第 2 章 做好程序员 2

第 2 章 做好程序员

在软件项目中，凡是参加编程的软件开发人员通称为程序员。程序员的基本任务是编程，可能还要参加需求开发、系统设计、测试和维护等工作。

程序员是软件企业最宝贵的资源之一，如何培养优秀的程序员是从 60 年代起就开始讨论的话题。

本章论述程序员“做人”和“做事”的道理。不要觉得程序员只应该钻研软件技术，可以不关心世事并且允许自由散漫。他们不该因为思想幼稚而显得单纯，应该是成熟了才变得单纯，才适应这个充满活力的职业。

2.1 漫谈程序员

《编程之道》[James] 是一本在全世界流传的关于程序员的寓言，讲的是早期程序员的故事，风趣而富有哲理。好笑的是书中很多观点现在看来明显是错误的甚至是荒唐的。但这毕竟是赞美程序员的书啊，怎么看都觉得错得有理，傻得可爱。于是大家心照不宣，乐呵呵地传阅，智者一笑了之，幼稚者信以为真。

早期的程序员干活能从软件直通硬件，个个生猛无比。又因他们的作息时间、言行举止与常人不太一样，久而久之就给人们留下了“神秘”、“孤僻”的印象。

如今软件行业被炒得热火朝天，有能耐的程序员即便躲在大山岙的军工厂里也能被挖出来。而更多原本不会编程的人操起几本“速成”、“二十一天通”等书籍也加入了这个行业。现在国内号称有上百万程序员，这支大军鱼龙混杂，已搞不清那些是正规军，那些是民兵游击队了。

真正的程序员有如下秉性：

一、诚实

程序员在学习与工作期间几乎天天与机器打交道，压根就没有受欺骗或欺骗人的机会。勤奋的程序员在调试无穷多的 Bug 时，已经深深地接受了“诚实”的教育。不诚实的人，他肯定不想做、也做不好程序员。

有一名市场营销员和一名程序员都在新闻发布会上发言，将一项新技术的消息公布于众。

市场营销员说：“这项技术比电话、晶体管和原子弹三项发明加起来对世界文明的影响都要大。”

程序员说：“这项技术在有限的领域内，在有限的程度上，解决了一些技术性的问题。”

看来为了让我们的民族更加诚实，学电脑真的要从娃娃抓起。

二、信奉简单实用主义

有人问一个数学家，一个物理学家和一名程序员：“一个盒子有几个面？”

数学家回答说：“有六个面，因为盒子是长方体。”

物理学家回答说：“有 12 个面，分为 6 个外表面和 6 个内表面 。”

程序员回答说：“只有两个面，里面放电路板和硬盘，外面放显示器和键盘。”

程序员的基本工作就是把复杂的问题转化为计算机能处理的一些简单的程序。如果一个问题复杂到连程序员自己都不能理解，他就无法编写出程序让更笨的计算机来处理。

也有不少做计算机“学问”的人颠倒行事。本来几句话、几行程序就能说明白的事，非得要抬高到理论创新的程度，写成玄乎的文章去评职称或混学位。幸好在第一线工作的程序员大多是实干的。

三、喜欢技术挑战

程序员大都喜欢技术挑战，不喜欢搞测试与维护。高水平的程序员喜欢与高水平的程序员一起工作，因为他们怕“与臭棋佬下棋，棋越下越臭”。

“喜欢技术挑战”听起来是好事，但实际上未必都是好事。有时候这种喜好常常导致程序员干活偏离项目真实的要求。

四、工作单调但不乏味

有人问编程大师：“程序设计的真正含义是什么 ？”

大师回答说：“饿了的时候就吃，困的时候就睡，只要时机恰当就进行程序设计。”

其实程序员的生活和编程已融为一体，尽管单调却不乏味，还能独享孤独。有诗为证：

我编程三日
两耳不闻人声
只有硬盘在歌唱

程序员小结：“编程”是程序员的基本任务但不是唯一的任务。水平高的程序员通常从事分析、设计、编程等技术难度较高的工作。而水平较“臭”的程序员将“沦落”到只干测试、维护等工作。优秀的程序员没有理由不让人喜欢，他们远比怪僻来得可爱。

2.2 职业道德

美国电气电子工程师协会（IEEE）和美国计算机协会（ACM）于 1994 年联合制定了软件行业职业道德准则：“Software Engineering Code of Ethic and Professional Practice”，将此作为工业决策、职业认证和教学工作的参考。

本节不讲太多道貌岸然的话。化繁为简，作者自己总结了关于程序员职业道德的“三不”准则，按从轻至重的顺序论述。

2.2.1 上班时间不干与工作无关的事情

老板给程序员发工资，并不是让他们来享受的，而是希望他们创造比工资多得多的

效益。所以要求程序员“在上班时间不干与工作无关的事情”是合情合理的。

争议在于什么算是“与工作无关”的事情。

从哲学角度讲世间万物都是相关的，怎么可以说我干的事“与工作无关”呢。由于对相关或无关的“度”不好把握，上述“准则”难以得到理想的贯彻，还得靠人们的自觉性。

上班时收发私人电子邮件或者打私人电话，行不行？按规定是不允许的。但万一有重要的私事，总不能不处理吧，于是就有了违反规定的理由。可是谁会觉得自己的私事不重要呢？

鉴于此，很多公司会对电子邮件和电话的使用设置权限，高级别的员工自由度大，低级别的员工自由度小。有一些公司对员工的电子邮件进行监视，但这样做会侵犯隐私，可能会引起纠纷。

也有人在上班时学英语，准备考这考那。这比收发私人电子邮件要严重，若被管理人员发现是要挨训的。这类事情不少见，建议从两个方面看问题：

（1） 如果员工有意不干正经活，应按怠工处理。

（2） 在人浮于事的大公司里，有不少员工的确很清闲，与其让他荒废时间还不如让他多学点知识。我从来不反对学习新东西，对那些无“正事”可干、只好偷着学其它知识的同事深表理解，于是我就睁只眼闭只眼。我认为发生这类事情的原因在于领导的失职，而不是“犯规”者的错。

