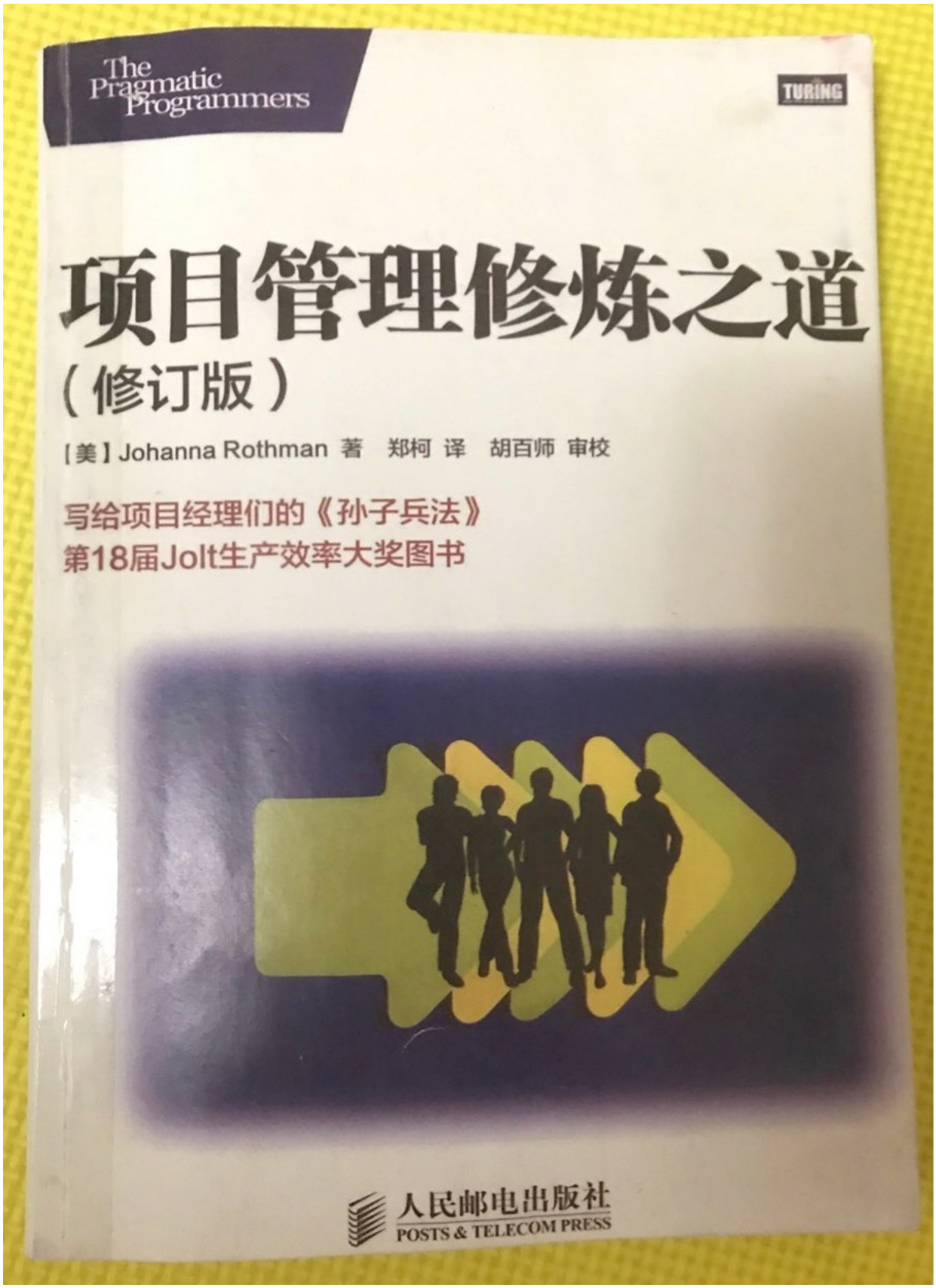
项目管理修炼之道



一、启动项目

1、定义项目和项目经理

- 项目：一个独特的任务或系统化的流程，其目的是创建新的产品或服务，完成交付产品和服务，标志着项目的结束。项目都有风险，且受制于有限的资源。
- 产品：项目所产生的一系列交付物。
- 项目经理：负责向团队清晰说明完成的含义，并且带领团队完成项目的人。完成是指产品符合组织对这个产品的要求，也满足客户使用这个产品的需要。

2、管理项目的关键驱动因素、约束和浮动因素

- 理想状况下，关键驱动因素应该只有一个，约束应该只有一个，而浮动因素可以有四个。
- 成本、时间、功能、质量。

3、与客户或出资人讨论项目约束

- 问客户是什么在驱动这个项目?

4、决定项目的关键驱动因素

- 驱动因素矩阵，草拟后让客户排序
    - 发布成本
    - 发布日期
    - 功能集合
    - 减少缺陷
    - 人员配备
    - 工作环境

5、应对喜欢过多干预项目的出资人

预测未来

- 设想现在离项目预定的交付时间只剩3周，有功能未实现，同时有很多严重缺陷等着修复，问出资人该怎么办?

