HZ BOOKS
华章科技

Microsoft

《代码大全》姊妹篇，资深软件开发专家30余年工作经验结晶，微软公司软件工程师必读之书，被誉为“软件行业的财富”

从软件开发流程、技术、方法、项目管理、团队管理、人际沟通等多角度总结出90余个具有代表性的问题并给出了解决方案，值得所有软件工程师研读

华章程序员书库

I. M. Wright's "Hard Code"

A Decade of Hard-Won Lessons from Microsoft

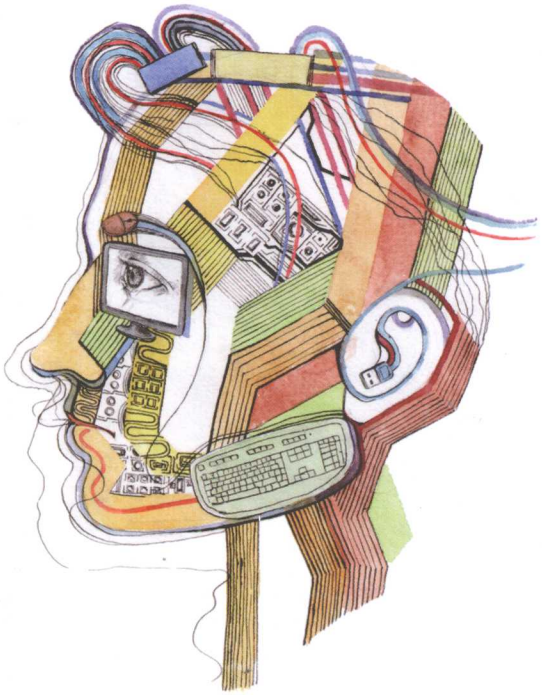
Second Edition

代码之殇

（原书第2版）

（美）Eric Brechner 著

林锋 译



机械工业出版社

China Machine Press

序员书库

I. M. Wright's "Hard Code"

A Decade of Hard-Won Lessons from Microsoft

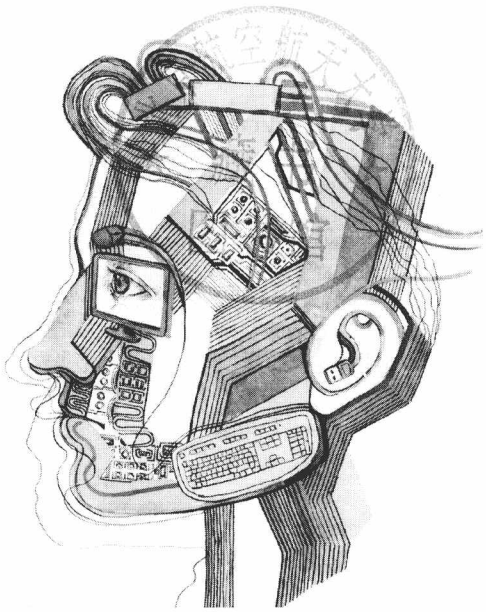
Second Edition

代码之殇

（原书第2版）

（美）Eric Brechner 著

林锋 译



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

代码之殇（原书第2版）/（美）布莱什纳（Brechner, E.）著；林锋译. —北京：机械工业出版社，2013.4
（华章程序员书库）
书名原文：I. M. Wright's "Hard Code"：A Decade of Hard-Won Lessons from Microsoft，Second Edition
ISBN 978-7-111-41682-1
Ⅰ. 代… Ⅱ. ①布… ②林… Ⅲ. 软件开发 Ⅳ. TP311.52
中国版本图书馆CIP数据核字（2013）第039323号

本书是《代码大全》的姊妹篇，资深软件开发专家30余年工作经验结晶，被誉为"软件行业的财富"，微软公司软件工程师必读之书。它从软件开发流程、技术、方法、项目管理、团队管理、人际沟通等多角度总结出90余个具有代表性的问题（大多数问题可能会给公司或软件项目带来毁灭性灾难），并给出了问题的解决方案和最佳实践，值得所有软件工程师和项目管理者研读。

本书将这90余个问题分为10章：第1章讨论如何通过管理风险、范围和沟通来保障项目按时完成；第2章介绍消除经验主义的大量过程改进的方法与技巧；第3章讨论消除低效率的策略；第4章主要讨论开发者与其他工种之间的关系；第5章重点阐释软件质量问题；第6章解析软件设计的基本原理和错综复杂的本性；第7章探讨如何规划职业生涯；第8章分析工作与生活中存在的缺点的原因与纠正措施；第9章讨论如何进行有效管理；第10章分析如何成功应对一个软件业务所面临的挑战。

