全国高等教育自学考试教材
计算机信息管理专业

软件开发工具

（附软件开发工具自学考试大纲）

全国高等教育自学考试指导委员会组编
陈　禹　方美琪　主编

经济科学出版社

软件开发工具

全国高等教育自学考试教材
计算机信息管理专业

软件开发工具

（附软件开发工具自学考试大纲）

全国高等教育自学考试指导委员会组编

陈　禹　方美琪　主编

经济科学出版社

责任编辑：王蜀伟
责任校对：段健瑛
封面设计：张卫红
版式设计：代小卫
技术编辑：舒天安

软件开发工具（附软件开发工具自学考试大纲）
——全国高等教育自学考试教材计算机信息管理专业
全国高等教育自学考试指导委员会组编
陈　禹　方美琪　主编
*
经济科学出版社出版、发行
北京市海淀区万泉河路66号
新华书店经销
中国铁道出版社印刷厂印刷
*
787×1092毫米　16开　15.25印张　400000字
1996年9月第一版　1999年1月第五次印刷
印数：21001—26000册
ISBN 7-5058-1038-3/G·158　定价：19.00元

软件开发工具

出版前言

编写高等教育自学考试教材是高等教育自学考试工作的一项基本建设。经国家教育委员会同意，我们拟有计划、有步骤地组织编写一批高等教育自学考试教材，以满足社会自学和适应考试的需要。《软件开发工具》是为高等教育自学考试计算机信息管理专业组编的一套教材中的一种。这本教材根据专业考试计划，从造就和选拔人才的需要出发，按照全国高等教育自学考试指导委员会颁布的《软件开发工具自学考试大纲》的要求，结合自学考试的特点，组织高等院校一些专家学者集体编写而成的。

计算机信息管理专业《软件开发工具》自学考试教材，是供个人自学、社会助学和国家考试使用的。现组织专家审定同意予以出版发行。我们相信，随着高教自学考试教材的陆续出版，必将对我国高等教育事业的发展，保证自学考试的质量起到积极的促进作用。

编写高等教育自学考试教材是一种新的尝试，希望得到社会各方面的关怀和支持，使它在使用中不断提高和日臻完善。

全国高等教育自学考试指导委员会

一九九六年七月

前　言

作为一种技术，软件的开发只是近几十年的事。但是，它的发展变化却是非常迅速的。可以设想，如果一位70年代初的软件工作者，突然处于Power Builder或Visual Age的环境之中，他将是怎样地吃惊和激动！所以，作为90年代的软件工作者，必须在掌握必要的基础知识（如数据结构等课程）的同时，认识与了解软件技术发展的全貌及趋势，以便在日新月异的技术进步的大潮中始终在发展的前沿，处于主动的地位。然而，要做到这一点是不容易的，除了软件技术资料的数量极大，各种分枝技术交叉结合之外，最主要的难点在于把握变动之中的不变的、基本的趋势与原则。许多资深专家指出，基于人们对软件开发过程的理解与认识，充分利用计算机本身处理信息的巨大潜力，不断提高软件开发工作的效率，这就是软件技术发展的总趋势，在一定意义上可以简要地概括为软件开发工具的不断改进与发展。正是基于这一点，本专业列入了软件开发工具这门课程。

由软件开发工具这门课程的来历中，很自然地引伸出以下两个基本观点。首先，它是学员们在掌握基本的软件开发技术的基础上扩大视野、加深认识，向主持大型软件开发的水平前进的一个重要台阶。由于在目前的专业课程中，没有软件工程或软件开发方法学的内容，本课程应当在程序员向项目负责人提高的过程中发挥作用。其次，它的学习重点是关于基本观点的理解，而不是具体的某一软件。如果把本课程仅当作多学一种或几种具体软件，那就会出现十分混乱的局面：一方面，由于软件工具的种类繁多及不断更新，学员将无所适从，组织者更无法考核。另一方面，学员将难于真正掌握学习本专业的实质内容，失去学习的意义，从而无法达到提高水平的目标。

