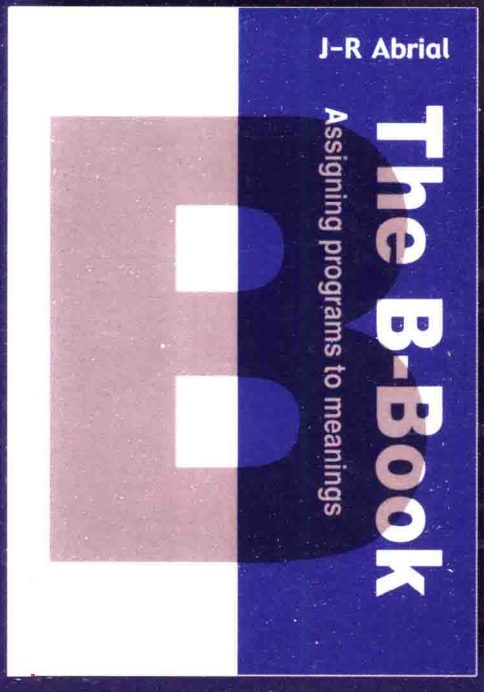
国外计算机科学教材系列

B方法

The B-Book

Assigning Programs to Meanings



[美] J-R Abrial 著
裘宗燕 译

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

国外计算机科学教材系列

B方法

The B-Book

Assigning Programs to Meanings

[美] J-R Abrial 著

裘宗燕 译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书是有关B方法的最重要的著作，由B方法的发明人J-R Abrial撰写。B方法是目前国际上最受重视的实用性软件形式化方法之一，人们用它编写软件系统规范，进行系统设计和编程。B方法已被用在一些极其重要的软件项目中并取得了很大成功。本书由4部分组成，内容涵盖了B方法的所有方面，这些部分分别介绍B方法所用的数学基础，用B方法描述软件系统规范的语言记法，基本程序结构和程序实例，系统模块化、分层设计和精化。本书适用于计算机科学工作者、软件系统开发工作者和计算机专业的学生，可作为高校有关软件形式化方法和软件系统设计课程的教材，或者作为B方法的标准参考手册。

版权贸易合同登记号　图字：01-2003-0367

图书在版编目（CIP）数据

B方法/（美）艾伯瑞尔（Abrial, J-R.）著；裘宗燕译. -北京：电子工业出版社，2004.6
（国外计算机科学教材系列）
书名原文：The B-Book: Assigning Programs to Meanings
ISBN 7-5053-9339-1

Ⅰ. B...　Ⅱ. ①艾...　②裘...　Ⅲ. 软件设计-教材　Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2004）第046567号

责任编辑：周宏敏　　特约编辑：赵宏英
印　　刷：北京智力达印刷有限公司
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 × 1092　1/16　　印张：34.5　字数：883千字
印　　次：2004年6月第1次印刷
定　　价：57.00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。

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳－希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯托林斯（William Stallings）、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

主　任	杨芙清	北京大学教授 中国科学院院士 北京大学信息与工程学部主任 北京大学软件工程研究所所长
委　员	王　珊	中国人民大学信息学院院长、教授
	胡道元	清华大学计算机科学与技术系教授 国际信息处理联合会通信系统中国代表
	钟玉琢	清华大学计算机科学与技术系教授 中国计算机学会多媒体专业委员会主任
	谢希仁	中国人民解放军理工大学教授 全军网络技术研究中心主任、博士生导师
	尤晋元	上海交通大学计算机科学与工程系教授 上海分布计算技术中心主任
	施伯乐	上海国际数据库研究中心主任、复旦大学教授 中国计算机学会常务理事、上海市计算机学会理事长
	邹　鹏	国防科学技术大学计算机学院教授、博士生导师 教育部计算机基础课程教学指导委员会副主任委员
	张昆藏	青岛大学信息工程学院教授

