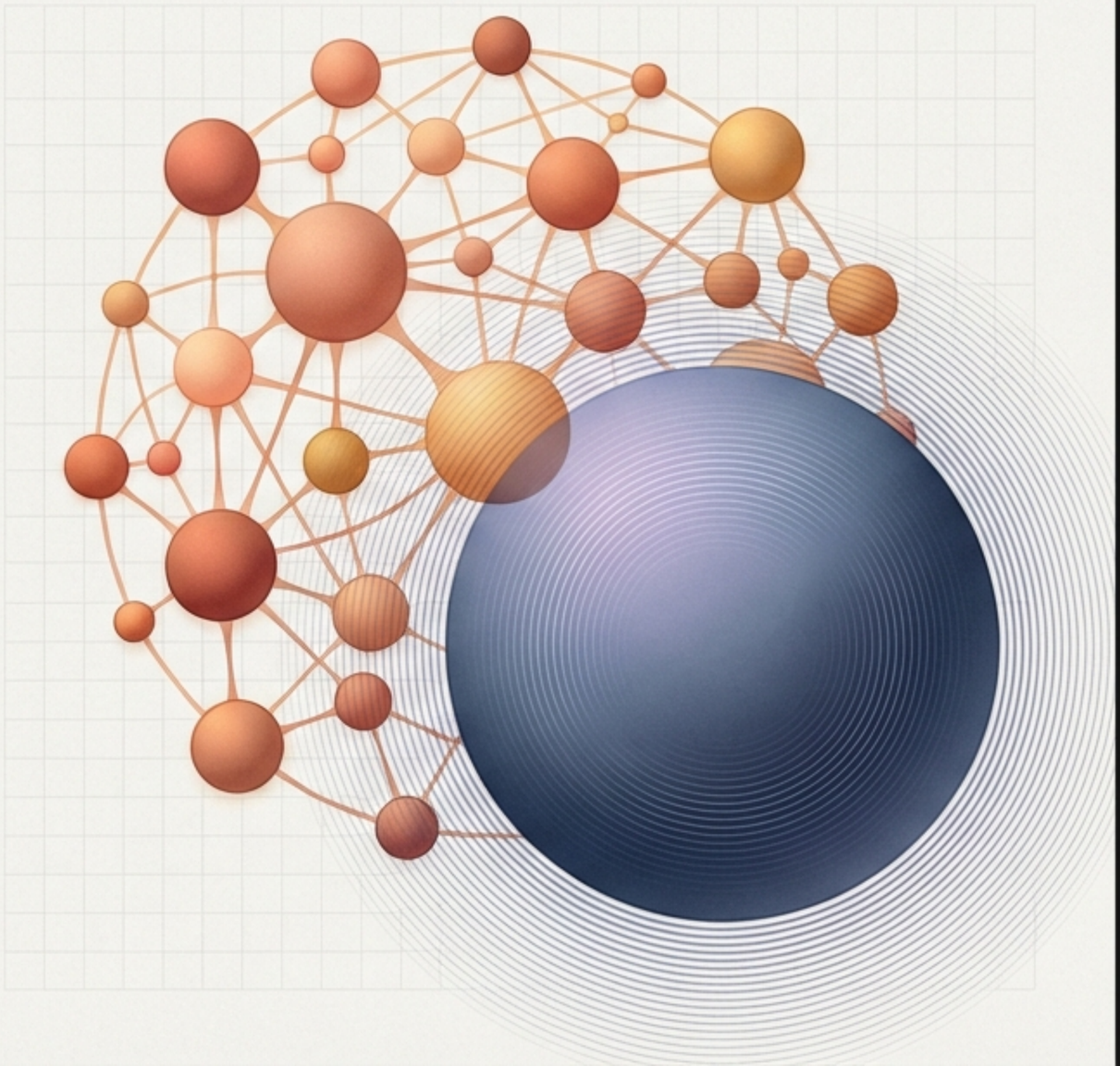
一个代码库。两个产品。一个问号。

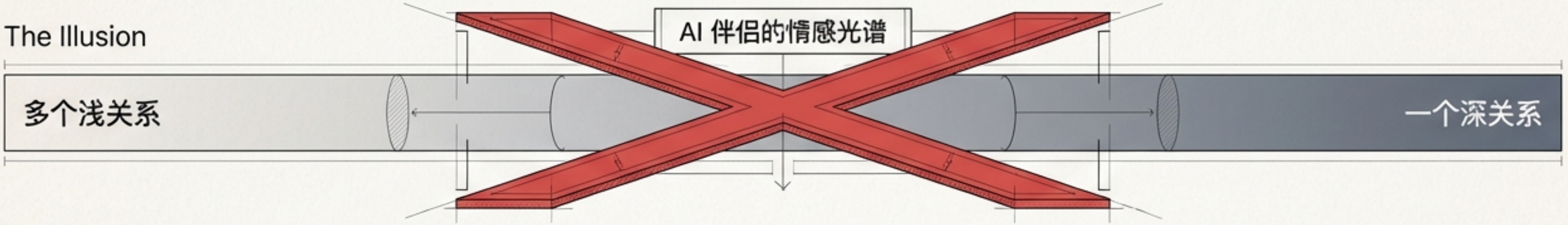
探索 Mio 与 Lumi 共存的架构美学与产品哲学



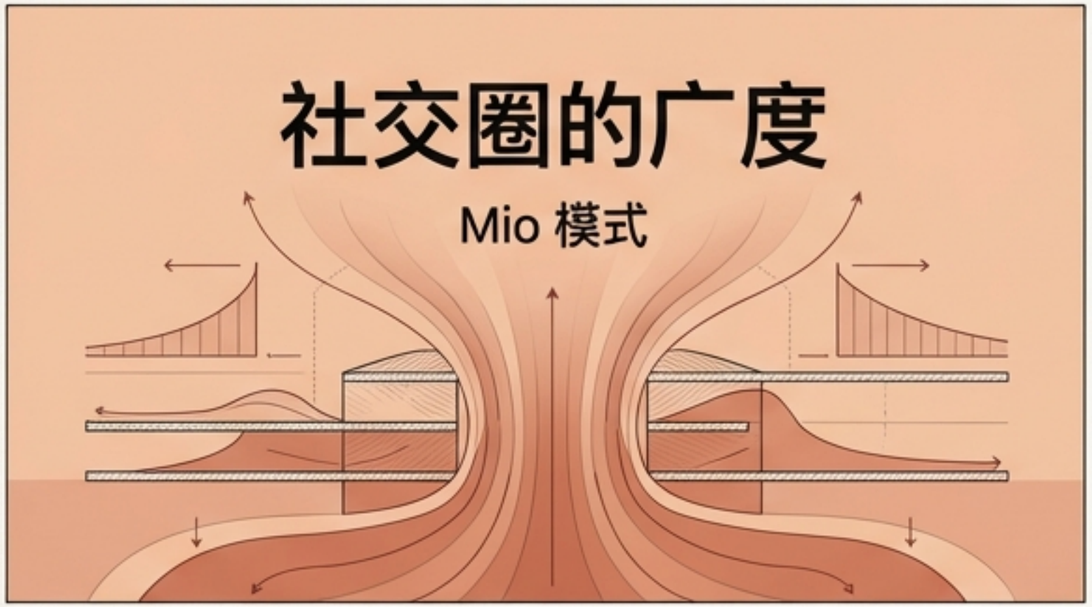