本书前两章介绍了软件开发工具的由来，着重讨论了软件开发工作的困难及克服途径。这里虽未具体讲软件开发工具，但是为后面的讨论提供了背景和基础。第三四章是全书的核心。这两章从理论基础和实际技术两方面，讨论了软件开发工具的基本要点，具有一定的普遍性。第五章则从实际工作需要出发，概述了选择、使用以及自行开发软件中的各种具体问题。第六章提供了软件开发工具的现状及发展的若干材料，供读者进一步研究之用。我们在附录中选择介绍了4个具有一定特点的软件工具，目的是为读者提供一些具体的、比较感性的材料，对于教学也有一定的典型意义。附录1介绍的Rose是Rational公司聚集了世界上多位最具权威的软件专家研制而成的，其基础是最近几年发展起来的面向对象的方法。附录2介绍的RDCASE试图支持软件开发的全过程。附录3介绍的PowerBuilder是目前应用范围较广泛的软件工具。附录4介绍的Mark V是一种主要用于系统分析阶段的支持多种方法论的元工具，它能支持用户创建自己的方法。

本书的第二、三、四章由方美琪执笔。第一、五、六章由陈禹执笔。附录1由杨小平和张

逸薇、严嫣、吴凌等执笔，附录 2 由欧琼霞执笔，附录 3 由王蓉执笔，附录 4 由邓洪涛执笔。全书由陈禹，方美琪统稿。

本书初稿完成后承蒙清华大学郑人杰教授、北京软件行业协会殷志鹤会长、北京信息工程学院邝孔武教授审阅，他们对本书提出了许多宝贵意见。在本书的编写过程中，上海交通大学陈敏逊教授和经济科学出版社的同志付出了大量的劳动。在此对以上各位致以衷心的感谢。

以“软件开发工具”为题编写教材还是一种尝试，加上作者学识与资料的局限，不妥之处是必然存在的，恳请同行及广大读者不吝赐教。

作　者

一九九六年七月

于中国人民大学信息学院

目　录

软件开发工具

软件开发工具自学考试大纲

第一章 绪 论

软件开发工具是计算机技术发展的产物。随着以电子计算机为代表的现代信息技术迅速地应用到社会生活的各个角落，社会对于各种软件的需求也日益紧迫。各行各业都要求软件开发者迅速地、高质量地提供各种各样的软件产品，从过程控制软件到各种管理软件，从辅助设计软件到辅助教学软件。软件产品的质量、效率、价格已成为各方关注的十分重要的问题。

作为对策之一，软件开发工具应运而生。它以计算机自身处理信息的能力为基础，在软件开发的各个阶段，对软件开发的各个方面提供了各种各样的帮助，从而成为今天的软件工作人员必须具备的重要技术手段。

本章从软件开发工具的产生讲起，概括地介绍了软件开发工具的基本概念、基本结构、一般功能、主要类别以及它的使用方法。这些内容基本勾画出了这一领域的概况，为以后各章详细讨论它的理论与技术打下了基础。

1.1 软件开发工具的由来

和任何技术一样，软件开发工具是在人类以往发展的许多技术的基础上，适应社会的实际需要，从无到有，从不完善到比较完善逐步发展起来的。回顾它的产生与发展的经过，就能很自然地理解它的作用与实质。

粗略地说，软件开发工具就是帮助人们开发软件的工具。当然，这样笼统地讲是不太确切的。一般来说，谁也不会把计算机硬件，或者编写软件时用的纸和笔归入软件开发工具，虽然它们确实是开发软件所必不可少的。事实上，我们所说的是一种软件，帮助人们开发软件的软件。但是这样说也还有不清楚的地方。比如在计算机技术中归入系统软件的操作系统、汇编程序、编译系统无疑也是软件开发的必不可少的工具，但是，一般来说并不把它们归入软件开发工具的范围。类似的还有一些服务软件，如磁盘的处理、病毒的防治、系统的备份等，尽管它们对于软件开发者来说是很有用的工具，甚至名字就叫工具（如 Pctools），但也还不是我们这里的软件开发工具。

就目前通行的理解而言，软件开发工具的范围大致可以描述为：在高级程序设计语言（第三代语言）的基础上，为提高软件开发的质量和效率，从规划、分析、设计、测试、成文和管理各方面，对软件开发者提供各种不同程度的帮助的一类新型的软件。

对于这个概念可以从以下几个方面去加深理解。首先，我们这里说的是一类软件。它本身就是一种软件，通俗地说，它是开发软件用的软件。其次，它是在第三代语言的基础上发展起来的。第三代语言在这里是已经形成的软件开发技术。所谓“帮助”或“支持”的意思是指在人们用第三代语言编写软件时，进一步的要求与希望，因此并不包含第三代语言本身。当然，这条界限似乎是人为划定的，没有多少道理，但是如果没有这条界限，我们的讨论就会因无法限定范围而失去控制。另外，这里强调对软件开发全过程中各个阶段的支持，即不仅包含狭义的编程阶段（Coding），而是包括了相当广泛的范围和相当丰富的内容。把开发工具局限于某一类

