LEARNING NOTE 01

AI 产品经理学习笔记 01

先成为产品经理，再谈 AI

这是一份面向 AI 产品经理方向的个人学习记录。我希望记录的不是零散的 AI 名词，而是一套能够真正用于分析产品、设计方案和复盘项目的基础框架。

> *AI 产品经理首先是产品经理，其次才是 AI 产品经理。*

AI 改变了产品能够解决问题的方式，但没有改变产品工作的起点：理解用户、判断问题、设计方案、推动落地并验证结果。

学习记录 · 公开整理版

2026 · AI Product Management

第 0 章：AI 产品岗的基础门槛

学习 AI 产品，并不意味着一定要成为算法工程师，也不意味着需要先背下大量模型、框架和技术名词。至少在入门阶段，我更希望自己能够真正理解：一个 AI 产品，是怎样从用户问题出发，一步步运行起来的。

完整链路大致可以拆成：

用户需求 → 产品方案 → AI 能力选择 → 产品流程 → 模型调用 → 异常处理 → 数据验证 → 产品迭代

如果只是会调用一个大模型 API，写几个 Prompt，再做出一个可以运行的 Demo，其实还不能说明自己已经形成了 AI 产品思维。真正需要理解的是：为什么这里需要 AI，以及 AI 被放进整个产品流程以后，究竟解决了什么问题。

0.1 这一阶段到底要学到什么程度?

完成这一阶段后，我希望自己至少能够回答下面十个问题。

1. 用户是谁?

产品不是为一个抽象的“用户”服务。用户不同，问题、场景和解决方案往往都会发生变化。

- 谁会真正使用这个产品?
- 是普通消费者，还是企业用户?
- 是内容运营、游戏策划、客服、销售，还是开发人员?
- 是新用户、老用户，还是高频核心用户?

2. 用户遇到了什么问题?

不能从“我要做一个 AI 功能”开始思考，而应该先问：用户原本在哪一个环节遇到了困难？AI 应该解决的是一个已经存在的问题，而不是为了使用 AI 而制造一个问题。

- 信息查找效率低；
- 重复工作太多；
- 内容生产成本高；

- 信息分散；
- 判断成本高；
- 个性化程度不足；
- 操作流程过于复杂。

3. 这个问题为什么值得解决?

一个问题存在，并不意味着一定值得开发产品解决。产品经理的重要工作之一，就是决定什么值得做，什么暂时不值得做。

- 有多少用户遇到?
- 多久会遇到一次?
- 问题造成的影响有多大?
- 用户现在有没有其他替代方案?
- 解决问题带来的收益有多大?
- 实现这个方案需要多少成本?

4. 为什么需要 AI?

并不是所有问题都需要 AI。如果一个普通的规则系统、数据库查询或者传统搜索就能稳定解决，就没有必要强行引入大模型。只有当 AI 的能力与用户问题真正匹配时，引入 AI 才有意义。

- 自然语言理解；
- 非结构化信息处理；
- 内容生成；
- 信息归纳；
- 多轮交互；
- 个性化生成；
- 复杂任务拆解。

5. 为什么使用 Prompt、RAG 或 Agent?

不同技术方案应该对应不同的问题：如果主要问题是让模型按照固定要求输出内容，可以首先考虑 Prompt；如果模型缺乏企业内部知识或特定领域资料，可能需要知识库、检索或 RAG；如果模型需要判断任务、选择工具并执行多个步骤，才可能涉及 Agent 或更复杂的 Workflow。学习这些概念，比记住定义更重要的是理解它们分别在解决什么产品问题。

6. 产品流程应该怎么设计?

真正的 AI 产品通常是一套完整流程，而不是一个模型调用接口。

- 用户从哪里进入?
- 第一步做什么?
- 系统需要收集哪些信息?
- 模型什么时候被调用?
- 是否需要确认用户意图?
- 是否需要调用外部工具?
- 结果如何展示?
- 用户能否修改?
- 用户不满意怎么办?
- 系统失败怎么办?

7. 模型可能在哪里出错?

