再从头做一遍

表3 三类角色的职责、产出与判断标准

三者之中，Delta是通常所说的FDE主体。它的工作方式是“一个客户，多种能力”——为了一个客户的某个具体问题，什么工具都可以用；Dev的工作方式恰好相反，是“一个能力，服务多个客户”——一个被抽象出来的能力，要能被后续的许多客户复用。Echo则在两端之间把关：它需要听得懂技术逻辑、清楚不同方案的可行性与代价，但不必精通编码；它的核心能力是从结果倒推行动，问的是“客户想达成什么、什么

在阻碍他们”，而不是“我们的产品能做什么”。

5.3 三类角色如何协作

协作的主线既是一条翻译链，也是一条反馈回路（见图3）：Echo把客户使命翻译成问题，Delta把问题翻译成可运行的方案，Dev把方案翻译成平台能力，平台能力再去支撑下一个问题的解决。



- **结构性张力。**三类角色之间不是简单的接力，而是互补且时常矛盾：Echo从业务使命出发，追求“做正确的事”，会在工程努力脱离业务时踩刹车；Delta从技术可行性出发，追求“正确地做事”，会踩油门推动落地；Delta追求“这个客户现在就要”的速度，而Dev追求“所有客户都能用”的通用性；Dev想的是“平台能做什么”，Echo想的是“客户需要什么”。这些张力不应该被消除——分歧本身就是判断质量的来源，谁在某个问题上更有依据，就听谁的。
- **碎石路到铺装公路。**现场经验回到平台的路径需要被显式设计出来，否则每一次交付都会变成一条新的定制分支。FDE在现场先铺一条“碎石路”——用最直接的做法把眼前的业务问题解决掉；Dev要持续观察这些碎石路，判断哪些是重复出现的模式，把它抽象、泛化、工程化，升级成能服务后续一二十个

客户的“铺装公路”。判断标准很直接：同一个问题，下一次交付还要不要再铺一遍碎石路。

- **与算法和平台工程师的横向接口。**三类角色都需要与AI算法工程师、平台工程师协作，接口是清晰的：FDE回答“业务要什么、判据是什么”，算法工程师回答“模型能不能做到、代价多少”，平台工程师回答“能不能稳定运行、权限与算力如何保障”。三者共同确定方案的可行边界。

最后要说明的是，这套分工在小团队里会退化成“一个人同时站在三个位置上”。这并非不可行，但三件事仍然要分别被问一遍：这个业务问题真的值得解决吗？现场这条路径能不能跑通？其中哪些部分下一个项目还能用？角色可以兼任，问题不能省略。

06 实战案例

金融与医疗业务中的许多问题横跨业务理解、工程实现与组织协作，难以直接归结为一个模型任务。FDE的工作，是在现场识别这些因素之间的关系，确定可以介入的范围，再组织数据、知识、智能体与工具形成解决方案。以下结合金融测试、金融开发、医疗行业和能源行业四个案例，说明问题定义如何影响技术选择，以及现场经验如何转化为可复用能力。案例中的客户与机构名称均已脱敏。

6.1 案例一 金融测试 FDE

看问题

金融测试面临的核心压力，是在有限交付窗口内，对持续变化的业务建立充分、有效的验证。存量系统的回归需求与新业务的测试任务叠加，执行效率、覆盖质量和维护成本相互牵制。FDE进入某银行现场后，首先与测试人员共同参与实际操作，从报文处理、数据准备、案例理解和结果核验等环节识别阻塞，区分能够快速解决的操作问题与需要沿流程追溯的质量问题。

对于高频、重复且处理规律较明确的工作，团队围绕报文处理、环境模拟、案例对比分析与自动化质检快速开发了一系列现场辅助工具，通过现场使用持续修正功能。阶段性反馈显示，报文识别、案例对比分析等高频工作的效率均实现了数量级提升。这类工作使FDE能够在真实交付中检验技术价值，也为进一步分析业务流程积累依据。

更深层的发现是，测试执行承接了前序环节累积的质量问题。一次案例执行的耗时，既包含操作、等待响应和核验结果，也包含重新理解需求、寻找依据、补充数据、协调环境和重复试错。后者往往对应需求理解

偏差、设计约束遗漏、案例描述不完整或依赖管理不足。执行耗时因此也是观察整条交付链路质量的重要信号。FDE需要沿问题来源分析技术与管理原因，将改进措施落实到需求澄清、测试设计、案例维护和执行准备等环节。

这又提出了覆盖与变更管理的问题：已有案例验证了哪些业务规则，关键状态和异常分支是否遗漏，需求变化后哪些案例仍然有效。案例数量和粗粒度的需求映射难以充分回答这些问题。理清现状，需要掌握覆盖依据；预判影响，需要维护业务关联；回溯问题，需要保留设计与执行证据。

成熟测试工艺已经包含需求拆解、覆盖设计、评审和变更分析等方法，落地难点在于持续投入：业务语言非形式化，专家经验分散，关系维护依赖人工，而项目预算与交付周期限制了实施深度。由此，团队提出测试工艺平台的建设方向，以AI承接可重复的分析和处理，将专家方法落实为持续运行的工作过程。单点工具改善当前执行，工艺建设减少前序缺漏，两者共同服务于质量提升和长期提效。