如果有人在上班时间玩游戏，不要让他找借口，一定要及时制止甚至作出处罚，因为影响太坏。除非他的工作就是开发或者推销游戏软件。

下班后呆在公司里干与工作无关的事情行不行？

这个问题不好回答。我所在的公司比较“客气"，只要你不干坏事，那是允许的。

我曾看到某个大公司有这样的规定：在公司里每天 24 小时不得干与工作无关的事情。

这个规定不但冷酷而且荒谬，正常人在每天 24 小时之内总得大小便吧！

2.2.2 不损害集体利益

从小学开始，学生守则里头就有“不损害集体利益”的条文。

出于嫉妒或者仇狠而有意去破坏集体利益的，这类人的心理不太正常，不在本书讨论之列。

在 IT 企业里，损害集体利益的常见行为有两类：一是泄密，二是盗卖成果。

泄密有两种：一是无意的，二是有意的。

无意的泄密比较常见，主要原因有：

（1） 不同的人在判断某东西“是否属于机密”时的看法不尽相同。不少工作成果具有实用价值但没有创新意义，商业人士认为它应当被保密，而在技术人员眼里它可能算不上是机密而被随意处置。

（2） IT 行业人士需要经常与外界交流，“闭门造车”很难做出好产品。“交流”自然会涉及有价值的东西，老谈些众所周知的事情不能算“交流"，那叫“唠叨"。同行

之间交流时显然不会捧着教科书朗读，有意义的交流通常聚焦在“如何解决企业面临的实际问题”上，难免会泄漏公司机密。

说来惭愧，我自己都无法避免无意识泄密。我经常做讲座，写技术性文章和书籍，在很多场合有人向我要电子文档，人们向我要东西是看得起我，我怎么可以不把东西给他们呢？

减少无意泄密的措施有：

（1） 对新员工进行入公司培训时，灌输保密意识。

（2） 加强保密管理，堵住通讯漏洞。

在 IT 企业里，如果有人存心泄密或者盗卖成果，一般是防不住的。如果发生这类事件，应当诉诸法律，使肇事者得到应有的惩罚。

我不建议企业为了保密而制定非常严格的管理制度，因为这样做会给大部分守法的员工造成不便，而事实上却又防不住那小部分坏人。

2.2.3 不干危害社会的事情

软件病毒防不胜防，常令受害者“无冤无故”地遭受损失。由于病毒对社会的危害极大。制造病毒被认为是犯罪行为。

黑客的行为通常是有针对性的，不象病毒那样“六亲不认”，对社会的危害比病毒轻些。但不论黑客是在搞恶作剧还是在犯罪，其快乐是建立在别人的痛苦之上的。正直的程序员不该去造病毒或者当黑客。

黑客有些时候也干一些正义的恶作剧，所以不可一概而论地谴责他们。例如有一些中国黑客经常性地把很多反华的网站给“黑”了，虽然我是“反黑人士”，但我是中国人，所以拍手称好，否则我就失节成了“二鬼子”。

不少人当黑客并不是想获取经济利益，而是为了寻求技术刺激。有些大学生把黑客当成技术天才来崇拜，争相模仿，这种念头会令更多的人误入歧途。

尽管找不出堂皇的理由，我还是要规劝程序员不要去当黑客。且不论黑客性质好坏，每个人总得考虑自己的前途吧？人不能一辈子当黑客，当他有了家小后，他总该设法找个好职业挣钱养家吧？

大部分黑客的致命缺点是综合才能不够强。由于天天想着搞破坏，他们很少有机会、有耐心去开发大型的软件系统，因此系统分析、系统设计的才能比较差，更别谈项目管理和企业管理了。但是如果综合才能不够强，他怎么能够获得很好的职位呢？前途岂不黯淡？

黑客们原本智力不错，很有发展潜力，但是心用歪了，自己断绝了提高综合才能的道路。不少年轻人之所以崇拜黑客，是因为他们还太幼稚，体会不到干事业的艰辛，尚不能分辨“狗熊”和“英雄”。为了防止人们误入歧途，以及挽救已入歧途的人们，软件业界有责任在任何场合消灭“黑客光荣”的邪念。

有位小姐找了一名技术不错的程序员做男朋友，她常对大伙说：“要是我男朋友是一名黑客，我会更加自豪的。”

后来她结婚了还在感叹：“要是我老公能成为一名黑客那该多好啊！”

这事情发生在我家里，真是个讽刺！

2.3 工作态度

如果某人在汇报工作时口齿不清、结结巴巴，虽然令听者难受，但尚可接受。倘若说话吞吞吐吐、支支吾吾，则会令人恼火。前者是表达能力差（或者生理问题）导致的，可以被原谅。而后者是工作态度差导致的，难以被原谅。

软件项目能否顺利完成与开发人员的才能以及工作态度密切相关。我们可以用“认真负责”四个字来概括“好”的工作态度，这是所有开发人员都有能力做到并且应该做到的。由于工作态度是可塑性很强的东西，不太好制成规范。本节仅表述作者对“认真负责”和“服务意识”的一些看法。

2.3.1 认真负责

程序员老老实实呆在办公室里干活，上班时间比别人长，是“认真负责”吗？

那是一种形式而非实质。怎样才算是“认真负责”呢？

对程序员而言，力求软件中没有缺陷才称得上是“认真负责”。这里以编写“无错”（bug-free）程序为例，对“认真负责”作具体的释义。

“无错”是编程应追求的目标之一。这个目标不容易达到，但如果不愿意为此付出努力，那么可能会导致程序错误百出。

程序员查找程序错误通常有两种方式：一是用编译器查错，二是测试。

如果编译时发现程序有问题，编译器会在相应的地方给出警告或错误信息。这种自动化的查错功能对程序员而言是莫大的帮助。所以应该将编译器的查错选项设置为最严格等级，以便最大限度地利用编译器的查错功能。这样设置后通常会导致程序在编译时出现更多的警告或错误信息，因此引起了一些程序员的不满，以为是编译器给他们带来了麻烦。真是糊涂啊！设想如果你去体检，有个医生草草看了一下说“没事没事”；另一个医生检查得很细致，结果发现很多毛病，让你吓一跳；难道你会觉得第一个医生比第二个对你“好”？

有不少程序员对编译时产生的警告满不在乎，认为“警告”不算是“错误”，反正程序能运行，何苦费力对付一堆“警告”呢。请想一想，编译器会平白无故地对程序发出警告吗？肯定是程序中存在不合理的东西，谁能保证那不是潜伏的 Bug？所以程序员应当把编译时产生的警告或错误消灭干净。