传统软件通常强调确定性，而 AI 产品存在明显的概率性。因此，设计 AI 产品时不能只设计“成功路径”，还需要设计 AI 出错时产品怎么办。

- 理解错用户意图;
- 遗漏关键信息;
- 产生事实错误;
- 输出格式错误;
- 出现幻觉;

- 生成不符合业务要求的内容；
- 调用错误工具；
- 输出不安全或违规内容。

8. 出错以后怎么办？

一个完整的 AI 产品不能建立在“默认模型永远回答正确”的假设上。异常处理本身也是产品体验的一部分。

- 重新生成；
- 让用户补充信息；
- 修改 Prompt；
- 切换模型；
- 降级到传统搜索；
- 调用其他工具；
- 触发人工审核；
- 提供结果反馈入口；
- 明确告诉用户当前能力边界。

9. 用什么指标判断效果？

“感觉不错”不能成为产品效果判断标准。如果连成功标准都没有定义，就很难判断一个 AI 功能究竟有没有真正创造价值。

- 任务完成率；
- 用户采纳率；
- 输出正确率；
- 用户修改率；
- 生成成功率；
- 平均完成时间；
- 用户满意度；

- 错误率；
- 人工介入率。

10. 下一版本如何迭代?

AI 产品不是做完第一版就结束。需要结合用户反馈、行为数据、模型表现和异常案例，继续定位真正的问题，再决定下一轮优化什么。

- 是需求本身判断错了?
- 是产品流程有问题?
- 是 Prompt 不稳定?
- 是上下文不足?
- 是模型能力不够?
- 是工具调用失败?
- 还是输出方式不符合用户习惯?

这一阶段的产品闭环

发现问题 → 设计方案 → 上线验证 → 收集数据 → 定位问题 → 继续迭代。

第一章：传统产品基本功

1. 什么是产品经理?

产品经理的核心工作不是写 PRD、画原型、开会或做 PPT。这些都是产品工作中的工具或者交付形式。

> *产品经理真正需要负责的是：发现值得解决的问题，并设计一个在用户价值、商业价值和实现成本之间相对合理的解决方案。*

可以简单概括成：

发现问题 → 判断问题 → 设计方案 → 推动落地 → 验证结果

AI 产品经理并没有改变这套基本逻辑。AI 只是增加了一种新的技术能力和解决问题的方式。所以，如果传统产品的需求分析、流程设计和产品判断都没有掌握，仅仅学会 Prompt、RAG 或 Agent，也很难真正做好 AI 产品。

2. 什么是用户需求?

用户说出来的话，并不一定就是真实需求。

> *“给这个 AI 加一个联网搜索功能。”*

乍一看，这是一个需求。但严格来说，它更接近于：用户直接提出了一个解决方案。继续追问后可能发现，用户真正遇到的问题是：“AI 经常不知道最近发生的事情。”再继续追问，可能得到更底层的原因：“回答中的信息必须足够新。”

到这里，真正需要解决的问题才逐渐清楚：用户需要的是信息时效性。

“联网搜索”只是解决信息时效性问题的一种方式。其他方案还可能包括：

- 定期更新知识库;
- 接入结构化数据库;
- API 查询;
- 搜索引擎;
- RAG;
- 人工维护。

需求分析中的四层区分

用户表达 ≠ 用户问题 ≠ 用户需求 ≠ 产品方案

不能把用户提出的功能直接当成产品答案。产品经理需要继续往下追问：用户为什么想要这个功能?

3. 需求分析的基础框架

当看到一个需求以后，我目前认为至少应该继续问五个问题。

第一问：谁遇到了这个问题?

不能只写“用户需要……”。“用户”这个词太宽泛，需要继续拆分。

- 新用户;
- 老用户;
- 高频用户;
- 游戏策划;
- 内容运营;
- 客服;
- 销售;
- 开发人员。

不同用户面对的问题可能完全不同。例如，同样是一个 AI 内容生成工具：游戏策划关注的可能是角色设定一致性；运营人员关注的可能是能否快速生成多个版本进行测试；开发人员关注的可能是能否稳定接入已有系统。

所以需求分析的第一步，通常不是讨论功能，而是明确到底是谁遇到了问题。

第二问：用户在哪个场景下遇到问题?

> *“AI 帮助生成游戏角色文案。”*

这个描述仍然过于宽泛。继续补充场景：游戏运营人员在活动上线前，需要在较短时间内生成多版角色宣传文案，用于后续筛选或测试。这样以后，产品场景就清晰了很多。

把抽象需求变成具体场景

谁，在什么时间，为了完成什么任务，遇到了什么问题。

第三问：用户现在是怎么解决的?

