代码之美 Eric Brechner著 林锋 译

第1章 项目管理失当

敏捷开发
传统型、长期战略性规划

- 团队或功能层次 10人左右
- 项目层次 50～5000人有特定版本
- 产品层次 由高级主管领导的多个版本开发

项目经理 Program Manager PM

- 最终用户体验
- 跟踪所有项目的整体进度

里氏震级评估

时间安排

- 困难 花费整个里程碑时间 6～12周
- 中等 2～3周
- 容易 2～3天

必须有的功能 放在第一个里程碑完成
最好有的功能 放在第二个里程碑完成
希望有的功能 放在第三个里程碑完成

风险管理

特别注意

① 过度劳累的员工
② 匆忙实现，质量很差
③ 花费几周时间且动用2～3位甚至更多的高级开发人员去解决一个棘手的问题

如果开发人员是围绕"功能交付日期"付出大量的努力，而不是帮助你在产品的关键功能上降低风险，那么真有可能浪费时间了。

合作开发，要依赖其它组开发的功能

- 提高优先级
- 沟通渠道
- 独立测试

项目激励

① 里氏震级估计
② 贴住里程碑日期 放宽功能交付日期
③ 解释清楚哪些功能是必须要有，必须优先完成

分诊 triage （未完成工作条目、Bug或设计变更）如何解决冲突

源于战地医学 ① 医疗 进
② 不医疗 不进
③ 医疗 不进

分诊黄金法则

① 关上门
② 所有的决定都是团队的决定
③ 每个专职领域只派一名代表
④ 指定一个可以做最终决定的人 一般是PM
⑤ 所有的决定都应遵从"黄格令"信条
遵从相同信条的一类人

项目死亡行军（在太少的时间要做太多的事）

出现原因 ①管理层神经太大条 没考虑后果

②管理层天真得难以置信

每天至少多工作4小时，每周多2个工作日

不会使生产力翻倍的，人是非线性的

③管理层不切实际

不明白速度快过一头牛（很难）与

快过子弹（不可能）之间的区别

④管理层没头脑，不负责任

⑤管理层都是些毫无责任感的懦夫

死亡行军 项目失败概率极高，极易破坏团队

失败的原因 ① 一开始就注定失败

② 总勇大家走捷径

③ 没有太多的时间去思考

④ 没有太多的时间去沟通

⑤ 制造紧张、压力和机能障碍

⑥ 士气受挫，动力受损

⑦ 信心在进程中消失

⑧ 不妥善解决问题

⑨ 降低你的标准

按计划行事

① 遇到问题，须及时反馈，保证透明，可控

② 不能因为匆忙赶工，牺牲质量

- 兑现承诺
- 风险控制

为什么会易搞混 ① 它们通常同时发生

② 两者都有时间表

③ 它们都可称为计划

④ 人们所告诫的往往只是承诺

⑤ 风险管理往往是自我学习的过程，是非正式的

第2章

过程改进，没有灵丹妙药

六西格玛DMAIC流程
- 界定 define
- 测量 measure
- 分析 analyze
- 改进 improve
- 控制 control

软件开发方法要适中，不能沉迷：
- 极限编程 eXtreme Programming, XP
- 敏捷方法
- 团队软件进程 Team Software Process, TSP

精益
轻型汽车概念 lean
- 拉模式
- 持续改进

不需要，就不做

有损于客户价值流的浪费
- 过量生产
- 运输
- 多余动作
- 等待
- 进程不当
- 库存
- 缺陷

毛家豪

Source Depot

测试驱动开发

软件度量
1. 不要想着怎么度量，而是度量什么
2. 不要对中间环节进行过多度量分析，而只对关键型结果进行度量分析
3. 不要只是收集数据而已，使用度量分析解决关键问题
4. 不要使用度量分析做决定，而是通过度量分析使你明白需要做出个决定
5. 不要只对原生度量进行比较，要有一些基准及范例，它们提供了参照细节

Bug剖析
- 标题 简略
- 指派
- 重现步骤 期望、事实
- 优先级别 0~3 0最高
- 严重程度 和优先级别独立
- 解决方案

第3章 根除低下的效率

规范变更请求 (Spec change Request, SCR)

设计变更请求 (Design change Request, DCR)

零Bug反弹 (Zero Bug Bounce, ZBB)

开发人员应该做的
- 分析Bug
- 为新门开发一些工具
- 讨好项目经理，把他们的设计思想变成原型程序
- 学习新技术或技能
- 跟研究人员交谈
- 撰写专利申明或白皮书
- 反思职业生涯

开会　为什么我们会在这里

我们正在做什么

我们是要做出一个决定吗？

我们是要共享信息吗？(比如一个状态报告会)

我们是要收集想法吗？

为什么这些人会在这里

试图做一个决定？

状态汇报会议？

头脑风暴会议？

为什么我现在才听到这个

接下来要做什么

会议纪要 Email 中：所有参与人员都在地址栏，受到影响的人需要被抄送。写明会议的决定、共享的信息或收集到的想法的简单总结。列出接下去的安排，指明谁在什么时候做什么事情

第4章 跨越工种

前期消除Bug的方法
- 单元测试 Unit Testing
- 设计 Design
- 设计复审 Design Review
- 代码复审 Code Review
- 代码分析 Code Analysis
  (代码分析可以用PREfast工具)

受过自由教育的人相信规则就是规则
受过自由教育的人尊重权威
受过自由教育的人不折腾
受过自由教育的人设想万事皆易
受过自由教育的人不关心细节
受过自由教育的人不关心纯度
受过自由教育的人关注感受和仪表

第5章 软件质量不是梦

《Writing Secure Code》

第6章 有时间就做软件设计

Segall 定律 有一块手表的人知道时间，而有两块手表的人对时间总是不太确定

软件设计维度

	外部	内部
静态	项目管理规范书 应用程序编程接口定义	开发规范书 测试驱动开发
动态	用例 应用范例和角色扮演	顺序图 状态图、流程图、 威胁和故障建模

第7章 职业生涯历险记

每个人都要明白自己的局限性

第8章 自我完善

工作与生活不平衡的情况

① 总是把工作放在首位
② 对工作有不同的价值观
③ 总是把工作与家庭分别对待

平衡工作与生活

①理解并接受你的生活方式
②跟你的管理者商定基本的规则
③不要那么快妥协
④需要的话可以用RAS、远程桌面或OWA
⑤抛开造成分离的精神分裂伪装

第9章 成为管理者，而不是那些的化身

优秀开发人员的特质

- 他们知道他们正在做什么
- 他们不相信魔法
- 他们了解客户业务
- 他们把客户和团队放在优先于自己的位置
- 他们有不妥协的精神和道德规范
- 他们有杰出的人际沟通技能
- 他们有一个广泛的支持网络

第10章 微软，你会喜欢它的

通过软件赋予每个人以及每台设备以力量
持续不断地改进产品
持续不断地提高自己

End

2018.2.13