优
EXCELLENCE

优秀计算机软件丛书

计算机第四代语言

张海藩　孟庆昌　主编

- 突出第四代语言的原理、特点、开发方法和管理技术、选用准则
- 详述Excel、Informix-4GL、PowerBuilder开发工具的功能和使用方法
- 介绍System-Z、FoxPro－4GL、Accell、Oracle7的应用

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

优秀计算机软件丛书

计算机第四代语言

张海藩　孟庆昌　主编

郭逢荣　林亨利　朱继生　编著

電子工業出版社

内 容 提 要

第四代语言，实质上是用来快速开发应用软件的高生产率软件工具。使用第四代语言开发应用软件，不仅可使生产率大幅度提高，而且具有容易使用、容易维护等一系列突出优点。本书系统全面地介绍计算机第四代语言：第一篇深入浅出地讲述第四代语言的基本原理、主要特点、开发方法和管理技术，以及选用第四代语言的准则，并对其在近期内的发展趋势作了一些预测；第二篇至第四篇分别详细地介绍 Excel，Informix-4GL 和 PowerBuilder 等三个典型的第四代语言开发工具的基本功能和使用方法；第五篇对 System-Z，FoxBase，FoxPro，Accell 和 Oracle7 等第四代语言工具作了简要介绍。

本书是应用软件开发人员和广大最终用户不可缺少的参考资料，也可做为高等院校相关课程的教学参考书。

优秀计算机软件丛书
计算机第四代语言
张海藩 谭庆君 主编
郭逢荣 林享利 朱继生 编著
特约编辑 王荣
责任编辑 陈晓莉

*

电子工业出版社出版
北京市海淀区万寿路 173 信箱(100036)
电子工业出版社发行 各地新华书店经销
电子工业出版社计算机排版室排版
中国科学院印刷厂 印刷

*

开本：787×1092 毫米 1/16 印张：29.25 字数：705 千字
1996 年 3 月第一版 1996 年 3 月北京第一次印刷
印数：5000 册 定价：42.00 元
ISBN7-5053-3197-3/TP · 1164

编委会名单

总　　序

无论你是正在从事计算机应用的人员，还是其他领域的工程人员，即使你是一位刚刚接触计算机的电脑入门者，一套全面介绍各种计算机软件应用的技术读物是你必不可少的工具，《优秀计算机软件丛书》则是你最好的选择。本丛书汇集了当今国内外各类软件之精萃，其中的每本书综合了某一类软件的所有产品，成为该类软件之总成。本丛书是由电子工业出版社、北京软件行业协会、中国计算机用户协会北京系列机分会和中国仪器仪表学会办公自动化学会共同组织国内外专家、教授及电脑科普工作者一起编著的，它将是我国有影响的、大型综合性的计算机软件丛书。

本丛书具有以下特点：

首先，它是按照"高角度，低起点"的原则编写的。一方面从技术发展的宏观角度和软件设计的高水平来论述各类软件的全貌和特点，综述某类软件的主要原理和技术；另一方面，对具体软件的介绍，则从低起点出发，以深入浅出的文字叙述为主，摒弃繁琐的理论介绍，注重该种软件的实际应用、操作和使用。因此，特别合适作为计算机普及教育、成人教育的培训用书。

第二，它不但注重每本书的题材精选，而且更注重对每本书内容的安排。即选题上是按照流行广、通用性强、影响面大、水平高、市场好、版本新的原则选取。每本书的内容既照顾面和点的结合，又着重于从每类软件中精选出两、三种产品加以详述（点的介绍）。每本书的内容大致都包括：1. 综述与原理；2. 概览与比较；3. 操作与使用；4. 应用与编程四部分。前两部分为面，后两部分为点。

第三，每本书既强调了对两、三个有代表性软件的详细介绍，使读者可作为学习、掌握这些软件的入门、使用、操作与二次开发应用的使用手册；又注重于软件设计的主要技术、实现原理、新发展等技术性的论述，作为读者学习专业的基础；最后每本书都尽可能详细总括了同类软件的各种不同产品，并提供了对各种产品的综合分析比较，以作为读者选购、选用软件的指南。

本着对读者负责的态度，编委会对每本书的选题与技术，内容及文字都做了精心的论证和审定。每本书都由国内外知名的专家、教授主编，同时聘请有经验的软件工作者、科普工作者及教师参加编写。

"海阔凭鱼跃，天高任鸟飞"愿本丛书能为您插上坚实的翅膀，伴您高翔在计算机的广阔天空，成为您最实用的工具和揭开电脑奥秘的钥匙！

《丛书》编委会

1995.8.

