# 一：架构师期刊

1：，在大部分的情况下，我们是不需要大教堂的，现实中为了整体环境，我们宁愿在大教堂中祈祷，但是程序设计中，我们不会因为喜欢一棵树木，而买下整块山头，我们几乎不喜欢任何看起来多余的部分。同时我们也不需要小集市，我们需要的是一个品质优良的商场。开发者是采购者，采购模块的过程既是架构的设计的过程

2：做文字功夫，大约三分靠灵感，五分靠学识，剩下的两分，就得留给才气来炫耀，方才能作得一篇好文。若论技术文章的写作，似乎又不然，没有技术功底，文字再好也白搭。可若是没有足够的语文学识，文字又干瘪瘪的，不耐看。若硬要留下那两分才气，余下的八分就留给学识与技术来分配好了。其实，技术文章没有别的要求，就是要把道理讲通透，讲明白，让读者能够看得懂。技术编辑不需要玩花活儿，不一定要把文章写得花团锦簇，关键还是看你对这门技术掌握得有多透彻，再转而以平实的语句描述出来，足矣。作文与设计一样，真正的大美，还在于一个“简”字。

董桥在《无灯无月何妨》中借一位老先生的话提点作文之道，曰“人贵饱学，文贵简练，古今中外大方之家都靠这两笔本钱讨生活。”道理平实，做起来其实不易。我在InfoQ讨生活算起来已有三年春秋，似乎无功无过，但求一个心安。小编的理想不是要改变IT世界，也不敢奢望像鲁迅那样希望通过文字唤醒国人的灵魂，无非就是一个知识的传递者，攒出文字让人吸收一点儿养分。我之所谓心安，其实就是一个道德底线。我们这种知识的传递者，就好比餐厅的厨师，把菜炒熟，是你的本分，把菜炒好，是你的本事，不在菜里放地沟油，那就是求得心安了。可惜，很多人连这点心安也不能做到。

不过，一个人的学识总是有限。从前看很多技术著作，作者或译者总会在后面写上“囿于学识，本书或有疏漏，敬请读者见谅”云云。我以为是作译者谦虚，等到自己成了作者和译者，才能体谅那种战战兢兢。于是，自己也学到这个套路，算是一种免责声明。然则，免责声明不是免死金牌，不要因为有了这几个大字做挡箭牌，就可以为所欲为了。那么，该如何衡量这个度？无它，还是一个心安。搜索这几年自己在InfoQ发表的新闻文章，有翻译有原创，有洋洋洒洒上万字的大文章，也有豆腐干一般的简短小文，也有采访——采访别人，又被别人采访。若要算字数，可能也有十余万字了吧。这些文字，这些观点，一定会有疏漏，一定会有错误，但却一定是在那个时刻，以我的学识我的功底我的精力我的能力所能做到最好的。

做个心安的编辑，我觉得已经是我最大的成就了。

3：

# 二：代码之殇

1：一定要破除“功能交付日期”的神话，因为开发人员专注于这种日期的时候会破坏风险管理。真正要关心的日期只能是“项目日期”，比如各个里程碑、测试版，等等，而绝不应该是“功能交付日期”。项目日期之间一般都有较长时间的间隔，而且这种日期不会很多。管理这几个日期要容易得多。如果要求开发人员在某个日期之前一定要实现某个功能，当他们不能按时完成时他们往往不会告诉你，而是对你说“我正在加紧做……我会加班……”之类的话。

在软件开发过程中进行风险管理，我们还要特别注意以下几个因素：一个是过度劳累的员工，一个是匆匆忙忙实现的、质量很差的功能，再一个就是你花费几周的时间且动用2～3 位甚至更多的高级开发人员去解决一个棘手的问题。如果你

