软件开发最佳实践

Hao Wang

软件开发最佳实践

Hao Wang

This book is for sale at http://leanpub.com/software-development-best-practices

This version was published on 2014-09-29

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

前言

成功学

现在市场上充满了教人如何成功的书籍，而大部分只能算作理智的文字而已，素补上成功学。难道这一本见软件技术的书会是成功学？是的，这本书的出发点其实就是成功学。所谓最佳实践，何为最佳，这个最佳的标准是什么，就是在真是世界把事情做成，事实上的成功才是实践的最佳。为什么这个章节的内容会包括在一本技术书中，为什么那个章节会提及那些内容？基本上都是以成功学为出发点。

成功的定义。外延和内涵的明确定义有助于避免无意义的争论，了解本书的范畴。把事情做成和成功学是不一样的。记得我读书的时候，每个学期开学，老师都会要求学生写个学期学习计划，而这个计划内容无非就是“每周写一篇周记”，“每天如何如何”。这样的计划也许很容易完成，也许很难达到，但是我不把这样的目标纳入成功学的范畴。某种程度的，我把这种计划看的毫无意义。不是说做这个事情本身无意义，而是扯上成功学显得无意义。一、这一个计划定下的目标是非常形式的，空洞或无物；二、这样目标其实是非常容易到达的，从而导致一个非常有害的结果：虚假的成功感。

有时候，混沌的我们有一天会突然觉悟，也会定下这样一些目标，比如：这个学期一定要进前十名，或者更为敏捷一点的目标，这个学期一定要进前五名。进入社会以后，我们有时也会励志地订下一些目标：争取半年把这个技术掌握，把这本书看完，甚至定下目标，半年之内一定要换一个工资更高的工作。仍然，我不把这样的目标纳入成功学的范畴。虽然，这样的目标已经非常具体而充实了，克服上一类目标的第一个问题；而且，也不是那么容易不做努力就能够达到的，真正努力并完成这样的任务，对个人提高也很大，克服了上一类目标的第二个问题。这类目标也是一般励志的出发点，，但这不是我想涉及的内容。这类成功虽然也有点难度，但不是那么难；虽然也有成就感，但不是那么大。

生活中，我们有时会碰到两难悖论的情景，左也不行，右也不行，进也不是，退也不是。比如，刚毕业找工作，就往往遇到这种情况：几乎所有的公司都只招聘有工作经验的人。作为应届毕业生就处于两难的悖论：我要应聘这份工作，我就必须有经验；你不聘我，那我的工作经验从哪里来呢？

如此种种，这类目标与前面两类最大的不同就是，这些都是你不曾经历甚至未曾想象过的事情。用《从优秀到卓越》这本书中的术语，这是一种“跨越”，从优秀到卓越的一种跨越。没有比这更为艰难的、比这更有成就感的、比这更有意义的成功了。小到个人奋斗，达到人类文明进展，无不是有这类成功组成。

软件开发难道是一个未知的事情？是的，至少目前以致将来十年，软件开发技术本身充满了未知和不定。有这么一个简单的事实，西方国家几乎所有其他行业都有工程师认证，如电气、建筑等等，唯独没有软件工程师认证。为什么？所谓工程，即有一套标准，有规律可遵循，而软件没有工程师认证，是因为软件还未能“工程化”！现在所谓的极限编程，敏

捷开发的种种兴起，其实就是业界对软件开发工程化的探索和推进。我们活在软件开发的“乱世”，这是一个大背景！

另一方面，即使软件开发技术本身未定成熟以后，软件本身的特殊性也会跳出来，干扰我们的工程化。软件是什么？软件是思维的固化，是业务知识的固化。为什么强调业务知识呢？让我们来慢慢梳理一下软件的本质吧，澄清一些误区。

软件的视线来源于需求，而其实需求有两种：一种是技术需求，另一种是业务需求。前面说软件工程化及乱世，指的就是技术需求实现的标准化。所谓的通用框架、开发库都是按这条线来走的。而业务需求则永远是不同的。我知道肯定有人反对这种说法。仔细想想，软件是业务知识的固化，如果相同的业务知识需要不同的软件实现呢？所谓软件蓝领的定位完全是一个错误！蓝领就是重复劳动，重复是软件本身最擅长的，为什么舍近求远呢？

正是因为有业务需求的存在，每一次的软件开发都是一个全新的历程，一个需要成功学的过程。回过头来，我们再看看敏捷的原则和提法，放在这个背景之下才能理解。

针对软件的以上两个层面，我使用敏捷也有两个层面：一是，在项目开发工程中，我们使用敏捷；二是，在团队建设和软件开发流程改进上，我们也使用敏捷的方式。而这本书，就是这两种深度敏捷使用的总结。

微观一致性的宏观效应

这个标题是有点拗口，举两个例子就明白了。石墨和金刚钻都是有碳元素组成，然而一个柔软，另一个却无比坚硬，为什么？仅仅因为金刚钻的碳在微观结构上拍了更整齐更一致。纳米技术，则是人类主动在微观上对物质进行的一致性改造，成就了完全类似金刚钻的惊人效应。如不粘油水的衣服等。

管理

> 领导是门艺术，管理是门技术
>
> 如果你不能一天把事情做完，就需要用到管理技术－－－长时间，大任务

任务分解

甘特图是项目管理的一个经典技术。这里无意冒犯它，只想把这个超能量的巨人限制在一个小笼子里。

任何任务的层次不要超过三层。当前并行任务不要超过 7 个。7 个，为什么是 7 个？因为对于一个普通人来说，超过 7 个就已经是抽象了（后面还会详细讲解抽象）。比尔盖茨说，当他的资产超过一个亿的时候，这个资产就只是个数字了。他所说的意思，翻译成我这里的术语：资产超过一个亿，就只有抽象的意义了。

当你的面前摆着一个苹果时，你知道你有苹果可吃；当你面前摆着两个苹果时，你能感觉到，吃完一个，还有一个：……；当面前摆了 7 个苹果，你就只能感觉到，哇，我有很多苹果可以吃。7 被“很多”取代，这就是我说的抽象的内涵。

很多的苹果当然会让你高兴，只是不是烂苹果。但当面前是很多任务的时候呢？带来的恐怕是压力和恐惧吧！

这不可能，如果现实项目中就有很多任务，怎么办？很简单，挑出五六个，剩下的推倒一边（Backlog），我强调的是当前 (Sprint 冲刺) 的任务量。

- 减少任务之间的依赖
- 尽量宏观依赖，减少微观依赖
- 任务划分：纵向划分，而不是横向划分－－－任务目标可视化
  - 横向划分是指按技术领域划分
  - 纵向划分是指需求功能划分
  - 全栈式开发人员－－－精英开发理念
- 任务实施：横向实施需要所有权的理念解决技术短板问题－－－任务的所有者不一定是是代码的实现者
- 任务实施基本粒子：单元测试（撰写＝》实现＝》通过）不可打断，可以用番茄时间的概念支持

需求追踪

DDD 本质论

代码与业务之间的鸿沟

正名

> 名不正，则言不顺；言不顺，则事不成

DDD 框架与 DDD

> 固定成本和可变成本

和任何技术一样，软件开发技术无非要解决的问题是：把尽量多的可变成本纳入固定成本的范畴。

DDD 框架关注技术需求

作为固定成本的 DDD 框架有这些特点：大投入，跨项目，一致性，不变性。其产品是以开发人员和开发团队为用户对象

DDD 关注业务需求

作为可变成本的 DDD 本身：业务需求是