供养：一个东西为何活不过上线日

——把商业里"交付完成"的思维，换成"能不能被持续供养"

只讲方法、技术与操作的实践文 · 约 8 千字 · 延伸自《供养》 · 德麦国际 Demai International · sdeuniverses.com

有一类死亡，在商业里几乎不被当作死亡来处理，因为它长得太像成功。产品在发布会那天达到它一生的顶点，媒体稿、下载曲线、内部庆功都停在那一刻，之后是一条谁也不愿意盯着看的下滑线。一套方法论被写成漂亮的手册、配上培训和海报，逻辑严密、案例齐全，没有一处可以被驳倒，可是三个月后没有一个团队还在用它做事。一项被投入重金建起来的组织能力——一个数据中台、一个用户研究体系、一个内部平台——上线验收那天所有指标达标，然后无人问津，慢慢荒废成一堆没人敢删又没人再碰的资产。

这些东西的共同点是：它们都足够正确，足够优秀，甚至一度足够被认可。可它们还是死了。通行的解释是执行不到位、市场没起来、时机不对，这些解释都不算错，但它们绕开了同一件更朴素的事实——**没有人再把自己的生命投进去用它了**。没有人带着自己真正关心的问题走进来，用自己的手去推它、改它、在用的过程中重新理解它、给它注入新的东西。它不是因为不够好而死，它是因为断了供养而死。

这里有一个被商业世界普遍误信的假设：好东西会自己被看见，被看见就会被采用，被采用就会活下去。这条链的每一环都靠不住。市场上从不缺被看见却无人采用的好产品，也从不缺被采用一次却再没有第二次的好方法。看见和采用都只是瞬间的事，而活下去是一件需要日复一日供给的事。把"被识别"当成存活的机制，是无数团队把资源砸错方向的总源头。

母论文《供养》讲的就是这件事：一个东西能不能活下来，不靠它自带的"正确/有用/优秀"被识别出来，而靠它被持续地供养——有人愿意把生命投进去，用它去做自己最关心的事，在用的过程中不断重新理解它、改造它、为它注入新生命。一旦没人再把问题带进来、没人再用自己的手推它，它就死了，无论它多正确。任何东西都**不能自存**：它不会因为自身的优秀而自动继续存在。这篇文章要做的，是把这个道理落到商业现场，给创始人、产品负责人、组织负责人一套可操作的手法——把注意力从"交付完成了没有、做得对不对"转到"它还能不能被继续供养"，并且学会主动为产品和能力**留生长缝**、主动**培育供养者**。

一、先看清：商业里"缺供养"长什么样

缺供养不是一种情绪，它有稳定的外形。学会认它，是所有后续动作的前提。它的核心特征是：这个东西看上去还在、还没被正式关停，但已经没有人再往里投入生命了。它靠惯性、靠预算、靠"当初立过项"活着，而不是靠有人真的在用它解决自己的问题。下面三个场景，是商业里最常见的三副面孔。

场景一：产品上线即巅峰，然后一路向下

一个产品在发布那天是它最热闹的时刻。之后每一天的活跃、留存、复购都在提醒你：热闹是一次性的。团队的第一反应通常是加功能、做活动、投流量——用外部的推力去补内部的失血。但如果你去看那些少数还在用、而且越用越深的用户，会发现他们和大多数人不一样：他们把这个产品接进了自己的工作流，用它做一件他们本来就非做不可的事，甚至发明了你没设计过的用法。**他们在供养这个产品**。而大多数用户从来没有把自己的问题真正交给它，产品对他们只是一次尝鲜，尝完就走。

上线即巅峰，几乎总是因为产品在设计时就把"被打动、被下载"当成了终点，而没有为"被持续地用来做自己的事"留出空间。它是一件成品，一件精美的、封闭的、不邀请你参与的成品。成品不会自己活下去，因为它没有留下任何可以被继续供养的接口——用户用完一遍，就无事可做了。

场景二：方法论完全正确，却没有一个人用

很多公司都有一份放在共享盘里的方法论：一套需求管理流程、一套复盘框架、一套决策模型。它往往是某个能干的人或某支咨询团队认真产出的，逻辑自洽，无可指摘。可你去问一线：上个季度你用它做过一次决策吗？答案常常是没有。它正确，但它是死的。