想数据

数据选择需要沿着“业务要求-测试设计-实际行为”展开。当前需求、制度条款和澄清记录提供业务目标、对象、规则及验收依据；历史与关联需求说明规则如何演进、哪些条件沿用、哪些流程共享约束；测试大纲和既有案例反映这些要求如何被转化为场景、前置条件、步骤和预期；执行结果、缺陷记录与日志报文则揭示实际行为及验证中的缺口。接口契约、数据字典、环境配置和专家经验补充系统实现与执行条件，代码及构建信息在现场可获得时进一步接入。

理解数据的关键，在于识别其能够说明什么。例如，接

口契约给出字段格式和调用约束,业务需求说明交易成立的条件与账务含义;历史案例保留过去的设计选择,缺陷记录暴露曾经遗漏的规则或失败条件。将它们关联,才能判断一次失败源于业务理解、测试设计、系统实现还是执行条件。单独依据接口或旧案例推断业务规则,容易混淆历史行为与当前要求。

因此,来源、版本、适用范围以及需求、案例、结果之间的关系,本身就是需要积累的数据资产。以交易限额调整为例,既要识别新旧规则的差异,也要找到引用该规则的案例、相应测试数据和执行版本。存在冲突或缺失时,应保留问题并推动澄清。围绕具体任务逐步补齐这些信息,才能为后续分析提供有依据的上下文。

◎关联模型

基于上述问题,方案以本体组织业务语义与测试知识,为智能体提供方法和依据。这里首先需要明确:本体服务于业务目标,建设内容必须从具体场景中提炼。业务规则、测试经验和系统依赖难以一次性表达完整,因而应围绕能够落地的功能分阶段建设,在使用中扩充。场景决定知识范围,实际效果检验建模价值。

三层本体的构想由此形成。业务本体描述“业务应当如何运行”,表达对象、动作、状态和约束,支持需求理解与业务关联;测试本体描述“如何验证业务要求”,沉淀测试方法、适用条件、覆盖准则和评审经验,支持测试分析、设计、生成与评审;应用本体描述“业务在具体系统中如何承载”,组织接口、页面、操作、数据和环境依赖,支持执行准备与系统适配。三者按职责协作,使业务规则变化、测试方法改进和系统适配调整可以分别处理。尤其是测试本体,将专家工艺沉淀为可复用知识,为跨业务应用提供方法基础。

知识还需要进入实际处理过程。测试中间表示,在系统内部承担中间语义层的作用,按阶段表达测试对象、条件、覆盖关系、预期依据和来源,使智能体与功能模块

能够围绕同一语义交互。需求分析形成规则与疑点,测试设计明确场景和覆盖意图,案例生成细化步骤与预期,评审结果再反馈到相应阶段。本体提供可复用知识,测试中间表示,承载当前任务的阶段产物。智能体负责语义理解、方法选择和判断,程序负责查询、计算与装配,专家参与业务澄清和关键评审。

在这一过程中,需求、规则、测试设计、案例和执行结果应同步建立双向关联。正向可以判断某项业务要求是否有相应测试,以及测试是否已执行;反向可以说明案例为何存在、预期从何而来、失败需要回查哪些依据。双向追溯把设计意图与执行证据连接起来,使覆盖分析和问题定位拥有共同基础。

变更影响分析进一步区分“存在关联”与“需要调整”。先识别变化涉及业务规则、测试条件还是系统依赖,再结合关系查找候选范围,依据具体差异和适用条件作出判断。例如限额调整需要检查边界取值及预期,操作入口变化可能只影响执行适配。由此确定案例修改、新增、复测或无需调整的范围,并保留判断依据。执行失败的根因则需结合版本、环境、数据与日志确认,追溯关系提供定位线索。

资产回流使这套机制能够持续完善。业务澄清补充规则,设计与评审反馈改进测试方法,执行证据修正系统映射和依赖,可重复操作沉淀为工具。反馈需要经过分析与确认,再进入相应资产。新增知识由此改变智能体的分析范围、方法选择和流程配置;新的操作需求通过工具开发与集成形成能力扩展。本体扩充与功能演进相互促进,使现场经验能够实际改变后续处理,而非仅以文档形式留存。

这一分工可以用图4概括:三层本体提供可复用的业务语义,测试中间表示,承载当前任务的阶段语义,生成通道负责扩大覆盖,判定通道给出可复现、可审计的结论,两者产物再回流为平台的数据资产。



图 4 基于三层本体与 测试中间表示 的测试生成与判定双通道

平台部署

· 平台建设需要把知识使用、任务协作与工具执行组
组织为完整的工作过程。按照应用层、流程编排、智能体
运行时、本体运行时、工具层和数据资产层的结构,各
部分职责见表4。

