软件项目管理

Managing a Programming Project

Processes and People (3rd E

软件项目管理

过程控制与人员管理（第3版）

（美）Philip Metzger
John Boddie 著

陈勇强 费 琳 等译

北京现代卓越管理技术交流中心 审校

電子工業出版社
Publishing House of Electronics Industry

图书在版编目（CIP）数据

软件项目管理：过程控制与人员管理（第3版）/（美）梅茨格（Metzger, P.），（美）博迪（Boddie, J.）著；陈勇强，费琳等译. —北京：电子工业出版社，2002. 8

（软件项目管理系列丛书）

书名原文：Managing a Programming Project: Processes and People（3rd Edition）

ISBN 7-5053-7712-4

I. 软…　II. ①梅…②博…③陈…④费…　III. 软件开发-项目管理　IV. TP311. 5

中国版本图书馆 CIP 数据核字（2002）第 039558 号

责任编辑：李海风

印　　刷：北京大中印刷厂

出版发行：电子工业出版社　　http://www.phei.com.cn

北京市海淀区万寿路 173 信箱　邮编 100036

经　　销：各地新华书店

开　　本：787×980　1/16　印张：22.75　字数：300 千字

版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷

定　　价：40.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077

第 3 版前言

自从本书第 1 版出版问世以来，管理的基本原则一直没有改变过。本书此前的几个版本，所阐述的基本原则已被越来越多的管理者应用于软件项目开发的实践之中，并取得了良好的效果。本书所依据的管理原则与前几版一脉相承。我在本书所做的工作只不过是把这些原则在一个全新的管理环境中重新组合应用。这种全新的环境反映了近年来软件开发行业几个根本性的变化。我就像一个宝石匠，将旧戒指上的宝石拆下，经过一番精心设计之后，又重新将宝石镶嵌在新的戒指上。

随着计算机功能的日臻强大，程序员花费在编程和设计上的时间大大减少了。从前我曾饱受由于微型计算机功能限制给编程带来的种种不便，而现在新一代功能强大的计算机的横空出世为软件开发的迅速崛起和兴盛扫清了障碍，并铺平了道路。此外，计算机功能的强大还带来了软件开发工具的革新。

现在开发的软件系统多是在旧的软件系统基础上进行的，它们很少用于替代原始的手工操作系统，这在很大程度上影响了客户评价新系统的方式，并对由旧系统向新系统平稳转换的具体工作提出了很高的要求。

系统集成在工作中所占的比例较以前有很大提高。从运行在个人电脑上的应用程序，到流转于局域网上的资料数据，再到横跨各大洲之间的大量的信息交流，所有这些都必须集成协同在一起进行工作以期获得客户期望的结果。而项目经理在组织开发这个满足客户要求的系统时，起着举足轻重的作用。

综合上述几种环境变化的影响，我们越来越清楚地认识到这样一个事实：从前“逐级开发”的软件开发模式正逐渐被一种“多过程并行”的软件开发模式所替代。这种崭新的理念和模式构成了本书向你提供的这顿“管理大餐”的基础。也许你会从这本丰富的“菜单”中发现最适合你“胃口”的“美味佳肴”。让我们现在就开始吧，祝你有一个“好胃口”！

原书第1版前言

当我还是一名小程序员时，我就认为自己能胜任我的经理所做的那些工作。终于我得到了一次机会，可以去证明自己的能力。突然间，我手下聚集了十几个求知似渴的程序员，好多还是刚刚毕业初出茅庐的年轻人。当然，我只是将一个个独立的小程序分派给他们去做，而我主要负责收集文件，做出评估，分发报告，并飘飘然地认为自己正在进行“管理”。一切似乎都很顺利。突然有一天，我被任命为 IBM 为美国联邦政府开发的一个大型软件系统的项目经理，一时间，我的手下有了几十个人，如何将他们团结起来，共同完成这个任务，立刻成为了一个棘手的问题。我发现自己对于系统开发简直一窍不通，我的上级和下级经理对于我所面临的困难似乎也一筹莫展。

于是，平生第一次，我被迫去真正思考有关系统开发的事情。思考使我获得了一些心得。之后，我又请了非常有实践经验的 Joel Aron 和 Al Pietrasanta 在 IBM 公司内部举办一系列管理类的讲座。这些培训课程对数以百计的软件项目经理的管理实践工作具有很强的指导意义，并受到各方的广泛好评。我的课程笔记也成了 IBM 内部使用的编程管理手册，这本手册后来经修订之后，被正式出版发行，即本书的第 1 版。