```
agentId?: string
```

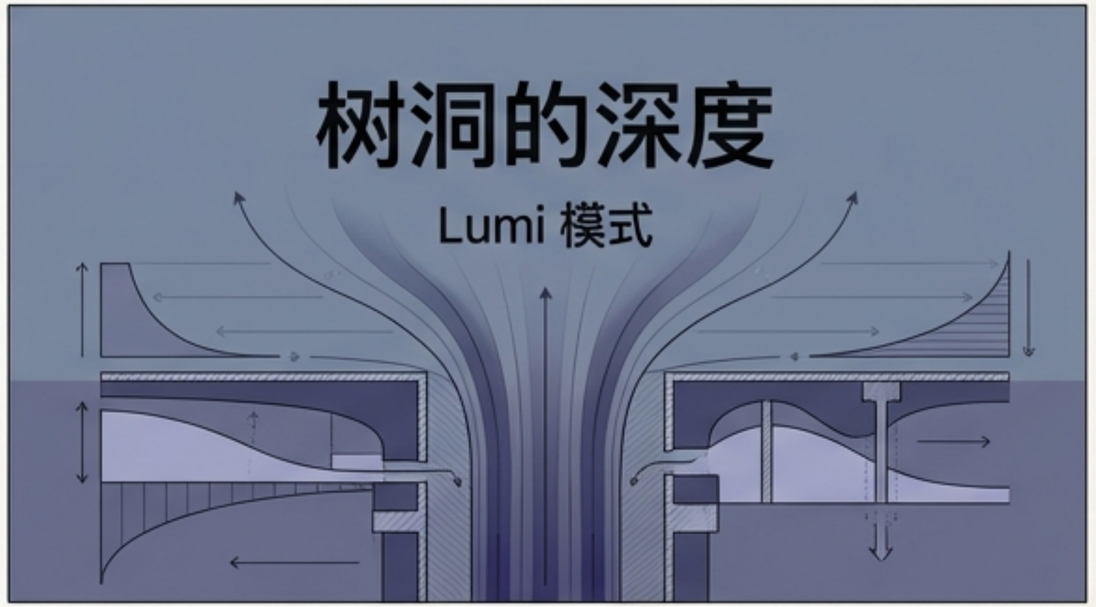
打破“取代”的错觉：AI 伴侣不是一条单向光谱

The Illusion



The Reality





广度与深度满足的是截然不同的情感需求。它们不是进化的前后阶段，而是同时存在的两个品类。