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：秦 健
北京市荣盛彩色印刷有限公司印刷
2013年5月第1版第1次印刷
186mm×240mm · 20.75印张
标准书号：ISBN 978-7-111-41682-1
定 价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

本书赞誉

任何大型组织都会有成为自身文化牺牲品的危险。关于业务模式及行为准则的神话想当然地成为金科玉律。任何组织都会存在这种倾向，但对于需要不断创新才能兴旺的技术公司来说，这却是致命杀手。Eric Brechner 做了件难以置信的事——他亮出了手术刀，深度剖析了存在于组织中的这种谬误。他毫不客气地亮出了利剑——往往一针见血。尽管有一些隐语和例子对于微软内部的员工更具吸引力，但他的智慧和至理名言，大都可以成为整个软件行业的财富。

——Clemens Szyperski，首席架构师

I. M. Wright 写的关于开发时间表的文章真的太棒了！它在我所属项目组参与的基础项目上同样适用。

——Ian Puttergill，项目经理

你不打算遇到死亡威胁这样的事，是吧？

——Tracey Meltzer，高级测试主管

这很可笑——很坦率地说，这类十足的谬论很危险。

——Chad Dellinger，企业架构师

Eric 是我个人崇拜的英雄——很大程度上是因为他长期以来一直代表着开发社区的一种声音。

——Chad Dellinger，企业架构师

软件工程师很容易就会迷失在代码中，更糟糕的是，甚至会迷失在过程中。此时迫切需要 Eric 在本书中提出的实用建议。

——David Greenspoon，总经理

我刚刚读完这个月的栏目……我不得不指出，这是我第一次认为你正在推行一个对公司完全错误并且带有灾难性的想法。

——David Greenspoon，总经理

你太有才了，Eric。几个月之前，我跟产品单元经理以及一些开发主管恰恰进行过一次这样的对话。好主意。

——Scott Cottrille，首席开发经理

我们真的很喜欢这些栏目。它们不仅实用，而且很全面！我喜欢它们的另外一个原因是，当我在指导初级开发人员的时候，我可以把这些栏目推荐给他们；他们也会记住这些栏目，因为它们都是那么有趣。

——Malia Ansberry，高级软件工程师

Eric，干得好！我觉得你在这个栏目中说得非常中肯。我想，该给管理者传递这样的信息——“不要害怕尝试。”事情的真实情况跟理想化的理论之间差别是非常大的。

——Bob Fries，合作伙伴开发经理

我只是想让你知道我有多喜欢你写的文章——它们充满智慧，见解深刻，你还神奇地把本来很严肃的问题变得如此有趣（采用的方法很不错）。

——Niels Hilmar Madsen，开发者传教士

你那篇关于死亡行军的栏目来得正是时候。我们正打算在未来几周内开会讨论功能削减的事情呢！那些我们以前付出很大代价才学到的教训，不知怎么回事，总是很容易就忘记了；你的栏目对大家起到了很好的提醒作用。

——Bruce Morgan，首席开发经理

我想让你知道的是，我真的很喜欢并感谢你在 EE 站点上发表的所有文章。不过，直到今天，当我读了“停止写规范书”这个栏目之后，我不得不说，我强烈不同意你的观点。

——Cheng Wei，项目经理

你到底是谁？你跟 Eric Brechner 都做了些什么？

——Olof Hellman，软件工程师

Eric，我刚刚读完你写的那篇叫“不可攀比”的文章。你不知道我有多感激你！你实际上把这个观点传递给了公司里面成千上万的人……你致力于正确领导和管理团队，并把其中的奥秘跟大家分享，对于你的这种热情我真的非常欣赏！

——Teresa Horgan，商务项目经理

译 者 序

没想过真的可以翻译完成一本书，但天性赋有一颗充满好奇的心，闲时，网游、肥皂剧又非我所好，就喜读读书，摆弄摆弄技术，既然喜欢，与其像他人在网游中找乐，不如一直好奇着读书倒腾技术为乐，而且总有所得。早年看侯捷先生的译作，我就在想，如果有一天我也能翻译，既可借翻译之机广为涉猎，又可担当传播学识、价值观的角色，岂不幸甚？今天，终于有了这样一个机会。

