# 读后感

早期的瀑布开发模式已经不适合这个版本了，需要更加灵敏和细致的开发手段。

通路模型指的是以明确项目最终形态为目的，同时启用多个原型进行验证，在过程中将无用的的原型剔除的方法。

敏捷宣言：

个体和互动高于流程和工具

工作的软件高于详尽的文档

客户合作高于合同谈判

响应变化高于遵循计划

瀑布模型在大型项目中显得过于臃肿和模糊，无法对各种突发状况做出敏捷反应。迭代模型不断明确核心玩法，通过设置价值优先级，保证每次迭代都找到关键问题并加以解决。

创建良好的团队结构，树立小组长，使问题能得到及时的解决，营造良好的沟通氛围，不是不重流程，只是更重效率。

迭代开发包括：

概念 设计 编码 游戏资源制作 调试 优化 调优和打磨

检视和适应调整原则是敏捷实践的基石。团队在每个迭代过程中审视自己团队出现的问题，不断调整。

敏捷项目流程：

概念 原型开发 正式制作 后期制作

Scrum的关键在于透明度，每次的迭代都需要大家如实的将自己的问题暴露出来，上级的执行力也非常重要，发现问题还要加以解决才行。

在项目初期，团队会从称为产品Backlog的特性优先级列表中选择一些特性进行开发，每个特性都称为一个PBI，团队再分别估计完成每个PBI需要做多少个任务。

团队只承诺自己认为能够在一个Sprint中完成的功能。

团队每天都要集中开一个15分钟的限时会议，大家分享进展和障碍（从这就能看出敏捷的特性，快，不将任何问题滞留，将整个项目牢牢看住）

时间盒（time box）是指某个会议、任务或工作中的一段固定的时间量，限制了花费。（提高时间利用率）

Scrum原则：

经验论：每日例会使团队快速做出反应

浮现论：在开发过程中我们会渐渐对整个项目有更明晰的理解，这是通过回顾和循环达到的

限时：定期交付价值。通过对每次的评估，使项目越来越好。

优先级：重要的是玩家体验，一切以玩家愿不愿意玩为准则。

自组织：将权利授予小型团队中的领导者，使开发变得更加高效。

产品Backlog是一个按照游戏需求或功能优先级排列的列表，在其中存放PBI（即要实现的功能模块），困难和有风险的PBI需要优先处理。

在迭代过程中，尽早对不确定的风险进行解决可以降低工作的不确定性。提前做出可以供市场评判的测试版本可以有效的对项目做出评估和调整。

一个Scrum团队包括ScrumMaster，PO（产品负责人）和开发团队。

ScrumMaster负责对团队的Ｓｃｒｕｍ思想进行指导。促进问题的解决。处理团队中的问题。

ＰＯ负责游戏的展示效果并使投入产出比（ROI）最大化。负责与市场和投资者的交流。制定发布时间和计划。

团队的工作能力主要基于过去的表现，由此设定每个迭代的工作量，每次的Sprint需要从backlog中选取优先级最高的PBI，将其加入Sprintbacklog中，直到将这次的工作时间用完。如果在最后几天前做完了工作，没有必要添加新的PBI，而是应该完善之前的工作。

Sprint的时间不止由工作能力，开发周期决定，也要考虑到工作强度的问题，过于频繁不好，战线拉的太长也不好。

使用任务卡，燃尽图等展示任务的完成进度，任务卡使得每个人参与到任务的创建和管理中，燃尽图则可以对开发情况做出大概的预测。一块任务板就可以将这两种东西放在一起。（电视剧硅谷中就使用了此种开发手法）

Sprint在最后一天会进行团队和项目干系人对游戏的试玩和讨论，由PO验收，如果没有通过，就要决定是再来还是放弃。总之不能放着不管。

开发过程是复杂和多变的，对任何形式的开发都是如此，但只要每个人各司其职，干好自己的活，最后的结果一定是有益的。

用户故事没看明白是什么。大概是指用户想要体验到的流程。用处主要是方便各类开发人员的交流。细化出来就是我们需要去做的游戏内容。什么是好故事呢，大概具有以下特点：

1.独立性

2.有价值

3.可商讨

4.可估算

5.大小适当

6.可测试

如何定义“完成”：大概就是可以在决定好的平台上正常运行吧。不同的要求使得成品成色不同。

故事按重要性可以分出史诗（即最主要的需求），在故事收集研讨会时，需要为下一个版本收集足够的故事细节，即确定接下来要开发什么。

在敏捷开发中，计划也是非常重要的一部分，但应控制在计划上放的精力。

在Sprint计划中，主要面向内部，而发布计划，由PO带头，在两年中一般只有一次，需要经过严格的测试，至少应满足对“完成”的要求。

做计划是使事情更加明了，而不是更加死板，所以计划要随情况调整，而不是强行完成计划。

在做计划的一开始，可能对情况有着过分乐观的预期，重要的是在过程中对计划的及时调整。保证进度和质量往往是以牺牲一定的内容以前提的。通过良好的沟通和商讨，可以极大降低团队压力和发售风险。

