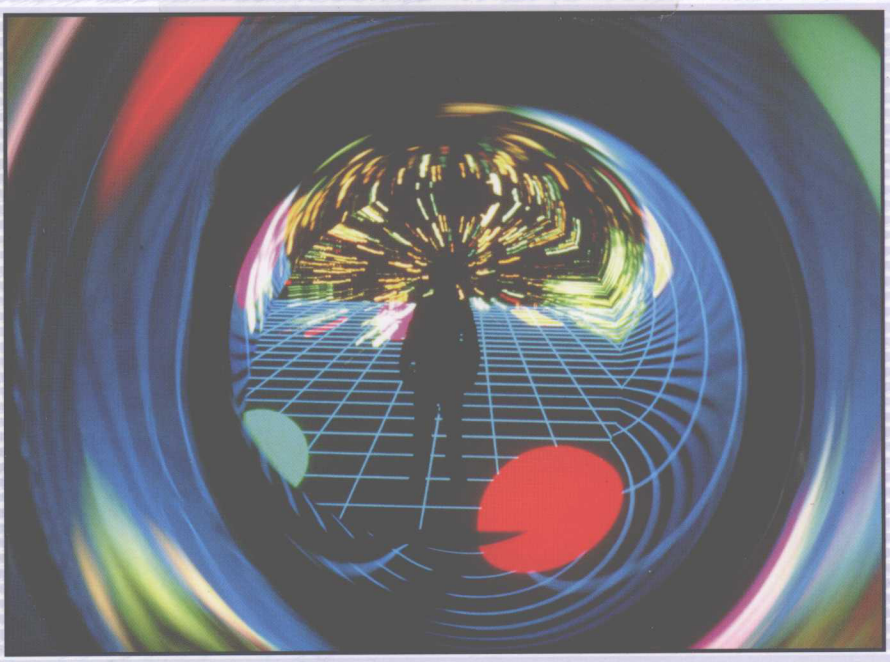
BO SHI 博士 丛 论 CONG LUN

软件维护中风险分析与故障管理策略研究

毛澄映 著



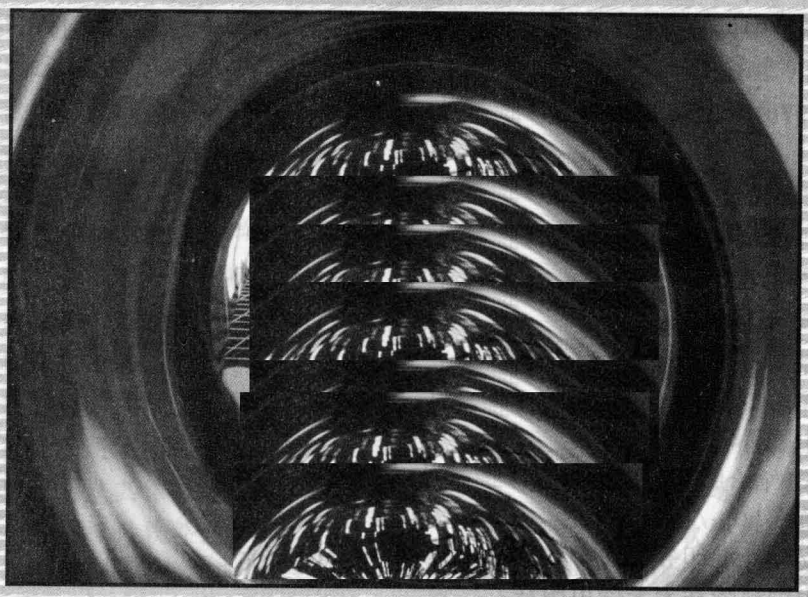
中国科学技术大学出版社

Ruanjian Weihu zhong Fengxian Fenxi yu Guzhang Guanli Celue Yanjiu

BO SHI 博士 论丛 GONG LUN

软件维护中风险分析与故障管理策略研究

毛澄映 著



中国科学技术大学出版社

内 容 简 介

本书在广泛调研软件维护活动中现有管理策略和方法等研究成果的基础上，在软件变更管理和风险分析方面开展研究，以软件需求变更和代码模块变更为切入点，综合运用人工神经网络、矩阵数值分析等手段量化地评估变更给系统带来的影响；在软件故障管理方面，运用 Rough 集推理技术和灰色系统理论实现对故障的有效定位和故障数目的预估计。