译 者 序

随着计算机应用逐步渗透到社会生产生活的各个领域，我们今天看到了一种非常奇特的现象：一方面，计算机已被看做是现代科学技术开发出的最强有力的工具，是“用之万事而皆灵”的魔杖。计算机的介入改造了所有的行业和部门，不断改变着我们的工作和生活方式，是否应用了计算机已经成了评判一个行业是否现代化的最重要标志。另一方面，促成这种变化的领域本身的情况却很不令人乐观。计算机系统的不可靠也已成为人所共知的事实。在听到“是计算机系统出了问题”时，世人都已经习惯于视如天命而自认倒霉，再也没有进一步去追究责任的兴趣和意愿了。看到计算机系统正在越来越多地控制着我们周围的环境，一些系统的失误可能给人们的生命财产造成巨大损失时，作为计算机工作者难道不该不寒而栗？难道我们的软件开发工作就应该永远停留在这样的水平？系统的质量就应该永远这样没有保证吗？

实际上，人们早已开始从各种角度研究软件的开发方法和途径，研究软件开发的本质和规律性，总的目标就是希望改变目前这种基于个人技术、缺乏科学指导的软件开发方式，为将来的软件开发过程和作为其产品的软件系统提供更坚实的科学基础。软件形式化方法就是这样一个研究领域。经过几十年的努力，软件形式化方法的研究已经取得了许多重要成果，一些成果已经被应用于实际的软件开发工作，取得了很好的效果。

本书作者 J-R Abrial 在软件形式化方法及其应用领域中做出了非常重要的贡献。J-R Abrial 从 20 世纪 70 年代开始研究数据结构和程序的形式化问题，他于 20 世纪 70 年代后期在牛津大学程序设计研究组（PRG）访问期间做了规范语言 Z 的开创性工作。后来 Z 被 PRG 和其他研究组织广泛研究和发展，现已成为许多计算机科学技术课程和教育项目的基础，并在英国及全欧洲的工业界得到应用，用于描述复杂软件系统的规范，支持严格化的软件系统开发过程。为使形式化方法与软件开发更平滑地衔接，J-R Abrial 从 20 世纪 80 年代前期开始了 B 方法的研究，目的是希望为实际软件开发过程提供一个坚实的数学基础。在 B 方法的研究和开发中，J-R Abrial 及其合作者不仅关心软件的严格规范描述问题，而且特别关注从严格规范到最终程序的演化过程（一般称为精化，refinement），以及对这一过程的自动支持。B 方法的开发从其早期开始就是与工业界的实际应用一起进步的，在其发展过程中，人们就已经用它作为工具开发了一系列关键性的应用软件系统。一个早期的重要实例是巴黎地铁列车的信号系统，这一系统为减少列车间距、提高整个地铁系统的安全性做出了显著贡献。这本《B 方法》就是 J-R Abrial 对 B 方法的总结整理。

B 方法是一种用于描述、设计计算机软件的严格方法，其作用一直延伸到代码生成，人们已经实现了一些工具系统，支持基于 B 方法的软件开发的全过程。本书完整地介绍了 B 方法的数学理论基础，精确定义了 B 方法中使用的规范语言形式，还给出了许多运用 B 方法进行软件开发的例子。其中一些例子有一定规模和很强的实际背景，有些例子就是实际系统的简化版本。从这本书的阅读中，我们可以看到一个复杂的软件系统可以怎样从一段有关其功能的严格描述中逐步演化出来，如何在这种演化过程中维持某些不变性，怎样保证后面做出的更加细节的描述不破坏已有的定义和已经证明的性质，如何自动地产生出这一过程中的“证明义务”。实际上，在今

天的软件开发过程中，人们也在不自觉地做着这些事情，基本上是靠开发者的直觉和朴素技术去控制和解决问题。这种缺乏科学指导的不严格的方式，是软件开发过程中产生错误的一个重要原因。因此，从某种角度看，B 方法（以及其他严格的软件开发方法）也就是希望把人们在软件开发中的实际工作过程整理清楚，为其建立坚实的理论基础，将其规范化和严格化。

