软件项目管理系列丛书

AntiPatterns in Project Management

项目管理反模式诊断

软件开发常见错误规避

William J. Brown
（美） Hays W. "Skip" McCormick III 著
Scott W. Thomas

杨晓燕 任树芳 王 虹 等译

WILEY

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http:// www.phei.com.cn

软件项目管理系列丛书

AntiPatterns in Project Management

项目管理反模式诊断

软件开发常见错误规避

William J. Brown
（美） Hays W. "Skip" McCormick III 著
Scott W. Thomas

杨晓燕 任树芳 王 虹 等译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
北京 · BEIJING

版权贸易合同登记号　图字：01-2002-3340

图书在版编目（CIP）数据

项目管理反模式诊断：软件开发常见错误规避 /（美）布朗（Brown, W. J.），（美）迈克考米克（McCormick, H. W.），（美）托马斯（Thomas, S. W.）著；杨晓燕，任树芳，王虹等译. —北京：电子工业出版社，2004.1
（软件项目管理系列丛书）
书名原文：AntiPatterns in Project Management
ISBN 7-5053-9432-0

Ⅰ. 项… Ⅱ. ①布… ②迈… ③托… ④杨… ⑤任… ⑥王… Ⅲ. 软件开发－项目管理 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字（2003）第 111993 号

责任编辑：冷元红
印　　刷：北京大中印刷厂
出版发行：电子工业出版社
　　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：21　字数：285 千字
印　　次：2004 年 1 月第 1 次印刷
定　　价：36.00 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。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

丛书总序

软件产业是一个朝阳的、新兴的知识型产业。一个国家软件业的发达程度，也体现了国家的综合国力，决定着国家未来的国际竞争地位。

目前，中国的软件企业正处于高速发展、急需规范管理并以项目为主导的环境中。企业每天所面对的不仅仅是几个越来越大的大型项目，而将是成百上千不断发生和进行的项目。产生这种变化的因素是多方面的，这包括客户需求的不断提高导致产品生命周期缩短；产品开发项目数量大增；新技术导致了对研究和开发项目需求的增加；为了提高业务赢利能力，改进业务模式的项目需求大增等。在这种多项目并发、技术含量高、变化速度快、资源有限的环境下，如何对企业、项目、资源实施科学的管理，加强团队能力，实现软件企业的生产规模化、规范化、国际化，是当前我国软件业面临的最大挑战。

一些调查表明，大约 70%的软件开发项目超出了估算的时间，大型项目平均超出计划交付时间 20%到 50%，90%以上的软件项目开发费用超出预算，并且项目越大，超出项目计划的程度越高。

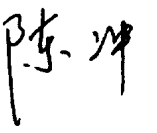
在中国成为 WTO 成员的今天，如何科学地管理企业，激励开发人员，提高软件开发的生产率，按时、按预算提供满足客户需求、具有国际市场竞争力的软件产品，一直困扰着软件企业的管理者，同时也成为阻碍中国软件业向国际化发展的绊脚石。

中国目前软件业的落后状况，实际已经不是技术本身的问题，而是科学管理的问题，软件工程理论与软件项目管理的实际应用与经验，是我国软件企业亟待学习与渴求的。中国软件业各个层次上的管理问题，目前可以说已经制约了中国软件业

走向国际化的进程。

电子工业出版社针对以上现状，组织出版了“软件项目管理系列丛书”。该套丛书针对如何有效进行软件企业管理、软件项目管理、软件质量管理这一主题，以软件企业CMM的实现与软件开发项目管理的问题为核心展开，围绕软件业中的技术、过程、产品、人四个维度进行讨论。该套丛书的出版，为中国的软件企业提供了科学的管理方法与技巧，特别是贯穿整套丛书中的国外软件企业和软件项目的管理实施方法、管理经验与教训、成功与失败的案例，对帮助中国的软件企业实施科学的管理，把项目变为可以控制，从而以更短的时间、更高的质量、可预测的成本生产功能更为丰富的软件产品，具有重要的作用。

该套丛书的内容独特、实用，多数均为世界软件开发领域的经典名著，是国际上软件项目管理大家智慧的结晶，代表了软件大国的先进管理理念与水平，可以说是一套很好的软件业“MBA”丛书。该套丛书的出版，对推动中国软件业向世界软件大国迈进，具有重要的意义，也希望中国软件企业能够从该套丛书中，学习国外先进的管理经验，尽快成为具有国际管理与生产水平的软件企业，为使中国尽快成为世界软件大国而努力，为推进中国信息化建设做出贡献。