IT 一词，通常让人联想到的是那些技术极客——一群苦大仇深的码农形象，就像本书的书名一样被人误解。其实，软件开发更体现的是一种团队合作、企业文化乃至一人为人处世的态度。本书不是高屋建瓴、看不见摸不着的软件工程理论教科书。作者是微软的一名资深项目经理，本书内容是他从日常工作生活的点滴体会中总结出来的。他不仅把软件工程的一些原理技巧在实践中加以诠释，同时阐述了如何处理团队成员之间的关系，如何面对职业生涯规划，如何树立正确的工作生活态度。

感谢我的挚友，曾晋瑞。他是一个乐于工作，享受生活，激情四射的人；凡事严谨，追求完美，理想主义的人。当大家都在追问中国如何才能出个乔布斯的时候，我只感叹中国没这土壤，否则曾晋瑞就是。当我还未正式翻译此书的时候，他就对我说："林锋，你翻译到哪了，你这书可要好好翻译，起码得从头到尾修改十遍。"我只能表示!!! -_-。他在本书的翻译过程中给予我莫大帮助，一旦我遇到晦涩难懂的内容，只要我求助于他，他就能从亢奋的工作状态中抽出时间，聊上好长时间；也会跑到我家，花上半天的时间，为的只是讨论某段文字的确切意思，甚至用哪个词更为精妙。这里再次表示感谢！

这是本人翻译的第一本书，且原书很多专业术语出自微软内部，译文中可能存在不少错误，误漏之处，请不吝指正，想必，这样的积累日后定能为读者带来更高质量的译本。

最后，分享一句译完本书的个人感悟：你的理想不用很具体，只要你有那份热忱，即使你看不清未来是什么，你也会像我一样，向最初我的景仰人物侯捷先生迈进一步。

如愿与本书译者进一步沟通，请 E-mail：xlfg@ yeah. net，或新浪微博：@ 大黄蜂的思索。

林锋

序

如果你想了解礼仪，你会直接光顾网站 Miss Manners；如果你在恋爱上遇到什么麻烦，你可能就会浏览论坛 Dear Abby；如果你想了解微软到底出了什么状况以及一位名叫 I. M. Wright 的大个头到底是怎么办事的，那么这本书就是为你而写的。I. M. Wright 就是在微软众所周知的 Eric Brechner。

软件开发是项颇具挑战性的工作。我已经把它当成一种开创性的团队运动，这项工作不仅需要你记住之前做过什么，同时要记住没干过什么。我在微软工作的那段时间，Eric 是我的知音。每当我受挫或失意的时候，通常不用我主动向他诉求，他似乎总知道该跟我讲些什么来帮助我；当我遇到难题需要帮助的时候，一个 I. M. Wright 专栏就会出现，它正是我所关心的问题之所需，这些问题在微软及其他软件开发机构都很常见。

一本《代码之殇》（Hard Code）在手就好比你办公室四周角落都呆着个 Eric。在应对变化时你遇到麻烦了吗？Eric 就是答案；你的团队缺乏斗志了吗？我笃定 Eric 对你总有好建议；质量问题困扰着你的代码吗？我知道 Eric 可以帮助你。通过使用笔名 I. M. Wright，Eric 以一种轻松愉快的方式为你解决软件开发问题，他在促使你思考的同时让你笑逐颜开。

但 Eric 并不只写一些你可能遇到的问题，他也萃取一些他所见到的公司的成功经验作为教程，这些公司开发并发布的产品及服务由全球数百万用户所使用。第 2 版囊括了丰富的最新建议及成功案例——它们很有价值，我经常将其作为我客户的必读之物。《代码之殇》是种珍宝，为人人书架之必备。

Mitch Lacey

本书顾问，微软前雇员，现供职于麦切·莱西联合有限公司

2011 年 5 月

前　言

献给当初对我说“为什么不由你来写”的人 Bill Bowlus。

献给当初对我说“好的，我帮你编辑一下”的人我的妻子。

你手上拿着的是一本关于最佳实务的书，它会比较乏味，但也许会有点吸引力，你能从中得到知识，读后甚至对你产生些许影响，但读起来肯定是干巴巴而无趣的。为什么这么说呢？

最佳实务的书是乏味的，因为这个“最佳”是跟具体的项目、具体的人、他们的目标以及偏好紧密相关的。一个实务是不是“最佳”，大家可能看法不一。作者必须把实务列举出来让读者自己选，并分析在什么时候、什么原因选择哪个方案作为“最佳”。但是这种做法是现实的、是非分明的，它会让人感到乏味和厌烦。通过多个案例的研究使原本模棱两可的概念泾渭分明，从而会使文字有味一些，但作者仍必须把选择的机会留给读者，否则作者就会显得傲慢、教条并且死板。

