软件工程丛书

Ada 软件工程

〔美〕 Grady Booch 著

麦中凡 梁南元 译

周伯生 校

科学普及出版社

内 容 提 要

Ada语言是美国国防部耗资五亿美元，历时八年研制的全军统一程序设计语言。由于体现了软件工程的成功经验和最新成就，它被誉为第三代计算机语言的顶峰，第四代计算机语言的成功代表。美国空军研究院的格兰地·布奇教授在美国国防部曾为2500名各类人员讲授过本书。他以优美的语言深入浅出地介绍了Ada；自然地描绘了软件工程的一些基本原理；给出了有良好设计风格的实例；介绍了能够充分利用Ada语言功能的、可以协助人们管理大型软件开发工作的、面向对象的设计方法学。

Software Engineering with Ada

Grady Booch

United States Air Force Academy

The Benjamin/Cummings Publishing Company, Inc.

软件工程丛书

Ada 软 件 工 程

麦 中 凡
梁 南 元 译

周 伯 生 校

科学普及出版社出版(北京海淀区魏公村白石桥路32号)

新华书店北京发行所发行 各地新华书店经售

北京市密云县印刷厂印刷

开本：787×1092毫米1/16 印张：25 字数：450千字

1986年2月第一版 1986年2月第一次印刷

印数：1—7,000册 定价：5.40元

统一书号：15051·1190 本社书号：1250

序

Ada不只是又一种程序设计语言，应当把学习用Ada编程序看作是一个令人激动的机会。这是因为它不仅基于程序设计的现代观点，而且也以软件工程的最新理论为基础。Ada是一个程序设计系统，它是新的软件文化的发祥地。

计算机对社会的影响堪与工业革命的重要性相比。它们已经在政府和工业的管理中，并因而对其它各个部门都产生了显著的效果。在本世纪八十年代，计算机技术正在改变着现代社会的结构。将有巨大计算功能的线路制作在一个芯片中的能力，已经导致计算机进入消费用品，例如汽车、微波炉、洗衣机、手表以及家庭娱乐系统。我们正在看到不仅为业余爱好者、而且也为非专业人员所喜爱的个人计算机广泛用来管理帐务，控制加热和空调系统，甚至管理家庭的各项杂务。

所有的计算机都需要编程。计算机的大量增加进一步加剧了对已经快速增长的软件需求。软件将是本世纪社会中最重要的技术之一。我们不能仅仅编写程序，我们必须懂得设计和管理软件系统。

六十年代末期和整个七十年代，一些计算机专家倡导了软件工程的基本原理。他们开始宣传精心设计、结构化编程和软件生存期的管理等准则。新的技术、语言特征和软件工具被用来研究改进软件质量，提高涉及整个软件开发过程的生产率。

一九七五年初，美国国防部一些具有远见卓识的人士意识到了迫在眉睫的软件挑战，认为有效的软件开发系统必须以一种标准程序设计语言为基础，这种语言应当具有支持现代程序设计方法的一切特色。于是他们定义了需求，并在了解到现有的语言都不能支持这些需求之后，发起组织了最后导致产生Ada的世界范围的语言设计竞赛。

在Ada的整个开发过程中，它的设计受到世界众多的第一流计算机科学家和软件工程师的关注及帮助，并且得益匪浅。为创造一种能够支持更复杂软件开发的语言所吸引，许多人不计报酬地、热情地提供了他们的忠告和经验。甚至批评性的评论对Ada的设计也有所裨益。

设计的结果产生了一种非常成功的语言，它引起了计算机界的广泛兴趣，这种语言已被称为“八十年代的语言”。虽然如此，它既不是完备的，也不是程序设计语言的终极。它是一项工程的成果，其中不可避免地存在一些折衷与妥协。不过始终没有放弃软件工程的原理。以软件移植和再使用为设计目标，Ada保证支持软件工业的发展。Ada软件程序包的封闭性设施及数据抽象能力，将有利于开发应用程序库和其它形式的可重复使用软件。

本书更多的从Ada的软件工程角度进行论述。它提倡用一致的设计方法以及恰当的风格开发软件。作者用优美的语言描述了Ada。他曾经在美国国防部的许多短训班上讲授过Ada。经过广泛的实验，他知道怎样才能最好地介绍这些内容。本书不仅是这种语言、而且是如何正确使用这种语言经精心斟酌后的指南。

如果你学习Ada只简单研究其语法，你将感到极大的失望。Ada是复杂的，你很可能设计出结构贫乏的程序。然而如果你学习Ada时注意把它作为得到现代软件工程知识的媒介，就会高兴地发现它比它呈现在人们面前的规模要简单得多，你也会发挥出自己清晰地表达设

