CHAPTER
第1章

新的产品形态：大模型产品的出现与能力边界

“人工智能”（Artificial Intelligence，AI）的概念极为广泛，本书主要围绕“大模型”（Large Language Model，LLM）这一狭义领域展开。本章作为全书的基础认识篇，主要目标是说明大模型产品的发展脉络，理清它与传统互联网产品的差异边界，使读者对大模型产品这一新兴事物建立清晰、系统的基础认识。

本章的历史叙事，主要从宏观角度出发，说明大模型技术对生产力的提升作用，及其对既有产品形态的重塑。

考虑到本章围绕基础概念、发展背景展开，概念密度较高，读者可根据自身兴趣选择性阅读。若更偏好实际技术，也可直接从第 4 章开始，学习技术之余再抽空回过头来看这部分内容。

1.1 大模型产品的发展脉络

本节以时间轴为脉络，依次梳理人工智能技术、互联网产品的关键历史事件及其影响；在此基础上，进一步剖析人工智能与互联网产品的双向作用，以及伴随这一互动过程产生的生产要素变革。

1.1.1 人工智能的技术发展

尽管技术的发展看似充满历史的偶然性，但其发展从来不是一蹴而就的，人工智能的演进同样如此，我们通过历史来感受人工智能的演化过程。

1940—1975 年：理论萌芽与人工智能的首次实践

现代计算机的冯•诺依曼架构与“图灵测试”的概念，均在 20 世纪 40—50 年代已经提出。1956 年，达特茅斯会议正式确立了“人工智能”这一术语。在此后的二十多年里，研究者们在模式识别、早期对话系统、自动控制、自动驾驶等方向进行了大胆尝试和探索。

然而，受限于算力、数据与算法理论的不成熟，这些早期系统的性能远未达到预期。过高的期望与现实之间的落差，使得人工智能在 70 年代中期和 80 年代末两次陷入发展低谷，即所谓的“AI 寒冬”。

1976—2000 年：理论沉淀和人工智能的二次实践

尽管经历了寒潮，人工智能并未完全停滞。相反，这一时期成为理论与方法的重要积蓄期。尤其是 20 世纪 90 年代，反向传播算法被重新审视并广泛应用于多层神经网络训练，为后来的深度学习复兴奠定了基础。与此同时，统计学习理论、支持向量机等方法逐渐成熟，推动人工智能从“纯符号推理”向“数据驱动”的方向转变。

2001—2010 年：数据、算力与算法要素发展

人工智能的发展离不开相关领域的协同进步。进入 21 世纪后，互联网的普及带来了海量的文本、图像与行为数据，为模型训练提供了前所未有的“燃料”；游戏与图形渲染需求推动了 GPU 的快速发展，使其逐渐成为高效的并行计算平台，为后续大规模深度学习训练提供了关键算力支持；再加上此前已逐步成熟的算法理论，人工智能的三大要素——数据、算力与算法在这一时期开始“集齐”。

2011—2020 年：人工智能应用落地

2010 年之后，人工智能迎来了从实验室走向产业的关键阶段。2012 年，AlexNet 在 ImageNet 图像分类任务上取得突破性成果，标志着深度学习在视觉识别领域的成熟；2016 年，AlphaGo 战胜围棋世界冠军李世石，让人工智能的能力得到全球关注。此后，图像识别、语音识别、自然语言处理等技术在安防、金融、互联网服务等行业迅速落地，人工智能开始真正改变人们的生产与生活方式。同一时期，Transformer 架构的提出证明了规模效

应（Scaling Law）的巨大潜力，预示着大模型时代即将到来。

2021 年一：大模型新时代

在长达半个多世纪的理论沉淀与技术实践之后，人工智能在 2020 年后迎来了新的爆发期。以大规模预训练模型为代表的“大模型”技术，使得人工智能在通用任务上具备更强的迁移能力，并逐渐成为“人工智能”的代名词。

2022 年 11 月 ChatGPT 的发布，标志着大模型技术正式进入大众视野。此后，生成式 AI、智能助手、代码生成等应用层出不穷，人工智能以前所未有的速度渗透到各行各业，也为产品创新与产业升级带来了新的机遇与挑战。

1.1.2 互联网的产品发展

技术的发展不是孤立的，在人工智能技术发展的同一时期，互联网以极快的速度带领人类进入信息化时代。互联网的演进除了其自身产品形态的持续发展外，该过程也推动了文本、图像、音频、视频等“全模态数据”的大规模积累，而这些数据正是后来大模型智能涌现重要的“数据”要素。

本书主要从产品视角来理解人工智能和大模型，因此有必要在此回顾互联网产品的发展与变迁。通过梳理互联网产品的演化脉络，我们可以更好地理解互联网产品形态的变迁对人工智能的促进作用，以及最终为何互联网必须全面拥抱人工智能。

1994—2002 年：从门户到搜索

互联网早期的产品形态以“门户网站”为主，用户通过目录导航获取 PGC（Professional Generated Content，专业生产内容）。随着信息量的爆炸，用户对“精准获取信息”的需求日益迫切，搜索引擎逐渐取代门户成为互联网主要入口。

这不仅是互联网产品形态上的变化，也为人工智能储备了第一批规模化的专业文本语料。

2003—2010 年：从搜索到社交

当用户的基础信息获取需求被满足后，“人与人的连接与互动”成为新的诉求。社交网络、博客、微博等产品形态相继崛起，推动互联网从“信息获取时代”迈入“关系构建时代”。

更为关键的是，这一阶段催生了海量 UGC（User Generated Content，用户生成内容），涵盖百科词条、聊天记录、评论留言、专业问答等多元文本数据——这些高价值数据，为

人工智能，尤其是自然语言处理、图像识别领域的早期研发奠定了坚实的数据基础。

2011—2016 年：从计算机到手机

移动互联网和 4G 网络的普及，彻底改变了产品的使用场景。随时随地的互联互通，使得互联网应用场景爆发式增长，从出行、外卖到支付、本地生活，移动互联网几乎重塑了所有行业。

移动互联网在便捷用户的同时，进入了“全模态数据”积累的快车道。海量的图像、短视频、语音数据被结构化存储，直接拉动了计算机视觉（Computer Vision，CV）与自然语言理解（Natural Language Understanding，NLU）技术的跨越式发展。

2017—2022 年：互联网的繁荣与困境

从 2017 年开始，互联网的产品形态逐渐发展到巅峰状态，快速吞噬用户量，直到 2020 年之后，新的用户增长基本停滞，互联网行业开始面临前所未有的挑战。

我们可以把当时的互联网想象成一个个商业村落。村落里有一片空地，大家都想把它做成最繁华的商业街：服务器工程师负责挖掘地基，网络工程师负责接入水电；基础打好后，后端工程师负责盖房子，前端工程师负责门店装修；产品经理负责研究商铺里卖什么产品，运营则负责在门口吆喝搞促销，吸引村民。

这个分工协作曾创造了黄金十年，但村子就这么大，村民就这么多，商业街已经盖满了房子，开满了门店。没有空地可以盖新的房子，没有房子需要装修，产品开始趋于同质化。后端工程师、前端工程师和产品经理的新增需求逐渐减少，只剩下运营方还在努力让已经“消费饱和”的村民继续进店。

为突破增长瓶颈，行业曾尝试“元宇宙”虚拟场景、“社区团购”本地生活渗透等新方向，但这些探索多为存量市场的再分配，并未找到新的增长引擎，互联网行业的发展节奏逐渐放缓，产品形态的发展更是几乎停滞。

2023 年—：All in AI

当互联网的流量增长空间见顶，此前数十年积累的全模态数据、成熟的信息化体系以及丰富的应用场景，恰好与大模型技术的爆发形成共振。数据积累达到临界点、算力成本持续下降、大模型技术实现突破性进展，多重因素叠加下，All in AI 成了互联网企业突破存量困境、寻找新增长曲线的必然选择。

由于本次人工智能的爆发性变革主要由大模型驱动，我们可以把 All in AI 理解为 All

in LLM（当然，AI 并不等同于 LLM，只是当前阶段 LLM 是其主要驱动力），互联网行业正式进入以大模型为主要驱动力的新发展阶段。大模型作为本书主要叙事对象，后续内容主要围绕“大模型”这个狭义的概念展开，仅在必要时使用“人工智能”字眼。

1.1.3 大模型重塑互联网产品

互联网的发展是大模型发展不可缺少的重要基础。尽管除了互联网之外，芯片、游戏、硬件等产业都为大模型奠定了基石，但互联网的贡献有其不可替代性——它不仅通过数十年的数字化进程提供了海量的语料，还有各种条件优渥的数字化平台，更在行业红利退去时，为大模型赛道输送了极具规模的人才群体。

在互联网从业者面临“村里开满商铺”的存量困局时，人工智能为他们提供了一个全新的赛道——大模型。如图 1-1 所示，展示出互联网的“人满为患”，与大模型提供带开辟的新土地，两者形成了鲜明对比。



图 1-1 大模型是提供新的待开辟土地

互联网从业者或主动跳出了原有无法增长的框架，或被迫迁徙到一个全新的阵地，总之他们逃离传统的互联网，从利用 AI 工具辅助工作开始，逐渐转向设计原生的 AI 产品。无论形式如何，最早接触大模型应用研发的他们，都或多或少在这个领域找到了新的发展节点。

然而，大模型产品与传统互联网产品在设计范式上存在巨大差异，这种差异主要源于两点底层逻辑改变：

（1）传统互联网是“规则定义智能”，所有产品能力都由人类预先编写的规则决定；而大模型是“数据定义智能”，产品能力来自海量数据训练后的概率性生成。

（2）大模型技术与互联网范式不同，从软件开发变成了算法迭代，产品效果从“先验”变成了“后验”。

这两点不同，诱发了产品从交互、形态到研发生产方式的全方位改变。

（1）产品形态的改变：从“执行预设逻辑”到“自主识别并执行内容”，从输入 A 必然得到 B，演变为有理解能力、生成能力与推理能力的智能体。

（2）交互方式的改变：交互重心从用户必须学习复杂的 GUI（Graphical User Interface，图形用户界面），转向了可以用一句话发布任务的 LUI（Language User Interface，语言用户界面），从“点选指令”变成了“表达意图”。这导致产品的交互设计工作也从设计原型图，变成了定义智能体的应答策略与交互边界。而在 GUI 与 LUI 并存的阶段，产品设计还面临一些额外的挑战，即什么时候让用户说话，什么时候给用户提供按钮。

（3）研发规范的改变：从“逻辑驱动”到“概率驱动”，过去产品的迭代依赖于精确的 PRD（Product Requirements Document，产品需求文档）和硬编码逻辑。现在，研发的重心变成了指令工程（Prompt Engineering）、模型微调（Fine-tuning）与解决方案设计。产研不再只需关注原型图、流程图，同时也需要深度参与模型效果的评测。评估一个产品的标准，从“是否有 Bug”变成了对 Goodcase（优良案例）和 Badcase（不良案例）的归因与治理。

（4）生产方式的改变：从“数据分析”到“数据生产”，传统产品的迭代主要依赖数据分析，而大模型产品的开发更加注重数据生产，无论是模型训练集还是评测集构建，都要求产研工作者对数据更加敏感。从“重型分工协作”到“超级个体”，在大模型产品的开发中，传统的“前后端 + 设计 + 测试”的大规模协作，正在向 AI 辅助下的超级个体或微型团队进化。