然而，人们喜欢看到傲慢、教条、死板的学者之间的针锋相对。大家喜欢引用学者们的观点片段，与朋友和同事一起讨论。为什么不将对这些最佳实务的争论作为观点栏目来发表呢？唯一的条件就是只要有人愿意将自己扮演成一个思想保守的傻瓜。

本书的由来

2001 年 4 月，在历经了 Bank Leumi、Jet Propulsion Laboratory、GRAFTEK、Silicon Graphics 及 Boeing 等公司总共 16 年的职业程序员生涯，并在微软做了 6 年的程序员和经理之后，我转而加入了微软内部的一个以在公司范围内传播最佳实务为职责的团队。当时这个组正在运作发行一个名叫《Interface》的月刊网络杂志的项目。它很有意义且富有知识性，但同样也是干巴巴而无趣的。我那时建议增加一个观点栏目。

我的上司 Bill Bowlus 建议由我来写。我拒绝了。作为一个半大孩子，我努力使自己成为一个协调员，撮合多方产生成果，而成为一个爱唠叨的实务学者会毁掉我的名誉和效力。因此，我当时的想法是说服一个大家公认的偏执的工程师来写，他可能是我在微软 6 年工作经历中接触过的一位特别固执的开发经理。

但 Bill 指出，我有 22 年的开发经验，4 年的开发管理经验，写作技巧也还行，而且有足够的态度来做这件事，我只需要放下自身的心理包袱。另外，其他的开发经理都忙于常规的工作，不可能每个月来为我们写一些个人观点。最后 Bill 和我想出了一个用笔名撰稿的点子，于是“代码之殇”（Hard Code）栏目诞生了。

从 2001 年 6 月开始，我使用“I. M. Wright，微软逍遥的开发经理”（I. M. Wright，Microsoft development manager at large）这个署名为微软的开发者和他们的经理写了 91 个“代码之殇”观点栏目。这些栏目的标签行都打上了“绝对诚实，直言不讳”（Brutally honest，no pulled pun-

ches）的标语。每个月，有成千上万的微软工程师和经理会阅读这些栏目。

前16个栏目在内部网络杂志《Interface》上发表了。这些栏目都是编辑（Mark Ashley和Liza White）给我分配的。我和《Interface》的美工Todd Timmcke还一起制作了作者的很多搞怪照片。当网络杂志停刊的时候，我才得以有喘息的机会，但也停止了写作。

14个月之后，在我们组的编辑Amy Hamilton（Blair）、Dia Reeves、Linda Caputo、Shannon Evans和Marc Wilson的帮助下，我又开始在内部站点上发表我的栏目。2006年11月，我将所有的栏目转移到一个内部的SharePoint博客上。

2007年春天，正当我打算度过几年前奖励给我的假期的时候，我现在的经理Cedric Coco给了我在休假期间将《代码之殇》出版成书的授权。而微软出版社的Ben Ryan也同意了。同年，本书第1版出版。

除了我已经提及的人，我还想感谢《Interface》的其他成员（Susan Fario、Bruce Fenske、Ann Hoegemeier、John Spilker和John Swenson），其他帮助本书出版的人（Suzanne Sowinska、Alex Blanton、Scott Berkun、Devon Musgrave和Valerie Wolley），支持我的管理层（Cedric Coco、Scott Charney和John Devaan），曾审核过我的栏目并提出过很多主题的团队成员（William Adams、Alan Auerbach、Adam Barr、Eric Bush、Scott Cheney、Jennifer Hamilton、Corey Ladas、David Norris、Bernie Thompson、James Waletzky、Don Willits和Mitch Wyle），以及写下第1版“前言”的Mike Zintel。关于第2版，我想重点提下本书的审核人员及长期读者（Adam Barr、Bill Hanlon、Bulent Elmaci，Clemens Szyperski、Curt Carpenter、David Anson、David Berg、David Norris、Eric Bush、Irada Sadykhova、James Waletzky、J. D. Meier、Jan Nelson、Jennifer Hamilton、Josh Lindquist、Kent Sullivan、Matt Ruhlen、Michael Hunter、Mitchell Wyle、Philip Su、Rahim Sidi、Robert Deupree（Jr.）、William Adams和William Lees）；还有James Waletzky，他在我休假的时候为我的读者写了两篇专栏文章；Adam Barr和Robert Deupree（Jr.），他们发动我为专栏记录了一段播客并发布出去；Devon Musgrave和Valerie Woolley，是他们让第2版顺利出版；我的上司（Peter Loforte和Curt Steeb）对我的努力给予了莫大的支持；Mitch为我写了第2版的“序”；以及我的妻子Karen，当我让编辑加入Xbox. com工作时，她接手了我的专栏编辑工作。