方法论之所以没人用，很少是因为它错，而是因为它被做成了一份"已完成的正确答案"，而不是一件可以被每个人拿去改造的工具。它不欢迎你往里塞自己的场景，不欢迎你删掉不适用的步骤，不欢迎你按自己团队的语言重写它。一份不能被使用者改造的方法论，等于要求所有人放下自己手头真正的问题，来伺候它的完整性。没有人会这么做。**不能被改造的东西，就不能被供养；不能被供养的东西，无论多对，都会死。**

场景三：能力建好即荒废，成为资产负债表上的僵尸

第三副面孔最贵。公司立项建一项能力——一个数据平台、一个中台、一个用户洞察体系、一个内部工具链。项目按里程碑推进，验收那天所有指标达标，团队解散，庆功结束。然后它开始荒废：数据没人补，看板没人看，工具没人维护，半年后它变成一个没人敢删、也没人再碰的僵尸资产。

问题出在，这项能力从立项第一天起，目标就被设成了"建成并验收"，而不是"被业务持续地拿去做自己的事"。建设团队交付的是一个完工状态，而完工状态不含任何"谁在日复一日地用它、靠它、改它"的机制。一旦交付团队撤走，就没有供养者了——没有人把自己的业务问题带进来，逼着这个平

台长出新的字段、新的报表、新的接口。能力和产品一样不能自存：**它不会因为"建得好"而自动继续有用，它只会因为"一直被用来做要紧的事"而继续活着。**

二、分诊：区分"正在被供养的东西"与"优秀但没人喂的死物"

看清了三副面孔，接下来要做的是分诊。商业里最浪费资源的错误之一，是把一个"优秀但没人喂"的死物，当成"暂时遇到困难的好东西"继续加投入——继续加功能、加预算、加人力，去救一个其实已经断了供养的东西。分诊的目的，是在投钱、投人、投时间之前，先判断你面对的到底是活的还是死的。判断不看它表面多完整、多正确，只看一件事：**有没有人在持续地把自己的生命投进去用它。**

逐条这样问，问的都是"供养"是否存在，而不是"质量"是否达标：

问一：过去一个月，有没有人不是被要求、而是主动地用它做自己真正关心的事？被考核、被要求填的使用不算供养，那是应付。要找的是那种"不用它反而更麻烦"的自发使用。如果所有使用都要靠推动和考核维持，它已经是死物。

问二：有没有人在用的过程中改造过它、抱怨过它、要求它变得更好？活的东西会招来意见、招来改动请求、招来"你这里能不能加一个"。一个从来没人提要求的产品或能力，不是完美，是没人真在用。**沉默不是满意，沉默是死亡的前兆。**

问三：如果今天把维护它的那个人或那支团队撤掉，一周内会不会有人叫？会有人立刻感到疼、立刻来找你，说明它嵌进了别人的生命里，有真实供养。没人叫、悄无声息，说明它早就是可有可无的摆设。

问四：它的价值是停在发布/验收那一刻，还是随着被使用在往上长？被供养的东西会积累——数据越用越厚、用法越用越多、社群越用越熟。价值停在某一刻不再增长的，是成品，成品在等待折旧。

问五：使用者能不能把自己的东西塞进来？能塞进自己的数据、自己的流程、自己的插件、自己的内容，它就有**生长缝**——一个允许别人注入生命的接口。全封闭、只能照既定方式用一遍的，没有生长缝，供养无处附着。

五问里只要有三问的答案偏向"没有人、没有改动、没人会叫、不再增长、塞不进来"，就应当停止把它当活物抢救，转而做一个冷静的决定：要么为它重新造出供养（下面几章讲怎么造），要么体面地让它退场，把生命投给真正有人喂的东西。**分诊的价值，一半在于救对，一半在于不救错。**

三、供养是怎么被切断的

要培育供养，先得懂它是怎么被切断的。断供养很少是某个人的恶意，通常是一整套看起来天经地义的管理习惯，安静地把生命的通道一根根掐掉。有三个子机制最常见，它们往往同时发作。

