做好软件开发的75条守则

1. 你们的项目组使用源代码管理工具了么？

应该用。VSS、CVS、PVCS、ClearCase、CCC/Harvest、FireFly、SVN都可以。

2. 你们的项目组使用缺陷管理系统了么？

应该用。ClearQuest、BugZilla、MQC或JIRA。

3. 你们的测试组还在用Word写测试用例么？

不要用Word写测试用例（Test Case）。应该用一个专门的系统，可以是Test Manager，也可以是自己开发一个ASP.NET的小网站。主要目的是方便Track和Browse。

4. 你们的项目组有没有建立一个门户网站？

要有一个门户网站，用来放Contact Info、Baselined Schedule、News等等。推荐Sharepoint Portal Server 2003来实现，15分钟就搞定。买不起SPS 2003可以用WSS (Windows Sharepoint Service)。

5. 你们的项目组用了你能买到最好的工具么？

应该用尽量好的工具来工作。比如，应该用VS.NET而不是Notepad来写C#。用Notepad写程序多半只是一种炫耀。但也要考虑到经费，所以说是“你能买到最好的”。

6. 你们的程序员工作在安静的环境里么？

需要安静环境。这点极端重要，而且要保证每个人的空间大于一定面积。

7. 你们的员工每个人都有一部电话么？

需要每人一部电话。而且电话最好是带留言功能的。当然，上这么一套带留言电话系统开销不小。不过至少每人一部电话要有，千万别搞得经常有人站起来喊：“某某某电话”。《人件》里面就强烈谴责这种做法。

8. 你们每个人都知道出了问题应该找谁么？

应该知道。任何一个Feature至少都应该有一个Owner，当然，Owner可以继续Dispatch给其他人。

9. 你遇到过有人说“我以为…”么？

要消灭“我以为”。**Never assume anything**。

10. 你们的项目组中所有的人都坐在一起么？

需要。我反对Virtual Team，也反对Dev在美国、Test在中国这种开发方式。能坐在一起就最好坐在一起，好处多得不得了

11. 你们的进度表是否反映最新开发进展情况？

应该反映。但是，应该用Baseline的方法来管理进度表：维护一份稳定的Schedule，再维护一份最新更改。Baseline的方法也应该用于其它的Spec。Baseline是变更管理里面的一个重要手段。

12. 你们的工作量是先由每个人自己估算的么？

应该让每个人自己估算。**要从下而上估算工作量，而不是从上往下分派**。除非有其他原因，比如政治任务工期固定等。

13. 你们的开发人员从项目一开始就加班么？

不要这样。不要一开始就搞疲劳战。从项目一开始就加班，只能说明项目进度不合理。当然，一些对日软件外包必须天天加班，那属于剥削的范畴。

14. 你们的项目计划中Buffer Time是加在每个小任务后面的么？

不要。**Buffer Time加在每个小任务后面，很容易轻易的就被消耗掉。Buffer Time要整段的加在一个Milestone或者checkpoint前面**。

15. 值得再多花一些时间，从95%做到100%好值得，非常值得。

尤其当项目后期人困马乏的时候，要坚持。这会给产品带来质的区别。

16. 登记新缺陷时，是否写清了重现步骤？

要。这属于Dev和Test之间的沟通手段。面对面沟通需要，详细填写Repro Steps也需要。

17. 写新代码前会把已知缺陷解决么？

要。每个人的缺陷不能超过10个或15个，否则必须先解决老的bug才能继续写新代码。

18. 你们对缺陷的轻重缓急有事先的约定么？

必须有定义。Severity要分1、2、3，约定好：蓝屏和Data Lost算Sev 1，Function Error算Sev 2，界面上的算Sev 3。但这种约定可以根据产品质量现状适当进行调整。

