# 开发过程调试技术

## 微软公司的软件开发技术

介绍如期开发高质量软件过程中经常遇到的问题和障碍，给出经实践检验切实可行的解决方案和策略

〔美〕Steve Maguire 著
岳晋生　等　译

電子工業出版社
Publishing House of Electronic Industry

# 开发过程调试技术

[美] Steve Maguire 著
岳晋生 等译

电子工业出版社

(京)新登字 055 号

## 内 容 提 要

本书集中讨论了程序员用来以最少的无用工作来使高品质产品上市的技术和策略。从某种观点上讲，本书更侧重于对软件开发队伍的管理策略，而并没有太多讲述与技术有关的内容。

本书的作者曾参与过美国微软公司的软件开发与测试项目，对开发过程的调试技术和对软件开发队伍的管理有着独到的见解。阅读完成书，相信读者一定会有意想不到的收获。

本书适用于软件开发公司的项目主管和技术主管，对计算机公司的高层管理人员，也有极高的参考价值。



**开发过程调试技术**

[美]Steve Maguire 著

岳晋生 等译

责任编辑：王世忠

*

电子工业出版社出版

北京市海淀区万寿路 173 信箱(100036)

电子工业出版社发行 各地新华书店经销

中国电影出版社印刷厂印刷

*

开本：787×1092 毫米 1/32 印张：6.625 字数：143 千字

1995 年 8 月第 1 版 1995 年 8 月第 1 次印刷

印数：1—5000 册 定价：15.00 元

ISBN 7-5053-3067-5/TP・1067

# 前　言

本书可能会使 Microsoft 听起来不再那么美好。

至少，这是我讲述了这么多 Microsoft 的灾难事件的担心之一。我考虑过只对一些事件轻描淡写，或干脆删掉它们，但是除了人名的改变以外，我决定让本书和书中的例子以事实为基础，使它尽可能的有用和具有实践性。况且，我想人们会认识到这点，如果 Microsoft 公司满是新手，那么它也不会在软件工业中有显赫的地位。确实不是。

我叙述的大多数事件来自我对 Microsoft 小组的重新培训的实践，这些小组的项目已经陷入某种麻烦：项目严重误期，代码质量达不到公司的标准，或是程序员疯狂工作仍然毫无进展……。

当与这些小组一起工作时，我发现他们都在犯同样的错误并且他们总是重复这些错误。不仅这样，一旦我习惯于这些小组犯的错误后，我发现甚至是那些项目成功的小组也正在犯一些同样重大的错误——他们只是错误犯得少些或建立了相应的反措施来克服这些错误的影响而已。

在每个我曾与之工作过的小组中，我发现项目主管把他们的时间几乎全部花费在编写代码上面，并且几乎没有人在考虑项目。主管们没有花费时间试图来维护进度，他们不预计会发生的问题，并克服它们，他们没有保护好其他小组成员不被不必要的工作所打扰，他们不特别注意培训其他小组成员，并且他们并不设立详细的项目目标或规划有效的进展计划。

主管们在他们本应思考时花费了太多的时间来工作。

在许多情形里，这种灾难事件状况并不是主管们的错误。这些主管们没有接受足够的培训来当好主管人员。他们只是一些程序员，一天早上醒来突然发现他们自己，因为这个或那个原因，被放置到领导位置上。这些新领导知道如何编好程序，但是他们不知道怎样管理好项目，于是他们专心干他们最熟悉的事情，而对项目放任自流——结果是彻底失败。

不幸的是，许多程序员不觉得他们应该知道如何领导一个项目："我不是主管，那么为什么我要操心领导事务？"他们似乎在想，一旦他们成为领导之后，他们有时间来学习为有效地经营好项目而应做些什么事情。这太晚了。

我写了本书的姐妹篇——《编程精粹》，给程序员们介绍实践验证过的技术和策略，他们可以立即使用这些来编写比他们现在犯的错误少得多的代码。我写《开发过程调试技术》，是给主管人员和程序员们介绍实践证明过的技术和策略，他们可以使用这些技术和策略来组织和经营项目，而不会出现混乱，超时工作和进度误期，这些在我们的领域中太常见了。

按时交付高质量、无错的代码，而不用加班加点——而且干起活来有意思，这些都是可能的。本书中的技术和策略将帮助读者做到这些。

## 致谢

许多许多 Microsoft Press 的人员曾为本书而工作，我不知道他们，或是任何别的出版社的工作人员，是否因他们的幕后工作而获得足够的荣誉，是他们接过手稿并脱稿成书奉献给读者的。我首先要感谢 Mike Halvorson，他在本书仅是构思

