银行业 AI Agent 的企业级架构——从“服务导向”到“智能导向”的体系化重塑

原创 twt社区 twt企业IT社区 2026年2月9日 07:35 辽宁

导读

从单点试点走向银行业的规模化应用，AI Agent战略仅仅依靠工具堆砌已难以为继。面对复杂的存量系统与严苛的合规要求，亟需一套具备前瞻性的顶层设计与企业级架构来支撑与引导。**本文是基于真实的银行实践，结合业务诉求与试点经验所沉淀出的阶段性思考与架构蓝图。**也期待抛砖引玉，与同业共商 AI Agent 破局之道。全文包括：AI Agent 广泛应用带来的架构级挑战、银行 AI Agent 的架构蓝图、架构中关键组件的建设策略、组织级的建设团队。

作者：周友道

某银行数据管理部高级经理，数据架构师，负责银行大数据平台、数据中台、AI智能体等的规划与建设，以及推进数据与大模型技术在银行数字化转型中的应用，相关工作多次获得人民银行科技发展奖等荣誉。

前言

大模型方兴未艾，智能体（AI Agent）已异军突起。作为商业银行的企业级架构师与一线工程实施者，我们在规划全行 AI Agent 战略时，正经历着一场“兴奋”与“焦虑”交织的心路历程。

兴奋在于，我们切实看到了智能体作为关键抓手，将大模型能力推向业务深水区的巨大潜力，部分先行试点的成效也印证了这一判断。然而，焦虑随之而来：从单点试点走向全行级推广、规模化应用，仅仅依靠工具堆砌已难以为继。面对复杂的存量系统与严苛的合规要求，我们亟需一套具备前瞻性的顶层设计与企业级架构来支撑与引导。

本文是基于真实的银行实践，在评估信创环境适配、数据治理现状、存量系统复杂度及安全合规红线的基础上，结合业务诉求与试点经验所沉淀出的阶段性思考与架构蓝图。我们期望以此文抛砖引玉，与同业共商 AI Agent 在银行业的破局之道。

一、AI Agent广泛应用带来的架构级挑战

回顾银行业过去十余年的科技架构演进，从大机下移、SOA 治理到微服务与中台化，其主旋律是走向“服务导向”，旨在解决业务解耦与敏捷交付的难题。然而，随着大模型的成熟与 AI Agent 的崛起，从客户交互到流程执行，再到底层数据支撑，科技体系正面临前所未有的范式更新。

在前期的场景探索中，无论是面向内部赋能的“办公助手”、“客服助手”，还是面向业务增效的“零售营销锦囊”、“对公尽调助手”，我们都遭遇了共性的“落地墙”：前端交互显得生硬反人性、插件生态匮乏导致能力受限、与核心系统对接的边际成本居高不下、跨域数据难以打通、以及 Prompt 工程化管理混乱等。

透过这些表象，我们意识到这已非常规的技术修补，而是深层次的架构级挑战：银行IT 架构需要从一套确定性的、被动响应的指令执行系统，进化为具备概率性推理、自主规划与主动感知的“智能导向”生态。

具体而言，我们总结了如下层面的挑战：

1、交互范式的重塑：从“人找服务”到“服务找人”。银行当前主流的应用模式是“人找服务”式的，银行的各种金融产品如手机银行、网上银行等展示出了层层叠叠的菜单、宫格和导航栏，用户的操作需要分解为界面上一系列的浏览点击，进而对应后台一系列的接口服务。而在Agent架构下，交互模式转变为“人机协作”甚至“机器自主”，用户表达意图，Agent 通过对意图的“语义理解”和任务的“推理规划”动态生成服务路径，这样对于金融产品的形态、客户旅程的设计都是新的挑战。