架构组成	主要职责
应用层	提供测试分析与执行、本体建模等工作入口
流程编排	将测试工艺组织为阶段任务,管理产物传递、人工参与及问题返回路径
智能体运行时	运行智能体,管理任务上下文、工具调用与处理状态
本体运行时	提供知识访问、语义关联和关系查询,支持各阶段使用相关本体
工具层	通过技能组件、标准接口与插件接入报文处理、环境模拟、质量检查、案例对比等能力
数据资产层	保存本体、案例、执行证据及其来源与版本关系

表 4 测试平台的架构组成与职责

这一结构将测试工艺与现场适配分开:平台承接可复用的分析、设计与评审过程,现场工具接入具体系统和操作。FDE可以先解决单点问题,再将成熟能力纳入平台,为后续任务复用。平台效果也应结合案例质量、等待与返工、执行耗时及人工修正成本评价。

部署深度必须服从现场可控范围。需求与设计过程可参与时,优先贯通需求到案例的质量链路;案例由外部提供、团队仅负责执行时,重点建设轻量执行知识和依赖关系,支持场景理解、数据与环境准备、报文处理;具备代码与构建信息后,再扩展实现关联与失败分析。

本案例已形成单点工具的现场应用,测试工艺平台及

完整追溯能力仍按阶段推进。FDE的持续职责,是将场景选择、建模、集成、真实应用与反馈改进连接起来,使当前交付中的问题成为方法改进的依据,也使平台能力接受现场效果的检验。

6.2 案例二 金融开发 FDE

看问题

某金融开发项目要求在固定窗口内交付一批核心业务功能的初版代码,现场团队规模有限,且需要比常规节奏提前入场。项目需求条目众多,分散在文档、邮

件和会议纪要中,同时需要客户现场、远端研发与行方内部等多方团队协作。交付压力因此同时来自需求规模、业务理解和跨团队协作。

FDE首先需要把交付要求转化为可安排的工作范围:哪些功能必须优先完成,哪些存在未澄清的业务规则,哪些依赖外部接口或环境,以及初版交付需要满足什么条件。继而沿需求理解、设计、实现和评审过程查找阻塞,区分信息缺失、系统约束不明、协作等待和实现工作量,避免将所有问题都归结为人手不足。

该场景的关键认识是,代码生成速度只是交付效率的一部分。模型能够快速产出代码,但业务意图不清、事务边界遗漏或接口约定理解错误,都会把节省的编写时间转化为后续返工。因此,方案重点是将需求组织为边界清晰的开发任务,提供必要的业务与工程约束,并建立问题反馈路径。明确规则的工作交给AI辅助完成,影响关键业务行为的判断由相应人员负责。

◎想数据

开发任务所需的数据包括需求及澄清记录、现有代码与架构文档、接口契约、数据库结构和编码规范。需求资料用于识别业务目标、约束和验收条件;代码与设计资料用于理解已有实现及依赖;接口、数据结构与规范用于限定新增功能如何进入现有系统。每项任务都需要将这些信息关联到具体业务范围与实现位置。

既有代码还包含文档未充分表达的工程约定,例如事务边界、并发控制、异常处理和日志方式。FDE 需要结合代表性实现与研发人员的解释,区分可沿用的规范、局部特殊处理和历史遗留问题。单独复制某段代码的模式,可能将原有缺陷一并带入新功能,因此推断出的约定需要确认其适用范围。

历史变更、构建失败、测试结果和评审意见也应纳入后续数据积累。它们能够说明哪些约束容易遗漏、哪些依赖经常造成失败,以及哪些修改反复出现。任务推进时,将相关证据补充到当前上下文;任务结束后,再提炼为后续开发可复用的规则和经验。这样,数据既支持代码生成,也支持偏差解释与持续改进。

◎关联模型

方案围绕“需求澄清-任务拆解-代码实现-测试与评审-问题回溯”组织智能体与人工协作。首先依据业务对象、动作、状态和约束建立必要的语义关系,将分散需求整理为可讨论、可分派的任务。每项任务明确业务意图、相关规则、实现范围、依赖条件和验收依据,并将尚未解决的问题保留下来,避免模型自行补齐关键业务含义。

人机分工随任务性质调整。规则明确、重复性高的工作可由AI自动处理并通过常规构建与测试检查;边界清晰但存在变化的工作由AI生成、人工确认;涉及资金处理、关键状态转换和跨系统一致性的工作,由人主导设计, AI辅助分析与实现;仍需探索的业务与架构问题,由相关专家组织讨论, AI提供资料整理和方案比较。这些分工需要根据实际表现持续调整。

上下文供给也按任务和阶段组织。需求分析侧重业务规则与待澄清问题,实现阶段侧重相关代码、接口和工程约定,测试与评审阶段侧重验收依据、实际变更及执行结果。各阶段通过明确的任务说明和结构化产物交接,保留关键依据与未解决事项,使后续处理能够理解前序决策。

同时建立需求、开发任务、代码变更、测试结果与评审意见之间的关联。需求变化时,沿关系判断哪些任务和实现需要调整;构建或测试失败时,结合失败信息区分依赖配置、接口理解和实现逻辑等原因;业务评审发现偏差时,则回查需求解释和设计依据。关联提供调查线索,具体结论需要由相应证据支持。

