HZ BOOKS
华章教育

计 算 机 科 学 丛 书

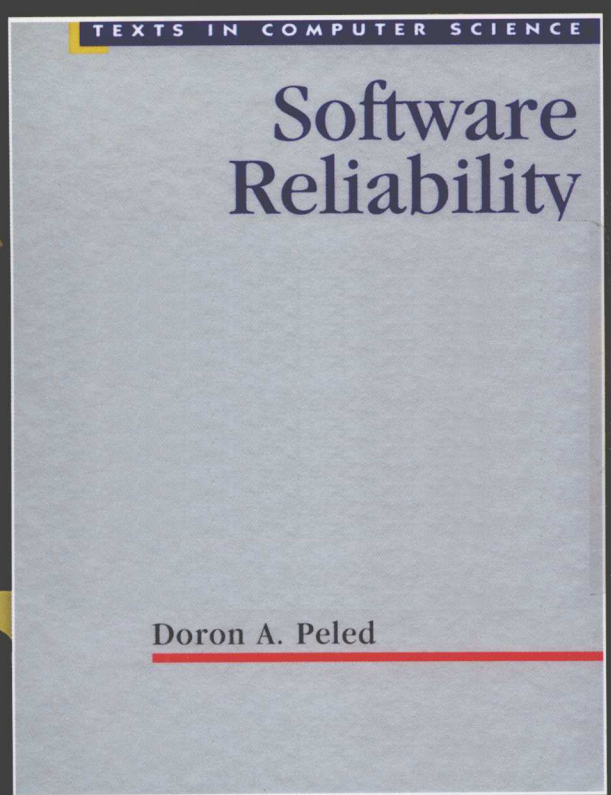
Springer

软件可靠性方法

（以）Doron A. Peled 著

王林章 卜磊 陈鑫 张天 赵建华 李宣东 译

Software Reliability Methods



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

软件可靠性方法

(以) Doron A. Peled 著

王林章 卜磊 陈鑫 张天 赵建华 李宣东 译

Software Reliability Methods

机械工业出版社
China Machine Press

本书通过大量的形式化表示和技术，向读者提供了各种用于提高软件可靠性的形式化方法，包括演绎验证、自动验证、测试以及进程代数。书中紧紧围绕逻辑和自动机理论这条主线，比较了各种方法的不同之处，并讨论了它们的优缺点。

书中包含一些在多个章节中使用的、具有连续性的实例，有利于读者通过跟踪这些实例来了解不同形式化方法的优缺点。本书还包括大量的练习和项目，可以使用软件可靠性工具来完成。

本书适用于从事软件开发的广大读者，尤其适合作为高年级本科生和硕士生的教材和参考书。

本书版权登记号：图字：01-2011-3360

图书在版编目（CIP）数据

软件可靠性方法 /（以）佩莱得（Peled，D. A.）著；王林章等译. —北京：机械工业出版社，2012.2

（计算机科学丛书）

书名原文：Software Reliability Methods

ISBN 978-7-111-36553-2

Ⅰ. 软…　Ⅱ. ①佩…　②王…　Ⅲ. 软件可靠性　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2011）第 242836 号

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：夏平

北京瑞德印刷有限公司印刷

2012 年 3 月第 1 版第 1 次印刷

185mm×260mm · 13.25 印张

标准书号：ISBN 978-7-111-36553-2

定价：45.00 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

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www. hzbook. com
电子邮件：hzjsj@hzbook. com
联系电话：(010) 88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章教育
华章科技图书出版中心

中文版序

Software Reliability Methods

很高兴看到我的著作《Software Reliability Methods》被翻译成中文。随着软件工业和硬件工业的兴起，研究提高计算机系统质量的方法日趋成熟，并逐渐成为较大的研究领域。形式化方法已经在蓬勃发展了，这不仅表现在大量的国际会议和致力于这方面研究的大学教授上，还体现在大公司里实际采用的工具和开发小组中。甚至有人声称，仅是投在验证软件（比如，软件测试）上的精力，就与编写软件的投入旗鼓相当。