2、架构拓扑的演进：从微服务到Agentic Mesh。微服务架构通过将单体应用拆分为独立部署的服务单元，解决了开发效率和扩展性问题。然而，微服务之间的交互依赖于刚性的API契约，缺乏灵活性。当业务需求变化时，往往需要重新编排服务调用链。而Agent则不然，它可以身处于“Agentic Mesh”（智能体网格）之中，能够自我描述、自我注册和动态发现，Agent之间也不是严格的架构层级关系，调用的链路从单向线性变成了多向网状。这种关系将使得底层的服务治理、通信协议和数据交换格式发生大的变化。

3、研发运营的变革：从Devops到LLMOps。近些年来，银行科技在研发运营层面大力推进了DevOps和DataOps，但在Agent层面缺少新的生命周期管理机制。Agent的开发不仅涉及代码，还涉及提示词（Prompt）、模型参数、知识库和技能（Skills|Tools）的配置。如何对这些非代码资产进行版本控制、测试和发布？如何构建Agent特有的CI/CD流水线（即LLMOps），实现从Prompt工程到模型微调的自动化？如何构建合适的评估框架进行非确定性调用测试？如何优化APM工具实现对Agent思维链中所使用资源的监控？这都是新的课题。

4、数据与资产的跃迁：从数据资产到新的数智资产。银行的数据当前以“结构化数据”为主，数据模型承袭于库表思想。所构建的数据资产是企业

业务模型的低维静态投影，以服务人的分析为目标，但对大模型和Agent并不友好。故而更容易为大模型所用的语义模型开始引人关注，实现方案之一的“本体模型(Ontology)”成为了新的资产建设目标。此外在Agent应用当中的非结构化数据、提示词、知识库、Skill等新的数智资产也需要进行规范和管理。

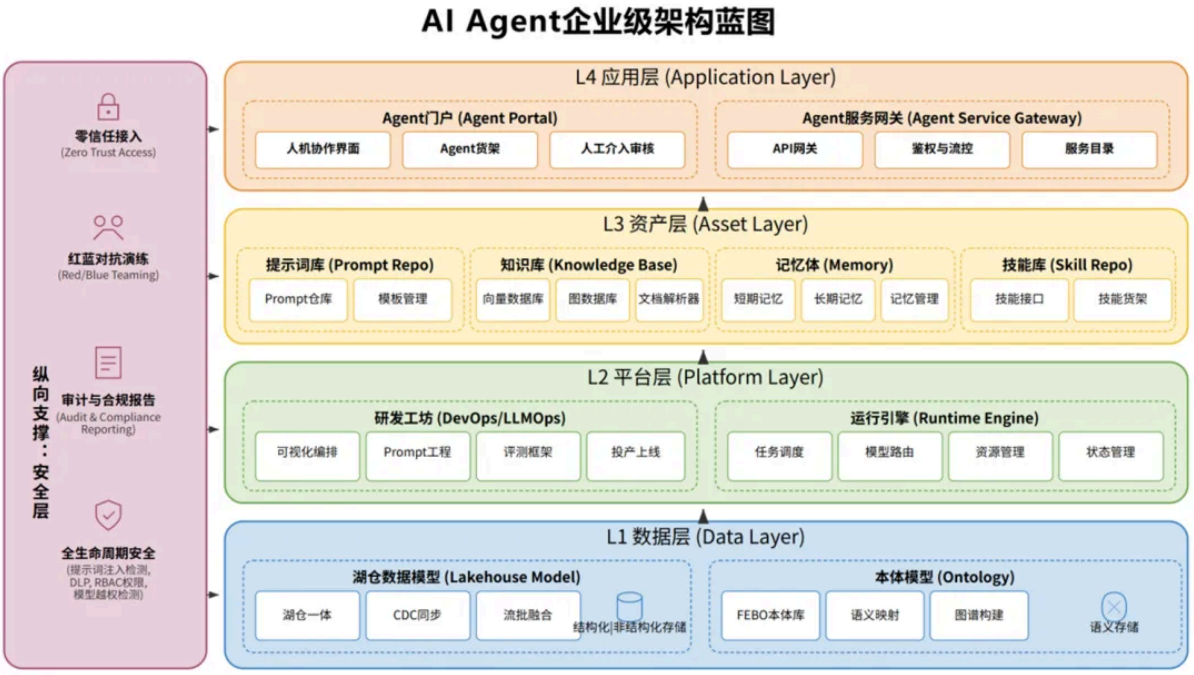
5、安全与合规的增强：新的防御要求。在传统的漏洞扫描和安全攻击之外，Agent的应用带来新的威胁，尤其是提示词注入（Prompt Injection）这种因自然语言的边界模糊不清而更难防御的攻击，以及越权访问这种难以觉察的隐患。如何构建新的Agent防护栏已经是迫在眉睫的要求。

