软件卖服务，还是服务卖软件

SaaS 这门最好的生意，怎么走到 AI 时代的十字路口

咕噜

费曼式写法 · 由多个 AI 代理撰写与互相审校

导读

SaaS 这门最好的生意，怎么走到 AI 时代的十字路口

2022 年底 ChatGPT 出现后，投资圈流传起一张让人心里发凉的对比图：某家 SaaS 公司花了十年、雇了几百个工程师做出来的产品，如今任何人对着一个聊天框敲一句话，几秒就能得到差不多的东西。于是一个问题冒了出来，而且越想越睡不着——

> 如果一个通用 AI 随口就能干你软件干的活，那 SaaS 这门生意，凭什么还值钱？

这本书，就是要老老实实回答这个问题。但它偏不从 AI 讲起。

因为「AI 会不会杀死 SaaS」这个问法，本身就问错了。要判断一场技术革命会不会掀翻一门生意，你得先真正搞懂：**这门生意当年到底靠什么赚钱？**护城河具体是哪几道、订阅凭什么成立、市场为什么肯给一家亏钱公司十倍营收的估值。搞不清它怎么活的，就没法判断它会怎么死，或者怎么变。

所以我们走一条更笨、也更靠谱的路：先回到**过去**，把这台「史上最好的生意」的机器拆开，看清每一个齿轮；再看**现在**，AI 到底动了软件的哪一层，引出一场三方混战——套壳的虚惊、老玩家的副驾、新物种的数字员工；最后推演**未来**，护城河会搬到哪、钱会怎么收、谁会死谁会活。

读完你会拿到一把尺子：面对任何一个 SaaS，只问一句「它的价值，压在界面上，还是压在墙里」，就能大致判断它在 AI 时代的命运。你也会明白，为什么最后所有的变化，都浓缩成名字里两个词的一次对调——从「软件即服务」，到「服务即软件」。

不用懂代码，也不用懂金融。你只要对「一门好生意是怎么运转、又怎么被时代改写的」有点好奇，就够了。

先别急着问「AI 会不会杀死 SaaS」，先想清楚 SaaS 当年到底靠什么赚钱。那个让投资人睡不着的答案，藏在这条线的尽头。

第一章 · 一个让投资人睡不着的问题

2013 年，如果你想开一家公司，最聪明的选择几乎写在明面上：做一家 SaaS 公司。那几年，投资人见面聊的都是同一门生意，好到不像真的。

好在哪？我们先把这个缩写拆开。SaaS（Software as a Service，软件即服务）的意思是：你不再买一份软件装在自己电脑上，而是**按月按年租用**一份跑在别人服务器上的软件。你付租金，它一直更新，你随时能用。Salesforce 卖客户管理、Slack 卖团队聊天、Zoom 卖开会——都是这个路数。

为什么这门生意让人睡得香？因为它几乎把生意里所有让人焦虑的部分都解决了。

- **收入是可预测的。**卖一台冰箱，卖完这单就结束了，下个月得重新找客户。SaaS 卖的是订阅，这个月的客户大概率下个月还在，收入像水管里的水，一直流。
- **多做一份几乎不要钱。**软件写好一次，第一百万个用户用的还是同一份代码。多一个客户，成本几乎为零，赚的钱直接变成利润。
- **客户很难走。**公司全员把数据、流程、习惯都搬进了你的软件，换一家的代价高得吓人，于是他们年复一年地续费。

> 大白话
>
> 一句话：SaaS 是一门「一次做好、反复收租、客户还赖着不走」的生意。这三点凑齐，几乎是商业世界里最舒服的位置。

市场也是这么给钱的。一家传统软件公司，市值可能是它一年利润的十几倍；而一家还在**亏钱**的 SaaS 公司，市值却常常给到它一年营收（一年收进来的钱，还没扣成本）的十倍甚至二十倍。投资人赌的不是它今天赚多少，而是那股「水管里的水会一直流、还越流越大」的确定性。这份确定性，值很多钱。

然后，2022 年底，ChatGPT 出现了。

一开始大家只当它是个会聊天的玩具。但很快，一张让人心里发凉的对比图开始在投资圈流传：某家 SaaS 公司花了十年、雇了几百个工程师，做出一个帮你写营销文案的产品；而现

在，任何人对着一个聊天框敲一句话，几秒钟就能得到差不多的东西。那家公司值几十亿美元的护城河，看上去薄得像张纸。

于是那个让投资人半夜睡不着的问题冒出来了，而且越想越具体：

> 如果一个通用的 AI 模型，随口就能干你的软件干的活——那你这家 SaaS 公司，凭什么还值钱？客户为什么还要付你租金？

这个问题分量很重，因为它质疑的不是某个功能，而是整门生意的地基。上面说的那三根让人睡得香的支柱——可预测的收入、几乎为零的成本、走不掉的客户——每一根都可能被 AI 撬动：如果 AI 让做软件变得极其便宜，那「一次做好」还稀不稀奇？如果客户能随时让 AI 现做一个替代品，那他们还「走不掉」吗？

接下来整本书，就是要老老实实回答这个问题。但我想先说清楚一件事：**「AI 会不会杀死 SaaS」这个问法，本身就有点问错了。**

因为要判断 AI 会不会掀翻一门生意，你得先真正搞懂这门生意当年到底靠什么赚钱——护城河具体是哪几道、订阅模式凭什么成立、那十倍营收的估值背后是什么逻辑。搞不清楚它怎么活的，就没法判断它会怎么死，或者怎么变。

所以我们不从 AI 讲起。我们先回到过去，去看看这门「史上最好的生意」，它的机器内部到底长什么样。想清楚了这个，再回来看 AI，那个睡不着的问题，答案会清晰得多。

这本书想带你回答的，其实是一个问题：**当做软件本身变得几乎免费，一家软件公司还剩下什么值钱的东西？**

第二章 · 从卖光盘到租软件

要弄懂 SaaS 后来为什么值那么多钱，得先看清它到底革了谁的命。有意思的是：**它革的不是技术的命，是收钱方式的命。**

先看看它取代的是什么

时间倒回到 2000 年。那时候你想给公司装一套管客户的软件，流程大概是这样：花几十万甚至几百万买一份许可证（License，也就是「允许你用这份软件」的一次性授权），软件装进你自己机房的服务器里，再雇一队 IT 人员负责维护、升级、修 bug。

这套模式，行话叫 on-premise（本地部署，字面就是「装在你自家院子里」）。它有几个绕不过去的痛点：

- **贵，而且是一次性掏空。**软件还没开始用，几百万先付了。买错了、不好用，钱也回不来。
- **升级是场灾难。**软件出了新版本，你机房里那套不会自己变，得 IT 团队手动升，常常一升就出事，很多公司干脆几年都不敢动。
- **每家客户都是一座孤岛。**软件公司卖出一千份,就等于在一千个不同的机房里，装了一千个略有差异、各自为政的版本，谁也管不过来。