型的代码生成器是不全面的。

为了深入地了解软件开发工具的概念和意义，必须从它的实际的产生与演变过程来分析与认识。

当现代的电子计算机诞生时，人们面对的是只能执行机器指令的硬件设备，即所谓“裸机”状态。机器的每一个动作都需要人们用二进制的字符串，即只由 0 和 1 组成的字符串书写出来，并用纸带等光电设备或通过控制台上的扳键送入机器，才能得以存储和执行。这时，软件的概念还只处于萌芽状态，人们的注意力还集中于如何设计出各种各样的物理器件，实现最基本的运算与操作，如加法器、乘法器、解码器等。这在当时是很自然的，也是必须的。在这个阶段，计算的方法似乎是已经定好了的，无须人们去设计。例如，某些方程组的解法（高斯迭代法）、某些运筹学的算法（线性规划所用的单纯形法）。这些在数学上已经证明了，并且完全形式化了的算法，已经有了明确的、严格的表述。当人们用计算机解这些问题时，人们要做的是，把这些表述逐条转换成机器指令，表达成机器能够识别和执行的 01 字符串。显然，这里的工作是一种十分机械、十分繁琐的枯燥事情。如果说这可以算做软件工作的话，那么需要的是熟记机器代码和非常大的耐性，而不是今天的软件技术。不过，从这种萌芽状态的软件工作中，已经可以看出日后大大发展起来的软件工作的基本特征：专业知识与计算机技术之间的桥梁。正是这个桥梁作用的有效发挥和不断延伸，构成了软件技术发展的历史主线索。这个阶段就是我们大家熟知的机器语言阶段，从发展的阶段来分，可以称之为第一代的计算机语言。

十分明显的是，这种机器语言的使用是十分艰难的。且不说强记那些机器代码有多么困难，就说要求把计算方法的过程一步一步地与计算机的基本操作对应起来，也足以使人望而却步。难怪在计算机产生的早期，它的应用只能局限于高科技和军事领域。的确，如果不是涉及到国家的安危和民族的存亡，没有国家的不计成本的巨大投入，不是到了人力实在无法承受那庞大的计算量的时候，恐怕就不会有哪个机关或研究所能够制造出第一批的现代意义下的电子计算机。然而，当硬件技术在微电子学成果的推动下得到突破性的进展时，这种使用上的困难就变得明显起来，并成为计算机扩大应用范围的主要障碍。硬件所提供的巨大的，几乎是无穷的信息处理能力，与计算机使用方式的困难形成了尖锐的矛盾。如果不解决这个问题，不简化使用计算机的方式，计算机的广泛应用将是不可想象的。

在这方面迈出的第一步是汇编语言，即第二代语言的出现。针对难以记忆的、无意义的、枯燥的 01 字符串，人们试图用在英语中具有一定意义的单词（或单词的缩写）来代替它，这就是助记忆码，或汇编码。用汇编码编写程序称为汇编语言，而把这些汇编指令转化为机器指令的程序则称为汇编系统（ASSEMBLER）。这一进步虽说不算太大（它并未改变通过逐条命令来指挥机器的状态，只是使得编程序的困难减小了），但是却代表了改进计算机应用的基本方向：建立一些专用的“工具”，使某些可以由机器来完成的信息处理工作交给计算机去做，而使得人们的知识、经验转化为计算机的操作。汇编语言至今还是计算机专业人员必须学习的基本知识之一。

差不多与汇编语言同时，操作系统的出现从另一方面改善了人们应用计算机的条件。操作系统利用计算机本身迅速处理信息的优势，自动地完成系统初置、文件管理、内存管理、作业管理、处理机管理等一系列工作，把计算机系统（包括主机与外围设备）中的各种资源有效地、协调地管理起来，把原先由操作员担负的大部分职责接过来。这从另一个方面改善了计算机使用的环境，也同样为计算机应用的前进发挥了十分重要的作用。