19. 你们对意见不一的缺陷有三国会议么？

必须要有。要有一个明确的决策过程。这类似于CCB (Change Control Board)的概念。

20. 所有的缺陷都是由登记的人最后关闭的么？

**Bug应该由Opener关闭。Dev不能私自关闭Bug**。

21. 你们的程序员厌恶修改老的代码么？

厌恶是正常的。解决方法是组织Code Review，单独留出时间来。XP也是一个方法。

22. 你们项目组有Team Morale Activity么？

每个月都要搞一次，吃饭、唱歌、Outing、打球、开卡丁车等等，一定要有。不要剩这些钱。

23. 你们项目组有自己的Logo么？

要有自己的Logo。至少应该有自己的Codename。

24. 你们的员工有印有公司Logo的T-Shirt么？

要有。能增强归属感。当然，T-Shirt要做的好看一些，最好用80支的棉来做。别没穿几次就破破烂烂的。

25. 总经理至少每月参加次项目组会议要的。

要让team member觉得高层关注这个项目。

26. 你们是给每个Dev开一个分支么？

反对。Branch的管理以及Merge的工作量太大，而且容易出错。

27. 有人长期不Check-In代码么？

不可以。对大部分项目来说，最多两三天就应该Check-In。

28. 在Check-In代码时都填写注释了么？

要写的，至少一两句话，比如“解决了Bug No.225”。如果往高处拔，这也算做“配置审计”的一部分。

29. 有没有设定每天Check-In的最后期限？

要的，要明确Check-In Deadline。否则会Build Break。

30. 你们能把所有源码一下子编译成安装文件吗？

要的。这是每日编译（Daily Build）的基础。而且必须要能够做成自动的。

31. 你们的项目组做每日编译么？

当然要做。有三样东西是软件项目/产品开发必备的：**1. bug management; 2. source control; 3. daily build。**

32. 你们公司有没有积累一个项目风险列表？

要。Risk Inventory。否则，下个项目开始的时候，又只能拍脑袋分析Risk了。

33. 设计越简单越好越简单越好。

设计时候多一句话，将来可能就带来无穷无尽的烦恼。应该从一开始就勇敢的砍。这叫scope management。

34. 尽量利用现有的产品、技术、代码千万别什么东西都自己Coding。

BizTalk和Sharepoint就是最好的例子，有这两个作为基础，可以把起点提高很多。或者可以尽量多用现成的Control之类的。或者尽量用XML，而不是自己去Parse一个文本文件；尽量用RegExp，而不是自己从头操作字符串，等等等等。这就是“软件复用”的体现。

35. 你们会隔一段时间就停下来夯实代码么？

要。最好一个月左右一次。传言去年年初Windows组在Stevb的命令下停过一个月增强安全。

36. 你们的项目组每个人都写Daily Report么？

要写。五分钟就够了，写10句话左右，告诉自己小组的人今天我干了什么。一则为了沟通，二则鞭策自己（要是游手好闲一天，自己都会不好意思写的）。

37. 你们的项目经理会发出Weekly Report么？

要。也是为了沟通。内容包括目前进度，可能的风险，质量状况，各种工作的进展等。

38. 你们项目组是否至少每周全体开会一次？

要。一定要开会。程序员讨厌开会，但每个礼拜开会时间加起来至少应该有4小时。包括team meeting, spec review meeting, bug triage meeting。千万别大家闷头写code。

39. 你们项目组的会议、讨论都有记录么？

会前发meeting request和agenda，会中有人负责主持和记录，会后有人负责发meeting minutes，这都是effective meeting的要点。而且，每个会议都要形成agreements和action items。

40. 其他部门知道你们项目组在干什么么？

要发一些Newsflash给整个大组织。Show your team’s value。否则，当你坐在电梯里面，其他部门的人问：“你们在干嘛”，你回答“ABC项目”的时候，别人全然不知，那种感觉不太好。

41. 通过Email进行所有正式沟通

Email的好处是免得抵赖。但也要避免矫枉过正，最好的方法是**先用电话和当面说，然后Email来确认。**

42. 为项目组建立多个Mailing Group

如果在AD+Exchange里面，就建Distribution List。比如，我会建ABC Project Core Team，ABC Project Dev Team，ABC Project All Testers，ABC Project Extended Team等等。这样发起Email来方便，而且能让该收到email的人都收到、不该收到不被骚扰。

43. 每个人都知道哪里可以找到全部的文档么？

应该每个人都知道。这叫做知识管理（Knowledge Management）。最方便的就是把文档放在一个集中的File Share，更好的方法是用Sharepoint。

44. 你做决定、做变化时，告诉大家原因了么？

要告诉大家原因。Empower team member的手段之一是提供足够的information，这是MSF一开篇的几个原则之一。的确如此，tell me why是人之常情，tell me why了才能有understanding。中国人做事喜欢搞限制，限制信息，似乎能够看到某一份文件的人就是有身份的人。大错特错。权威、权力，不在于是不是能access information/data，而在于是不是掌握资源。