SaaS 换了个思路：软件不卖了，改租

SaaS 的核心动作只有一个：**把软件从「你家的服务器」搬到「我家的服务器」，然后按月收租，而不是一次性卖断。**你打开浏览器就能用，不用装、不用维护。这背后其实是三件事同时成立了，缺一不可。

第一件，多租户。这是 SaaS 技术上最关键的一步。多租户（multi-tenancy）的意思是：所有客户共用**同一套**软件、同一批服务器，只是各自的数据被隔开，互相看不见。

大白话

想象一栋公寓楼。老模式（on-premise）是每个客户各自盖一栋别墅，软件公司要跑去一千个工地分别盖、分别修。多租户是盖一栋大楼，一千户住进去，共用地基、电梯、水管，物业只维护这一栋。谁家想装修（升级），物业统一在后台改一次，全楼当天就变了。

这一下就把老模式那三个痛点全解掉了：升级只需在后台改一次，所有客户瞬间用上新版；再也没有一千个乱七八糟的版本；维护成本被一栋楼分摊，薄到可以忽略。

第二件，边际成本近乎为零。这个词听着唬人，其实很简单。边际成本就是「你每多服务一个客户，要额外掏的钱」。因为大家共用同一套软件，多来一个客户，无非是服务器上多存一点数据、多跑一点计算——几乎不要钱。

这一点是 SaaS 高利润的根子。卖冰箱，每多卖一台就得多造一台，成本压在那儿，利润薄。卖 SaaS，软件写好一次，第一千个客户和第一百万个客户用的是同一份代码，**收进来的钱几乎原封不动变成利润**。所以你会看到很多 SaaS 公司的毛利率高到 70%、80%——传统制造业想都不敢想。

第三件，订阅带来的现金流。从「卖断」改成「按月租」，看起来只是收钱节奏变了，其实是生意性质变了。

卖断是一锤子买卖：这个月签下一百万，下个月归零，你得重新去找下一个客户，永远在追着新单子跑。订阅不一样：这个月的一百万客户，下个月大概率还在，还继续付钱；你再签下新客户，是**加**在老客户上面的。收入于是像滚雪球——只要客户留得住，新单子是纯增量。

大白话

换个说法：卖断的生意每个月都从零开始跑，订阅的生意每个月都站在上个月的肩膀上往上加。前者是走平路，后者是坐电梯。

为什么「租」反而比「卖」值钱

这里有个反直觉的地方：明明卖断能一次性收一大笔，为什么华尔街反而更爱按月收小钱的租金模式？

因为投资人买的从来不是「今年赚多少」，而是**「未来能赚多少、这个数字有多稳」**。卖断模式下，明年的收入得靠明年重新去拼，谁也说不准；而订阅模式下，只要客户续费率够高，明年的收入今天就能估得八九不离十。这种「看得见的、还在长大的未来」，正是资本市场愿意给高价的东西。

所以说，SaaS 真正的发明，不是什么高深技术——多租户、浏览器都是现成的。它的发明是**一种收钱的姿势**：把一次性的买卖，变成一条源源不断、还越来越粗的现金流。

但光有一条现金流还不够。现金流能不能一直流下去，取决于客户到底走不走得掉。这就引出了下一个问题：SaaS 凭什么让客户年复一年赖着不走？答案是三道护城河——下一章我们一道一道拆开看。

第三章 · 三根护城河

上一章我们说，SaaS 的收入像一条越流越粗的现金流。但这条河能不能一直流，全看一件事：**客户到底走不走得掉。**

如果客户随时能拍拍屁股换一家，那再漂亮的订阅收入也是纸糊的。SaaS 之所以让投资人放心，是因为它在客户身上砌了三道墙，越用越高，最后高到「换一家」这件事本身变得不划算。我们一道一道看。

第一道墙：切换成本

切换成本说白了就是「你想换掉我，得付出的全部代价」——不只是钱，还有时间、风险、人的痛苦。

拿一家公司用的财务软件举例。用了三年，里面躺着几万张单据、全套的科目设置、和银行/税务系统打通的接口、以及五十个财务人员早已练成肌肉记忆的操作习惯。现在你要换一家，意味着：把三年数据搬过去（还不一定搬得干净）、接口全部重连、五十个人重新培训、而且切换那几个月账目一旦出错就是大事故。

> 大白话
>
> 换软件不像换手机，更像给一栋住满人、还在营业的大楼换地基。不是不能换，是一想到要停业、要搬家、要担风险，就宁可继续凑合用着。SaaS 公司要的就是这个「宁可凑合」。

关键在于，切换成本是**时间的朋友**——客户用得越久、塞进去的东西越多，这道墙自己就越砌越高，SaaS 公司什么都不用做。

第二道墙：数据沉淀

第二道墙和第一道有点像，但更狠一层。切换成本是「搬走很麻烦」，数据沉淀是「你压根搬不走那个最值钱的东西」。

你用一个软件越久，它就越懂你。它攒下了你所有的历史记录、你的偏好、你团队独有的流程。这些数据是你自己一天天喂进去的，它们只在这个软件里、按这个软件的方式组织着，才有意义。

更妙的是，很多 SaaS 产品越用越好用，靠的正是这些沉淀：软件根据你的历史数据给你推荐、帮你自动补全、替你预判。**你走的时候能导出一堆表格，却带不走这个「越来越懂你」的状态。**这一点，让「换一家从头开始」的心理门槛，比技术门槛还高。

第三道墙：工作流锁定

前两道墙锁的是数据，第三道墙锁的是**人的动作**，也是最结实的一道。

工作流就是「一件事从头到尾要走的一串固定动作」。当一个软件不只是你自己用，而是变成了整个团队协作的地基——销售在它里面交接客户、经理在它里面审批、财务在它里面对账、几个部门的活儿都串在它上面跑——这时候它就不再是一件「工具」，而成了公司运转的**操作系统**。

> 大白话
>
> 一个人用的工具，说换就换。可一旦几十个人的日常动作都嵌进了同一个软件，换掉它就等于让整个团队同时改掉肌肉记忆、重新对齐协作方式。这种混乱，没人愿意主动惹。Slack、Salesforce 这些产品最深的护城河，不是功能多强，而是它们成了一群人「习惯在一起干活」的那个地方。

而且这道墙常常还连着外面：你的软件如果又接了银行、又接了物流、又接了十几个别的系统，那你换掉它，等于要把这一整张网重新织一遍。用行话说，它把自己变成了一个平台——别人都围着它转，它就更动不得了。

三道墙叠起来，就成了那个估值

现在我们能回答第一章那个问题了：为什么市场肯给一家还在**亏钱**的 SaaS 公司,十倍甚至二十倍营收的估值?