1.1.4 大模型重塑产品经理角色

互联网产品形态的变革，直接推动了今天互联网岗位角色与职能的颠覆性迭代。必须认清的现实是，在当下这个大模型时代，传统的研发模式在可预见的未来，就会被彻底打破——前端、后端的代码编写与执行工作将基本由 AI 驱动和接管，互联网已经不再需要由人类组成的庞大“建筑施工队”。对于传统的程序员群体来说，这不仅是生产工具的升级，更是生存空间的结构化重塑，他们的角色正逐渐从“代码生产者”向“AI 产出的审核者与系统架构维护者”转变。

相比之下，在这个“数据与模型生成”的时代，算法工程师凭借对底层模型的掌控获

得了前所未有的发展空间；而产品经理的传统职能虽然也在被不断压缩，但大模型赋予了他们一种可迁徙的生存形态。当研发执行不再是瓶颈，产品经理将面临工作内容的巨大重塑。下面将重点探讨这一演变过程。

（1）首先，在 Web 互联网的早期，所有的产品形态都处于一种原始的“萌芽”状态，产品经理的职业使命是探索与尝试。无论是 UGC 社区、在线音乐、视频播放器还是早期的社交软件，每一个产品的发布都是对未知领域的探索。那个时代的产品经理就像是初代建筑师，主要工作是功能设计，首要任务是确定房子的基本结构——是否有留言板、是否有加购按钮等。只要功能跑得通，有用户量，产品就能立足。

（2）后来，随着智能手机的普及，互联网村落进入了高密度建设期。当功能实现逐渐成为一项基础设施，产品经理的角色和职责开始精细化分化。最开始，大家卷 UI 设计与交互体验；到了中后期，业务逻辑日趋庞杂，产品间的竞争加剧，为了应对复杂的业务线，产品经理的角色发生了前所未有的裂变。

- 面向人群：C 端产品经理、B 端产品经理、交付产品经理、硬件产品经理、平台产品经理等。
- 面向逻辑：功能产品经理、策略产品经理、接口产品经理等。
- 面向价值：电商产品经理、商业产品经理等。

（3）如今，大模型的全面渗透带来了全方位的颠覆，尤其是互联网这一数字化程度最高的领域，最先迎来了生产要素的正向赋能与重组。当研发工作被 AI 自动化驱动，产品经理与开发之间的传统边界正在迅速消融。现在，单纯使用大模型工具已经不再是加分项，而是产品经理底层工作的基础。除了将大模型作为辅助工具外，更关键的转变在于：产品经理需要开始逐渐理解大模型的技术原理，并尝试将这种基于概率、涌现与推理的新技术范式，深度植入新一代的产品设计中。

技术革新所驱动的生产要素重构是行业发展的必然趋势，无法逆转。作为互联网从业者，我们依旧可以主动认识这种新的技术，并将新技术加入我们的产品中，使其成为产品的一部分。

1.2 大模型产品的范围

从先前的描述中，我们已经知晓了互联网对人工智能的促进作用，以及人工智能如何反过来重塑互联网的产品形态和生产要素。本节会对大模型的产品范围进行定义，并通过

对比大模型产品与软件工程、机器学习、互联网产品三个方面，展开探讨大模型产品的特征，希望可以让我们对大模型产品的边界有更加深入的认识。

1.2.1 大模型产品的定义

当前，大模型产品尚缺乏行业公认的规范定义，本书认为，以大模型技术为关键价值来源或主要功能支撑，能够给用户带来独特智能价值的产品，可称之为大模型产品。根据大模型技术占比，我们可以将大模型产品分为三种形态：大模型本身、AI 原生应用以及赋能型功能，如图 1-2 所示。



图 1-2 大模型产品的三种形态

（1）大模型本身，这是一种纯粹的底层基座。典型代表是各种开源模型及各厂商开放平台（Model as a Service，MaaS）提供的 API 接口。这类产品难以直接面向 C 端用户，需要开发者进行二次开发，其主要目标在于提供强大的自然语言理解、生成与逻辑推理能力。我们熟知的 GPT、豆包大模型、GLM 就属于大模型本身。

（2）AI 原生应用，即完全依托大模型智能能力构建的独立产品，典型代表如 ChatGPT、豆包、智谱清言等。这类产品没有传统互联网产品的固定功能链路（如电商“浏览－下单－支付”的闭环），以 LUI 自然语言交互为主，聚焦满足用户创作、推理、问题求解等无固定流程的创造性需求（如文案、代码、解决方案等）。为了更好地区分前两者，

我们用一个简单的例子来说明：GPT 是模型，ChatGPT 是产品；豆包大模型是模型，豆包 App 是产品；GLM 是模型，智谱清言是产品。

（3）赋能型功能，即作为非核心模块嵌入传统互联网产品，为其补充“智慧属性”，典型代表包括淘宝智能客服、飞书会议助手等。这类形态下，大模型不独立承载用户需求，而是依托互联网产品的现有功能形态（如交易流程、社交链路、办公场景），通过数据理解与生成能力优化体验、提升效率，作为互联网产品形态的补充功能。

1.2.2 大模型与软件工程的差异

大模型开发与传统软件工程开发，这种底层架构和驱动力的不同，直接导致了算法工程师与软件研发工程师在面对问题时，存在着截然不同的思维路径。

首先，软件工程与大模型开发两者的目标不同：

- 软件工程的目标是生成一套按特定顺序组织的计算机数据和指令集合，它高度依赖工程和逻辑驱动，开发者需要输入诸如 Java、Python 等结构化的计算机语言。软件工程的研发过程除了负责逻辑和数据的实现外，还要考虑并发、分布式、集群等大规模的工程实现，研发过程是模块化或服务化的开发过程。软件工程研发架构图如图 1-3 所示。

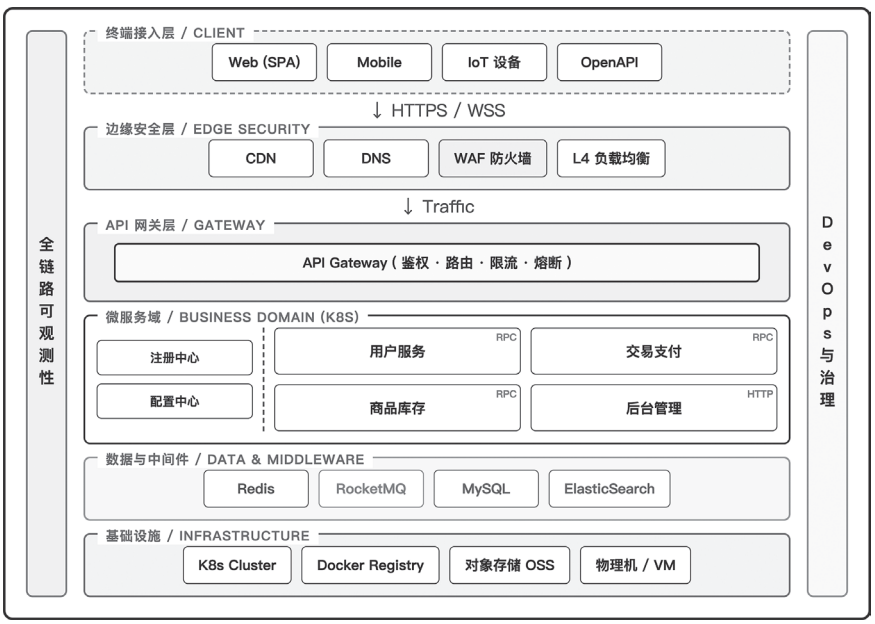


图 1-3 软件工程研发架构图

- 大模型的目标是让机器通过算法学习并模仿人类行为，从而实现推理、决策等复杂的智能性工作。它主要由数据和算法驱动，接收的往往是自然语言或非结构化、半结构化的数据。大模型的研发过程不像软件工程那样分模块地去做，而是阶段化地开发。大模型的研发架构图如图 1-4 所示。

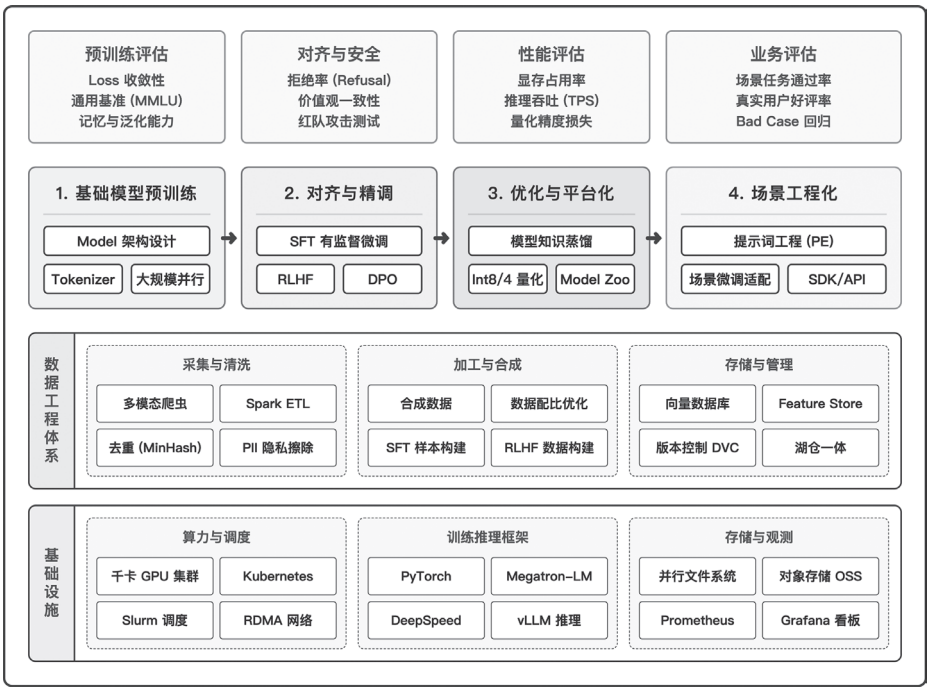


图 1-4 大模型研发架构图

两者最直观的差异主要体现在输出结果的“先验”与“后验”：

- 软件工程的先验性在于软件是一个典型的确定性系统，在动手开发前，产品的全部细节与业务逻辑就已经被设计完毕。只要输入是固定的，输出必然是唯一的。整个研发过程实际上就是将规则与逻辑“写死”，一旦系统出现问题，研发工程师都可以通过查看日志、断点调试等方式，沿着清晰的代码链路精准溯源，找出明确的错误节点。
- 大模型的后验性在于大模型是一个概率性系统，在开发和训练彻底完成前，没有人能完全清晰地预知最终的结果。即使是面对完全相同的输入，大模型也可能给出不同的输出。这种不可预知性导致大模型的问题溯源难度较大，很多异常表现不仅难以复现，且问题根源往往深埋在海量的训练数据质量或复杂的模型结构中，需要依靠大量的实验与统计结果来反推优化方向。

两套体系对系统缺陷的认知存在明显分歧，表现在对“概率”的包容度上：

- 确定性逻辑认知。习惯了确定性结果的软件工程师在接触大模型时，常常会陷入一种思维误区：他们期望模型像传统软件一样绝对可靠，一旦模型给出了离谱的回答，他们就认为是系统出了 Bug，因为在传统的工程世界中，1+1 等于 2 是理所应当的铁律。
- 概率性逻辑认知。在大模型的概率世界中，1+1 有极低概率等于 3 是一种由技术范式决定的、无法被彻底消灭的特性。因此，在大模型的研发语境中，我们较少使用 Bug 来定义这类问题，而是称之为 Badcase（坏案例）。一句话总结：在应用层面，大模型主要存在一些难以预测的 Badcase，而不是软件工程传统意义上的 Bug。试图用传统软件工程中“修补代码漏洞”的线性思路来彻底消灭 Badcase，就像试图通过缝补来阻止云朵变形一样徒劳。