60 年代初期，FORTRAN，ALGOL 和 COBOL 等高级程序设计语言的成熟与普及，标志

着计算机真正走出了难以应用的困窘局面，开始了向社会生活各个领域的全面渗透。这就是第三代语言的时代。与汇编语言不同的是，第三代语言突破了与机器指令一一对应的限制，用尽可能接近自然语言的表达方式描述人们设想的处理过程，而把这种表达方式向机器指令的转化工作，交给专门的“工具”——编译系统去完成。另一个重要的变化是高级设计语言实现了对机器的独立性，即它不依赖特定的硬件系统，抽象地逻辑地描述处理和算法，而把硬件系统之间的区别交给不同的编译系统去处理，从而大大提高了程序的可移植性。这一进步的影响非常巨大、非常深远。从60年代开始，面向各个领域的各种应用的程序设计语言如雨后春笋般涌现出来，至今势头不减。也正因为有了这些语言，在短短的二三十年中，计算机技术已经深入到了各行各业，如水银泻地，无孔不入。

然而，技术的发展并未止步。在第三代语言的应用当中，人们又发现了新的瓶颈——处理过程的描述。原来，第三代程序设计语言一般都是过程化语言，即需要由编写程序的人一步一步地安排好机器的执行次序，虽然不是一对一地指挥到机器指令（像汇编语言那样），但是还必须在人的头脑中安排好实际的执行过程。人们希望机器能够自动地完成更多的工作，包括自动安排某些（不是全部）工作的顺序，而做到只要给机器下达做什么的命令，由机器自己去安排执行的顺序，这就是第四代语言——非过程化语言的思想。另一方面，从60年代末期开始，人们对于软件工作的认识大大深入了。在认识到软件工作的重要性的同时，也认识到了软件工作的困难性。这就是所谓“软件危机”问题。

软件工作的重要性现在已经为越来越多的人所认识。正如许多地方提出的，软件是计算机的灵魂。的确，如果没有相应的、可运用的软件，计算机是不可能在各种应用领域中发挥作用的。我们可以进一步说，软件是广大使用者与计算机之间的桥梁，软件是人类在各个领域中积累的知识的结晶，软件是人类文明与知识得以延续的新的载体，软件是人类进一步成为一个整体，得以进一步相互联系的纽带。当我们从这样的高度去认识软件，而不是像最初那样仅限于逐句翻译某一算法的时候，软件工作的困难性也就成为义中之理了。人类社会客观世界的复杂性，决定了软件本身的复杂性；人类知识的丰富多采，决定了软件的极大的多样性；客观世界的动态性，决定了软件对可维护性、可重用性越来越高的要求。面对着日益扩大规模的软件，如何保证它的正确性、可靠性，如何控制软件开发工作的进度与成本，很自然地成了理论上十分重要、实际中十分迫切的课题。

围绕这个课题，人们从开发方法的研讨、语言的改进、人员的组织等各方面想了许多方法。从结构化程序设计、软件工程、面向对象的程序设计方法直到最近的即插即用的程序设计方法(Plug and Play Programming)，许多专家作了大量研究工作。在语言方面，许多类型的第四代语言问世。与这些研究与发展相配合、相联系的，则是许多在第三代语言基础上的软件开发工具的涌现（关于第四代语言与软件开发工具的关系、软件开发工具与软件开发方法学的关系将在以后详细讨论）。

在70年代末，80年代初，苦于软件开发中遇见的各种困难的许多技术人员，已经想到了用软件来进一步支持软件开发工作。例如软件开发工作中要涉及大量文档编写工作，利用各种文字处理软件（如早期的WORDSTAR）显然就会比用手工书写方便得多，特别是需要反复修改的文件。又如，软件开发工作中有许多图要画，如流程图、结构图等，如使用绘图软件处理，无疑效果会更好，也更便于存储和修改。当然这时所用的都是一些通用的、并非专门为软件开发工作而设计的软件。所以，这个阶段可以称为利用通用软件作为辅助工具的阶段，或者反过来说：没有专用的软件开发工具的阶段。