因为这三道墙合在一起，等于给了投资人一个近乎确定的承诺：**今年的客户，明年基本还在；不但还在，用得越久还越走不掉、还愿意花更多钱。**于是收入不但稳，还会自己往上

长。市场买的就是这份「稳且长」的确定性——一家公司当下亏不亏钱是次要的，只要那条现金流被三道墙牢牢护住，未来的利润几乎是写好的。

这就是 SaaS 被称作「史上最好的生意」的全部秘密：不是软件多神，而是它找到了一种办法，把客户越锁越紧，把收入越滚越大。

但请记住这三道墙具体是什么——**切换成本、数据沉淀、工作流锁定**。因为等 AI 登场，真正要被重新审视的，恰恰就是这三道墙还牢不牢。这是我们后半本书的主线。不过在那之前，还得再看一眼：这台机器每天是怎么运转、怎么长大的。

第四章 · 一台增长机器的内部构造

前面三章讲的是 SaaS 为什么值钱。这一章我们钻进机器内部，看它每天到底怎么运转、怎么长大。因为接下来 AI 冲击的，不只是抽象的护城河，还有这台机器里一个个具体的齿轮——你得先知道齿轮长什么样，才看得懂哪个会被卡住。

好在这台机器的核心，用三个数就能说清楚。

第一个数：获客要花多少钱（CAC）

CAC（Customer Acquisition Cost，获客成本）就是「你多拉来一个付费客户，平均得花多少钱」。把一段时间里投在市场和销售上的钱，除以这段时间新签下的客户数，就是它。

这个数常常大得吓人。SaaS 卖给企业，往往不是客户自己上网点一下就买，而是要销售一遍遍打电话、开演示、陪着走完一场几个月的采购流程。算下来，拉一个客户花掉一两万甚至几十万，都很常见。

> 大白话
>
> 关键是：这笔钱是**今天一次性先垫出去的**，而客户的租金要一个月一个月慢慢收回来。所以一家 SaaS 公司拉的客户越多，短期反而亏得越狠——钱都先砸在获客上了。这就是为什么那么多 SaaS 公司一边高速增长、一边巨额亏损，账面吓人却没人慌。

第二个数：一个客户一辈子给你多少钱（LTV）

获客先亏钱，那这钱怎么赚回来？靠客户留得够久。

LTV（Lifetime Value，客户终身价值）就是「一个客户从进门到最终离开，前前后后总共给你交的钱」。他每月付你一千块，平均留五年，那他这辈子就值六万——这就是 LTV。

于是整台 SaaS 机器的成败，浓缩成一个简单的比较：**LTV 得比 CAC 大得多，这门生意才成立。**行业里有条粗糙的经验线：LTV 至少要是 CAC 的三倍。花一万块拉来的客户，这辈子至少得给你三万，你才算把获客那笔垫款赚回来还有得赚。

你看，两个数一摆，第三章那三道墙的意义立刻就落地了：**护城河的全部作用，就是把 LTV 撑大。**客户越走不掉、留得越久，LTV 越高，先前砸出去的 CAC 就越值。墙砌得牢，这道除法才算得过来。

第三个数：老客户自己会长大（NDR）

前两个数是 SaaS 的基本盘，但真正让投资人两眼放光的，是第三个——它解释了 SaaS 那种「躺着也在长」的魔力。

NDR（Net Dollar Retention，净收入留存率）问的是这么个问题：**去年这批老客户，今年一分新客户都不算，光他们自己，付的钱是多了还是少了？**

你可能觉得，客户总会流失一些，这个数顶多接近 100%。但好的 SaaS 公司，这个数能超过 100%，做到 120% 甚至更高。怎么做到的？因为一个客户往往越用越多：团队从 10 个人用扩到 100 个人用、从基础版升到高级版、又加买了别的功能。这些增购，盖过了那些流失掉的客户还有余。

> 大白话
>
> NDR 超过 100% 意味着一件几乎违反直觉的事：**哪怕这家公司明年一个新客户都不签，收入还会自己往上涨。**就像一个水池，即使关掉进水的龙头，水位居然还在升——因为池子里的水自己在膨胀。这是 SaaS 最迷人的地方，也是它值那么多钱的最后一块拼图。

把三个数连起来

现在整台机器就清楚了：花 CAC 把客户拉进门（先亏），靠三道墙把他留住、让 LTV 远大于 CAC（赚回来），再靠 NDR 让他自己越用越多（利滚利）。三个齿轮咬合，就是一台会自己加速的增长机器。

也正因为看懂了这台机器，我们才能看懂 AI 到底威胁的是哪个齿轮。剧透一下后面的主线：AI 可能让做软件、做营销变得极便宜，从而**压低 CAC**；但它更可能松动的是那三道墙，进而**动摇 LTV 和 NDR**——一旦客户变得容易走、容易被一个 AI 现做的东西替代，NDR 从 120% 掉到 90%，这台一直往上滚的机器，就会开始往下滑。

到这里，SaaS 的「过去」我们讲完了：一门靠订阅现金流、三道护城河、三个漂亮数字撑起来的、近乎完美的生意。下一章，主角登场——我们去看 AI 到底动了软件的哪一层。

第五章 · AI 到底动了软件的哪一层

前面四章讲的是 SaaS 的过去。现在，主角登场了。但要看懂 AI 对 SaaS 意味着什么，得先问一个更基本的问题：**AI 到底改的是软件的哪一层？**

这个问题很关键。因为如果 AI 只是给软件加了个新功能，那它顶多是又一次升级；可如果它动的是软件跟人打交道的**最底下那一层**，那事情就大了。答案是后者。要说清楚，我们先把「一个软件」拆成三层。

把软件拆成三层

- **界面层**：你看得见、点得着的部分——按钮、菜单、表格、下拉框。你通过它把你的意图告诉软件。
- **逻辑层**：软件内部的规矩——你点了「提交」，它就去校验、计算、按流程走一遍。这是写死在代码里的因果。
- **数据层**：你的东西存在的地方——客户名单、订单、聊天记录。这是软件真正替你保管的资产。

过去几十年，软件的进步几乎都发生在前两层：界面更好看了、逻辑更聪明了。但有一件事从没变过——**人必须迁就机器**。你想让软件干活，就得学会它那套按钮和菜单，把你脑子里「我想要个上季度华东区的销售报表」这句人话，亲手翻译成一连串点击：先点这个下拉框选华东、再点那个选季度、再勾几个字段、最后点生成。