时就给予了充分肯定。我要特别感谢 Erin O'Connor，我的手稿编辑人员，同她工作很愉快。她同我一道为本书工作了近一年。本书是我的，同样也是她的。Jett Carey 对本书内容的热情给我极大的鼓舞。Deborah Long 对于成语的敏锐的耳朵，使我比预计的节省了更多的时间，并且她的小组成员，另几位校对员/誊写编辑——Alice Smith，Therese McRae 和 Pat Forgette——敦促我用最适合的方式来写东西。排字员 Peggy Herman 耐心而极富创造性地工作来适合文本和布局的修订，Kim Eggleston 改编她给《编程精粹》一书的绝妙设计，以适用于本书，并在有新要求提出时机智地调整了设计。我感谢所有这些专业人员，以及许多 Microsoft Press 的其他工作人员，是他们将本书付之于实：Judith Bloch，Wallis Bole，Barb Runyan，Jeannie McGivern，Sandi Lage，Shawn Peck，John Sugg，Geri Younggren 和 Dean Holmes。

我将提到两位教育工作者，他们这些年来极大地影响了我对项目的领导方式。Anthony Robbins 开办训练班来帮助 CEOs 和其他的经理人员怎样有效率地经营业务（Robbins 也许因他个人成功的学习班而出名，这些是极好的学习班）。任何熟悉 Robbins 工作的人将会看到他对全书中观点的影响。Michael E. Gerber 给商业听众讲述如何“找回美国商业之梦”并著述《The E-Myth：Why Most Business Don't Work and What to Do About It》。Gerber 的思想、书和演讲看起来似乎同编写软件毫无关系——它们主要是有关如何营运特权公司——但是 Gerber 的许多见解完全改变了我对软件工程以及它们应该如何运作的看法。我要感谢这两位教育工作者。有关他们公司的信息在本书书末的“参考资料”中提及。

我有幸拥有一个绝妙的小组，小组里的程序员和项目主

管是本书的评论者。他们慷慨地从他们的经验和见识里提供宝贵的意见。他们是 Microsoft 的程序员和项目主管，多年来一直为 Microsoft 的大多数重大项目而工作。作为评论人员，他们确使我的建议中充分考虑到了他们在项目中遇到过的问题和事情。我感谢 Paul W. Davis, Melissa Glerum, Eric Schlegel 和 Alex Tilles. 我要特别感谢 Dave Moore，他用他作为程序员、项目主管、部门经理和更近些时的总经理以来长期积累的经验来帮助我完善本书中的观点。我也要感谢 Ian Cargill，他在我写这本书的早些时候提供了几点重要的见解。

西雅图　华盛顿

1994. 6. 21

# 简　介

善于鼓舞人心的领导用一种有趣的方式来看待世界。公司大楼被烧垮了，他们不会因担心失业而惊慌失措，这些善于鼓舞人心的领导看着熊熊燃烧的火焰，拿出了热狗和软糖。当围着他们的每一个人都感到悲观失望的时候，这些领导在激发起人们的自信，尽管每个理由都应让人感到悲观。他们是一群乐观的家伙，喜欢从好的方面来解释事情。这些有眼光的、善于激励人们斗志的领导不喜欢把失败看作是失败，而是仅把失败当作是学习的经验，这会帮助他们克服到来的下一个障碍。并且因为这些领导不会有失败的感觉，他们会涌出一些稀奇古怪的想法，可能导致重大的突破。如果一种奇特的想法归于失败，这些善于鼓舞人心的领导不把这件事情看作是失败，而是仅当作提供了更多的有用信息。这种领导才能同经验毫不相干。它是各种品质的综合物：坚强的意志，看待世界和机会的独特方式，以及清晰的幻想力，以及把这种幻想传递给旁人以激励他们团结在领导的周围来让梦想成真的能力。

尽管人们相信这类领导是天生的而不是后天培养的，但还是可以通过学习来成为这种善于激励人们斗志的领导。然而，这并不容易。通常，这个人必须改变他或她的主要信仰和态度来以那种独特的方式看待世界。你可能会说，这要求对个性进行改造——对于这种想法，大多数人会认为不可能而且许多人会对此有反感。我想这就是为什么极少有人在人生的中途转变成这类有号召力领导的缘故。人们通常不会这样急