★ 大模型的 Badcase 不能与软件 Bug 划等号

两者的迭代流程与测试方案也大相径庭，传统软件工程遵循着“原型设计、工程设计、软件研发、工程测试、Bug 修复”的明确路径，其测试方案主要围绕功能点、系统性能以及安全性（兼容性、稳定性）展开。相比之下，大模型的迭代则是一场围绕数据的闭环游戏：从初期的应答策略设计开始，经历烦琐的训练数据构造与模型训练，随后进入评测集测试，最终落脚于长期的 Case 分析与持续优化。在测试环节，大模型不再局限于功能的验证，而是高度依赖大规模的评测集来覆盖多场景以验证生成效果，同时还需要进行高强度的对抗性攻击测试，以此来探明模型的健壮性与安全边界。

1.2.3 大模型与机器学习的差异

在对比过大模型与软件工程后，我们来进一步探讨大模型与传统机器学习、深度学习

的差异。首先，我们对几个概念进行辨析。

（1）机器学习是实现人工智能的一种技术方式，通过算法让机器从数据中学习规律，无须显式编程即可完成特定任务，这个概念完全包含深度学习和大模型。除了深度学习外，决策树、逻辑回归、支持向量机、随机森林等算法也属于机器学习，这些算法广泛应用于传统数据分类、回归预测场景，比如垃圾邮件识别、用户信用评分等，无须复杂的网络结构就能完成基础智能任务。

（2）深度学习是机器学习的一种类型，是通过神经网络实现的机器学习。例如，感知机、卷积神经网络（Convolutional Neural Network，CNN）、循环神经网络（Recurrent Neural Network，RNN）等。其中，CNN 擅长处理图像类任务，比如图片识别、目标检测；RNN 则适合处理时序数据，比如语音转文字、文本情感分析，都是通过多层神经网络模拟人脑的信息处理方式。

（3）大模型是深度学习的一个分支，当前主流采用 Transformer 架构，因其参数量和网络层级远超传统深度学习模型而得名。例如，自然语言领域的 GPT、GLM、Qwen 等系列，多模态领域的 GPT-4V，以及垂直领域的医疗大模型、金融大模型等，这类模型具备通用的理解、生成、推理能力，可适配多场景任务。由于这个分支在现阶段涌现出了其他机器学习方案无法比拟的能力，所有的研发资源、产品都开始向大模型靠拢。

经过辨析，我们可以得知大模型与机器学习在技术角度并非同一层级的概念，二者无直接可比性。两者的关系和其他人工智能概念可以参考图 1-5。

但是在研发过程中，我们需要明白为什么选择大模型，而不选择机器学习。其原因主要有两个，一是更加智能，二是成本更低。更加智能不必再解释，但“成本更低”需要进一步说明。假设我们要打造一款人工智能产品，相对于传统机器学习和深度学习而言，需要从头做起，这就需要大量的算法、算力、数据标注投入；但大模型因为其通用能力更强，作为应用侧的产品，我们不需要再从头训练一个模型，而是直接使用大模型或在大模型的基础上进行指令工程、微调训练。

因此，对于产品侧而言，完成同一个 NLP 任务，要达到同样效果，所需的人力成本和实践成本远低于重新训练的成本。或许这与真实场景中我们的感受不同——现实中大模型产品的研发成本往往非常高。其主要原因在于当模型的能力更强大时，我们对人工智能产品也提出了更高的要求，导致现实情况是，大模型使用成本可能反而更高。

结合上述分析，我们用表 1-1 来总结传统机器学习与大模型在不同方面的成本投入差异。

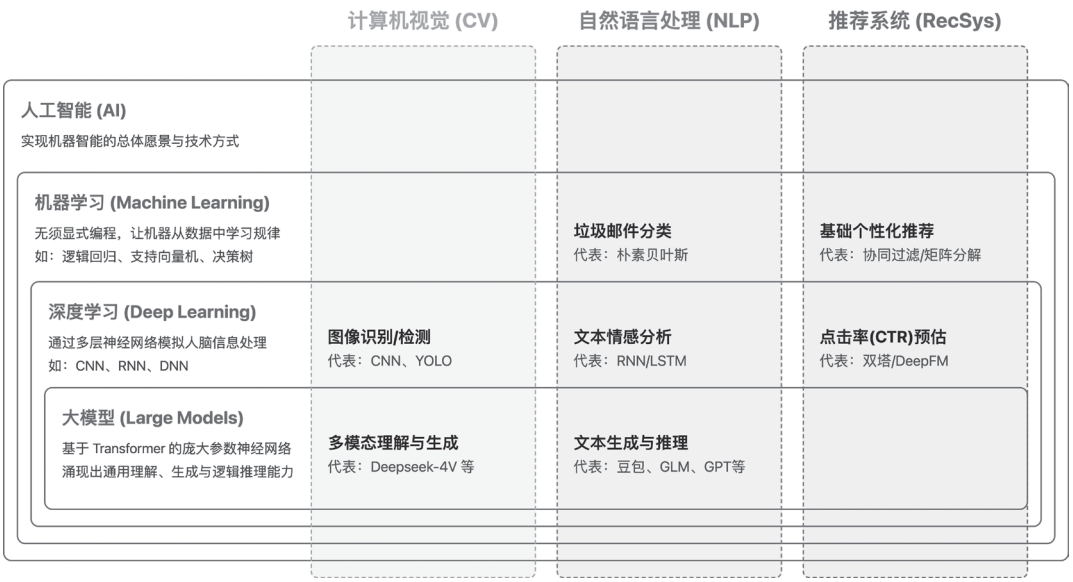


图 1-5 大模型与机器学习和其他人工智能概念的关系

表1-1 传统机器学习与大模型在不同方面的成本投入对比

成本维度	传统机器学习	大模型
人力投入	高：每个场景需定制算法，依赖资深算法工程师设计结构与参数，迭代周期长	中低：通用性强，主要通过Prompt工程或指令微调实现，产品经理/研发即可上手
算力成本	中：单模型参数小，但多场景需多次独立训练，累积投入大	中低（应用端）：虽然预训练极耗算力，但对于产品侧和应用端而言，通过API或小规模微调即可，试错成本极低
时间成本	高：全流程（数据→模型→测试）周期长，场景适配性差，新增场景甚至需重新研发	低：原型落地快（上午有想法，下午可能就有Demo），场景拓展通过微调可快速覆盖
标注成本	高：需数十万条专属场景数据，任务间无法复用，标注工作量是主要瓶颈	低：复用通用预训练知识，微调阶段仅需数千至上万条高质量数据
维护成本	高：各任务模型独立，问题排查需针对特定代码和数据，维护效率低	中：不同的场景可以统一模型底座，多场景共享资源，能使用同一套Case治理体系

1.2.4 大模型产品与互联网产品的差异

前文中，我们明确了研发侧从“逻辑驱动”向“概率驱动”的范式转移，以及与传统机器学习的边界。这些差异更多聚焦于底层“技术实现”层面。本节将切换至应用侧视角，

进一步拆解大模型产品与传统互联网产品的主要区别，厘清二者的定位边界与协同逻辑。

两类产品最显著的区别在于“智能”属性的有无，这种差异首先体现在产出物的价值创造逻辑上：

传统互联网产品不产生创造性、决策性智能数据，其主要职能是承载用户行为数据与信息的存储、通信与流转，通过优化链路实现高效连接；而大模型产品的关键优势在于对数据的深度理解与生成能力，能够基于用户需求输出全新价值数据。换句话说，互联网产品的基础价值是“解放体力”，通过简化流程降低用户的行为成本；大模型产品的独特价值则是“解放脑力”，通过智能能力降低用户的思考与决策成本。

从用户群体分布与付费逻辑来看，两类产品的差异同样显著：

互联网产品的用户付费与留存，既依赖便捷性、基础功能的满足，也看重交互体验的流畅度，用户甚至愿意为“小而美”的小众场景价值买单。这一特征决定了互联网产品能够形成百花齐放的行业格局，中小玩家可凭借细分场景优势立足；而大模型产品驱动付费的关键诉求围绕“智能能力”展开——用户仅愿意为精准的意图理解、高效的内容生成、可靠的决策支撑等智能价值付费。这种单一付费导向，叠加大模型研发对算力、高质量数据、生态搭建的极高资源依赖，决定了其行业格局注定走向中心化，头部玩家将凭借资源壁垒形成绝对竞争优势。

此外，交互形态的底层逻辑差异，也进一步划分了两类产品的应用边界：如前文所述，大模型与互联网产品的交互逻辑，其显著差异在于 LUI 与 GUI 的区别；然而，当前主流的电子屏幕（适配 GUI 的视觉化交互）并非适配大模型的最优媒介，需通过多模态、工程映射等技术进行桥接适配，存在一定的体验损耗。

结合当前技术发展阶段和行业落地实践，可以清晰地感知到，AI 原生应用尚未达到“理想态”（即稳定的智能输出、无交互壁垒、全场景覆盖）——在智能输出稳定性、全场景适配精度、GUI 与 LUI 无缝融合等方面仍有明显短板。

因此，大模型产品与互联网产品并非直接竞争关系，而是协同共生的关系：互联网产品定义了产品的“功能形态”，搭建了用户需求落地的基础链路与载体；大模型产品则定义了产品的“智慧形态”，为基础功能注入智能增量，突破传统体验边界。

从产品经理视角来看，这种协同关系对产品设计提出了明确要求：

设计互联网产品时，重点需聚焦功能流程的闭环与交互体验的优化，确保用户确定性需求的高效满足；设计大模型产品时，关键需聚焦智能能力的适配与价值输出的精准性，

确保用户创造性、决策性需求的有效响应，同时需兼顾与互联网产品的协同逻辑，避免陷入“为智能而智能”的设计误区，即脱离用户真实需求堆砌智能功能，或在缺少完善评测体系的情况下贸然新增大量优化策略。

随着技术的持续迭代，当 AI 原生应用能够彻底抹平 GUI 与 LUI 的交互差异，实现稳定、全场景的智能服务闭环时，才可能真正对传统互联网产品形成直接竞争，届时行业格局或将迎来新一轮重构。

1.2.5 大模型产品产业的分化差异

先前我们已经讨论过大模型产品的三种主要形态，除了产品类型的划分外，大模型产业在发展过程中也出现了多维度的路线分化，其中最受行业关注的就是中美 AI 产业的发展路线差异。讨论这种分化前，我们先直面一个行业内高频出现的尖锐问题：为什么 ChatGPT、Gemini、Claude 等海外模型的通用智能体感更强，甚至在行业发展初期，外界出现了“国产模型仅是 GPT 套壳”的刻板印象？

本节从产品视角出发，对应前文的三类产品形态，拆解中美大模型产品的发展差异，旨在帮助读者理性看待这些分化。

1. 基座模型层：技术追赶与差异化路线并存，差距持续收敛

可以明确的是，国内已有多家优秀企业建成了完全自主的大模型研发生产体系。尽管国内大模型的通用技术研究起步相对较晚，这种差距在产业爆发初期尤为明显，但 2025 年后已快速缩小。同时，国产模型长期受限于高端算力资源的供给约束，反而倒逼行业走出了低成本研发的差异化路线，尤其在高效模型压缩、分布式训练优化等方向实现了技术突破，形成了独特优势。这一特点在 2025 年初深度求索发布的 DeepSeek R1 系列模型上已得到充分体现。

相比之下，美国头部模型更聚焦前沿技术创新与通用能力上限的突破，其研发投入与算力成本普遍高于国内同级别模型。总体来看，二者在基座层的技术差距已收敛至相对稳定的区间。

2. AI 原生应用层：大众普惠导向与前沿技术导向的路线分化

国内拥有规模庞大的互联网用户群体，无须面向专业研究人员即可获得海量用户反馈，这使得国内 AI 原生应用从设计之初就优先考虑大众用户的使用体验：一方面，通过技术优化降低使用成本，推动产品下沉，从而覆盖更广泛的用户群体；另一方面，重点打磨响