程序员排错与软件开发的历史一样长。而如今，软件规模庞大，参与的开发人员众多，不再允许把检测和改正代码的责任加在程序员自己身上。同时，在人们日常生活中软件的广泛使用使得保持其正确责任重大。因此，软件测试和验证往往由独立专家小组使用能将大部分工作自动化的工具来执行。

本书着重论述在质量保证任务中所用到的主要方法，其中包括推理验证、测试和检验，此外还讨论如何协调这些方法以及它们之间的联系。在20世纪60年代后期，有学者提出用推理验证的方法来证明程序的正确性，用基于逻辑的数学形式化方法描述的程序来执行验证，并开发了证明系统来形式地证明程序行为与规约一致。开发这种证明系统时，必须研究程序的形式语义，将代码与程序行为及相关的形式化逻辑联系起来。早期的证明系统仅处理顺序程序，之后出现了能处理并行程序的证明系统。20世纪70年代后期，出现了最早的具有形式化定义的软件测试方法，并有了理论基础，此后出现了基于数学规律的代码采样和检查过程，因此能够用各种方法提高测试的效率，加强测试的效果。20世纪80年代初期，有人提出了验证有限状态系统的自动化方法，即模型检验，随后出现了各种相关支撑工具。

这三种软件可靠性技术在许多方面都大相径庭。对于推理验证技术来说，只有能够使用证明系统的逻辑学家才能驾驭。同时，由于其整个过程基本上都是细致繁琐的人力劳动，因此很难将其扩展应用于大规模甚至中等规模的代码。测试是现阶段保证软件可靠性的最佳方法。这种技术往往不对代码进行通彻的全面检查，也不保证能找到所有的编码错误或者设计问题。但是，它可以很好地用于大系统，如果仅从一种技术所能找到的错误以及耗用的投资来看，它的投入产出比最高。模型检验使用高端计算机、借助于高效算法和数据结构的好的研究成果，其最终目的就是自动完成全面检验，一方面其检验的通彻程度能与推理验证技术相媲美，另一方面其在大系统上的运用效果能和软件测试相匹敌。

最近十年，形式化方法研究上进行了巨大投入，其结果是利用现有的工具和技术，我们可以用更快、更节省内存的算法来检查更大规模、更多类型的系统。部分研究致力于发现效率更高的算法。另一部分研究的目标是将不同方法和技术组合在一起，利用各个方法的优点，减少其限制的影响，完成追求软件可靠性的整体任务。结合了验证的思路之后，测试工具就可以自动找到很好覆盖被检系统的测试用例集；组合了测试和模型检验的原理，允许我们在缺乏完整的设计描述的情况下直接测试系统。而用组合、抽象的技术，我们可以先对代码的紧缩版本或子集进行模型检验，然后直接利用所得到的结果或者将其与其他结果组合起来，得到对真正的更大的系统的推出结论。

形式化方法的研究牵涉了数学、计算机科学中的不同领域。本书中描述的主要技术是基于自动机理论和逻辑。而形式化方法、技术和工具中还用到一些更深的思想，比如机器学习、微分方程、数据结构、博弈论等。经过对其多年的研究，我已经确信这是最有意思的研究领域。同时，我也有幸见证了它从仅仅理论研究发展到被大的软硬件公司部署使用。但是，为了追寻更高效、更好用的技术，还需要克服很多困难，解决很多挑战。比如说，如何自动根据形式规约生成正确的代码？尽管已经有些很不错的结果了，但这一问题的研究仍处于起步阶段。

对这个令人兴奋而又实用的跨学科研究领域，我充满了热忱，并将其灌注在本书行文当中，现在它被翻译为中文并出版，作为作者，我最大的希望就是它能够感染、打动并帮助那些渴望学习、使用甚至研制软件可靠性方法的中文读者。