使用与上下文无关的问题识别项目真正的驱动因素

- 项目要怎么样才算成功?
- 为什么想要得到这样的结果?
- 这种解决方案对你来说价值何在?
- 这个系统要解决什么样的问题?
- 这个系统可能会造成什么岩的问题?
- 少用“为什么”、“怎么做”之类的问题

6、编写项目章程、共享现有决策

远景

- 每个项目的背后有一个缘由，发起这个项目的缘由何在？项目的价值何在?

需求

- 在某个特定的日期到达之时发布某些功能

目标

- 项目经理希望通过项目要达成的目的，客户可能不赞同这些目标
    - 重构解决技术债
    - 添加更多的自动化测试
    - 设计冒烟测试
- 项目不一定要交付它的目标

成功标准

- 项目完成后，客户能用交付的产品做什么，这就是成功标准的定义

ROI（投资回报率）

- 操控数字太容易，不过可以试着去计算

7、理解质量对于项目的意义

- 根据项目产品所处的不同市场应用阶段，客户对于质量的定义也有所差别
    - 早期市场：关注于独特之处
    - 初期市场：关注于功能，可以容忍缺陷
    - 大众市场：关注于稳定和功能，不能有过多缺陷
    - 末期市场：客户群体日益增大，尽量要求减少缺陷，项目的压力会更大

8、小结

- 每个项目启动时都要有章程
- 对项目章程的反复修改要有心理准备
- 章程不一定完美，他的意义在于帮助整个项目团队进行规划活动
- 要知道质量的意义以及项目的驱动因素，这样随着项目的不断推进，项目经理和团队才可以做成正确的决策

二、规划项目

规划和日程安排是两种不同的活动。

- 规划：是指制订带有发布条件的项目计划
- 日程安排：是对项目工作的有序描述

1、踏上征程

- 人数不超过10个人，不妨一起完成项目规划
- 未来几天或几周要做什么?

2、使项目足以启动的规划

- 规划不必完美无缺，只要这个规划能让项目启动起来，同时让大家看到成功的希望，就可以了
- 时间盒：指特定的时间长度，个人或团队用它来完成每项特定的任务
- 用时间盒来限制启动规划活动
- 要根据经验而不是预言来规划项目
    - 先做少量规划，再根据实际收集到的反馈规划将来
    - 预言式规划试图预测未来，这很难奏效，除非你有水晶球
- 规划毫无用处，但是制定规划必不可少

3、开发项目规划模版

产品意图

- 为什么公司要开发这个产品，它能为公司带来哪些效益
- 可以借用章程里的远景

历史记录

- 复查已发布版本中发现的缺陷数量和类型，就可以理解即将使用的代码库中技术债务的严重程度和类型
- 对项目的情况知道的越少，将来感到惊讶的几率也就越大

发布条件

- 详细列举项目产品的关键可交付物
    - 功能
    - 性能
    - 质量

目标

- 产品目标（有点费解）
    - 不在当前发布版本完成的需求
- 项目目标
    - 缺陷数目明显减少
- 团队目标
    - 增加自动化测试比例
- 组织目标
    - 锻炼组织

项目组织

- 明确说明团队在项目中的职责分配
- 指明项目经理如何使用生命周期组织项目工作，要采纳哪些关键实践
- 是否有决策人可以影响当前项目
- 如果使用受时间盒限制的迭代，需要说清楚每个时间盒的长度是多久

- 如果使用版本火车，要说明每列火车持续多久

日程总览

- 日程总览里标有主要里程碑
- 说明里程碑或迭代的持续时间
- 说明里程碑或迭代结束后可以得到哪些产出
- 不要放工作分解结构（WBS），可以给参考链接
    - 工作分解结构：它是任务的组织形式，展示它们之间的依赖、持续时间和所有者等信息
- 以矩阵的方式给出项目团队不同的运作方案

方案	实际时间	架构时间	开发时间	测试时间	文档时间
功能1、2	2个月	1人月	6人月	6人月	2人月
功能1、2、3	4个月	2人月	10人月	10人月	3人月
功能1、2、3、4、5、5	12个月	6人月	60人月	50人月	8人月

人员配备

- 要在何时需要多少人，需要何种类型的人

建议日程

- 使用黄色便利贴安排日程
- 很少使用完整的甘特图
- 小心过早细化日程
    - 反复修订项目日程
    - 随项目进程补充细节

制定项目风险列表

- 要将排名前十的风险记录在案
- 要经常监控这些风险
- 适当时机更新这个列表
- 项目风险如果不到10个，不妨和团队成员一起来一次头脑风暴
- 风险列表

序号	描述	发生概率	严重程度	暴露程度	反应时间	应对计划
为每一个风险排定序号	用一个短语或一句话为风险命名	风险发生概率	如果风险发生，会造成影响的严重程度	发生概率乘以严重程度	应对风险的时间	风险应对方案
1	除非到了项目的后期，否则不知道算法的速度是否够快	中等	高	（M，H）	7-14	增加测试人员，和开发人员一起测试
2	启动操作2分钟	低	高	（L，H）	8-21	在每次构建时监控启动过程