在这本书中，我努力将一些我第一次成为经理时所急需的指导性建议编了进来。无论你处在软件开发的什么过程或正在从事什么样的具体工作，这本书都将会对你有所帮助。尽管书中称呼你为某个软件项目的项目经理，但本书却适于处于软件开发项目的各个层次和水平的所有经理阅读。此外，专业技术人员也可将本书作为参考。书中没有高深的理论、复杂的公式，有的只是一种对管理方法和思想较为实际的探讨。在本书中以人为本的管理思想将贯穿始终。衷心希望这本书会对你有所帮助。

目　录

PART 1

第一部分 介 绍

我喜欢让生活尽量简单化，而不愿意同时去处理很多事情。基于这点，写这本书我只有一个目的 介绍一种高质、高效、低成本管理软件项目的方法，并使所有相关人员都愉快地参与其中。这本书忽略了开发一个软件系统的社会、政府和公司的因素，这些因素在其他书里会加以介绍。在这本书里，我们关注的是一旦决定去开发一个软件系统，该如何去做好这个项目。

我希望这本书对一个初出茅庐的管理者能有很大的帮助，但如果你已经具有了十年以上的管理软件项目的经验，也不要觉得这本书对你没有什么用。

本书的第 1 版是在 1973 年出版的，时过境迁，现在计算机已经发生了巨大的变化。摆在我桌上的这台电脑比起我最初用来编程的那台在功能上已经强大了很多。以前，一台电脑便可占据整个一层楼，并需要专门的空调，要由专门的技术小组去维护它，以保证其顺利运行，而且一般只有大型集团才能拥有两台以上的电脑。而现在，程序员在自己的办公室里就能拥有两台或更多的电脑，除此之外，在家里应该还有。

软件也随着硬件的发展而发展起来，它变得更强大、更精密、更复杂。现在我只要用一个晚上通过在电脑上移动各个程序模块，并用线条将它们连接起来，就能为我儿子的足球联盟编写一个非常有用的程序。我所在的咨询公司里，几乎有 20%使用中的程序是在其他已有程序基础上编写而成的。这些程序只要获得一定的设计结构和接口的定义，便可自动生成可执行代码。

虽然变化是巨大的，但是管理者似乎并没有跟上这些变化的步伐。电脑强大的功能提供给管理者一些有用的工具，但是管理者掌握使用这些工具的速度总是慢半拍。搞了大半辈子的软件开发，管理者所能掌握的仅仅是那些独立于技术之外的最基本的技能。

可能是因为探索技术总是比管理人员要容易，管理水平一直处于滞后的状

态，就像我们能够探索月球，却很难牢牢把握住人生的幸福。我认为我们经常把管理复杂化，实际上管理并不是这样的。我们可以简化管理，因为如果技术变得越复杂，就越需要简化管理。本书的目的就是描绘一个组织开发软件项目的模型，并对项目中最重要的成分——人员的选择和管理给出建议和指导。

我的基本规则

这不是一本关于团队组织、政治或高深管理理论的书，也不是一本关于软件开发的教科书，而是一本重点突出而且很实用的书。它强调的是如何成功并愉快地管理一个软件项目，无论项目的规模如何。各位读者，在这本书里我把你当做一位正在管理一个中等规模软件开发项目的项目经理，这个项目大概需要四十个人，包括程序员、经理和其他人员。在本书的第九部分，我们会谈到各种规模的项目。工作之间的区别并不是你所要考虑的主要事情，你要关心的是对于不同的工作你需要投入多少资源。

我并不想列出所有实施项目的不同方法。在这本书中，你看到的只是一个不错的“烹饪秘方”，而不是一本“烹饪书”。像任何一个好的“烹饪秘方”一样，在每一步我都会告诉你怎么做这个，如何做那个，希望你最后能做出一份美味佳肴，同时你可以用不同的方法使用这些原料，并添加一些你认为会使之更美味的调料。

“经理”这个头衔对于不同的组织有不同的含义。我所指的“经理”是负责计划和指导工作的人员，他们直接负责雇用人员、解雇人员，调整薪水和提升职位。第一层经理监督实际开发软件的人员，第二层经理监督第一层经理的工作，依次类推。