应速度、隐私合规、全端适配（尤其是移动端优化）等基础体验。同时，国内专业研究型用户更倾向于使用通用能力更强的海外头部产品，这也促使国内原生应用放弃了盲目对标通用底座的内卷，转而深耕垂直场景的差异化竞争。

而美国头部 AI 原生应用更聚焦全球化布局与前沿研究工具属性，主打通用能力的极致突破，在新兴市场的本土化渗透与大众用户的适配优化上节奏较慢。这种分化的一部分原因是市场导向的差异：国内产品强调对全量用户的包容性与普适性，美国产品追求技术能力的前沿性与上限。

3. 赋能型功能层：垂直行业定制化的分化

得益于过去十年“互联网 +”的深厚积累，国内拥有全球最丰富的数字化场景与产业落地经验，以场景需求为驱动，将大模型快速转化为生产力工具的“实用主义”导向尤为凸显。智谱、阿里、讯飞、商汤等头部模型供应商均深耕 B 端场景化交付，已摸索出成熟的大模型定制化解决方案交付体系，大模型在政务、制造、电商、医疗、物流等垂直领域的渗透速度与落地深度稳居全球第一梯队。

中美大模型产品的差异，我们可以用一个通俗的比喻总结：美国大模型打造的是性能顶尖的专业级发动机，面向专业用户与前沿场景，追求极致的性能上限；而国产模型则保持低成本特性，持续追赶智能水平的同时，还搭建了更适配大众场景的整车体系，优化了操控门槛、使用成本与场景适配性，更贴合海量普通用户与全行业的落地需求。

1.3 小结

本章首先按时间顺序，简要梳理了人工智能技术与互联网产品的发展历程，分析了大模型对互联网产品形态和生产要素带来的变化。在此基础上，介绍了大模型产品的三种形态——大模型本身、AI 原生应用、赋能型功能，并从研发范式、产品定位和产业格局三个维度，比较了大模型产品与软件工程、机器学习及传统互联网产品的差异，明确了大模型产品的能力边界。最后，从基座模型、AI 原生应用和赋能型功能三个层面，分析了大模型产品产业的发展差异，帮助读者建立对大模型产品的基本认知框架。对于刚刚进入 AI 产品开发领域的初学者和产品经理，本章可以帮助读者理解大模型时代 AI 产品的特征，为今后的实践工作奠定基础。

CHAPTER

第2章

新的职能定位：大模型时代的产品经理

互联网发展的黄金十年，使得产品经理角色得到了前所未有的发展，同时也使得该岗位分化得极其细致。然而，这种基于 Web 与移动互联网业务建立的职能划分，在大模型产品设计场景中已无法直接套用。正如第 1 章所述，大模型产品与传统互联网产品存在底层范式的转移，这种转移导致产品经理的工作内容和思维模式发生了不可逆转的变化。

本章将聚焦大模型产品经理的工作内容、思维模式、协作方式和分化情况，帮助读者清晰认识这一角色在大模型时代的定位变迁，并从中找到适配自身的发展方向。

2.1 大模型产品经理概述

每次有新的技术诞生，都会对产品经理岗位有所改造，有的技术可能仅仅需要产品经理将新技术融入自己的工作中就可以，例如小程序、二维码等技术。这种技术对应的功能体量小，可优化空间也小，因此，产品经理完全可以将技术融入自己的工作中。

但是当技术变得复杂时，就会产生专门的岗位来应对，我们以搜索引擎和推荐系统这两种比较重要的互联网技术为例。回溯互联网早期，软件只是简单接入搜索引擎或推荐系统，传统产品经理完全可以兼顾。但随着产品功能的复杂化，行业内逐渐衍生出了专门的“搜索产品经理”与“推荐策略产品经理”。

由于大模型对整个信息化体系的重构与冲击，在深度和广度上远超搜索引擎与推荐算法，因此大模型产品经理的重要性更加不可替代，并且也逐渐发展出专门的“大模型产品经理”。本节将深入探讨“大模型产品经理”的角色定位及其要解决的问题，明确该角色在新时代存在的必然价值。

2.1.1 大模型产品经理的定位

当产品的主要交互方式从 GUI 转向 LUI 时，底层链路发生了深刻演变：传统互联网产品依赖“页面－控件－业务”，而 LUI 的底层支撑则是“数据－模型－应答”。这种链路的切换，是大模型产品经理与传统产品经理产生职能分野的根源。

在大模型重构产业的当下，绝大多数数字化产品都在探索“AI 化”。对于互联网产品经理而言，几乎没有可以完全脱离大模型的职业避风港。对应地，产品经理都要考虑向大模型方向靠拢，这不是一个“要不要靠拢”的问题，而是“如何靠拢以及靠拢到什么程度”的问题：

- 应用型适配：原有互联网产品的主要业务逻辑不变，产品经理的关键工作内容基本稳定，仅需借助大模型的能力优化产品体验、提升业务效率，这也是绝大多数产品经理的类型。
- 专职型适配：聚焦大模型技术本身的产品化落地，专门围绕模型的能力规划、效果优化、场景落地以及商业变现等工作展开，是大模型时代衍生的专属角色，也就是本书重点讨论的大模型产品经理。

2.1.2 大模型产品经理的分化

从产业分层的角度来看，大模型相关的产品经理可归纳为以下 4 类，各角色职责边界清晰且相互联动。

1. 应用侧产品经理

这是行业内规模最大的角色群体，主要聚焦业务场景定义与用户需求挖掘，首要目标是在现有产品体系中落地大模型的能力，通过“智能元素”的注入优化产品体验、提升业务效率。该角色无须深入掌握大模型的底层工作流（Workflow）与全量评测工作，重点职责是精准理解用户痛点，定义产品的业务目标与智能化功能形态，并向模型侧输出明确、可落地的业务需求。例如，在招聘软件中设计简历自动解析、人岗匹配度智能评估等功能，均为应用侧产品经理的典型工作范畴。

2. 模型侧产品经理

这是本书的主要叙事主体，深耕大模型相关的全流程产品化工作，首要目标是保障模型的能力边界、交互效果与业务适配性，是大模型产品的关键设计者。 其主要职责涵盖模型选型、关键能力规划、评测体系搭建、模型效果优化、Prompt 策略设计、工作流搭建、微调方案落地等全链路工作。例如，用户向对话产品发起提问时，意图识别、垂域路由、应答方案、结果渲染的全链路策略设计，均为模型侧产品经理的重要工作范畴。

3. 数据侧产品经理

这是相对辅助的角色，负责大模型数据全生命周期的产品化管理，主要目标是为模型优化提供高质量、高适配性的数据支撑，是大模型能力迭代的基础保障。 其具体职责涵盖数据体系搭建、数据质量管控、标注体系设计、训练与评测数据生产、数据迭代闭环搭建等，部分企业中该角色也会以数据运营的形式呈现。

4. 平台侧产品经理

独立于上述纵向分层体系，重点职责是构建支撑大模型研发、应用与迭代的基础设施平台，为其他三类角色提供稳定、安全、高效的底层支撑。 其工作范畴涵盖 MaaS 平台设计、训练与推理工具链搭建、模型运维（ModelOps）体系设计、平台安全风控体系搭建等。该角色的工作模式与 SaaS 产品经理有一定相似性，但需全面掌握大模型全链路的产品与技术知识。

4 类角色的层级划分与协作关系如图 2-1 所示。

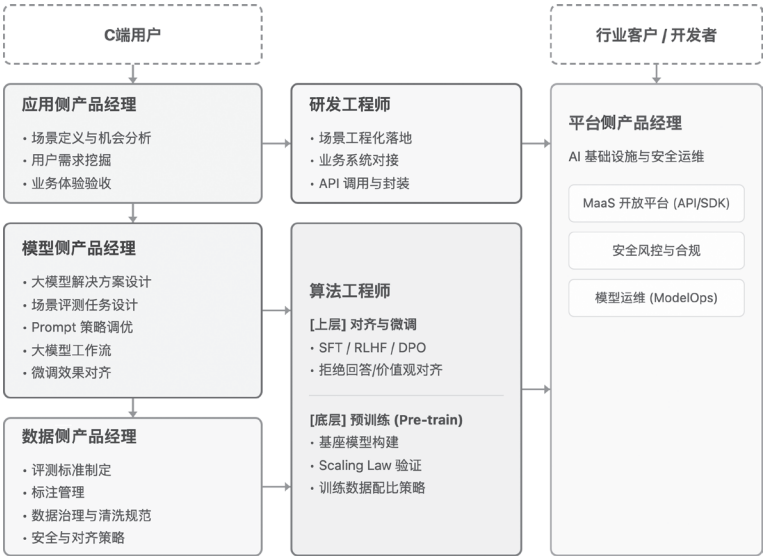


图 2-1 大模型产品经理的 4 个层级

由于本书的案例大多以酒吧场景为载体，我们就以酒吧场景对话机器人的设计为例，前三类角色的主要分工如下：

- 应用侧产品经理：聚焦酒吧场景的用户特征、消费习惯与场景氛围，定义机器人的人设形象、语音特征与业务功能，包括智能点餐、酒水推荐、会员服务、场景互动等，确保所有功能贴合用户真实诉求与酒吧的业务目标。
- 模型侧产品经理：主要职责是保障模型侧的智能交互效果，包括用户多轮对话意图的精准识别、闲聊能力与拒答策略设计、基于用户画像的酒水推荐逻辑优化、多轮对话的记忆机制设计等。
- 数据侧产品经理：围绕机器人的预期能力，搭建完善的酒水行业知识库，构建覆盖多口音、方言、酒吧嘈杂环境的语音识别训练集，以及全场景对话的评测数据集，为模型调优提供高质量的数据支撑。

2.1.3 大模型产品经理的能力要求

在明确产品经理在大模型时代的角色分工后，结合传统互联网产品经理的能力体系，我们进一步拆解不同类型产品经理的能力维度，以及各角色对本书后续内容的重点关注方向。各角色的能力维度权重如表 2-1 所示。

表2-1 4种大模型产品经理的能力维度权重

能力维度	应用侧产品经理	模型侧产品经理	数据侧产品经理	平台侧产品经理
行业/场景洞察	☆☆☆☆☆	☆☆☆☆	☆☆☆	☆☆☆
指令工程	☆☆	☆☆☆☆☆	☆☆☆	☆☆
数据治理/评测	☆☆☆	☆☆☆☆☆	☆☆☆☆☆	☆☆
模型原理/微调	☆	☆☆☆	☆☆	☆☆
项目管理	☆☆	☆☆☆	☆☆	☆☆☆
团队管理	☆☆	☆☆	☆☆☆☆	☆☆
基建/运维	☆	☆☆	☆☆	☆☆☆☆☆
需要初步了解的章节	第3、5、9、10章	第8、10章	第5、8、10章	第3~8章
需要重点熟悉的章节	第4章	第3~7章、第9、10章	第3、4章	-
注：星级基于各角色职责权重判定：5星＝核心能力，4星＝重要能力，3星＝基础能力，2星＝了解即可，1星＝无须重点关注。				

基于表 2-1 中的能力权重，我们对 4 类角色的能力边界与协作边界做补充说明。

1. 应用侧产品经理

无须精通模型底层工作原理，如检索增强生成（Retrieval-Augmented Generation，RAG）优化方案、对话机器人多意图调用机制等，但需要理解模型的能力边界，对大模型的输入输出效果保持敏感，同时具备基础的 Prompt 书写能力，用于 Demo 测试或简单智能功能的落地。其高频协作角色包括模型侧产品经理、软件研发工程师、运营、UI 设计师、项目经理等。

