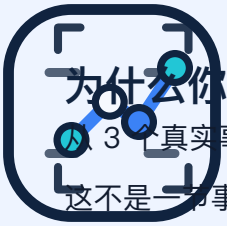
Week01｜Lesson01｜为什么你的 AI Demo 一上生产就翻车?

从 3 个真实事故，看懂企业 AI 的第一道生死线

本讲结构



为什么你的 AI Demo 一上生产就翻车?

从 3 个真实事故，看懂企业 AI 的第一道生死线

这不是一场事故八卦课，而是整门实战营的第一讲：先拆掉 Demo 幻觉，再建立 AI 系统交付视角。

返回 Week01 下一讲

下载讲义

提供适合离线阅读的 PDF 版和适合批注整理的 Word 版。

PDF 版 · 打印 / 离线阅读 Word 版 · 批注 / 二次整理

> ! 一句话结论
>
> 如果一个 AI 系统只能在受控样例里回答正确，却不能解释依据、约束动作、处理失败和明确责任边界，那么它仍然只是 Demo，不是生产系统。

现实冲突：为什么今天必须讲

今天不是普通导论，而是一节案例驱动的企业 AI 生产事故解剖课。

很多团队看见 Demo 能跑，就会默认后面只是工程化细节；但真实企业环境里，一旦进入更新、权限、PII、工具动作、监管和责任追溯，系统面对的是完全不同的世界。

这也是为什么 Lesson01 先不谈“用哪个模型更强”，而是先问：

- 事故最先从哪一层开始?
- Demo 阶段为什么没有暴露?
- 如果你是架构师，第一反应该补哪一层?

这也是为什么 Lesson01 不从模型、Prompt 和工具开始，而是先建立一套后面 15 周都会反复用到的事故拆解框架。
如果第一讲不先把“问题到底出在哪”讲清，后面的 Done、路线和工程基线都会显得太早、太重、太碎。

三个真实事故

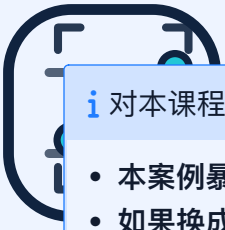
Air Canada chatbot misinformation liability

背景知识

- **这个公司做什么**
  Air Canada 是加拿大的大型航空公司，官网不仅是营销入口，也是正式的旅客服务入口，涉及购票、退改签、会员权益和特殊票务政策。
- **出事的产品是什么**
  出事的是官网上的客服聊天机器人。对普通旅客来说，它不是“实验工具”，而是和官网表单、帮助中心、人工客服并列的正式答疑入口。
- **事故是怎么发生的**
  一位旅客因为家庭成员去世，去官网询问丧葬票政策和报销方式。聊天机器人给出了一套看起来完整、可执行的说明，告诉用户可以先按普通流程购票、事后再申请相关优惠或退款。用户相信这是官网正式答复，于是按照这个建议行动。等到他后续凭机器人答复去申请时，Air Canada 才表示真实政策并不是机器人说的那样，最终争议进入裁决阶段，企业也不能再把责任推给“只是机器人说错了”。溯源链接：Moffatt v. Air Canada, 2024 BCCRT 149。

事故分析

- **事故表面是什么**
  旅客按照官网聊天机器人的说明行动，结果发现票务政策与真实规则不一致，企业最终要为错误信息承担责任。
- **详细分析**
  这个事故的关键，不是模型随机胡说，而是企业把一个没有被严格证据约束的回答通道，放进了正式服务链路。用户并不会区分“官网知识库”“搜索结果”“AI 生成内容”之间的内部差异，他只会判断：这是你官网里的正式答复。于是，一次看似普通的 FAQ 错答，立刻升级成合同解释、赔付争议和企业责任问题。真正的危险，是企业把“回答能力”上线了，却没有把“证据、版本、责任边界”一起上线。
- **真正失控的是哪一层**
  数据层先失控，随后检索层和证据约束缺失把风险放大了。
- **为什么 Demo 阶段没暴露**
  Demo 一般只会验证“能否答出一个像样的政策说明”，不会压测高风险政策条款、例外条件、引用来源和后续赔付责任。