我们常常可以听到有关“软件形式化方法有什么用”的质问，我们认为这实际上是一个错误的问题。任何不幼稚的计算机工作者都不会期望能开发出某种新技术或者新方法，一劳永逸地解决软件领域的问题。计算机软件系统是人类有史以来制造出的最复杂的产品，可具有成百万甚至成千万行代码的巨大规模，其各组成部分之间可能存在的错综复杂的直接和间接交互作用方式，其静态结构与动态行为之间构成难以琢磨的关系。我们还没有过制造和把握如此复杂的系统的需要和经验。另一方面，我们需要开发出许许多多的软件，以满足千变万化、层出不穷的实际需要，这些工作又常常必须在较低开发成本和较短开发周期的条件下完成，因此就不可能像开发基础硬件那样组织一支最优秀的庞大队伍，不可能投入足够的金钱和时间。所有这些因素造成的结果就是软件的复杂性问题将永远陪伴着我们。

由于这些情况，任何有助于我们理解、控制、管理软件系统复杂性的理论、技术或方法都是非常有价值的。今天，软件工作者们最重要的努力方向就是去克服或缓解软件复杂性的障碍。他们在许多层次上工作：通过高层管理手段组织软件开发活动；通过严格的工作规范，企图深入控制软件开发的整个过程；通过安排丰富而烦琐的人际信息交互活动，设法使在需求、设计和各个后续阶段引进的错误和不良结构可能及早被发现；通过不同抽象技术，设法隔离各个层次、各个部件内部的复杂性，提供层次间或者部件间的清晰界面；通过研究各种程序开发模型及其支持技术，使软件工作中的一些模式和难点能够被屏蔽在模型之中，得到模型的充分支持；通过各种软件的组件形式尽可能地利用已有工作成果，降低开发新软件的代价和复杂性；通过开发合适的语言和其他表述形式，以满足在各个层次上严格而方便地描述软件及其不同侧面的需要；通过各种技术手段和工具，对软件开发过程提供尽可能多的支持，或者支持人们对开发结果进行更深入的分析；通过人员的训练，提高参与软件工作的每个人的认识能力和工作能力，使他们能够成为软件质量的支点而不是潜在的破坏者；如此等等。软件形式化方法可能在许多层次上起作用。

实际上，软件形式化方法正在逐步渗入我们的日常软件开发活动。各种所谓的“无差错”或者“干净”软件开发技术，貌似新颖的“基于契约的软件开发”等，实际上不过是软件形式化方法研究成果的实践性版本。“断言”、“前条件和后条件”、“不变式”等术语已经在我们周围不绝于耳。一些国家和部门的软件开发规范已经明确提出，关键性软件的开发过程必须有软件形式化方法的参与。我们相信，终归会有一天，不知道什么是循环不变式或者数据不变式的人，将没有资格参加任何关键软件的开发工作。到了那个时候，软件的质量一定会提高到一个新水平。每个关心自己的事业和个人发展的计算机专业工作者，都必须重视这些情况。我们也希望本书的出版将能为国内计算机教育和实践提供一些参考。

作为译者，我非常感谢电子工业出版社支持出版这本形式化方法的重要著作，感谢周宏敏编辑为本书付出的辛勤劳动。最后还要感谢夏萍和丛欣对我持之以恒的支持和帮助。

裘宗燕
北京大学数学学院信息科学系
2004 年 5 月

序　言

本书远远强于一本新的程序设计手册，它介绍了一种方法，将程序的设计包容在一个全局性的进程之中，这一进程从对问题的理解开始，一直延续到验证其解答的合法性。

这一方法的数学基础提供了一种精确性，所提出的记法形式完全排除了日常语言的歧义。与此同时，这一工作过程对于产业界的使用而言又是足够简单的。事实上，“产业界的”是一个非常关键的词语。

形式化方法的一般性目标是为问题的规范描述提供正确性基础。在这里，我们可以看到怎样能找到一个解，而且是一步一步地通过一个受到连续监控的进程。对其中每一步之合法性的数学验证与精化活动如此紧密地联系在一起，使人根本无法将其中的设计决策与检验过程相互分离。这使得人们的想像力得到了精确性的极大帮助！