前　言

应用软件的开发生产率不能与计算机硬件的巨大进步相匹配，已有的软件很难维护等等，是困扰软件行业多年的难题，已成为限制计算机应用的瓶颈。今天，在市场经济环境中的广大计算机用户，迫切需要可以立即投入使用的应用系统，并且要求应用系统在日益激烈的经济竞争中能够适应业务经常变化的特点而及时调整。显然，遵循经典的生存周期方法学使用第三代语言开发出的应用系统，无法满足上述要求。

正是在上述背景下出现了第四代语言(4GL)，从而在计算机语言和应用软件开发途径等领域中引发了一场革命。第四代语言，实质上是用来快速开发应用软件的高生产率软件工具。使用第四代语言开发应用软件，不仅可使生产率大幅度提高，而且具有容易使用、容易维护等一系列突出优点。当然，为了达到第四代语言所能带来的最大生产率增益，必须在开发方法和管理技术两个方面都进行变革。

本书深入浅出地讲述第四代语言的基本原理、主要特点、开发方法和管理技术，以及选用第四代语言的准则，并对其在近期内的发展趋势作了一些预测。

书中详细介绍了 Excel、Informix-4GL 和 PowerBuilder 等三个典型的第四代语言开发工具的基本功能和使用方法，同时对 System-Z、FoxBase、FoxPro、Accell 和 Oracle7 等第四代语言工具作了简要介绍。

本书第一篇由张海藩编写，第二篇由郭逢荣编写，第三篇由孟庆昌编写，第四篇由林亨利编写。朱继生、张海藩和孟庆昌等人共同编写了第五篇。参加本书编写工作的人员还有：刘振英、孟平、林文、赵建国、李红、马淑芳、王平一、朱国华、张晋生、杨晓红等。

由于在此之前国内尚未见到系统全面地论述计算机第四代语言的专著，可供借鉴的资料较少，加之编者水平有限，书中缺点错误在所难免，不当之处敬请读者批评指正。

编者

一九九五年三月

目　录

第一篇　基　础

第一章　第四代语言的产生

社会现代化，关键在于科学技术现代化，而计算机的广泛应用则是科学技术现代化的前提和标志。

利用计算机解决社会上的各类问题，不仅需要有计算机硬件，同时也需要有指挥硬件工作的计算机软件。计算机硬件和软件密切配合协同工作，共同构成计算机系统。如果用人来比喻计算机系统，硬件就好像是身体，软件则相当于大脑。也有人把硬件比喻为人的肉体，把软件比喻为人的灵魂（精神），这种比喻方法不仅形象通俗地说明了软、硬件功能的不同，同时也表明了它们在物理形态方面的区别：硬件是看得见模得着的物理部件，软件是看不见模不着的逻辑部件。

随着计算机应用的日益普及和深化，正在运行使用着的计算机软件的数量以惊人的速度急剧膨胀，而且现代软件通常规模十分庞大，逻辑非常复杂。由于微电子学技术的进步，计算机硬件的性能/价格比平均每十年提高两个数量级，硬件质量也在稳步提高；与此相反，计算机软件成本却在逐步上升，质量没有可靠的保证，软件开发的生产率也远远跟不上普及计算机应用的要求，软件已经成为限制计算机系统发展的瓶颈。

人们把软件开发和维护过程中所遇到的一系列严重问题统称为“软件危机”，并且从60年代后期开始认真研究消除软件危机的方法，从而逐步形成了计算机科学技术领域中的一门新兴的学科——计算机软件工程学（通常简称为软件工程）。软件工程传统的基本途径，是采用生存周期方法学来开发与维护软件。

对于某些类型的软件来说，生存周期方法学确实有效，实践中也曾取得过巨大的成功。但是，人们在实践中也逐步认识到，对于另外一些类型的软件来说，生存周期方法学并不适用。也就是说，生存周期方法学并不是软件工程所应采取的唯一途径。在某些类型的计算机应用领域中，一种更新型的开发途径正在取代经典的生存周期方法学，新途径的特点是快速地构造原型系统，因此，软件工程的这种新途径通常称为快速原型法。为适应原型法的需要，出现了一类新的软件工具，通常统称为计算机第四代语言（简称为第四代语言，缩写为4GL）。