在本书里，“程序”和“软件”这两个术语的意思基本相同。应用程序是指那些应用于实际工作环境的软件，如工资表核算系统、空间导航系统等；辅助程序是指那些帮助开发应用程序的程序。

你的基本规则

在阅读一些文献时，你会发现许多术语的定义是模糊的，或自相矛盾的，如软件、模块、集成、系统测试等。我不能确定你和我对于这些术语是否有清楚一致的认识，但是在你的项目中，你一定要采用明确的术语，并一直延用下

去，这一点对管理软件项目是非常重要的。在项目实施过程中，不应出现一些人员把某项工作称为“系统测试”，而另一些人员则把它称做“集成”的情况。团队内部术语的一致性能促使项目顺利地完成，我在下面列出一些你应该制定的基本规则。

第一，定义你的项目开发周期，并把所有的时间表和工作过程与它联系起来。这本书便是围绕这个周期和其中的方法过程展开的。比如当我提到软件开发过程时，你应该意识到我指的是与这个过程相关的一系列定义明确的活动。在项目开发周期中，编程过程和其他过程之间有明显的区别。不要在这里把某项工作称为“接收过程”，放在那里又把它称为“转换过程”。你可以随意选择对术语的称呼，但使用时要保持前后一致。

第二，定义活动，例如对各种水平的测试的定义。采用并坚持使用那些你自己明确而且能被大多数人接受认可的一系列定义。比如在这本书里，我明确定义了“系统测试”，但一些人把这个术语与“集成测试”混为一谈，这种认识的偏差应尽量避免，你至少应该知道各个术语在你的项目中的定义，并确认其他人都清楚这些术语的定义。

第三，项目初期就应定义文档系统，并确保它的明确性和一致性。在任何项目中都会有大量的书面文档，如果你不对这些随意摆放的文档进行控制管理，到项目中后期，这些文档就会使你应接不暇，无法控制。

概括地说，要在你的项目中定义开发周期，并把它作为你制定计划和开展工作的基础，在实践中牢记推广并实施。

你的合同

你以为自己与合同无关吗？如果你真这么想，那你就错了。如果你没有签署合同，你要进行的将是一场艰苦的持久战。软件项目的许多麻烦都是因为没有很好地签署合同，或根本没有签署合同而引起的。

无论你是代表一个独立的公司，还是一个公司的数据处理小组，总之你是要管理一个软件项目，这也就意味着你将运作一项业务。你会有供应商，有一个或多个客户，有雇员，有委托，同时还会有财务目标，有可度量的产出成果。你的工作可能一帆风顺，也可能一塌糊涂。你的责任是管理好你的业务，以便每一个人——你的“投资者”（给予你任务的上司）、你的客户（将在今后一段较长的时间使用你的系统的人）、你的职员（一群优秀的人员，他们将献出休息

时间来开发这个系统）和你能够一起密切地关注你的工作及成果并能对其感到满意。

请记住，现在我把你当做超级软件有限公司的一个项目经理，你拥有一定的权利和义务。你最好在通读合同并为你的业务制定了基本的规则之后，再开始所负责的软件开发工作。

合同是你与客户之间的协议，你将在一定额度资金的限制下开展业务。即使你的客户是你工作上的伙伴，并且你们都是为同一个组织工作，也不要以口头协议或临时通话记录的形式来实施项目。在公司里，可以把合同称为“协议信”或“数据处理服务要求”等一些听起来比“合同”更加亲切的称呼。在任何情况下，你都需要一份正式的书面报告书，它能明确表明客户的需要和你所要提供的产品。没有上述协议就开始实施项目是不可想像的，这一点已被许多软件开发项目经理和客户所认可。

如果你的组织较小，没有相关的合同部门，你可以自己编写一份包括以下要点的“合同”：

- 工作范围。说明你要进行的工作内容。如果工作的定义非常模糊，那么你可能需要两份合同，一份用来定义工作，另一份用来编写程序。
- 进度表和支付交货。说明你要交付给客户的专门的项目（程序或文档），何时、何地、以何种形式（是以源代码，可执行模块，草稿还是明确的文档形式）交付，以及副本的数量。
- 关键客户。说明由谁来批准变更和接受完成的产品。
- 审查。说明客户何时和用何种方法才能获得进度审查报告并对其进行审查。如果客户不批准进度报告，那么客户有什么其他要求。
- 变更管理程序。说明处理原始工作范围项目中的变更机制。
- 测试条件。说明进行测试的人员、客户可参与测试的部分、需要测试的系统，以及由谁来确认测试应该涉及的内容。
- 接收标准。说明用来判断产品可接受性的定量标准。
- 附加条件。说明是否存在一些关于工作环境的专门条款；如果你使用了客户的员工，你将怎样管理这些人员；是否存在专门的数据安全问题；如果由客户提供测试数据，那么客户采用何种类型、何种形式、何时以及如何提供数据。
- 价格。说明这个项目的造价（或预算）。它是固定的还是可变的，如果是可变的，说明它在什么情况下发生改变。