最后，我要感谢我才华出众的中学英语老师（Alan Shapiro），以及那些慷慨给予我反馈的读者们。尤其是，我还要感谢妻子Karen和儿子Alex和Peter，他们让我做任何事情都充满信心。

本书的读者对象

组成本书的91个观点栏目最初是写给微软的开发者及其经理看的，尽管它们也是我过去在软件行业6个不同的公司、32年的工作经验中提炼出来的。编辑和我一起修改了语言表达，注解了那些微软内部的特殊用语，使得本书适合于所有软件工程师和工程经理阅读。

我在这些栏目中表达的观点是我个人的，不代表我现在和以前任职过的任何一家公司，包括微软。我在栏目中的注解以及本简介中的言论同样都是我个人的，与任何公司无关。

本书的结构

根据主题的不同，我把所有栏目分成10章。前6章剖析了软件开发流程，接下来3章重点讨论人的问题，最后1章批判软件业务运转方式。解决这些问题的工具、技巧和建议遍布全书。本书的最后还附有术语表方便大家参考。

每一章的各个栏目均按照当初在微软内部发表的时间顺序排列。每章开头我都给出了一个简短的介绍，随后就是当初我以“I. M. Wright”笔名发表的栏目内容。当将其编辑成书的时候，我还适时在栏目中加上了“作者注”，以解释微软的术语，提供更新内容或者额外的背景知识。

编辑和我尽力保持原有栏目的完整性。我们做的工作仅仅是纠正语法和内部引用。称得上改动的其实只有一处：就是将原来一个叫“你被解雇了”的栏目标题改成了“最难做的工作——绩效不佳者”，因为以前那个标题太容易让人误解了。

每个栏目都以一段激昂的演说开场，然后是问题根源的分析，最后以我对这个问题如何改善的建议结束。我酷爱文字游戏、头韵和通俗文化，因此栏目中充斥着这些东西。特别是大部分栏目的标题和副标题都直接取材于歌词、电影对白和谚语。是的，我自娱自乐，但撰写这些栏目确实给我带来了些许乐趣，同时也使我得到了痛快宣泄。希望你也会喜欢！

微软的组织结构

因为这些栏目最初是写给微软的内部员工看的，因此有必要简要了解一下微软以及我在工作中扮演的角色，这有助于更好地理解这些文字。

目前，微软的产品开发分成七大业务部门，各个部门相应负责我们的主营产品——Windows、Office、Windows Phone、Interactive Entertainment（包括 Xbox）、服务器软件及工具（包括 Windows Server 和 Visual Studio）、Dynamics 以及在线服务软件（包括 Bing 与 MSN）。

每个部门大约包含 20 个独立的产品单元或管理小组。通常情况下，这些产品单元共享源代码控制、创建、安装、工作条目跟踪和项目协调（包括价值主张、里程碑安排、发布管理和持续性工程[⊖]）。除了这些相应的服务之外，产品单元或小组还有高度的自主权，可以对产品、流程和人员做出自己的安排。

一个典型的管理小组通常由三个专职经理组成：项目组项目经理（Group Program Manager，GPM）、开发经理（Development Manager）和测试经理（Test Manager）。一个产品开发单元通过这三个专职经理向产品单元经理（Product Unit Manager，PUM）负责。如果没有产品单元经理，那么他们分别向他们的上司并最终向部门主管汇报。其他工程领域，比如用户体验、内容发布（比如在线帮助）、创建和实施，这些可能单独对某个产品单元负责，也可能在整个部门中共享。