计的能力。本书努力以这种方式引导读者，给出导致这个语言的特征的原因，并说明它们的正确使用。

Larry E. Druffel于弗吉尼亚州，阿灵顿

1983年1月

前 言

Ada是一种表达能力很强的通用程序设计语言，它是由美国国防部为克服软件开发危机发起研制的。虽然Ada是针对大型、实时、嵌入式计算机系统设计的，但是它也将影响到其它应用领域。

Ada与FORTRAN、COBOL、甚至Pascal等大多数高级语言不同，它不仅体现了许多现代软件开发原理，而且将这些原理付诸实现。使用好的软件开发方法学，以Ada为表达语言将发挥这种通用高级语言的最大能力。因此，使用Ada语言在改善软件系统的清晰性、可靠性、有效性以及可维护性方面提供了巨大的可能。

Ada不仅是又一种程序设计语言，它和Ada支持环境一起就成为非常强有力的工具。它帮助人们认识问题，并以直接映射多维客观事物的方式表达问题的解。

目 的

本书不仅是又一本介绍Ada的新书，它是为满足以下三个特定目的而著的：

- 它是对Ada性能的深入研究。
- 促进好的Ada设计和程序设计风格，并给出具体实例。
- 介绍一种面向对象的设计方法学，它充分利用Ada的功能，并且帮助人们管理大型软件开发工作的复杂性。

简言之，本书不仅介绍Ada程序设计的细节，而且提出在开发软件系统时最好地应用这个语言的特色的方法。

在Ada的开发过程中，1980年7月和1980年11月公布的两个Ada语言参考手册版本，给出了Ada的最初定义。根据美国国家标准协会（ANSI）经仔细研究后提出的建议，其语言设计小组于1981年末对它作了许多精细的修改。我随之修改了教程以适应这些变化。故本书现在反映了ANSI标准的Ada。此外，为确保正确性，我将本书绝大多数设计例题和程序段提交Ada编译程序测试过。

本书是我代表美国空军研究院以及Ada工程联合办公室（AJPO—Ada Joint Program Office）对遍及全国的2500多名学生讲授Ada的经验总结。我对许多不同水平的小组教授过此语言的技术细节，这些小组包括非程序员班、大学生班、研究生班、专业程序员班以及软件管理人员班。正是由于这些经历，我试验了描述语言性能的各种不同的方法，研究了它们的成功与失败，也听取了实际软件开发人员的具体要求。

所以，本书作为一本完整的Ada语言参考书，对希望设计和实现Ada系统的程序员，以及对需要了解如何应用这个强有力工具的管理人员都是适宜的。考虑到这两种人员有时会有完全不同的要求，我把管理人员感兴趣的章节标以〔M〕符号，程序员感兴趣的章节标以〔P〕符号。本书假定读者对程序设计的基本原理是了解的。

内 容 特 点

结 构

很多教程仅从语法和语义的角度讲述程序设计语言的细节。本书作了一些改变，从软件设计方法开始，然后以好的程序设计方法学的形式，自顶向下的介绍Ada。这种方法是计算机协会程序设计语言小组（ACM SIGPLAN）的教育专业组于1980年12月波士顿Ada讨论会上推荐的，它也是AJPO关于Ada教育和培训方针所奉行的宗旨。

本书分为八篇，每篇有逻辑上相关的三章。第 1 篇从分析Ada的问题域开始，它包括对Ada开发史的考察，以便纵观本语言的某些特征。第 2 篇考察了许多现代软件开发原理，并介绍了面向对象的设计方法学。

第 3 篇到第 7 篇围绕着五个完整的设计实例，详细地叙述了体现上述方法学的Ada语言。实例的复杂性是逐步增加的，它们合在一起几乎用到了Ada的每一种特征。此外，在强调可理解性这种程序设计风格的同时，这些例题起到了示范面向对象的设计方法学的作用。在五个大例题之间的各个章节详细地讨论了Ada的结构。本书以第 8 篇结束，这一篇考察了Ada 程序设计支持环境，以及Ada在整个软件生存期中的应用。

资 料

在多数章节末尾都为读者准备了一组练习题。此外书末还有七个附录，它们提供了关于Ada进一步的技术细节。前两个附录是语法图和风格指南摘要，下三个是语言的预定义元素，最后两个附录一个是各设计实例的解答，一个是习题选解。难题都标以星号（★）。本书还附有Ada术语汇编和广泛的补充读物目录。