对本课程的直接启示

- **本案例暴露的第一缺口：** 高风险政策问答缺少证据绑定与责任边界。
- **如果换成我们的课程项目，Week01 先要建立的意识：** 高风险回答必须来源受控、引用可追溯、必要时转人工。
- **后续哪些周会系统补齐：** Week02 输入边界，Week07/08 证据链与检索，Week11 评测。

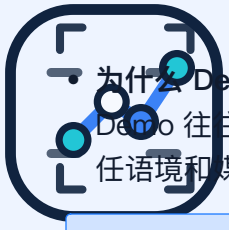
NYC MyCity chatbot wrong / potentially illegal advice

背景知识

- **这个公司做什么**
  MyCity 是纽约市面向小企业和市民提供公共服务信息的数字入口之一，背后承载的是政府服务、政策解释和办事引导。
- **出事的产品是什么**
  出事的是 MyCity 集成的官方聊天机器人。用户会拿它当成政府办事助手，而不是普通搜索框。
- **事故是怎么发生的**
  这起事故不是某一个用户偶然试出一个错误答案，而是媒体和研究者在它上线后持续拿真实业务问题去询问，例如租赁规定、最低工资、用工合规、商业限制等。结果发现，它会给出措辞很肯定、但与真实法规不一致，甚至可能把用户引向违法操作的回答。由于它嵌在官方服务体系里，这些错误回答天然带着“政府解释”的权威感，事故就从回答质量问题迅速升级成公共治理问题。溯源链接：Audit Report on the New York City Office of Technology and Innovation’s MyCity System。

事故分析

- **事故表面是什么**
  官方聊天机器人给商户和市民提供了错误、甚至可能违法的政策建议。
- **详细分析**
  MyCity 这个案例最值得讲的地方，是它说明了“权威场景”对 AI 的要求远高于“普通问答场景”。当系统在官方入口里承担政策解释角色时，错误回答不仅误导用户，还会破坏公共信任，甚至制造合规风险。这里的工程缺口不是“模型不够强”，而是没有把法规来源、证据绑定、拒答边界、人工接管和持续观测做成硬约束。系统上线了回答能力，却没有上线公共责任控制面。
- **真正失控的是哪一层**
  检索层先失控，随后治理/观测层没有及时阻断高风险公共场景。



- **为什么 Demo 阶段没暴露**
  Demo 往往只用低风险示例证明“能回答政策问题”，不会在真实法规变动、歧义案例、责任语境和媒体压力下持续压测。

> 对本课程的直接启示
>
> - **本案例暴露的第一缺口：** 官方高风险问答缺少法规来源约束、拒答边界和人工接管。
> - **如果换成我们的课程项目，Week01 先要建立的意识：** 权威场景里的回答不能只追求“像正确”，必须先讲清证据、边界和升级路径。
> - **后续哪些周会系统补齐：** Week02 输入边界，Week07/08 检索与证据链，Week12 治理与观测。

DoNotPay deceptive AI lawyer claims / FTC order

背景知识

- **这个公司做什么**
  DoNotPay 早期以帮助用户处理罚单、订阅取消和基础法律流程为卖点，后来逐步把自己包装成更广义的 AI 法律服务工具。
- **出事的产品是什么**
  出事的不是单一聊天页面，而是整个“AI lawyer / 机器人律师”能力主张，以及围绕这一主张提供的法律建议与服务包装。
- **事故是怎么发生的**
  这起事故不是因为某次回答被截屏嘲笑，而是因为 DoNotPay 长期把产品往“可以替代专业法律服务”的方向宣传。随着用户规模变大、监管关注提高，问题开始变成：它到底有没有证据证明自己能胜任这种高风险服务？有没有把高风险动作隔离出去？有没有向用户清楚说明边界？FTC 最终介入并下达命令，核心不是某个模型答错一句话，而是整套能力宣称、服务边界和监管责任之间长期错位。溯源链接：FTC Finalizes Order with DoNotPay That Prohibits Deceptive “AI Lawyer” Claims。