机制一：只认"交付完成/正确"，不认"持续被使用"

绝大多数商业组织的奖惩、汇报、验收，都建立在"完成"和"正确"这两个词上。产品上线了就算成功，方法论写完了就算产出，平台验收了就算交付。这套语言里没有"之后有没有人一直在用"的位置。于是所有人的努力都涌向那个"完成"的时刻，而完成之后的漫长供养期，无人负责、无人计分、无人在意。

这是最根本的一刀。它把一个东西的生命，压缩成了它被交付的那一个瞬间；瞬间一过，供养就在制度上"下班"了。母论文说得直接：**把"做对了、交付了"当成终点，恰恰是断供养的起点**，因为一个东西真正的存活，全部发生在交付之后那段没人计分的时间里。

机制二：惩罚"未完成"，把生长缝一条条封死

第二刀更隐蔽。组织普遍把"未完成、还在变、还不定型"当成缺点来惩罚：一个还在演化的产品会被说"不稳定、没想清楚"；一套还留着口子让人改的方法论会被说"不严谨、不成体系"；一个故意留着待填空间的平台会被说"没做完就上线"。于是负责人学会了一件事——赶紧把东西做成封闭的、定型的、"完成"的样子，以免被追问。

可是母论文里的**生长缝**，恰恰是那些"看上去未完成"的接口：一个可以被继续填、被继续改、被继续接的口子。它天生就带着"还没做完"的样子，因为它就是留给未来供养者的入口。当一个组织习惯性地惩罚"未完成"，它就是在系统性地封死每一条生长缝——把所有能被继续供养的接口都抹平、封上、定型。结果是一件件精美的、完成的、再也长不出任何东西的死物。**惩罚未完成，就是惩罚生命本身。**

机制三：没有人把自己的问题带进来改造它

第三刀是前两刀的自然结果，也是死亡真正落地的地方。当一个东西被当成"完成的正确答案"交付、又不留任何可被改造的缝，使用者面对它的姿态就只剩一种：接受或不接受。他们无法把自己的问题带进来——因为它不欢迎问题；无法用自己的手推它——因为它不允许改。于是使用者和这个东西之间，永远建立不起那种"我靠它做我的事、我也在把它变成我要的样子"的活关系。

没有这种活关系，就没有供养。产品成了别人做的、与我无关的成品；方法论成了别人写的、我照抄或不抄的文档；平台成了别人建的、我用或不用的工具。它们始终是"他者"，从未进入任何人的生命。母论文反复强调：**东西是靠使用者把自己的问题和双手投进去才活着的。**当没有人这样投入，再正确的东西也只是一具等待落灰的陈列品。

四、核心重构：培育供养的四个杠杆

诊断到此为止，剩下的全是操作。培育供养不是一句口号，它可以拆成四个可以动手的杠杆，每个杠杆都配着具体技术。四个杠杆分别对准前面三刀：留出接口、请人进来、把人聚起来、给"未完成"正名。

杠杆一：留生长缝，造可扩展的接口

第一件事，是在产品和能力里刻意留下**生长缝**——让别人能把自己的东西塞进来的口子。一件全封闭、只能按你设计的方式用一遍的东西，是没有供养入口的。落地技术：其一，**把关键部分做成可配置、可扩展、可插拔**，而不是写死——让用户能接入自己的数据、自定义自己的流程、挂上自己的插件或模板。其二，**公开接口而不是封闭黑箱**，无论是给外部的 API，还是给内部业务方的字段扩展、报表自助、规则自定义；开放的接口就是被继续供养的通道。其三，**刻意留白**：方法论里留出"此处按你的场景填写"的空位，平台里留出待接的模块，产品里留出用户可自建的空间。留白不是没做完，是给未来的供养者预留座位。

留缝要趁早。生长缝不是等东西做完了再补上去的补丁，而是从设计第一天起就长在骨架里的结构。一件先被做成封闭件、事后再想开放的东西，往往开放不动——它的每一处都为"独自完成"而优化，没有给别人预留任何位置。所以在动手之前就要问：这里将来要让谁把什么塞进来？把这个问题的答案变成结构，缝才留得住。