45. Stay agile and expect change 要这样。

需求一定会变的，已经写好的代码一定会被要求修改的。做好心理准备，对change不要抗拒，而是expect change。

46. 你们有没有专职的软件测试人员？

要有专职测试。如果人手不够，可以peer test，交换了测试。**千万别自己测试自己的**。

47. 你们的测试有一份总的计划来规定做什么和怎么做么？

这就是Test Plan。要不要做性能测试？要不要做Usability测试？什么时候开始测试性能？测试通过的标准是什么？用什么手段，自动的还是手动的？这些问题需要用Test Plan来回答。

48. 你是先写Test Case然后再测试的么？

应该如此。应该先设计再编程、先test case再测试。当然，事情是灵活的。我有时候在做第一遍测试的同时补上test case。至于先test case再开发，我不喜欢，因为不习惯，太麻烦，至于别人推荐，那试试看也无妨。

49. 你是否会为各种输入组合创建测试用例？

不要，不要搞边界条件组合。当心组合爆炸。有很多test case工具能够自动生成各种边界条件的组合——但要想清楚，你是否有时间去运行那么多test case。

50. 你们的程序员能看到测试用例么？

要。让Dev看到Test Case吧。我们都是为了同一个目的走到一起来的：提高质量。

51. 你们是否随便抓一些人来做易用性测试？

要这么做。自己看自己写的程序界面，怎么看都是顺眼的。这叫做审美疲劳——臭的看久了也就不臭了，不方便的永久了也就习惯了。

52. 你对自动测试的期望正确么？

别期望太高。依我看，除了性能测试以外，还是暂时先忘掉“自动测试”吧，忘掉WinRunner和LoadRunner吧。对于国内的软件测试的现状来说，只能“矫枉必须过正”了。

53. 你们的性能测试是等所有功能都开发完才做的么？

不能这样。性能测试不能被归到所谓的“系统测试”阶段。早测早改正，早死早升天。

54. 你注意到测试中的杀虫剂效应了么？

虫子有抗药性，Bug也有。发现的新Bug越来越少是正常的。这时候，最好大家交换一下测试的area，或者用用看其他工具和手法，就又会发现一些新bug了。

55. 你们项目组中有人能说出产品的当前整体质量情况么？

要有。当老板问起这个产品目前质量如何，Test Lead/Manager应该负责回答。

56. 你们有单元测试么？

单元测试要有的。不过没有单元测试也不是不可以，我做过没有单元测试的项目，也做成功了——可能是侥幸，可能是大家都是熟手的关系。还是那句话，软件工程是非常实践、非常工程、非常灵活的一套方法，某些方法在某些情况下会比另一些方法好，反之亦然。

57. 你们的程序员是写完代码就扔过墙的么？

大忌。写好一块程序以后，即便不做单元测试，也应该自己先跑一跑。虽然有了专门的测试人员，做开发的人也不可以一点测试都不做。微软还有Test Release Document的说法，程序太烂的话，测试有权踢回去。

58. 你们的程序中所有的函数都有输入检查么？

不要。虽然说做输入检查是write secure code的要点，但不要做太多的输入检查，有些内部函数之间的参数传递就不必检查输入了，省点功夫。同样的道理，未必要给所有的函数都写注释。写一部分主要的就够了。

59. 产品有统一的错误处理机制和报错界面么？

要有。最好能有统一的error message，然后每个error message都带一个error number。这样，用户可以自己根据error number到user manual里面去看看错误的具体描述和可能原因，就像SQL Server的错误那样。同样，ASP.NET也要有统一的Exception处理。可以参考有关的Application Block。

60. 你们有统一的代码书写规范么？

要有。Code Convention很多，搞一份来发给大家就可以了。当然，要是有FxCop这种工具来检查代码就更好了。

61. 你们的每个人都了解项目的商业意义么？

要。这是Vision的意思。别把项目只当成工作。有时候要想着自己是在为中国某某行业的信息化作先驱者，或者时不时的告诉team member，这个项目能够为某某某国家部门每年节省多少多少百万的纳税人的钱，这样就有动力了。平凡的事情也是可以有个崇高的目标的。

62. 产品各部分的界面和操作习惯一致么？

要这样。要让用户觉得整个程序好像是一个人写出来的那样。

63. 有可以作为宣传亮点的Cool Feature么？

要。这是增强团队凝聚力、信心的。而且，“一俊遮百丑”，有亮点就可以掩盖一些问题。这样，对于客户来说，会感觉产品从质量角度来说还是acceptable的。或者说，cool feature或者说亮点可以作为质量问题的一个事后弥补措施。