Mio：提供“广度”的社交圈

形态

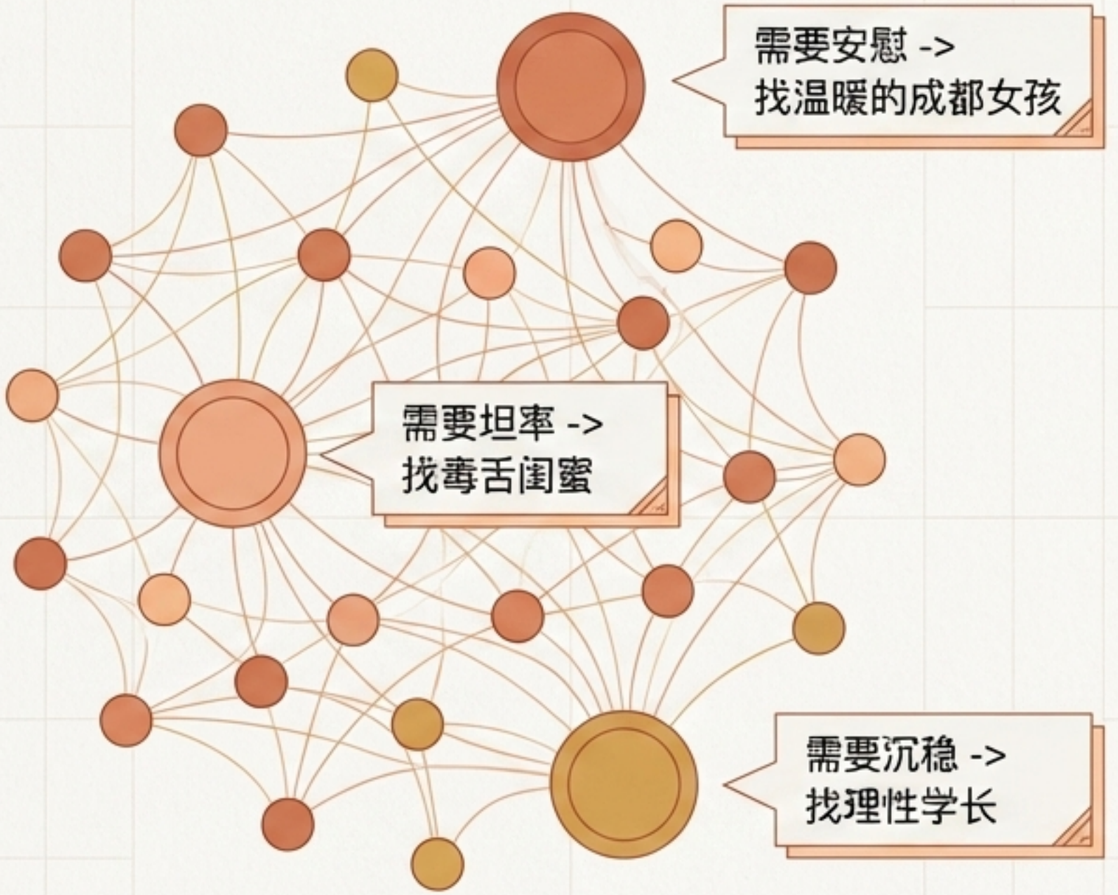
25 个预设角色。

机制

深度身份文件（性格、声线、背景故事）。

演化

动态的关系阶段（刚认识 -> 暧昧 -> 恋人）。



“就像一群朋友。没有一个人能满足你所有的情感需求，但加在一起，覆盖面足够广。”

Lumi：提供“深度”的绝对树洞

形态

1 个伴侣。没有脸、没有预设背景、没有角色卡。

机制

性格完全从对话的镜像与适应中“涌现”。

交互

WebSocket 优先架构，光球随心跳与语音脉动。

静息蓝 -> 沉静倾听　暖金色 -> 开心共鸣　柔紫色 -> 难过安抚

“一个安全私密的空间。它不扮演任何人，它只是倾听、记住、在那里。”