这是一本“通用”的教材，它可以用于不同的级别。我曾在一学期（40学时）课程中讲授过此教材，也曾把它作为五天讨论班的教材。大纲如下：

第 1 课	第 1 章	**绪 论**
第 2 课	第 2 章	**软件危机**
第 3 — 4 课	第 3 章	**Ada开发史**
第 5 — 8 课	第 4 章	**软件开发方法学**
第 9 —10课	第 5 章	**面向对象的设计**
第11—12课	第 6 章	**Ada语言概述**
第 13 课	第 7 章	**第一个设计问题：二叉树树叶计数**
第14—15课	第 8 章	**数据抽象和Ada的类型**
第 16 课	第 9 章	**第二个设计问题：数据库查询**
第17—18课	第10章	**子程序**
第19—20课	第11章	**表达式和语句**

第 21 课 第12章 **第二个设计问题：续**
第22—23课 第13章 **程序包**
第24—25课 第14章 **类属程序单元**
第 26 课 第15章 **第三个设计问题：集合的类属程序包**
第27—28课 第16章 **任 务**
第29—30课 第17章 **异常处理及低级特征**
第 31 课 第18章 **第四个设计问题：过程控制**
第 32 课 第19章 **输入／输出**
第33—34课 第20章 **大型程序设计**
第 35 课 第21章 **第五个设计问题：座舱显示器**
第36—37课 第22章 **Ada程序设计支持环境**
第38—39课 第23章 **Ada软件生存期**
第 40 课 第24章 **趋向与结论**

此外，下述日程安排适用于向应用Ada的程序管理人员作简单介绍：

第 1 单元 第 2 章 **软件危机**
第 3 章 **Ada开发史**
第 2 单元 第 4 章 **软件开发方法学**
第23章 **Ada软件生存期**
第 3 单元 第 6 章 **Ada语言概述**
第 4 单元 第21章 **第五个设计问题**
第22章 **Ada程序设计支持环境**
第24章 **趋向与结论**

致 谢

感谢在本书手稿准备过程中帮助过我的所有人。特别是Dick Bolz和Larry Schwartz，他们二位在成书过程中校阅了本书并提出了许多有益的建议我。的朋友和空军研究院的同学、智能机器公司的Mike Devlin和Paul Levy，他们付出了许多时间和我讨论Ada的技术和管理等问题；Mike还参与了我的第一个Ada教程的撰稿。

Ada工程联合办公室的Larry Druffel和Vance Mall提供了一个环境，在那里我得以考察Ada的教育论点，而且也得到向全国讲授Ada的机会。这些机会是成稿过程的一部分，对此我深表感谢。

其他许多人提出的评论也影响到本书的形成，尤其是Russ Abbott, Lucie Bennett, Ken Bowles, Bill Carlson, John Goodenough, Richard Kaufmann, Hal Hart, Bob Mathis, Nico Lomuto, Mark Sadler, Tim Standish, Keith Schillington, Peter Wegner, 以及Bill Whitaker加上本序言附表中提到的各位评阅人。此外我从参加我的Ada课程的学生中得到许多非常有益的批评，使我对这些材料的组织和叙述作了进一步的精炼。

还要感谢本书编辑Susan Newman，她对我不断地给予支持和帮助。我尤其要感谢我的妻子在著书过程中对我耐心的照顾和鼓励。

评阅人

加利福尼亚大学和航空公司
Russel Abbott
数据控制公司
Cheryl Allen
软件技术公司
Christine Ausnett
美国空军研究院
Richard Bolz
专用仪表公司
Ben Brosgol

电讯软件公司
Kenneth Bowles
Ada工程联合办公室
Larry Druffel
德克萨斯设备公司
John Foreman
软件技术公司
John Goodenough
国际商用机器公司
Larry Schwartz

目 录

第1篇　问题的域

只要不存在计算机，就谈不上有什么程序设计的问题；当我们有简单的计算机时，程序设计成为一个轻度的问题；现在我们有了庞大的计算机，程序设计变成同样庞大的问题。在这个意义上，电子工业没有解决什么问题，它只产生问题——它已经产生了由于使用它的产品而造成的问题。

E. W. Dijkstra
图灵奖颁发仪式上的演说
1972〔1〕

第1章　绪　论

本质上，我们是工具的制造者和使用者。假若考察不同的人类进化过程，我们可以经常指出一些革命，它们是由使用新的和更有力的社会和科学工具导致的。大家很清楚，书面文字的使用以及随后电视播送的推广，改变了人类文明生活的结构。医用工具（例如显微镜和爱克斯光机）已经拯救了许许多多的生命，艺术工具（包括吉他和画笔）更加丰富了我们的生活。在各种情况下，我们都创造或改善一个工具以满足某种特殊的需要。工具使得我们的一些行动更加有效，而且常常能使我们去做那些曾经超过我们能力的事情。