事故分析

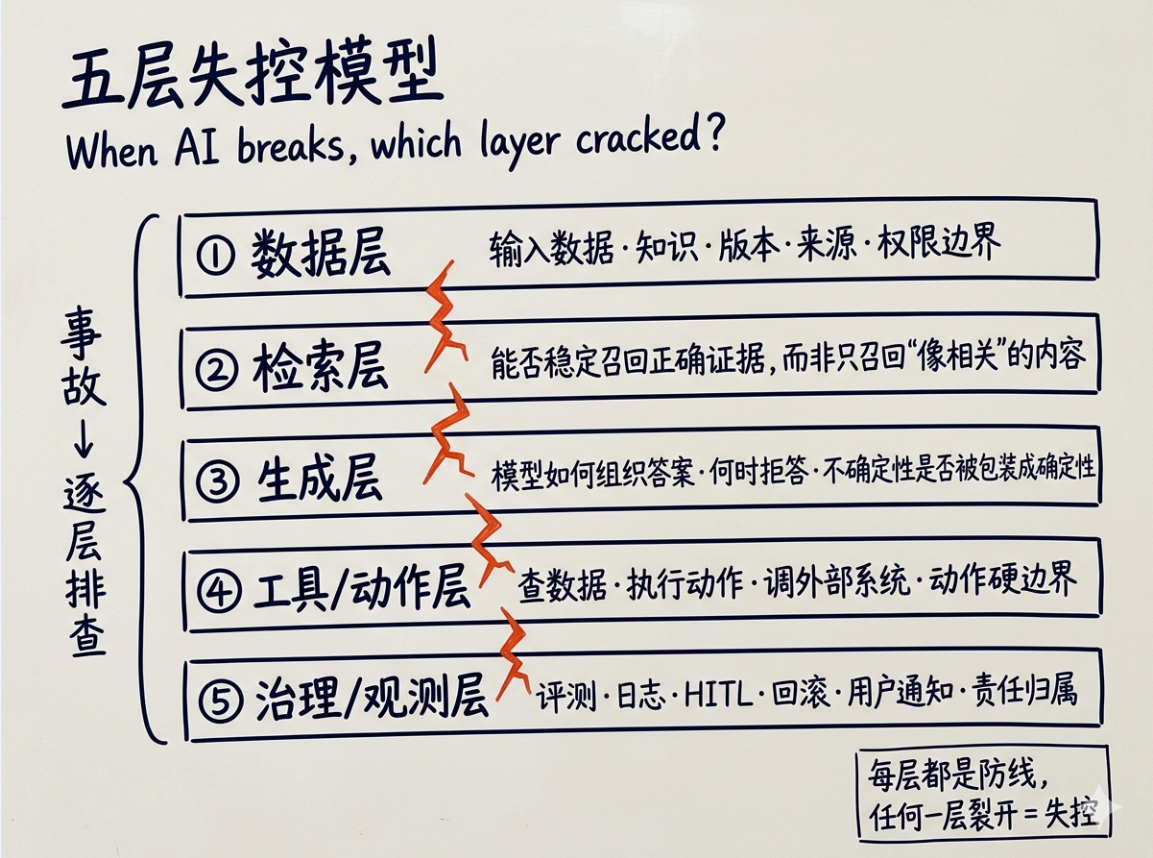
- **事故表面是什么**
  产品把 AI 能力包装成接近“机器人律师”的专业服务，最终引来 FTC 监管命令。
- **详细分析**
  这个案例提醒你，企业 AI 翻车不一定先从生成层开始，也可能先从“能力宣称”开始。只要产品对外承诺超过了系统真实能力，且又缺少证据、评测、人工复核和责任边界，系统就会在发布层面先违规。换句话说，这不是“功能跑没跑通”的问题，而是“你到底向用户承诺了什么、有没有资格做这种承诺”的问题。

- **真正失控的是哪一层**
  工具 / 动作层和治理 / 观测层共同失控，产品承诺越过了真实系统能力和监管边界。
- **为什么 Demo 阶段没暴露**
  Demo 只能证明功能局部成立，证明不了产品有资格承担专业服务责任，更证明不了营销和上线口径是合规的。

