Three

Part

第三篇

推出时期

对于软件开发团队来说，没有什么比产品准备推出（Ship Mode）更令人兴奋的了。大多数人认为产品在经过这个阶段后，就可说是诞生了，但我还是倾向于将此一时期细分为三个小阶段：激活、推移和完成（onset, transition, endgame）。这三个小阶段没有明确的间隔，彼此也有重叠的部分，但三者的工作重点却不相同，而每个团队必定都会一一经历，才能将产品推出。每个人对这三个小阶段的区分方式也可能不尽相同，但对于将整个产品推出的催生阶段，每个人都会有类似的看法或感受。

产品的推出：激活

激活产品推出（ship mode: onset）是一段漫长而渐进的过程，就好像婴儿出生前的阵痛。这时候团队开始为产品的推出做一切繁杂的准备工作，就好像分娩前产妇要练习呼吸，身体的机能也会为出生的那一刻作准备一般。在产品推出的那一刻，团队必须全神贯注、倾全力使它顺利出门，实在是个紧张而痛苦的时刻。

假定我们有四或五个里程碑，大概到M3时产品就应该大致成形，M3的到达就是产品推出的第一次“模拟考”。往后的第四个和第五个里程碑，应该更强调产品推出时的

练习，不但是产品本身愈来愈接近推出时该有的模样，里程碑的到达也应该更像是产品的真正推出。

通常进入激活阶段的第一个征兆是，一部分认真思考的组员开始焦虑：“我们真的能做到吗？”这并不表示团队害怕自己做不来，而是一些先知先觉者开始思考许多该做的事、可能降临的困难，想到那么多那么杂的事情要做，觉得千头万绪，简直不知如何是好，尤其是开始焦虑的人还是少数时。逐渐地，恐惧甚至惊慌，像是野火般蔓延到整个团队，也许领导者在公开场合也流露出这种内心的情绪。在整个团队都染上这种紧张的情绪时，领导者反而可以松一口气，因为现在每个人的心里都关心着同一件事，产品的推出是每个人现在强烈的心愿，大家有一种携手同心的感觉，我们正在共同为一个伟大的产品而努力的感觉，经过了那么辛苦的开发终于要有成果的感觉。

有些组员会怀疑，领导者大概是乐观过了头。

现在，领导者要做的事是为大家加油打气，为大家灌

输自信和热忱，有些组员可能会觉得领导者大约是乐观过了头，但大部分的人都会受到鼓舞，理解在这时候焦虑是正常的，是可以被接受的自然反应，而且是创造智能财产过程中必经的压力。

在激活阶段的团队工作效率会达到巅峰状态，因为大势已定，不需要再揣摩或协商。这时候的团队心理有以下几项特点：

- **大势已定**　产品的功能特色、团队中每个人的角色几乎是完全定型了。所有犹豫不决的特色不能早点定案到现在都得删除，组员之间的工作负荷很平均，产品架构完全确定而且趋向成熟，测试计划正在进行，文件撰写原本只有纲要，现在要加入细节。此时一切都没啥好争论的，没有任何疑惑，每个人都知道产品就要诞生，剩下来该做的事也很清楚，去做就是了。艰辛的开发工作终会过去，此时，赶紧把它完工是最重要的。
- **共同一致的信念**　每个人都有一致的信念，我们即将完成产品，时间一到产品就能推出。现在所有的痛苦抉择或权衡都已经过去，大家只要专注于眼前产品的状态就好。产品的催生时期就代表着所有的

争议结束。如果现在还有怀疑或争论（到现在还没解决的话），也只有暂时搁置，到下一版时再说，否则永远没完没了。此时若是还有人对项目有严重的怀疑，产品是永远无法进入催生时期的。

- **每个人都明白**　每个人都明白自己该做什么事，也明白整个团队该做什么事，才能使产品顺利推出。现在已经没有什么事是不确定的了。所以，只要继续把手上的工作做出来即可。延误或是对延误的恐惧，只单纯地和自己的工作有关，不再和别人的工作有太多的依存。在组员心里曾有的焦虑、飘浮无依的不安，现在都可以放掉。我常把这个阶段称作“工作清单”阶段，每个小组都把自己的一大串的工作逐条列在清单上，现在要把清单上的工作逐项完成，打个勾勾，清单上每一项工作都做完时，产品就完全推出了。

开发团队的最高领导人必须身先士卒，率先进入产品推出时该有的心理状态，并且信守一切的承诺，才可能带动所有组员进入这个紧张的时期。对于细节的认知会愈来愈清楚，在工作步调上也许会有点凌乱的危机，但这时候领导人最重要的工作就是带领大家专注于产品的推出，凡