4、制定发布条件

- 发布条件会告诉你项目中“完成”的含义
- 避免大家对“完成”的定义各不相同
- 制订发布条件时，先判断项目的关键因素时什么？
    - 为什么公司要做这个项目？
    - 为什么客户会买这个产品？
- 项目经理和各职能部门共同商讨“成功”的含义，为项目经理指明了成功的方向
    - 销售人员能够卖出满足条件的产品吗？
    - 技术支持人员能够为这个产品提供技术支持吗？
    - 培训人员能否制定培训计划并展开相关培训？
- 确定当前项目最重要的因素
    - 客户关注的是这个产品是否解决了困扰客户的问题，而不会考虑其中有无缺陷或是缺陷数量
    - 有时，发布日期是最重要的
    - 有时，某一个或几个功能决定了项目的成败
    - 通常来说，日程安排、功能特性、和低缺陷率共同构成项目的关键因素
- 草拟发布条件
    - 如果条件许可，要在上市时间、客户需求、缺陷状况、性能和可靠水平之间达到平衡
    - 注明“草拟”字样，让大家知道只是供讨论的草稿，而不是已经作出的承诺
    - 发布条件草案样例
        - 所有代码必须针对所有运行平台编译并构建

        - 没有高优先级的Bug
        - 所有未解决的Bug和临时解决方案都要记录在版本发布说明中
        - 所有计划好的测试都要运行，通过率98%以上（比例可以讨论）
        - 在最后三周，未解决的缺陷数目要下降
        - 在发布之前，要由开发人员完成单元测试，由测试组完成系统测试，由客户A、B完成验证
        - 准备6-1日发布
        - 所有未解决的缺陷都要由跨职能团队进行评估（讨论有无必要）
- 让发布条件符合SMART原则
    - 要确保团队每个人对发布条件的理解完全相同，可以使用SMART原则：
        - 确定的
        - 可测量的
        - 可达成的
        - 相关的
        - 可跟踪的
- 在发布条件上达成多方共识
    - 取得共识的过程中，可以提以下问题
        - 在发布日期之前，我们是不是必须满足这个条件?
        - 要是我们不能在发布日期之前满足这个条件，会发生什么?
        - 如果不能满足这个条件，我们的产品或公司是不是会因此而承担风险？人们的安全感是不是会因此被破坏?
    - 通过以下方式来获得大家共识
        - 草拟发布条件，针对其展开讨论，在团队会议上就其达成共识
        - 与整个团队草拟发布条件
        - 与团队各职能经理一起草拟发布条件

5、使用发布条件

- 到产品发布的时候，发布条件职能有“满足”或“未满足”两种状态
- 要跟团队一起讨论距离满足发布条件还要多久
- 发布条件可以作为早期的告警信号，让大家知道团队可能无法准时完成当前版本
- 在接下来的每周项目团队会议上，用发布条件来确认项目在不断取得进展
- 下列两种情况发生时，项目经理可以考虑变更发布条件
    - 进一步了解项目中关于“完成”的含义时
    - 认识到无法在预定发布日期前满足所有发布条件时
- 如果管理的项目无法满足发布条件
    - 要跟团队确认为什么无法满足条件
    - 要向管理层解释无法满足条件的原因

6、小结

- 项目规划是在不断进行的，这只是开始

- 为项目团队、出资人和项目经理自己制定发布条件，以及明确定义“完成”的含义
- 项目规划不必完美无缺，但是必须要有

三、使用生命周期组织项目

一旦踏上项目之路，没有哪种生命周期是完美无缺的，可能需要对其做些拓展，并且不时修修补补，所选的生命周期模型要保证足够灵活。

1、理解项目生命周期

- 生命周期是项目经理和团队组织产品开发的方式
- 定义需求、设计、开发、测试以及这些工作同时进行的过程，都算生命周期的一部分
- 从整体上组织项目时，不要把现实情况理想化
    - 不要在规划时就希望能先产生完整的需求
    - 选择的生命周期能随着项目的推进不断发现新的需求
    - 选择的生命周期要有助于识别项目风险，帮助项目经理交付成功的产品
- 很少有项目能够沿着需求到发布这样一条直线推进，也没有多少项目能够规规矩矩按计划进行

2、生命周期概览

各种生命周期如何管理风险：

生命周期类型	范例	优势以及成功的必要条件	项目优先级	成功预期
顺序式	瀑布、阶段-关卡式	管理成本风险；需求已知并已达成共识；系统架构已被深入理解；项目需求不会发生变化；项目团队不会发生变化	1、功能集合；2、低缺陷率；3、发布时间	成功，并可得到反馈
迭代式	螺旋、不断演化的原型	管理技术风险；不断演化的需求	1、功能集合；2、低缺陷率；3、发布时间	迭代中的任务已经做规划，并按计划完成
增量式	项目日程由设计决定，并按阶段交付	管理日程风险；可以应对小的需求变更；但是难以处理影响到架构的变更	1、发布时间；2、低缺陷率；3、功能集合	成功
迭代/增量式	敏捷（Scrum、XP）	管理日程和技术风险；如果团队成员不在同一物理位置，人员职能不完备，就难以实施	1、发布时间；2、功能集合；3、低缺陷率	成功
即兴式	编码并修复		1、发布时间；2、功能集合；3、低缺陷率	不可能成功

各种生命周期的各阶段表示：

- 顺序式
  - 需求收集
  - 分析
  - 设计
  - 编码
  - 集成
  - 测试
- 迭代式
  - 需求收集
  - 原型阶段：分析、设计、编码
  - 原型阶段：分析、设计、编码