故而，当前的银行IT架构——以结构化数据为底座、以刚性API为连接、以偏静态规则引擎为中枢——在面临Agent广泛应用时，需要提前思考并应对架构的挑战，从而构建一个既具备高度智能又符合金融级风控要求的Agent生态系统。

二、银行AI Agent的架构蓝图

2.1 总体架构视图

基于前面的分析，银行AI Agent架构蓝图，不能只考虑模型能力的引入和Agent功能的构建，必须进行体系化设计。我们总结出“4+1”的架构蓝图，即一个总体架构视图，包含四个核心横向层次（数据层、平台层、资产层、应用层）以及一个纵向贯穿的安全层，具体结构如下：



2.2 数据层(Data Layer)：数据模型与本体模型的双模驱动

数据层是Agent智能的基石。在传统的数据模型的基础上，银行必须引入本体模型，完成从“Human-Ready”到“AI-Ready”的跃迁。

在实际的权衡上，我们虽然深知全行级的本体模型是目标态，但是实施的成本和推行难度极大。一方面湖仓数据和集群规模体量巨大，部分能力还需要持续建设和运营；另一方面业务部门基于宽表的数据分析中沉淀了大量的经验，尚未形成成熟的本体分析方法论。因此我们的策略是双模驱动，将数据模型的优化和本体模型的局部建设并行推进。

在数据模型层面，主要是保鲜和提质。持续利用 CDC 及流批一体技术，将银行核心资产（账户、交易、余额）实时孪生至数据湖仓中，为 Agent 提供精准、鲜活的“事实”来源；同时通过持续的数据治理，捍卫数据的一致性、完整性与时效性。

在本体模型层面，主要是补齐与建模。一是补齐非结构化数据的存储和处理能力，二是引入金融企业业务本体（FEBO - Financial Enterprise

Business Ontology）,构建一套机器可读的业务语义模型。考虑到实施周期和成本，我们优先针对针对客户画像、交易分析等业务领域进行“微本体”建设，其核心步骤包括：

1、语义构建

从业务实体出发，结合已有的数据模型，设计FEBO本体模型。既可以将本体模型中的实体、属性、关系等与物理表进行映射，也可以重新设计模型与物理数据进行编织。这样，当Agent接收到“查询高风险客户”的指令时，它能通过本体模型理解“高风险”和“客户”的定义，并自动生成正确的SQL查询。

2、知识推理

将散落的业务规则、策略以及本体间的约束进行语义化封装，作为本体模型的一部分、或者是独立的业务逻辑层，赋予Agent“业务逻辑自洽”的能力。例如，本体定义了“某个金融产品只能推荐给风险承受等级高的客户”，当Agent进行个性化推荐时，即使没有显式规则，它也能基于本体约束自动推理并校验客户的风险等级，确保业务合规。

2.3 平台层(Platform Layer)：Agent开发运营一体化

平台层主要解决Agent“如何制造”和“如何运行”的问题。

在实际的权衡上，我们深入对比了开源（如 Dify）、商业私有化（如 HiAgent）和 SaaS（如 Coze）三种模式。考虑到我行目前的 AI 平台工程师资源相对紧缺，且行内需求较为旺盛，我们当前采取了“商业软件私有化部署为主，开源技术跟进为辅”的策略。这既能快速获得成熟的工具链和生态能力，快速满足研发态和运行态需求，又能为后续自主掌握和深度定制做好储备。