剧地改变个性。

## 剩余我们这些人

幸运的是，大多数的软件项目主管不是在一片未规划的领土上开创一家新公司或从事冒险行动的。典型的主管人员通常是着手开发一个应用的 4.21 版，或是一些有相当明朗前景的项目，这前景是每个人都大体赞同的。一般的软件主管不需要是极有感召力的领导来让小组成员们干一些稀奇古怪的事情。他们仅需要有效率，这是可以学会的，而且不要求任何个性的改变。只要求学会一些习惯和策略，这些习惯和策略可以按时让高质量的产品上市——而不用一周工作 80 小时。

所有高效率的领导明白，要使一个项目取得成功，小组的每一位成员必须依策略行事，这些策略用来促使高质量的产品按期完工。读者不一定要是主管人员才能使用我这里描述的技术和策略。本书适用于每位小组成员，而不只是主管人员。除非每一位小组成员知道怎样做才能让高品质的产品上市而不用一周工作 80 小时，否则这不会发生。

## 编程精粹

开发队伍努力让产品上市的过程中要涉及许多步骤——从代码设计到同市场小组合作过程中的每一件事情。在开发过程中的每一个步骤，都可能犯错误。这在观察中没有什么新鲜的东西。

我将本书取名为《开发过程调试技术》，是让程序员如同他们考虑编程算法一样考虑开发过程：开发过程中可能会有

错误，造成浪费和错误导向的工作，这可以进行优化，从而工作得更好。

在本书的姐妹篇《编程精粹》一书中，我集中论述了我认为在开发过程中是最为重大的“错误”：有太多的编程错误。《编程精粹》一书叙述了一些技术和策略，程序员可以用这些尽可能早地查出错误，并叙述了程序员如何在开始时预防这些错误。

在《开发过程调试技术》一书中，我集中论述了程序员用来以最少的无用工作使高品质产品上市的技术和策略。在头三章中，我谈到一些基本概念和策略，一小组如果想发布产品而不用一天工作十二小时，一周工作七天，他们应该采用这些概念和策略。最后五章基于前三章，主要是有关企业过程的过度膨胀、进度的复杂性、程序员培训、态度以及长时间工作。

《编程精粹》和《开发过程调试技术》是姐妹篇。读者会发现两本书中的思想在一定程度中相互交溶。当两本书中的思想重叠时，读者会发现《编程精粹》一书更集中于代码本身。一个实例是，我从《编程精粹》一书中摘取了一个章节的一部分到本书中，因为我认为它说明的观点在平稳进行项目时比在《编程精粹》中更为重要。

## Microsoft 的软件开发——快照

本书中的大多数例子来自我在 Microsoft 的经验。对于 Microsoft 公司里如何在主管人员之间划分责任进行一个简短的描述以及一个典型项目如何进行的概貌可能会帮助读者理解这些例子的背景。

一个 Microsoft 的项目一般至少有三种不同的主管人员

直接为产品的开发而工作。

- 项目主管。项目主管最终为代码负责。他或她也负责开发和监督进度，使项目按预定计划进行、培训程序员、为上层管理进行程序检查，等等。项目主管通常是小组中最有经验的程序员并且常写代码，但只是把这当作第二手的工作。
- 技术主管。技术主管是小组里的程序员之一，他比其他任何人都更了解产品的代码。技术主管负责产品的内容完整性，使所有的新功能都用脑中已有的代码来设计。他或她通常也负责确保项目的所有技术文档是最新的：文件格式文档、内部设计文档，等等。如同项目主管一样，技术主管通常是小组中最有经验的程序员之一。
- 程序经理。程序经理负责协调同市场、文档，测试和产品支持等关系产品的开发工作。简短的说，程序经理的工作是监督产品——每件放在产品封装盒中的东西——的完成，并且是以公司希望的质量标准完成。程序经理通常同产品支持小组一起工作来协调产品的外部beta 版的发布，并同最终用户一起工作来看怎样提高产品。程序经理通常自己也是程序员，但他们的编程工作限于使用产品的宏语言（如果有的话）来编写“指南”（wizards）和其他有用的最终用户宏。不是别人，正是程序经理负责产品应是什么样的“构想”。

“程序经理”的名字可能让人误解，因为它暗示程序经理在职位上高于项目主管、测试主管、文档主管和市场主管。实际上，程序经理是同其他主管位于同一层次的职位。程序经理的更合适的名字应是“产品主管”，因为程序经理负责确保产