是不太相干的事情尽量排除；在团队中的每一个成员都渴望推出，看到自己辛勤耕耘的结晶是多么欣慰，领导人必须将大家对于推出的渴望，转化成完美的推出工作。

产品的推出：推移

推移阶段（ship mode: transition），在时间上可以跨过一个以上的里程碑，视产品性质的需要与产品的推出而定。基本上，产品推出的推移阶段是产品从开发迈向完成的逐渐稳定阶段。开发工作仍在进行，直到最后一个里程碑为止；如果开发工作的重心是增加功能，势必很难同时追求产品的稳定，自然严重影响产品的推出；由于同时追求增长和稳定实在太耗费心力，这时候主要的开发活动是除虫，凡是无法完成的功能特色都暂时放弃。要放弃任何的功能特色，都难免引起组员的不安情绪，所以，下决定之前必须审慎地分析评估。

如果我们把产品推出当成一条数学上的连续曲线，你会发现当产品愈来愈成熟，所需的开发人力愈来愈少，产品推出就进入了完成阶段（我们稍后再详细讨论）。现在，产品推出的推移阶段即将结束，而将进入早期的完成阶段时，会有下列的迹象：

- 组员的思考更加保守而小心谨慎，避免掉入开发的泥淖。
- 程序置入的动作审慎再审慎，通常会成立一个临时的特勤小组，检查每一个程序置入的动作，务求事先防范大幅的不利影响。
- 增进执行速度和改善使用者接口成为工作的重点，把产品磨亮打光，以期带给外界深刻的印象。收尾成为最重要的方向。
- 修饰的工作完成后，接下来加强产品的稳定度，以达到推出的要求水准，所有的开发工作已经结束，清除错虫的工作也已接近尾声。

产品的推出：完成

在产品推出（激活、推移和完成）的三个阶段中，工作重心发生移转，从准备进入催生时期的心理变化，产品逐渐稳定下来，直到产品推出的完成阶段（ship mode: endgame）时，只剩下一个里程碑了，就是“推出”(shipping)。在概念上，这个阶段的工作相当单纯，就是一张工作清单，当上面所列的项目逐一完成，产品就推出了。工作清单上有几百项，甚至几千项，也不过只是一张

清单罢了，很多但不会太困难，你没有时间想或做这张清单以外的事情，每个人只有心思专注在自己该做的事情上。也许会发生一些状况，使得工作清单上必须增加一些项目，团队会在一阵“适度抗拒”（appropriate resistance）之后接受工作项目的增加；这种适度的抗拒，我把它称之为“谦逊的抗拒”（profound resistance）。

这时候每天召开例行的检讨会是个很不错的方式，如果能常常邀集决策者参与更好。会中的议程不拘泥于惯例或任何形式，但必须是能立即决断的问题，凡是长期性的目标或一时无法解决的问题，都暂时撇开不谈，下次改版时再说。管理者参加这个会议的作用之一就是将议题引导在这个方向上。现在，团队的主要目的是在推出日期时，产品的品质水准要够好。已经进入倒计时阶段，开发人员主要的工作是错虫的更正，而且是重要的错虫——这会影响很多的使用者，或是造成严重具毁灭性的错虫，都必须优先清除。至于使用者界面的美化、执行速度加快、或是新的功能，都不宜在完成阶段讨论。Beta测试版的测试结果，以及外界的反馈，都以“追求产品的稳定性与避免严重错误”为大前提，作小幅度的修改。而其他的建议案，都只是预警相关人员在销售上可能遇到的困难或不利影

响，或是下一版的改进时的建议，但基本上，目前不打算在产品上响应这些项目。

有多少无伤大雅的错虫，存在于你推出去的产品中？

你必须了解，顾客对于软件品质的要求程度，或是说对错虫的忍受程度。在你推出的产品中，有多少错虫因为优先顺位较低，而未及修正就随着产品出门？这些较次要的错虫会不会造成问题？使用者会不会因此而在操作上遇到麻烦？由于在产品的推出阶段，稳定是第一优先考量，所以在决定是否要更正一个错误以前，必须先考虑会不会因此而造成更大更多的错误；有些错误的修正太冒险，错误本身的危害并不大，把它列在产品的读我（readme.doc）档中说明，会比较好。

有时候，就像有些人的阵痛期特别长一样，产品的推出工作如果不太顺利，也可能使上市的日期受到延误，以下有几个法则提供软件开发团队的领导人参考，以期能避免在产品推出时发生延误的情形。