以上都是一些要点，在本书里，我还会详细介绍。

软件项目的性质

即使各种外部条件都已具备，开发一个软件系统也不是一件容易的事。同样，管理一个软件项目也是如此，项目可能偏离正轨。以前有过成功经历的项目经理，在管理一个新的项目时，他还是可能会遇到麻烦。Frederick 在他的著作《神奇的人·月》(1975 年）一书中回答了 Thomas J. Watson Jr.的关于“为什么开发软件如此困难”的问题，在本书里我想说明“为什么软件项目如此难以管理”的三个原因。

1. 系统

我们要开发的是一个系统，它是用来满足一系列目标的相互影响的各个部分的结构化的组合。如果我们举一些例子来说明系统特征的话，这个古板的定义就会变得生动起来。

到处都有“系统”的例子，如太阳能系统、供应能源系统、人体消化系统、公司系统、计算机系统等，这些都满足“系统”的定义。每个系统都是另一个系统的一个子系统。比如本书经常提到的软件系统，它是数据处理系统的一个子集，而数据处理系统可能又是电话系统的一个子集，电话系统又是电信网络系统的一个子集，依次类推。

为了确保你所开发的系统具有一定的作用，原则上它应该满足其他系统的需要。而理解并定义这些需要并不是一门确切的技术，因为不存在（或几乎不存在）百分之百全面完整的资料来源，能告诉你今天其他系统需要什么，而明天其他系统又需要什么。理解并满足客户的要求是一个循环反复的过程，不仅存在于项目实施阶段，而且贯穿整个系统的生命周期。

2. 相互作用

“系统”包括彼此相互作用的部分。在软件系统中，这些部分可能是运行系统、应用程序、服务程序、硬件、操作人员、使用人员等。当系统的规模和复杂性都在增加时，控制这些部分的相互作用将成为一项重要的任务。

图 I-1 描述了当系统里元素的数目增加时，相互作用可能随之增加的数量，对于项目经理来说，控制、简化系统中的相互作用的数量是非常重要的，对于管理只有一个功能的小程序的开发和管理有许多部分的复杂系统的开发（如集

成金融管理系统的开发）而言，相互作用的数量有显著差别。以后我们将讨论到关于模块、界面定义和项目组织等理论，这将帮助减少相互作用的影响。

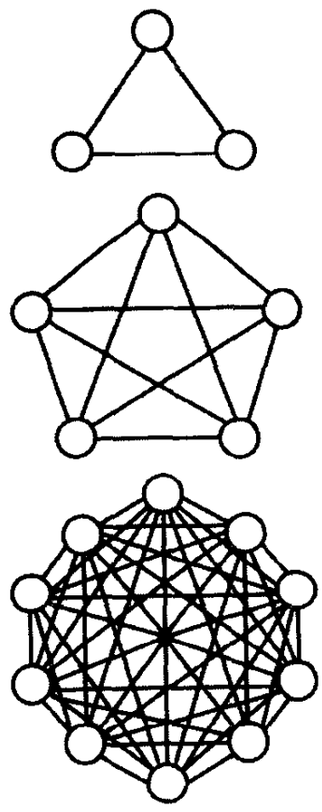

图 I-1　相互作用

3. 变更

任何一项进行了两周以上的工作肯定会遇到一定的变更。作为一个项目经理，你应该预先想到可能遇到以下一些变更。

- 需求的变更。在项目初期，分析部门对工作的定义，常常在不停地改变。工作的任务越复杂，需求就越可能发生改变。
- 定义的变更。在第 4 章我们要谈到基准设计，它是你进行软件开发的基础。每个房主都知道，如果房基出现裂缝，就需要修补。软件系统也不例外，虽然基准设计是一个良好的开端，但它也有可能发生改变。
- 技术的变更。政府赞助的大型软件项目（如 NASA 系统，军事指挥和控制系统）特别容易发生技术改变。这是因为：第一，项目实施的时间相当长，新设计的出现和科学的发展（如武器设计和数据处理设备）是不可避免的；第二，由于这些项目的设计理念代表着科学发展的方向，因