> ℹ 对本课程的直接启示
>
> - **本案例暴露的第一缺口：** 产品能力主张越过了真实系统能力和监管边界。
> - **如果换成我们的课程项目，Week01 先要建立的意识：** 上线前先划清“能辅助到哪、必须人工到哪、绝不能自动到哪”。
> - **后续哪些周会系统补齐：** Week09/10 工具与动作边界，Week12/15 治理、上线标准与责任边界。

统一事故解剖框架

固定使用五层失控模型来拆事故：



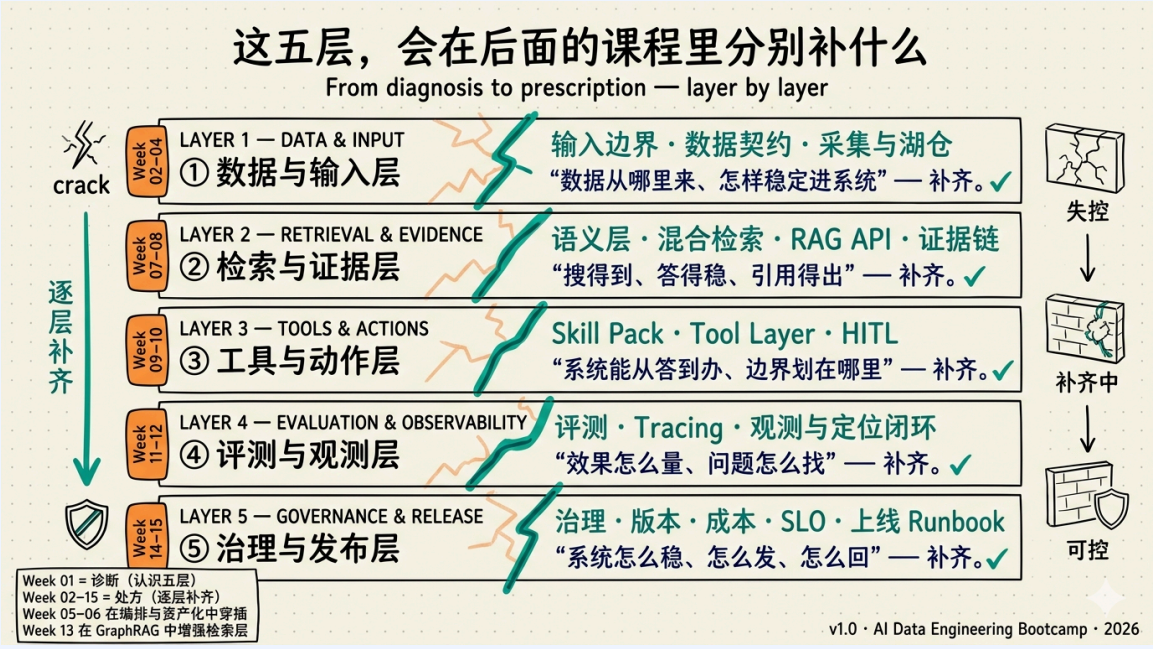
以后遇到 AI 事故，不先问参数，不先问换模型，而是先问：

到底是哪一层链路失控了？

为了让这套方法真正可用于复盘，我会把动作固定成这个顺序：

1. 先补背景事实和责任语境
2. 再看表面事故
3. 再判断哪一层先失控
4. 再判断哪一层把伤害放大
5. 最后再决定工程动作

这五层，会在后面的课程里分别补什么：



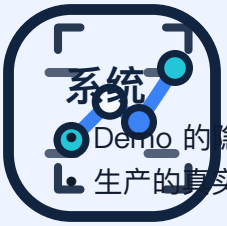
Demo 世界 vs 生产世界

数据

- Demo 的隐藏假设：静态样例、手工准备、无版本差异
- 生产的真实约束：持续更新、口径漂移、权限和 PII

用户

- Demo 的隐藏假设：提问者友好、上下文完整、不会故意试边界
- 生产的真实约束：问题歧义、行为分布极广、风险用户会直接撞边界



- Demo 的隐藏假设：单人掌控、单路径运行、失败可口头兜底