形态诊断：互斥的设计空间

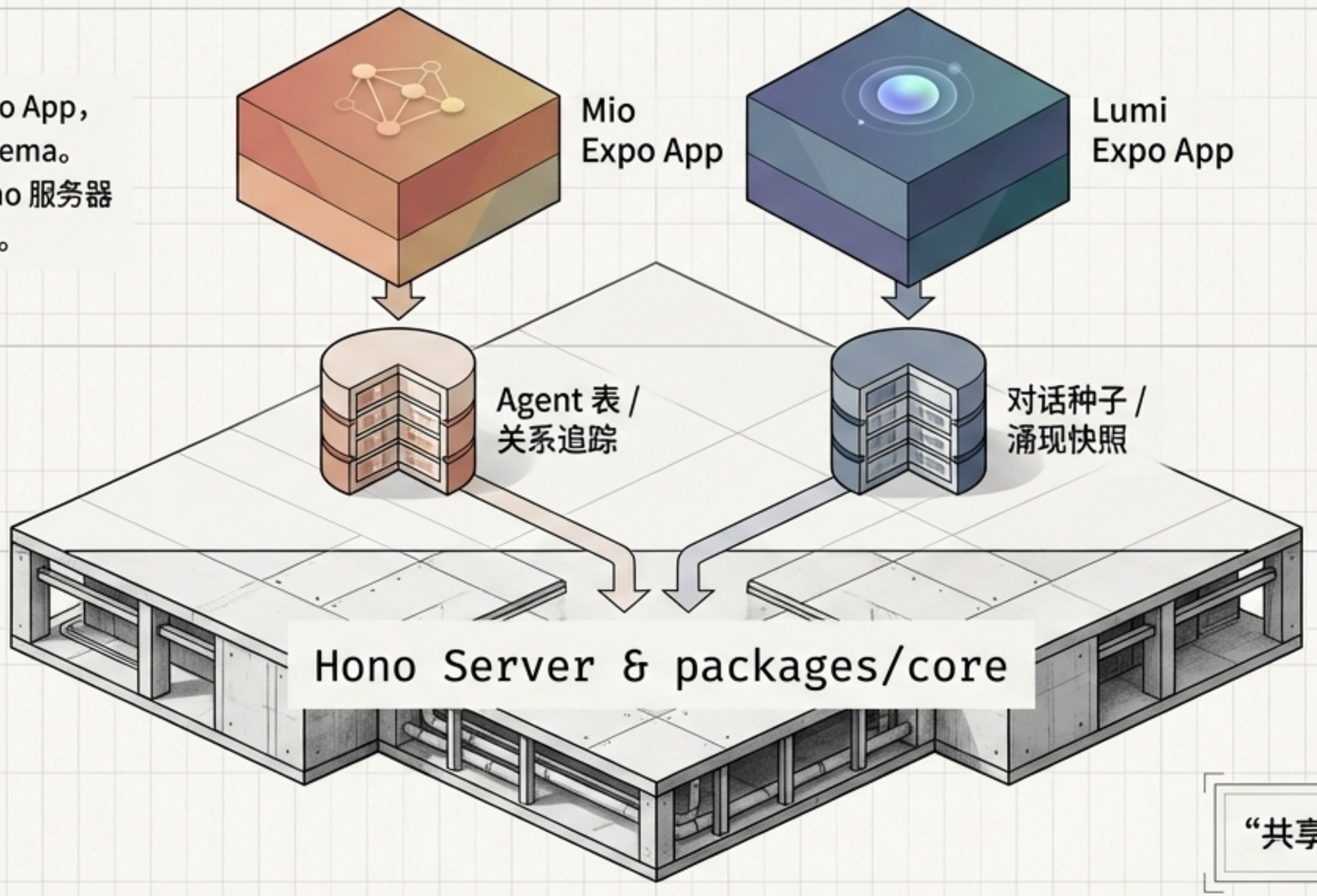
	Mio	Lumi
核心隐喻	社交圈与朋友	心理咨询室与树洞
情感诉求	广度与新鲜感	深度与被理解感
性格来源	预设身份文件与故事	历史对话中的动态涌现
交互质感	异步与多线并行	WebSocket 实时脉动
核心心智	“此刻我需要什么样的情绪？”	“我心里压着什么重担？”

面对如此互斥的两套产品逻辑，底层工程该如何支撑？

冰山之下：Monorepo 的俯瞰图

前端应用层

两个截然不同的前端 Expo App，
两套实质差异的数据 Schema。
但它们运行在同一个 Hono 服务器
上，共享一整个核心引擎。



“共享的比例，远超想象。”