> 大白话
>
> 说白了，过去用软件，是人学机器的语言。界面就是一本你必须先背熟的说明书。软件越强大，说明书越厚——这也是为什么很多复杂软件（比如企业系统）要专门培训好几天才会用。

AI 动的是这层「翻译」

大模型来了之后，变的正是这层翻译。大模型（也就是 ChatGPT 背后那种 AI）最本质的能力，是**听得懂人话，还能生成东西**。这一下，界面层被从根上改写了。

还是刚才那张报表。现在你不用再学那一串点击，直接打一句「给我上季度华东区的销售报表，按城市排」，AI 听懂，自己去调数据、生成表格。**不再是人迁就机器，而是机器迁就人了。**这个方向的翻转，是几十年来第一次。

更进一步，它连逻辑层的一部分也松动了。过去软件能干什么，是工程师提前一条条写死的——没写的功能就是没有。而大模型能**当场生成**本来不存在的东西：现写一段文案、现分析一堆杂乱数据、现草拟一份合同。软件的能力，从「工程师预先备好的一份菜单」，变成了「你点什么它现炒什么」。

这就是为什么第一章那张图让人心里发凉

现在我们能解释第一章那个让投资人睡不着的场景了。为什么一个通用 AI 随口就能干某些 SaaS 的活?

因为很多 SaaS 产品的价值，恰恰就集中在界面层和逻辑层——它们的本事，是把一件复杂的事包装成一套好用的按钮和固定流程，帮你省去自己动手的麻烦。比如一个「AI 写作助手」，本质是把「生成文字」这件事，套上一层精心设计的界面。可现在，「生成文字」这个内核，通用大模型自带了，而且更强。于是那层精心设计的界面，突然显得可有可无——**它辛苦包装的东西，如今成了 AI 的出厂配置。**

> 大白话
>
> 打个比方：过去你是村里唯一会用打字机的人，靠帮人打字为生，手艺值钱。忽然家家户户都发了一台会自动打字的机器，你那点手艺，一夜之间就不稀奇了。这不是你退步了，是整个地基被抬高了。

但别急着下结论

看到这儿你可能觉得：完了，SaaS 要凉。先别急。请回头看那三层——AI 主要改的是**界面层**、松动了一部分**逻辑层**，可它几乎没碰到**数据层**。

你公司三年攒下的客户、订单、流程，还牢牢躺在原来那个软件里。通用 AI 会说会写，可它不知道你上周三跟哪个客户谈崩了、你们公司报销要走哪三道审批。**它有一颗聪明的脑子，却没有你的记忆，也没有伸进你业务里的手。**

这就为整个「现在」定了调：AI 确实抹平了界面这层的优势，谁包装得好看不再稀奇。但真正的战斗，转移到了它还没拿下的地方——数据、流程，以及「谁能真的替你把活干完」。

接下来三章，我们就顺着这条线，看这场仗的三种打法：一场虚惊（套壳恐慌）、老玩家的防守（副驾）、和新物种的进攻（数字员工）。

第六章 · 「套壳」恐慌

2023 年，创业圈流行起一个带着嘲讽的词：套壳（wrapper，本意「包装纸」）。它专指那种产品——本事全是大模型给的，自己只在外面套了一层薄薄的界面，然后就敢出来卖钱。

嘲讽背后是一种普遍的恐慌，逻辑是这样的：既然核心能力是 OpenAI 那几家提供的，你不过是转手加个输入框，那你有什么护城河？更要命的是，模型厂商自己随时可能把你做的那点东西，变成它的一个自带功能，一夜之间把你抹平。这种恐惧有个专门的说法，叫「被大模型**顺手收编**」。

这个恐慌一半是真的，一半是错的。得拆开看。

为什么恐慌是真的：地板确实被抬高了

先说真的那一半。过去，「做出一个能生成文字/分析图片的软件」本身就是门槛，得有一支懂 AI 的团队干上几年。现在，这个能力被大模型厂商打包好、开了个接口（API，就是「一个让你的程序去调用别人能力的插座」）挂在那儿，任何一个普通程序员，接上这个插座，几天就能拼出一个像模像样的 demo。

> 大白话
>
> 换句话说，「能做出个 AI 产品」这件事的地板，被啪一下抬到了所有人脚边。你昨天引以为傲的那点技术活，今天成了插座上人人能取的电。地板一高，只靠「我会做」就站不住了——因为大家都会了。

所以那些价值真的只有一层薄壳的产品，恐慌得有道理。它们其实从来没有护城河，只是过去 AI 太难做，这层薄壳看起来才像门槛。潮水退去，才发现它没穿泳裤。

为什么恐慌又是错的：demo 不是生意

但另一半，恐慌错得也很彻底。错在把「几天能拼出一个 demo」当成了「几天能抢走一门生意」。这两件事之间，隔着一条又深又宽的河。

回想第三章那三道墙。一个真正长起来的 SaaS，它的价值从来不主要在那层界面上，而在墙里面——客户三年攒下的数据、几十个人嵌进去的工作流、和外部系统连成的一张网。**套壳能一夜复制的，恰恰是最不值钱的那层壳；一夜复制不了的，是壳背后那三道墙。**

举个例子。做一个「AI 法律助手」的 demo，接上大模型，让它读合同、挑毛病，一个周末就能搞定。但要变成律所真敢用的产品，你得跨过一堆 demo 里看不见的坎：它引用的法条得准到不能出一点错（错一条就是事故）、它得接进律所现有的案件系统、它出了错谁来担责、它得符合行业的保密规矩……**把这些一一啃下来，才是真正的活，也才是真正的墙。而这些，一个套壳周末项目连边都没摸到。**

> 大白话
>
> 一个精准的类比：大模型像发电厂，供的是电（智能）。套壳恐慌，相当于以为「既然电这么便宜、插座人人能接，那所有用电的生意——冰箱、地铁、医院——是不是都不值钱了？」当然不是。电便宜是好事，但真正的生意，是你用这份便宜的电，去解决一个具体行业里又脏又难、别人啃不动的问题。**电人人能买，把电变成谁都离不开的服务，是另一回事。**

那条真正的分界线

于是「套壳」这个词其实分不清好坏，得换个问法。判断一个 AI 产品有没有前途，不看它用没用大模型（这年头谁都在用），而看一件事：**把那层大模型的壳扒掉，底下还剩不剩东西？**

剩不下东西的，是真套壳，模型厂商一个更新就能顺手收编——它死得不冤。剩得下东西的，壳只是入口，真正的价值在壳背后那些啃了很久的硬骨头：独有的数据、难缠的行业流程、别人接不进去的系统、以及有人愿意为结果担责。这种，大模型抹不平，因为大模型自己也不想去啃那些又脏又累的活。