信息产业部电子信息产品管理司副司长
中国软件行业协会副理事长
2001年12月12日

译者序

这几年关于软件项目管理的中外书籍很多，但是还没有人能像本书作者那样将自己丰富的经验汇集成书，以研究失败为出发点，分门别类地探讨如何在软件开发过程中避免犯错误。

自称为技术项目经理人的 William J. Brown，事实上是目前国外软件开发业的先导者之一，他深谙成功交付软件所需的人员、技术和过程三大要素的平衡艺术。毕业于美国海军学院计算机科学专业的 Skip McCormick，在多层分布组件架构、安全协作系统、人工智能、知识管理和其他相关信息系统领域有着丰富的软件工程经验。而 William J. Brown 是 SAGA 软件公司的高级产品开发经理，他在工程生命周期方面有着 20 多年的经验，目前关注的重点是开发分布式的对象中间件。

目前中国的软件开发空前繁荣，但是面临的问题也越来越多。工程师们常常要应付诸多影响软件交付的问题，如怎样接受领导不断增加的需求，怎样能在使用不稳定技术的情况下出色地完成任务，以及如何按期完工。正是为了解决这些问题，电子工业出版社世纪波公司慧眼识珍珠，将本书介绍给中国从事软件项目管理的工程师们，以期抛砖引玉，促使中国的工程师们在解决好问题的同时，从失败中积累经验，指导实践，提高软件交付的成功率。

书中包括 3 部分内容：软件开发的标准；与人的因素相关的微观管理、派系纷争、人员配备、各种变化和过程瓦解；与技术因素相关的不成熟商业软件导致的艰难开发与维护问题、使用分布对象技术产生的问题、额外需求或设计属性的堆积等；与软件开发过程相关的计划编制、生命周期、多米诺效应等。所有这些都是工程师

们在实际项目管理中常常遇到的问题，如何解决这些问题关乎项目的成败。作者从产生背景、一般形式、症状与后果、化解办法等角度详细做了回答，令感到困惑的项目工程师们豁然开朗。

本书的读者是从事软件开发的工程师和项目经理，已经有了软件开发方面知识、希望运用管理技巧成功交付软件的人们。

承担本书主要翻译工作的有任树芳、王虹、杨晓燕、阳天青、姜明和刘洁，另外张志明、王锦程、曹秀云、梁晓虹、郝秦生等也参加了部分章节的翻译和讨论工作。翻译过程中，译者得到了航天信息中心研究员孙广勃、张会庭等的热忱指导和帮助，特此致谢。由于时间仓促，书中难免有词不达意之处，敬请原谅。谢谢朋友们！

译者

2003 年 3 月　北京

序

2000 年在“千年虫”的恐慌中静静地过去而成为历史，人类社会和我们生活的这个星球此前从来没有像今天这样受到软件编程的影响。尽管曾有过种种担忧，但我们并没有受到太大的影响。由于采取了最长久运行的化解办法（这种办法可能是最昂贵的），问题的本质才得以确认，正确的解决方案才能被制定出来且予以实施。在化解“千年虫”问题的过程当中，许多变化情形在软件业内被人们数以千万次地重复提到。

尽管世人已经对“千年虫”感到厌烦了，但是几乎很少有人听说过反模式。因此，虽然软件工程师们特别重视软件项目管理，但是仍在重复着同样的失败。在这里，我们将不再讨论“千年虫”，而是通过教你了解软件开发项目管理过程中易犯的错误，使你作为一名经理人、项目经理或项目团队成员而为未来的软件开发做好准备。反模式不可能挽救软件开发世界，但是可以使你更好地完成软件开发项目，并且处理频繁遇到的问题。

与反模式系列的前两本书类似，本书也从实际角度探讨这个问题的解决方案。前两本书是《反模式：化解危机中的软件、架构与项目》（*AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*）和《软件配置管理的反模式与模式》（*AntiPatterns and Patterns in Software Configuration Management*）。我们希望本书能够使你联系生活中的实际情况，使得理解和实施更加容易；也希望你在阅读本书时，如果发现书中谈到的某个反模式恰恰与你生活中发生的情况类似，你的脸上能露出会心的微笑。但是更重要的是，我们希望它能够帮助你以更为有效的方式来管理软