本书可以作为软件工程、计算机科学与技术等信息类学科的研究生教材，也可供有关科研人员和工程技术人员参考。

图书在版编目(CIP)数据

软件维护中风险分析与故障管理策略研究/毛澄映著. —合肥：中国科学技术大学出版社，2010.3

ISBN 978-7-312-02689-8

Ⅰ. 软… Ⅱ. 毛… Ⅲ. 软件维护 Ⅳ. TP311.53

中国版本图书馆 CIP 数据核字(2010)第 030422 号

出版 中国科学技术大学出版社
安徽省合肥市金寨路 96 号，230026
网址：http://press.ustc.edu.cn
印刷 安徽江淮印务有限责任公司
发行 中国科学技术大学出版社
经销 全国新华书店
开本 710 mm×1000 mm 1/16
印张 5.75
字数 119 千
版次 2010 年 3 月第 1 版
印次 2010 年 3 月第 1 次印刷
定价 18.00 元

前　言

随着软件系统应用的日益普及，其质量及可靠性问题引起广泛关注。然而，软件企业和开发人员常常将解决问题的思路集中于技术层面，认为有足够先进和成熟的技术就可以开发出质量上乘的软件。当我们总结众多软件项目失败的原因时发现：虽然有些时候软件项目失败的确由软件开发方法和技术引起，但更为普遍的、也无法回避的原因是在软件项目管理上的缺失或者不当。当前是我国软件产业的快速发展时期，软件出口和外包业务不断增长。但是，软件项目管理的水平和规范化程度远远滞后，指导、加强和规范我国软件开发组织的现代化管理任重而道远。目前软件项目管理大多是定性方法，对软件维护过程中保持高效、优质控制的方法研究甚少。正是在这样的一个背景下，笔者力图寻求有效的管理途径，保证软件维护阶段的资源科学配置和任务合理分配，促进软件产品质量的提高，适应我国信息社会和企业环境快速发展的要求。

软件维护主要是应对需求变更和软件故障而进行的活动。已有实验证明，耗费在软件维护阶段的代价一般要占整个软件项目的一半以上。同样地，目前软件工程界在软件维护方面的研究主要集中在相关辅助技术的改进方面，而没有认识到管理的重要性。从本质上看，在软件维护过程中的管理活动主要应对如下两个方面的问题：一是分析由于需求及模块变更导致的风险；二是对软件系统陆续出现的故障实施有效的管理。软件维护活动应着眼于软件开发全过程，针对不同阶段出现的问题采用不同的管理方法和策略加以应对。探求高效、可操作的软件项目变更风险分析及其维护管理策略是当前软件工程界亟待解决的课题。

本书主要探讨软件系统维护活动过程中的管理问题。针对软件需求和单元模块这两类变更，给出潜在风险的量化分析技术；并针对软件系统中的故障定位和预测问题，力图结合知识发现理论提高其准确性和有效性。

全书共分6章。第1章为绪论，分析了对软件项目维护活动实施管理的意义，介绍了国内外在软件维护中管理方法和策略的研究现状。第2章给出了一个需求变更分布变化原因的评定框架。结合用模糊逻辑方法得到的量化评定结果，运用人工神经网络(ANN)对需求变更(RC)的分布进行预测。此外，在预测得到的分布的基础上，提供了一个估计RC代价的模型。第3章以构件软件为例，分别运用构件依赖图和依赖矩阵两种模型进行变更风险分析。在计算过程中，分别就单个构件变更和多构件变更两种情况提出了各自的系统变更率的计算算法。并结合实例

系统的分析给出了所提出的变更风险度量的若干性质。第4章给出了一个基于Rough推理的故障诊断框架。针对功能性测试(黑盒测试)结果运用Rough集推理技术生成用于指导故障定位的知识性规则。进一步地,根据每个关键的输入参数可以进一步进行静态切片,从而指出与程序异常行为关联紧密的语句集。对于任一关联规则,调试人员可以针对每个输入参数计算出它对应的动态切片语句集,再通过交集运算得到可能潜伏故障的程序部分。第5章首先给出了基于GM(1,1)模型进行预测的基本框架性方法,随后针对另外两类特殊的应用要求给出了两种对应的预测模型及其实施步骤:一类是对软件故障数目进行区间预测;另一类则是结合维护过程中所记录的相关因素数据序列开展综合预测。第6章对全书的主要创新工作进行了总结,并展望了后续可能的研究方向。

本书的主要内容是笔者在华中科技大学管理科学与工程博士后流动站从事科研工作期间,结合国家自然科学基金项目《基于全寿命周期的企业IT项目风险测度研究》(No. 70571025)、《Web服务软件测试中用例生成及结果诊断方法研究》(No. 60803046)、教育部高等学校博士点基金项目《企业IT项目风险评估模型与规避策略研究》(No. 20060487005)、第41批国家博士后科学基金项目《软件项目过程与质量管理方法研究》(No. 20070410946)和华中科技大学第7批博士后科学基金项目《软件项目过程管理与改进关键策略研究》(No. 2007-07-44)等课题研究而形成的学术成果之一。在写作过程中,华中科技大学计算机科学与技术学院卢炎生教授、管理学院张金隆教授给予了悉心的指导并对书稿进行了耐心的审阅。还要感谢江西财经大学软件与通信工程学院的领导和老师给我提供了诸多便利与帮助,特别是夏家莉院长和尹爱华副院长的关心与督促,使得本书能及时出版。最后,衷心感谢家人多年来的鼓励与支持。