判断生长缝留得好不好，只问一句：**一个带着自己问题来的人，能不能不经过你就把自己的东西接进去、把它改成更适合自己的样子？**能，就有缝；不能，你交付的只是一件精美的死物。

杠杆二：让用户和员工用它做自己最关心的事

光有缝还不够，得真有人带着生命走进来。供养的本质，是有人用这个东西去做他自己本来就非做不可的事。所以第二个杠杆是：**把产品或能力，对准使用者真正关心的问题，而不是你希望他关心的问题。**

落地技术：其一，**找到"非它不可"的场景**——使用者有一件要紧事，用了它会明显更省力，不用会明显更麻烦。只有嵌进这种事，使用才会自发延续，而不是靠考核硬撑。其二，**降低把自己的问题带进来的门槛**：让用户能用最少的步骤把自己的真实数据、真实任务放进去，越快尝到"它在帮我做我的事"，越可能留下来供养。其三，**顺着自发用法长，而不是纠正它**：当你发现有人用出了你没设计的用法，那是最珍贵的供养信号——顺着它加强、把它变成正式能力，而不是判它"不合规范"。其四，对内部能力，**把它交给一个真有此痛的业务方去主用**，让能力跟着这个业务方的真实问题一起长，而不是让建设团队自娱自乐地维护指标。

杠杆三：培育供养者社群，让供养互相传染

单个供养者会流失，一群互相看得见的供养者却能自我维持。第三个杠杆，是把分散的使用者变成一个**供养者社群**——让用它、改它、为它注入新东西的人彼此看见、彼此激励、彼此传帮带。

落地技术：其一，**让供养可见**：把用户创造的用法、模板、插件、案例摆出来，让别人看到"原来还能这样用"，被看见的供养会吸引新的供养。其二，**给重度供养者身份和权力**：让最投入的用户或员工成为共建者、版主、模板作者、内部布道者，让他们对这个东西有话语权、有署名、有归属——人会为自己参与塑造过的东西持续投入。其三，**让改造和贡献能沉淀、能被下一个人接着用**：一个人的改造若能变成所有人的起点，供养就会累积而不是流失。其四，**创始人或负责人自己下场供养**：亲自用、亲自在社群里回应、亲自把用户的改造接进主干——负责人对待它的姿态，决定了别人敢不敢把生命投进去。

杠杆四：把"持续演化、暂未完成"正当化

第四个杠杆针对第二刀——它要把组织对"未完成"的惩罚，改成对"还在被供养、还在生长"的尊重。这是四个杠杆里最像文化、其实最能靠制度动手的一个。

落地技术：其一，**改口径**：在汇报和评审里，把"是否完成/是否正确"换成"是否有人在持续用、是否在长出新东西"。当评价标准变了，人们才敢留缝。其二，**为供养期设立责任人和资源**：产品上线、能力验收之后，明确谁负责它接下来的持续被使用，并给这份工作正式的时间和计分——供养不能是无人认领的义务劳动。其三，**公开地把"还在演化"当成优点来讲**：对用户、对内部都说清"它会一直长、你们的使用会塑造它的下一步"，把"未完成"从需要遮掩的缺点，变成邀请参与的理由。其四，**停止用"定型即优秀"奖励人**：不再只表彰那个把东西做完、封盘、交付的人，而是同样表彰那个让一个东西被越来越多人用起来、活得越来越旺的人。**你奖励定型，就会收获死物；你奖励供养，才会收获活物**。

五、产品／组织能力／社区／知识管理工具箱

四个杠杆是通用的，落到不同场景各有其手法和话术。这一章按场景给具体做法，可直接照搬。

产品

做法：把"发布"从终点改成起点，在产品里内建生长缝——用户可导入自己的数据、自建自己的模板、接入自己惯用的工具；开放一批扩展接口，哪怕最初只有少数人用。上线后的核心指标不看下载和首日活跃，看**"带着自己的问题深度使用并回来改造"的用户**有多少、在不在增长。设专人盯供养期，把用户自发用法快速转成正式能力。