件开发项目，帮助你成功地识别你所面临的或者将要遇到的问题，并采取行之有效的办法来解决这些问题。我们希望你能够精通软件开发和项目管理的有效实践。本书简介部分列出的那些书能给你提供这方面的更多知识。哈罗德·科兹纳（Harold Kerzner）博士所著的《项目管理：计划、进度和控制的系统方法》（*Project Management: A Systems Approach to Planning, Scheduling, and Controlling*）*提供了大量有关项目管理的基础知识，是项目管理基础教育必备之书。但是本书无论是对自学成才的编程人员，还是对具有计算机科学或系统工程硕士学位的专业人员，都同样适用。我们假设你已经具有这方面的实际知识，只是还没有找到获取这些知识的路径。

本书的目的是帮助个人或团队管理软件开发项目。这是一个不易解决的难题，因为每个项目都有其自身的特征，有许多可变而且确实在变的因素。影响一个项目的三大要素是人员、技术和过程。如果你学会如何管理好它们，你就可能取得成功，不管周围会发生什么情况。

这个工作通常来说是很有趣的，而且我们试图将其中的部分乐趣传递到我们提供的一些项目管理难题的解决方案中。如果你已经是一个项目管理领头人或开发团队的领导者，或者已经将项目管理作为毕生的事业，那么反模式对你将非常适用。

本书既可以作为项目管理的指导教材，又可以作为参考书。从战略角度使用本书将使你避免在软件项目管理过程中重复犯一些经常性的错误，而从战术上它将使你和项目从已有的失误阴影中摆脱出来。

最后，我们并不认为我们一定能识别出下一个千年的“千年虫”反模式，但是我们坚信本书将为这一问题提供解决办法。

* 本书中文版已由电子工业出版社出版。——出版者注

引　言

软件开发失败的主要原因是缺乏恰当的项目管理，本书将告诉你如何改进软件项目管理，提高项目的成功率。软件项目要取得成功，就必须以平衡的方式管理软件开发的各个方面，包括人员、技术、过程、预算和时间。一项计划周全的软件开发的成功意味着要实现下列目标：

- 交付准时。
- 成本在预算之内。
- 交付期望的软件。
- 开发团队成员心情愉快。

但是要实现上述目标却很难，因为在任何工程领域中项目管理都是最艰难的活动之一。目前，软件工程还未形成完全的体系，这意味着要获得软件项目管理方面的成功是极为复杂和艰难的。通常所说的软件工程平衡是指：

- 利用预先确定的组件格式化架构。
- 严格遵循可复用过程，以取得一致的结果。
- 精巧解决方案的艺术性创造促使新的技术结合。
- 利用明确的模式解决已知的问题。

本书的目的是指导软件项目经理解决软件开发管理中的主要难题，帮助他们掌握使软件开发过程走向成熟的关键技术。其他有关过程如软件配置管理也很关键，但是它们只有在项目管理过程允许的前提下才能获得成功。

1. 项目管理的关键方面

项目管理中需要认真控制的 3 个重要方面是人员、技术和过程。许多人认为人员和过程是成败的关键因素，但却忽视了技术的关键性。特别是在今天，随着越来越多的技术可以为我们所用，那些被创造出来的不同的技术体系必须被集成。当企业运用不同的软件技术集成各类体系时，其结果是一个无穷尽的 $n \times n$ 的问题，企业应用集成（enterprise application integration, EAI）“金弹”无法自行解决这个问题。

关键是要认识到人员、技术和过程缺一不可。它们之间的相互依存关系通常表现得不明显，但是在软件开发周期中必须以平衡的方式对各方加以控制，这是项目经理的关键关注点所在。其他的一切仅仅是平衡或不平衡结果的副产品，比如和谐的工作、清晰的软件配置管理过程及对如何有效实施一项技术（所具备）的理解，或者是不和谐的人际关系、不明了的测试过程及一项技术的实施失败。