鉴于作者水平有限、经验不足,书中难免存在错误之处,恳请广大读者批评指正。

毛澄映

2009年9月

目　　录

1 绪　论

1.1 软件项目维护管理的意义

软件作为信息技术应用的载体已成为信息社会的重要基础并得到了广泛的应用。随着实际工作中对相关软件依赖程度的不断增加，其质量优劣问题成为开发者和使用者共同关注的焦点。虽然采用新型的编程语言、先进的开发技术和完善的开发过程可以在一定程度上减少缺陷的引入，但不可能完全杜绝软件中的错误。其原因包括多个方面，如程序设计错误、交流不畅、需求变更和时间压力等。因此，即使软件投入生产性运行，以后仍需要进行跟踪性的维护和管理。

所谓软件维护就是在软件已经交付使用之后，为了改正错误或满足新的需求而修改软件的过程[1,2]。按照软件维护过程中活动的性质可以将软件维护划分为如下四类：

(1) 改正性维护。虽然软件经过早期的测试，但不可能暴露系统中所有潜伏的故障。那么，在其后续的使用和测试中必然会出现错误。这类诊断和改正错误的过程称之为改正性维护。

(2) 适应性维护。计算机科学技术的各领域均在迅速发展，无论是硬件设备还是软件支撑系统均在不断地改进和升级。为了配合和适应软件应用环境的改变，必须对软件系统进行必要的修改，这种性质的维护我们称为适应性维护。

(3) 完善性维护。在使用软件的过程中用户往往会提出增加新功能或修改已有功能的建议，还可能提出建设性的改进意见。为了满足这类要求，需要进行完善性维护。

(4) 预防性维护。这是指为了改进软件未来的可维护性或可靠性，或为了给未来的改进奠定更好的基础而进行的修改活动。一般来讲，这类维护活动比较少见。

不难看出，软件维护主要是应对需求变更和软件故障而进行的活动。目前，软件工程界在软件维护方面的研究主要集中在相关辅助技术的改进方面，而没有认识到管理的重要性。然而，事实证明，许多软件维护的失败案例中并不是技术不够先进、人员不够优秀，而是由于管理上的缺失或者不当造成的。因此，软件项目维护管理方法及其策略的研究成为管理界和软件工程界都十分关注的热点问题。

当前是我国软件产业的快速发展时期，软件出口和外包业务不断增长。但是，软件项目管理的水平和规范化程度却远远滞后。根据中国科学院软件技术研究所的调查报告显示：截至2004年底，我国已实施CMM/CMMI的软件企业仅占全国软件企业的2%左右[3]。由此可见，指导、加强和规范我国软件开发组织的现代化管理任重而道远。目前软件项目管理大多是定性方法，对软件维护过程中保持高效、优质控制的方法研究甚少。正是在这样的背景下，笔者力图寻求有效的管理途径，保证软件维护阶段的资源科学配置和任务合理分配，促进软件产品质量的提高，适应我国信息社会和企业环境快速发展的要求。

1.2 软件维护及其管理的主要内容

软件维护是软件生命周期中一个极其重要的环节，通常占到整个软件预算的60%以上[4]。例如，文献[5,6]估计全球一年用于修正错误的直接费用就达到一亿美元。因此，多年来软件维护一直是软件工程界关注的重点。但遗憾的是，已经取得的成果主要侧重于技术、方法方面，而对于软件维护环节的管理问题涉及得不多。

软件后期出现问题大多数都可归因于软件开发阶段没有严格而科学的管理和规划。该问题需要引起足够的重视，在软件维护阶段不能重蹈覆辙。维护过程本质上是修改和压缩了的软件定义和开发的过程，而且事实上早在提出一项维护要求之前，与软件维护相关的工作就已经开始了。例如，近年来出现的测试驱动的软件开发(Test-Driven Development，TDD[7])就是最好的例证之一。现在软件维护活动需要贯穿于整个软件生命周期，逐渐得到认同并广为接受，相应地，软件维护过程中的管理问题也就涉及多个方面。在软件组织中既需要建立一个专门的维护组织，还需要对可能出现的故障和风险做出估计、处理和评价，还应该建立一个适用于维护活动的记录保管过程，并且制定审查标准。

