所谓软件工程

所谓软件工程

很多编程的人包括我，头衔叫做“软件工程师”（software engineer），然而我却不喜欢这个名字。我喜欢把自己叫做“程序员”（programmer）或者“计算机科学家”（computer scientist）。这是为什么呢？这需要从“软件工程”（software engineering）在现实中的涵义谈起。

有人把软件工程领域的本质总结为：“How to program if you cannot？”（如果你不会编程，那么你如何编程？）我觉得这句话说得很好，因为我发现软件工程这整个领域，基本就是吹牛扯淡卖“减肥药”的。软件行业的大部分莫名其妙的愚昧行为，很多是由所谓“软件工程专家”发明的。总有人提出一套套的所谓“方法论”或者“原则”，比如 Extreme Programming，Design Patterns，Agile，Pair Programming，Test Driven Development（TDD），DRY principle，…… 他们把这些所谓方法论兜售给各个软件公司，鼓吹它们的各种好处，说使用这些方法，就可以用一些平庸的“软件工程师”，制造出高质量低成本的软件。这就跟减肥药的广告一样：不用运动，不用节食，一个星期瘦 20 斤。你开头还不以为然，觉得这些肤浅的说法能造成什么影响。结果久而久之，这些所谓“方法论”和“原则”成为了整个行业的教条，造成了文化大革命一样的风气。违反这些教条的人，必然被当成菜鸟一样的鄙视，当成小学生一样的教育，当成“反革命”一样的批斗。就算你技术比这些教条的提出者还高明不知道多少倍也一样。

打破这些软件工程专家们制造的幻觉的一个办法，就是实地去看看这些所谓专家们自己用这些方法论做出了什么好东西。你会惊奇的发现，这些提出各种玄乎其玄的新名词的所谓“专家”，几乎都是从不知道什么旮旯里冒出来的民科，没有一个做出过什么有技术含量的东西，他们根本没有资格对别人编程的方式做出指导。这些人做出来少数有点用的东西（比如 JUnit），其实非常容易，以至于每个初学编程的人都应该做得出来。可世界上就是有这样划算的职业，你虽然写不出好的代码，你对计算原理的理解非常肤浅，却可以通过一些手段，得到评价别人的“代码质量”的权力，占据软件公司的管理层位置。久而久之，别人还以为你是什么泰斗。你仔细看过提出 Java Design Pattern 的四个人（GoF），到底做出过什么厉害的东西吗？没有。提出“DRY Principle”的作者，做出过什么好东西吗？没有。再看看 Agile，Pair Programming，TDD……的提出者？全都是一群饭桶。他们其实根本就不懂很多编程的东西，写出文章和书来也是极其肤浅，一知半解。

所谓“软件工程”，并不像土木工程，机械工程，电机工程，是建立在实际的，科学的基础上的。跟这些“硬工程”不一样，软件弄得不好不会出人命，也不会跟做芯片的公司那样，出一个bug立即导致上亿的损失，身败名裂。所以研究软件工程，似乎特别容易钻空子，失败了之后容易找借口和替罪羊。如果你说我的方法不好，你有什么证据吗？口说无凭，我浪费了你多少时间呢？你的具体执行是不是完全照我说的来的呢？你肯定有什么细节没按我说的做，所以才会失败。总之，如果你用了我的办法不管用，那是你自己的问题！

想起这些借口我就想起一个笑话：两夫妻睡觉发现床上有跳蚤，身上被咬了好多大包。去买了号称“杀伤率100%”的跳蚤药，撒了好多在床上。第二天早上起来，发现又被咬了好多新的大包。妻子责怪丈夫，说他没看说明书就乱撒。结果丈夫打开说明书一看，内容如下：

> 本跳蚤药使用方法：
>
> 1. 抓住跳蚤
> 2. 掰开跳蚤的嘴
> 3. 把药塞进跳蚤嘴里
> 4. 合上跳蚤的嘴

我发现很多软件工程的所谓方法论失败之后的借口，跟这跳蚤药的说明书很像 :)

人都想省钱，雇用高质量的程序员不容易呀，所以很多公司还是上钩了。他们请这些“软件工程专家”来到公司，推行各种各样的软件方法论，可是发现最后都失败了。这是为什么呢？因为再高明的方法论，也无法代替真正的，精华的计算机科学教育。直到今天还有很多公司推行所谓的Agile，煞有介事的搞一些 stand-up meeting, scrum 之类的形式主义东西，以为这些过家家似的做法就能提高开发质量和效率。很多开发人员也很把一些软件工程的工具当回事，喜欢折腾 Git，Maven 等工具一些偏僻的“新功能”。他们很在乎所谓的版本控制，测试等东西，以为熟练的掌握这些就能开发出高质量，可靠的代码。可是你最后发现，无论你如何高效的使用这些工具，它们都只能起到辅助的，次要的作用。编程工具永远不是程序本身，对编程工具的熟练掌握，永远也无法代替真正的对程序和计算的理解。过分强调这些工具的使用，是本末倒置的，让工程走上失败道路的作法。

编程真的是一门艺术，它完全符合艺术的各种特征，编程界也充满了艺术界的独有特征。有些初学艺术的人（比如10

年前的我），总是挑剔手上的工具，非要用最新最炫的工具，用它们最偏僻最难用的“特性”，才觉得自己能够做出优秀的作品。很多人照不出好的照片，就怪相机不好。买了几万块钱的笨重高档相机，照出来的照片还不如别人用手机照的。这些人不明白，好的摄影师和不好的摄影师，区别在于眼睛，而不是相机。一个真正的艺术家，可以用任何在手上的工具创造出色的作品。有些甚至可以用一些废品垃圾，拙劣的工具，做出杰出的，别具风味的艺术品。因为艺术存在于人的心里，而不在他们使用的工具里面。