    - 原型阶段：分析、设计、编码
    - 集成
    - 测试
- 增量式
    - 收集部分需求
    - 分析选择整体架构
    - 设计、编码、集成、测试
    - 设计、编码、集成、测试
    - 设计、编码、集成、测试
    - 最终集成
    - 最终测试
- 敏捷
    - 收集部分需求和规划游戏
    - 时间盒
    - 时间盒
    - 时间盒
        - 每个时间盒产生已通过测试并可以运行的功能
- 生命周期组合创新
    - 了解初步需求
    - 将目前了解到的需求原型化，收集反馈，选择技术架构
    - 完整实现3个功能，并随即进行集成
    - 测试架构，演示现有功能
    - 继续实现，并不断集成
    - 最终集成

3、在项目中寻求反馈

- 开发人员首次获得代码的反馈是通过测试完成的
- 如果测试很晚才集成，这些反馈也是很晚才得到
- 开发人员得不到反馈，项目经理就很难评估剩余的项目风险、项目的状态以及团队的工作效率
- 缺少反馈的生命周期（顺序式）被称为“预测式”生命周期

4、大规模项目需要组合使用多种生命周期

- 没有哪种生命周期可以满足所有人的要求
- 对于大规模的团队或是多个地点展开的项目，可能每个团队都使用自己的生命周期

5、管理架构风险

- 架构风险主要是指团队选择的架构能否满足当前项目的要求
- 顺序式周期中，最终的集成和测试是在最后发生的，对于架构风险的控制是最差的
- 项目经理需要随时注意架构能否满足项目要求，这方面发现的越早越好，如果不能及时觉察，整个项目可能因此而功败垂成

6、从瀑布中摆脱出来

- 用迭代来规划所有的工作，包括规划、需求收集和原型化等工作
- 将产品原型化，并尽早先客户展示，从他们那得到的反馈越多，后面的日子越好过
- 从项目一开始就加入测试工作
- 功能要逐个实现，完成后随即进行集成和测试

7、我最钟爱的生命周期

- 尽早向代码库中提交功能代码
- 除非团队完成了几个功能的开发、测试、集成，否则架构师无法确定下来的
- 敏捷是首选的生命周期，如果客户没有足够投入、或者团队对敏捷式协作好无兴趣，可以选择阶段式交付生命周期，并用时间盒限制需求收集和架构设计阶段

8、小结

- 在组织项目时，要自由使用任何生命周期或是多种生命周期的组合，让项目踏上成功之路
- 不要怯于创建反应你自己项目实际情况的生命周期。“完美的”生命周期只是模型而已，你是生活在现实世界中的
- 阶段-关卡流程或瀑布式生命周期，只有在确定使用它可以获得成功时才使用，而不是不经思考，上来就用

四、安排项目日程

最初的规划只要能让项目启动就可以了。在安排项目日程时，要准备改进规划。在重新规划时，如果需要改变日程，也不必担心。

1、注重实效的项目日程安排

- 你之前的项目规划，足以启动项目了
- 接下来的日程安排，只要能把项目启动起来，也就行了
- 既然知道项目会随着时间演变，就没有必要详细安排整个项目的日程了
- 建议只思考接下来大约一周的事情，然后构建主要的里程碑，并对后续几周的工作进行波浪式日程安排
- 创建日程时，可以将生命周期用作指导方针，但不是严格的限制条件
- 日程排定要安排工作任务的顺序，展示它们之间的依赖关系
- 使用时间盒限制初始规划
    - 用一个小时制订项目的章程
    - 用一个小时完成项目的规划
    - 用一个小时完成日程安排的第一版草案

2、可供选择的项目日程安排技术

自顶向下式日程安排

- 自顶向下式的日程安排通常从里程碑开始

- 将项目日程分为阶段、迭代或是几大块；把它们华仔白板上或者用便利贴粘在墙上
- 使用放在墙上的卡片安排日程，每个人把应该由自己完成的任务写在卡片上；然后再用线把卡片连接起来
- 最底层的任务越小，就越容易估算完成任务所需要的时间

自底向上式日程安排

- 自底向上式的日程安排从特定任务开始
- 使用增量式生命周期时用自底向上还不错
- 项目团队成员会从任务中产生里程碑

由内向外式日程安排

- 安排日程前，先画一个思维导图，把所说知道的和项目相关的有关东西都放进去
- 保证团队成员在一起制订思维导图，否则思维导图不好理解

哈德逊湾式启动

- 运送皮毛的商船，离港后先停留在几海里的位置一段时间，确保不会忘记工具和给养
- 可以让项目团队先尝试在项目的实际环境中开展某些工作，目的是帮助理解环境；项目经理应该尽量缩短这个过程
- 使用时间盒来限制哈德逊湾式启动

短期迭代

- 当团队对如何估算任务没有什么把握时，可以使用短期迭代
- 可以在哈德逊湾式启动后再进行一次短期迭代
- 使用时间盒限制短期迭代（不要炒作2周），再看看团队能够王恒多少工作
- 如果团队可以一起进行短期迭代和短期回顾，大家就能更清楚的知道如何安排当前项目的日程