然而，效率又怎么样呢？设计过程会不会太长？做设计的人们能胜任这种工作吗？机器的能力能否实现这种方法？对于这些问题的回答都很容易给出。让我来告诉你们。

我的公司从 20 世纪 60 年代开始涉足于铁路控制系统的实现，这类工作必须满足极高的安全性要求。在开始使用编程逻辑不久(20 世纪 70 年代末)，我们就不得不设法去解决软件的正确性问题。与其他方法一起，我们选择使用了由 C. A. R. Hoare 提出的程序证明方法。1986 年 J-R Abrial 向我们介绍了 B 方法，我们决定学习和使用它。当时还没有相应的工具，我们对于工具开发的贡献就是用自己的应用提供了一个真实世界的测试实例，并提出了一些改进建议。现在有关的工具已经能够从市场上找到，因此这个方法也能够以其最有效的方式使用了。那么，我们又学到了些什么呢？

- 首先，这一方法的原理相当容易理解，用不了一年的时间就可以熟练掌握。
- 进而，这一方法鼓励重用性，并能使重用更加方便。重用的基础是不断增长的已经证明其正确性的库。
- 由测试和合法性检验阶段节约的时间非常明显，结果从总体经济上衡量是非常有利的。
- 产生出的程序是高效的，虽然其结构的组织方式采用了更多的抽象层次。
- 有关工具可以在简单的工作站上使用。

对于将软件用于与安全有关的应用领域而言，对于这一方法的使用已经成为我们增强自己信心的决定性因素。进一步说，新的国际标准也建议在安全攸关软件的规范描述和设计方面使用形式化方法。

感谢 J-R Abrial 使我们有了一种构造正确软件的产业化方法。我们相信本书将使读者确信，采用这一方法能使他们节约金钱。

Pierre Chapront
技术主管
GEC-ALSTHOM Transport

前　言

本书是一个长篇论述，按照我的见解，它要解释的是程序设计的工作（无论是大是小）将可能怎样通过回归数学而取得成功。

对于这一说法，我的第一层意思是，有关一个程序要做什么的精确数学定义必须首先给出，并将其作为其构造的本原。如果缺乏这种定义，或者它过于复杂，我们或许会怀疑将来做出的程序能否真正表示什么东西。我的信念是，抽象地说，一个程序根本就不表示任何东西。要说一个程序表示了些什么，只能是相对于某个在它之前就有的意图，无论这一意图是以这种或者那种形式表示的。在这一点上我并不反对某些人，他们可能感到用“英语”这个词代替“数学”就更舒服一些。我只是怀疑，能不能交给这样的人一件更困难的工作。

我还认为，这种“回归数学”应该表现在一个真正的程序构造过程中，其中的工作就是要给程序一种定义良好的意义。这里的想法是让程序构造的技术过程伴随着一个证明构造的类似过程，由它保证所提供的程序与其预期的意义相符。

同时关注一个程序的体系结构及其证明其实是非常有效的。例如，当证明变得非常难搞时，多半程序也会是那样；而构造证明的各种成分（抽象、实例化、分解）也与构造程序中的那些东西很类似。理想地说，在一个程序的结构与其正确性的证明之间的关系应该非常紧密，以致我们根本无法查清两者中究竟哪一个是从另一个派生出来的。而后我们就可以很合理地说，构造一个程序不过就是构造一个证明。

今天只有很少的程序是按照这种方式描述和构造出来的。那么，这是不是正好对应着另一个事实，即今天有那么多的程序都是极其脆弱的呢？

J-R Abrial

致　谢

本书的撰写差不多延续了 15 年。在此期间我遇到了许多人，他们中的一些确实对本书中所描述的工作有着正面的影响。我愿意感谢所有这些人们。

很清楚，这一影响的主要根源应该是 C.A.R. Hoare 和 E.W. Dijkstra 所传播的思想，如果没有这些思想，这本书就不可能变为现实。把程序看做一个数学对象的观点，前条件和后条件、非确定性、最弱前条件的概念，所有这些思想都是本书所讨论的问题的中心。

B 方法是一种“基于模型”的软件构造方法。与 VDM 和 Z 方法相近。显然，这两种方法的许多思想可以在 B 方法中看到。对于 Z 而言这是很容易理解的，因为我也是它的发明人之一，那是在我 1979 年到 1981 年期间访问牛津大学的程序设计研究组之前和那段时期中。对于 VDM 也很好理解，因为在这段时间，我与 C.B. Jones 共用一个办公室。我从他那里学到了有关程序开发的思想、精化及其实际应用的概念，这些都表现在证明责任的形式之下。