看清了这条线，接下来两章就顺理成章了。既然真正的墙在壳背后，那两拨人就有了两种打法：手握三道墙的**老玩家**，会把 AI 缝进自己产品里防守（下一章）；而新来的**挑战者**，不满足于套壳，索性从头去啃那些硬骨头，做全新的物种（第八章）。这场仗，才刚开打。

第七章 · 每个产品都长出一个副驾

上一章我们说，真正的墙在壳背后。手握三道墙的老玩家——那些已经装进千万家公司、攒了多年数据的存量 SaaS——自然最先反应过来。它们的打法几乎一模一样：**给自己的产品，塞进一个 AI 副驾。**

这个词是微软带火的。2023 年，微软把接进 Office 的 AI 助手叫 Copilot（字面「副驾驶」）——你还是主驾，握着方向盘，它坐在旁边帮你搭把手：帮你在 Word 里起草、在 Excel 里分析、在会议里做纪要。一时间，几乎每家 SaaS 都宣布自己长出了一个副驾。

副驾这个姿势，藏着老玩家的算盘

为什么大家不约而同选了「副驾」，而不是「自动驾驶」？这里面有精明的算计。

副驾意味着：AI 只在**你原来的软件里、你原来的流程上**帮忙，方向盘还在你手里，也还在这家 SaaS 手里。这一下就把上一章那个恐慌，巧妙地反了过来——通用大模型再聪明，它坐在一个空荡荡的聊天框里，不知道你的数据、不懂你的流程；而老玩家的副驾，是坐在**已经装满你三年数据、连好你全套流程**的驾驶舱里帮你。同样一颗聪明脑子，一个两手空空，一个手里攥着你的一切。

> 大白话
>
> 说白了，老玩家是在用第五章那个「AI 没碰到的数据层」当武器：你那颗通用大脑再强，也得先有我这具身体、我这套记忆，才使得上劲。副驾不是把 AI 请进门当客人，是把 AI 焊死在自家墙里当帮手。

所以副驾首先是一场**防守**：它对客户说，别去外面找 AI 了，你要的 AI，我在你早就习惯的地方给你备好了，还比外面那个更懂你。它把三道墙，又加高了一截。

但防守之外，还悄悄动了一样更要紧的东西：钱怎么收

副驾真正深远的影响，不在功能，在**定价**。而这一下，动的是第二章那条 SaaS 立身之本的现金流。

回想传统 SaaS 怎么收钱：按席位收费（per-seat，一个用户一个「座位」，用的人越多、买的座位越多，收的钱越多）。这套逻辑几十年来天经地义——软件是给人用的，几个人用就付几个人的钱。

可 AI 副驾把这套逻辑戳了个窟窿。问题出在这里：副驾干的活，是真金白银烧算力换来的——你让它写一份报告、跑一段分析，背后是实实在在的计算开销，用得越狠，SaaS 自己的成本越高。这和传统软件「多一个人用几乎不要钱」（还记得第二章的边际成本近零吗）完全反着来。

> 大白话
>
> 矛盾就在这儿：老办法按「多少人用」收钱，可 AI 的成本是按「用了多少」烧的。一个客户可能只有 10 个座位，却让副驾疯狂干活，烧掉的算力远超那 10 个座位的租金。**按人头收的租，盖不住按用量烧的成本。**这台一直很赚钱的机器，第一次出现了「用得越多、越可能亏」的漏洞。

于是各家开始试各种打补丁的收法：有的在原来的月租之外，单收一笔「AI 加装包」的钱；有的给副驾的使用量设个额度，超了另算；有的干脆开始琢磨，能不能别按人头、改成按干了多少活来收。**「按席位收费」这条几十年的铁律，第一次被撬动了。**

副驾的天花板：帮手终究不是干活的人

副驾是老玩家聪明的一步，但它有个天生的天花板，而且恰恰藏在「副驾」这个词里：它始终是**副**的。它把每个环节都加速了，可整件事的主体，还是那个坐在主驾、一步步操作软件的人。

它让你写邮件更快，但邮件还是你在写；它让你分析数据更快，但你还得看着它、盯着它、把结果串起来。**它优化了「人用软件」这件事，却没有取消「人得亲自用软件」这个前提。**省了力气，但那个活儿，归根结底还是你的。

那么，有没有一种更狠的打法，干脆把主驾也拿掉——你不用再一步步操作软件，只要说一句「把这事办了」，然后有个东西真的替你从头办到尾？

这正是新物种要干的事。老玩家守着墙、加个副驾，稳扎稳打；而挑战者盯上的，是一个副驾故意不碰的位置——不做帮手，做那个**替你把活干完的人**。下一章，我们看这个新物种。

第八章 · 从卖工具到卖「干完的活」

上一章结尾我们留了个问题：有没有一种打法，干脆把主驾也拿掉——你只说一句「把这事办了」，就有个东西替你从头办到尾？

这正是这两年最受关注的新物种。它有个名字：AI 智能体（AI agent，也常直接叫「智能体」或「数字员工」）。

副驾和智能体，差在哪一个字

副驾和智能体，听着像近亲，本质却差着一整个层级。差别就在一件事上：**谁在干活。**

副驾里，干活的还是你，它在旁边帮衬。而智能体，是你把一个**目标**交给它——「处理完今天所有客户的退款申请」——然后它自己拆解步骤、自己去点、自己去查、自己走完流程，最后把办好的结果交给你。你从「一步步操作的人」，退到了「派活和验收的人」。

> 大白话
>
> 一个最直白的分法：副驾是给你一把更快的**铲子**，坑还得你自己挖；智能体是你说一声「挖个坑」，它把坑挖好了喊你来看。前者卖的是更好的工具，后者卖的是**挖好的坑**——一件干完的活。

这个转变，行业里有句话概括得很精辟，叫从卖**工具**，到卖**结果**。过去几十年，所有软件（包括副驾）卖的都是工具——它让你更高效，但活儿是你干的。智能体想卖的是结果本身——它不让你更高效，它直接把活干了。

几个已经跑起来的例子

这不是空想，已经有产品在真金白银地这么卖了。

- **客服领域**：过去客服软件卖的是「一套帮你的人工客服更快回复的工具」，按客服的席位收费。现在有的产品直接卖一个能独立处理用户问题的智能体——它不辅助人工，它**就是**那个客服，一单一单地把用户问题解决掉。

- **写代码**：过去卖的是让程序员敲得更快的自动补全（典型的副驾）。现在有的产品，你给它派一个任务——「修好这个 bug」——它自己读代码、自己改、自己测，改完提交给你审。
- **专业服务**：一些面向律师、分析师的产品，开始承接过去要初级员工熬几个通宵才能做完的活——通读上千页文件、整理成一份初稿。它交付的不是一个更好用的软件，是一份**本来要花人力去做的成果**。