3、用低技术含量工具安排项目日程

- 把任务写在便利贴上，然后在贴墙上，根团队成员一起讨论任务的顺序、任务的分配和风险
- 便利贴可以让整个团队参与项目的日程安排的讨论，团队也会不断说明对风险的理解，并为项目经理提供掌握项目的有价值信息
- 项目团队拥有时间表的所有权，它们就会对其负责到底
- 如果项目经理拥有所有权，这就像是在对团队展开微管理，导致管的太细，而不是管理他们任务之间的依赖关系

使用便利贴安排项目日程的基本技术

- 团队全员参与，每人准备一叠便利纸和笔，准备一个可以粘贴的墙
- 将主要的里程碑写下来贴在墙上，这样大家可以看到项目的整体结构
- 让每个人把其所有的任务都写在便利贴上，一个便利贴写一个任务，写完后粘贴在墙上
- 墙上留一个区域用于粘贴问题和假设
- 项目团队成员一起讨论事件的顺序、前置条件、假定以及问题
- 项目经理如果看到很长的任务安排列表，就得问问团队是什么阻碍了他们以并行的方式工作

使用便利贴和箭头安排项目日程

- 箭头可以辅助跟踪依赖关系

- 方便将结果录入日程安排软件
- 便利贴掉下来后，更容易知道原来位置

为每一个职能组使用便利贴安排项目日程

- 按职能划分的团队会拖项目后腿
- 大白版画几条竖线，每条表示一周
- 使用不同颜色的便利贴表示不同职能组织的不同人的工作状况
- 同时标明项目的结束时间
- 按职能组织的项目在结束时会很难受，在项目最需要他们的时候，开发人员却很难全心投入

按功能使用便利贴安排项目日程

- 将每个可交付物用一张便利贴写好
- 同时写好截止日期

使用便利贴安排项目的好处

- 每个人都要有意识去思考关键路径

基于可交付物的规划

- 使人们思考必须为项目剩下的工作交付哪些东西

4、小结

- 用低技术含量的工具开始安排项目日程。如果的确需要相关软件工具，过后再转换数据。
- 按可交付物安排日程，而不是按功能
- 要准备好以迭代的方式安排日程。一次性完成的项目时间表，起作用根本对不起花在上面的时间。

五、估算工作

任何通过计数或是计算的估算技术，都没有成功使用过

1、实用的项目估算方式

- 对比历史数据进行估算
- Delphi和宽带Delphi方式进行估算
    - 团队成员各自写下自己的任务列表和时间估算，同时注明自己对项目的预设条件
    - 团队一起开会，收集任务列表并复查，项目经理汇总
- 何时不应相信团队的估算
    - 不是每个人都善于估算，有些人会过于乐观，有些人会过于悲观
    - 总体来说，不要为任务估算添加过多松散时间，要提供一个估算信心百分比、一个日期范围，甚至可以考虑提供三个日期：最佳状况、最有可能、“墨菲”日期
        - 墨菲法则：如果某事可能会出错，他就一定会在最不应该出错的时候出错
- 小心顺序式生命周期的估算陷阱

    - 如果必须使用顺序式生命周期，是很难对项目做出准确估算的
    - 可以先动手做一点开发工作，测量完成这些工作需要多长时间
    - 再看包括多少类似工作量的工作，并通过迭代方式得到估算
- 使用自信心范围进行估算
    - 自信心范围图
- 使用日期范围进行估算
    - 可以让项目不至于受过早承诺之苦
- 使用三个日期：最佳状况、最有可能、“墨菲”日期
    - 最佳状况：在生产甘特图时，最佳状况日期就是工具提供的日期
    - 最有可能：在最佳状况时间上将一些经验系数、缓冲时间后纳入考虑之中
    - “墨菲”日期：在最有可能日期之上加上一些经验系数
- 规划扑克
    - 让所有团队成员都能参与到估算活动中
    - 每个人都估算任务的时间，然后讨论接受哪个结果
    - 用人时而不是人日进行任务估算
        - 一个很难在一个工作日完成8个小时的工作
        - 我说见过最好的状况，一天也就完成6个小时的技术工作
        - 由于各种会议或其他干扰，他们每天最多就能干4个小时的技术活
- 在估算前用试探性开发收集数据
    - 把任务拆分成一个一个的小块，每个小块耗时4-6小时
- 让估算更容易的方法
    - 记住，估算只是一个大概值
    - 很多软件开发人员都很乐观
    - 完成一项任务总是花费比预计更多的时间
    - 估算小块的工作更容易
    - 估算建议使用人时
    - 项目经理和团队需要练习估算并收集反馈
    - 做好反复估算的准备
    - 如果项目经理必须遵守某个截止日期，那就什么都别估算了，把所有工作按优先级排序，按优先级开发
    - 如果任务过大，不容易估算好，可以先试探性开发

2、用里程碑切分项目

- 用交付物作为里程碑，不要用于功能相关的活动
- 使用基于可交付物的规划来安排任务（而不是与功能相关的活动）
- 将里程碑的结束安排在一周中的周中，哪些完成哪些未完成，可以及时调整