与其它研究领域比较，计算机科学是一门非常年轻的学科。它也有它的革命，这些革命大致与我们称为计算机硬件的代相对应。每次革命都是由于计算机设计师的工具所引起的。这些工具包括真空管、晶体管以及现在的集成电路。不过如同Dijkstra在他的图灵奖演讲中所说的：硬件工具的能力已经远远超过我们管理它们的能力。

计算机使得做一些事情更有效，它开创了以前不可能解决的应用领域。与此同时，我们开发了如程序设计语言那样的软件工具，以帮助我们解决问题和控制机器。但是大多数工具仍然无助于控制问题解的复杂性。这样，软件开发不再是节省人力、而是非常耗费人力的行动。我们一般称之为**软件危机**。

〔P〕　　**1.1 问题的域**　　〔M〕

如同我们在下一章将详细讨论的，软件危机的征兆“以不反映用户的需要、不可靠、过分昂贵、不准时、不灵活、难于维护以及不可重复使用的软件形式出现”〔1〕。在过去几十年里，出现过不少已经失败的或仍在勉强开发的软件课题的例子。仅在现在，我们才开始了解大型软件系统的复杂性和如何管理它们的开发。在过去很长一段时间里，我们一直依赖于我们的软件“魔术”解决我们的问题。

为了解决危机的基本问题，我们现在就必须采用适当的软件设计方法学，用有规则的方法进行软件开发。不过，仅仅一个方法学是不足以有效地克服软件危机的，我们还必须用适当的程序设计语言作工具来表示和执行我们的设计。

最常使用的语言FORTRAN和COBOL是在计算机科学历史的早期产生的，当时对大型系统开发的问题还不很了解，所以这样的语言并不反映现代的设计方法学，我们不得不用预处理器、扩充和管理控制来充实它们，以适应现代的方法。在某种意义上，这些语言限制了我们对问题的认识，它们基本上是顺序的和指令式的，我们把这种方式称之为 *Von Neumann* **思维方式**。

此外，当时要这些语言工具解决的问题，与今天的应用相比是相当简单的。FORTRAN是为科学应用设计的，COBOL是为商业应用设计的。今天它们在专门问题领域里仍是相当适用的。不过这些语言开发之后，大型的、即嵌入式计算机系统的应用领域出现了。

简单地说，**嵌入式计算机**是大型系统的一部分，例如导弹中的制导计算机，过程控制器，事务通讯网络，甚至控制一个汽车发动机或微波炉的微处理器等等。应当注意到这个问题域的多样性。一个嵌入式计算机系统可以小到一个单独的微计算机，也可以大到一个大型计算机网络。然而它们之间有相似性。一般讲，嵌入式系统是大型的，而且对于并行处理、

实时控制和高可靠性有类似的要求。

FORTRAN、COBOL 和大多数其它程序设计语言都不是为这个问题域设计的，然而我们仍然看到，许多课题把COBOL用于实时处理，把FORTRAN用于多任务应用。这些应用领域中的软件常常包含数十万行代码。难怪说我们面临软件危机。我们一直在使用的工具是古老的，而且一直用于非它们的擅长之处。

〔P〕 **1．2 Ada文化** 〔M〕

作为对软件危机的反应，美国国防部资助开发了一个非常有力的工具——Ada程序设计语言。与其它大多数语言不同，Ada是为一个专门的问题域、即嵌入式计算机系统设计的，这个领域对语言有一系列特殊的要求。Ada不是由一个委员会，而是由一个小型设计组设计的。然后通过专家和公众评审进行细化。如我们将在第 3 章叙述的，这个开发过程进行的相当有条理。

Ada代表着“程序设计技术的重大进展，它以井井有条的方法，把关于这个主题的最好观点汇集在一起，以满足实际的程序员的真正要求”〔2〕。然而对Ada并不是没有批评。大多数反对意见说它太复杂，因此不实际。我们完全不同意这种观点。Ada是基于一个不大的、容易理解的概念集合，例如数据抽象、信息隐藏以及强类型等基础上开发的。

某种意义上，Ada是直接体现许多现代软件设计方法学的一种语言，因此是表示程序设计解的一种优秀工具。它不仅鼓励使用好的设计和程序设计实践，而且如以后各章看到的，它实际上能强制进行这样的实践。正如其它人类工具的变革一样，Ada帮助我们用新的、经常是更有效的途径来寻求问题的解。