如果程序在编译时完全正确，是否就没有错误了？不见得。

编译器通常只能查出语法错误，其查错能力有限，我们还要靠测试来找出更多的问题。

测试人员有两类：一是自己，二是别人。关于测试的原理和方法本书不多论述。这里强调，在开发过程中，程序员应该对自己的程序执行“逐步跟踪调试”，程序员不要认为测试这活是别人干的。“逐步跟踪调试”能发现很隐蔽的错误，非常有用，但是很多程序员嫌其麻烦而不用。这一偷懒往往使他们在项目开发后期碰到更多的麻烦。

2.3.2 服务意识

程序员是搞软件开发的，又不是饭店里的服务员，谈何服务？

人生活在社会里，总是既为别人服务，又得到了别人的服务。对程序员而言，服务对象至少有两类：一是开发团队里的其他人，二是软件的用户。

在软件开发过程中，每一阶段的工作成果都将作为下一阶段工作的输入。位于前面阶段的开发人员必须为位于后面阶段的开发人员提供服务。例如，若系统设计人员对需求难以理解，那么需求开发人员必须对设计人员解释清楚需求（即提供服务），不能说“我的工作都写在需求文档里头了，你自己看着办吧”。

软件产品不可能是完美无缺的，用户在使用过程中会遇到这样那样的问题，于是要求软件开发方给予解决。如果的确是软件有问题，不论是有偿服务还是无偿服务（根据合同而定），总之开发方必须对用户“有求必应”。

将软件卖给用户赚了钱后，如果软件出了差错时却不理睬用户，这是很缺德的。如果开发软件时埋下了一些“祸根”，虽然刚卖出软件时赚了钱，但将来的维护代价也许很高，总算起来可能是“赔了夫人又折兵”。别怨用户给你找麻烦，麻烦通常是自己没把工作做好造成的。用户“缠”得紧表示他“认真负责”，更应该敬重他。

除了本职开发工作外（如分析、设计、编程等），程序员对服务工作也要认真负责。用户还要为他自己的用户服务哩。服务正可谓“我为人人，人人为我”，社会因此而更加和谐。

2.4 高效率地工作

社会道德倡导人们应该吃苦耐劳，为个人、家庭、集体、国家多作贡献，但并不因此要求人们天天辛苦地工作、辛苦地生活。聪明勤劳的人应该愉快地工作、愉快地生活才对。

IT 行业竞争很激烈，绝大多数 IT 人士工作比较勤奋，生活节奏较快。可是太多的人不知不觉地以低效率的方式工作，这样日复一日，真是生产力的巨大浪费。

如果不想使自己活得太累，人们应该摸索出一套适合自己的富有效率的工作方式。本节总结这方面的一些经验，供上班族参考。

2.4.1 合理安排一天的时间

一天的标准上班时间是 8 小时，说短不短，说长也不长。利用好了可以干很多事，用得不好时间一晃而过。由于人不是机器，生理决定了人的机能不可能在每天的所有时刻处于同一水平线，有时好有时差。所以我们应当在机能处于最佳状态时做最重要的事情，在最差状态时干最次要的事情。

一般地，上午几个小时人们的状态最佳。开发人员应集中精力从事最费脑筋的研发工作，不要让自己分心干一些次要的事情。如果管不住自己，那么注定会降低效率。团

队也应当达成共识，尽量不在上午打扰他人的智力创作。

中午吃饭后一个小时以内，由于人体忙于“消化、吸收”，此时脑袋晕乎乎的，俗称“饭醉”。大部分企业中午只留半小时吃饭，没有午休时间。少数企业有午休时间，但用提前上班或者延迟下班的方式来补回时间。学校里的午休时间比较长，大部分学生有午睡的习惯。所以毕业生刚到企业工作时，头几个星期一到中午就哈欠连天。有些人能够调整生物钟，不少人改不了，我属于后者。我曾在多个地方多次尝试不午睡，这样勉强能利用中午约一个小时的时间，但是导致整个下午乃至整个晚上昏昏沉沉，极不划算。相信与我同感的人肯定不少。于是我在吃完中饭后就毫不犹豫地午睡，虽然违反了公司规定，但我会在晚上补偿，所以心安理得。

如果公司没有午休制度，那么个别员工午睡时要注意“形象”。别躺着睡，别打呼噜，至多扒在桌上打个盹儿。我看到有些人能笔直地坐在椅子上入睡，如老僧入定，令人羡慕不已。我没有那么高的水平，就用胳膊支着腮膀睡，一副思考的模样。

国内一家著名的大型 IT 企业有 2 个小时的午休时间，我拜访过几次，真是大开眼界啊：每个员工都有床垫、被子，平时放在大柜子里，一到午睡便拿出铺在地板上。纳头就睡，鼾声此起彼伏，天皇老子来了也不理。经过这样的“养精蓄锐”，员工们在下午、晚上工作时就精神抖擞。我在该公司尝试过一次午休，感觉非常好。

无论是否经过午休，人在下午的状态总不如上午好。所以下午适合于干不需要太费脑筋的研发工作。例如在上午“挖空心思”设计算法，下午编程去实现它。离下班前的半个小时里，很多人心不在焉，编程出错率增加。这时候可以处理一些必要的事务，例如发通知、写小结、开会、布置明天的任务等等。

下班后的时间原则上可以自由支配。尽管我不提倡加班，但我认为 IT 人士应当在工作日的每个晚上花一两个小时处理工作事务以及学习。这个行业不允许我们荒废太多的时间，尤其对于年轻人。从周一至周五，我几乎每天晚上都会在办公室里呆上两个小时，但从来不“美其名曰”加班。由于我白天午睡、喝水、上厕所、接电话花费了不少时间，我肯定没有完成心目中的 8 个小时工作任务。所以必须在晚上继续研发或者处理事务，干这份内的事情不能算是加班。时间宽余就上网浏览业界动态，看看新闻，收发电子邮件，翻阅资料等等。

如果周一至周五你的工作效率很高，你就会觉得很充实。到了周六周日，你就可以尽情享受生命中的其他美好的事物，例如吃喝、运动、看电影、看闲书、与朋友聊天等等。这样过日子很幸福。不过对于二十出头的毛小伙子来说，除了体育运动外，大部分业余时间还是应该用于学习的。等到年龄稍大“成家立业”后，才应把业余时间投入到生活中去。

2.4.2 减少路上花费的时间

在上下班路上花费很多时间是无谓的浪费。在大城市里工作，员工的住宿地可能离公司比较远，而且交通拥挤，上下班单程时间花掉一个小时很常见。这样长年累月下去，对 IT 人士而言无疑是浪费生命。所以无论是公司还是个人都应该好好考虑如何减少路上花费的时间。