与 C.C. Morgan 有关规范和精化的讨论对于本书中的材料产生了深刻影响。他有关扩大程序的概念,使规范可以居于其中的思想,从而催生了这里所讨论的工作。

程序设计研究组在 20 世纪 80 年代所完成的有关精化概念的一系列工作被直接借来,用在我有关精化的讨论中。就我的了解,参与这些工作的人有 P. Gardiner, J. He, C.A.R. Hoare, C.C. Morgan, K.A. Robinson 和 J.W. Sanders。

在这一方法的实践性加工中,也有许多人对这一工作做出过重要贡献,这些人是 G. Laffitte, F. Mejia, I. McNeal, P. Behm, J.-M. Meynadier 和 L. Dufour。这里向他们表示衷心感谢。

G. Laffitte 对这一工作的影响包括他认真的审查,一语破的的批评意见,以及有时对本书中某些数学表述所做的深入的重新整理的建议。

F. Mejia 在构造大型软件结构方面提出了一些重要建议。他与 B. Dehbonei 一起为 B 开发了一个完整的工具集,现在商品化为 Atelier B。

I. McNeal 对于这一方法的早期开发做出了重要贡献,并对证明的机械化产生了一些有益的影响。

P. Behm, J.-M. Meynadier 和 L. Dufour 提出了一些非常有意思的建议,构造了一个原型性的证明器,其基本机制非常有用。

DIGILOG 的强大队伍做了 Atelier B 的产业化和商品化工作,用它开发软件系统,也值得特别祝贺。他们的能力、热情与和善使与他们一起工作成为真正的乐事。我愿意特别感谢 F. Badeau, F. Bustany, E. Buvat, P. Lartigue, J.-Ph. Pitzalis, C. Roques, D. Sabatier, T. Servat, C. Tognetty 和 C. Zagoury。

另外一些人间接地参与了 B 项目,他们中有些人审阅了本书,有些人教授了这一方法或者应用、推广了该方法。我要感谢他们中的每一个人,特别是那些未留姓名的审阅者,还有 P. Bieber, P. Chartier, J.-Y. Chauvet, C. Da Silva, T. Denvir, P. Desforges, R. Docherty, M. Ducassé, M. Elkoursi, Ph. Facon, H. Habrias, N. Lopez, I. Mackie, L. Mussat, P. Ozello, J.-P. Rubaux, P. Ryan, S. Schuman, M. Simonot 和 H. Waeselynk。

与 B. Meyer 和 M. Sintzoff 的偶遇和讨论对这一工作也有间接影响。与他们的会面总像一种智力美餐,可惜的是这种机会出现得太少了。

在产业界,一些机构也以这种或者那种方式帮助了本书的撰写。我特别要感谢 ADI、BP、DIGILOG/STERIA 集团、DIGITAL、GEC-ALSTHOM Transport、GIXI、INRETS、INSEE、MATRA Transport、RATP 和 SNCF。在这一项目的多年开发中,这些机构以各种方式对其提供了支持。我特别要感谢下面的人们:P. Barrier, P. Beaudelaire, J. Betteridge, P. Chapront, A. Gazet, A. Guillon, C. Hennebert, J.-L. Lapeyre, J.-C. Rault 和 O. Sebilleau。

本书的出版是一个漫长而且有时令人痛苦的过程,特别是在它的最后阶段,有些不寻常的困难显露出来了。Bertrand Meyer, Cliff Jones 和 Tony Hoare 在试着克服这些困难的过程中扮演了重要角色,感谢他们的热情帮助。