- 生产的真实约束：跨系统协作、高并发、失败需要系统级处理

治理

- Demo 的隐藏假设：先做出来再说
- 生产的真实约束：评测、观测、责任边界、发布与回滚都必须前置

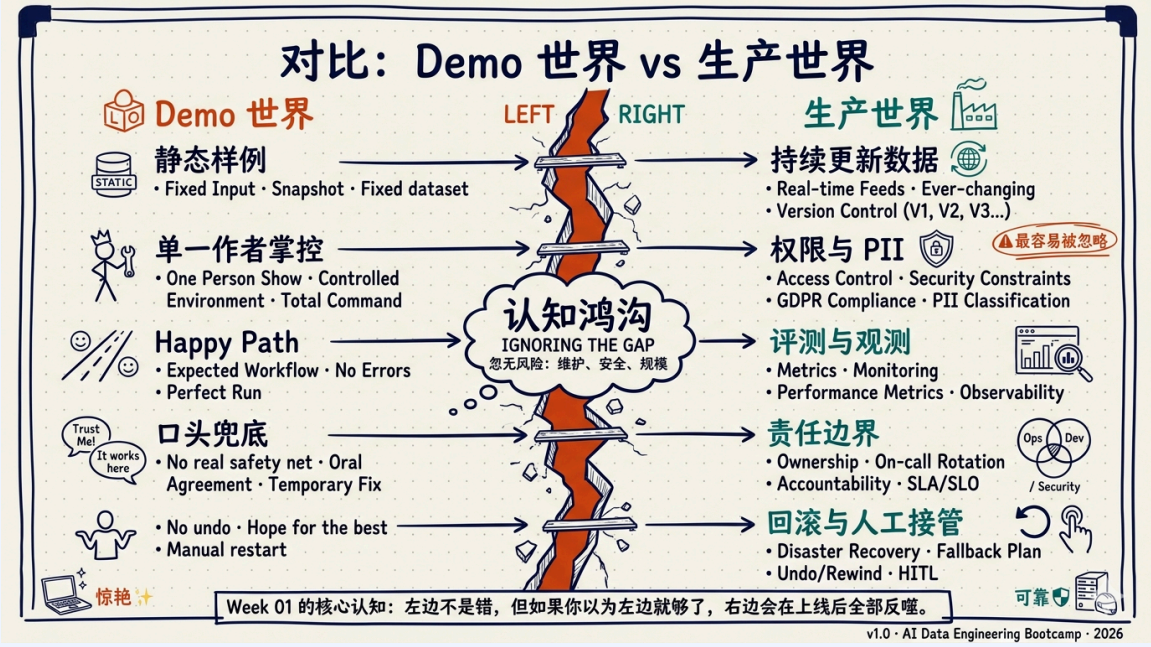


Figure 1

一旦你接受 Demo 世界和生产世界不是同一个世界，下一步就必须问：
企业 AI 想从 0 到 1 真的走到上线，交付链上到底要补哪些环节？

企业 AI 从 0 到 1 的交付链

后面 15 周不是 15 个散点，而是在补这一条固定主链：

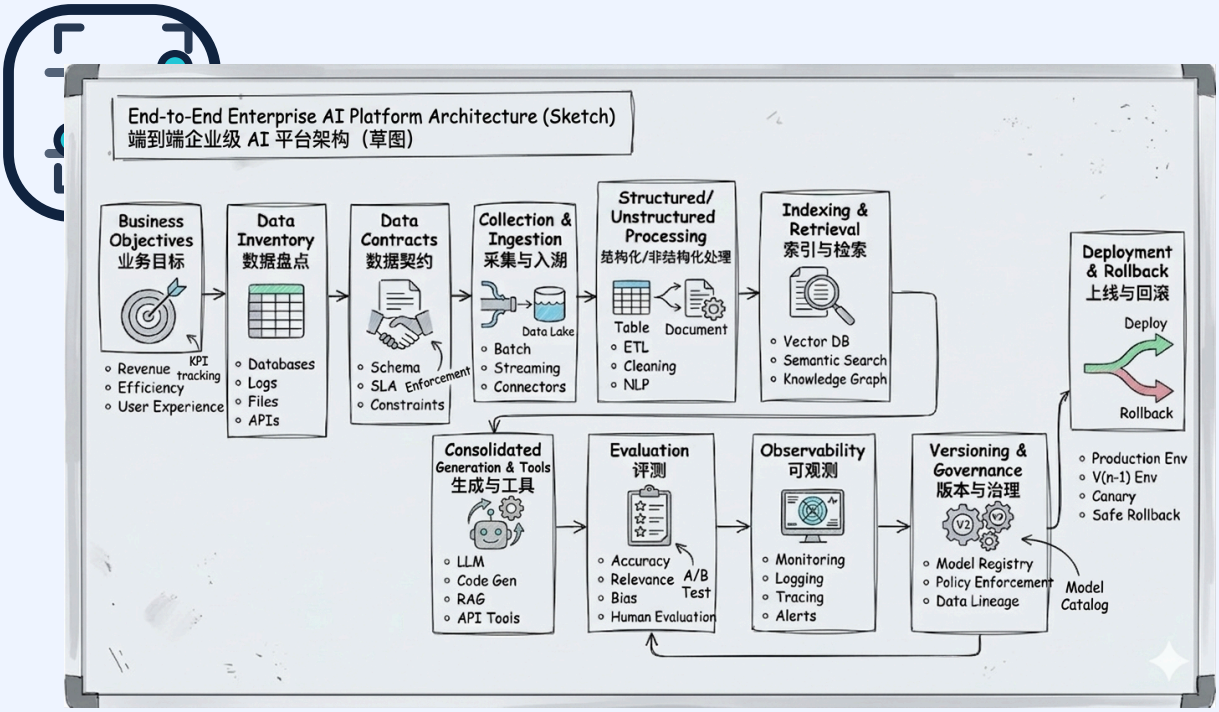
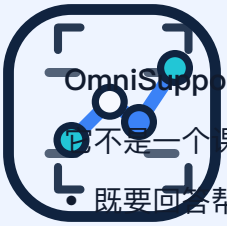


Figure 2

> ℹ 图解说明
>
> 这张图真正要让你记住的，不是名词本身，而是顺序：
>
> 业务目标
> → 数据盘点
> → 数据契约
> → 采集与入湖
> → 结构化/非结构化处理
> → 索引与检索
> → 生成与工具
> → 评测
> → 可观测
> → 版本与治理
> → 上线与回滚

主案例：客服知识库 + 工单联动 Copilot

为了把前面的事故、五层失控模型和交付链第一次压回真实场景，这门课会反复回到同一个贯穿案例：



OmniSupport Copilot。

它不是一个课堂演示用的“知识库问答 Demo”，而是一个更接近企业实战的系统原型：