2. 模型侧产品经理

主要工作是设计 LUI 的输入与输出，其工作产出的“原型图”，实际上就是模型输入输出的策略方案。这一关键职责决定了其深度熟悉数据，部分场景下还需配合算法工程师完成模型训练工作。因此，模型侧产品经理不仅要掌握数据本身，还需清晰认知数据分布特征，包括用户高频查询意图、长尾查询意图、高优先级 Badcase、低优先级案例等。除常规的策略设计外，指令工程、工作流搭建、模型评测等环节也占据其大部分工作时间。其日常协作角色包括应用侧产品经理、算法工程师、软件研发工程师、数据侧产品经理、项目经理等。

3. 数据侧产品经理

首要工作聚焦训练数据与评测数据的全流程优化，几乎不参与产品功能层面的设计，重点深耕数据构建流程与模型评估方法。由于数据标注属于劳动密集型工作，数据侧产品

经理除完成本职的流程规划外，还需承担标注数据管理、标注团队管理等职责，多数场景下需配备专属标注支撑人员。其日常协作角色包括标注团队、算法工程师、模型侧产品经理、项目经理等。

4. 平台侧产品经理

需全面了解大模型相关的全链路知识，包括大模型工作流、模型评测、数据标注、模型部署、模型微调等，才能设计出适配生态用户（开发者、模型侧产品经理、数据侧产品经理、算法工程师等）的平台产品。其日常协作角色包括各类生态用户、软件研发工程师、UI 设计师、项目经理等。

2.1.4 大模型产品经理身兼数职的趋势

需要特别说明的是，当前大模型领域的职能分工，主要目的是提升中大型团队的协作效率，但在实际业务中，“全链路的大模型产品经理”身兼数职的趋势明显，原因有以下 4 个。

1. 业务边界聚焦，具备全链路统筹的客观条件

多数场景中，传统互联网产品的复杂业务流程（如订单、支付、账户体系）仍由应用型产品经理承接，大模型产品设计者可聚焦模型侧模块，无须被繁杂的传统业务逻辑占用精力。

2. 团队规模限制，跨职能协作效率更高

大模型创业团队普遍规模较小，成员往往身兼数职，无须划分过于清晰的角色边界；同时，大模型产品设计者本身熟悉 AI 工具的使用，可借助工具大幅提升跨职能工作的效率，全链路统筹反而能减少沟通损耗，提升决策效率。

3. 产品迭代要求高，需快速完成闭环验证

大模型 API 的调用门槛极低，MVP（Minimum Viable Product，最小可行产品）的验证周期大幅缩短，需要产品经理快速完成“需求定义－模型验证－应用落地”的全链路闭环，过于精细化的分工反而会导致迭代滞后，错失市场窗口。

4. 技术链路高度耦合，全链路统筹更易实现协同优化

大模型产品的研发链路高度耦合——数据质量直接决定模型效果上限，模型的能力直接决定应用体验下限，而应用场景又反向指导数据采集与模型优化的方向，单一角色统筹

全链路，可有效避免各环节脱节的问题，实现协同优化。

至少在当前大模型领域，若产品经理具备全链路能力，高效承接数据、模型与应用层面的工作，在中小规模的应用侧产品或创业团队中，基本可以包揽大部分核心流程职责。尤其在创业环境中，这种“全栈型大模型产品经理”能够最大限度地减少跨角色沟通损耗，加快产品迭代与验证速度，往往是团队的竞争力所在。

无论角色如何分化、职责边界如何界定，有一项工作是所有大模型产品经理都要深度参与的——数据评测与数据生产。数据是大模型“智能”的基础，模型效果是大模型产品体验的基础，而产品体验最终决定产品的价值。这一逻辑决定了大模型产品经理对技术的理解深度（尤其是模型原理、技术边界），以及对数据的敏感性（包括数据质量、数据分布、数据适配性等），会显著高于传统互联网产品经理。

2.2 大模型产品设计的底层思维转变

对于产品经理而言，想要做好大模型产品，首先要完成的就是底层思维的转变——这些底层思维将贯穿产品设计、研发、优化、运营的整个流程，是从“会用大模型的产品经理”到“会做大模型产品的产品经理”的转变。

本节总结了 7 个大模型产品设计过程中的底层思维，帮助我们完成从传统互联网产品经理到大模型产品经理的思维转化，建立适配新范式的完整思维体系。

2.2.1 使用模型产品和做模型产品的思维差异

绝大多数产品经理都能熟练使用 ChatGPT、豆包等通用大模型产品，但“使用大模型产品”与“设计大模型产品”有明显的思维差异，其主要差异体现在消费者思维与生产者思维的方面，区别集中在以下两个维度。

1. 诉求与决策标准不同

消费者追求“单次成功”，而生产者追求“系统级稳定”。个人使用大模型时，只要能通过调整指令拿到一次满意的结果，就达成了目标，可以容忍模型偶尔的幻觉、逻辑错误或答非所问；但作为产品设计者，要服务海量用户的全场景需求，就必须通过工程机制与策略设计，把幻觉率、错误率控制在业务可接受的范围内，保障绝大多数用户在绝大多

数场景下，都能获得稳定、符合预期的结果。

2. 对待问题的态度与解决方法不同

消费者遇到问题时，以“单次问题解决”为终点；而生产者遇到问题时，以“系统级问题解决”为目标。个人使用时遇到 Badcase，只需调整 Prompt 重新生成，拿到满意的结果即可结束；但作为产品设计者，我们不能指望用户去反复试错，需要在系统层面解决该问题，同时举一反三覆盖同类型的衍生场景，通过完善底层逻辑避免同类问题再次发生。

行业内的“超级个体”，就是典型的大模型高水平消费者——他们的目标是用 AI 工具提升个人效率，无须考虑系统级的泛化能力与全场景风险防控；而大模型产品的构建者，是游戏规则的设计者，如同顶级电竞选手是游戏的顶尖使用者，而游戏设计师的目标不是个人操作的极致发挥，而是保障千万玩家的稳定游戏体验。

作为大模型产品设计者，需要时刻保持清醒：我们的目标是打造一个稳定可靠的系统级产品，而不是追求模型偶然性输出的单次优质结果。有时候在使用大模型时觉得效果还不错，但做成产品后效果差异可能远不及预期。



★ 消费者与生产者的思维差异

2.2.2 对大模型效果的怀疑思维

“大模型不是万能的”，是有经验的大模型产品设计者的基础常识，却是传统产品经

理转型时要跨过的第一道门槛。在个人使用场景中，大模型能出色完成文案撰写、逻辑梳理、信息查询等绝大多数日常工作，很容易让人陷入“大模型能解决所有业务问题”的认知误区——这正是消费者思维向生产者思维转型时，最容易踩的坑。

认清大模型的能力边界，建立清晰的边界感，不是否定大模型的价值，而是让其能更好地适配场景。产品经理的职责，不是强行要求大模型完成它不擅长的事，而是通过产品设计与策略设计扬长避短：让大模型专注于它最擅长的自然语言理解、生成与推理工作，用产品规则与工程机制兜底它的原生短板，最终作出稳定、可靠、真正有用户价值的产品。本书后续的技术章节会详细拆解大模型产品如何通过多层策略封装，最终实现符合业务预期的产品化输出。

基于这一认知，我们在全流程中需要时刻坚守一条底线：不要盲目信任大模型的生成结果，在每一次产品决策与系统设计中，都要对模型输出保持怀疑与严格校验的态度，落地动作需注意以下三点。

1. 全流程建立标准化评测机制

在大模型产品的全生命周期里，任何关于模型效果、产品优化、需求落地的结论，都需要基于严谨的评测数据，而非主观感受、单一案例或经验判断。无论是内容生成、信息查询，还是功能调用、逻辑判断，都需要建立标准化、可量化的评测与校验机制。

2. 预设出错场景，设计兜底方案

永远不要假设“模型永远是对的”，而是要提前预判模型可能出现错误的场景，并设计严密的兜底策略。例如，当模型生成的内容触碰合规红线时，必须有独立的安全风控机制进行拦截，坚决避免违规内容流出。

3. 建立 Badcase 闭环思维，将每一次出错视为优化的起点

绝不能因为模型在大部分场景下表现良好，就忽略少数的严重错误案例。对于大模型产品而言，致命的 Badcase 就可能彻底摧毁用户对产品的信任。酒吧对话机器人若给未成年人推荐高度酒，会造成不好的业务后果。产品经理需要通过持续的 Badcase 归因分析，不断将边缘场景转化为优化数据，压低严重问题的发生概率。



★ 产品设计时对边缘场景的考量

对大模型的生成结果保持审慎，并非否定其技术能力，而是对产品和用户负责。一个合格的大模型产品经理，不会向业务方保证“模型肯定不会出错”，而是会明确说明：“我们已经通过多层工程机制，把模型出错的概率降到了业务可接受的水平，并且备有完善的兜底方案应对所有极端场景。”

2.2.3 从“先验”到“后验”思维

“先验”与“后验”思维的切换，是传统产品经理转型大模型产品设计时，最难适应的思维差异，也是 GUI 产品与 LUI 产品设计逻辑的区别。我们可以从“产品设计”与“线上运营”两个阶段，清晰拆解这种转变。

首先，是产品设计阶段，这种思维差异体现得最为明显——传统互联网产品设计遵循典型的先验思维，而大模型产品设计需要建立后验思维。

- 传统互联网的先验思维，是“先定义、后验证”。产品经理凭借经验，在上线前就定义好了所有的流程与交互，并穷举了异常场景。例如点餐 App 的结账流程，连“支付失败”的弹窗都在预设之中。只要研发照做，系统逻辑就是 100% 可控的。产品上线后面临的唯一未知是“用户愿不愿意用”，而不是“系统会不会按规矩跑”。
- 大模型产品的后验思维，是“先验证、后定义”。因为模型产品在开发完成前，其输出结果具有较高的不确定性，即便是经验丰富的产品经理，也只能预估大致的效果区间，且无法提前预判所有的用户输入和模型应答，因此无法提前 100% 定义模型行为。这导致产品经理

只能从“实测”开始：先构建基础的 Prompt 和评测集进行测试，通过分析真实对话中产生的 Badcase，再反向划定产品的能力边界、补充示例数据、完善拒答机制。例如设计酒吧机器人的闲聊功能，我们不可能提前写好所有剧本，只能在不断的真实测试中，逐步收敛模型的行为，让它越来越符合产品预期。

其次，是线上运营阶段，两者的思维差异则不再明显。在这个阶段，虽然两类产品经理具体做的事情不一样，但在“后验”思维的逻辑上是一致的。传统互联网产品上线后的指标监控与转化率分析，本身就是一种后验思维；大模型产品上线后同样如此，都是根据真实用户的反馈、线上产生的数据和实验结果来进行效果迭代。在这个阶段，两者都依靠后验的真实数据来驱动产品持续生长。

2.2.4 以示例数据为起点的设计思维

传统互联网产品设计的起点是原型图，而大模型产品设计的起点是示例数据——这是两类产品在日常工作中最直观的区别，也是大模型产品经理需要掌握的核心工作内容。

传统产品经理即便没有任何真实业务数据，也能通过绘制原型图、梳理交互流程，让研发团队启动开发工作，业务数据可以等到产品上线后再补充录入。但大模型产品的体验，由“输入－输出效果”决定，其需求文档的主体不再是线框图与交互说明，而是示例数据与评测集。没有示例数据，我们就无法定义模型的应答规则，无法验证模型效果，产品设计工作根本无从谈起。示例数据的价值集中体现在两个方面：