最后,我还要将深深的谢意献给牛津大学出版社的 David Tranah,特别感谢他使本书的出版成为可能,以及从不妨碍这一科技工作得以在其中进行的那种独立性。

什么是B

B是一种对软件系统进行描述、设计和编码的方法。

覆盖范围

从本质上看，B方法处理的是软件生存周期的核心方面，即，技术性的规范化，通过一系列精化步骤进行设计，产生层次性体系结构，以及可执行代码的生成。

证明

前面提出的每个事项都被看成是一项涉及到写出数学证明的活动，以便为其结果的合法性提供证据。准确地说，这样的一批证明使人能确信有关的软件系统确实是正确的。

抽象机

这一方法的基本机制是抽象机。这是一个概念，非常接近人们在程序设计中所熟知的一些概念，如模块、类或者抽象数据类型。

数据和操作

通过这一方法所表达的软件系统由一些抽象机组成，每一机器包含一些数据，提供一些操作。这些数据不能直接触及，而总是通过有关机器的操作去使用。它们被封装在机器里。

数据的规范化

刻画一个抽象机中数据的方式是使用一些数学概念，例如集合、关系、函数、序列和树。这些数据必须遵守一集通过特定条件定义的静态法则，这些条件称为不变式。

操作的规范化

抽象机中一个操作的规范用一段不能执行的伪代码表述，其中不包含任何顺序性或者循环。在一段伪代码里，每个操作描述为一个前条件和一个原子动作。前条件表述一种必不可少的条件，在不满足它的情况下操作就不能执行。原子动作通过一种广义替换的方式进行形式化。这种广义替换中的非确定性为在随后精化步骤中的进一步决策留下了空间。有关伪代码的形式定义使人可以证明一个抽象机所提供的操作总能维持不变式。

趋向实现的精化

抽象机的初始模型(它的规范)可能被精化为一个可执行模块(它的实现)。从规范到代码的这一进展历程完全在本方法的控制之下。其间必须提供某些证明，这些证明的目的就是说明机器的最后代码确实满足其初始规范。

用精化作为规范化的技术

除了前面提出的各种(经典)作用外，精化还有另一项非常实际的用途。可以通过它将问题的更多细节纳入到形式开发之中。对于初始问题陈述的这种形式变换是逐步进行的，而不一定是一蹴而就的。

精化技术

精化以三种不同的方式进行：删除伪代码中不可执行的语句(前条件和选择)，引进程序设计的典型控制结构(顺序性和循环)，以及将数学的数据结构(集合、关系、函数、序列和树等)变换到可能编程的其他结构(简单变量、数组或者文件等)。

精化步骤

为了仔细控制上述变换，一个抽象机的精化将一步步地进行。在其中的每一步，初始抽象机都要整体性地重新构造。当然，从用户的角度看，虽然相应的伪代码已经被改变了，但它仍然继续提供同样的操作。在中间的精化步骤里，我们将拥有一种混合型的结构，它已经不再是一个数学模型，但也还不是一个程序模块。

层次性体系结构

经验说明，只经过不多的几个精化步骤可能更合适一些。随着层次的增加，复杂性会变得过高，这时就应该建议将一个精化分解为若干更小的片段。这样，一个抽象机的最后精化将通过另外的一个或者几个抽象机的规范实现，而那些规范本身又能进一步精化。完成这一实现的方式就是调用有关抽象机所提供的各种操作。正如你将看到的那样，一个抽象机的“用户”总是另外一些抽象机的最终精化。按照这种方式，我们的软件系统(或者其变换中的非形式规范)的层次结构将被一块块地构造出来。

库

作为某个给定抽象机的实现的那些抽象机，完全可能在所做的精化之前就已经存在了。事实上，与此方法同时存在的是还提供了一些预定义的抽象机，它们组成了一个抽象机的库，其作用就是封装起一些最典型的数据结构。

重用

对于一个确定的项目，建议去扩充这个库，以形成一种基础，基于它们去实现未来的更高层次的抽象机。正如你可以看到的，这一方法允许人们在采用纯粹自上而下的设计方式和采用自下而上的设计方式间进行选择，另外还可以有更好的方式，即采用某种混合方式将规范的重用和代码的重用集成在一起。

代码生成

一个抽象机的最终精化可能很容易地变换到一种或者几种程序设计语言中。在这样做时，本方法也为将应用从一种语言移植到另一种语言提供了一个解决方案。

B 用户组织

有一个讨论和交换有关 B 的信息的用户组织，称为 BUG。它的邮件地址为 bug.@estasl.inrets.fr。本书的邮件列表为 bbook.@estasl.inrets.fr。