反馈处理应落到问题产生的环节:业务规则遗漏需要补充需求理解,工程约定缺失需要更新任务上下文,实现错误需要修正代码并重新测试。经确认的共性内容再沉淀为业务知识、工程规则、任务模板和工具能力。通过这种方式,评审与修正能够持续改善后续任务的处理,而不仅是完成一次代码修改。

平台部署

开发场景可沿用前述平台结构，重点对接需求库、代码库以及构建和测试流程。应用层呈现任务、待澄清问题与处理结果；流程编排组织分析、实现、构建、测试和评审；智能体运行时，承担智能体运行与工具调用；本体运行时，提供业务及系统关系；工具层连接现有研发工具链；数据资产层保存需求、任务、工程规则及过程证据。

部署可以从边界清晰、依赖可获得的功能开始，贯通一个完整开发任务，再扩展到同类任务。涉及未确认的业务规则时，将问题交回负责人员；构建与测试失败时，根据错误类型返回相应处理环节；评审意见则与具体任务和变更关联，支持继续修改。需要并行开展的任务，应先明确共享接口和修改范围，减少相互等待与重复处理。

效果评估应覆盖完整交付过程，包括需求澄清时间、任务完成周期、构建与测试情况、评审返工及人工修正成本。FDE据此判断哪些任务已适合扩大AI应用，哪些仍需补充知识、工具或人工参与。现场团队、研发团队与业务人员也需要明确规则确认、实现修改和结果验收的责任。随着实际应用积累，成熟的任务组织方式、系统适配和工程经验才能形成可跨项目复用的交付能力。

工程工艺

FDE不能把AI视为某个环节的孤立插件，而应从软件开发的全生命周期出发，对需求澄清、方案设计、任务拆解、编码实现、代码评审、测试验证、构建发布、部署运维、监控反馈等过程进行整体分析和环节分解。要梳理每个环节的输入输出、参与角色、工具链、数据流、质量门槛和瓶颈，明确哪些工作适合由AI增强、哪些需要人机协同、哪些仍需人工决策。

在此基础上，将AI工程技术——如代码生成与补全、智能检索、自动化重构、测试用例生成、缺陷定位、日志分析、文档生成、CI/CD优化等——嵌入具体流程，使AI

能力与现有工程体系、规范、平台和数据打通。其目标不是追求某个单点的速度提升，而是从系统工程角度重新设计软件开发的生產方式：通过标准化、自动化、智能化和持续度量，形成端到端提效闭环，降低协作成本，提高交付质量与迭代效率，并让AI成为工程系统的一部分，而非游离于流程之外的工具。

6.3 案例三 医疗行业 FDE

看问题

医疗行业面临的核心压力，是在多个系统、多个时间尺度和多个专业视角之间建立一致的判断，并让这个判断可以被追溯。以医院感染管理为例，监测需要覆盖患者的整个住院过程，从检验、影像、医嘱、护理、手术和器械记录中判断一次感染是否成立、何时发生、是否属于院内获得，并据此触发上报与干预。判断依据分散在不同系统中，时间粒度不一致，专业口径也不统一；判错与漏判都有现实后果——过度判定会带来不必要的隔离与用药，漏判则可能延误处置并形成暴发风险。

FDE进入现场后，与院感科、临床科室和信息科共同复核一批历史病例，沿记录产生的过程识别阻塞。发现的问题多数不是“没有数据”，而是同一事实在不同系统中的表述不同：检验系统记录标本与培养结果，医嘱系统记录用药，护理记录记录体温与管路，影像报告用自然语言描述。一次感染判断需要把这些记录在时间轴上对齐，而人工对齐本身消耗的时间往往超过判断本身。因此，FDE需要区分哪些阻塞能够被工具直接消除，哪些属于跨系统的语义问题。

更深层的发现是，判断难以一致、难以追溯，根源在于缺少一个显式的事件对象。现有记录都以就诊或住院为单位，而感染是发生在住院过程中的事件：一次住院可能发生多个感染事件，可能跨科室，也可能在出院后才被回溯确认。没有独立的事件对象，就没有稳定的比较单位，不同人的判断差异也无法定位到具体

是哪一条证据上的分歧，讨论只能停留在结论层面。由此形成建设方向：把感染事件显式建模，让每个结论都能回到原始记录，使AI承担证据组织与候选生成，而把最终认定留给院感专家。医疗FDE的核心不是“接入一个模型”，而是把临床意图、数据语义、确定性规则与人工责任边界共同设计出来。

◎想数据

数据选择需要沿“事件-证据-结论”展开。首先要建立跨系统的患者与就诊主线，把患者、就诊、住院、转科、转床、入院出院和既往就诊统一到同一条时间轴上；其次要保留临床记录中的主体、时间、肯定或否定、可能性和来源，避免把“考虑肺炎”直接当作“确诊肺炎”；再次要记录治疗、操作与设备暴露，因为许多医疗事件的含义取决于它发生在什么操作之后、持续了多长时间。落到院感场景，数据至少包括患者与就诊轨迹，体温与症状体征、护理记录、血常规、尿常规、炎症指标、病原学检验、培养与药敏和影像报告，抗菌药物的名称、剂量、用法、起止时间与治疗意图，手术时间、切口类型、手术部位与植入物，中心静脉导管、导尿管、呼吸机的置入、维护、更换与拔除事件，以及隔离医嘱、院感诊断、预警处理、人工复核结论与复核理由。