（1）示例数据是产品需求的“翻译器”：它能无歧义地告诉产研团队，“产品在什么场景下，需要输出什么样的结果”，彻底解决传统需求文档中“主观描述带来的认知偏差”问题。

（2）示例数据是需求可行性评估的依据：一个需求能不能落地，最直接的验证方式，就是用示例输入在目标模型上完成跑测，看它能否稳定输出符合预期的结果。跑测通过，需求才能立项进入研发；跑测不通过，则说明该需求在当前技术路径下不具备落地可行性。

例如，设计酒吧机器人的酒水推荐功能，最开始我们要设计的就是示例数据：用户会怎么提问？高频和长尾场景有哪些？对应的合格应答是什么？有了示例数据后，才能基于这些数据来设计 Prompt、验证模型能力、评估算力成本。如果脱离了示例数据，画出再精美的交互界面也是空中楼阁，因为我们只画了软件产品的 GUI 原型图，却没有画出模型产品的 LUI“原型图”。



★ 对应用场景种类的考量

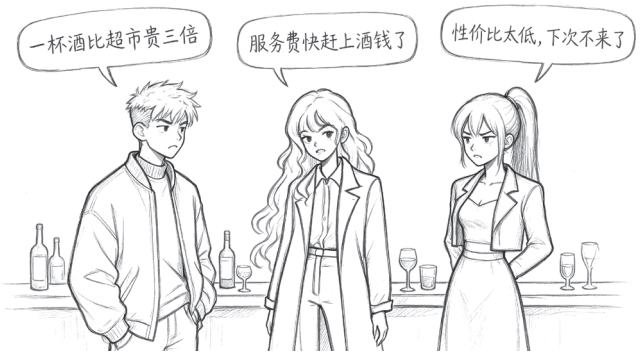
在实际工作中，若业务方提出智能化需求却无法提供业务数据，我们必须主动推动数据收集；若暂时无法获取真实业务数据，也要先模拟场景的示例数据，与业务方确认预期、对齐目标，再启动后续的研发工作。

2.2.5 解决问题是产品经理与工程师的共同工作

传统互联网研发中，“找问题”是产品经理的职责，“解决问题”是工程师的职责，边界清晰。产品逻辑与代码实现耦合度高，产品经理定需求、明确问题标准，工程师负责落地实现，权责分明。

但在大模型产品研发中，找问题、定位问题，是产品经理、研发工程师、算法工程师共同的责任，而非单一角色的职责。这是因为模型本身就是产品的一部分，一个 Badcase 的出现，可能是 Prompt 策略、训练数据、模型能力边界、工程链路等单一或多重因素导致的，无法靠单一角色定位根因。

例如，点酒智能机器人的典型 Badcase：用户问“3 人小聚的平价酒”，模型却反复推荐超预算的高端酒。产品能判断结果不符合预期，却定不了根因；研发看系统无 Bug，无法理解问题所在；算法懂模型逻辑，却不懂场景下用户的诉求。



☆ 没有软件 Bug 产品效果依旧不能保障

三方各有认识盲区，只有配合对齐业务标准与技术边界，才能快速精准定位问题、找到最优解。这也要求产品经理跳出传统“需求提报者”的定位，成为产研协作的枢纽，全程参与模型效果优化。

2.2.6 产研与算力成本时刻纳入设计考量

在传统互联网产品设计中，产品经理极少需要为服务器、带宽等资源开销焦虑，只要需求逻辑合理、具备业务价值，基本都可以直接推进落地。但在大模型产品设计中，算力成本、数据成本、产研人力成本是产品经理在需求定义的第一天就紧盯的指标，它们直接决定一个需求能不能立项、一个产品能不能活下去。

这种底层思维转变的根源，在于两者成本结构的差异：

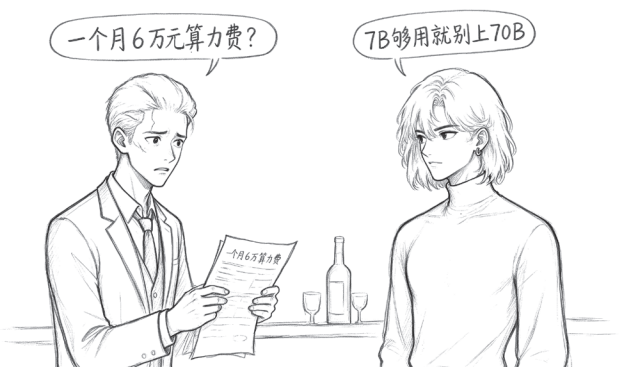
（1）传统互联网产品多为“固定成本”，边际成本较低。例如开发一个电商 App，主要的投入是前期的研发人力，这笔费用是固定的。上线后，服务器带宽成本虽然会随用户量增长，但均摊到每个新增用户头上的边际成本几乎微乎其微。因此，产品经理在设计功能时，只需专注评估“这个功能能否带来业务价值”，不用太去操心“用户每点击一次按钮，公司要花几分钱”。

（2）大模型产品依赖“算力成本”，且呈现出较高的“边际成本”特征。用户每一次使用产品产生的 token 都是真金白银；每一次为了优化效果而微调模型，都会产生高昂的训练成本。算力开销与用户的使用量、模型的参数规模（如选 7B 还是 70B 模型）、单次交互的上下文长度直接挂钩。我们可以简单用两个模型做一个对比，如表 2-2 所示。

表2-2 不同参数量的大模型调用成本对比

对比项	7B模型	70B模型	差　异
API单价	0.002元	0.02元	10倍
日均调用次数	10万次 / 天	10万次 / 天	-
单次调用平均消耗token	1000 token / 次	1000 token / 次	-
月算力成本	6000元 / 月	60000元 / 月	10倍
效果分数	83/100分	92/100分	提高9.8%

一个看似简单的智能化需求，如果调用频次非常高，上线后巨大的算力消耗有可能会击穿产品的收支平衡，直接吃掉所有利润，甚至让产品陷入“用户越多，亏得越惨”的死局。除此之外，高质量数据的持续采买与人工标注成本、高昂的算法人才成本，也远超传统互联网项目。比起软件开发的服务器资源，大模型产品的算力资源对于产品而言非常奢侈。



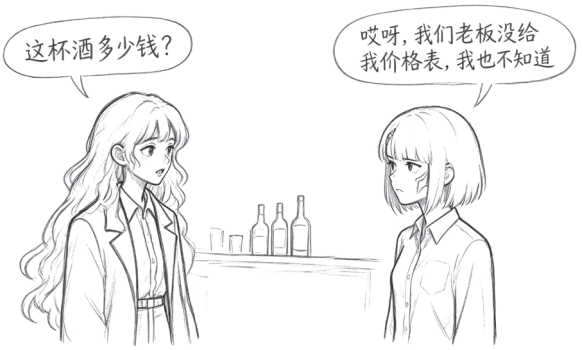
★ 模型选型对成本的影响

因此，大模型产品经理需要建立全流程的成本核算意识，在产品体验与成本之间找到精准的平衡点，比如能用小参数量模型配合工程策略解决的问题，就不盲目使用大参数量模型；能用 Prompt 工程解决的问题，就不轻易启动成本更高的微调训练，最终实现“体验达标、成本可控、商业可闭环”的目标。

2.2.7 问题可能不是模型导致的

大模型产品代表着“智能化”。但在实际落地中，很多业务场景连基础的“信息化”（即将线下的业务动作转化为系统的数字记录）都未完成。在这种缺乏数据沉淀的土壤上强行

推进智能化，就像是跨过造火药直接造导弹，显然不现实。然而，在全面向大模型转型的当下，这种问题几乎无法避免。例如，在老板没有给出酒水价格的情况下，再聪明的服务员也没办法给顾客报价。



★ 信息化不健全对智能化的影响

大模型要想在业务中输出稳定精准的结果，高度依赖覆盖面广、结构化的底层数据支撑。因此，做产品时我们要时刻警惕：当效果不佳时，问题到底是不是模型导致的？很多时候，根源并非模型不聪明，而是业务侧粗糙的数据根本“喂不饱”模型。

如果排查发现，问题确实是业务本身的信息化建设拖了后腿，产品经理需要做两手准备：

- 务实保底，先做到 60 分：在当前数据受限的场景下，思考如何通过传统的产品规则或人工策略进行兜底，保障业务能先顺畅跑起来。
- 利用模型，反向加速信息化：大模型虽然擅长处理杂乱无章的非结构化数据，但要输出精准的业务结果，高度依赖底层结构化的业务数据支撑。如果业务本身的基础数据混乱、缺失（比如酒吧没有统一的酒水价格表、库存数据），即便大模型理解能力再强，也无法给出准确的答案。这时我们可以先利用大模型来读取、清洗那些杂乱无章的线下记录，自动抽取关键信息，从而反向协助业务更快地完成信息化建设。

2.3 大模型产品的研发流程

在明确了大模型产品设计的底层思维转变后，我们需要搭建一套可落地、可复用的标准化研发流程，为产品落地提供全流程的执行框架。

与传统互联网产品的瀑布式、敏捷式研发流程不同，大模型产品的研发流程是一个以评测为主轴、以数据为驱动的闭环迭代体系。整个流程围绕“需求定义→评测验证→研发落地→效果优化→再评测迭代”的逻辑展开，每一个环节都离不开数据支撑与效果评测，这也是大模型产品研发与迭代的主要特征。

我们将整个研发流程拆解为 8 个关键环节，明确每个环节的主要目标、执行角色、执行动作与交付物，形成完整的全生命周期闭环，可作为产研过程的参考。这些节点往往是并行的或存在耦合，在工作中，不用刻意追求这种瀑布流的研发方式，参考图 2-2。

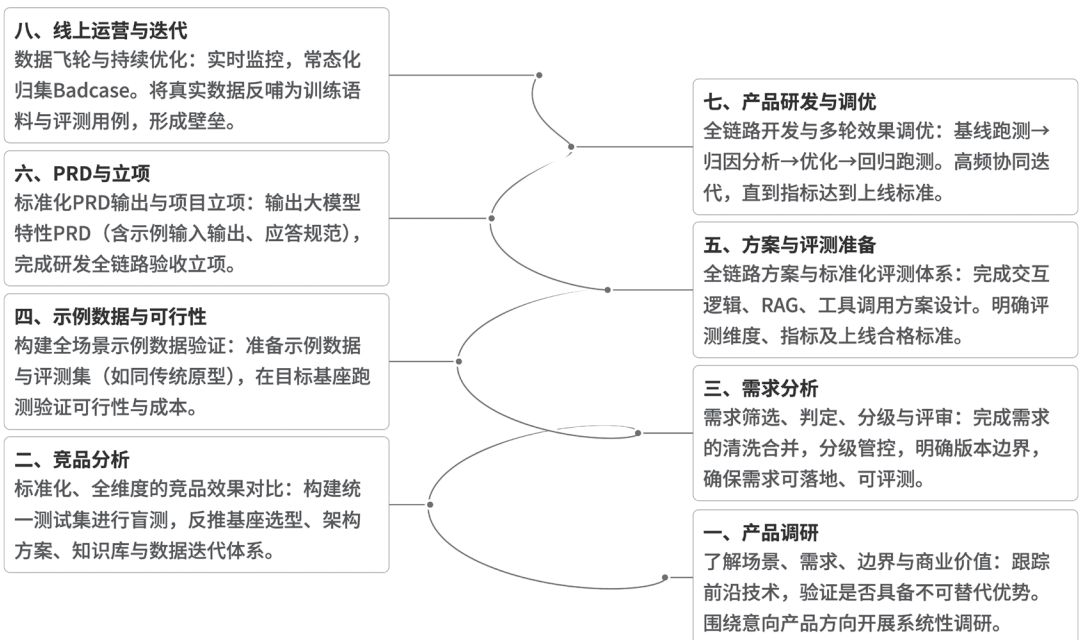


图 2-2 大模型产品研发的 8 个关键环节

2.3.1 产品调研

产品调研分为两个维度：一是产品经理的常态化行业调研，属于日常基础工作，重点是持续跟踪行业前沿技术、新兴产品形态与标杆落地案例，保持对行业的敏感度；二是项目定向深度调研，是大模型产品研发流程的正式起点，主要围绕意向产品方向开展系统性调研，为后续项目启动、需求定义提供可落地的决策依据。本节聚焦后者，即项目启动前的定向深度调研。