如果公司有班车，单程时间在一个小时左右勉强可以接受。如果是挤公共汽车，那就不值得了。比如在炎热的夏天，你天天满头大汗、浑身湿漉漉地赶来上班，哪还有好心情去工作。这样的隐性损失很严重。你应该尽可能地住得离公司近一些，哪怕房租贵一点也值得。节省上班时间比下班时间更重要。如果住宿地不能变动（如买了房子），也没有班车，那么不妨在上班的时候坐出租车，这样既快又舒适，当然这要视你工作的重要性和收入而定。

中华民族历来提倡节约，这是传统美德。但是要节约的东西很多，并非只有钱。有时候时间比钱值“钱”得多，前者浪费了就追不回来，但是后者还有机会。大概中国人穷怕了，我们从小就被灌输了“省钱”的观念。人长大后应该懂得分析“时间成本”，如果花费时间的代价比节省下来的钱多得多，那么宁可花掉钱也别浪费时间，一味省钱是很愚昧的。

我有时候早晨睡迟了，没赶上公司的班车，若挤公共汽车则要花费一个小时，我就毫不犹豫地坐出租车上班。看起来有些奢侈，实则很值得。因为我上班一个小时的工资比出租车费多，我又何必花一个小时挤公共汽车呢，把省下来的时间和力气用于创造价值岂非更好！

我常从经济学书籍中找出一些理论为自己“撑腰”，上例就是。有一回我搞砸了，我曾读了关于“货币流通”的论著，没弄清楚我就照般行事。我以为花出 10 元钱，会给某甲带来 10 元的收益，当钱从甲流到乙时又给乙带来 10 元的收益，如此传递下去将会产生无穷的价值。我为“发现”这滚雪球般的增值效应而兴奋不已，于是把自己为数不多的收入基本花光，并劝说朋友们学我样，为促进“货币流通”做贡献。不久前我请一位很有学问的朋友吃饭，我兴高采烈向他阐述关于“货币流通”的“新发现”，被他几句话点醒过来。这事我犯了常识性的经济学错误，给了自己一个告戒。但到目前为止，我认为本节所讲的“时间成本”是正确的，多花钱来减少上下班路上的时间是值得的。

2.4.3 开会

开会的目的是获得见解，但它会消耗大家的时间，成本比较高。我搞不清楚什么时候起中国人有了开会的癖好。讽刺开会的小品、相声看多了，你会觉得“开会”是个贬义词。

很多时候，人们开会的目的不够明确，以至会议期间得不到共识，只是白白浪费时间。为了让会议有成效，应该先让所有与会者明白究竟要做什么。而主持人在发出开会通知前，应该先问自己这样的问题：

这个会议是否真的重要，即使中断很多人的工作也值得？

是否没有其它不影响别人工作的方法来取代开会？

这次会议的目的是什么？我该怎样做才能达到目的？

如果你能清楚地回答上述问题，一般不会让会议变成漫无目的的讨论。

安排开会的时间也挺讲究，要注意尽量别中断做了一半的工作。若无重要原因，请不要把开会的时间定在上午 10 点或者下午 3 点，这样会把上午或者下午的时间切割得太零碎，最好安排在一清早或者快下班前。

每周的例行会议应集中在同一时间段，例如周一上午或者周五下午，因为这个时间段比较适合于处理工作事务。如有可能，不妨把所有的会议都集中一起，痛痛快快地一次性解决，免得在其它时间开会妨碍别人工作。[Maguire94, p127-p136]

2.4.4 处理电子邮件

电子邮件是个很棒的通讯工具，如果没有它简直无法想象开发人员怎么交流。电子邮件能提高工作效率，但是如果使用不当，同样会伤害生产力。

开发人员的工作经常被电子邮件打断。很多程序员只要发现有新的电子邮件进来，便停下手头的工作，看看有什么新鲜事发生了。最糟糕的是马上回复电子邮件，因为“回复”比“阅读”花更多的时间。

由经验可知，有一大半的电子邮件并不重要，用不着实时地处理。如果每隔几分钟就处理一次，这样一天下来人们可能什么事情也做不成，因为开发工作是无法分割成很多个几分钟去完成的。

团队应当达成共识：最好是在刚上班、中午、下班前集中处理电子邮件。这个建议适合于大多数程序员。

请不要钻牛角尖，如果处理电子邮件是你的重要工作，或者你经常有紧急的电子邮件，那么你没有必要非得等到某个时间才处理邮件。

2.4.5 随时记录

随时随地记录你在工作中遇到的问题，以及你产生的灵感。不要等到将来再靠回忆来写总结，那时候你可能想不起来了，岂非浪费了一笔“财富”？

据说唐朝诗人李贺每当在路上想到好诗句，就顺手写下来，仍进一个专用袋里，回到家后再取出那些诗句拼接成诗。怪不得人们说：李贺的好诗不多，但是好句子很多。

程序员每天应该写一份简短的工作日记，花几分钟就行了。很多程序员觉得他们的光荣任务就是开发软件，写工作日记很费时间很烦人，这是心理作怪而非客观事实。整个团队都写工作日记不仅对个人有好处，而且有利于项目的管理。如果这些工作日记保存在数据库里，那么管理者会对各人的工作进展一目了然。如果大家还没有写工作日记的习惯，项目经理就强迫大家写，用不了几天就习惯了。

2.5 程序员服什么样的项目经理

项目经理是企业的基层干部，是推动企业发展的中坚分子。很多管理学书籍对“什么样的人适合于当项目经理”有许多高见，但行业不同见解也不尽相同。本节把范围限制在软件项目经理上。

了解“程序员服什么样的项目经理”很重要，这样企业就可以有目的地招聘或者自己培养合格的软件项目经理。

2.5.1 丰富的产品开发经验和比较高的技术水平

对于软件产品开发而言，很多时候开发经验比技术水平、学历还重要。就如战斗中，如果队伍中有一些老兵，那么战胜的机会就多些，士兵们心里也踏实些。

如果项目经理拥有丰富的产品开发经验，那么意味者他有能力不让项目发生混乱，并且在出现一些意外时，他自己不会手忙脚乱。

不少大公司让刚刚毕业的博士担任项目经理，这是非常危险的做法。我自己就是博士，也算是高材生，也担任过经理职务，还招聘过不少员工。以我冷眼观察，如果让刚从学校毕业的所谓的高材生担任经理职务，十有八九会把事情搞砸了。产品开发经验可不是高材生在学校里能学到的。