一般而言，软件维护涉及的内容非常繁杂，既包括软件代码实体，还包括文档、数据，乃至软件开发人员和组织结构等。这些均加大了软件维护活动的复杂性，并导致该领域派生出多个研究方向。这里，我们选取其中最为重要的软件测试、变更管理、软件理解与重构和故障管理四个方面加以说明，并介绍其中涉及的管理问题。

1.2.1 软件测试及其管理问题

软件测试(Software Testing)是软件维护领域研究得最为深入也最为成熟的技术。软件测试是对软件功能、设计和实现的最终审定，是发现软件故障，保证软

件质量，提高软件可靠性的主要手段[8]。软件测试应尽可能有效地发现软件中存在的缺陷，同时软件测试也应该是高效的。IEEE 把软件测试定义为：从通常是无限大的执行域中恰当地选取一组有限测试用例，对照程序已经定义的预期行为，动态地检验程序的行为[9]。

由于软件是程序设计人员智力活动的产物[10]，因此在它之中不可避免地会潜留一些错误。这些错误大体上可以分为两类：语法错误(Syntax Errors)和语义错误(Semantic Errors)。对于语法错误，目前这些现代的编译器均能检测出来；而语义错误则需要通过测试来排除。软件测试是一个复杂的活动流程，涉及多个环节，需要科学地协调人力资源和资金等。特别是近年来提出的测试驱动的软件开发 TDD 和“全程软件测试[11]”的概念，更是将软件测试活动贯穿于整个软件生命周期，这样对该活动进行管理就在所难免了。软件测试管理是一种活动，可以对各阶段的测试计划、测试案例、测试流程进行管理、跟踪、记录，并将其结果反馈给系统的开发者和管理者[12]。同时将测试人员发现的错误即时记录下来，生成问题报告并对之进行管理。软件测试管理的对象可以归结为三大类，即产品(Product)、过程(Process)和人员(People)。

软件测试管理就是通过一定的管理方法和工具来对整个软件测试过程进行监控，从而提高软件测试工作的绩效。其管理的基本过程可用图 1.1 所示的模型框架进行刻画。可以看出，关于测试方面的管理主要体现在如下六个方面：测试需求管理、测试用例管理、测试过程管理、测试人员组织与管理、测试结果管理和测试报告管理。

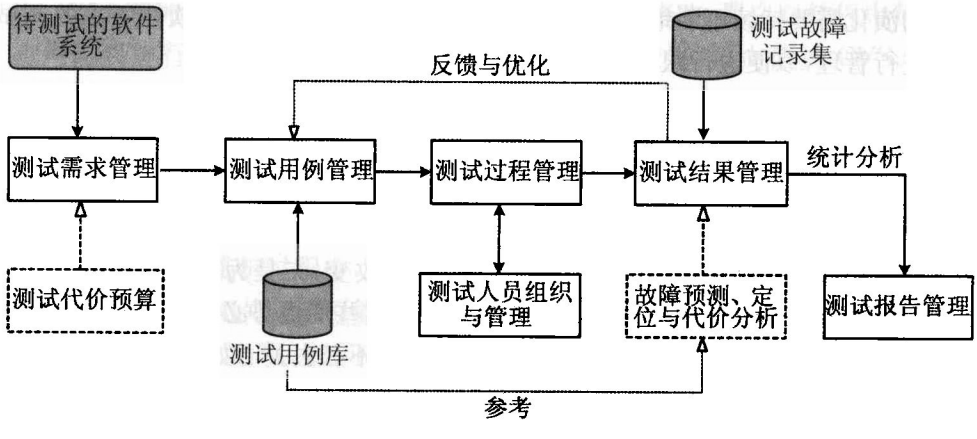


图 1.1 软件测试管理的基本过程模型

(1) 测试需求管理。在进行任何一个软件项目的测试之前，需要明确测试的目标，并依此制定测试计划。相关的工作包括：对测试需求进行重要性排序，初步预算需要涉及的人员及代价等。

(2) 测试用例管理。测试用例是测试活动中极其重要的输入数据，直接关系

到测试的成败。为了提高测试的效率和效果，通常需要对测试用例进行优化[13,14]、排序[15]和选择[16]等，并建立用例库对它进行合理的存储管理以便在后续测试活动或其他软件项目中进行复用。

(3) 测试过程管理。测试活动同样需要按照一定的策略来进行。在常规的测试过程模型(如V模型、W模型等)基础上，制定较为翔实的实施步骤。例如，对软件中的模块按照何种顺序开展测试，各个不同级别的测试又该采用何种策略，如何找到最佳的测试强度，以便获得最佳的代价-效益比[17]等。