但无论怎么削减，项目应满足最低需求组合：

8-12小时的单人游戏内容

固定发布日期

最低要求的功能集

在做计划时，应将游戏各个部分所占的时间和资源提出来，进行取舍，在开始时，这个时间一般是由PO的经验给出，

概念（想法）->原型制作（创建价值载体资源）->制作阶段（集中制作游戏流程）->后期制作（完善相应的游戏体验）

PO的经验非常关键，因为他可以大致估算出一个新主意所需增加的工作量，并加以选择，是分配时间还是选择放弃。通过回报曲线，可以对投入的时间所得到的收入进行预估，从而不亏。需要让团队里每个人都忙碌起来，不浪费资源，通过时间窗，将每个环节衔接起来。时间窗在制作阶段会发生变化，重新定义并减少时间消耗

。在精益制作中，我们关注整个流程，想尽一切办法减少循环时间。在时间窗上可以通过缓冲使项目更加有条理而不显得慌乱。减少无关的活动，限制工作的范围可以使人们更好发挥自己的创造力。在资金和时间足够时，许多问题不会影响到工作，但是问题还是存在，那就要尽早加以解决。

良好的团队是成功的保障之一，它的执行力和热情超乎任何想象，不是大家一起干一件事就叫团队，不需要大家互相关系多好，多么熟悉，只要大家目标一致，即将自己负责的那部分做到最好，有问题处理问题，避免拖沓和内耗。任务板的存在也是方便大家明确自己的职能，是一种激励，也是一种荣誉，大家可以看到自己的付出和整体的进度。良好的团队应能自我组织和管理，在一个模块上具有完整的职能，成员一般为7~9人。领导负责激励和引导团队，制定要求，审核成果。

特性团队是开发游戏核心功能的跨职能团队，它应拥有制作一个玩法所需要的所有人，实际上有许多人员是由多个团队共享的。

功能团队通常只用于底层或基础构建方面的工作，从书中的描述来看，主要针对程序员集体（好像是说技术人员就不要参加那些需要审美的环节了）....

制作团队通常出现在制作阶段，由特性团队发展而来，成员发生了转变，组成中有更多的效果制作者（直接给玩家看的那种，而不是底层实现）。

公共结构团队（SI）提供许多团队都要用到的共享的支持服务，如引擎开发和动画过场等，优先级顺序很重要，时间是开发中最珍贵和不可再生的资源，SI必须将每一秒时间都用到关键的价值上，在理想情况下，SI与管理层有着良好的沟通，可以随时对优先级冲突做出判断。拥有自己独立的PO和backlog ，他们还“运营支持团队”的模范，支持一个或更多已经发行的游戏。

工具团队由好几个工具制作者组成，他们的客户是使用常用工具集和流程的用户。类似于上面的公共结构团队。

资源池团队是属于一个单一职能的开发者集合，他们没有自己的Sprint目标，一切为了团队，称为内包，他们需要更多的计划和管理，以保证被完全利用。

集成团队负责将各个团队的工作进行集合，emm，就这，一般在大型项目中使用。

大型团队证明了人数的增多和效率的增长是不成比例的，交流的代价是极大的，所以我们需要Scrum的SoS会议

这个会议可以看作是团队内部的峰会，代表们让整个团队了解自己的每个部分的状态，把自己对其他团队可能产生的影响进行阐述，SoS没有产品Backlog。但会有公共障碍的简短Backlog。处理那些对团队有较大影响的障碍，如关键开发人员生病。

在大型项目中，一个PO可能不够用，这时候就要建立PO层级，让一个领导PO掌控整个项目的一致愿景，通过频繁的会议将大家的愿景统一起来。

是否需要一个统一的Sprint，独立的进度可能使PO的使用变得不那么紧张，但统一的Sprint可以有效的提高团队间的交互和时间利用。

实践社区是一个不同团队的类似职能人员知识分享并消除重复工作的地方，人员自己决定频率，不是发布和处理任务的地方，而是分享信息的地方。

分布式团队的问题很明显，就是最后的整合能不能将内耗降到最低，游戏的质量能不能保证。解决办法是高迭代高沟通高责任心。

迭代分为个人迭代和版本迭代，个人迭代是指个人所用工具和方法的更新，版本迭代是编码和资产在整个项目中的变更。过程是提交，测试，部署。

敏捷游戏开发有四个要素，敏捷技术，敏捷美术和音频，敏捷设计，敏捷QA和制作。

结对编程，我看了下定义，大概是一个人编，另一个在旁边看并bb，我觉得不行。带新人也不应在这么紧张的Scrum项目中，这是在浪费时间。

敏捷技术优化很重要，算好自己的技术债也很重要，利用硬性指标来帮助团队做决定。

敏捷美术也是很重要的，因为美术的资产不好直接衡量，而且对工具的要求很高，我在书中也没看明白用Scrum好在哪？

设计中如果设计出了价值为0的部件，那么一切加在上面的工作都是徒劳，一定要注意。

为什么这本书看到后面都是一些看过了例子或者废话呢。

Scrum用好了才有效果，但也不是包治百病，许多开发商在使用它时缺乏安全感，只有真正使用Scrum才能掌握其精髓。

全书完