Doron A. Peled

2012 年 2 月 1 日

译者序

Software Reliability Methods

进入21世纪以来，“高可信软件”先后被美国、欧盟列为优先资助的研究方向，我国科技部和国家自然科学基金委员会也在2007年分别为其设立了重点项目和重大研究计划，由此广泛推动了国内高校和科研单位在相关领域的研究工作。译者所在的南京大学软件工程组长期以来一直在开展与可信软件方法和技术相关的教学和研究工作，Doron A. Peled的《Software Reliability Methods》近年来一直被我们用做研究生的入门教材，是进入我们研究组的研究生必读的书籍。该书围绕软件可靠性较为全面、系统地介绍了相关方法和技术，深入浅出、通俗易懂。该书还受到了图灵奖获得者E. Clarke的大力推荐，作者本人在相关领域也很有建树。

2010年12月底，我与来访的机械工业出版社编辑姚蕾女士交流时，她了解到我们在用这本书，建议我们将它翻译成中文，以便让更多的国内读者能够深入理解作者的观点以及书中介绍的方法，从而在研究与实践中放手尝试。我们接受了她的建议，从2011年1月初开始着手翻译。参与本书翻译工作的主要是南京大学计算机软件新技术国家重点实验室、计算机科学与技术系软件工程组的老师和学生，分工情况如下：王林章（第9，10章、结束语）、卜磊（第5，6章）、陈鑫（第7，8章）、张天（第4，11章）、赵建华（第1，2，3章），王林章、李宣东对全书译稿进行了统稿、审校和修改。在翻译过程中得到研究生崔展齐、陈华杰、丁文旭、姜鹏、李游、李袁奎、柳溪、潘敏学、陶永晶、王寒非、邢雨辰、叶楠、张琦、周筱羽等的帮助，在此对他（她）们的辛勤劳动表示感谢。也感谢编辑王春华女士、夏平女士为本书所做的大量工作以及耐心等待。

尽管本书的翻译工作前后历时一年，但这是在大家繁忙的教学、科研以及各项服务工作之余完成的，所以仍感到时间紧张，有些内容的翻译表达仍不够理想。我们采用自己的研究方法来理解原书内容，对原文中我们认为表达错误或不确切的地方，也通过注释进行说明。限于我们的水平，中文表达难免有不当之处，在此敬请读者批评指正。如发现文字错误，请发邮件给王林章（lzwang@nju.edu.cn）。

王林章

2011年12月31日

我很荣幸能够为 Doron Peled 关于软件可靠性方法的新书写序。当我第一次翻开这本书时，我立刻被这本书的覆盖范围之广所深深打动，它覆盖了下列方面：

- 规约和建模
- 演绎验证
- 模型检验
- 进程代数
- 程序测试
- 状态与消息序列图

除了对每个方法进行了相当深入的介绍以外，本书还讨论了应当在何时选取何种方法以及在选择这些方法时所必须做出的权衡。书中结合当前最新的工具，使用很多具有挑战性的实例来说明各种技术。书中甚至告知读者在网络上到哪里去获取这些工具！我还没看见过其他任何覆盖同样内容的书籍能达到如此的深度。

同时，本书描述了应用形式化方法的过程：从建模和规约开始，然后选择一个合适的验证技术，最后测试程序。这些知识在实践中是十分必要的，但是却很少在软件工程的课本里面出现。绝大多数相关书籍都重点关注某一项技术，例如程序测试，而没有覆盖到其他的验证技术，或者对如何将相关技术结合在一起使用进行讨论。由于 Doron 已经为书中所描述的许多验证方法的发展做出了重要的贡献，因此他对这些关键问题的相关见解非常重要。

这本书适用于参与软件开发的广大读者，尤其适合高年级本科生软件可靠性课程或者硕士阶段软件工程课程使用。实际上，书中非常充分地标注了相关进阶文献，这些文献可以用做从事代码验证的软件工程师或者形式化方法方面研究人员的参考书目。

我刚刚和 Doron 一起合作完成了一本关于模型检验技术的书，在此过程中，他作为计算机科学家的智慧与作为作家的写作技巧给我留下了深刻的印象。我确信这本书将会取得巨大的成功。我向所有对软件可靠性问题感兴趣的读者强烈推荐这本书。