理解数据的关键，在于识别其能说明什么、不能说明什么。培养阳性说明检出病原体，但不能单独说明感染，也可能是定植或污染；单次发热可能是感染信号，也可能是术后吸收热或药物反应；一次抗菌药升级可能源于感染加重，也可能源于经验性调整。因此数据进入处理过程时，必须同时保留其证据强度与来源，而不是只保留一个结论。

时间与来源本身就是数据。同一条记录需要保留采集时间、录入时间、来源系统和最后修改时间——记录时间与事件实际发生时间常常不一致，而感染判定高度依赖时间窗。采样时间不能被报告时间替代，器械置入时间不能从一条护理描述中臆测，手术部位感染需要区分无植入物 30 天和有植入物 90 天两个监测窗

口。缺失与冲突同样应当作为数据保留：某一天没有记录体温，不能等同于体温正常；无法确认的字段不能擅自补齐，而应输出“证据不足”。

◎关联模型

本体以感染事件为中心组织，而不是让患者或住院直接承载唯一的院感标签。感染事件作为独立对象，一次住院可以包含零到多个事件，每个事件独立登记感染部位、起始时间、状态、置信度及其临床表现、检验与影像发现、手术、器械和抗菌药物暴露；获得性评估负责院外或本院获得的判断，耐药菌评估负责耐药菌带入与非带入的判断，感染、定植与污染则由独立的临床评估承载。在证据层，所有结论都通过证据条目连接到原始记录，并记录来源系统、文档、观察时间、原文或结构化值以及抽取方式；评估记录保存结论、确定性、理由、规则版本以及支持证据与反对证据。

在上下文与智能体设计上，从检验与影像数据中提取与感染相关的异常信号和时间关系；从医嘱与护理记录中提取干预动作与器械使用；从制度与规范中提取判定条件与时间窗要求。智能体按感染部位与人群分别组织证据并提出候选结论，确定性程序负责时间轴对齐、字段校验、监测窗口计算与规则执行。以一位因社区获得性肺炎入院、随后接受气管插管和中心静脉导管的患者为例，住院第4天出现发热、氧合下降与新的肺部影像改变，第6天血培养检出耐药菌，第12天出现尿液异常：系统应同时给出入院时已有肺部感染、可能的VAP、可能的CLABSI与后续尿路感染四类候选，并判断该耐药菌是入院带入、住院后获得还是定植或污染。系统展示的应该是一条可点击的证据链——气管插管时间 → 发热与氧合变化 → 新发肺部影像 → 培养与药敏 → 抗菌药物升级——并同时并列展示缺失证据与相反证据，而不是只给出一句“疑似VAP”（如图5所示）。



图5 院感案例：以感染事件为中心的本体与可点击证据链

安全边界必须显式写进设计：不替代院感专家的最终认定，不把单个检验阳性、单次发热或单次用药等同于感染；对证据冲突、关键字段缺失、时间窗无法确定或来源无法追溯的病例，系统应明确输出“不确定/需复核”。这既是院感系统的安全边界，也是医疗行业AI建设能够持续扩展的前提。

平台部署

平台的部署首先要服从医院现有的信息系统与责任链。数据从哪里来、谁可以看、结论在哪个环节被确认、失败后如何回退，这些问题比单次模型调用更决定方案能否落地。平台应把本体运行时、规则服务、模型网关、证据服务和权限审计组织成一个受控工作流，并让结果回到医生、护士和院感人员熟悉的业务入口。

在院感场景中，应用层提供全院、院区/片区、科室三级视图，并按数据范围与操作权限设置默认着陆页：全院视图展示在院患者、新增院感病例、感染率、多重耐药菌、手术部位感染、“三管”感染和抗菌药物使用强度等指标，院区与科室视图则下钻到预警患者、床位分布视图与具体证据。预警列表支持按在院/出院、院区、科室和患者层级筛选，AI研判标签与患者基本信息、器械暴露、检验、影像、用药和人工处理记录同屏呈现。工具层封装数据采集、语义映射、规则管理、证据包生成、结构化解码、权限控制与审计接口；管理端

点击预警后先展示核心结论、浓缩依据与可能原因，展开后显示完整时间线、支持与反对证据、缺失项、规则版本和原文定位；临床端使用更轻量的列表与颜色提示，未经复核的AI结论不得直接写入正式诊断。数据资产层保存医疗本体、院感术语与同义词、规则版本、证据图谱、历史病例、人工复核记录与反例卡片。