不可否认，每个程序员骨子里头都有一股傲气。大家都是靠技术吃饭，谁怕谁呀！如果项目经理能够技压群雄，程序员们就会听他指挥，哪怕他霸道一些也无所谓。如果项目经理的技术水平虽然不是团队里最好的但也相当不错，大家在技术方面有共同语言，甚至能产生“英雄所见略同”的感觉，那么大伙儿也能齐心协力地干活。

一个技术水平较差的人被任命为软件项目经理真是个悲剧，就象一个略有权势的太监，表面上有人对他恭维，背后却被人鄙视。

我们经常会听到有经理头衔的人在高谈阔论：“编程我不会，做个项目还不 easy？派个人去搞系统分析，回头再叫几个程序员把需求译成程序，不就 OK 了吗？”

不懂英语的人准以为 easy 和 OK 是贬义词。

要让软件项目失败很容易，只要符合下列条件之一即可：

（1）项目经理对软件开发一窍不通。

（2）程序员对软件开发不感兴趣，应付了事。

（3）程序员是临时雇用的。

如果上述三个条件同时具备，那就死定了。

2.5.2 懂得管事和管人

很多项目管理书籍把“人员管理”放在第一位。而我强调，对于软件项目管理而言，应当把“管事”放在“管人”前面。这是软件项目的特征决定的。

国内绝大部分软件项目的人数比较少，通常是十人以下。很有趣的是，软件开发人员所受的教育非常相似，他们的言行举止、甚至喜好都比较相似。所以，软件开发人员的管理相对而言是比较简单的，但是管理软件开发工作则是复杂的。

如果把“管事”放在项目管理的首位，那么项目经理的管理目标就很清楚。他自己必须懂得“做事”才能“管事”。

如果把“管人”放在项目管理的首位，那么项目经理会把管理重点集中到人际关系上，于是经常吃喝玩乐，拉拢关系，不知不觉地脱离了第一线工作，这是一种风险。

项目经理想把事情管好，他必须学习并熟悉“软件工程”和“CMM”。说得更加明确些，项目经理应当：

（1） 熟悉经典的软件工程，如需求开发、系统设计、编程、测试和维护等。

（2） 熟悉 CMM 2-3 级的关键过程域，如项目计划、项目监控、需求管理、配置管理、质量保证、技术评审等等。

2.5.3 较好的人格魅力

项目经理应善于了解组员的心理，懂一些人情世故，但不靠“拉拢关系”办事。管理不是管制，不是去卡住人家的脖子，因为程序员不是一群野鸭子。管理的目的是让大家一起把工作做好，并且让各人获得各自的快乐和满足。当一个团队被出色地领导时，雇员甚至不知道他们已被领导。在项目完成时，他们会自豪地说：“看看我们通过努力取得的成绩吧”。所以管理者不能老惦记着自己是一个官，而应时刻意识到自己是责任的主要承担者。

软件开发是智力创作过程，不能把程序员当机器人，别指望他们会百分百地按照领导的意图行事。如果领导缺乏人格魅力，很少有人会信服你，团队就缺乏凝聚力，乌合之众不可能开发出优秀的软件来。

很多软件公司的经理都不是管理专业出身的，技术出色的经理一般少有心术不正的，他们也不可能为了搞好管理而成天玩弄心机。但既然当上了经理，就不能放任自己，有必要培养一些人格魅力，至少要做到“以身作则”和“公正待人”。

一、以身作则

项目经理干活要快且好，别人要花一天时间的活，他半天就能做完，这样才会有精力去搞管理。项目经理应当参加最难的开发工作，并指导不同水平的程序员把各自的工作做好，与大家同甘共苦。如果人手不够，项目经理要能同时干几个人的活。

好榜样的力量是无穷的，程序员们看到“头儿”干活那么卖命，他们也会激发工作热情，至少不好意思偷懒。

有一种盛传的说法：聪明的领导尽量让下属多干活，自己少干活。

这句话象河豚鱼那样鲜美，但有毒！

这是董事长而不是项目经理可以做的美梦。

“勤劳致富”是永恒的真理。如果不想多干活，却想多获益，那就别搞软件开发，应该去搞投机倒把。

有些人当上项目经理后以为“当官了可以享福”，在技术领域，这样的人很快会被淘汰出局的。

二、公正待人

如果项目经理在上班时趴在桌上睡觉，就别训斥程序员们学样。

如果项目经理发现有两个程序员趴在机器旁睡觉，不能只对其中一个大声吼叫：“你一编程就想睡觉，看看人家，在睡觉时都想着编程。”

公正是指不偏心，但不是指“平均主义”。如果奖罚不公正，虽然谈不上失人心、失天下，但团队的凝聚力肯定下降。这是人之常情。

项目经理小结：一个有活力的软件公司的各级经理都不会这样感叹：“因为我啥也不

会干，所以只好当领导。”

2.6 将程序员培养成为经理

如果是经营一个加工厂或一个饭店，经理们可以不必懂技术。因为他们的常识，以及通过耳闻目睹或者咨询都能解决实践中的问题。在软件领域，技术的力量是无穷的，一天之内就可使整个产业发生巨变。也许你在商业上很精明，但无法保证自己在技术浪潮中安然无恙。

软件公司总希望能物色到既精通技术又懂管理的优秀人才做经理。通常这种愿望并不能变为现实。已经出名了的优秀人才难以请到、也难以留住，而企业也不敢轻易请“陌生的高人”来任经理。所以把公司中的好员工培养成为经理是重要的举措，是长久之计。

公司的领导不要对程序员抱有偏见，以为他们只该与机器打交道。一个高水平的程序员既然能学好数字逻辑，能理得清楚软件中很多象“嵌套”这类“鸡生了蛋并且蛋又生了鸡”的错综复杂的关系，从理论上讲当个县长也不成问题。

现在很多女士不会烧菜，却能把菜的营养讲得头头是道。虽然这是个值得哀叹的社会问题，但我们应该有信心期待：如果她们非得天天烧菜不可，那么不久就能把菜烧得又好吃又有营养。