开发Ada语言的同时，美国国防部还开发了一个程序设计环境的需求，称为**Ada程序设计支持环境**（APSE）。APSE的目的是为Ada软件开发的所有方面提供支持；我们将在第22章详细分析这些需求。Ada 语言、APSE与Ada思维方式构成了我们所说的 Ada 文化，它能帮助我们管理计算机解的复杂性。

〔P〕 **1．3 Ada对软件工程的影响** 〔M〕

著名的心理学家Berjamin Whorf说过，语言“对思维过程可以有重要影响，即使它们并不决定所有这样的过程”〔3〕。我们希望从Ada文明得到相同的结果。Ada帮助打破 Vom Neumann思维方式，使我们按照实际的问题空间考虑问题的解，从而使解变得更可读、更可靠和更可维护。事实上Ada允许我们处理一个全新的问题集合，这些问题的解以前太复杂了，以致在当时无法管理。

“程序设计语言既不是软件危机问题的原因，也不是它的解决办法，但是因为它们在所有软件行为中所起的中心作用，它们不是能扩大存在的问题，就是能简化这些问题的解”〔4〕。对于Ada，我们期望达到的效果是后者。虽然单独一个Ada语言并不能解决软件危机，但是与其它适当的软件设计方法学相结合，它为软件系统的创建提供了一个有力的工具。

第2章　软件危机

设计软件系统是需要大量智能的活动。要准时完成一个有效的、可靠的和可维护的软件系统，通常是一项非常费力的任务，大型实时程序设计系统则更是如此。创建这类系统的专业人员，要具有科学家和艺术家的双重气质。

一方面，程序员在形式理论和一系列应用原理的基础上工作，在这个意义上，他是一位科学家。例如，他可能使用结构分析和面向对象的设计等软件技术，或者使用排队理论和数值分析等数学概念。另一方面，他在数据结构和算法的原料上刻划出一个系统的成分，然后把这些成分组合成一个整体。在这个意义上，程序员也是一位艺术家。也许是这种二重性使得计算机科学如此具有挑战性；同时它也引出了许多问题。

我们研究的学科是一门相当新的学科，而且迄今我们尚未得到多少关于这个学科的有普遍意义的专业知识。我们过分依赖自己的技巧来解决碰到的问题。然而在帮助克服大型软件系统开发的复杂性问题方面，技巧通常已不再有效；我们发现自己处在软件危机之中。

〔P〕　　## 2．1　危机的性质　　〔M〕

说存在软件危机实际上已是陈词滥调了〔2〕。因为我们已经在这种危机中生活了很长时间——甚至可以追溯到Augusta Ada Lovelace第一次用鹅毛笔在纸上编写Babbage的分析机程序的日子。然而直到1968年在西德加米施(Garmisch）召开的国际软件工程会议才普遍认识到危机的存在〔3〕。

某种意义上，软件危机的本质仅仅是建立软件系统比直觉告诉我们的要困难得多。我们从内心深处体验到，仅仅把告诉计算机要做什么的一些符号拼凑在一起，并不能获得好的软件。经验表明，软件系统设计比我们所认识的要复杂得多。

在正规的计算机科学教育中，我们能够建立数百行代码组成的系统，并且有把握充分了解全部操作。修改这样的软件并不困难，因为在完成它的第一次版本后的几天内，我们能够记住它的设计结构。如果测试发现了重大的问题，我们甚至能够相当容易地重新设计整个系统。

然而设计和实现由数万行——如果不是数百万行的话——代码组成的系统则完全是另一回事。完成这样一个系统显然超出了一个人的智力和体力的能力。当增加一些人参加设计并分担工作时，又引入了令人烦恼的通讯和协调问题。修改这样一个系统是困难的，因为通常没有一个人了解它的全部结构；必须代之以参考外部文档。这些文档应当看起来似乎总是在实际设计工作之后几个星期编写的。如果测试揭示了重大的问题，我们通常没有那种奢望（或者没有时间，或者没有费用）去重新设计整个系统。

当构造一个大型软件系统时，每个程序员的创造精神都曾遇到挫折。如果你请程序员告诉你基本的问题是什么，你可能得到不同的回答。例如："那个模块有奇怪的副作用"，或者"这个接口定义得不好"。像我们将看到的，这些实际上仅仅是潜在问题的征兆。不过，一般讲，所有这些征兆都导致软件系统延迟交货，昂贵，不可靠，以及经常与它们的规格说明不一致。这些是软件危机更明显的表现。

我们在直观上了解到存在软件问题，但是要把握它们的全部影响是困难的。正如 David