平台宜在院内部署，按最小权限访问HIS、EMR、LIS、PACS、护理、手术麻醉、药学和现有院感系统，并让模型、提示词、语义图谱、证据包与输出结构纳入统一的版本管理。高风险或低置信度的结果自动转人工，人工确认或修改必须填写理由并保留原机器结论，支持审计、回放与灰度发布。

工程工艺

医疗场景的工程工艺，首先要把临床责任、数据质量、模型能力和平台运行拆开管理，再通过版本化流程重新连接起来。任务分级与前述开发场景同构：模板化、字段明确的任务适合自动化；涉及语义但边界清晰的任务适合AI生成、人工确认；涉及诊疗标准、核心风险与正式结论的任务必须由专业人员主导；研究探索类任务先离线验证，不直接影响生产。对应到院感系统，时间格式校验、采样早于报告的时序校验、器械置入与拔除完整性检查、证据去重和预警分层可以由程序或AI自动完成；感染事件拆分、来源归因和病原体定植

/污染提示由AI生成候选、院感人员确认;监测标准解释、复杂冲突证据与正式病例认定则由院感专家主导, AI负责整理证据并提出反例。

质量门禁也应从医疗行业的一般指标下沉到院感业务指标:关键字段完整率与时间一致率、感染事件边界一致率、院内/院外归因准确率、VAP/CAUTI/CLABSI/SSI/MDRO分类型的敏感度与特异度、支持证据引用正确率、无来源断言率、反对证据识别率、人工复核一次通过率、预警处理时延和系统吞吐。评估数据按时间、科室、感染类型、病原体、器械和记录完整度分层, 并采用双人独立复核加第三人仲裁形成金标准, 不能只用成功案例验证。

本体建设同样不能追求一次性的“大而全”, 而应分三个阶段推进:第一阶段打通住院轨迹、感染事件、感染部位、标本、微生物结果、器械、手术、用药与证据引用, 落地少数高价值预警;第二阶段扩展复杂冲突、耐药菌来源、暴发信号与床位分布视图;第三阶段纳入人工反例、规则变更与历史回放。每次规则或本体变更, 都要在锁定的测试集上回放, 经院感专家审核后灰度发布。

6.4 案例四 能源行业 FDE

看问题

能源行业的一个典型场景是新型发电, 例如垃圾焚烧发电厂。这类电厂在安全、环保与稳定运行的硬约束下运营, 同时还要在燃料成分波动、负荷变化和设备老化中维持效率。客户需要的不是几个单点的AI功能, 而是在上述约束下, 把分散的数据、机理知识和运行经验转化为可复制的智能决策能力。顶层设计通常以“AI全景大脑”为中枢, 依托云边协同平台集成AI算法与机理模型, 围绕设备健康、燃烧优化、安全管控等多个专业领域组织场景方案;再结合数字化基座与工业互联网平台实现全域数据感知与多源信息融合, 基于

数字孪生形成虚实互动的决策闭环, 推动电厂运营从“人工经验决策”转向“数据—知识双轮驱动”, 逐步落地本体与智能体应用。

FDE进入现场后, 需要先弄清智能化落地究竟受制于哪些因素:数据质量、机理与AI的分工、IT与OT的边界、现场操作习惯、安全约束, 以及顶层设计与场景落地之间的衔接。数据分散在DCS控制系统、设计与控制资料、运行资料、访谈记录和操作日志中;更麻烦的是, 控制逻辑与联锁约束大量并不写在文档里, 只存在于运行人员的经验之中。协作方也更多:FDE团队、客户现场的运行/检修/管理团队, 以及平台与模型团队。

行业已经形成的基本思路可以概括为三件事:理清现状、预知风险、回溯历史。理清现状是把数据、知识与本体结构化;预知风险是识别哪些场景可由AI全自动、哪些必须人机协同、哪些必须由人主导;回溯历史是让每一次偏差都能定位到是数据、本体、模型还是流程的问题。这里的核心, 是把“专家经验瓶颈”转化为“本体与智能体协同的范式问题”——不是让更多专家追着问题跑, 而是改变知识供给与决策产生的方式。

现有方法的精华与不足同样清晰。传统信息化与数字化靠不断叠加系统, 沟通与集成成本非线性上升、边际收益递减;而直接把大模型接到电厂去生成控制建议, 又受制于现场隐性约束与可解释性, 把成本转移到了验证环节。前者提示瓶颈在知识与协同, 后者提示瓶颈在约束供给与工程化。

想数据

数据选择要覆盖数据、机理、控制、运行与安全这几类因素, 因此至少包括近年的DCS等运行历史数据、设计与控制资料、运行资料、数据资产清单、现场勘查与盘前操作观察记录、运行/检修/管理人员访谈, 以及典型工况与操作习惯记录。

第一步是给电厂做一次全面的“数据体检”:与客户对接确认数据资产清单, 现场收集历史数据与资料, 观

察盘前操作,访谈运行、检修与管理人员,记录典型工况与操作习惯,形成高质量数据集。这些都是“数据背后的深意”:DCS曲线背后是控制逻辑、联锁约束与操作习惯,运行规程背后是安全边界与优先级,访谈记录背后是隐性经验。数据是表面的,数据背后的工程约束与业务意图,才是模型真正需要理解的内容。