1、研发态：一站式的 Agent 工厂

（1）可视化编排：提供拖拽式的工作流定义能力，支持将大模型推理、RAG 检索、工具调用等原子能力灵活组装，降低业务人员的开发门槛。

（2）Prompt 工程化：建立 Prompt 的全生命周期管理，支持版本控制、在线调试与 A/B 测试。例如，可直观对比Deepseek与千问在同一Prompt下的生成差异，辅助模型选型。

（3）自动化评测：内置针对 Agent 的测试集，对回答准确率、响应延迟、Token 消耗等指标进行量化打分，确保上线质量。

（4）敏捷发布：打通现有的 DevOps 流水线，支持 Agent 的容器化部署与独立发布。特别是在上线前增设合规检测卡点，防止潜在的安全风险流入生产环境。

2、运行态：智能调度与可观测性

（1）模型路由：根据任务的复杂度与成本预算，动态选择模型。简单任务路由到低成本的小模型，复杂推理任务路由到大模型，实现“算力成本”与“智能水平”的最优平衡。

（2）任务调度：支持同步与异步双模式。对于生成研报等长耗时任务，自动降级为异步队列处理，保障系统吞吐量。

（3）全链路可观测 ：不仅是对容器、Token 吞吐量的监控，更重要的是对 Agent“思考链” 的完整记录与审计。确保每一个决策步骤都可回溯、可解释，满足金融审计要求。

2.4 资产层(Asset Layer)：核心数智资产库

这一层是Agent架构中区别于传统架构最显著的部分，管理着Agent的“大脑”内容。

在实际的权衡上，鉴于当前Agent 技术生态仍在剧烈演进，我们在资产治理上确立了“集中管控，标准先行”的原则。为了避免各业务条线重复造轮子或形成新的“知识孤岛”，我们暂不鼓励领域级的独立建设，而是构建全行统一的资产库；同时，面对多团队协作的现状，提前制定资产接入的标准化规范，在建设初期就通过形式化约束确保资产的高质量与可复用性。

具体资产包括：

1、提示词库 (Prompts)

Prompt是Agent的源代码。我们建立企业级的Prompt仓库，支持：

（1）版本化管理：类似于Git的代码版本控制，确保每一次Prompt的修改都有迹可循，且可回滚。

（2）模板化：提供通用的Prompt模板（如“角色设定+任务描述+约束条件+输出格式”），降低开发门槛。

2、知识库 (Knowledge)

知识库用于存储银行的非结构化知识（制度文档、产品手册、投研报告）并提供向量化检索，同时也是基于本体模型构建的实例填充。

我们使用RAG (检索增强生成)进行常规的文本分片和向量检索，进一步可以利用GraphRAG (图增强检索)，将文档中的实体与关系提取出来构建知识图谱。Agent在检索时，不仅能找到关键词匹配的片段，还能沿着图谱

关系进行多跳推理（Multi-hop Reasoning），从而提升回答的全面性和准确性。

3、记忆体 (Memory)

记忆体是Agent维持对话连续性的关键。

（1）短期记忆：管理当前会话的Context，采用滑动窗口或摘要机制，防止超出Token限制。

（2）长期记忆：基于向量数据库存储历史交互信息，并支持将会话中的关键信息（如客户的风险偏好变化）“写回”到L1层，实现记忆的持久化，并可经过数据加工完善用户画像。

4、技能库 (Skills)

技能库封装了Agent可调用的能力。包括基于服务接口的原子能力，例如：“查询余额”、“冻结账户”等；也包括执行一个业务流程的复杂技能，如“对客户进行尽调分析并给出报告”。这些能力需要标准化封装与分目录呈现，支持被不同的场景复用。