Edmund M. Clarke

前 言

Software Reliability Methods

许多书籍描述了如何通过应用形式化方法来提高软件的质量。然而，大多数书籍都是围绕着某个特定的方法，并将该方法作为软件可靠性问题的推荐解决方案。本书通过大量的形式化符号表示法和技术，向读者呈现了形式化方法更加全面的描述。书中将会比较它们之间的不同之处，并讨论它们的优点与缺点。

形式化方法的主要挑战之一在于怎样将研究人员开发出的技术传播到软件开发社区中去。最近一段时间，我们似乎开始对形式化方法工具的重要组成部分有了更好的理解。这表现为此类工具越来越多地被软、硬件开发业界所接受。理想情况下，形式化方法必须可以被直观地使用（最好是使用图形化界面），不要求使用者花费大量的学习时间，并且在开发过程中只产生比较小的额外开销。相比于 10 年或 20 年前，形式化方法更容易被人们所接受，这在硬件领域尤为显著。然而，不同的方法之间仍然存在激烈的争论。

本书重点是通过一系列技术来讲述形式化方法的主要原理。有很多在撰写本书时已有的先进技术没有包括在本书中。这些先进的技术涉及实时系统和混成系统、描述规约的形式化体系，以及一些专用的数据结构，比如二元决策图。本书没有包含这些特定的技术，但这并不意味着本书中的方法优于它们，而是因为本书描述的算法和方法已经应用于目前最先进的软件可靠性工具中了。挑选出来这些方法仅仅是想以举一反三的方式来表现形式化方法这一主题。然而，挑选时总是不可避免地倾向于选择和我的研究方向比较接近的主题。贯穿本书的主线是*逻辑和自动机理论*。有兴趣的读者可以在相关章节后列出的其他书籍和研究论文中找到这些高级方法的细节。

如果没有直接使用相关工具的实际经验，对形式化方法的学习就是不完整的。本书包含了大量的练习和项目，可以使用软件可靠性工具来完成它们。书中还有一些在多个章节中使用的、具有连续性的实例。学习形式化方法并且了解其优点和缺陷的一个有效方法是跟踪理解这些跨章节的实例。在一些情况下，后面的章节会详述某个在前面章节的练习部分提到的连续性实例。这样做的另一个目的是帮助读者检查对前面章节练习的解答（而不是仅仅提供一个明确的答案）。我们鼓励读者去检查所获得的关于这些例子的直觉感受可否帮助改进他们对之前练习的解答。

书中提供的绝大部分练习和项目可以选择某个工具来完成。尽管一些软件可靠性工具要求不菲的许可费用，但是很多工具允许出于非盈利目的免费使用。使用这些工具通常需要从这些工具的万维网页下载并按照网页上的指示进行安装。相关章节后面列出了一些工具和相应网页。需要注意的是，即使有些工具不需要取得许可证就可以使用，它们也常常要求使用者向工具的开发者发送一份信件，同意遵守工具的使用条款。在许多情况下，这些条款限定工具只能用于学术目的，并且开发者不对因使用工具而可能造成的损害负责。由于网页和网址往往会变更，同时也因为新的工具不断被开发出来替代已有的工具，因此我不能保证书中列出的网页信息一直有效。此外，本书不能够保证这些工具可以在任何环境下都正常工作。

不同的团体对形式化方法有不同的兴趣，当然不可能写出一本对项目经理、软件开发人员、质量保证团队和研究人员具有相同吸引力的书，但是我仍然尝试着在书中加入令每个读者群中的成员都感兴趣的内容。因此，读者在阅读的时候可以跳过那些理论性或技术性太强的章节。应当指出的是，本书的重点主要是*技术*，而不是*方法学*。

本书在描述一些形式化方法的同时还给出了相应的算法，理解这些算法对于运用这些方法

并不是必不可少的，但是理解它们可以更加深入地了解这些方法的工作原理。本书省略了大多数和书中的形式化方法相关的数学证明，但有时会包含简略的证明，以便增加读者的直观理解。