的开发人员是在围绕“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。

2：**软件工程绝对是含糊的**

我对关于不能也不应该对一个功能的开发做时间安排的论断表示怀疑。文中精确

地论述了“编码”活动。遗憾的是，这是初中生干的事情——拼凑一个VB 程序来

解密信息、相互通信。我们可是软件工程师啊，不是电脑苦工。

——一个充满怀疑的无知者

我经常听到这种说法，但请就此打住。银行经理并不管理银行，软件工程师也不

在软件上做工程。他们开发软件，定制软件，通常事无巨细、从头至尾参与其中，

并不需要了解操作范围、公差、故障率、压力条件等度量标准。的确，我们的系

统有这些标准，但这些标准不是为软件编码准备的。

我曾到一个工程学校进修过。我的朋友当中也有很多是电力、基建、航空、机械

等方面的工程师。这些工程师做的项目，其构造模块和结构流程都经过了很好的

定义和提炼，而且都是可预测的。虽然有时候为了达到客户的要求需要一个合适

的设计，但只要用一种新颖的方法把各个模块组合在一起就很有创造性了，就算

是最标新立异的建筑也会符合一定的公差要求，并且具有严格的可控质量和功能。

但对软件开发来说，情况就不一样了，尽管很多人竭力想让这两者达成一致。软

件的各个构造模块太底层了，变数太多。它们之间的交互影响太难预料了。像

Windows、Office、Visual Studio 及MSN 等大型软件系统的复杂度，已经远远

超过了工程的正常范围，以致哪怕只在这些系统中做微小的功能改动，也无法粗

略估计出这些改动所引起的“平均失效时间”。

因此无论好坏，还是抛开痴心妄想和崇高理想，回到现实中来吧！我们必须承认，

我们是开发者，而不是工程师。我们不能奢望轻易得到传统的工程领域积累了成

百上千年的经验才做到的“可预测性”。这无异于我们奢望：不用跟电脑说什么，

电脑就能按照我们心里的想法去做事。我们还办不到！

**3:相信一半你看到的，别信你听到的**

如果我在某个功能或者一段代码上依赖于另外一个团队或产品组，我肯定不想听

到像“你要的东西应该可以在这个里程碑期间内完成”这样的说法。我需要一个很

具体的交付日期。我要有具体细节。

——一个需要日期的人

我想写几个关于依赖关系和组件团队的栏目，也许将来会吧，但眼下我只想讨论

依赖方的开发时间表。首先，假设你的依赖方确实有一份开发时间表，你会相信

它吗？你也许会说：“当然要信，我有其他选择吗？”建议在你的胃病恶化之前赶

紧吃一点PepcidTM（一种胃酸抑制剂）。不光只是开发时间表，不要相信依赖

方所说的任何事情。如果他们坐在隔壁房间，他们告诉你外面正在下雨，你首先

要做的是到自己的窗口去看一下。

但我并不是说你不能跟依赖方合作。相反，你应当与他们很好地合作，因为依赖

方可能为你的团队、产品和客户带来大量的经验和意外的收获。我只是告诫你要

高度警惕当前正在发生的事情。要向他们提出定期多次交付的要求，并对交付的

东西进行自动化测试。获取他们对RAID 进行读和写的RDQ，观察它们的数量

以及存在问题的地方。派你的项目经理去参加他们的分诊会议。加入他们的邮件

列表。

作者注：查一下本书最后的“术语表”，以便理解这些用于Bug 跟踪的词汇。

基本上，你需要像鹰一样盯着依赖方。他们是你的团队和产品的一个扩充。你跟

他们接触、沟通得越多，你在规避其短处以及促进其改变方面的能力就越强。至

于他们承诺的功能什么时候能够完成，你必须依赖你在如下3 个方面上的影响力：

提高优先级，沟通渠道，独立测试（为了知道他们的功能是否真正可用了）。

4：**激励：不能光靠比萨和啤酒**

总的来说，你的观点用在项目早期的计划上还行，但对于产品发布前的最后一个

里程碑就不那么合适了。时间表怎样提供最后期限和时间约束，让团队遵照执行，

使得它能作为一种日常管理工具去激励团队的执行力？你必须要解决诸如此类

的问题。

——一个找不到（汽车）油门的人

首先我要重申：如果你坚持让开发人员遵从“功能交付日期”，那他们为了准时交