- 既要回答帮助中心、FAQ、Release Notes、API 文档里的规则性问题
- 也要理解工单、评论线程、状态字段等实时业务事实
- 还要在受控条件下完成工单查询、创建、更新等动作
- 并且全过程要具备证据引用、权限约束、审计字段、版本与回滚意识

换句话说，这个案例的价值，不在于“它能不能回答问题”，而在于：

> ! Important
>
> 它天然把企业 AI 最容易失控的几个关键难点，同时摆在了同一个系统里。

一个看似简单的问题，背后往往会同时牵出：

- **规则知识**
  产品说明、FAQ、操作文档、Release Notes 到底怎么解释
- **实时工单事实**
  当前用户的工单状态、历史处理记录、最新评论、升级节点是否一致
- **权限边界**
  当前提问者是否有权看到这条记录、这类指标、这类内部说明
- **动作边界**
  系统到底只能“建议”，还是可以真正“创建 / 更新 / 推进工单”
- **人工介入（HITL）**
  一旦遇到高风险、高不确定性或越权场景，系统应当在什么位置停下来并转人工

这也是为什么我不想把整门课讲成一个“纯 FAQ 问答机器人”项目。
因为真实企业里的支持知识，不会只存在于文档里，还会同时散落在工单、对话、音频转写、视频教程、截图、错误日志和支持流程里。
如果课程项目只是一套“PDF + 向量库 + LLM”的演示链路，那么后面关于数据契约、湖仓、评测、Tracing、版本治理和回滚的内容，就会全部悬空。