作者在此感谢下面这些参与和本书相关的启发性讨论并提出有益建议的人员：Nina Amla、Christel Baier、David Basin、Shai Ben-David、Roderick Bloem、Glenn Bruns、Ed Clarke、Dennis Dams、Xiaoqun Du、Kousha Etessami、Amy Felty、Elsa Gunter、Doug Howe、Orna Kupferman、Bart Knaack、Bob Kurshan、Bengt Jonsson、Leonid Libkin、Anca Muscholl、Kedar Namjoshi、Wojciech Penczek、Kavita Ravi、Natarajan Shankar、Natasha Sharygina、Marian Srenby、Richard Tefler、Wolfgang Thomas、Moshe Vardi、Igor Walukiewicz、Thomas Wilke、Mihalis Yannakakis 和 Lenore Zuck。事实上，写这本书的最大收获之一是有机会从实践者和专家就某些特定主题给出的建议和评论中进一步学习。

我并不是第一个引用刘易斯·卡洛尔（Lewis Carroll）的探险小说[⊖]中的词句的。但是，鲜为人知的是 Charles Lutwidge Dodgson（笔名是 Lewis Carroll）是一个研究逻辑可视化表示的数学家。他的“biliteral”（两字母）和“triliteral”（三字母）图是卡诺图的前身。卡诺图能够用一种易教易学的方式来表示逻辑，这也是很多形式化方法的最新研究趋势。

Doron Peled

2001 年 3 月于新泽西州莫雷山

⊖ 本书中所引《Alice's Adventures in Wonderland》中词句的中文翻译引自陈复庵先生所翻译的《阿丽思漫游奇境记》（中国对外翻译出版公司 1987 年版），所引《Through the Looking Glass》中词句的中文翻译引自许季鸿先生所翻译的《艾丽丝镜中奇遇记》（文化艺术出版社 1986 年版）。本书译者将书中主人公名译成“爱丽丝”。——译者注

目 录

Software Reliability Methods

第1章
Software Reliability Methods

引　言

“请示陛下，我从哪儿念起?”他问道。“从一开头念起，”国王非常庄重地说，“一直念到末尾，然后停止。”

刘易斯·卡洛尔《爱丽丝漫游奇境记》

在1999年年底，整个世界忧虑地等待着2000年的到来，人们忧虑的重点是控制要害系统的计算机可能会造成一些潜在的破坏。日历中年份的变化和计算机存储器中表示20世纪年份的传统方式是可能导致这些破坏的原因：传统的方法仅使用从00到99的两位有效数字来表示年份。这个出乎意料的小细节使一些人预感到将会发生的极大损失。它会影响到由软件驱动的电子系统，例如交通控制、核导弹、核反应堆、银行系统、抚恤金计划、电力和用水供应。仅美国就花费了超过1 000亿美元以应对这种危害，这就是“千年虫”。在日期发生变化的前一刻，一些人躲进了自制的避难所，同时手电筒和瓶装水成为普遍的需求。美国和俄罗斯的军事联合小组则在北美航空航天防御司令部（NORAD）度过了1999年12月31日的夜晚，他们共同监测天空，随时对可能发生的导弹计算机错误进行预警，这类错误可能会导致导弹失控而自动发射。午夜，1999年12月31日过去了，时间跨进了新的千年，这期间除了一些小故障以外，没有发生重大的事件。

计算机系统已经控制了我们生活中的方方面面，电话系统、商店的结账登记系统、票务预订系统、医疗系统、金融系统都已经高度计算机化。大多数情况下，计算机之间的金融数据通信已经代替了实际的纸币交易。计算机甚至可以负责处理民用飞机飞行中要求的很多活动。计算机系统的故障已经导致了一些严重的后果，其中包括死亡事故、自动关闭要害系统以及金钱损失。