看出共同点了吗？它们都不再把自己摆在「工具」的货架上，而是摆在**「人力」**的货架上——它们要抢的，不是别的软件的预算，而是那笔本来要发给一个员工的工资。这就是为什么「数字员工」这个说法这么传神：它对标的，从来不是另一个软件，是一个真人的岗位。

这一步，捅到了 SaaS 最舒服的假设

为什么说这是新物种，而不是又一次升级？因为它动了 SaaS 一个几十年没被质疑过的底层假设：**软件是拿来「用」的。**

SaaS 那套漂亮的生意（第二到四章），全建立在「有个人会来用我的软件」之上——正因为有人用，才有按席位收费，才有越用越离不开的工作流。可智能体把「用软件的人」这个角色给架空了。**当活儿是智能体干的，没人再去一个个点你的界面，那「按有多少人用来收钱」还怎么收？那道靠「几十个人的操作习惯」砌起来的工作流之墙，又锁得住谁？**

> 大白话
>
> 这是整本书最关键的转折。前面 AI 抹平的是界面（第五章），松动的是定价（第七章）；到了智能体这里，它开始抽走的是 SaaS 那个最根本的地基——「人来用软件」这件事本身。地基一动，上面那套按席位收费、靠习惯锁客的老逻辑，就都得重算。

一个新物种，逼出两个新问题

智能体的出现，等于把牌桌掀了。当软件开始卖「干完的活」，两个憋了半本书的问题，再也躲不过去了：

第一，**护城河怎么办？**如果谁都能调用同样聪明的大模型来造数字员工，那老玩家那三道墙还牢吗？新的墙又该砌在哪里？（第九章）

第二，**钱到底该怎么收？**「按席位」已经被戳破，那卖「干完的活」，是按干了多少活收，还是按干成了没有收？（第十章）

「现在」这场三方混战——虚惊、防守、进攻——到这里就看完了。接下来的「未来」三章，我们就顺着这两个被逼出来的问题，一个一个往下推。

第九章 · 护城河会搬到哪里去

「现在」那场混战打完，我们攒下一个躲不过去的问题：如果谁都能调用同样聪明的大模型，那护城河还砌得起来吗？如果砌得起，砌在哪儿？

这一章先破一个最流行的误会，再说清楚墙真正会搬到哪四个地方。

先破一个误会：模型本身不是护城河

很多人第一反应是：护城河当然是模型啊，谁的模型更强谁赢。这个判断，大概率是错的。

原因有两个。其一，最强的模型就那么几家在做，而且它们你追我赶，今天你领先三个月，明天就被反超——**没有谁能长期锁死「我的脑子比你聪明」这件事**。其二，也是更要命的，模型是**买来的**。你调 OpenAI 的模型，你的对手也能调同一个。花钱就能买到的东西，从定义上就当不了护城河——护城河的意思是「别人花钱也拿不走」。

> 大白话
>
> 还是第六章那个比方：模型是电。电厂很牛，但你家附近开了个电厂，不等于你就有了生意——因为你邻居也能从同一个电厂接电。**护城河从来不在电本身，而在你怎么用电，用电干成了什么别人干不成的事。**

想通这一层，思路就打开了：既然聪明的脑子是买来的、人人有份，那真正的胜负手，全在这颗脑子**够不着**的地方。它够不着哪儿，墙就砌在哪儿。一共有四处。

第一处墙：私有数据

大模型什么都懂一点，唯独不懂**你的**那点事。它读遍了全网，却没读过你公司三年的客户记录、你工厂产线的实时数据、你那个行业里没人往网上发的内行诀窍。

这就是第一道新墙，其实也是第三章那道「数据沉淀」的升级版。过去数据是用来「锁住老客户」的；现在它多了一个更值钱的用处——**喂养一个只属于你的智能体，让它干出通用**

AI 干不出的活。谁手里攥着别人拿不到的数据，谁就能造出别人造不出的数字员工。数据从「护城河」升级成了「弹药」。

第二处墙：难啃的工作流

大模型擅长干那种「一句话就能说清、一步就能干完」的活。但真实世界里值钱的活，往往是又长又绕、还全是例外的。

报销一笔款，听着简单，背后可能是：金额到多少要谁审、跨部门怎么走、碰上特殊情况怎么破例、还得跟财务系统和银行对上账……**把这一整条又脏又长、全是坑的流程，稳稳当当地自动跑通，是极重的活**——而这活越重、坑越多，越是护城河。因为通用大模型只想给你一颗聪明脑子，它没耐心、也没兴趣，去啃某个行业里这条具体而琐碎的烂泥路。谁把这条路修通了、修得别人不敢碰，谁就立了一道墙。

第三处墙：分发与信任

这一处最容易被技术出身的人忽略，却往往最硬。

分发，就是「你的东西怎么送到客户面前」。老玩家最大的本钱，是它的软件已经装进了千万家公司——这本身就是一条别人烧钱也难买的路。第七章那些副驾之所以有底气，正因为它们不用重新找客户，只要给**已经在用它的人**顺手递上一个 AI，就赢在了起跑线。

比分发更硬的是**信任**，尤其在「卖结果」的时代。当智能体替你把活干完，一个要命的问题冒出来了：**它干错了，谁负责？**它把一笔款打错了、把一份合同审漏了，客户告谁去？敢站出来说「出了事我兜着」，是一道极高的墙——它拦住的不是技术，是责任。这也解释了为什么在律师、医疗、金融这些一错就是大事的领域，一个虽笨但可靠、还敢担责的老牌子，可能比一个更聪明但没人敢信的新玩意，更卖得动。

大白话

在卖结果的世界里，客户买的其实是**安心**：出了事有人兜着。这份安心，聪明的大模型给不了——因为你没法冲一个通用模型问责。谁能站出来对结果负责，谁就握着一道大模型永远够不着的墙。

第四处墙：被记住的入口

最后一道墙，藏在一个新战场里：当越来越多的活由智能体来干，那个**你习惯去开口吩咐的入口**，本身就成了兵家必争之地。

就像今天你想搜东西会条件反射地打开某个搜索框——将来你想「让 AI 办件事」，会条件反射地打开谁？谁成了你的默认入口，谁就守住了最靠前的位置。这道墙砌的是**习惯**：你懒得换、想不起别家，就是它最深的护城河。

把四道墙收拢

于是问题的答案清楚了：模型不是护城河，因为它是买来的；真正的墙，砌在模型**够不着**的四个地方——**你独有的数据、你啃通的烂泥路流程、你占住的分发与敢担的责任、你占据的那个入口**。