今天的项目经理必须是一个多面手，对人员、技术和过程有着透彻的了解。如果其中的任何一个能够被完全授权，那么对于软件开发项目就不会有任何约束。遗憾的是，这 3 个软件开发方面之间的相互依存关系相当复杂，处理它们任何一方的惟一有效途径就是对它们进行综合处理。

这并不是说项目经理不能指望高级项目职员支持对这些关键软件开发因素的管理，而是说应该有一个协调者来保证恰当地平衡战略、战术和资源应用。

本书将帮助初级和高级管理者更加有效地管理软件开发的各个方面，认识其中的起因、症状与后果，并提供可复用解决方案。

2. 软件工程作为一门工程学科

许多人认为软件工程渴望成为一门工程学科，但却尚未实现。这种认识很大程度上依赖于这样的前提，即所有软件开发都可以通过参考一本关于可复用组件的蓝皮书或红皮书得以完成，其中可复用组件使用的是完全重复的过程。这种认识并不完全正确。如果世界上所有的软件开发都利用现有的所有经验重新开始，那么就不太容易获得更好的工程方法。

但是就建筑工程而言，它是一门纯工程学科。正在建设的一座大楼，不管它是一幢房子还是一座摩天大楼，其前提是各个组成部分已经预先定了下来，可复用过程也完全可行，并且必须按预期完成；但是无疑这并不适用于软件工程中的维护与增强。相反，多数软件开发人员在软件维护过程中遵循的是开发中所用的程序，倾向于利用各种细小的组件提高可维护性；而房屋建筑的维护则是由一群技术工程师利用自己的特殊手段完成的，通常很少会使用与原来的结构类似的东西来建造。

因此，问题就变为：工程学科到底需要些什么？就软件工程而言，这当然应该是一整套一致的和可复用的软件开发过程，这些过程的实施应该是快速、经济且高效的。软件工程的3个主要部分——软件开发、软件配置管理和项目管理，必须陈述清楚，以实现这个目标。为成功地完成所要求的软件开发活动，本书研究了项目管理的诸多问题：

- 软件概念
- 需求分析
- 架构
- 详细设计
- 编码与调试
- 测试
- 质量保证
- 软件配置管理
- 发布管理

3. 软件工程反模式

在所有项目管理中反复出现的、具有普遍性的坏的实践称为“反模式”（AntiPattern）。反模式是软件项目管理在成功交付路上经常遇到的障碍，是由于对正确的软件开发项目管理方法缺乏认识而直接造成的，常常是在毫不知情的情况下重复采用不正确的方法。大体上，反模式将从起因、症状与后果3个方面对那些坏的实践进行定义，并提供相应的化解办法，从而避免反模式的发生，指导那些已受到影响的软件开发，使其从反模式中挣脱出来。

反模式的发生有两种基本形式：

- **孤立反模式** 由于它们是独立存在并且没有隐藏在背后的起因，因此极易识别。本书第2章~第4章分别探讨了与人员、技术和过程相关的孤立反模式。
- **相互作用且具有等级性或序列性的反模式** 当一个项目管理反模式引起了一个软件开发反模式，并且由于缺乏控制而使该反模式又顺着阶段性活动的等级传了下去的时候，这些反模式就非常地显著了。这是第5章“反模式冲突”中探讨的重点。

本书从人员、技术和过程3个方面对主要的项目管理反模式进行考察，并指出该如何分析反模式的起因、症状与后果。本书还详细考察了有关等级性或序列性反模式的复杂问题，并且定义了一种模式化的解决方案以成功化解这类反模式。

反模式在软件开发过程中随处可见，但是解决这些问题却存在时间选择和先后

次序上的问题。关键反模式是指那些能导致完全失败的反模式，它们常常超越了现有项目本身，导致了员工流失、组织崩溃和项目失败。本书将帮助软件项目经理识别严重的问题（甚至是灾难性的问题），并提供能够使软件项目经理从额外的反模式中解脱出来的化解办法。介绍软件项目管理的反模式将有助于你识别下一层次中存在的反模式，比如软件配置管理、架构、需求和测试。利用反模式将使软件开发人员和过程走进一种更易预见和风险降低了的冒险之中。想要证实的话，请浏览亚马逊网站（www.amazon.com）和巴诺网站（www.barnesandnoble.com）上的评论文章，以及《软件开发杂志》（*Software Development Magazine*, www. sdmagazine.com，该杂志 1999 年授予了我们 Jolt Productivity 奖）上刊登的我们的第一本书：《反模式：化解危机中的软件、架构和项目》。有人指责我们只是以实际和重复的方式为一些复杂的动态问题提供了常识性的解决办法。从某种程度上讲，通过对重复的且可复用的反模式的定义，我们已经朝着使软件更像是一门工程学科的目标迈进了一步。经常使用这些反模式将使软件开发过程更加容易实施，我们还希望反模式将有助于减轻你的负担。