付可能会撒谎。他们会隐瞒自己的工作状态，会在质量和完成度上给你虚假信息。

如果你不想你的开发团队这么对你的话，你必须建立起一个更好的激励机制。我

用过3 种不同的方法，这些方法能使大家互相协调工作并产生良好效果。

第一种，也是最基本的方法，就是应用里氏震级估计。我的开发人员知道，我期

望的是每个功能在大致那么多的时间内完成。如果一个原先估计需要2 周的任务

实际上花了2 周半，可能关系不大。但如果花的时间比原先估计的要长得多，那

么通常是有实实在在的原因的，那个开发人员必然会让我知道这个原因。如果缺

乏充分的理由去延期交付，则足以对开发人员形成一种鞭策。然而，因为没有卡

得很死的日期，大家几乎不会去想到隐瞒和欺骗。

第二种激励工具是瞄准里程碑日期。这有招致大家走捷径的危险，但总体的效果

是鼓励开发人员从一开始就努力工作，并且让他们对自己是否落后于进度做到心

里有数。“功能交付日期”和里程碑日期关键的不同在于，后者是给整个团队设置

的日期，需要整个团队一起努力去达到它。因此，个人抄近路的压力就会小很多。

然而，这种危险性仍然无法杜绝，逼得我使出最后也是最有效的一招。

作者注：一个自我导向的团队向着一个清晰的共同目标一起努力，这是很多

敏捷方法的核心概念，尽管在2001 年我还不知道有敏捷方法。

最后一种激励工具是迄今为止我使用起来最有效的。我向团队解释清楚哪些功能

是必须要有的，必须优先完成。我告诉他们，必要时任何其他的功能都可能被放

弃不做。遗憾的是，这些必须要有的功能常常做起来比较乏味，没有意思，甚至

不值得一提。因此我告诉我的团队，如果他们想要做那些很酷的功能，必须首先

保质保量地完成之前的这些关键功能。之后，再去做那些不那么关键却要炫得多

的东西。这种激励是积极的，有建设性的，并且非常有效。屡试不爽！

5：**在日期上沉沦**

继续前面的讨论：时间表在不同的功能单位之间（不只是开发，还有项目管理、

测试、用户体验、市场推广、外部合作）同步工作时，绝对是必要的；你还必须

要解决这个问题。

——一个出格的人

如果你确实需要具体的“功能交付日期”来同步各个方面和依赖方的工作，那么你

的软件永远也没法发布。当然，我们的软件一直在发布——我们甚至准时发布了

庞大的Office XP。要知道，它的发布日期是在两年之前计划的。因此，肯定存

在其他的一些关键因素。

其实真正起决定作用的，是要在顺序、成本和方法上达成一致，并且提供及时的

状态报告。各个方面之间要协商达成协议，定义好状态汇报的流程，并且避免工

作的相互牵制。

顺序：讨论协商各个功能实现的先后顺序已经不是什么新鲜事了，尽管有些

部门从来都不能对优先顺序达成一致。

成本：成本的协商通常发生在开发人员和项目经理之间（例如一个开发人员

说：“如果我们使用标准控件，可以帮你节省2 周的时间。”）但有时候就只是开

发人员做决定。其实，成本的协商也应该让测试和实施人员参与进来。

方法：讨论协商使用哪种方法通常在项目管理规范书中做，但很少在开发和

测试的规范书中做——这对他们不利。

状态汇报：至于状态的及时汇报，你务必要把邮件和测试发布文档（Test

Release Document，TRD）登记起来，或者两者任选其一，以便让项目经理、

测试和实施人员了解项目的进展情况。测试部门需要对阻碍工作继续进行的Bug

使用警报。项目经理采用类似于“规范书变更请求”（Spec Change Request，SCR）

的方式来汇报规范书的更改。（了解更多关于SCR 的内容，可阅读第3 章的“迟

到的规范书：现实生活还是先天不足”栏目。）

如果各个不同的部门能够对他们的工作顺序进行合理的安排，知道各自的工作需

要花费多少时间，对他们使用的方法也很有信心，并且保持着最新的状态报告，

那么项目就上轨道了！问题找到了，风险降低了，意外也很少发生。更重要的是，

没人顶着因为人为的交付日期而犯错的压力。相反，每个人都朝着同一个目标在

努力——为我们的客户交付一个令人愉快的体验。

6：