将程序员培养成为项目经理困难吗？

想想农村里的老爷爷老奶奶吧，他们子孙满堂，三代之内就有几十号人。他们没有读过多少书，也没有学过家族管理，还不是安稳地过日子吗？

普通的项目经理只是带领几个或者十几个开发人员，能有什么困难！

许多程序员不懂管理，不是智力上的原因，主要是个人兴趣和环境所致。公司的领导应该这样鼓励有灵气的员工：“你能把技术做得那么棒，还怕搞不好管理？放心干吧！”

如果程序员的悟性不低，为人不坏，技术不差，企业肯定能够比较快地把他培养成为合格的经理。的确，很多程序员经过挫折与磨练，逐渐升为组长、项目经理，乃至成为公司高层经理。

2.7 程序员升为项目经理后是否还要编程

让我们先看看 Microsoft 公司的系统软件部门与应用软件部门的领导是怎样看待这个问题的[Cusumano, p66-p67]。

Windows NT 3.0 项目的软件经理娄·帕雷罗里让他手下的经理们像他一样每天花一半的时间编写代码：

我在组内制定了许多规则，其中最重要的一条是每个人都得编程，谁也别想坐在那儿发号施令……我发现管理者很容易失去目标，他们总是无法认识到问题的本质并且反应迟缓。如果你始终不放弃编写代码，你就能对项目的进展情况了如指掌，及时发现并解决问题……我大概每天花一半的时间编写代码并寻找项目的缺陷。

作为应用软件领域的经理，克里斯·彼得斯也持同样的看法。在他任 WORD 项目总经理时就认为：

在一些大公司内部，各部门经理把具体操作的层次向下移。你一旦当上开发部门经

理，很快就会以自己身居高位、日理万机为由放弃编程；同样地，开发小组的组长会以自己重任在肩而不愿编程；至于程序员也会觉得自己十分繁忙、分身无术而不再多编写程序。虽然我是 270 名员工的领导，似乎不再需要做什么具体的工作了，但我还是为 Word 新版本编写了一个特性。

Microsoft 公司被誉为是“天才”群集的地方，程序员们个个出类拔萃，“编程水平”是被非常看重的才能。

编程只是软件开发过程中的一个重要环节，对于普通软件公司而言，软件项目经理是否应该参加编程要视具体情况而定。

一般地，程序员成为项目经理后，他应当参加一部分最复杂、难度最高的开发工作，如从事需求开发与系统设计。因为他的“水平”最高嘛，他不干，谁放心得下？

根据经验，如果项目经理脱离了开发工作，他往往会陷入行政事务。除非他的经验已经相当丰富了，否则很难控制项目的进展。久而久之，他的技术水平就会下降。有一则“程序员的演化”笑话，讽刺当程序员成为经理后，其职位越高技术水平越臭。

The Evolution of a Programmer

(1) High School Student

```
10 PRINT "HELLO WORLD"
20 END
```

College Student

```
program Hello(input, output)
begin
    writeln('Hello World')
end.
```

(2) New professional

```
# include …
void main(void)
{
     char *message[] = {"Hello ", "World"};
      int i;
      for(i = 0; i < 2; ++i)
         printf("%s", message[i]);
      printf("\/n");
}
```

(3) Seasoned professional

```
# include …
class string
{
```

```
private:
   int size;
   char *ptr;
public:
   string() : size(0), ptr(new char('\/0')) {}
   string(const string &s) : size(s.size)
      {
          ptr = new char[size + 1];
          strcpy(ptr, s.ptr);
      }
   ~string() { delete [] ptr; }
   friend ostream &operator <<(ostream &, const string &);
   string &operator=(const char *);
};
ostream &operator<<(ostream &stream, const string &s)
{
          return(stream << s.ptr);
}
string &string::operator=(const char *chrs)
{
          if (this != &chrs)
          {
               delete [] ptr;
               size = strlen(chrs);
               ptr = new char[size + 1];
               strcpy(ptr, chrs);
          }
          return(*this);
}
int main()
{
          string str;
          str = "Hello World";
          cout << str << endl;
          return(0);
}
```

(4) New Manager

```
10 PRINT "HELLO WORLD"
20 END
```

Middle Manager

mail -s "Hello, world." bob@b12

Bob, could you please write me a program that prints "Hello, world."?

I need it by tomorrow.

^D

(5) Senior Manager

% zmail jim

I need a "Hello, world." program by this afternoon.

(6) Chief Manager

% letter

letter: Command not found.

% mail

To: ^X ^F ^C

% help mail

help: Command not found.

% damn!

!: Event unrecognized

% logout

2.8 学无止境

IT 领域的新技术、新业务发展十分迅速，如果不思进取，无论昨日多么辉煌，他很快就会落伍甚至被淘汰。这是 IT 行业的生存规则。企业的命运取决于人，人要想进步必须学习，学习永无止境。俗话说“活到老，学到老”，如果真的能够做到，他绝对会感到幸福而不是痛苦。

2.8.1 不断学习新技术

软件开发人员的“饭碗”是技术。如果不学习新技术，那就捧不到好“饭碗”，甚至有掉“饭碗”的危险。

拒绝学习新技术的开发人员很少见，但是工作认真却不懂得“主动”学习新技术的人很多，这才值得担忧。不少“好苗子”在不知不觉中落伍，多么可惜！

Stephen Maguire 在其著作[Maguire, p170-p171]中谈到 Microsoft 公司存在“有五年资历的笨蛋”：

……有一位程序员，在第一年从事“文件格式转换”程序设计工作，成为能手。在此后的四年里他专门为各个软件写“文件格式转换”程序。这项工作的确是重要的，但是他的技术只在第一年里大幅度提高，其余四年都是在重复旧的工作。没有学习新技术，

事实上他停滞不前了。他有五年的工作经历，但不是有五年的工作经验，他只是用五年的时间重复第一年的经验罢了。……

类似的事情在国内大企业里比比皆是。如果工作勤恳的员工有一天发现自己是个“有五年资历的笨蛋”，真是难过啊。

我在公司里曾主持过几次内部招聘，在面试时，真的发现不少老员工（工作三年以上）的技术已经落后了。我有一个项目是“软件新技术的研究”，我在招聘组员时很在乎他是否“年轻”。根据我的经验，通常越年轻的员工越有活力，越能激发出钻研热情。