3、能够不做哪些事情

- 能够以“不做哪些事情”的方式去思考时很重要的特质
- 不妨问问管理层“能够不做哪些事情”，如果都得做，那就帮大家澄清了他们的假设

4、身背多个项目时的估算

- 不要估算，也不能估算，想都别想
- 团队中有身背多个项目任务的成员，这个项目必然会延迟，而且项目经理都无法预测会晚多久
- 项目经理需要和出资人讨论：如果不能在我需要他们的时候提供这些人，那我就不能交付你想要的东西，不能满足你对时间和质量的要求，咱们看看哪些东西是你不需要的吧

5、主动安排人们进行多任务

- 有些职位不必专职投入（比如DBA、GUI设计人员）
- 项目经理可以让他们以周为单位来切换项目，确保一周只做一个项目的工作

6、使用波浪式规划

- 波浪式规划就是一个持续不断而且详细的日程安排方法，只覆盖几周的时间
- 完成一周详细工作安排之后，再继续安排一周详细的工作
- 使用四周的波浪式规划
    - 2周的时间太短，无法给我足够信息，难以预见风险
    - 多余4周容易犯错，并偏离安排工作
- 只要在项目进行时牢记每个里程碑，项目经理就会发现日程越来越容易维护了，花在上面的时间也就越来越少

7、决定迭代的持续时间

- 复杂的项目会有大任务，应该将它们拆分成团队可以在四周或更短时间内完成的小任务

8、尽可能使用“小石子”进行估算

- 小石子就是大任务被拆分成的小任务
- 每个小任务的完成时间不会超过2天
- 小石子只有“完成”和“未完成”两种状态，不存在所谓百分比
- 基于每天或每周规划的“小石子”是有用的
- 不要在项目一开始就定义所有小任务，毫无意义
- 项目中的每个人都可以划分自己的“小石子”
- 项目经理、技术带头人或架构师不要试图去为别人产生小石子（可以指导），这样就管的太死，陷于“微管理”

9、小结

- 绝对不要提供确定的项目结束日期
- 任务越小、估算起来就越容易
- 寻求估算的准确性，而不是精确性
    - 准确性：估算离实际情况有多大的差距
    - 精确性：测量的精密程度（比如某一天的几点完成）

六、识别和避免日程安排游戏

即使项目经理自己努力做好估算、规划和日程安排工作，你遇到的出资人、管理层、和团队成员还是有可能视日程安排为儿戏；项目经理需要把这些人带回现实。

1、给我一块石头

- 当出资人希望项目能更快交付，但是不告诉你何时需要或为什么的时候，就是在玩“给我一块石头”的游戏
- 应对实践
    - 提几个问题：日程、人员、功能、品质，什么是最重要的
    - 找出是什么原因促成他们期望的截止日期
    - 让出资人明白你做出选择以及背后的原因
    - 为你提供的日期说明信心范围
    - 在提供日期时要说明发布条件
- 反复发生的应对实践
    - 制订排好优先级的产品待办事项列表
    - 按优先级逐个实现功能
        - 如果让出资人更多的了解项目进程，他们就不会那么纠缠截止日期了
    - 使用短小的时间盒，这样出资人就可以看清项目进程
        - 如果每隔几周就能展示项目有价值的进展，截止日期就没那么重要了

2、“希望”是我们最重要的策略

- 仅有希望，不足以交付一个成功的项目
- 应对实践
    - 识别风险并记录下来
    - 不到万不得已，不要选择瀑布式生命周期
    - 可以用“哈德逊湾式启动”看看是不是能够做出什么东西
    - 确保大家具备相关的技术能力，还有解决问题必备的领域知识
    - 考虑所有工作都采取迭代的方式进行，特别是项目规划和日程安排
    - 由于缺少经验和专业知识可以寻求相关领域的帮助和信息
    - 制订里程碑条件
    - 使用有时间盒限制的迭代
    - 使用速度图表来展示项目进度

3、拒绝女王

- 有些老板不愿意面对项目延期的现实
- 应对实践
    - 找出拒绝的原因，可以尝试问一些上下午无关的问题：“对于这个项目来说，要怎样才算成功？”
    - 写下项目的风险及其潜在影响
    - 展示你能做的事情，并测量团队在项目中的实际开发速度
    - 保证参与项目工作的每个人都有相关领域的知识

    - 如果管理层认为“拒绝”是鼓励团队的方式，建议他采取其他鼓励技巧
    - 只要项目经理能够接受现实，“拒绝女王”就不一定能造成灾难
    - 现实总是会在某个时候跟“决绝女王”面对面，这不可避免
- 反复发生的应对实践
    - 制订排好优先级的产品待办事项列表
    - 按优先级逐个实现功能
        - 等到“拒绝女王”如梦初醒，团队也完成了若干高优先级的功能，这样项目经理也会得到一些有价值的东西
    - 使用短小的时间盒，这样出资人就可以看清项目进程

4、把灰尘扫到地毯下面

- 优先级的变更 — 灰都被扫到地毯下面了
- 应对实践
    - 把某个版本需要的功能先按优先级进行排序
    - 使用有时间盒限制的迭代逐个实现各功能

5、幸福日期

- 善于口舌的管理层，他们威逼、利诱、或用权力来说服项目经理和团队，让他们相信可以满足管理层对“幸福日期”的要求（加上逃避困难话题的文化使团队更容易接受）
- 应对实践
    - 说明日程安排范围
    - 使用迭代式生命周期，并说明将会通过信息方位实现哪些功能
    - 使用速度图表让每个人都了解进度