(4) 测试人员组织与管理。测试过程中的人员管理是与测试过程管理密切相关的。针对不同的测试阶段，需要对测试人员进行重组；针对软件中不同复杂程度的模块，需要提供的测试强度也不相同。另外，除了要建立好测试组织内部人员之间通信、协调机制外，还需要对开发组织和测试组织之间的通信进行有效的管理。最近，结合社会网络(Social Network，也可译成社交网络)对软件组织进行管理[18]的途径较具特色。

(5) 测试结果管理。为了保证软件系统的可靠性，通常用于测试的用例一般数量巨大且覆盖完整。那么，经由测试产生的结果集往往也较为庞大。相应地，对这些测试结果进行管理也就成为一个突出问题。所涉及的问题包括：剔除重复的故障记录信息[19]，进行有效的故障定位[20]以及对潜伏在软件内部的故障数目作出预估[21]等。

(6) 测试报告管理。一方面最终形成的测试报告需要在软件开发组织内部流动，达到信息传递的目的，其本质是软件项目中的文档管理。另一方面，随着软件版本的演化需要持续不断地对软件实施测试，因此有必要对各个软件版本的测试报告进行管理，以便进行跟踪和比较。

1.2.2 软件变更管理

引起软件变更的原因大体上可以归纳为两个方面(如图1.2所示)：一是由于用户在试用或使用软件过程中需求的变化引起的改变；二是为了修正软件中已经发现的故障或增加某些新的功能而引起的改变。这些变更势必导致重新进行一次软件开发周期的微型循环。对于相关的管理而言，不必去关心如何修改代码，需要重点分析这些变更可能导致的后果，并且对修改的部分(例如语句行)进行可追溯性的管理。

美国Standish Group公司对8400个软件项目的调查和研究指出三种最经常使项目"遇到困难"的因素[22]，其中"不断改变的需求和规格说明"占所有项目的12%。并且B. Curtis研究表明，需求变更是软件开发中的三个重要问题之一[23]。用户在开发过程中提出需求变更的现象不可避免，能否合理地控制和管理需求变更是检验成熟软件组织的一个重要的标志。在用户需求变更的管理方面，主要是

通过对历史数据进行简单的统计分析(例如采用 SPC 技术[24]),预测后续阶段将会出现的变更数目,以便进行前瞻性的应对。同时,还需要分析需求变更过程中的相关数据,找出引起需求发生变更的主要因素,以便给出降低需求变更的应对措施。另外,在分析逐个变更可能引起的风险的基础上,在项目总体上进行变更风险的量化估计也是十分必要的。

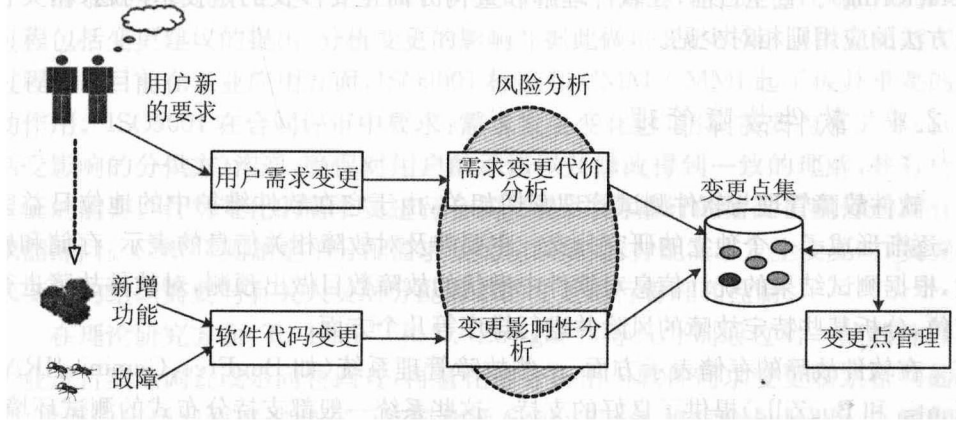


图 1.2 软件变更管理框架

在软件模块变更管理方面,主要是结合静态或动态的程序分析技术找出修改的模块可能会引起哪些波及影响效果,进而分析该模块的变更会给后续的维护造成多大的影响。

以上两种类型的变更,最终均会导致程序代码的修改,那么对软件中每个变更点均需要进行记录和管理。目前这方面工具支持较好,可由诸如 CVS、SVN 和 Visual SourceSafe 等版本控制系统辅助完成。