2.5 应用层(Application Layer)：全渠道触点

包括面向人的统一入口以及面向系统的智能服务。

1、Agent门户。为银行内部员工和外部客户提供统一的Agent交互界面，通过多层权限进行管控。支持多模态交互（语音、文本、图片），并集成了人工介入（Human-in-the-loop）的审核流程。

2. Agent Service。将Agent封装为标准的API或其他服务（MCP|A2A等），供手机银行App、柜面系统、OA系统以及其他Agent调用。通过

AI Gateway进行流量管理、鉴权和计费。

2.6 安全层(Security Layer)：纵向支撑

在实际的权衡上，考虑到Agent打破了传统边界，安全必须无处不在。从零信任接入到红蓝对抗演练，再到针对Prompt注入的全生命周期防护，需要构建动态的防御体系，并渗透在上述所有层级中。

1、数据层安全：数据分类分级、向量数据库的加密存储。

2、模型层安全：输入端的Prompt注入检测、输出端的敏感词过滤。

3、应用层安全：基于RBAC的权限控制，确保Agent只能访问其被授权的数据和工具。

三、架构中关键组件的建设策略

3.1 平台建设选型

在平台选型上，银行面临着“自主可控”与“功能丰富度”的权衡。我们对主流的三种模式进行了对比评估：

维度	开源/私有化部署 （如 Dify）	商业软件/私有化部署 （如 HiAgent）	商业化/云端 （如 Coze/SaaS）
数据隐私	高。支持完全私有化部署，数据不出内网，满足金融监管要求。	高。支持完全私有化部署，数据不出内网，满足金融监管要求。	低。数据需上传至云端，存在数据主权与合规隐患。
可扩展性	高。源码级开放，可根据银行需求深度定制（如对接内部系统、审批流）。	中。一般不开放源码，通过商业项目由厂商进行定制，定制有一定限制。	中。依赖插件机制，核心逻辑无法修改，难以深度嵌入银行遗留系统。
运维成本	高。需自行维护 Kubernetes 集群、数据库、中间件，对技术团队要求高。	高。需自行维护 Kubernetes 集群、数据库、中间件，对技术团队要求高。	低。全托管服务，开箱即用，无需关注底层基础设施。
生态集成	中。需自行开发或通过社区插件集成外部工具。	中。可自行开发或者通过安全协议使用厂商工具	高。内置大量第三方插件（如水滴信用、豆包等），生态丰富。

由于银行对于合规性要求较高，一般不选取云端模式，但是上面的方案也不是非此即彼的关系：

1、创新验证域。对于非敏感、不涉及数据出行或需要快速试错的场景（如会议助手、营销文案生成等），可以尝试使用Coze等SaaS平台，利用其丰富的插件生态快速构建应用，验证可行后再迁移行内。

2、核心业务域。对于科技资源较匮乏的银行，可以使用商业软件私有化部署的方式，实现Agent的开发运营。对于科技资源较为充足的银行，可以在行内同时部署开源与商业软件，再分场景应用：对于需要深度定制、有复杂处理逻辑的场景，可以使用开源软件进行研发；对于较简单的功能性Agent或者工作流场景，可以使用商业化软件进行快速构建和批量复制。

3.2 连接协议：MCP vs Skills

Agent如何连接银行庞大的遗留系统和存量IT资产，这也是架构中要考虑的问题。

Model Context Protocol (MCP)是由Anthropic推出的一种开放标准，存量的服务可以基于它封装为MCP资源（Resources）和工具（Tools），从而将系统集成的“网状复杂度（N x M）”降低为“星型复杂度（N + M）”。Skills更侧重于应用层的业务逻辑封装。它定义了完成某项任务的“流程知识”，描述了“如何做”，通常以代码或Markdown文件的形式存在。