每个工程领域抽出一个或多个代表组成一个虚拟团队，由该团队向这三个专职经理负责并为一个单独的功能模块工作，该团队称为功能团队（feature team）。有些功能团队选择敏捷方法，有些喜欢精益模型，有些采用传统的软件工程模型，有些则根据实际情况综合采用上述多种方法。

微软如何整合所有这些多样化又独立自治的团队并使其朝向一个共同的目标有效地工作呢？这就是部门公共项目协调组所要扮演的角色了。例如，部门的价值主张是为所有的专职管理小组和他们的功能团队设置统一的关键示例、质量尺度和准则。

示范工具和文档

本书提到的称为“在线资料”的示范工具和文档，都可以通过如下地址下载：

http：//go. microsoft. com/FWLink/？Linkid 220641

⊖ 持续性工程（sustained engineering）指直接发布软件产品而不像通常先出测试版改进后再出正式版，产品发布后根据用户使用情况再改进，然后再次发布新版本，如此反复。——译者注

在线资料列表

工　具	栏　目	章节
SprintBacklogExample. xls；SprintBacklogTemplate. xlt	敏捷子弹	2
ProductBacklogExample. xls；ProductBacklogTemplate. xlt	敏捷子弹	2
Spectemplate. doc；Speccheckl ist. doc	糟糕的规范书：该指责谁？	3
InspectionWorksheetExample. xls；InspectionWorksheetTemplate. xlt；Pugh Concept Selection Example. xls	复审一下这个	5
InterviewRolePlaying. doc	面试流程之外	9

系统要求

本书提供的工具都是微软的 Office Excel 2003 和 Office Word 2003 格式的。只要你的电脑上安装有 Word Viewer 和 Excel Viewer，就能使用这些文件。你也可以从如下站点下载这两个应用程序：

http：//www. microsoft. com/downloads/en/details. aspx？ familyid = 941b3470-3ae9-4aee-8f43-c6bb74cd1466&displaylang = en

勘误及图书支持

我们尽最大的努力保证本书及其附属内容准确无误。自本书出版以来发现的错误已经发布在 oreilly. com 上的 Microsoft Press 站点：

http：//go. microsoft. com/FWLink/？ Linkid = 220642

如果你发现有错误还未公布出来，可以通过这个网站告知我们。如果你需要更多帮助，请发 E-mail 到 Microsoft Press 图书支持部：

mspinput@ microsoft. com

请注意，上面的邮箱地址不对微软的软件产品提供支持。

第 1 版前言

一直以来，我就是 Eric Brechner（以 I. M. Wright 为笔名）的忠实读者。当我第一次遇见他的时候，我花了好长时间才发觉跟我谈话的人似曾相识，因为 Wright 先生之前给人印象是高傲自大的，而眼前的他是一个恭谦、礼貌且友好的人，他看起来更像 Clark Kent[⊖]。

我所喜欢的栏目主要集中于微软内部团队在软件开发的过程中是如何处理技术与人际交流之间的关系的。看到大量的公司内幕被写了出来，我常常会感到吃惊——我不知道还有多少不为人知的故事没有说出来。

大型项目的软件工程管理者面临着 3 个基本问题。第一个基本问题是，程序代码太容易改变了。与机械或土木工程不一样，它们在现有系统上做一次改变总是要付出实实在在销毁某些东西的代价，而软件程序的改变只需要敲敲键盘就行了。如果对一座桥的桥墩或一架飞机的引擎做一个错误的结构性更改，由此产生的后果，即使不是专家也很容易就能明白。然而，如果在一个现有程序上做修改，对于其风险性，即使经验丰富的软件开发者进行了充分的讨论，其结论常常还是错的。

将软件以建筑作为比喻实际上可以给予其很好的解释。基于程序代码在系统中所处的层次，它们可以被比作“基础、框架和装饰”。“基础”代码具有高度的杠杆作用，它们的改动常常会引起严重的连锁反应。“装饰”代码比较容易改动，而且也需要经常被改动。问题是，累积了几年的改变之后，复杂的程序就跟经历了几次装修的房子差不多——电源插座躲到了橱柜的后面，浴室风扇的出风口通向厨房。再做任何改变的话，其副作用或最终的代价都是很难预知的。

第二个基本问题是，软件行业还太年轻，实际上还未提出或建立关于可复用组件的正确标准。龙骨间的间距必须为 16 英寸才能安装 4 英尺 ×8 英尺的石膏板或夹板？我们不仅在这类问题上还没有取得一致意见，甚至还没有决定，是否龙骨、石膏板和夹板这样的组合更可取，还是我们去发明像泥浆、稻草、石头、钢铁和碳化纤维这样的组合更好。