6、屁股着火

- 项目经理和团队周旋于几个项目之中，或是在两个项目之间不停切换，所有项目的紧急程度不断攀升、攀升、攀升。。。
- 应对实践
    - 考虑创建单一项目的工作环境

7、分散注意力

- 如果管理层无法把精力投入到一个项目或是项目群之中，就会引发此游戏
- 应对实践
    - 为每个项目设定一周的迭代，并确保每个迭代结束时有可发布的产品

8、日程等于承诺

- 日程安排只是估计而已，并不是预言，但是有些出资人会将这个估计视为承诺
- 应对实践

- 当出资人要求得到承诺时，可以用信心水平与他们沟通
    - 比如90%的信心8-1号发布，100%的信心10-1号发布
- 交付日期渐进法
    - 下半年交付 -> 某个季度 -> 某个月 -> 某天
- 使用有时间盒限制的迭代，同时使用按优先级排序的待办事项列表
    - 迭代周期2-4周
    - 每个迭代的成果都可以投入使用
    - 可以应对任何时间的发布要求
    - 项目经理需要从出资人那得到承诺，需要说明哪个需求在何时需要

9、到了之后，我们会知道身处何方

- 出资人不停的改变项目的目标，当管理层无法或不想决定产品的方向时，这个游戏就发生了
- 应对实践
    - 确定已有的项目愿景、项目目标和发布条件，并就其达成一致
    - 如果出资人不愿意定义愿景，那项目经理就承担起这个责任，完成之后把它公布出来，并坚守这个愿景；如果做不到，就赶紧结束这个项目，再用新的目标启动新的项目
    - 用迭代来组织项目，如果已经达成足够多的目标，就结束这个项目，再启动新的项目
    - 不要允许别人让你的项目失去方向，否则只能发现自己陷入一片迷茫

10、日程安排工具总是对的（梦幻时间日程）

- 出资人希望在项目的第一周就看到甘特图，然后一切按部就班进行（关键路径永远保持不变）
- 原因：决策层看到的大部分报表都是由已完成的工作构成，而项目日程时对工作未来进程的猜测
- 应对实践
    - 制订波浪式规划
    - 使用低技术含量的日程安排技术，比如黄色便利贴
    - 提供带有信心水平的估算代替甘特图
    - 使用有时间盒限制的迭代，而且每次只规划一个迭代能做的工作
    - 用速度图表测量团队的开发速度

11、我们必须拥有这个功能，否则就完蛋了

- 出资人提出新的急迫的、重要的额外新功能，项目经理和团队也同意在当前版本加入；只是额外新功能的加入，会导致项目失去按日程交付的希望
- 应对实践
    - 要求改变功能集合
        - 看看哪些功能是这个版本不需要的
    - 要求更多的时间
        - 如果你愿意推迟交付时间，我们可以考虑加入这个功能

- 如果按功能逐个实现，而且按期提供可供发布的产品，项目经理就可以和出资人重新安排下个版本要实现功能的优先级，使用不超过4周的迭代，人们最久只需要等8周就可以得到新功能

12、我们不能说“不”

- 当你开始更团队成员讨论这个新功能带来的影响时，大家都不愿意说“不”，大家两眼一闭，就把新工作接下来了，没有去评估对当前工作的影响
- 应对实践
  - 询问团队成员，看他们能否针对新加的额外功能制订计划
  - 如果项目团队说，“这次就忍了，会加班完成”
    - 可以考虑加班一周，看看工作效率如何，疲劳程度怎样，缺陷率有无提高
  - 有时可以加入额外的人力来做更多的工作（不一定总好使）
    - 如果组织中有人具备领域专业知识，而且团队也希望有这些人加入，可以考虑加入
    - 不要加入对产品或团队不了解的人
  - 如果以上方法都不起作用，项目经理就要帮助团队说“不”
    - 用数据来抵消团队的内疚或是反驳完成公司要求的渴望
    - 速度图表和迭代内容图表在这里特别有用
  - 如果项目经理不能帮助团队学会说“不”，那大家就踏上了死亡征途

13、日程小鸡

- 项目经理问大家进展情况如何，每个人都说自己在按日程安排进行，而实际情况却是没有一个人做到
- 应对实践
  - 避免逐个进行的进度报告会议
    - 每日站立会议
    - 每周一对一会议
    - 每周的电子邮件进度报告
  - 将任务分解成更小的部分，这样每人每天都能交付一、两小块任务
  - 按功能逐个实现
    - 如果有更多的人把精力集中在一小块可交付的功能上，就可以尽快完成这个功能，而且人们也更容易看到自己取得的进展
  - 考虑使用短期迭代
    - 就不必坐下来开每周的小组项目进度会议了
    - 有了每日站立会议，人们就不能掩饰自己的真实进度了

14、90%完成状态

- 很多技术工作者对估算并不在行，在任何情况下，但有成员以为自己完成了90%的任务，而实际还有90%的工作尚未完成时，“90%完成状态”就发生了
- 应对实践

- 帮助大家定义出自己的“小石子”
    - 可以跟对方坐在一起，然后提问：“要完成这项任务，你需要多久？这周要处理的细分任务都有哪些？”
- 要让大家把自己的工作进度展示给你
    - 代码中所有的覆盖情况
    - 风险列表
    - 测试用例
- 教给大家如何跟踪自己的估算