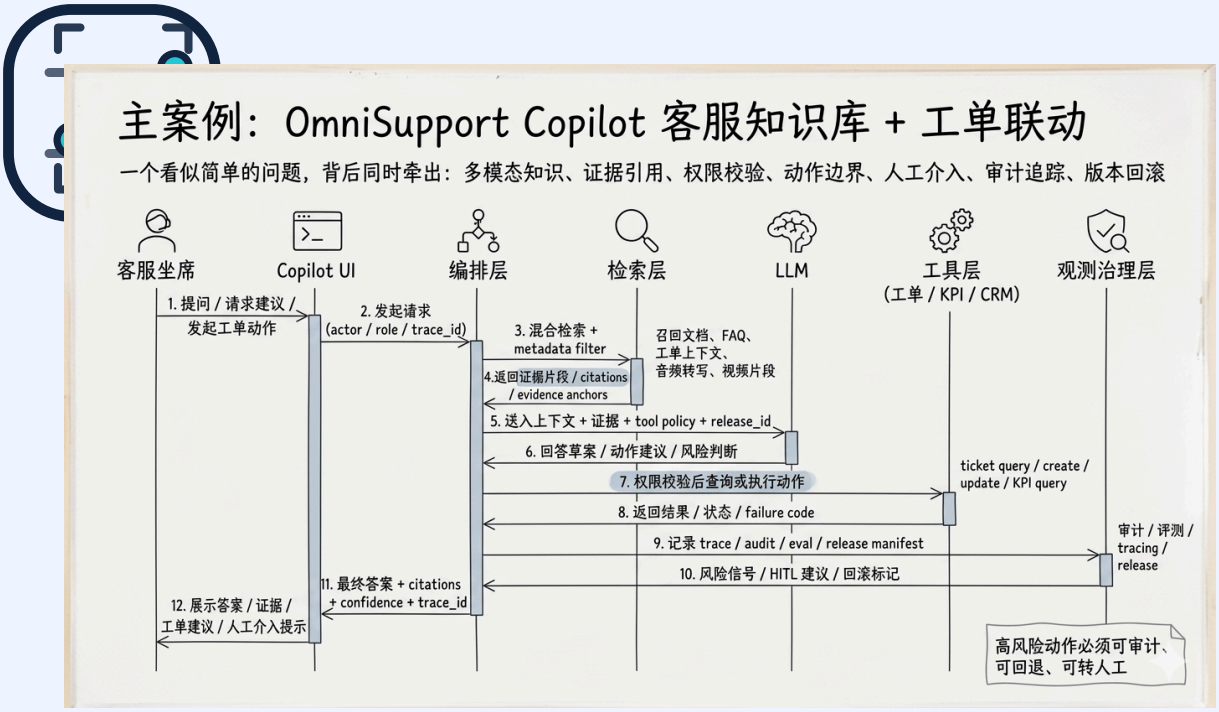


Figure 3

Figure 3：OmniSupport Copilot 的受控请求链。它不是单点问答，而是一条知识、检索、生成、工具与审计协同工作的交付链。

你现在不需要记住每个组件名字，但要先记住这个系统的本质：

它不是单点问答，而是一条“知识 → 检索 → 生成 → 工具 → 审计”的受控交付链。

这个案例在 Week01 的意义，不是把系统做复杂，
而是让我们第一次把前面讲过的“生产级 AI 必须前置的约束”压到实践里：

- 项目工程基线要先立住
- PII 分级与处理方式要先讲清
- 可执行 / 不可执行动作要先划线
- HITL 节点要先定义
- 风险边界文档要先写
- 《AI 系统落地蓝图》要先产出

所以从这一讲开始，你要把 OmniSupport Copilot 看成：

这门课的方法论验证器、后续 14 周的统一练习场，以及整门实战营唯一的工程基线。

今天行业的主流答案

今天行业真正收敛的，不是“再调一轮 Prompt”，而是下面四件事：

1. 评测要前置
2. 观测要前置
3. 风险边界要前置
4. 能力主张要和证据绑定

基于公开资料，我这里引用两个非常贴近本讲主题的信号：

- McKinsey 在 2024 年的 AI 状态调查里提到，已有 78% 的受访组织在至少一个业务功能中使用 AI，但从试点走向规模化交付仍是难点。
- NIST 的 Generative AI Profile 强调要把 trustworthiness 纳入设计、开发、使用和评估。
- OWASP 2025 LLM Top 10 继续把 prompt injection、敏感信息泄露等风险放在企业 AI 的关键风险面上。