一个很容易忽略的问题是：用户现在通常已经有解决方案。AI 产品真正竞争的对象，并不一定是另一个 AI 产品。很多时候，它竞争的是原来的人工工作流程。

例如游戏运营人员制作角色宣传文案，目前可能需要：

1. 查看角色设定；
2. 阅读历史宣传内容；
3. 手动寻找卖点；
4. 写出第一版文案；
5. 反复修改；
6. 找其他人员审核。

那么设计 AI 产品之前，需要先理解这条现有流程。否则就无法判断：AI 到底应该替代哪一步、辅助哪一步，又应该保留哪一步。

第四问：现有方案哪里不好?

- 太慢；
- 太贵；
- 太复杂；
- 容易出错；
- 学习成本高；
- 重复劳动多；
- 信息分散；
- 个性化不足。

只有理解旧方案的问题，才能进一步定义新方案的价值。例如，如果用户真正的问题是“生成速度慢”，产品应该优先验证效率；如果核心问题是“质量不稳定”，产品就不能只关注生成速度。产品目标必须与用户原本的痛点对应。

第五问：这个问题值得解决吗?

用户价值：问题被解决以后，用户能够得到什么?

发生频率：这个问题是每天发生、每周发生、每月发生，还是一年才发生一次?

痛苦程度：如果不解决，会带来什么影响？只是轻微不方便，还是会严重影响任务完成?

实现成本：为了这个问题，需要投入多少开发、设计、模型调用和维护成本?

如果一个问题极低频，却需要投入大量时间开发，那么它的优先级可能并不高。

4. 真需求与伪需求

一个很粗略但实用的判断框架是：

$$\text{需求价值} \approx \text{用户数量} \times \text{使用频率} \times \text{痛苦程度} \times \text{解决收益}$$

这里并不是要真的计算一个精确数字，它更像是一种帮助自己判断需求价值的思考方式。

需求 A

10,000 人每周遇到一次问题，每次会浪费约 30 分钟。这种问题通常值得认真研究。

需求 B

10 个人一年遇到一次，每次只浪费约 2 分钟。除非它存在其他重要价值，否则优先级通常不会特别高。

所以，“有人提出需求”并不等于“这个需求值得立即开发”。

5. 需求优先级

产品资源永远有限。因此，需求分析之后还需要解决：先做什么？

一个比较简单的方法是：

$$\text{Impact} \times \text{Effort}$$

Impact：解决以后能产生多大价值？

Effort：需要投入多少开发和维护成本？

简单理解，高价值 + 低成本通常应该优先；低价值 + 高成本通常需要谨慎。

RICE

Reach：能够影响多少用户？

Impact：对这些用户产生多大影响？

Confidence：对于当前判断有多大把握？

Effort：需要投入多少资源？

对我来说，学习这些方法真正重要的并不是背公式，而是能够回答：为什么这个需求应该排在另一个需求前面？如果无法解释优先级背后的判断依据，那么框架本身也没有太大意义。

6. 用户流程

产品经理需要能够把一个用户完成任务的过程画出来。例如，一个最基础的 AI 问答产品可以表示为：

用户打开产品 → 输入问题 → 系统检查输入是否合法 → 调用模型 → 模型生成结果 → 展示结果 → 用户是否满意？

如果满意，则结束；如果不满意，则可以重新生成、修改问题或提交反馈。

流程图的意义并不是把页面画得复杂，而是帮助自己看清：

- 用户需要做什么；
- 系统需要做什么；
- 哪些地方需要判断；
- 哪些地方可能失败；
- 失败之后应该去哪里。

7. AI 产品流程有什么不同？

传统软件往往具有较强的确定性，例如：

用户点击 A → 系统执行 B → 得到 C

但 AI 产品中会出现更多不确定状态，例如：

用户输入 → AI 理解用户意图 → 理解是否正确？ → 模型生成 → 结果是否正确？ → 格式是否符合要求？ → 内容是否安全？ → 是否需要调用其他工具？

因此，AI 产品流程中需要特别考虑两个问题：概率性和异常分支。

产品设计不能只考虑“AI 正常工作时，用户怎么使用”，还必须考虑“AI 判断错误、生成错误、工具调用失败的时候，产品应该怎么办”。

8. 什么是 PRD?