软件开发行业在过去的数十年里正以史无前例的速度发展。硬件成本，尤其是内存价格不断下降。因特网实际上已经将整个世界变成了一个巨大的计算机网络，在这个网络里，通信、信息处理和商业交易持续不断地在线进行着。随着这些技术的不断更新，软件开发行业也在发生巨变。现在，由一个天才程序员单枪匹马地开发出整个软件系统的情况已经不再可能，取而代之的是，数十个甚至数千个程序员一起参与到开发过程中。不同的程序员开发同一个程序的不同模块，生产出几千行甚至数百万行代码。这些程序员可能在不同的地方工作，甚至可能素昧平生。

毫不奇怪，软件开发过程中的质量控制变得越来越困难，确保不同软件开发团队开发的产品能够成功地组合在一起正确运行则是一项不简单的任务。应对这一问题的主要技术之一是运用适当的设计方法学，软件工程学因此提出了很多技术来提高软件产品的质量。

正如许多软件工程方法所指出的，即使遵循了很多优秀设计准则和优良编程规范，程序可能仍然包含错误。不同的统计表明，即使由富有经验并训练有素的程序员来编写代码，每千行代码中总是会包含少量错误。因此，使用一些方法来消除程序代码中的人为错误就变得非常重要，这就是*形式化方法技术*（formal methods technique）的目的所在。多种软件工程方法试图指导软件的开发过程，但形式化方法的目的是为开发过程提供一些辅助性的技术和工具，用于发现并指出软件中的潜在问题。

1.1 形式化方法

形式化方法是一系列用于描述和分析系统的符号表示法和技术。这些方法被称为*形式化方法*的原因在于它们是以一些数学理论为基础的，例如逻辑、自动机和图论，它们都致力于提高系统的质量。形式化规约技术引入了一种能够准确、无二义地描述系统属性的方法。它对于消除误解十分有效，也可以进一步用于系统调试。形式化分析技术能够用于验证系统是否满足它的规约，或者系统化地发现它不能满足规约的情况。形式化方法能够减少查找设计或编码错误所需要的时间。它还能够有效降低因为未能在系统部署之前发现错误而导致损害的风险。本书中我们将会关注用于软件的形式化方法，我们也称之为*软件可靠性方法*（software reliability method）。

这些技术有时由程序员自己使用，但是通常情况下，使用它们是专门的软件质量保证团队的责任。在后一种情况下，开发人员和那些关注可靠性的人员（例如测试团队）分工合作，以实现更为可靠的软件为目标，一起创建有利的协同工作方式。实际上，在许多情况下，质量保证团队的规模要大于开发团队的规模。

形式化方法研究的早期，研究的焦点是保证系统的*正确性*。这里所说的系统正确性是指系统能够满足客户提供的规约。*演绎软件验证*（deductive software verification）技术是当时主要的形式化方法技术研究之一。这一领域的先驱，例如 Dijkstra、Floyd、Gries、Hoare、Lamport、Manna、Owicki、Pnueli 等，提出了不同的可用于验证程序正确性的证明系统。这类技术的目标是形式化证明一些关键系统，例如导弹、航空控制。有人建议验证方法也能够用于辅助软硬件系统的开发 [94]，根据这一建议，程序开发时从它们的形式化规约开始，按照逐步精化的方式最终得到实际的代码，这个过程中的每个精化步骤都保持了正确性。

人们很快认识到了形式化方法的一些局限性。软件验证方法不能够保证*实际*代码的正确性，而是允许人们对实际代码的一些*抽象模型*进行验证。因此，由于真实系统和其抽象模型可能有所区别，所以对模型的正确性证明可能对实际代码无效。不仅如此，正确性证明本身可能不正确。证明的过程通常只覆盖了系统功能的一小部分。其中的原因在于验证是针对一个给定的系统规约而进行的，而这个规约是手工定义的，有些时候它可能不完整，忽略了系统的一些重要属性。而事实上，评定一个给定的规约是否完整通常是很困难的，有时甚至不可能完成。尽管演绎验证的某些部分可以自动完成，但是一般而言它仍是一种靠手工完成的技术。因此，演绎验证主要通过很小的例子进行展示，而且需要大量的时间和专门的技术。