话术（对内）："我们不是要做一个上线就完美的产品，而是要做一个能被用户一直用下去、一直改下去的产品。发布只是第一次邀请。"话术（对用户）："这个产品会跟着你的用法一起长，你怎么用它、要它变成什么样，都算数。"

组织能力

做法：任何中台、平台、体系类建设，立项时就为它绑定一个**真有此痛的业务主用方**，把"验收达标"降级、把"业务方持续靠它做事"升级为真正的成功标准。建设团队不在验收后立即解散，留一支小队伴随供养期，专门把业务方新带来的问题变成能力的新字段、新接口。给能力留足生长缝，让业务方能自助扩展，而不是每次都排队等建设团队。

话术："这个平台建成不算成功，有业务天天靠它打仗、并且逼着它长出新东西，才算成功。我们不是交付一个完工的系统，是交付一个会一直被喂养、一直长的系统。"

社区

做法：社区最怕"拉起来就冷"，冷的根源是只有运营在投入、成员只是围观。手法是把成员从围观者转成供养者——给他们能带进自己问题、能被别人看见、能获得身份的位置。让老成员带新成员，让成员的贡献沉淀成社区的资产，让最投入的人成为有话语权的共建者。运营者自己持续下场，而不是发完公告就走。判断社区死活不看人数，看**有多少成员在为社区注入新东西、有多少人会因为它停摆而立刻来问**。

话术："这里不是我们办给你看的，是你和大家一起长出来的。你带进来的问题、你贡献的东西，就是这个社区的生命。"

知识管理与方法论

做法：这是"正确却没人用"的重灾区。把方法论从"一份完成的正确答案"改造成"一件人人可改的活工具"：明确写上"此处按你的场景改写/删减/替换"，鼓励每个团队产出自己的版本并互相看见，把最好的改造版反向合并回主干。不再追求一份放之四海皆准的完美文档，而是维护一个持续被使用者改造、持续长出新用法的活体系。判断方法论死活只问一句：**上个季度有没有人真的用它做过一次要紧的决定，并在用的过程中改动过它。**

话术："这套方法不是拿来供着的标准答案，是拿来用的、拿来改的。你按自己团队改出来的版本，比原版更值钱。"

关于 AI 与基底

在这些场景里越来越多会用到 AI。原则不变：把 AI 或**基底**当成一个需要被持续供养的接口来对待，而不是一个买来就一劳永逸的成品。给它留生长缝——让业务方能把自己的场景、自己的数据、

自己的判断持续喂进去，让它跟着真实使用长；如果 AI 能力被当成"接入即完成"的封闭件，它一样会断供养、一样会荒废。它和产品、能力遵循的是同一条法则：不能自存，只能被养。

六、落地流程

把上面的东西串成一条可以走的路，分五步，任何产品、能力、社区、方法论都可套用。

第一步，分诊。对手上每一个重要的东西，跑一遍第二章的五问，判断它是"正在被供养"还是"优秀但没人喂"。诚实标注，别把死物标成活物。这一步决定你把生命投给谁。

第二步，找供养者与场景。对决定要救或要养的东西，找出谁真会带着自己关心的问题来用它、用它做哪件非做不可的事。找不到这样的人和事，就先别急着加功能——先解决"有没有人愿意投入生命"这个更根本的问题。

第三步，留缝与降门槛。照杠杆一、杠杆二动手：把关键部分改成可扩展、可插拔、可自定义，公开必要接口，刻意留白；同时把"带自己的问题进来"的门槛降到最低，让人尽快尝到"它在帮我做我的事"。

第四步，聚人与正名。照杠杆三、杠杆四动手：把分散的供养者连成社群，让供养可见、给共建者身份；同时改评价口径、设供养期责任人、公开地把"还在演化"讲成优点。文化和制度这两件一起做，缺一不可。

第五步，按供养复盘，而非按完成复盘。定期回看的不是"我们做完了多少"，而是"有多少东西还在被真的用、在长；有多少悄悄断了供养、该被体面退场"。把生命从死物那里撤回来，投到活物上去。复盘的问法永远是母论文那句：**它曾被怎样供养、现在还有没有人喂、还为未来的供养留了什么接口。**