这两种其实并不是一个层面的东西，可以进行分层协同：

1、底层：利用MCP协议标准化封装银行的原子能力，如数据库读写、API调用。这层关注的是连接性和安全性。

2、上层：利用Skills封装复杂的业务流程逻辑。Skill内部调用底层的MCP工具。这层关注的是业务逻辑和可复用性。

3、演进路径：在网关层（Agent Gateway）适配MCP协议，使其成为银行内部Agent互联互通的标准总线。长期来说更应推进核心系统原生支持 MCP。

3.3 记忆体构建

银行客户的交互往往是跨渠道、跨周期的。Agent必须具备长时记忆，才能提供连续、个性化的服务。

1、短期记忆：上下文窗口管理。

大模型的Context Window是昂贵且有限的资源，可优化的方式包括：

（1）滑动窗口：仅保留最近N轮对话，确保Token不超限，但可能丢失早期关键信息 。

（2）摘要机制：利用一个小模型对历史对话进行摘要，提取关键事实（如“客户有贷款需求”），保留摘要而丢弃原对话。这在长对话场景中效果更佳。

2、长期记忆：向量数据库与用户画像的融合。

（1）向量数据库：存储历史交互的Embedding。当客户再次咨询时，Agent通过语义检索召回相关的历史对话，实现“未卜先知”的体验。

（2）结构化用户画像：这是银行Agent记忆体建设的关键。Agent不仅要“读”记忆，还要能“写”记忆。会话信息和短期记忆数据要进行集中存储与分析，其中的一些客户关键信息（如“我刚生了宝宝”），可以构建新的标签写入数据层的客户画像中，从而增强动态个性化服务的能力。

（3）构建策略：Memory Bank模式。一般而言，商业化的Agent平台软件会自带记忆模块，但从长远看，银行应该构建统一的Memory Bank服务。它独立于具体的Agent存在，维护着全局的用户记忆图谱。无论是客服Agent、理财Agent还是App搜索Agent，都连接同一个Memory Bank，确保用户在不同触点获得一致的体验；并可以为其他的业务服务。

3.4 Agent安全

Agent的引入打破了传统的网络边界，攻击面从网络层延伸到了语义层。对应的安全机制也需要随之升级。

1、Prompt注入防御：多层过滤

（1）输入侧：部署专门的“指令分类模型”，识别并拦截试图覆盖系统指令的恶意Prompt。

（2）隔离沙箱：Agent执行代码（如Python解释器）必须在严格隔离的沙箱环境（Sandbox）中运行，禁止访问外网，限制系统调用，防止恶意代码逃逸

2、业务权限控制：身份认证与人工介入

（1）Agent在访问受控系统时，至少需要两层权限控制，第一层是Agent本身这一个“虚拟用户”是否具备访问系统或者服务的权限，第二层是使用Agent的用户对应的身份（如员工或者客户）是否具备访问具体内容的权限。这两部分都需要相关的权限认证。

（2）人工介入（Human-in-the-loop）：对于高风险操作（如大额转账、修改利率），Agent无权直接执行，必须触发一个审批卡片，由拥有权限的人类主管确认后方可执行

3、数据权限控制：一般认证与差分隐私

（1）Agent在访问具体数据模型或者本体模型时，应当复用已有的数据控制策略进行访问控制，部分验证还可以委托到业务系统进行，以防止访问越界；

（2）更为复杂的是，Agent在运转时，可能会把敏感信息发送大模型，或者保留在记忆中，再通过其他的处理或者使用被共享出去。这里的一个做法是，在Prompt发送给LLM之前，先经过一个个人身份信息过滤器。利用NLP技术识别姓名、卡号、手机号，并进行脱敏处理。模型返回结果后，再还原为可读信息来保障敏感数据不落地、不出域。

3.5 待解的困惑

尽管蓝图清晰，上述的策略也已在分阶段实施中，但也**确实遇到了一些意料之中与意料之外的困境，这也是我们希望和同业共同探讨的课题。**

1、商业产品路径与本地化的矛盾。商业化 Agent 平台的 Roadmap 往往由厂商掌控，这与行内深度的二次定制需求（如特殊的审计逻辑、复杂的权限映射）常发生冲突，如何把握平衡点？