1.1　软件危机

1.1.1　计算机系统的发展过程

过去人们往往按照计算机硬件的演变来划分计算机系统发展的时期，然而，为了了解计算机软件发展演变的过程，特别是为了了解软件危机是怎样产生的，又是如何加剧的，从而

探索解决软件危机的途径，应该更全面地回顾计算机系统发展的简短历史，按照计算机应用领域的演变而不仅仅是根据硬件特点的演变来划分计算机系统发展的时期。

从世界上出现第一台计算机到60年代中期是计算机系统发展的早期时代，包括第一代和第二代计算机。在这个时期人们用很大力气研究和发展计算机硬件，经历了从电子管计算机到晶体管计算机的变革；可是，人们对计算机软件的研究和发展却不够重视，认为软件开发是有了计算机硬件后才需要考虑的问题，基本上没有系统化的软件开发方法。软件开发在那个时期只是根据应用的需要写出能够运行的机器语言程序。由于硬件价格昂贵，运算速度低，内存容量少，所以当时的程序员非常强调"程序设计技巧"，把缩短每一微秒CPU时间和节省每一个二进制存储单元，作为程序设计（也就是那个时期的软件开发）的重要目标。

在计算机系统发展的早期时代，大多数系统采用批处理方式工作，只有个别系统是交互式系统。当时大多数软件的开发者和使用者是同一个（或同一组）人。编写程序的人要设法使程序在机器上运行，并且负责使用这个程序解决预定的问题，如果使用过程中程序发生错误也得由这个人设法修改。由于这种个体化的软件环境，使得软件设计通常是在人们头脑里进行的一个隐含的过程，而且除了程序清单和上机说明书之外，一般没有其它文档资料保存。

从60年代中期到70年代初期是计算机系统发展的第三代时期。在这个时期硬件经历了从晶体管计算机到集成电路计算机的变革，CPU速度和内存容量都有了很大提高，从而为计算机在众多领域中的应用提供了潜在的可能性。为了更有效地利用硬件的性能，计算机系统普遍采用多道程序和多用户分时的工作方式；能够从多个数据源采集数据，高速进行处理，并在很短的时间内（几毫秒）产生输出信息，从而控制实际过程进行的实时系统，开辟了计算机应用的一个新领域；联机辅助存储设备的进展导致出现了第一代数据库管理系统。

这个时期的另一个重要特征是出现了"软件作坊"，广泛使用商品软件。这是因为随着计算机应用的普及和深化，需要的软件往往规模相当庞大，以致单个用户无力开发；此外，许多不同的部门和企业往往需要相同的或类似的软件，各自开发就会使人力浪费很大。在这种形势下"软件作坊"就应运而生了，许多用户不再是自己开发软件而是购买或定做软件。不过，在这个"软件作坊"时期，开发软件的方法，基本上仍然沿用早期时代形成的个体手工业式的开发方法。

随着计算机系统数量的不断增加，计算机软件库就开始膨胀。但是，在程序运行中发现错误时必须设法改正；当用户的要求有改变时也必须相应地修改程序；当买进新硬件或操作系统更新版本时通常必须修改程序以适应新的环境。上述种种软件维护工作，以令人吃惊的比例耗费资源。更严重的是，许多程序的个体化特性使得它们最终成为不可维护的。"软件危机"出现了！1968年北大西洋公约组织的计算机科学家曾在当时的联邦德国召开国际会议讨论软件危机问题，在这次会议上正式提出并使用了"软件工程"这个名词，一门新兴的工程学科就此出现了。

计算机系统发展的第四代从70年代初期开始，一直沿续到80年代。这个时期硬件发展的特点是从集成电路计算机进步到大规模和超大规模集成电路计算机，高性能低成本的微处理机大量涌现，发展日新月异。

这个时期计算机应用又有了进一步的普及和深化，开辟了许多新的应用领域。分布式系

统使用多台计算机共同解决一个问题，各台计算机并行工作分别完成各自的子任务，同时各台计算机之间进行必要的通信，增加了计算机系统的复杂程度；利用计算机硬件提供的简单算术运算和逻辑运算能力，模拟人类复杂的思维过程的人工智能系统，更需要十分复杂的计算机软件才能实现。