七、必须避开的误区

方法用错方向，比不用更耗生命。四条最常见的误区，每一条都值得贴在墙上。

误区一：以为"做得够好"就能自己活下去

这是最深、最贵的误区，也是母论文最想拔掉的那根刺。无数团队相信只要把东西做到足够好、足够正确、足够优秀，它就会被识别、被采纳、自动活下去。于是他们把全部资源砸在"做得更好"上，却一分钟都没花在"谁来持续供养"上。结果是一件件无可挑剔却无人问津的死物。请记住：**没有任何东西能靠自身的优秀自存。**优秀不产生生命，供养才产生生命。做得好只是让它值得被养，不等于它会被养。

误区二：把供养做成无止境投入，不看回报

反过来的误区同样致命：把"培育供养"理解成对任何东西都无限地投人、投钱、投时间，不问它到底有没有真正的供养者。供养不是慈善，不是给一个没人喂的死物强行续命。第二章的分诊就是为了防这一条——**当一个东西反复验证出没有自发供养、没人会因它停摆而疼，正确的动作是让它体面退场，而不是继续烧钱抢救。**培育供养是把有限的生命投给真的有人喂、喂得起来的东西，是一种选择，不是一种滥情。

误区三：把生长缝做成失控的口子

留缝不等于放任。有人一听"开放、可扩展、留白"，就把产品或平台变成谁都能乱塞、毫无秩序的烂摊子，最后供养者被劣质贡献淹没，反而不敢投入。生长缝要开放，但要有边界、有质量的守门、有把好贡献沉淀进主干的机制。**留缝是请人进来共建，不是拆掉所有的墙。**开放和秩序要一起给，才有人敢把生命放心地投进来。

误区四：嘴上讲供养，奖惩还在奖励定型

最容易自我欺骗的一条：口号换成了"持续被使用""留生长缝",但真正决定人行为的奖金、晋升、汇报口径，仍然只奖励那个把东西做完、封盘、交付的人，仍然把"还在演化"当扣分项。只要底层的奖惩没变，所有人都会继续制造死物，无论墙上贴着什么。**供养不是靠讲出来的，是靠奖惩结构长出来的。**你到底在奖励"完成"还是在奖励"被养活"，员工比你更清楚。

八、一张随身检查清单

把整篇文章压成七个问题，随身带着。每次要发布一个产品、验收一项能力、推一套方法论、办一个社区之前，逐条过一遍；任何一条答不上来，就是断供养的风险点。

一、这个东西现在有没有人不被要求、就主动带着自己关心的问题在用它？没有，它就是死的或将死的，先解决这个再谈别的。

二、使用者能不能把自己的东西塞进来、能不能改造它？不能，它没有生长缝，供养无处附着。

三、它的价值是停在发布/验收那一刻，还是随着被使用在往上长？停住了，它已在折旧，等待落灰。

四、交付之后，谁负责它的持续被使用？有没有正式的时间和计分？无人负责，供养就在制度上下班了。

五、我们的奖惩，是在奖励"做完、定型"，还是在奖励"被越来越多人用起来、一直在长"？奖励定型，就会持续收获死物。

六、有没有一群互相看得见的供养者，在为它注入新东西？只有孤零零的使用者、没有社群，供养随时会断。

七、它曾被怎样供养、现在还有没有人喂、还为未来的供养留了什么接口？这是母论文的总问法，也是所有问题的总纲——一个东西的生死，从来不看它多正确，只看它还有没有人把生命投进去。

把这七问用熟，你看商业世界的眼光会换一套坐标：不再问"这个东西够不够好、对不对"，而是问"它还能不能被继续供养"。前一个问法让你不断打磨精美的死物，后一个问法让你培育活着的东西。产品、方法论、组织能力、社区、理念，无一能靠自身的优秀自存，它们全都**不能自存，只能被养**。你的工作，不是把它们做得更完美，而是让它们一直有人喂——留好每一道生长缝，请进每一个愿意投入生命的人，并且永远停止惩罚那件东西身上"还没做完"的地方，因为那里正是它活着的证据。

德麦国际