法则 48

Violate at least one sacred cow

关怀多于要求

身为团队的领导人，你可能巴不得所有的属下都拼命工作。但事实上，虽然在产品推出阶段所面临的挑战可能真的需要大家拼命工作，你却不可能“命令”大家为工作牺牲奉献自己的一切。我曾经见过一些没有人性也不懂人性的管理者，要求他的组员晚上加班、周末来上班，结果组员要不是他的命令理都不理，就是在背后嘲弄这位可恨的管理者。

事实上在产品推出的阶段，管理者的作用并未消失，有些仪式性的动作，虽然很小或很简单，却是意义非凡。此时领导人应该用实际的行动来表达他对组员的关怀，肯定他们对组织的贡献，让每一位组员都感受到自己的每一个角色都是那么的重要，领导者的关怀，可以增强组员的向心力，对产品的顺利推出有莫大的影响。

领导者对组员的沟通应该是具有鼓励性，不是强迫加班，而是提供方便的加班环境。例如，写一张小卡片给某位组员（不能用虚假的客套话、公式话，要真诚的关心）、走廊上餐厅里不期而遇时的几句赞美、提供加班时的托婴托儿服务、自掏腰包来个点心宵夜，或是让组员可以很方便地在家工作（简单的远距办公室通信设备）；这些小动作花费极少，却让组员体会你的关心与诚意，而感到自己

的加倍付出是值得的。还有，这些小动作似乎都是与传统规矩背道而驰，你为了他们而打破这些传统的规矩，也让组员们觉得你对他们的肯定超过公司给你的规定。一般来说，优秀的开发人员多少都有点叛逆心理，喜欢对传统的权威挑战，你用别出心裁的方式鼓舞他们，比用传统的方式要来得有效。

优秀的开发人员多少都喜欢违抗传统。

以行动表示的关怀，同时还告诉组员产品推出是件值得纪念的大事。大家合拍一张照片，写一篇小故事，收集一些开发过程中辛苦的笑料，送个电子邮件恭贺大家产品终于要推出，或是表扬一下杰出的组员或事迹。这些都是领导人在这时候可以做的事情，让大家留下难忘的回忆。

好好享受这欢笑的收割吧！因为大家都曾经流泪播种。产品推出是大家形成一个团队的目的，也是大家同甘共苦情绪的最高点。这时候团队士气是最高昂的，所有曾经犯下的愚蠢行为和几近干戈的争议，现在一律抛诸脑后。大家已经完成任务，产品推出了。

法则 49

Beta is not the time to change

Beta测试版不是修改功能的时候

几乎全世界的人都有这种误解，以为Beta测试版就是邀集外界对产品提出设计方面的改进建议，然后让软件公司对功能做加强或修改。这是完全错误的。（Alpha测试又名内部测试，Beta测试又名外部测试，Beta测试版是把即将推出的产品提供给部分顾客试用，另一方面也作为对市场的试探。）Beta测试的目的是确定产品是否能在预计的各种硬件平台与操作系统中正常运作。虽然Beta测试的反馈意见很有参考价值，但除非Beta测试中发现产品有重大问题，否则不应对功能再做修改，顶多只是更正错误。所有的建议和反馈都留到下一版时再考虑纳入。

这么做绝不表示忽视顾客的意见，相反地，如果你要把Beta测试的意见纳入，那你永远没有办法推出产品。你与顾客之间的关系应该是更亲密、更持续的（请参考本书第一篇中的顾客关系）。

法则 50

The Beta is for spin development

Beta测试是暖身活动

顾客对产品的第一个印象往往决定了他对产品的评价。基本上，Beta测试者的反应，也会是大部分顾客的反应。行销人员应该把握Beta测试的机会，了解测试者对产品的感受，分类并记录测试者的反应，以便在产品的包装、信息传达等方面抓对方向。即使在这时候主要的信息已经发布给媒体，敏锐的行销人员仍然能够从Beta测试的结果，得知那些应该强化或淡化的信息。行销人员应该建立一套顾客心理学的模型，用来了解顾客在使用产品时的心理状态及其变化，以及产品信息应该如何根据试用顾客的心理反应来做调整。

毫无疑问，你一定会在Beta测试中得到一些负面的评价，这很正常，而且应该在你的意料之中。如果负面反应很强烈的话，你得设法找出其他的产品优点去平衡它；如果负面反应很普遍，每个测试者都有，那你得让大众降低对产品期望的水准，让这个负面反应在消费者的意料之中，而不致产生吃惊受骗的感觉，这样负面的反应就不会那么强烈；最后，你得设法让这个产品的缺点看起来不那么严重，比方说提供解决的操作方式等等。