2、资产的属主与管理职责问题。Agent的资产，如Prompt、Skills等的属主如何确定，是业务部门还是科技部门，谁来定标准？资产的管理执行端放在哪个团队，谁来负责全生命周期的维护？

3、记忆体的合规问题。记忆体中数据涉及跨部门（如零售与对公）的客户数据共享。在当前的隐私保护法规下，如何在合规的窄门中实现记忆共享？如何进行记忆体中数据的分类分级？

4、安全的“体验税”。如何在安全防控、建设成本与用户体验这‘不可能三角’中找到平衡点？

四、组织级的建设团队

Agent架构的落地无法依靠单一的开发团队，我们组建了包含架构专家、算法工程师、Prompt工程师与安全专家等的联合战队。

团队	关键职责	人员构成
管理团队	负责战略规划、场景决策、ROI 分析、小组协同等。	企业架构师、AI 战略专家等
平台团队	负责 L2 平台层的建设，维护 Agent 开发运营环境，建设 AI 网关与 MCP 协议标准。	架构师、SRE、DevOps 工程师
模型团队	负责基座模型的引入、微调、蒸馏，以及路由策略的优化。	算法工程师、数据科学家
数据团队	负责 L1 数据层的建设、非结构化数据治理，构建 FEBO 本体与知识图谱。	数据工程师、本体专家
资产团队	负责 Prompt 工程、知识库构建与 Skill 封装。这是连接业务与技术的关键桥梁。	Prompt 工程师、业务分析师(BA)、知识管理专员
安全团队	负责安全网关、防护栏等的构建，制定攻防策略、安全合规流程等	安全专家、合规官
应用团队	嵌入在各业务线（零售、对公、风控）的敏捷小组，负责具体 Agent 场景的最后一公里交付。	全栈工程师、产品经理

同时，为了解决产能瓶颈，我们以联合团队为核心，将业务人员也催动起来，以用促建，逐步培育Agent的生态。

五、结论

从“服务导向”到“智能导向”架构的演进，也是银行从“数字化”迈向“数智化”的必由之路。以企业级Agent架构为指引，逐步解决Agent落地过程中的幻觉、安全、集成等工程难题，更能构建起一套能够自我进化、自我优化的企业级智能操作系统。

在这个架构体系中：

- 数据层通过FEBO本体实现了对世界的“理解”；
- 平台层提供了强大的“动力”；
- 资产层沉淀了银行的“智慧”；
- 应用层重塑了“体验”；

- 安全层保障了“底线”。

只有系统化地构建这一工程，银行才能真正将AI Agent从“演示玩具”转化为驱动业务增长的核心引擎。

而“独行快，众行远”。企业级 Agent 架构的蓝区，仅靠一家银行的摸索是远远不够的。**我们也希望就其中一些课题，与TWT 社区的同仁们成立专题小组，**共同调研、分享实践或组织PoC验证，进一步找出更合适的答案，共同推进行业的发展。

> **支持社区支持本文同行观点，请点赞、转发或点击“♡”**
> **欢迎点击文末阅读原文，可以直接看到社区中本文中可能不包括的的全部信息和最新更新**

关联推荐：

AI 治理法律法规体系梳理

破茧与重生：AI趋势下三甲医院HIS系统更新换代的深度思考

AI智能体在银行业的高价值场景与现实挑战

回看2025：银行业打造大模型研发能力，构筑AI时代的护城河

欢迎关注社区 “**AI**”相关内容，了解最新行业同行专家的分享和大家的观点。地址：https://www.talkwithtrend.com/Topic/116059

*本公众号所发布内容仅代表作者观点，不代表社区立场

点击下方↙↙↙阅读原文，更丰富，更精彩

阅读原文