此它们直接肩负着推动技术革命的历史任务。

- 社会的变更。许多软件项目都是社会行为方式改变的潜在受害者。例如，当银行在超市里采用自动取款机后，在出纳员窗口用于处理取款业务的程序就发生了改变：手工人员的变更、人员的离去、死亡、生病都会使工作发生改变，当你失去了一个重要的职员或一个大客户时，你的工作可能会遇到变更的问题。
- 更正。每个人都会犯错误，这些错误或大、或小，或为技术上的、或为管理上的，或是明显的、或是微妙的，但不管怎样，这些都是错误，我们必须改正它。

对于这些变更来说，重要的是管理它们，而不是试图消除它们。如果变更存在于问题说明书中，那就应正视它们。下一步你需要做的是估计这些变更对项目费用和工期的影响。如果客户想要另外改变需求，他应该在双方对合同的变更进行协商，签订正式的变更通知后才能改变。

无论变更什么时候出现，有一点是明确的，即如果你和客户对变更不能达成一致意见，你就不能解决所面临的问题。如费用的改变，你会说："客户先生，这个改变将需要额外的十个人干一个月，并导致在交付时间上两个月的滞后。"客户则回答说："什么改变？难道我的钱就不应该用在这些关键的地方上吗？不要欺骗我了。"然后，你去疯狂地查阅问题说明书，希望找到一个段落、一个条款、甚至一个句号来支持你的论点。这是普遍的做法，但它经常会导致项目停滞不前，这就是为什么在这本书里我始终强调建立准确的、有目的性的基准设计文档的原因。

学会协调与权衡

有时，项目经理想努力做好他的本职工作，然而他所处的组织经常发生变更。在这种情况下，项目经理不能想当然地认为组织没有发生改变，更不能任由这些改变破坏目前他所进行的项目。虽然项目经理扮演的是一个矛盾而又复杂的角色，但是他必须保证项目成功地完成，这就要求项目经理有协调组织变更的派生需要和其项目本身需要的能力。但最重要的是项目经理应承担起"家庭"（也就是项目）的重担。

用长远的眼光看问题

也许你会把所进行的项目看做一个全新的东西，认为你的团队在开发以前没有在电脑上运行过的新软件。但是，对你而言它可能是新事物，对你的项目而言，它的任务只不过是替代已有系统的一部分或全部，或者只是在已有软件基础上所做的提高或优化。

Charles Bachman 曾说过 “其实根本不存在软件的开发，那些所谓的软件的开发只不过是对程序的基本维护而已。”事实上，现在许多项目所花费的大量费用、时间、精力都集中在程序的维护上。基于此种情况，你需要扩展你在项目中的管理责任，始终牢记你的最终目标是创造一个在其整个项目生命周期内都通俗易懂，并且灵活可变的软件产品。假设现在是项目生命周期的开始，且项目将持续 6 个月，在开发第一个程序的时候，你就要力求让它具有可维护性。如果没有做这些维护工作，你会发现代价是惨重的。

在这本书里反复强调的重点是“文档和标准化技术”。它不仅能够帮助你获得工作上的成功，也会确保在你之后为这个系统工作的人也获得成功。开发一个容易维护的系统会对你的职业生涯产生深远影响，随着你所开发的系统广受好评，你的信誉也会随之上升。

流行的技术

在 20 世纪 60 年代、70 年代和 80 年代，早期的软件开发业由于相关规则的引入而取得了很大的发展。虽然还有许多需要提高的地方，但是，所取得的成果是令人振奋的。特别是如果你回忆起在 20 世纪 80 年代，一些人用汇编语言编程，并只能使用一些最基础的开发工具，你就更能感觉到这种变化的存在。感谢那些天资聪颖并为软件开发行业付出了巨大努力的人们，正是他们的努力使软件开发经历了真正的变革。现在，有各种各样的自动化工具可以用来帮助分析员、设计者、程序员和测试员（甚至项目经理）完成他们各自的任务。当然更重要的可能还是软件开发人员在进行开发工作时，基本方法和思想上的变革。

现在，一些在开发计算机程序上战功赫赫、经久不衰的项目组多半使用了以下技术：

- 分析和设计的自顶向下的方法；