导　论

本书全面细致地考察了软件开发项目中经常出现的、更具共性的项目管理问题，利用反模式技术来明确这些问题，告诉你如何回避并阻止这些问题的出现，从而使自己的项目免受影响。本部分概括了本书的行文结构，并向读者介绍了反模式的概念及其模板。

1. 反模式

一种反模式就是一种机制，该机制能为产生极大负面后果的软件开发描绘出一种解决办法。反模式通过考察所用的不良解决办法的起因、症状与后果，提出化解办法，为软件开发提供很成功的方法。

反模式是一种代表重复出现的软件开发问题及其化解办法的标准形式。无论是问题／化解办法的级别还是可适用观点，都是确定反模式具体范围（scope）的重要因素。

（1）反模式的级别

级别（scale）是指在软件开发中反模式出现的那个层次。图 a 表示了**软件设计层次模型**（software design-level model, SDLM）中所使用的 7 个级别：

① **全球级别**　全球层次上的设计发布允许该设计跨越所有系统而为大家所使用。这个层次所关心的是所有参与沟通与信息共享的组织之间的协调问题，而这些组织之间存在的沟通与信息共享又是相互交叉的。

② **企业级别** 企业层次上的关注点是某个单一组织内部的协调与沟通，该组织可以跨多个地域、不同种类的硬件和软件系统而分布。

③ **系统级别** 系统层次研究的是应用之间的沟通与协调。

④ **应用级别** 应用层次关注的是各种应用的组织方案，开发出这些应用是为了满足一系列用户需求。

⑤ **框架级别** 框架级别也称为宏观组件（macro-component）级别，所关注的是应用框架的组织与开发。

⑥ **微观架构级别** 微观组件（micro-component）级别以软件组件的开发为中心，该软件组件旨在解决再次出现的软件问题。每种解决办法相对来说都是独立的，只能部分地解决一个大问题。

⑦ **对象级别** 对象这一层所关心的是对象与类的开发，它更关心编码的可复用性而不是设计的可复用性。

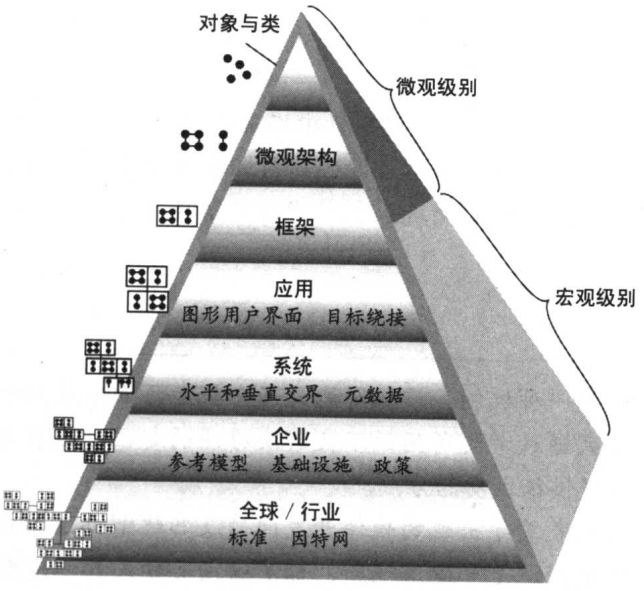


图 a 软件设计级别模型

（2）反模式观点

观点（viewpoint）是一种看待形势或考虑形势的立场。它们之所以重要是因为它们能影响认识问题与随后解决问题的方式。对于某个特定的软件开发方面来说，它们依赖于主要决策者在软件开发中所扮演的角色。通常可以将它们分为管理者观点、设计者观点和开发者观点。