这四件事，恰好也是后面课程为什么这样排的原因：
评测会在 Week11 系统补齐，观测在 Week12，风险边界会在 Week01、Week10 和 Week14 持续加固，证据绑定会在 Week07—08 变成硬能力。

> ℹ 延伸来源
>
> - McKinsey, The state of AI in early 2024
> - NYC Comptroller, Audit Report on the New York City Office of Technology and Innovation's MyCity System
> - FTC, FTC Finalizes Order with DoNotPay That Prohibits Deceptive "AI Lawyer" Claims
> - NIST, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile

会演示 vs 可上线

维度固定用 10 个判断：

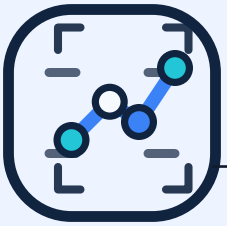


Table 1

维度	会演示	可上线
数据	静态样例	有版本、更新和口径边界
检索	能召回一点内容	证据稳定、可解释、可追溯
输出	看起来像能答	有拒答、置信与风险边界
权限	默认无边界	用户、租户、PII 分层明确
工具	演示动作可跑	参数受控、动作可审计
评测	靠人工感觉	有指标、用例和回放
观测	不出错就算好	请求、证据、动作、失败可观测
回滚	出事再说	有降级、停机和回退路径
成本	不太关心	延迟、推理和资源成本可管理
责任边界	口头说明	owner、升级路径和治理边界清楚

最后只收一句：

> ! Important
>
> 能演示，只说明局部成立。可上线，要求链路可控。

Launch Readiness Scorecard

```
//| echo: false
dimensions = FileAttachment("../../assets/lesson01/readiness-dimensions.json").json()
viewof readinessMain = Inputs.checkbox(dimensions, {label: "勾选你已经具备的上线能力", format: d => d.label, value: []})
mainScore = readinessMain.length * 10
mainBand = mainScore >= 80 ? "green" : mainScore >= 50 ? "yellow" : "red"
mainBandLabel = mainBand === "green" ? "Green · 可进入上线评审" : mainBand === "yellow" ? "Yellow · 仍需补齐关键边界" : "Red · 仍停留在 Demo 阶段"
```

```
//| echo: false
html`<div class="grid-two lesson01-stress-card">
  <div class="info-card">
    <h3>Readiness 分值</h3>
    <p style="font-size:2.4rem; font-weight:700; color:#103f7d; margin:0 0 0.35rem;">${mainScore}</p>
    <p><strong>风险等级：</strong>${mainBandLabel}</p>
  </div>
```

```
  <div class="info-card">
    <h3>缺失能力解释</h3>
    <ul>
    ${dimensions.filter(d => !new Set(readinessMain.map(x => x.id)).has(d.id)).map(d => html`<li>${d.label}: ${d.detail}
</li>`)}
    </ul>
  </div>
</div>`
```

ℹ 如果你的分数偏低，后面优先补哪几讲?

- **数据 / 权限 / PII 缺口大** → 优先看 Week02
- **证据 / 检索 / 输出结构缺口大** → 优先看 Week07—08
- **工具 / 动作 / HITL 缺口大** → 优先看 Week09—10
- **评测 / Tracing / 回滚缺口大** → 优先看 Week11—14

本讲小结 + 思考题 + 下一讲导航

固定收 3 条 takeaways：

1. AI 翻车通常不是模型单点失误，而是链路共同失控。
2. 后面 15 周是在补企业 AI 的交付链，而不是堆技术点。
3. 下一讲必须先定义 Done 和验收标准，而不是先选工具。

既然问题本质上是交付链失控，那么下一步最不能靠感觉推进的，就是“到底怎样才算完成”。
所以下一讲不先讲工具，而是先定义 Done 和最小可签字的交付标准。

思考题

1. 你当前项目里，最像哪一种真实事故?
2. 如果按五层失控模型拆，你最先会补哪一层?
3. 你的系统今天为什么还不能算“可上线”?

返回本周 下一讲