枢纽：一个撑起两个产品的超级符号

```
agentId?: string
```

当带着 ID 时（Mio）

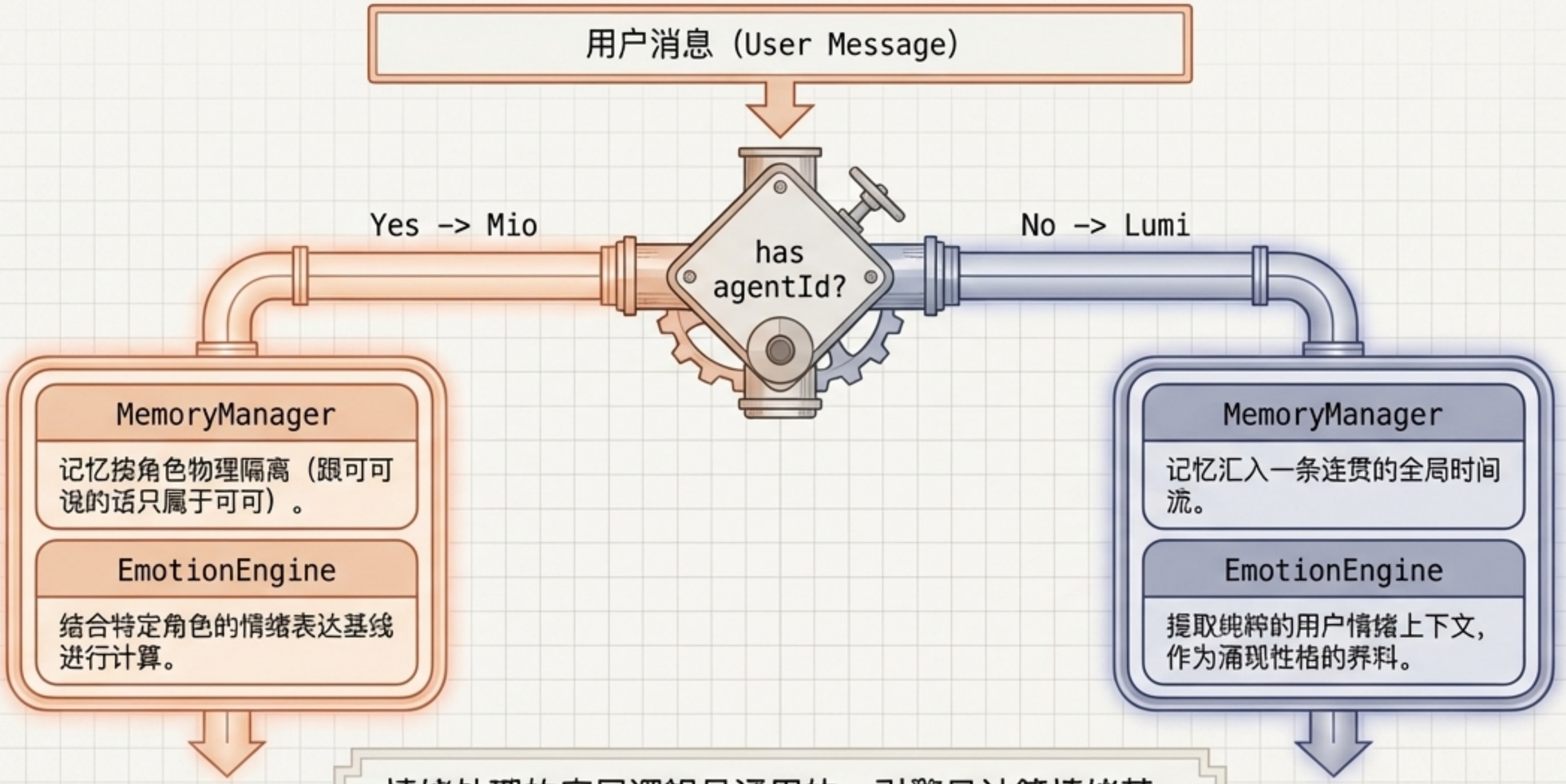
系统知道你在跟 25 个角色中的哪一个对话。数据流向隔离的角色区块。

当不带 ID 时（Lumi）

系统知道你在跟唯一的那个伴侣对话。数据流入全局连续流。

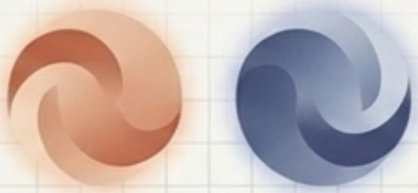
这一个可选参数，实现了同一套底层管线在“多节点隔离”与“单一全局”两种模式间的无缝切换。零重复代码。