数据进入能源AI平台后,通过AI模型、机理模型与大数据分析模型的综合诊断,输出“电厂体检报告”,覆盖三类内容:运行分析报告,量化AI方案的提升空间与可行性;设备智能报告,给出可优化的设备物理参数与控制规律分析;优化方案报告,给出可落地的控制优化方案。

关联模型

模型部分重点是基于业务知识设计人机分工。电厂智能化需要把IT与OT有机结合,在管理、控制、操作各层构建AI原生架构:IT侧面向信息管理、业务处理与数据分析,采用LLM、VLM等大模型和大数据分析模型驱动的分析决策智能体;OT侧面向现场控制、设备监控与数据采集,采用AI小模型和机理模型驱动的执行动作智能体。两者之间形成IT与OT融合下的协同模式:大模型负责规划、解释和指导,小模型负责执行、反馈和迭代(如图6所示)。

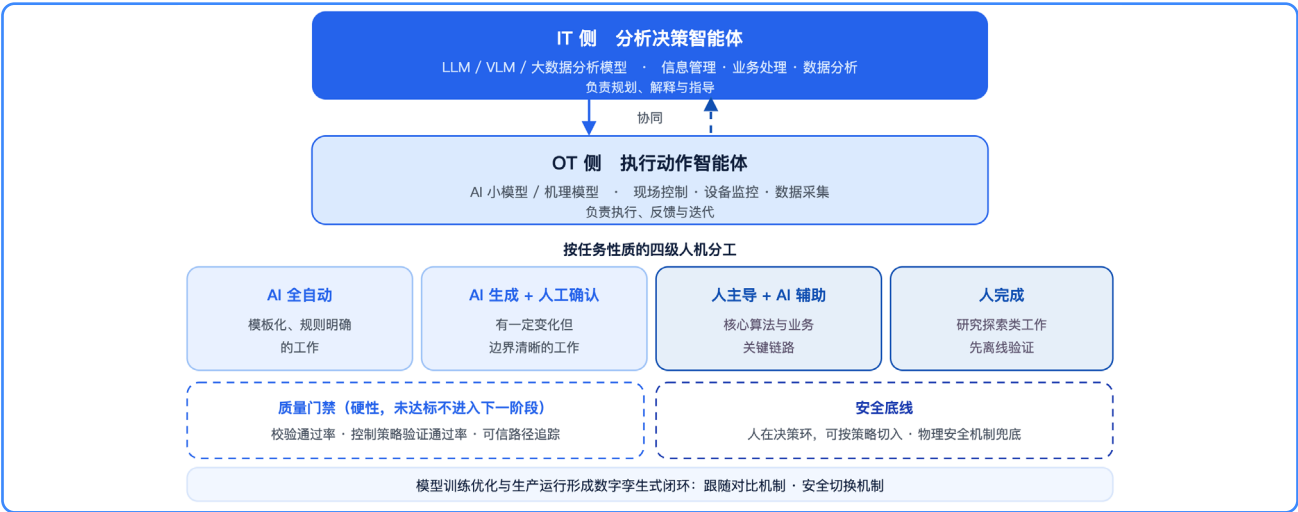


图6 能源行业 FDE:IT 与 OT 双层智能体架构与四级人机分工

模型选型上,随着基础大模型底层能力提升,结合企业本体,多数场景已可替代专门训练的行业模型。团队通过针对客户业务特点的评估体系选择模型,在算力成本与效果之间取平衡,给出性价比更高的选型;对AI小模型与机理模型,则建立跟随对比机制和安全切换机制,让模型训练优化与生产运行形成数字孪生式闭环,推动智能系统持续进化。

任务分级同样按性质划分四级:模板化、规则明确的工作由AI全自动完成;有一定变化但边界清晰的工作由AI生成、人工确认;涉及核心算法与业务关键链路的工作由人主导、AI辅助;研究探索类工作由人完成,先离线验证、不直接影响生产。验证阶段提供可信路径追踪,并以校验通过率、控制策略验证通过率等指标作为硬性门禁,未达标

不进入下一阶段;生产运行采用人在决策环,人可按策略切入,同时保留物理安全机制作为底线保障。

平台部署

平台采用分层化、组件化架构。用户层既提供开箱即用的电厂主流场景智能体,也支持针对业务特色做自定义或定制化开发;智能服务层依靠本体平台与智能体平台对接原有业务系统和数据中台;基础设施层提供算力与模型服务。

除了提供端到端的整体方案,FDE还要针对客户已有的数字平台或AI平台做对接改造,充分复用已有资产、避免重复建设,保障客户的既有投资。

◎ 工程工艺

FDE不能把AI视为某个环节的孤立插件,而应从电厂智能化的全生命周期出发,对数据治理、本体构建、需求澄清、方案设计、任务拆解、模型开发、策略验证、系统集成、运行监控与反馈迭代等过程进行整体分析和环节分解,梳理每个环节的输入输出、参与角色、工具链、数据流、质量门槛和瓶颈,明确哪些工作适合由AI增强、哪些需要人机协同、哪些仍需人工决策。