品的所有部分——不仅是代码——按进度完成并且是按可以接受的质量标准完成。

在一个典型的项目中，程序经理（如果项目足够大的话，则是经理）在前期同市场、开发和产品支持小组一起工作，提出产品的改进列表。在创建功能列表之后，程序经理撰写产品说明书，它详细描绘每项功能应该怎样提供给用户——例如，提供一个新对话框的绘制方法以及如何使用它，或是提供一个新宏函数的名字以及它的参数描述。一旦产品说明书的草图完成之后，将把它传递给该产品涉及的所有小组，进行彻底的检查。一旦最终说明书确定下来，各小组就开始工作。

程序经理同时使用功能的实体模型来进行可用性研究，以确保所有这些新功能直观上同每个人开始想的那样易于使用。如果一项功能显得难以使用，程序经理建议对说明书进行修改。程序经理也为产品盘的样例文档和我前面提到的最终用户宏而工作。当功能完成时，他或她将逐项检查以确保它满足交付产品的所有质量标准——特别是，功能在低档机上也有足够的活力。

开发工作继续向前，并最终到达一点，称之为"可视冰点"（Visual freeze），是指所有会影响显示的功能都完成了。一旦代码到达可视冰点，用户手册将用程序运行时的屏幕画面来写成。结果是，从该点开始，开发人员要小心不要以任何方式影响显示，从而使手册中的屏幕画面同程序运行时用户看到的画面保持一致。当然，程序员希望在所有代码完成之后再截取屏幕画面，但是手册需要很长的时间，在代码完成之前必须送去印刷。在一些情形里，为了让所有功能及时达到可视冰点以便在产品发布时能准备好手册，程序员仅部分地完成功能——例如，显示一个无功能的对话框，只用于屏幕截取而不干

别的事情。以后，程序员再返回来全部地完成它们。

一旦所有的功能完成之后——“代码完成”阶段——程序员将努力修正错误列表中的所有重大的错误，并做一些必要的功能上的改进。当代码最后准备交付时，项目主管或技术主管生成“金主盘”。程序经理将金主盘送去复制生产，并将软件和手册、注册卡和其他东西一起装盒。稍稍压紧并包装之后，产品就准备好可以出售给用户了。

我省掉了许多细节，但是这样简短的总体描述足以让你理解本书中偶尔出现的可能过于 Microsoft 化的例子的背景。

我也该指出，e－mail 是 Microsoft 的生命血液，所有内部事务是通过 e－mail 进行的。并且，至少是在开发周期中，你必须真正有充足的理由才能用电话打断他人的工作。开发人员之间的大多数交流通过 e-mail 进行，以及在许多自发进行的大厅会议中进行。这种全体对打断他人工作的敏感性很好地说明了 Microsoft 的政策：给每个人一个带门的个人办公室。如果你在工作并且不想被打扰，你只要关上门就行。

## 做起来比听起来难

我最后的担心是，本书可能让人听起来似乎应用了它提供的所有建议后，一夜之间，能转变一个不完美的项目。你确实可以立即应用书中的许多技术和策略，可以很快取得效果；但是其他的——例如培训技巧——要花上一段时间才能有效果。如果你的小组现在出了问题，则不能指望读完本书一星期后就将项目扳转过来。以我的经验来说，扭转一个遇到麻烦的项目要两到六个月时间，大多数的好转发生在头两个月里。然后，进一步的好转来得很慢并且幅度不大。

# 目　录



如果一个软件项目想要取得成功，每位小组成员都必须充分理解可以促使高质量的软件按时交付使用的原理、指导方针和策略，本书是为每一位小组成员而写的。它是《编程精粹》一书的姐妹篇，《编程精粹》一书主要讲述开发过程中的最为严重的“错误”：有太多的软件错误。本书中的建议用来调整好开发过程，主要讲述一些技术和策略，软件队伍用这些来取得不断的成功。本书用许多轶事作例子，它们中的大多数来自 Microsoft 的实践。为使这些例子易于理解，在简介中简单地讲述了在 Microsoft 是如何组织软件开发项目和项目是如何进行的。



有几点原则是所有取得成功的软件项目主管人员牢记脑海中的。最重要的思想是程序员应该做那些直接或间接提高产品质量的工作。为其他小组成员的主要工作开道是主管人员的职责，主管人员要无情地去除那些阻挡提高产品质量的工作——例如，过分追求进展报告和会议，或开发对产品或公司不是很重要的功能。为容易判定哪些任务是重要的而哪些是白费力气，主管人员应该建立详细的项目目标和优先级规定。项目目标和编程优先级规定越详细，就越容易找出无用的