最后一个问题实际上是第二个问题的另一种表现形式。每个项目中重复创建的软件组件，它们也被重新命名了。软件行业里对现有的概念发明新的名字是很常见的，即使用的名字相同，这些名字也以新的方式被重用。行业里有一个心照不宣的秘密：对于如何选择最好的开发方式已经有不少的讨论，这些参与讨论的人使用不同命名，他们之间的沟通完全是一头雾水。

表面上看来，这些都是很简单的问题：建立一些标准，然后强制执行它们。在快速进步的大容量、高价值和低成本的软件世界里，这样做可是一个让你的业务落败的捷径。实际情况是，软件最大的工程障碍，同时也是它最大的优势是无处不在的软件（运行在低成本的个人电脑和互联

⊖ Clark Kent 是“超人”的名字，他具有超强的本领，是一个虚构的超级英雄，美国漫画中的经典人物。——译者注

网上）已经使其以惊人的步伐进行创新成为可能。

随着微软的成长，公司已经不再能在最佳工程实践方法的研究方面大量投入，然后经过深思熟虑，挑选出其中质量最好的方法。个人电脑和 Windows 的成功，已经把公司从按传统方式做些小项目的形态转变为谱写开发有史以来最庞大、最复杂软件的新篇章。

为了能够创建出在风险、效率与创新之间取得平衡的最佳系统，微软面临着持续不断的斗争。考虑到我们的一些项目有着极度的复杂性，这些努力甚至可以称得上“英勇无畏”。在过去的一段时间，我们已经设置了专员，建立了专门的组织，他们都一心一意致力于这个行业里最困难的事情——“软件发布”。我们已经形成了一系列的群体风俗、行业惯例、企业文化、开发工具、流程及经验总结，这些都有助于我们建造和发布这个世界上最复杂的软件。但与此同时，每天都处理这些问题难免让人心惊胆战、意志消沉。Eric 的栏目正是大家一起分享和学习的极好途径。

Mike Zintel

微软公司 Windows Live 内核开发部门总监

目　　录

第1章

项目管理失当

本章内容：

2001年6月1日："开发时间表、飞猪和其他幻想"
2001年10月1日："竭尽所能，再论开发时间表"
2002年5月1日："我们还开心吗？分诊的乐趣"
2004年12月1日："向死亡进军"
2005年10月1日："揭露真相"
2008年9月1日："我得估算一下"
2009年5月1日："一切从产品开始"
2009年9月1日："按计划行事"
2010年5月1日："敏捷的团队合作"

我的第一个栏目是在2001年6月微软内部网络杂志《Interface》上发表的。为了进入I. M. Wright的人物角色，我需要找到一个真正让我苦恼的主题。而工作时间表和进度跟踪再好不过了。

项目管理的伟大神话至今都让我疯狂，它的威力远胜过其他任何主题。这些神话是：

1. 人们可以按期完成任务（事实上，项目可以按期交付，但项目员工能按期完成各自任务的概率不会高于击中曲棍球的概率）。

2. 有经验的人估算日期比较准（事实上，他们能够较好地估算工作，但不是日期）。

3. 人们必须准时完成各自的任务从而使整个项目按时交付（事实上，员工们不能按时完成自己的任务，所以你若想你的项目能够按期交付的话，你必须进行风险管理、范围管理并通过沟通来减轻人性的弱点可能给项目带来的负面影响）。

在这一章中，I. M. Wright将讨论如何通过管理风险、范围和沟通，来保障项目能够按时完成。前两个栏目专门讨论开发工作时间表，接着讨论善后事宜的管理（我们称之为"Bug分诊"）、死亡行军、撒谎以掩饰问题、快速而精准的估算、服务管理、风险管理，以及在一个大型项目中整合各种方法以便协同运作。

不得不提的是，通过我在微软这些年工作期间的观察，在不同的规模、不同的抽象层次中，项目管理有不同的表现形式。这些层次包括：一个团队或功能层次（大概10人左右）、项目层次（50～5 000人为一个特定版本工作）以及产品层次（由高级主管领导的多个版本开发）。敏捷开发在团队层次很适用，传统方法在项目层次中很适用，而长期的战略性规划方法在产品层次很适