硬件的迅速发展已经远远超出人们所能提供的软件支撑能力，然而，硬件只提供了潜在的计算和逻辑能力，如果没有软件来驾驭和开发这种能力，人类并不能有效地使用计算机。在计算机系统发展的第四代，软件危机进一步加剧了。软件维护费用占数据处理总预算的50%以上，软件开发生产率远远满足不了对于新系统的需求。为了对付日益严重的软件危机，计算机科学家和软件工程师开始认真研究和采用软件工程学。

那么，什么是软件危机的含义呢？

1.1.2 软件危机的含义

软件危机指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这些问题绝不仅仅是“不能正常运行的”软件才具有的，实际上几乎所有软件都不同程度地存在这些问题。概括地说，软件危机包含下述两方面的问题：如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件，以延长这些软件的使用寿命。具体地说，软件危机主要有下述一些表现：

1. 对软件开发成本和进度的估计不准确。实际成本比估计成本高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。

2. 用户对“已完成的”软件系统不满意的现象经常发生。软件开发人员常常在对用户要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就仓促上阵匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”必然导致最终的产品不符合用户的实际需要。

3. 软件产品的质量不稳定。软件可靠性和质量保证的确切的定量概念才提出不久，软件质量保证技术（审查、复审和测试）还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

4. 软件常常是不可维护的。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需要在原有程序中增加一些新的功能。“可再用的软件”还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

5. 软件通常没有适当的文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新式的”（即和程序代码完全一致的）。软件开发组织的管理人员可以使用这些文档资料作为“里程碑”，来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是至关重要的和必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

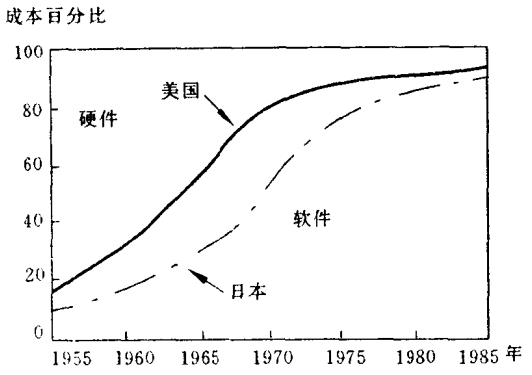

图 1-1　美、日两国计算机硬件和软件成本的变化趋势

6. 软件成本在计算机系统总成本中所占的比例逐年上升。由于微电子学技术的进步和生产自动化程度不断提高，硬件成本逐年下降，而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。图 1-1 描绘了美、日两国计算机硬件成本和软件成本在计算机系统总成本中所占比例逐年变化的情况，可以看出在 1985 年软件成本已占总成本的 90%左右。

7. 软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象使得人类不能充分利用计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些明显的表现，与软件开发和维护有关的问题远远不止这些。那么，为什么会发生软件危机呢？

1.1.3　软件危机的原因

在软件开发和维护的过程中存在的许多严重问题，一方面与软件的特点有关，另一方面也和软件开发与维护的方法不正确有关。

软件不同于硬件，它是计算机系统中的逻辑部件而不是物理部件。在写出程序代码并在计算机上试运行之前，软件开发过程的进展情况较难衡量，软件开发的质量也较难评价，因此，管理和控制软件开发过程相当困难。此外，软件在运行过程中不会因为使用时间过长而被“用坏”，如果运行中发现错误，则很可能是遇到了一个在开发时期引入的在测试阶段没能检测出来的故障。因此，软件维护通常意味着改正或修改原来的设计，并根据修改后的设计重新编写有关的程序，这就在客观上使得软件较难维护。

软件独有的特点确实给开发和维护带来一些客观困难，但是，人们在开发和使用计算机系统的长期过程中，也确实积累和总结了许多成功的经验。如果坚持不懈地使用经过实践考验证明是正确的方法，许多困难是完全可以克服的，过去也确实有一些成功的范例；可是，目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念，在实践过程中或多或少地采用了错误的方法和技术，这可能是使软件问题发展成软件危机的主要原因。

与软件开发和维护有关的许多错误认识和做法的形成，可以归因于在计算机系统发展的早期时代软件开发的个体化特点，今天这些错误认识仍然迷惑着不少软件人员。因此，为树立正确的概念来指导软件开发和维护工作，有必要对若干主要的错误认识做一番剖析，在下面的叙述中引号内是典型的错误认识，在每个错误认识后面的论述是对它的剖析。