主要目标：

了解意向产品方向的落地场景、用户需求、技术能力边界与市场商业价值，为后续需求定义与项目决策提供完整、可参考的判断依据。

执行角色：

主责：应用侧产品经理、业务提出方、模型侧产品经理。

主要执行动作：

- 行业与技术调研：跟踪行业前沿技术概念、产品形态与落地案例；针对意向产品方向，结合现有同类产品的落地情况，初步判断技术可行性、市场空间与潜在商业价值；梳理当前主流基座模型的能力边界、成本差异与适配场景。
- 场景调研：明确产品的主要落地场景，理清场景的痛点、用户使用习惯、场景边界与业务限制。可以反复询问自己如下问题，规避伪场景：这个场景是否真的需要大模型？大模型能否解决场景的主要痛点？大模型相比传统方案是否具备不可替代的优势？

主要交付物：

《产品调研分析报告》，包含场景分析、用户画像、技术调研等内容。

2.3.2 竞品分析

竞品分析是产品经理的基础能力，但大模型竞品分析的关键点在于，除了“功能对比”外，还需要考虑“效果对比”。如果仅做功能对比，容易陷入“竞品有啥我做啥”的被动局面。例如，两个酒吧的对话机器人都有“酒水推荐”功能，但一个推荐准确率为 95% 且应答自然，另一个准确率为 70% 且频发幻觉，用户体验天差地别。

此外，大模型竞品调研存在一个天然难点：在以 GUI 为主要交互逻辑的传统互联网产品中，调研者往往能一次性纵览产品的全部信息架构，推测其整体业务逻辑非常高效；而对于以 LUI 为主导的大模型产品，其能力隐藏在对话框背后（呈“黑盒”状态）。调研者需要精心预设各类问题与场景，通过多次、多轮次、持续地向产品发起“探针”式请求，才能逐渐摸清其实际效果与能力边界。这种特性使得调研的复杂度与难度大幅提升。

主要目标：

通过标准化、全维度的竞品分析，明确行业竞争格局、竞品能力边界与差异化竞争策略，为自身的产品设计与技术选型提供参考基准。

执行角色：

- 主责：应用侧产品经理、模型侧产品经理。
- 配合：测试工程师、算法工程师。

主要执行动作：

- 功能与交互评估：还原用户真实使用路径，拆解主要业务场景的全链路交互逻辑。重点分析竞品如何应对用户的多轮追问、需求变更或异常输入，如何设计任务完成的闭环路径，以及如何真正解决痛点，借此反推其产品设计思路与用户体验策略。
- 量化效果评测：为了保证评测的公平性与客观性，需构建一套统一的、覆盖主要业务场景的测试集来横测所有竞品。建议优先采用盲测模式，重点对比关键量化指标，包括但不限于：意图识别准确率、任务完成率、答案准确率、幻觉发生率、应答自然度、拒答率及响应延迟等。
- 技术路径反推与拆解：结合黑盒测试表现、公开技术资料与行业研报等信息，反推竞品底层的技术实现逻辑，包括：基座模型选型、架构方案、知识库构建逻辑、数据迭代体系以及安全风控策略。同时，挖掘竞品的技术壁垒（如垂直领域独家数据积累、场景适配深度、算力成本控制能力等），明确行业竞争的门槛。

主要交付物：

《竞品分析报告》，包含竞品能力矩阵、经典示例输入 / 输出对比、能力边界分析与技术拆解等。

2.3.3 需求分析

需求分析是产品经理的另一个基本能力，大模型产品的需求评估逻辑与传统互联网产品大体一致。二者最大的差异在于，大模型产品需求评估过程中，需要注重需求的可行性与成本。

主要目标：

完成需求的筛选、判定、分级与评审，明确产品当前版本的需求清单，确保所有需求符合用户真实痛点、可落地、可评测、可匹配商业目标。

执行角色：

- 主责：应用侧产品经理、模型侧产品经理。
- 配合：业务提出方、算法工程师。

主要执行动作：

- 需求收集与初步筛选：整合业务方诉求、用户调研、竞品分析、行业标杆案例等多渠道的需求，完成初步清洗、去重与合并；剔除明显不符合业务目标、超出技术能力边界、无明确用户价值的无效需求，形成初步需求池。
- 需求分级与优先级排序：通过优先级对初筛的需求进行分级管控，明确当前版本的需求边界，避免需求发散。

主要交付物：

《需求分析报告》、分级需求池、项目背景与场景说明文档。

2.3.4 示例数据与可行性分析

准备示例数据与评测集，等同于传统互联网研发中的“交互与业务逻辑原型设计”，是大模型产品设计的开始。

示例数据的价值有两点：一是作为产品需求的“精准翻译器”，无歧义地告诉产研团队“产品在什么场景下，针对什么样的用户输入，需要输出什么样的结果”；二是作为需求可行性验证的评判依据，是后续技术跑测的关键素材。没有示例数据，大模型产品就没有明确的落地实现方向，更无法完成需求的可行性验证。

主要目标：

构建全场景示例数据，完成技术可行性、成本可行性验证，从源头规避无效研发投入。

执行角色：

- 主责：应用侧产品经理、模型侧产品经理。
- 配合：研发工程师、算法工程师。

主要执行动作：

- 构建全场景示例数据：联合业务团队，基于真实业务场景，构建覆盖高频常规场景、长尾边缘场景的全量示例数据，明确每个场景下的用户输入、预期输出、合格判定标准。示例：设计酒吧机器人的酒水推荐功能，需覆盖“单人独饮、多人小聚、商务宴请、特定预算、特定口味偏好、特殊人群（驾车顾客、未成年人）”等全场景的示例数据，明确每个场景的合格应答标准。
- 技术可行性跑测验证：基于构建的示例数据，在目标基座模型上进行初步跑测，验证需求的实现效果；若模型在关键场景的关键意图上表现出极差的理解力、严重的幻觉，且通过 Prompt 调

整无法改善，则判定该需求在当前技术路径下可行性存疑，需终止项目或调整需求方向。

- 成本与性能测算：结合跑测情况，估算单次交互的平均 token 消耗、API 调用成本、响应延迟，验证成本模型是否符合商业模式要求，响应速度是否满足用户体验标准，明确项目落地的成本边界。

主要交付物：

《产品应答方案设计与示例数据》《效果可行性报告》《性能及成本可行性报告》。

2.3.5 方案设计与评测集准备

这是项目进入实质性研发的前置环节，也是研发前投入最重的准备阶段。产品的全链路方案设计必须基于可行性验证通过的示例数据开展；标准化评测集的搭建可与方案设计同步推进，二者互为补充、互为校验。

执行角色：

- 主责：模型侧产品经理、算法工程师、应用侧产品经理。
- 配合：业务团队、研发工程师、测试工程师。

主要目标：

基于已验证的需求与示例数据，完成产品全链路落地的产品与技术方案设计，搭建可量化、全场景的标准化验收评测体系，完成项目全流程的成本与资源规划，为项目立项与实质性研发提供完整、可落地的执行基准。

主要执行动作：

- 全场景产品与技术方案设计：基于可行性验证通过的示例数据，完成产品全貌与细颗粒度的方案设计，内容涵盖但不限于：关键交互逻辑、模型应答规范、RAG 链路设计、工具调用逻辑、安全风控与拒答流程、多轮对话记忆策略以及各类异常场景的兜底方案等，最终输出全链路业务逻辑流程图与技术实现方案。
- 标准化评测集搭建：基于示例数据与量化效果标准，搭建一套覆盖全场景的标准化评测集。需在这一环节提前明确并死磕：评测维度、评测指标（如准确率、相关性、流畅度）、评分规则（如人工打分标准或大模型作为裁判的自动化评测规范）以及最终的上线通过标准。
- 全流程成本与资源评估：完成项目的人力成本、算力成本（训练与推理）、数据标注成本及研发周期的全面盘点与评估。制订详尽的项目排期与产研资源规划，明确各里程碑节点与交付物要求。

主要交付物：

《产品技术方案说明书》《人力与算力成本评估》《评测集与评测规则》。

2.3.6 产品文档撰写与立项评审

这一环节是将前序所有环节的成果整合为标准化的大模型 PRD，并通过正式的立项评审完成研发决策，确定核心需求与验收标准，正式启动研发工作。

整体来看，大模型产品 PRD 的篇幅，因缺少大量原型图与复杂的交互逻辑说明，通常短于传统互联网产品 PRD，但这并不代表其难度更低。传统互联网产品 PRD 的难度在于用户群体分析、产品价值梳理与交互逻辑设计；而大模型产品 PRD 的难度在于全场景示例数据与评测集的构建、需求可行性的全链路验证，对产品经理的技术理解能力、数据敏感度、场景拆解能力提出了更高的要求。

主要目标：

输出符合大模型研发特性的标准化 PRD，明确产品全貌与量化验收标准，顺利拉起产研共识并完成项目立项评审。

执行角色：

主责：应用侧产品经理、模型侧产品经理。

主要执行动作：

按照大模型产品的研发特性，完成 PRD 全模块撰写，与传统互联网产品 PRD 的差异对比如表 2-3 所示。

表2-3 大模型产品与互联网产品PRD的差异对比

文档模块	互联网产品PRD	大模型产品PRD
背景说明	重点阐述产品价值、目标用户群体、业务目标	除产品价值、用户与业务目标外，需补充说明大模型能力的依附场景、与原有产品体系的融合逻辑
竞品调研	重点分析竞品的界面设计、功能矩阵、用户群体、运营策略	重点分析竞品的效果、技术实现路径、能力边界与壁垒
可行性评估	主要评估软件开发的人力成本、研发周期、技术实现难度	需全面评估算法能力、算力资源、数据成本、数据标注难度、运营成本、商业闭环可行性

（续表）

文档模块	互联网产品PRD	大模型产品PRD
设计方案	由原型图、交互逻辑说明、业务流程图组成	由示例输入输出规则、模型应答规范、全链路逻辑流程图组成
验收标准	以功能实现度、交互还原度、业务流程通顺为验收标准	以标准化评测集的跑测结果、量化效果指标为验收标准，评测集是PRD的附件
需求迭代逻辑	基于用户反馈与业务数据，迭代产品功能与交互流程	基于Badcase与评测数据，迭代数据、策略与模型，持续优化模型效果

主要交付物：

《大模型产品需求文档（PRD）》及配套的示例数据库、标准化评测集、全链路方案流程图、应答方案设计等附件。

2.3.7 产品研发与策略调优

这是产品从方案落地为可交付版本的关键执行阶段，也是产品、研发、算法深度绑定、高频协同的阶段，重点是以前序定好的标准化评测集为唯一标尺，通过多轮调优让产品效果稳定达到上线验收标准。