工作。



令人惊奇的是，相当细微的工作习惯或过程如何能够导致结果的截然不同。确实，这些习惯或过程不用花费半点力气就可付诸实践，而且它产生的效果也不依赖于使用它的程序员的技能水平。为了获取最好策略而有效率地工作，主管人员应该把他们试图要解决的问题形成不断细化的提问。例如，主管人员不能这样问，“我们怎样总是按期交付产品？”，它可能导致许多不期望的回答。相反，主管人员应该问一个更明确，更有益的问题：“我们怎样总是按期交付产品，而不用雇用更多的人和迫使开发人员超时工作？”主管人员应该试图将负反馈循环并入到他们采用的策略中。并且当他们把工作策略传授给小组其他成员时，他们应该确保提醒小组，一个好的策略和指导方针不必要在所有情况下都有效。



项目可能以如此之多的微妙方式误入歧途，主管们永远不能对项目放任自流，以为它们会按计划进行并会自己运转。为使一个项目平稳运转，主管必须时时监视着项目，目光要向前看，在问题尚小的时候解决它们。为促使项目按期交付，主管每天要问自己这个问题，“我今天可以做什么事情来让项目在下几月里按预定轨道前进？”通过每天询问自己这个问题并认真寻找答案，主管可能预见到各种问题，否则这些问题会阻碍项目。为了防止浪费人力，主管应该审评每项请求，来辨明真正的问题或目标，而且应该确保每项任务是实现项目的目标和遵循优先级规定。有一些任务，可能根本不具备战略重要

性，例如满足市场小组要求扩充功能集的请求，或是实现从程序员的设计中冒出来的不花钱的功能。一位好主管应学会说“不”。



如果主管希望软件开发小组创造性地活动，他或她必须创建一种开发氛围，孕育这种创造激情。不幸的是，当公司从小小夫妻店发展成大企业公司时，程序员日常担负的非开发性工作激剧增多。主管应该努力工作消除不必要的报告和会议，以及有碍开发工作的企业事务。这里，事务越简单越好。如果程序员有机会好好工作而不被过多的企业事务所防碍，他们有更好的机会抓住灵感的火花并促使项目向前进行。重要的一点是，主管人员应该总是讲明他们真正的（而不是正式的）需要。要求写报告或召开会议是收集信息的一般方法，但是如果有其它更有效的办法来收集信息（确有），为什么要把报告和会议压在程序员肩上呢？



在大多数公司里，开发小组要维持某一进度，这样，公司里别的组就可以配合他们的工作。最少，市场组应该知道他们该何时开始为产品做广告，但正如计划对于协调公司内各个小组的工作非常重要一样，如果计划制订或执行得不理想的话，它也会给开发造成灾难性的影响。一旦不能达到计划，就会打击开发小组的士气，最终会损害其工作量。太紧迫的计划会导致歇斯底里症，程序员为达到短期的计划会采取短期行为，从而使产品从长远来看面临危险。一个项目计划应该足够紧迫，使得项目能快速进行，但如果计划过于紧迫，程序员们

就会不去理会他们正确的判断而做出愚蠢的决定。任何程序员如果认定他没有足够的时间来详尽地测试其代码，就犯了把计划凌驾于产品之上的错误。使用“阶段性计划”，项目负责人不仅可以更好地协调其它几个小组的工作，还能使项目更令人兴奋，培养一种富有创造性的气氛，使程序员以惊人的速度写出高质量的代码。



项目负责人可以将开发过程优化到某一点上，在这个水平上，每个小组成员只将注意力集中在富有战略意义的工作上。但如果负责人真想项目腾飞，他就要关注人员培训，使每个小组成员经常性地学习一些广泛使用的新技术。一种能保证小组成员主动地提高自己的办法就是将个人成长目标与第五章中提到的两个月的项目阶段计划统一起来，这样可使每个程序员每年至少学会 6 种新技能，程序员在工作过程中能够，也确实学到了新的技术，但这种被动的学习方式速度要慢得多。要通过工作分配和明确的教育目标，确保程序员能主动地学习新的技能，从而使项目和公司受益，还可以促进程序员职业生涯的发展。



通过主动学习来提高小组成员的技能很好，但当项目负责人着眼于改正不正确的态度和提倡有益的态度时，他就可以获得更加令人瞩目的结果。新的态度其作用力将遍及程序员要做的每一件工作，这就是好的态度的杠杆作用。第七章严肃地审视了导致项目失败的一些程序员中普遍存在的态度：错误难免，我晚些时候再改正错误，要把事情做好花的时间太