“有一个对目标的概括描述就足以着手编写程序了，许多细节可以在以后再补充。”

事实上，对用户需求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。只有用户才真正了解他们自己的需要，但是，许多用户在开始时并不能准确具体地叙述他们的需要，软件开发人员需要做大量深入细致的调查研究工作，反复多次地和用户交流信息，才能真正全面、准确、具体地了解用户的要求。对问题和目标的正确认

识是解决任何问题的前提和出发点，软件开发同样也不例外。急于求成仓促上阵，对用户需求没有正确认识就匆忙着手编写程序，这就如同不打好地基就盖高楼一样，最终必然垮台。

“软件开发就是编写程序并设法使它运行。”

一个软件从定义、开发、使用和维护，直到最终被废弃，经历了一个漫长的时期，通常把软件经历的这个时期称为生存周期。正如上面讲过的，软件开发最初的工作是问题定义，也就是确定要求解决的问题是什么；然后要进行可行性研究，决定该问题是否存在一个可行的解决办法；接下来应该进行需求分析，也就是深入具体地了解用户的要求，在所要开发的系统（不妨称之为目标系统）必须做什么这个问题上和用户取得一致的看法。经过上述软件定义时期的准备工作后才能进入开发时期，在开发时期首先需要对软件进行设计（通常分为总体设计和详细设计两个阶段），然后才能进入编写程序的阶段，程序编完之后还必须经过大量的调试工作（需要的工作量通常占软件开发全部工作量的40%～50%）才能最终交付使用。所以，编写程序只是软件开发过程中的一个阶段，而且在典型的软件开发工程中，编写程序所需的工作量一般只占软件开发全部工作量的20%左右。

另一方面还必须认识到程序只是完整的软件产品的一个组成部分，在上述软件生存周期的每个阶段都要得出最终产品的一个或几个组成部分（这些组成部分通常以文档资料的形式存在）。Boehm曾经指出：“软件是程序以及开发、使用和维护程序需要的所有文档。”这也就是对软件的定义。所以，一个软件产品必须由一个完整的配置组成，应该肃清只重视程序而忽视软件配置的其余成分的糊涂观念。

“用户对软件的要求不断变化，然而软件是柔软而灵活的，可以轻易地改动。”

确实，用户对软件的要求经常改变，特别是一个大型软件开发项目持续的时间往往相当长，在这段时间内由于外界环境变化以及人的认识不断深化，都会或多或少地改变对软件的要求。但是，必须看到也有相当多的改动不是由于用户要求的变化所造成的，而是由于软件开发人员在开发初期没有完全理解用户的要求，直到设计阶段甚至验收阶段才发现“已完成的”软件不完全符合用户的需要，从而必须进行修改。

严重的问题是，在软件开发的不同阶段进行修改需要付出的代价是很不相同的，在早期引入变动涉及的面较少，因而代价也比较低；而在开发的中期软件配置的许多成分已经完成，引入一个变动要对所有已完成的配置成分都做相应的修改，不仅工作量大而且逻辑上也更复杂，因此付出的代价增加很快；在软件“已经完成”时再引入变动，当然需要付出更高的代价。根据美国一些软件公司的统计资料，在后期引入一个变动比在早期引入相同变动所付出的代价高2～3个数量级。图1-2定性地描绘了在不同时期引入一个变动需要付出的代价的变化趋势。图1-3是美国贝尔实验室统计得出的定量结果。

“软件投入生产性运行以后需要的维护工作并不多，而且维护是一种很容易做的简单工作。”

由于有这种看法，给维护分配的预算往往比较少，配备的人员也比较弱；然而，轻视维护是一个最大的错误。许多软件产品的使用寿命长达十年甚至二十年，在这漫长的时期中不仅必须改正使用过程中发现的每一个潜伏的错误，而且当环境变化时（例如硬件或系统软件更新换代）还必须相应地修改软件以适应新的环境，特别是必须经常改进或扩充原来的软件以满足用户不断变化的需要。所有这些改动都属于维护工作，并且是在软件已经完成之后进行的。因此，维护是极端艰巨复杂的工作，需要花费很大代价。统计数据表明，实际上用