我们公司有个很重要的程控交换机产品，它的操作系统、数据库系统、编程语言都是二十年以前的。由于多年来国内对程控交换机的需求庞大，该产品利润丰厚，现在公司里有数百名开发人员围着它转。不论该产品以前多么先进，现在它的确老了，业内人士都称之为“夕阳”产品。我发现很多开发人员日复一日地做技术面很狭窄的老活。尽管他们目前还能创造价值，但是几年以后他们该怎么办？虽然开发人员为了工资不得不从事自己不太喜欢的工作，但千万不要忘了考虑自己的前程。

据我了解，没有当上“小官”的那些开发人员有以下几种工作倾向：

（1）领导让我干什么我就干什么，有活干我就干，没活干就闲着，将来的工作将来再说。

（2）呆在原岗位，一有清闲就看英语、复习考研资料等等，为将来“走人”作准备。

（3）呆在原岗位，利用现有资源做新的项目。

（4）申请换到另一个部门，寻找新的工作岗位。

（5）辞职。

没有学习新技术的机会是软件开发人员莫大的悲哀，但别用“无可奈何”来为自己的落魄找借口。有一些机会是靠运气得到的，而更多的机会是自己努力创造的。其实人在任何地方、任何时候都可以学习新知识。职位低、工资少都不可怕，可怕的是丧失了进取心。

如果你是一名在校学生，你喜欢学什么就可以学什么。如果你在企业里工作，要注意，学习新技术应当对本人和公司都有用。要是仅仅出于个人的喜好而学习新技术，通常得不到领导的支持。

2.8.2 提高综合才能

一个技术出色的软件开发人员可以自豪，但不可以目空一切。上天不可能赋于一个人太多的优点，以致于他没有表示谦虚的余地。

我们在求学、工作时可能太功利太挑剔，导致知识结构非常单薄。长此以往，只怕到了晚年也成不了大器。软件开发人员喜欢钻研技术是好事情，但不能轻视“非技术”才能。在开发产品时应该“扬长避短”，而平时培养自己才能时则应该“扬长补短”。

不想当将军的士兵不是好士兵。同理，不想当领导的开发人员也不是优秀的开发人员。开发人员的升迁途径基本上有两条：

（1） 走技术专家的道路，最高可升为总工程师。要求他具备“很高的技术才能”和“较好的表达能力”。

（2） 走管理的道路，最高可升为总裁。要求他具备“较好的技术才能”、“很强的表达

能力”和“很强的管理能力”。

缺乏表达能力和管理能力是软件开发人员的通病，值得业界关注。

表达能力主要是指“写”和“说”的能力。很多开发人员怕写文档和报告，讲述问题和想法时语无伦次。由于表达能力差，他就无法胜任于需求开发、系统设计、管理等高层次的工作。即使他的技术水平很高，但发挥不出来有什么用？只好长期干编程、测试的活。

在允许自由竞争的环境中，如果有人埋怨其才能被“埋没”了，通常是他自己的错。如果真有本事，你就应该自己冒出来，怎么会被“埋没”呢？难道非要等着别人来照顾你不成？

练习“写”和“说”绝对不比钻研技术难。我认为导致“表达能力差”的主要原因是软件开发人员的观念有问题：**他认为表达能力是可有可无的！而技术才能才是唯一重要的。**我认识的大部分默默无闻的开发人员都有这种幼稚的想法。

我一向喜欢头脑灵活、手脚勤快的年轻家伙，希望他们能吸取我“后知后觉”好不容易悟出来的那些道理。如果他们流露出上述“愚蠢”的想法，我会不厌其烦地进行规劝，并让他阅读我写的《大学十年》以期他能“大彻大悟”。如果仍不“懂事”，我不仅要训他还要刺激他：你是不是想一辈子当低层的程序员！真没出息！

有些人把“表达能力差”归结为读小学、中学时文科学得太差，现在补习已经来不及了。这是谬论，我就是极好的“证据”。

我读小学、中学时文科学得极差，高考语文成绩才 54 分（总分 120 分）。我写作文的最高目标就是不逃题，考试前我总是反复祈祷：我没干过坏事，保佑我作文不逃题吧！上大学的第一天我竟然无法用普通话说出“去洗澡怎么走”，只好晃动澡票与辅导员打哑语。可我现在呢？我确信在同行中自己的“笔功”算得上一流。口才也不错，虽然不能把死人说活过来，但花一个小时让蔫蔫的人振作起来还是容易的。

为什么进步那么大？有什么技巧吗？引用卖炭翁的话：“无它，唯手熟尔”。

我读大学十年来一直都是带头干活，经常参加竞赛、展示活动。所以不管是长篇大论还是写标语作宣传，我都自己干（谁能替你做呢？）。这类事情做多了，连笨蛋也会成为行家。我是一个智力平平，标标准准的普通人。既然我能把“表达能力”练得那么好，说明大部分人也能办得到。

同理，管理能力也是练出来的。有人说：“上级不提拔我当项目负责人，我哪有机会锻炼管理能力啊。”

这话就象“在没有学会游泳之前我绝不游泳”那样矛盾。如果你不具备管理能力，你就很难能当上经理。万一你当了经理，并不表示你自动具备管理能力。

这里要强调一点：软件开发领域的管理，其专业性很强，与常见的工商、行政管理很不相似。

如果项目经理对软件一窍不通，他肯定不懂得配置管理、质量保证、软件度量、技术评审等知识，连管理的“门槛”都迈不进去。即便此人是哈佛大学的工商管理硕士也无济于事。

很多人对“管理”有误解，以为就是“搞好关系”。在技术领域，人们都受过高等教育、大部分人想专心搞创造。维持良好的关系本来就是很自然的事，用不着刻意去做。如果需要费力去“搞好关系”，这样的团队还有什么前途！

我一向认为技术蹩脚的人不适合于当软件项目经理，无论他人缘多好，是否会拉关系。企业领导应该挑技术水平较高并且通晓软件工程的人当软件项目经理。

计算机系毕业的人在大学里大都学过软件工程知识，但是学过并不见得就懂。就象我们小时候读、背古代思想家的论著，那时侯能懂什么啊？只有自己的人生经历丰富了，才能品味出哲理来。很多人在小时候会背“三字经”，长大了发现“三字经”里的每一句话都可以写成论文。

我要提醒刚出校门的软件开发人员，只有当你对软件工程的思想方法产生共鸣、感想翩翩时，才算是懂了。如果你没有实践经验，到哪里找共鸣和感想啊。软件开发的全过程中，“技术开发”与“管理”是密不可分、相辅相成的活动，两者都是开发人员“份内”的工作。如果你想成为优秀的软件开发人员，你就应该通晓软件工程，这与你是否当项目经理没有关系。