尽管演绎软件验证有这些局限性，但它还是取得了一些重要的成就，它通过定义*不变式*这个概念影响了软件开发过程。不变式是一个正确性断言，在执行过程中，它必须在某些特定的控制点上成立。它断言了程序中变量之间的某些联系。程序员向程序代码中添加不变式可以增加他们对该程序的运行方式的直观理解。不仅如此，他们还能够插入简单的运行时检查代码，以检查不变式在特定的程序控制点上真的成立。如果某个不变式不成立，程序的正常执行就会被中断，同时给出一个警告信息。

最近，新的并且更高效的工具正不断地开发出来，为演绎软件验证提供更好的支持。这些工具试图将部分证明过程机械化。净室方法 [98，99]（本书的后面会讲述该方法）是一个非形式化地应用软件验证的例子，它与其他软件可靠性方法一起被用于一个软件开发方法论中，致力于提供高质量的软件。

早期的自动化验证技术是由 West 和 Zafiropulo [146，150] 提出的。这些方法可以用于各种*有限状态系统*（finite state system），例如硬件电路和通信协议。美国的 Clarke、Emerson [28，42] 和法国的 Quielle、Sifakis [120] 首先提出了*模型检验*（model checking）的概念。该技术可以验证一个有限状态系统相对于给定的时序规约的正确性。从较小规模的例子（如交通信号灯

控制器）开始，这些方法逐渐成熟并形成了数百万美元的工具产业。

新的自动化和半自动化的验证技术表明了自动验证方法的可扩展性。这些技术包括二元决策图 [24]、偏序约减技术 [55，113，142]、对称约减 [43]、归纳技术 [81，149]、抽象技术等。这些技术证明了验证技术不仅仅局限于理论中，在实践中也一样出色。但是，在使用自动验证时仍然需要考虑到若干限制，这些限制包括处理较复杂的数据结构（如队列和树）时效率低下。此外，自动验证方法通常局限于有限状态系统，因而它们最适用于验证通信协议和某些算法的抽象表示。它们可能无法处理那些带有实数与整数变量或带有指针与数组引用的成熟程序。和演绎验证一样，自动验证技术也需要首先对系统进行建模，而不是直接进行验证。这样可能会造成被验证的模型对象与实际系统的不一致。

软件测试（software testing）也许是使用最为频繁的质量保证方法了。它不会提供对系统的全面检查，软件测试的重点是根据某些覆盖准则对程序运行过程进行采样，并将程序的实际行为与程序规约中规定的期望行为进行比较。测试主要是根据实验证据和经验来完成的，并且常常通过非形式化的方式完成。测试方法的优点在于能够对一个实际的系统，而不是系统的模型，进行检查。和模型检验不同，测试并不局限于有限状态系统。但是测试不是和演绎验证、模型检验一样的全面性技术，它不能覆盖所有可能的执行过程。因而即使是经过彻底测试的代码，其中仍可能包含错误。

一些形式化方法术语

不同形式化方法的研究者和实践者以不同的方式使用验证（verification）这一术语。有些时候，验证仅仅是指获取系统的形式化正确性证明的过程，也就是说狭义上的验证即是演绎验证。在其他语境下，验证是指试图寻找程序中错误的任何活动，包括模型检验和测试。

因为本书中讲述了不同的形式化方法技术，我们有时候不得不为某些术语选择适当的解释，而这些术语在不同团体中的解释可能并不一致。在提到验证时，我们指的是某个手工或自动化技术的应用过程，这些技术可以确定代码是否满足某个属性，或者它的行为是否和某个高层次描述相符。根据这个定义，我们主要指的是演绎验证和模型检验这两类活动，但不是指测试，因为后者更像是抽样检验，而不是一个全面的正确性检查。

我们会区分下面两种不同的活动：验证一个软件是否符合特定的规约属性（property）（有时也称作确认（validation）[70]），以及验证对软件的两个描述之间的一致性（conformance）。在后一种情况下，其中一个描述通常包含更少的细节，即更加抽象。