落到电厂场景,需要面向AI的数据治理与AI原生能力构建。团队采用本体方法论,把发电侧通用行业本体、垃圾电厂专项本体与现场定制本体结合起来,通过工艺流程与本体平台完成数据、知识、本体的构建,并提供智能化

维护流程,保障本体动态持续演进;同时引入智能体的强化学习机制,形成不断进化的本体工程工艺。其目标不是某个单点的速度提升,而是从系统工程角度重新设计电厂智能化的生产方式:通过标准化、自动化、智能化和持续度量形成端到端提效闭环,降低协作成本,提高交付质量与迭代效率。

● 6.5 四类FDE的对比

金融测试、金融开发、医疗行业和能源行业四类FDE面对的问题结构差异很大,行业约束、语义载体与责任边界各不相同,需要维护的资产、人机协作的重点和检验成效的指标也随之不同。四者的对比见表5。

维度	金融测试 FDE	金融开发 FDE	医疗行业 FDE	能源行业 FDE
核心问题	现场执行阻塞与前序质量问题叠加,测试依据和变更关系难以维护	需求规模、系统隐性约束与协作成本限制交付效率	跨系统、跨时间、跨专业的判断难以一致且不可追溯	电厂运行优化与智能化改造难以快速见效,隐性约束多
主要交付物	现场提效工具、测试工艺能力,以及关联需求与证据的测试资产	符合业务与工程要求的代码,以及可复用的开发任务与交付方法	可复核的感染事件与可点击回溯的证据链,以及院感本体与规则资产	电厂体检报告、可落地的控制优化方案,以及本体与智能体场景应用
知识组织	业务本体、测试本体与应用本体协作,测试中间表示 承接阶段语义交互	业务语义、系统依赖与工程约定协作,任务产物承接阶段交互	医疗本体与患者时序证据图谱协作,以感染事件为中心组织证据	发电侧通用行业本体、电厂专项本体与现场定制本体结合,大数据、AI与机理模型融合协同
人机协作重点	专家参与澄清和方法建设,智能体执行测试工艺,工具处理具体操作	人负责关键业务与设计判断,智能体辅助分析和实现,工具完成构建与测试	AI组织证据与生成候选,院感专家作最终认定,程序负责时间与规则计算	AI生成与优化建议,机理和规则做校验判定,人管安全边界、核心控制链路与最终决策
追溯重点	需求、规则、测试设计、案例、执行结果与缺陷	需求、开发任务、代码变更、测试结果与评审意见	感染事件、临床评估、证据条目与原始记录来源	数据、本体、模型与流程,使每次偏差可定位到具体环节
成效观察	等待与返工、案例质量、有效覆盖、执行耗时和人工修正成本	任务周期、构建与测试情况、评审返工和人工修正成本	事件边界一致率、归因准确率、无来源断言率与人工复核一次通过率	场景运行优化收益、控制策略验证通过率、人工介入率
资产回流	业务规则、测试方法、执行依赖与现场工具	业务规则、工程约定、任务模板与研发工具	院感本体、判定规则、反例卡片与评测集	通用、专项、定制本体、控制优化方案与模型资产
主要风险	幻觉用例导致的虚假覆盖	隐性约束违反导致的技术债	无来源结论与过度断言(把定植当感染)	数据不完善导致效果提升有限;有效控制往往还需设备技改配合

表5 四类FDE对比

四类实践共同体现了 FDE的工作方式:在现场发现并定义问题,结合行业知识组织数据与任务,通过工具和智能体形成解决方案,再依据实际应用持续修正。能够长

期复用的资产,来自对问题、方法、依据和结果的持续积累。平台承载这些能力,FDE则推动它们在具体业务约束下发挥作用,并将新的现场认识反馈到后续建设中。

结束语

AI时代已经来临，如何把模型能力转化为业务价值，将关乎行业发展乃至产业格局，需要一批优秀的FDE来承担时代赋予的使命。FDE需要快速学习行业知识、深度理解数据与业务语义、精通智能体与上下文工程技术、熟练运用企业级AI平台，并具备AI侧端到端的解决方案能力；同时还要树立正确的AI落地观，并不断提升认知能力、学习能力、创新能力和沟通能力四大基础素质。

FDE以前线部署为方法论来认识和改造业务现场，解决各类行业问题、创造业务价值，不断扩大前线部署工程的外延，并在此过程中，不断沉淀可复用的行业本体与工程范式、反哺产品能力，深化前线部署工程的内涵。从这个意义上说，FDE不是AI的替代者，而是AI与业务之间的翻译者、约束者和兑现者。



扫码预约专家交流
获取1v1专属方案

江苏润和软件股份有限公司

网址: www.hoperun.com

电话: 400-980-0339

邮箱: contact@hoperun.com

地址: 江苏省南京市雨花台区软件大道168号润和创智中心

©2026 江苏润和软件股份有限公司 版权所有。

未经书面授权, 禁止商用转载、修改。内容仅作行业交流参考, 不构成任何承诺。