PRD 的英文全称是 Product Requirements Document，也就是产品需求文档。一个基础 PRD 至少应该说明以下内容。

1. 背景：为什么要做这个功能或产品?

2. 用户问题：用户当前遇到了什么问题?

3. 产品目标：希望解决什么?

4. 目标用户：谁会使用?

5. 使用场景：用户会在什么时候、什么情况下使用?

6. 产品流程：用户如何完成任务?

7. 功能需求：具体需要实现哪些功能?

8. 异常状态：系统或用户操作失败以后怎么办?

9. 数据指标：如何判断产品是否成功?

10. 验收标准：做到什么程度才能认为需求完成?

> **PRD 的本质**
>
> 不是“写一篇文档”，而是把问题、方案、边界和验收方式描述清楚，让不同角色对要做的事情形成一致理解。

9. AI PRD 需要额外说明什么?

AI 产品的 PRD 除了传统产品内容之外，还需要进一步描述 AI 能力本身。我目前认为至少需要关注下面几个方面。

Model Input

模型到底会接收到什么内容？例如：用户输入、系统规则、历史对话、用户资料、数据库内容、检索结果。

Prompt

系统通过什么方式指导模型完成任务？不能只写“调用大模型生成结果”，而需要进一步定义模型承担什么角色、需要遵守什么规则、完成什么任务。

Context

模型生成答案时，需要携带哪些上下文？例如：用户历史信息、角色设定、游戏世界观、历史文案、企业知识、当前任务数据。

Output

模型最终应该返回什么？最好定义清晰的输出结构，因为它明确说明了模型输出需要被产品如何使用。

```
title
selling_points
copy
risk_warning
```

这要比简单写“AI 生成营销文案”清晰得多。

AI Boundary

AI 能做什么？不能做什么？哪些场景必须拒绝？哪些场景应该提醒用户？哪些情况下需要人工介入？

异常处理

如果模型没有理解用户、输出错误、返回空内容、输出格式异常或调用工具失败，产品分别应该怎么办？

Evaluation

如何判断模型表现是否足够好？不能只验证“功能有没有运行”，还需要验证“AI 输出是否真的满足任务要求”。

10. Figma 学到什么程度？

学习 AI 产品经理，并不意味着需要把自己训练成专业 UI 设计师。至少在第一阶段，我认为更重要的是能够使用 Figma 把产品逻辑表达清楚。

需要掌握的基础能力包括：

- Frame；
- Text；

- Shape；
- Auto Layout；
- Component；
- Prototype；
- 页面跳转；
- Overlay；
- 基础交互。

重点不是追求复杂的视觉设计，而是能够把页面结构、用户路径、状态变化和交互关系表达出来。对于产品经理来说，原型首先是一种沟通工具。

它需要帮助自己和其他人理解：用户从哪里进入？点击之后发生什么？系统如何响应？异常情况如何处理？不同页面之间是什么关系？

> *先把产品逻辑表达清楚，再考虑把界面做得更漂亮。*

写在这一阶段学习之后

整理完这一部分以后，我对“学习 AI 产品”这件事情的理解也发生了一点变化。

以前很容易把注意力集中在 Prompt、RAG、Agent、大模型、Workflow……这些看起来更加“AI”的东西上。

但继续往下学习以后会发现：真正决定一个 AI 功能是不是产品的，往往不是用了多少 AI 技术，而是能不能解释清楚下面这些问题：

- 用户是谁；
- 他遇到了什么问题；
- 这个问题值不值得解决；
- AI 为什么比原来的方案更合适；
- 产品应该怎么运行；
- AI 出错以后怎么办；

· 最后怎么证明它真的有效。

所以，对我来说，第一阶段最重要的并不是继续堆更多 AI 名词，而是先补齐传统产品基本功。需求分析、用户场景、流程设计、PRD、原型、指标和迭代，这些看起来没有那么“AI”，却决定了后面学到的 Prompt、RAG 和 Agent 最终能不能真正变成一个产品。

这一阶段给自己的学习原则

不要从“我会什么 AI 技术”出发寻找产品，而要从“用户有什么问题”出发判断是否需要 AI。

下一阶段，再继续进入大模型基础认知，以及 Prompt、RAG、Agent 等 AI 产品能力。