1.2 开发与学习形式化方法

形式化方法包括规约、验证和测试技术，这些方法被用于提高软件开发和硬件设计的质量。在过去的 20 多年中，人们认识到了各种方法的好处、优势、权衡和局限性。理解形式化方法的局限性和介绍它们的成功案例同样重要。例如，经验表明计算方法比那些需要大量人工技巧的方法更好。从另一方面看，自动化方法的处理范围也有局限性，例如它们常常只能处理中等规模的有限状态系统。作为形式化方法研究成熟的表现之一，人们总是用怀疑的目光看待那些试图保证系统正确性的方法，而倾向于使用那些以寻找错误为目的的方法。

各种形式化方法的工具和支撑系统首先由大学和研究所的研究者倡导。最近几年中，我们已经目睹一些公司开发出他们自己的形式化方法和相应的工具，尤其是那些通信和硬件领域的企业，因为其产品的可靠性是一个关键指标。在其他地方，人们开发了一些接口工具，将软件转换成现有工具可以接受的形式，这些工具的输出可能又被转换回某些规定格式。我们也开始逐渐看到一些现有的工具可以增强软件的可靠性，但是这些工具中的大部分都和软件测试或软件

建模中的不同任务相关。

形式化方法的研究还是相对较新的，选择正确的技术和工具是一件困难的事情，而开发一种新的验证或测试工具则需要相当大的工作量。本书中，我们将概述各种形式化方法之间的异同，同时也指出它们可以怎样组合在一起，弥补各自的不足。

当一个新的软件产品被开发出来时，需要决定是否使用某个特定的软件可靠性工具或相应的技术。在有些情况下，应该尝试着开发一个新的工具或裁剪现有的工具以满足某些特定的需要。软件开发中担负不同角色的人们有着不同的考虑。下面列出了一些相关的问题：

- 项目经理
  - —在我的软件项目中使用形式化方法会获得什么好处？
  - —需要哪些时间、人力和金钱上的投入？
  - —我是不是应该建立一个内部的形式化方法小组，以便开发新的符合我部门需求的工具？
- 质量保证团队主管
  - —我们应该在开发过程的哪个阶段使用形式化方法？
  - —我如何合理安排使用形式化方法的活动和其他的开发活动？
  - —我们需要多少人力来完成这项任务？他们需要什么样的资质？
  - —我们应该如何雇用或训练员工来应用形式化方法？
- 工程师（用户）
  - —哪个工具或技术能够最好地帮助我们获得更高的可靠性？
  - —它的成本是多少？
  - —这个工具能够提供什么样的支持？
  - —当前正在开发系统的最适当的形式化表示是什么？
  - —使用不同的形式化方法找到错误的概率各是多少？
  - —什么样的形式化规约最适合被开发的系统？
  - —为达到我部门的目标，开发一个新方法或改进一个已有方法需要多长时间？
  - —我们是否需要特殊的设备或软件来支持这些工具？
  - —我们怎样使用选定的某个方法或工具对被开发系统（或其一部分）进行建模？
  - —我们应该如何解释形式化方法给出的结果？例如，当工具报告错误时，系统真的有错吗？当没有发现错误时，我们对系统的把握有多大？
  - —当我们使用的工具失败了怎么办？例如当验证工具没有在适当的时间内得出结果或者出现了内存耗尽的情况。
  - —我们的规约正确吗？其中是否包括了一些自相矛盾的情况？它们是不是包括了该系统所有必要的需求？
- 形式化方法研究人员
  - —我怎样才能增加形式化方法技术的*表达能力*（expressiveness），以便可以规约和验证更多的系统需求？
  - —我怎样才能提高这些技术的*效率*（efficiency），使得这些方法和工具能够运行得更快并能够处理规模更大的实例？
  - —是否存在*启发式的方法*（heuristics）能够在很多实际情况下比标准方法更好？
  - —我怎样向潜在的用户推广新技术？
  - —软件开发人员常使用哪些表示方法？我怎样将它们整合到我开发的技术和工具中去？

我们无法在书中回答所有这些问题，因为对其中某些问题的回答强烈依赖于被开发的特定系统、环境和开发组织。但是我们试图通过提供一些现代形式化方法的相关信息、它们的用途和