64. 尽可能缩短产品的启动时间要这样。

软件启动时间（Start-Up time）是客户对性能好坏的第一印象。

65. 不要过于注重内在品质而忽视了第一眼的外在印象程序员容易犯这个错误：太看重性能、稳定性、存储效率，但忽视了外在感受。而高层经理、客户正相反。这两方面要兼顾，协调这些是PM的工作。

66. 你们根据详细产品功能说明书做开发么？

要这样。要有设计才能开发，这是必须的。设计文档，应该说清楚这个产品会怎么运行，应该采取一些讲故事的方法。设计的时候千万别钻细节，别钻到数据库、代码等具体实现里面去，那些是后面的事情，一步步来不能着急。

67. 开始开发和测试之前每个人都仔细审阅功能设计么？

要做。Function Spec review是用来统一思想的。而且，review过以后形成了一致意见，将来再也没有人可以说“你看，当初我就是反对这么设计的，现在吃苦头了吧”

68. 所有人都始终想着The Whole Image么？

要这样。**项目里面每个人虽然都只是在制造一片叶子，但每个人都应该知道自己在制造的那片叶子所在的树是怎么样子的**。我反对软件蓝领，反对过分的把软件制造看成流水线、车间。参见第61条。

69. Dev工作的划分是单纯纵向或横向的么？

不能单纯的根据功能模块分，或者单纯根据表现层、中间层、数据库层分。我推荐这么做：首先根据功能模块分，然后每个“层”都有一个Owner来Review所有人的设计和代码，保证consistency。

70. 你们的程序员写程序设计说明文档么？

要。不过我听说微软的程序员1999年以前也不写。所以说，写不写也不是绝对的，偷懒有时候也是可以的。参见第56条。

71. 你在招人面试时让他写一段程序么？

要的。我最喜欢让人做字符串和链表一类的题目。这种题目有很多循环、判断、指针、递归等，既不偏向过于考算法，也不偏向过于考特定的API。

72. 你们有没有技术交流讲座？

要的。每一两个礼拜搞一次内部的Tech Talk或者Chalk Talk吧。让组员之间分享技术心得，这笔花钱送到外面去培训划算。

73. 你们的程序员都能专注于一件事情么？

要让程序员专注一件事。例如说，一个部门有两个项目和10个人，一种方法是让10个人同时参加两个项目，每个项目上每个人都花50%时间；另一种方法是5个人去项目A，5个人去项目B，每个人都100%在某一个项目上。我一定选后面一种。这个道理很多人都懂，但很多领导实践起来就把属下当成可以任意拆分的资源了。

74. 你们的程序员会夸大完成某项工作所需要的时间么？

会的，这是常见的，尤其会在项目后期夸大做某个change所需要的时间，以次来抵制change。解决的方法是坐下来慢慢磨，磨掉程序员的逆反心理，一起分析，并把估算时间的颗粒度变小。

75. 尽量不要用Virtual Heads 最好不要用Virtual Heads。

Virtual heads意味着resource is not secure，shared resource会降低resource的工作效率，容易增加出错的机会，会让一心二用的人没有太多时间去review spec、review design。一个dedicated的人，要强过两个只能投入50%时间和精力的人。我是吃过亏的：7个part time的tester，发现的Bug和干的活，加起来还不如两个full-time的。参见第73条。73条是针对程序员的，75条是针对Resource Manager的。

我现在做的项目是采用如下方法管理的：

BD：基础设计。在这个文档里，我们把程序的界面全部画出来；界面上的功能全部描述完整。如：一个查询界面的条件是什么，查询出来的结果如何显示等等。

FD：功能设计。在这个文档里，对BD阶段的各个页面里包含的数据逻辑处理做说明。如：查询时调用的数据处理函数该如何设计，入口参数，返回参数，关联的表等等各方面的说明。

因为程序的界面已经定了，数据处理逻辑也定了。所以，就开始编码阶段。当编码过程中发生什么问题，程序的整个功能还是必须满足BD和FD设计文档中的要求。程序中的各种疑问，都以BD和FD文档中的说明为准。

基本上我们一个小项目的开发周期为2个月，BD为2-3周，FD1周，PG（编程）2-3周，CT(测试)2周。

测试完毕后交出去的就是成品，基本上不会再有系统要求变更的问题了。如果有变更，且不在BD设计范围内，那就是新增需求。就是一个新项目了