15、我们马上会变得更快

- 项目经理一直在测量团队的开发速度，可是项目进展速度没有达到预期，由于某些原因，团队成员对于如期交付很乐观
- 应对实践
    - 与项目团队成员讨论紧张速度
        - 收集数据，什么原因让他们认为接下来可以更快
    - 测量估算质量因子
        - 每隔一段时间问团队成员“你们认为项目什么时候可以完成”，团队就预期时间达成一致，记录这个时间
    - 测量团队所做的每一件事，确定每个人都全力投入到项目中
    - 如果项目涉及到硬件组件、要测量其挣值
        - 计划值(PV，Plan Value)，又叫计划工作量的预算费用(BCWS，Budgeted Cost for Work Scheduled)。是指项目实施过程中某阶段计划要求完成的工作量所需的预算工时（或费用）。
            - 计算公式是：`PV=BCWS=计划工作量*[预算定额]`
            - PV主要反映进度计划应当完成的工作量，而不是反映应消耗的工时或费用。
        - 实际成本（AC，Actual Cost），又叫已完成工作量的实际费用（ACWP，Actual Cost for Work Performed）。指项目实施过程中某阶段实际完成的工作量所消耗的工时（或费用）。主要反映项目执行的实际消耗指标。
        - 挣值（EV，Earned Value），又叫已完成工作量的预算成本（BCWP，Budgeted Cost for Work Performed）。指项目实施过程中某阶段实际完成工作量及按预算定额计算出来的工时（或费用）。
            - 计算公式是：`EV=BCWP=已完成工作量*预算定额`
        - 进度绩效指标（Schedule Performed Index）。指项目挣值与计划值之比。
            - 计算公式是：`SPI=EV/PV=BCWP/[BCWS]`
            - 当SPI>1时，表示进度超前
            - 当SPI=1时，表示实际进度与计划进度相同
            - 当SPI<1时，表示进度延误

16、令人恍惚的日程

- 下周二，总部的大人物就要来视察了，或10个星期后就需要在展会上做演示，不管怎样，你要面对一个不可改变的日期，一系列充满也行或是不可能实现的功能集合
- 应对实践
    - 首先创建项目的仪表板（dashboard详见11章）
    - 测量进度
    - 测量速度
    - 如果还没有使用迭代式开发，把剩下的项目分解成迭代，迭代时间越短越好
        - 每个迭代要完成某个功能或一组功能
        - 包括功能的开发、文档、测试，以及其他产品对于“完成”的要求
    - 在一个迭代中要集中精力
        - 不要让团队成员受其他项目、未来工作或是技术债务干扰
    - 按功能逐个实现
        - 这样可能会产生不完整的系统架构，不过不必担心，如果一个功能需要架构提供某些方面的支持，团队会去实现的
        - 如果功能上不需要，那就没人会用它

17、小结

- 日程安排游戏总会发生，项目经理的工作就是要识别它们，然后管理项目，让项目仍然可以取得成功
- 绝大多数情况下，人们玩这些游戏都不是出于恶意
- 即使没有恶意，日程安排游戏还是会拖项目后退，使其停滞不前
- 如果能够以不超过4周的时间来组织项目，按功能逐个实现，随进度集成，并测量进展速度，就可以完全避免绝大多数日程安排游戏

七、创建出色的项目团队

项目经理要创建一个出色的项目团队，需要两个步骤，首先要招聘或引进所有需要的人才，其次要发挥他们的能力，一起工作，成为高效的团队

1、招募需要的人

- 团队中应该有下面这些角色

名称	项目中的角色
架构师	组织并指导整个系统的开发，包括测试系统
开发人员	设计、编写产品代码
测试人员	设计、编写测试，包括测试代码
技术文案	设计、编写产品文档
业务分析师	收集、编写需求
发布工程师	设计、编写、维护系统，包括与系统相关的任何脚本
项目经理	组织项目工作

- 项目可能还需要其他角色（比如UI设计师、固件开发人员），项目经理应该知道项目中的技术风险何在，以确定哪些角色是必需的
- 如果项目经理没有吸引、招募或淘汰团队成员的权利，那失败几乎无可避免
- 小心只会用PPT的架构师
    - 不是每个项目都需要架构师
    - 如果架构师喜欢画各种设计图，不愿意写代码，开发人员想知道如何把设计结果融入到现有结果中，他也无法给出实质回答，那此人不算是一名真正的架构师
    - 架构师就像一只海鸥，突然出现，扔下一堆用PPT展示的不知所云的架构图，就像海鸥排下的粪便，然后就迅速离开了
- 项目的风险越大，团队的多样性就应该越丰富

2、形成团队凝聚力

- 帮助形成一体的最佳方式，就是让他们一起工作，而不是去参加什么团队拓展活动
- 如果大家有同一目标，彼此承诺完成相互依赖的任务，而采取商定好的工作方式，这就是一个团队
- 如果希望团队凝结在一起，那么就帮助他们制订一些只有共同工作才能完成的短期目标吧