1.2.3 软件理解与重构

在软件系统维护过程中,一方面涉及的维护人员不可避免地会发生流动,另一方面早期设计时相关文档可能不十分齐全,那么当非系统开发人员维护软件时难以理解其结构、功能等。该问题在软件工程界早就得到重视,形成了一个重要的分支——软件理解(Software Comprehension/Understanding)。其主要涉及的工作包括:① 软件系统的逆向工程[25]。在得到需要维护的软件系统的代码后,可能一时无从下手,从而需要从系统全局的角度了解其模块构造。目前,比较典型的做法是根据程序源代码逆向生成方法调用图、类图等。在得到这些程序的概念模型之后,给维护人员理解软件功能分布及其结构提供了极大的帮助。② 软件代码的切片分析[26]。当维护人员深入到某个具体的程序片段中时,对某些语句难以理解。那么,需要将与当前语句相关的代码提取出来进行针对性的分析,也就是排除那些

不相关的语句。此外，维护人员还可能需要结合版本控制系统对某些模块的历史变更轨迹进行理解。

在软件演化过程中，由于需求的改变，原有的体系结构可能难以适应新的要求；另一方面，不断地对软件系统进行打补丁式的修复使得系统逐渐“老化”。基于这两个原因需要对系统的体系结构进行重新设计，即所谓的软件重构（Software Refactoring）。直至目前，在软件理解和重构方面主要涉及的是技术问题，相关管理方法的应用则相对少见。

1.2.4 软件故障管理

软件故障管理与软件测试管理密切相关，由于它在软件维护中的地位日益突出，逐渐形成了一个独立的研究分支。主要涉及对故障相关信息的表示、存储和检索，根据测试结果的统计信息对软件中潜伏的故障数目做出预测，对软件故障进行定位，分析某些特定故障的风险并进行排序等几个方面。

在软件故障的存储表示方面，一些故障管理系统（如 BugFree、Gemini、JIRA、Mantis 和 BugZilla）提供了良好的支持。这些系统一般都支持分布式的测试环境，除了能记录和管理故障信息外，还提供了测试人员的管理功能。故障数目预测则主要是根据历史故障信息和当前项目的一些特征数据（如软件复杂性度量指标）建立诸如回归分析等预测模型进行预测[21]。

在软件故障定位方面，基本的思路是根据测试用例执行的记录信息，利用统计分析和数据挖掘相关手段得到隐藏故障候选语句集[20]。典型的做法是，在测试时通过植入探针记录程序的执行剖面，接下来分析成功运行用例的剖面和失败运行用例的剖面之间的差异，进而从概率意义上给出最有可能包含故障的语句部分。

在定位得到一批软件故障后，并不是没有规划地开展调试并排除它们。科学的做法应该是对这些故障实施风险分析，根据它们所造成的软件失效的严重性程度进行排序，以便在后续的排错阶段集中精力和资金解决那些最为紧迫的问题，而其他不是十分重要的故障最后进行排除或暂时忽略处理等。

以上这四个方面是软件维护中的核心问题，从管理的角度看，测试、变更以及软件故障方面的管理更加值得关注。软件测试方面的管理渐趋成熟，在工业界已经形成了初步的管理规范和国际标准。本书主要在软件变更和软件故障的管理方面进行研究，力图给出一些可行且实用的解决方案。接下来，对这两方面的研究现状进行分析和总结。

1.3 软件变更管理研究现状

软件变更往往呈现出不确定性，因此如何应对这些不定期出现的变更成为一

项极其困难的事情。如果不能对软件变更及时有效地控制管理，很可能对整个项目造成“牵一发而动全身”的影响，所以对软件整个过程管理的一个重要内容就是对软件变更加以控制，使变更对工作量、工期及质量的影响降到最小。如 1.2.2 节所述，软件变更的管理问题主要涉及两个方面，其一是对需求变更的管理，其二是对软件模块变更的管理。

需求变更管理主要是一个在项目开发过程中对需求变更进行控制的过程，该过程包括变更建议的提出，分析变更的影响并据此做出变更决策，监督变更的实施过程等。目前在工业应用方面，ISO9001 标准和 CMM/CMMI 起了极其重要的推动作用。ISO9001 在合同评审中要求：需求发生变化时，组织需要与有关部门（包括受影响的分供方）沟通，确保对用户的承诺以及修改得到一致的理解，并有持续保证的能力。针对变化的需求要进行合同修订，并将修订的情况正确传递到有关职能部门。CMMI 则制定了分配需求变更控制规则：分配需求发生变更时，负责需求变更的组织需要与有关人员对分配需求的变更进行管理和控制。