就算Beta测试活动不是由行销人员所主导，也应该让行销人员密切参与。在这个时候，与外界的沟通就可说是

产品成败的关键了。产品信息比产品本身还重要，现在已经不是用开发活动来创造产品的形象。

法则 51

Triage ruthlessly

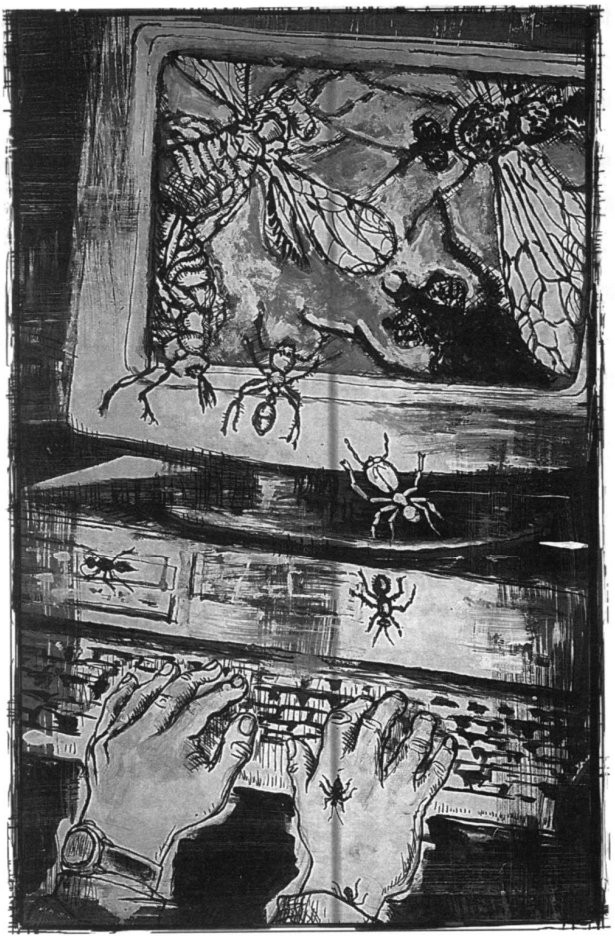
急救术

如果软件代表开发团队，那老实说，它一定有数不清的缺点。欠缺完美是所有智能财产的必然性质。世上所有的事物都不完美，所以问题不在于判断这个产品好或不好，而是决定应该修改那一部分，使它比较能被顾客接受或喜爱。我们把这个判断并修改的过程称之为“急救术”(triage)。

急救术，当然是个医学名词，在急诊室中医生必须非常迅速地查看所有的问题，采取最急迫必要的急救医疗措施，然后再依优先级分别施救。软件急救术是非常类似的观念，先分析各项错误与瑕疵，以及它的严重性，以下是判断是否施救的准则：

- **错误的严重性：**如果错误严重到必须回收所有已推出的产品时，那么就必须立即改正这项错误。
- **明显程度：**错误会很明显而立刻被使用者发现吗？会不会影响到产品的品质？会影响到产品的形象，而成为竞争者攻击或嘲笑的把柄吗？
- **影响范围：**有多少比例的使用时间，会遇到这个错误？这个错误是大部分的使用者都会遇上的吗？
- **修正错误的风险：**如果要修正这个错误，会不会造成软件的不稳定？这需要非常资深又对产品了如指

掌的开发人员来判断。



错虫满天飞

- **团队动态：**为了修改这项错误是否需要动用大批的人力，抑或是一小部分人员投入即可？会不会使已经忙碌不堪的开发人员更加人仰马翻？
- **修正错误后所需要的测试成本：**为了任何理由修改任何部分都需要测试，团队有没有足够的人力来执行测试工作？时间上允不允许测试？

一般而言，急救小组的任务是选择和修改产品中最重要的错误，尽量让大部分的使用者在大部分的时间都能使用愉快。对使用者而言，这项不完美的产品是帮助还是灾难？他是用这个不完美的产品比较好，还是宁愿不用比较快乐？这个复杂且严肃的问题，需要工作人员不断设身处地去设想使用者的情况，揣摩使用者的心情，与使用者双向沟通，才能探得正确的答案。

法则 52

Don't shake the Jell-0

小心保持软件的稳定

处于完成阶段，你必须灌输组员一个观念：修改软件是一件危险的事。软件马上要推出，这时候稳定绝对是第一要务。错误修改的成本或风险如果太高，宁愿不要修改；不必要的修改必须极力避免。

产品的推出，就像一盘特大号的、结构脆弱的果冻，你一摇动盘子，果冻必定会颤动、摇晃，然后逐渐静止。你摇动愈用力，果冻晃得愈厉害，需要变回静止的时间就愈长。同样地，你修改任何一个错误，都不可避免地影响到整个软件的稳定性，等到你的软件像果冻一样恢复静止，恭喜你，那就是你推出产品的时机了！

然后，它会再度震动。