你会发现，这四道墙没一道是新发明——它们全是第三章老护城河，在「卖结果」时代的升级版。**护城河没有消失，它只是从「锁住用软件的人」，搬到了「谁能真的、可靠地、担着责地把活干完」。**谁守住这四道墙，谁就在 AI 时代重新站稳了。

墙的问题解决了，还剩最后一个更现实的问题：这活干完了，钱到底该怎么收？下一章，我们看那场正在发生的定价地震。

第十章 · 定价的地震

墙的问题上一章解决了。还剩一个最现实、也最见血的问题：活干完了，钱到底怎么收？

这事听起来像财务细节，其实是整门生意的命门。因为一门生意怎么收钱，决定了它长什么样、能长多大。而 SaaS 这几十年赖以为生的那套收钱方式，正在 AI 手里裂开一道大缝。

先看看那套要塌的老办法

回想第七章那个词：按席位收费——一个人用，就付一份钱，用的人越多，收的钱越多。这套逻辑几十年来天经地义，因为它背后有个朴素的假设：**软件的价值，是靠「人来用」兑现的。**用的人多，说明它有用，多收钱理所当然。

但第八章那个新物种，智能体，把这个假设连根拔了。当活是智能体干的，「有多少人用」这个数，突然就没意义了。

> 大白话
>
> 设想一家公司上了个客服智能体，把原来 50 个客服干的活全包了。按老办法，它该付几个席位的钱？50 个？可现在一个人工客服都没有了。0 个？可这智能体明明替它干了 50 个人的活、省了 50 份工资。**「按人头收费」这把尺子，突然量了个寂寞**——要量的东西还在，尺子却对不上了。

更要命的是第七章说过的成本反转：智能体真在烧算力，用得越狠、SaaS 成本越高。老办法按「几个人用」收固定的租，却挡不住按「用了多少」猛涨的成本。旧尺子不但量不准价值，还兜不住成本——**两头都塌，它非换不可。**

三把新尺子，越量越贴近「价值」

于是行业开始换尺子。方向很清楚：从量「有多少人用」，一步步挪向量「真正创造了多少价值」。眼下有三把新尺子，可以看成三级台阶。

第一把：按用量收费。不管你几个人，用多少算多少——发了多少条消息、处理了多少份文件、烧了多少算力，就收多少钱。它最直接的好处，是把成本兜住了：客户用得多，你收得也多，不会再出现「用得越狠、你亏得越惨」。这是最稳的一步，很多公司先退到这里保命。

第二把：按结果收费。这一步更进一层，也更刺激：不看你用了多少，只看你**办成了多少**。客服智能体，成功解决一个用户问题，收一次钱，没解决不收；招聘智能体，帮你约成一场面试，收一次钱。它对标的，不再是软件，而是第八章说的——**那笔本来要发出去的工资**。这也是「卖结果」在账单上的样子。

> 大白话
>
> 为什么按结果收费这么诱人？因为它把买卖双方绑到了一条船上。老办法里，软件卖出去就完事，你用得好不好是你的事；按结果收费，你不成功我就不赚钱，于是我拼了命也要让你的活真的办成。**客户买的从此不是一个工具，而是一个结果——办不成，我不收你钱。**这是几十年来商业模式头一回，让软件公司和客户真正利益一致。

第三把（还在远处）：按省下的钱分成。最激进的设想是：智能体替你省下了 100 万人力成本，我从里面抽一份。这把尺子最贴近「价值」，却也最难落地——省了多少、算不算你的功劳，双方能吵到天亮。所以它现在更多是个方向，而非通行做法。

换尺子，是一场伤筋动骨的手术

别以为换个收钱方式只是改改价目表。它牵动的是第二到四章那整台机器，处处硌得慌。

最疼的一处：按席位收费，收入是**稳的**——一家公司买了 100 个座位，明年大概还是 100 个，收入平得像铁轨，这正是第二章说的、投资人最爱的那份确定性。可一旦改成按用量、按结果，收入就随客户这个月干多干少上下颠簸。**SaaS 用几十年、用高估值换来的那份「稳」，恰恰是新尺子最先弄丢的东西。**这也是为什么很多老玩家嘴上喊着拥抱 AI，手却死死攥着老价目表——不是不懂，是那份「稳」太值钱，舍不得撒手。

一句话收束这场地震

所以这场定价地震，说到底是一次**丈量方式的迁移**：从量「有多少人在用软件」，挪向量「软件到底创造了多少价值」。

它和第九章那四道墙，其实是同一件事的两面：护城河从「锁住用的人」搬到「可靠地把活干完」，那么账单自然也得从「按人头」改成「按干成的活」。**墙怎么砌，钱就怎么收，两者必须对齐。**

墙和钱都理清了。最后一个问题，也是你最关心的：在这场大迁移里，具体哪些 SaaS 会死、哪些会活得更好、哪些是全新的机会？下一章，我们上一套能自己动手判断的框架。

第十一章 · 谁会被吞，谁会被抬起来

前面两章把墙和钱都理清了。现在到了你最关心的问题：具体到一家一家的 SaaS，谁会死，谁会活得更好，谁是从天而降的新机会？

这一章不给结论清单——清单会过时。我们给一套你能自己动手用的判断框架。核心是一个问题，问完就能大致归类。

那个决定生死的问题

面对任何一个 SaaS，只要问一句：**它的价值，主要压在「界面」上，还是压在「墙」里？**

回想第五章那三层。如果一个软件的本事，主要是把某件事包装成一套好看好用的界面（界面层），那它危险——因为 AI 最擅长抹平的就是这层。反过来，如果它的价值压在墙里——独有的数据、啃通的烂泥路流程、别人接不进的系统、敢担的责任（第九章那四道墙）——那它就稳，因为这些正是 AI 够不着的。

拿这把尺子一量，市面上的 SaaS 大致落进三格。

第一格：会被一口吃掉的

最危险的，是那种**价值几乎全在界面层、墙又薄的**软件。特征很好认：

- 它干的事，本质就是「生成点内容」或「简单加工一下信息」——写文案、做个简单图、套模板出文档。这些正是通用大模型的出厂本事。
- 它没沉淀多少独有数据，换一家几乎没成本，也没什么非它不可的流程。

这类产品，其实就是第六章说的**真·套壳**——扒掉大模型那层壳，底下不剩什么。它们会被两头夹死：要么被通用大模型一个新功能顺手收编，要么被用户发现「我直接问 AI 就行，何必再订阅你」。**它们不是被某个对手打败，是被整个地板抬高给淹没的。**