主要目标：

基于 PRD 与标准化评测集，完成产品全链路开发与多轮效果调优，使产品关键效果指标稳定达到上线验收标准，完成上线前的所有验收测试。

执行角色：

- 主责：算法工程师、研发工程师、测试工程师、模型侧产品经理。
- 配合：应用侧产品经理、数据产品经理。

主要执行动作：

- 初始版本开发：基于前序定好的产品与技术方案，完成产品全链路开发，主要包括底层 System Prompt 编写、RAG 检索链路搭建、工具调用逻辑开发、多轮对话策略落地、异常场景兜底方案实现，输出可跑测的产品初始版本。
- 基线效果跑测：初始版本产出后，立即用标准化评测集完成全量自动化跑测，得出版本的“基线（Baseline）成绩”，明确当前版本与上线验收标准的差距，锁定主要的不达标场景与问题。
- 多轮调优迭代：围绕跑测中涌现的 Badcase，产研团队开展联合归因分析，匹配对应的优化

路径：是优化 Prompt 策略、调整 RAG 检索逻辑、补充知识库内容，还是补充微调数据优化模型的能力。每完成一轮优化，都得在全量评测集上完成回归跑测，验证优化效果的同时，避免出现局部优化、整体效果退化的问题。这个“调试 – 跑测 – 归因 – 再调试”的循环会持续进行，直到产品关键效果指标稳定跨过上线验收的“及格线”。

- 上线前全量验收测试：关键效果指标达标后，完成全场景全量回归测试、性能压力测试、异常场景兜底测试，最终输出完整的验收评测报告，确认版本达到上线标准。

主要产出物：

可上线的产品安装包 / 部署包、全量版本评测报告、Badcase 归因与优化台账、策略调优过程文档、上线验收报告、产品交付与运维文档。

2.3.8 线上运营与效果迭代

大模型产品上线只是迭代的开始。传统互联网产品的上线，意味着主要功能研发完成、进入稳定运营阶段；而大模型产品的上线，只是产品全生命周期迭代的起点，线上运营的重点是搭建一套可持续的“数据飞轮”：实时监控产品关键运行指标，将用户真实交互中的问题沉淀为新的 Badcase，通过对这些真实场景数据的持续清洗、分析与优化，反哺模型的策略体系与能力边界。产品在线上运行得越久，沉淀的真实业务数据越丰富，其建立起的场景壁垒就越深厚。

主要目标：

搭建线上产品的监控与应急体系，建立常态化的 Badcase 归集与效果迭代机制，通过线上真实业务数据持续优化产品效果，形成可持续的数据飞轮，构建产品的长期竞争力。

执行角色：

- 主责：数据产品经理、应用侧产品经理、模型侧产品经理、算法工程师。
- 配合：研发工程师、测试工程师、业务提出方。

主要执行动作：

- 线上实时监控与告警：搭建线上监控体系，实时监控 API 调用指标、模型关键健康指标（意图准确率、任务完成率、拒答率、合规拦截率等）、算力成本指标、安全风控数据，一旦出现指标异动，快速触发告警与应急响应。
- Badcase 常态化归集与分析：建立自动化的 Badcase 归集机制，将用户点踩、不符合业务预期、触发安全拦截的真实对话数据精准归集，定期组织产研团队开展分类、归因与深度分析，

定位模型的能力盲区，明确优化方向。

- 持续的效果优化与迭代：针对线上归集的 Badcase 与用户需求变化，持续优化 Prompt 策略、知识库质量、工具调用方案，在必要时开展模型微调训练。每一次优化上线前，都可以通过标准化评测集的全量回归测试，确保优化效果达标，不出现能力退化。
- 高质量数据的持续生产：将线上真实场景的对话数据、Badcase 案例转化为优质的示例数据、训练语料与评测用例，持续完善数据体系，形成可持续的数据生产流水线，不断提升模型的场景适配能力，构建产品的长期壁垒。

主要交付物：

线上用户反馈与 Badcase 数据库、月度产品效果评测报告、产品迭代优化方案、数据集迭代与更新方案、线上运行监控月报。

2.4 模型侧产品经理的工作范围与团队协作变迁

由于模型侧产品经理的工作范围与团队协作方式变动最大，而且本书更加关注模型侧的产品经理，因此本节专门对模型产品经理的工作范围与职能进行重点补充。

2.4.1 工作内容重构：原型工作的弱化与新的工作项

之前，我们提到过，在 GUI 设计过程中，画原型图是产品经理的基本工作之一，然而在 LUI 设计过程中，画原型图的工作比重会下降。越是纯粹的大模型应用，原型图工作占比越少，模型侧产品对原型图的需求大幅降低，核心工作几乎不依赖原型图，仅需设计简单的承载界面。在模型侧的实际工作中，我们更加关注以下几个方面。

1. 示例数据构建

基于设计好的模型应答规则，构建全场景的输入输出示例数据。这既是把产品需求精准传递给算法、研发团队的关键载体，也是后续 Prompt 设计、模型调优、效果评测的主要依据，和我们前面提到的“以示例数据为起点的设计思维”相呼应。

2. 模型应答方式设计

这是 LUI 设计的重点，也是模型侧产品经理最基础的工作。产品经理需要基于业务场景，

定义不同条件、不同用户意图下模型的应答规则、语气风格、内容边界。不同于GUI设计中“一个按钮对应一个固定跳转”的一对一逻辑，复杂业务场景下，用户意图与模型应答是“多对多”的复杂关系，这是设计的重点，也是难点。

例如，车机场景下，用户说“我有点热”，模型需要结合场景条件作出差异化应答：车辆低速行驶时，可询问并帮助用户开启车窗；车辆高速行驶时，优先帮助用户开启空调；高速行驶且空调已开启时，直接帮助用户调低温度并给出安抚话术。同一个用户需求，在不同场景下需要不同的应答与动作，这都需要产品经理提前定义清楚。



★ 高速开车过程中热，优先开空调

3. 数据分类与分布把控

清晰定义并监控用户输入数据的场景分类与占比分布，这是模型优化优先级判断的重要依据。只有明确了用户输入的结构，才能精准找到影响绝大多数用户体验的主要问题。例如，当前产品中点餐类任务占比 20%、用户满足率 90%，投诉类任务占比 1%、用户满足率 50%，尽管投诉类任务的满足率更低，但影响范围极小，应当优先优化覆盖用户更广的点餐类任务，避免陷入“抓小放大”的优化误区。

4. 模型调用流程设计

定义从用户输入到模型输出的全链路流程：用户的一句话输入，需要经过哪些前置工程处理、需要调用几次模型、每次调用的 Prompt 规则是什么、模型输出后需要经过哪些工程判断，最终才能呈现给用户。这是模型侧产品经理对产品全链路可控性的关键保障，替代了传统互联网产品的业务流程图。

5. 模型效果评测

在产品初步搭建完成后，通过标准化评测集，全面评估模型效果是否达到业务预期；同时基于线上真实对话数据与 Badcase，定位模型的主要短板，明确后续的优化方向。这是模型侧产品经理贯穿全流程的重点工作，也是“没有评测就没有结论”思维的具体落地。

6. 模型策略优化

基于评测结果与 Badcase 分析，持续优化模型的全链路策略，包括 Prompt 工程、工作流设计、知识库优化、工具调用规则调整等，让模型在真实业务场景中的表现持续贴合产品的预期效果。

2.4.2 产品团队的角色变化：与算法、数据、研发的协作模式

模型侧产品经理变化的不仅是工作内容，协作团队和协作方式也有变化。主要变化分为以下两部分。

1. 协作对象发生转移

模型侧产品经理与 UI 设计师、前后端工程师的配合度大幅弱化，主要协作对象变成了算法工程师、数据工程师、大模型工程研发工程师。这一转变的主要根源在于，纯粹的模型侧产品，关键竞争力是模型的效果与能力，GUI 界面只是承载模型能力的容器，而非产品的主要价值，因此 UI、前后端研发的工作占比变低，不再是产品研发的关键环节。

2. 协作模式逐渐转向“双向深度共创”

传统互联网产品中，产品经理与研发人员的分工边界清晰，而模型侧产品的研发中，产品经理与算法、数据团队的边界逐渐消融。一个 Badcase 的出现可能是 Prompt 策略的问题，可能是训练数据的问题，也可能是模型本身的能力边界，产品经理懂业务但不懂模型底层逻辑，算法工程师懂技术但不懂业务场景的细节要求，只有双方深度配合、联合归因，才能快速定位问题根源，找到最优解决方案。

在这种新的协作模式中，产品经理既要给算法、数据团队明确业务目标与效果标准，也要和团队一起，全程参与到模型效果评测、Badcase 归因、策略优化的全流程中，和技术团队共同对模型的最终效果负责。

2.4.3 数据能力升级：比算法更懂业务数据

数据能力是模型侧产品经理的关键竞争力，而这种能力，与传统互联网产品经理的数据能力有着显著区别，重点是完成了“从数据的分析者，到数据的生产者”的升级。传统互联网产品经理使用的是“用户行为数据”，这些数据是用户使用产品过程中自然产生的；而大模型产品经理生产的是“模型训练与评测数据”，这些数据是主动设计和构建的，直接决定了模型的能力边界与效果上限。

在传统互联网产品体系中，产品经理的数据能力主要是数据分析能力：产品经理是数据的使用者，基于现成的用户行为数据、业务数据，分析产品的问题、验证优化的效果、挖掘用户的需求，目的是“用数据找问题、验证结论”。数据的生产、清洗、存储由数据研发团队负责，产品经理只需要完成后续的分析与应用。

但在大模型产品体系中，数据是模型能力的源头，模型的效果上限由数据的质量与覆盖范围决定。无论是示例数据、训练数据，还是评测数据，都得深度贴合业务场景，而产品经理应该成为整个团队里最懂业务场景、最懂用户需求、最懂产品效果标准的人。这就要求模型侧产品经理来完成数据能力的关键升级：从被动的数据分析者，变成主动的业务数据定义者、高质量数据生产者、数据闭环的搭建者。

这种能力升级，集中体现在以下三个维度。

（1）业务数据的定义能力：模型侧产品经理必须比算法工程师更懂业务数据，对产品的数据分布有着清晰认识，即哪些是用户真实的高频输入场景、哪些是容易出错的长尾场景、什么样的输出是符合业务要求的合格结果、哪些数据是能提升模型效果的优质数据。算法工程师懂数据的结构、训练的逻辑，但只有产品经理能定义数据的业务价值，这是数据工作的基础前提。

（2）高质量数据的生产能力：产品经理需要主导完成全链路高质量数据的生产，包括核心场景的示例数据、模型微调的训练数据、全场景覆盖的评测数据、Badcase 优化的补充数据。这不是简单的文本撰写，而是基于业务场景、用户需求、产品目标，构建标准化、高质量、全场景覆盖的数据体系，这是模型效果优化的重要基础，也是模型侧产品经理不可替代的独特价值。

（3）数据闭环的搭建能力：模型侧产品经理需要和数据产品经理搭建一套可持续的“数据飞轮”：把线上用户的真实对话、Badcase、用户反馈转化为新的优质数据，反哺模型的策略优化与能力迭代，再通过线上效果验证，持续沉淀更多的业务数据。这套数据闭环是

大模型产品长期竞争力的关键，而产品经理是这套闭环的主要设计者与操盘者。

2.5 小结

本章介绍了大模型时代产品经理的职能变迁。首先介绍大模型产品经理的角色定位，将其划分为应用侧、模型侧、数据侧和平台侧 4 类，并分析了各角色的能力要求。然后，总结了大模型产品设计的 7 种底层思维转变，包括消费者与生产者的思维差异、对大模型效果的怀疑思维、从先验到后验的思维切换、以示例数据为起点的设计思维、产研共同解决问题的协作方式、成本纳入设计考量，以及问题可能不在模型本身的判断等。最后，梳理了大模型产品从调研到线上运营的全流程研发体系，并重点补充了模型侧产品经理的工作内容重构、团队协作方式变化和数据能力升级，帮助读者全面认识大模型产品经理在新时代的职能定位与能力要求。