在理论研究方面，T. Hall[27] 和 D. Zowghi[28] 等人分别通过对一些软件开发企业进行实地调查或是问卷调查，再量化地分析出与软件需求变更联系密切的因素，进而给出适当的改进措施。英国学者 Simon Lock 和 Gerald Kotonya 多年来一直致力于需求变更影响性的分析工作，提出了一个用以对需求变更进行影响性分析的集成框架[29]，并利用基于场景的可视化技术对需求变更进行分析、追踪和控制[30]。类似地，G. H. Galal 和 R. J. Paul 在基于场景的需求演化分析当中，运用量化的分析技术对需求进行有效的评估[31]。Jane Cleland-Huang 等人则提出了基于事件的软件需求追踪机制[32]，该方法通过采用事件服务使得追踪工件（Artifact）间不再紧密耦合。在国内，中国科学院软件所互联网软件技术（iTechs）实验室一直致力于需求变更相关的管理和度量方面的工作[24,33]。西北工业大学的张强等人运用软件度量的方法改进传统的变更管理过程，并给出了在软件变更管理中应用度量的完整实施过程[34]。南京大学的江宴等人[35]提出了一种基于估算的软件过程变更方法，并介绍了该方法在一个过程支持系统 CPMS 中的实现。

除了需求变更是导致软件项目风险的一个重要因素外，由于排除缺陷或功能改进等原因不可避免地需要对系统中的某些模块进行修改，这类修改所引起的风险（影响）分析问题形成了一个研究子方向（即变更影响分析）[36]。在分析变更部分对系统的影响方面主要分为两大类：一类是在程序代码上分析，另一类则是在程序模型上分析。对于前者主要是采用传统的程序切片技术[26]，由于当今的软件系统往往都比较庞杂，基于软件设计结果（例如软件体系结构）的变更影响分析的应用更加普遍。

赵建军和杨宏戟等人[37]通过分析软件体系结构的形式化描述规约，并实施切片（Slicing）和削片（Chopping）两种操作，进而得到受变更影响的软件模块。由于软件模块的修改以及对其他模块的波及，最终的效果就是会对软件的可靠性造成

影响。因此，软件变更条件下的软件可靠性影响分析同样是近年来的研究热点。在这方面较为突出的有：Ammar 等人将着色 Petri 网作为工具进行模拟分析用以度量软件系统的动态复杂性[38]；Yacoub 等人运用场景模拟实现动态度量，进而完成软件的可靠性风险分析[39]；Popstojanova 等人在标准 UML 表示的基础上形成离散时间 Markov 链，进而依此评估软件系统的风险[40]。

1.4 软件故障管理现状分析

如前所述，软件故障的管理主要体现在故障定位和故障数目预测这两个方面。该方向的研究由来已久并形成了不少可取的方法，近年来随着数据挖掘和知识发现领域的快速发展，一些相关的方法被应用到这个方向上来，获得了不少新颖的成果。

1.4.1 软件故障定位

为了保证测试覆盖的充分性，在实际测试活动中一般会用大批量的测试用例进行测试，从而导致引发程序失效的用例数目不可小视。那么，在后续的程序诊断过程中如何快速而全面地找出故障就困难起来，仅靠调试人员进行人工检查难以奏效，而结合数据挖掘技术往往能得到比较理想的效果。

在软件维护实践中能发现这样一条规律：程序的多次失效往往仅由同一个故障引起。如此看来，就没有必要对每次失效都进行调试，对它们进行聚类就能避免这类重复。Dickinson 和 Podgurski 首先提出了对程序失效进行聚类以降低调试代价的思想[41]，毛澄映等人通过将失效与非失效用例分离、相异性度量改进等手段提高了聚类精度[42]。还有研究者通过分析程序失效与函数返回值、函数调用对等之间的关联信息进行故障定位[43,44]；类似地，Li 和 Zhou[45] 运用频繁项集挖掘技术对程序元素（函数、变量和数据类型等）进行使用模式挖掘，再运用挖掘结果指导故障定位。

程序切片[26]是最传统的故障定位技术，但维护人员在理解大型复杂系统的切片（特别是静态切片）结果时仍存在困难，从而难以准确定位故障。基于上述事实，可以在切片结果或程序插装记录上作进一步地统计分析[46]，大大缩小了故障查找的范围。另一种相近思想的处理方法是：对用例的每次执行跟踪均作记录，再对这些跟踪作谱分析并可视化地显示与失效联系紧密的程序语句集[47]。刘彦斌等人采用程序谱来抽象表达程序执行轨迹，根据成功的运行和含有故障的运行之间的差异来进行故障定位[48]。

此外，Renieris 和 Reiss 在分析程序运行剖面的距离和差异的基础上，给出了

基于最近邻查询的故障定位方法，并提出了故障定位的通用框架[49]。吴际等人[50]进行软件输入级的故障定位，定义了软件输入响应对象及其构成的测试序列，以及测试用例对应的失效观察序列，建立了故障定位的统计模型。Ren 等人[51]则从程序演化的角度看待故障定位问题，运用分类技术和启发式规则发现可能引起程序失效的变更。