本书概览

本书是 B 方法及其有关记法形式的标准参考书。

本书包括了作为这一方法的基石的数学基础，以及所用记法形式的精确定义。它也包含了大量实例，展示了这一方法在实践中的使用方式。本书包括4个部分和若干附录。

第一部分	数学
第二部分	抽象机
第三部分	程序设计
第四部分	精化

第一部分

第一部分包括了对谓词逻辑和集合论的系统构造，还包含了对于形式化软件系统所需要的一些数学结构的定义。这里特别强调的是证明的概念。第一部分由以下几章组成：

第1章	数学推理
第2章	集合记法
第3章	一些数学对象

第二部分

第二部分包含对广义代换语言(GSL, Generalized Substitution Language)和抽象机记法(AMN, Abstract Machine Notation)的介绍。这些就是我们用于刻画软件系统的记法形式。在介绍它们的同时还给出了一些实例，说明如何系统化地构造大型规范。这里还给出了GSL和AMN的集合论基础。第二部分由以下几章组成：

第4章	抽象机引论
第5章	抽象机的形式定义
第6章	抽象机的理论
第7章	构造大型抽象机
第8章	抽象机的实例

第三部分

第三部分引进了两个最基本的程序设计概念：顺序和循环。在展示有关的理论之后有很重要的专门一章，其中研究了一些算法开发实例的系统化构造。第三部分由以下几章组成：

第9章	顺序和循环
第10章	程序设计实例

第四部分

第四部分给出了广义代换和抽象机的精化概念。精化的数学基础用集合论的方式给出。

这里还解释了通过模块层次构造的方式构造大型软件系统的方法。最后研究了若干完整开发的大型实例，其中特别强调了方法论问题。第四部分包括下面各章：

第 11 章	精化
第 12 章	构造大型软件系统
第 13 章	精化的实例

附录

附录中综述了所有逻辑和数学定义，还包括对规则和证明义务的总结：

附录 A	记法综述
附录 B	语法
附录 C	定义
附录 D	可见性规则
附录 E	规则和公理
附录 F	证明义务

怎样阅读本书

许多关心不同问题的人都可能以适当的方式阅读本书。

例如，你可能希望学习这一方法，以便成为一个形式化方法的实践者。在这种情况下，你对本书中更细节的数学问题可能就不那么有兴趣（当然，也可能你确实有兴趣）。此时建议你阅读本书的如下部分：

附录 A	记法综述
第 2 章	集合记法(2.7 节)
第 4 章	抽象机引论
第 7 章	构造大型抽象机(7.2 和 7.3 节)
第 8 章	抽象机的实例
第 11 章	精化(11.1.1、11.2.1、11.2.5、11.2.7 和 11.2.8 节)
第 12 章	构造大型软件系统(12.1 和 12.2 节)
第 13 章	精化的实例

作为这个连续谱系的另一个极端是：你是一个计算机科学家，可能更关心这一方法的数学基础。这样你就可以按下面的顺序阅读本书：

附录 A	记法综述
第 1 章	数学推理
第 2 章	集合记法
第 3 章	一些数学对象
第 6 章	抽象机的理论
第 9 章	顺序和循环
第 11 章	精化
附录 C	定义
附录 E	规则和公理

还有一些人想看看本方法怎样用于构造大型的规范和大型的设计。此时建议读者阅读如下的章节：

附录 A	记法综述
第 4 章	抽象机引论
第 6 章	抽象机的理论
第 7 章	构造大型抽象机
第 11 章	精化
第 12 章	构造大型软件系统
第 13 章	精化的实例

关心以系统的方式开发小型程序的人们可以阅读本书的以下部分：

附录 A	记法综述
第 4 章	抽象机引论
第 10 章	程序设计实例

对于记法的形式化细节感兴趣的人们可以阅读本书的如下部分：

第 5 章	抽象机的形式定义
第 7 章	构造大型抽象机(7.4 节)
第 11 章	精化(11.3 节)
第 12 章	构造大型软件系统(12.6 节)
附录 D	可见性规则
附录 F	证明契约

目　录

第一部分　数　学