那些工作了好几年还没有当上项目经理，以及稀里糊涂当了项目经理的人们，应该好好琢磨自己：我还缺什么才能？怎样去提高？

2.8.3 向错误与失败学习

不管是生活或工作，人们都应该向错误与失败学习，目的是让我们在短暂的健康年华中少犯错误、少失败，多做几件正确的对社会有贡献的事。

导致软件项目失败的因素很多，如果不去找借口的话，就会发现错误的根源在自己身上：知识贫乏、才能低下、经验不足、骄傲自负……。我们必须正视自身的不足与缺点，才会学到经验教训。可人们常有太多的虚荣，为了克服心理障碍，白白浪费了很多本该用于创造价值的精力。

假设犯错误的人是诚实的并且是勤奋的，他愿意不带虚荣地改进自己。当这个人突然面对失败时，可能觉得自己一无是处，也许会不知所措，也许会病急乱投医。

程序员都有一种共同的体会：在调试程序时，时常碰到只有十几行的程序竟会产生几十个编译错误；最后发现这么多的错误其实是由某一行程序错误引发的。

当我们在工作中碰到挫折时，先要冷静地分析问题（事出有因哪），找出问题的内因与外因。内因是最主要的，应该最先予以解决。

前几年，中国出现了一个叫“法轮功”的邪教，教徒达数百万之多，人民群众深受其害。全国的主要媒体对“法轮功”进行连续数月的声讨与揭露。目睹了很多受害人的哭诉后，相信人们能够明白“法轮功”是邪恶的、反动的。但在愤怒与心痛之余，我们不禁要反思：为什么那么多人轻信邪教？人们是否接受了教训？

在电视上看到很多人的确作了深刻的检讨：“我真是后悔啊，跟错了李洪志（法轮功的头头）这个坏蛋，我对不起社会……。以后我一定要听党组织的话，党叫我干什么我就干什么，决不上坏人的当。”

我并不觉得这些受害人已经真正醒悟：他只知道法轮功是个邪教，并不知道自己为什么信了邪教。有些事情只要用脑袋去想一想就能分辨是非，可人们就是不去思考，却渴望能跟对“福星”，甘愿把自己的脑袋拴在别人的裤带上。

所以说“迷信”通常是傻子碰到骗子的结果。傻是内因，骗是外因。傻子碰到好人

未必能做出好事，傻子碰到另一个骗子就会做出另一件傻事。为了不让自己“傻”，善良的人们应该用脑子去多学一些知识，努力让自己来把握命运，不要急着把一生托给某个人或某个组织。

软件开发人员在遭受项目失败并开始反省时，不要只是就事论事地仅把眼光锁在特定的项目上，吃一堑应该长好几个智才对。

我在开发软件产品时曾经有不少小的成功，但在 1998 年我遭受了一次很大的失败，使自己的软件公司倒闭（参见附录《大学十年》）。我并不是太爱虚荣的人，知道那次失败是我的毛病积累到一定水准忍不住喷发出来的结果。我绝不能以年纪尚轻不太懂市场与管理为理由轻率地敷衍过去。我把自己察觉到的数十个毛病列出来，日后一个一个克服掉。从失败中吸取教训不仅能长经验，还能使自己懂得如何更好地去做人和做事。以前我“薄积厚发”爱出风头，现在已经能沉下心来，转成了“厚积薄发”。我相信这是进步。

接受批评和改正错误都是应该的，但要注意不要改得“太过分”。有些事情你无法让所有的人都满意，你的特色之中通常包含了不可分割的优点与缺点，你不可以为了改掉某个缺点而把自己的特色砍掉。在我“落魄”的好长一段时间里，我谦虚无比，曾想把任何毛病都改掉，结果又因此而挨了不少训：“改什么改，搞得一点个性都没有，当你几乎没有毛病了，你也成了庸人”。

我不禁想起了电影《大话西游》的一段情景。强盗头子“至尊宝”为了取悦“白晶晶”把自己打扮得斯斯文文，却被“白晶晶”臭骂一通：“瞧你这副模样，一点个性都没有，你还是去干强盗这份很有前途的职业吧！”

假如能回到中学时代，我希望能把文科学好。那时侯盛传“学好数理化，走遍天下都不怕”。我读中学时很无知，鄙视一切文科。不仅语文学得极差，历史、地理课也被我糟踏了，考试时只会填写任课老师某年某月某日在我家乡英勇就义，比谁的成绩更接近零分。更让我沮丧的是，这些行径都不是我发明的，我顶多是个跟屁虫而已，一点回忆的自豪感都没有。扔掉文科只学理科并不等同于“放下包袱，轻装前进”，倒象是摘掉了控制系统的机车，开不了多远就翻车了。在日常生活中，我时常感觉到自己懂得的人文社科知识太少，真是“少壮不努力，老大徒伤悲”。

我搞了十年的软件开发，还没做出令自己自豪的产品来。倒是意外地发现自己的文笔不错，是当作家的料。我发现自己在不该开花的地方结了一颗瘦涩的果子。

曹操之子曹彰曾说：“大丈夫当学卫青、霍去病，立功沙漠，长驱数十万众，纵横天下，何能为博士耶？”

唉，要后悔的事情太多了，只能现在做得勤快些。虽知不成大器，但愿意亡羊补牢，力求学得更深更广。

2.9 小结

本章没有幻想让程序员们成为中国的 Bill Gates，也没有鼓吹可以让程序员一夜之间暴发的妙计，只是论述怎样才能成为出色的程序员。为了剥去阻碍我们进步的那些虚伪，本章讲述了程序员“做人”“做事”的道理，并唠叨了作者自己的一些经历。

中国经历了很多打斗、整人的革命，却没有一次赶上工业革命。在如今国泰民安的

形势下，我们再也不能误了“信息革命”这趟车。90 年代初期，中国出现了一些程序员英雄，曾让我们激动过、崇拜过。但这些孤胆英雄们很快地几乎全消亡了，他们只留下故事，没留下更多别的。再一次让我们意识到“振兴民族软件产业”不能依靠几个人一朝一夕的辉煌。程序员们应当勤奋学习和工作，不仅要图自己能做成几件事情的快意，还要力求事业长盛不衰，这样才能推动整个民族软件产业持久稳健地发展。