1.4.2 故障(失效)预测

根据软件的度量值或初步测试结果对其中的故障进行预测将为软件系统的后期维护提供有意义的指导。目前这方面的研究可分为两大类[52]：① 基于量化分析技术，对软件系统中潜伏故障的数目进行预测；② 基于分类技术，关注预测系统中哪些模块具有故障倾向。就研究现状而言，数据挖掘领域的绝大多数分类和预测技术在软件故障或失效的预测应用方面均有体现。

1. 软件故障数目预测

故障数目预测可用静态预测和动态预测两种方法解决。所谓静态方法就是利用程序的一些静态结构信息(如程序复杂性度量值)进行预测；动态方法则是根据历史数据进行学习后开展预测。通常，静态预测方法是根据实践统计数据形成一些经验公式，有 Akiyama 模型、Halstead 模型和 Linpow 模型等。

早期的动态预测方法是根据已有的部分测试记录构造一些预测误差尽可能低的回归模型(典型的有线性回归模型、泊松回归模型等)；近期，Ostrand 通过采用负二项回归模型(Negative Binomial Regression Model)进行实证研究表明回归模型仍不失为一种较好的预测方法[53]。Khoshgoftaar 等人首先将神经网络方法引入到软件故障的预测中，将程序复杂性度量作为输入变量，通过实验发现神经网络模型相较传统的线性回归模型预测精度更高，且降低了数据预处理的要求[54]；随后又提出了案例推理(Case-based Reasoning，CBR)的方法[55]，并通过实验证实较多元线性回归有更好的性能。回归树(Regression Tree)是由抽象树模型表示的决策规则的集合，一些独立变量(称为预测器)用于预测诸如故障数目等响应变量。该模型具有很好的解释性，相对于神经网络方法在速度上有优势，且容易转化为 SQL 语句。基于这些特点，Seliya 将 S-PLUS、CART 两种回归树应用到程序故障预测问题中，并通过实验发现 CART 回归树在解决该问题时效果相对较好[56]。Amasaki[57]和罗云峰[52]两个研究小组各自独立地运用 Bayes 信念网预测软件故障，其相同点是从软件开发全生命周期来考察故障，但两者最终的信念网形式存在差别。最近，Pai 和 Dugan 提出了一个预测类中故障数和故障模块的 Bayes 信念网框架[58]，其特点是分别就过程度量值和软件产品度量值建立 Bayes 网，再综合开展预测。而 Pighin 等人[59]则运用线性规划的方法预测程序(即文件)存在故障的风险(在他们的研究中风险用故障数目度量)，通过设定一个故障数目阈值可以过

滤出高风险的程序，实验结果表明该方法在预测精度上比决策树(Decision Tree)方法要高。

2. 故障倾向模块预测

除了预知故障数目为后期维护提供科学决策外，维护人员往往更关心哪些模块最有可能包含故障，称这样的模块为故障倾向(Fault-Proness)模块。由于故障倾向模块的预测在维护活动中更具应用价值(例如可为模块测试及快速诊断提供指导)，因此这方面的研究非常活跃，我们将一些代表性的方法进行归纳，形成表 1.1，文献[60]也对一些常用预测方法作了对比分析。从数据挖掘的角度看，故障数目与故障倾向模块的预测之间并没有本质区别，因而所采用的方法均以分类技术居多。再者，两种预测结果往往可以互相转化：根据预测的故障数目，结合故障分布的分析可以估计出故障倾向模块；反之，根据故障倾向模块的发现也可以初步估算出整个系统潜伏的故障数目。

表 1.1 故障倾向模块预测方法对比

采用技术	提出者	特点
最优子集回归 OSR	Briand 等[61]	使用分类回归树作为预测模型，其核心思想是对训练集进行适当的裁减来获得更好的分类回归树结构，目前应用较少
分类树/回归树	Khoshgoftaar 等[62]	树通过 TREEDISC 算法(CHAID 算法的一种改进)构造，能平衡两种类型的误分(非故障模块当作故障模块，反之为第二类)，当故障模块比较少时特别有效
The Top Ten List	Hassan 和 Holt[63]	运用最频繁修改 MFM 等四种启发式规则列举出前十个最有可能包含故障的模块，列表可动态调整，非常适合测试资源受限的项目
Bayes 信念网	Cheng、Cheung 和 Tse[64]	从他们对一个开源多媒体系统的实验分析来看，三种技术在预测准确性上并无明显区分，平均在 69.3%左右
kNN		
神经网络		

3. 关键度量因素选取

在进行软件故障预测时，关于软件的度量信息是至关重要的输入，通常可分为软件结构度量、软件过程度量和软件执行度量三大类。一般而言，众多的度量指标之间存在关联，并且如果预测模型的输入量过多将降低可操作性。为此，从度量指标集中选取具有代表性的度量子集也是一项重要任务，目前使用的技术也主要是一些统计分析技术，如区分分析、主成分分析等。