第二格：反而被抬起来的

这一格最反直觉，也最容易被唱衰 SaaS 的人忽略：**有一大批 SaaS，不但不会死，还会被 AI 抬得更高。**

哪些？就是那些价值本来就压在墙里的——它握着别人拿不到的数据，或者早已是某个行业运转的操作系统（第三章的工作流锁定）。对它们来说，AI 不是敌人，是一件**捡到的神兵**：

- 它有独家数据，正好拿来喂一个别人造不出的智能体（第九章的「数据即弹药」）。
- 它早占住了分发和信任——客户本就在用它、也信它，它顺手递上一个 AI，比谁都名正言顺（第七章的副驾）。

> 大白话
>
> 说白了，AI 对这类公司是**杠杆**不是威胁：它把自己攒了多年、别人偷不走的家底（数据、流程、信任），用 AI 放大成新的产品和新的收费。墙越厚的老玩家，这根杠杆越使得上劲。所谓「AI 颠覆 SaaS」，颠覆的从来不是它们——它们是这场变化里悄悄的赢家。

第三格：凭空长出来的新物种

第三格不在存量里，而是 AI **凭空劈出来的新地盘**——那些过去因为「太贵、太难、不值得」而根本没被软件化的活。

逻辑是这样：过去要把一件事做成软件，得先请工程师把它的每一步都想清楚、写死成代码。有些活太灵活、太依赖当场判断（比如通读一堆杂乱资料写份摘要、处理各式各样的客户来信），根本没法提前写死，于是几十年来只能靠人肉干，软件插不上手。

而智能体（第八章）恰恰擅长这种「没法提前写死、要当场随机应变」的活。于是一大片过去软件够不着的领域，头一回能被自动化了。**这里诞生的不是「更好的软件」，而是过去只能是「人力」、如今头一回变成「软件」的全新生意。**这也是这场变革里最大的一块蛋糕——它不从别人碗里抢，是凭空多出来的一桌菜。

把框架收成一句话

所以不必笼统地问「AI 会不会颠覆 SaaS」，那个问法太粗。换成一句更利落的：

> 价值全压在界面、墙又薄的，会被吃掉；价值压在墙里、家底厚的，会被 AI 抬得更高；而过去贵到、难到没法做成软件的活，如今头一回能做了——那是最大的新机会。

你看，同一场 AI 浪潮，对三格里的公司，是三种截然不同的命运。**决定命运的从来不是「用没用 AI」，而是「你的价值到底压在界面，还是压在墙里」。**

框架有了，三格也分清了。最后一章，我们回到第一章那个让投资人睡不着的问题，给它一个干脆的答案：SaaS 到底死没死？

第十二章 · SaaS 没死，是那个「S」变了

兜了一大圈，我们回到第一章那个让投资人睡不着的问题：**如果一个通用 AI 随口就能干你软件干的活，那 SaaS 还值钱吗？它会死吗？**

现在可以给个干脆的答案了：**SaaS 不会死。但那个我们熟悉的 SaaS，正在变成另一样东西。**变的关键，恰好藏在它名字里那个字母的位置上。

那个精妙的文字游戏

SaaS 的全称是 Software as a Service——**软件**即**服务**。琢磨这个词序：主角是「软件」，「服务」是用来形容它的——本质是个软件，只不过用「服务」（按月租、云端跑）的方式交付给你。你买的核心，始终是那个软件；服务，只是它的包装。

而这两年，硅谷开始流传一个把这两个词**调了个个儿**的新说法：Service as a Software——**服务**即**软件**。别小看这一调。现在主角成了「服务」，「软件」反倒成了形容词——本质是一项服务（一件替你干完的活），只不过这项服务是用软件（智能体）来提供的。

> 大白话
>
> 两个词一模一样，只是换了个位置，卖的东西却彻底变了：
> · **Software as a Service**：我卖你一个**工具**，你自己拿去用。（过去，第二到四章）
> · **Service as a Software**：我用工具替你**把活干完**，交给你结果。（未来，第八章往后）
> 从「卖工具」到「卖干完的活」，整本书讲的所有变化，全浓缩在这一次词序对调里。

这就是为什么「会不会死」是个问错的问题

还记得第一章那句话吗——「AI 会不会杀死 SaaS」本身就问错了。现在你该明白它错在哪了：这个问法，暗含了一个假设，以为 SaaS 是个固定不变的东西，只能整个活着或整个死掉。

可 SaaS 从来不是一样死物。回头看它的来路：它当年就是从「卖断软件」变来的（第二章）——那次，「软件」没死，只是交付方式从卖光盘变成了按月租。而这一次，是又一次

同样量级的蜕变：**软件还是那个软件，只是它从「一个你得亲自用的工具」，变成了「一个替你把活干完的服务」。**

所以真相是：会死的，是那些死抱着「我只卖工具、用不用是你的事」的公司——它们的价值压在最容易被 AI 抹平的界面上（第十一章第一格）。而会活、甚至活得更好的，是那些跟着往前迈一步、开始交付「干完的活」的公司。**不是 SaaS 这个物种要灭绝，是它又一次要蜕皮了。**

把整本书收成一条线

让我们最后把这条线捋一遍，你会发现十二章其实在讲同一件事：

- **过去（2–4 章）**：SaaS 靠订阅现金流、三道护城河、三个漂亮数字，成了史上最好的生意——一门「卖工具、按人头收租」的生意。
- **现在（5–8 章）**：AI 抹平了界面（第五章），惊醒了套壳（第六章），逼出了老玩家的副驾（第七章）和新物种的智能体（第八章）——「人来用软件」这个地基，开始松动。
- **未来（9–11 章）**：护城河从「锁住用的人」搬到「可靠地把活干完」（第九章），收钱从「按人头」改成「按干成的活」（第十章），于是有人被吃掉、有人被抬起、有人凭空诞生（第十一章）。

一整套变化，最后全落回那次词序对调：**从 Software as a Service，到 Service as a Software。**

回到那个更本质的问题

第一章我说过，这本书真正想回答的，其实是另一个更根本的问题：**当做软件本身变得几乎免费，一家软件公司还剩下什么值钱的东西？**

走完这一圈，答案已经浮出来了。剩下的，恰恰是软件之外的那些东西——你独有的数据、你啃通的烂泥路、你占住的分发、你敢担的责任、你替客户真正办成的那件事。**软件本身越来越不值钱，「用软件替人可靠地把活干完」这件事，反而越来越值钱。**

这大概就是 AI 时代给 SaaS 的最终判词：写代码这件事，AI 让它越来越廉价；可把这些代码，变成一个客户离不开、还敢托付、愿意为结果付钱的**服务**——这件事，从来没像今天这样值钱过。

睡不着的投资人可以睡了。SaaS 没有死。它只是又一次，长成了它下一个该有的样子。