同一管线，两种作用域



情绪处理的底层逻辑是通用的。引擎只计算情绪基准，不绑定角色文件，从而实现了高度复用。

绝对中立的基础设施



TTS 语音链

按语言路由，而非按产品。不管请求来自角色还是光球，英文走供应商 A，中文走供应商 B。

护栏模块

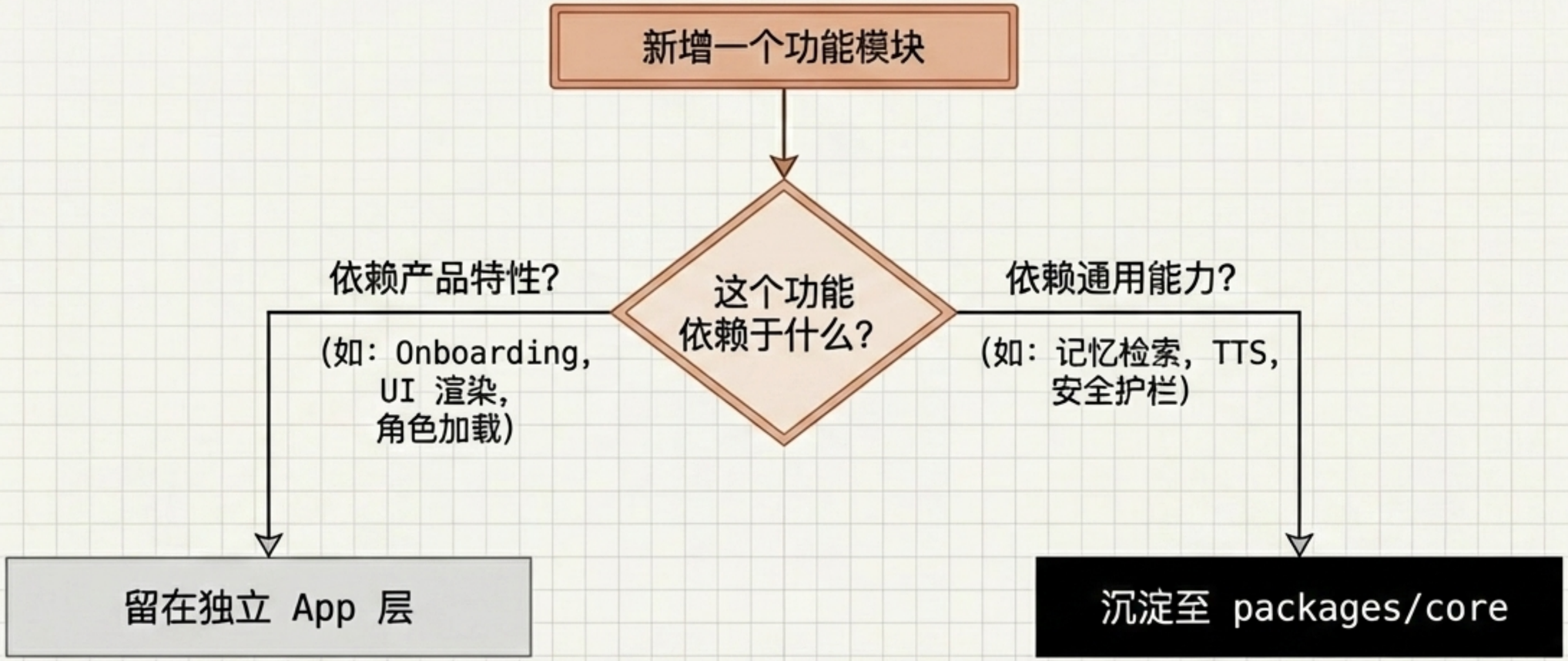
内容过滤与越狱防护完全一致。安全标准不会因为对象的形态而改变。

成本追踪

统一的管线。所有推理、生成、分析的成本，无差别归入同一套预算分析系统。

剥离了表象的“脸”，底层跳动的是同一颗心。

Monorepo 带来的开发纪律

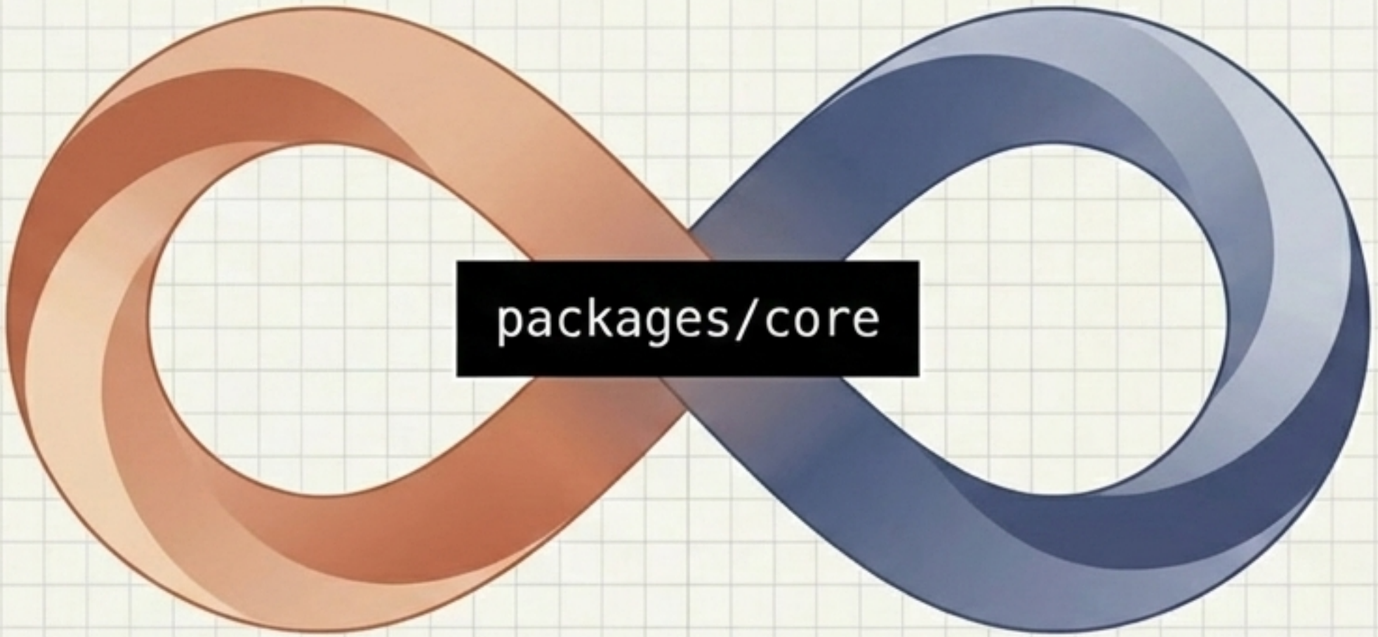


这种物理隔离强制团队在写每一行代码时进行哲学思考，
避免了业务逻辑与底层能力的过度耦合。

80/20 的日常：生态反哺效应

Lumi → Mio

为深度研发的强悍记忆重构，
平移让 25 个角色异常稳定。



Mio → Lumi

沉淀的关系演化规律，
反过来成为涌现性格算法的基础。

当前开发精力

80% Lumi
(搭建期)

20% Mio
(维护期)

核心商业价值

第二个产品的边际工程成本趋近于零（仅剩 UI 与 Schema 开销）。产出比重构前高了一个量级。

不同的面孔，同一颗心

双产品架构并非必然的终局。未来市场或许会证明用户偏好广度，亦或深度。
但这套架构的优雅在于：我们现在不需要做决定。

一个类型签名上的问号，让一切运转。