① **管理者观点**仅限于实际上管理着团队、项目和计划的人的观点，他们对整个

软件开发生命周期的计划编制与进度安排负有特殊的责任。项目经理通常还要负责随后的软件开发生命周期，包括软件配置管理。

② **设计者观点**关注的是确认所用的技术，以架构的形式指定技术配置，并确保该技术正确实施。架构配置涵盖了架构的逻辑与物理表象，并划定了使用所选技术的范围。实施中要保证设计与架构一致，实施期间要以编码发现为基础来更新架构。

③ **开发者观点**关注的是软件开发过程的贯彻落实。开发者可以是分析员、高级设计人员、初级程序员或测试工程师，换句话说，就是在主流开发中由所有的角色来执行软件开发的任务，比如需求汇总、设计、编码与测试，或者其他相关的基础任务（如软件配置管理、质量保证）。

（3）反模式模板

反模式模板提供了一种编写下列文档的一致路径：

- 共同的但又有各自特点的软件开发问题
- 把我们自己从症状与后果的影响中解救出来的方法
- 避免这些问题的方法，并因此在未来也避开这些问题的起因的方法

以下是反模式模板的标准格式及对其内容的定义。

① **反模式名称**（AntiPattern Name） 反模式名称是所讨论的问题的关键名称，用以代表相应的反模式所蕴涵的基本原则。这些名称非常重要，因为在讨论及编写软件和架构文档时，它们是术语基础。

② **又称**（Also Known As） 这一部分指出了反模式的其他流行名称、描述性名称或者幽默名称及惯用语。

③ **最适用的级别**（Most Applicable Scale） 这一部分确定了反模式在 SDLM 中的级别，划定了解决办法的范围。级别用下列这些关键词表示：全球、企业、系统、应用、框架、微观架构、对象。注意有些模式同时出现在几个级别中，可能为每个级别都提出了不同的化解办法。

④ **化解办法的名称**（Refactored Solution Name） 化解办法的典型称谓是一种很重要的参考，尤其是当它与反模式名称一起被用来特指某个问题时。

⑤ **化解办法的类型**（Refactored Solution Type） 这用于识别使用反模式化解办法所取得的改进的类型。

- 软件——包括新软件的创造。
- 技术——通过采用技术来解决问题。
- 过程——提供对不断重复的活动的定义。
- 角色——清晰的组织干系人职责分配。

⑥ **根本原因**（Root Cause） 下面这些根本原因是造成反模式的主要起因：

- 仓促——导致软件质量受损。
- 贪婪——导致贪婪的决策和不必要的复杂性。
- 骄傲——“并不只有爱迪生才会发明”的综合症。
- 无知——懒散导致理解失败。
- 冷漠——对软件开发问题的解决漠不关心。
- 心胸狭窄——拒绝实施已知的、行之有效的软件解决办法。
- 懒惰——寻找容易答案的懒惰做法。
- 责任——普遍性起因。

⑦ **不平衡力**（Unbalanced Force） 这部分识别在反模式中被忽略了的、误用了的或过分使用了的主要力量：

- 功能管理——满足需求。
- 性能管理——满足所要求的运作速度和运作范围。
- 复杂度管理——定义抽象。
- 变更管理——控制软件的演变。
- IT 资源管理——资源评估、规划和控制。
- 技术转让管理——控制技术的变更。
- 风险——普遍力量。

⑧ **典型言语**（Anecdotal Evidence） 能言简意赅地说明问题的常见短语和幽默话语。

⑨ **背景**（Background） 这部分内容设定了反模式的场景并介绍将要讨论的问题。

⑩ **一般形式**（General Form） 确定反模式的一般特征，对问题的本质提出大概看法。

⑪ **症状与后果**（Symptom and Consequence） 在这部分当中，我们提供了反模式的症状及其造成的相关后果。

⑫ **典型起因**（Typical Cause） 这是一个反模式特有起因的列表。如果有可能，就应该与相应的症状与后果联系起来。

⑬ **众所周知的例外情况**（Known Exception） 反模式行为和过程并非总是错的，有的时候也有例外。这部分将简要说明每个反模式的主要例外情况。

⑭ **化解办法**（Refactored Solution） 化解办法解决反模式中存在的不平衡力、起因、症状与后果等问题。

⑮ **变化**（Variation） 列出了反模式已知的主要变化及所有可以使用的解决办法。通常，这个部分研究的是一种异常的反模式形式，这种异常的反模式形