用通用软件来帮助软件开发人员编写文档或画图可以减少不少工作量，但是与整个软件开发工作相比，这种帮助实在是太表面、太初步了。用通用软件帮助软件开发有三个主要的弱点。第一个弱点是，有许多工作是通用软件所无法完成的。例如，大量的程序编写工作中有不少重复的或基本重复的段落，完全可以找出一定的普遍规律，只用一些参数来加以控制，就可以从某些固定的模式出发，自动地制造出用第三代语言书写的程序段落。这些工作，一般的通用软件是做不了的。类似的一些报表生成、文档生成也不是一般意义下的文字处理软件所能完成的。第二个弱点是，用通用软件完成某些工作，只能表现其表面的形式，而不能反映其逻辑内涵。用形象化的语言来说，则是只能做到“形似”，而不能做到“神似”。例如用一般的绘图软件来画程序框图，虽然可以画出圆形、长方形、平行四边形等形状，但是却不能反映它们所代表的含义，更无法根据这些逻辑含义来对图的正确性或一致性作出判断，并据此向软件开发人员发出警告或提示信息，而这才是人们希望通过使用工具得到的帮助。因此，即使通用软件能够帮助软件开发人员做某些工作，也常常只是片断的、表面的工作。第三个弱点是，用通用软件来帮助人们完成软件开发工作时，常常遇到难于保持一致性的困难。例如，当用某个通用绘图软件画出关系实体图（ER 图），而用另一个通用数据库管理系统编制数据字典的时候，往往很难进行交互检验，特别是当发生变更时，更是难以保持一致。事实上 ER 图和数据字典描述的是同一个事物——客观系统中的数据结构，但是由于采用两个不同的软件来描述，就必须用两种不同的方法去各描述一遍，还必须记住随时保持它们之间的一致性。类似的问题在软件规模越来越大，涉及开发与维护的不同阶段时就会显得更为突出。

针对这些问题，80 年代以来，一些专门用于支持软件开发的软件开发工具陆续问世，从而进入了专用的软件开发工具的阶段。这期间涌现出来的软件开发工具种类很多。其中主要的有以下几类：面向特定功能模块的各种代码生成程序（包括报表生成器、菜单生成器、对话生成器等），综合性的第四代语言（一般是立足于某种数据库管理系统或某种第三代语言之上的）专用于某种文档的编写工具，数据字典管理系统（DDMS），专用于画数据流程图、ER 图或程序框图的绘图软件等等。在这些工具的推动下，软件开发工具的有关理论也逐渐发展起来，并引起了软件界的重视。软件开发工具或开发环境的概念越来越为人注目。

但是，一批专用的软件开发工具的出现，并未解决上面提到的第三个困难，一致性的保持仍然是悬而未决的难题。问题的根源很明显，那就是对软件的开发缺乏全面的、统一的支撑环境。这些零散、分散地支持各个工作阶段、各项具体工作的专用工具之间没有有机地联系起来，从而必然造成冲突与矛盾。这种冲突与矛盾对于用户来说，造成了沉重的、不堪忍受的负担，使用工具越多这种负担越重，以致抵消了使用工具带来的益处。这种情况导致了集成的软件开发工具的产生。1989 年，IBM 公司宣布了一个名为 AD/Cycle 的巨大的理论框架，作为它和它的软件合作伙伴开发一致的、统一的软件开发环境的纲领。这可以被看作是进入集成的软件开发环境阶段的标志。虽然 IBM 公司的 AD/Cycle 由于它在硬件平台与网络结构上的偏颇，在后来不得不基本放弃了，但是集成的软件开发环境的研制并没有因此而停滞下来（IBM 的主要失误在于两点，一点是拘泥于集中式的处理思路，对于新的客户机/服务器结构未能及时适应，另一点是它坚持在 OS/400 等自己的特殊硬件平台上开发，背离了开放性的大趋势）。例如，集中了 Grady Booch、James Rumbaugh 等著名专家的 Rational 公司最近推出了 Objectory、Rose 等集成的软件开发工具。国内北京大学等许多单位研制的青鸟系列软件也属于这一类的产品。

回顾软件开发工具从无到有，从分散到集成的发展过程，可以清楚地看到软件开发工具的出现决不是偶然的，它是软件发展的必然趋势，是软件技术发展的一个新的阶段。

1.2 软件开发工具的概念

从上一节的讨论中，我们已经提出了软件开发工具的概念，它的要点是：(1) 它是在高级程序设计语言（第三代语言）之后，软件技术进一步发展的产物；(2) 它的目的是在人们开发软件过程中给予人们各种不同方面、不同程度的支持或帮助；(3) 它支持软件开发的全过程，而不是仅限于编码或其他特定的工作阶段。在理解这个概念时，应当同时认识它的继承性与创新性。也就是说，一方面要充分认识到，软件开发工具是软件技术发展的必然产物和自然的趋势，它的基本思想仍是致力于软件开发的高效优质。另一方面，随着人类对软件与软件开发过程理解的深入，它又具备了一些以前的软件开发工作所没有的新的思想与方法，而这些正是它区别于以前的软件技术的关键所在。

首先，我们需要对软件的实质进行再认识。众所周知，软件（Software）这个名词是有了计算机之后才产生的，而硬件（hardware）则是自古就有的。我们从上一节的说明中已经看到，只会执行若干基本指令的机器本身，虽然具备高速运算与海量存储的潜在能力，但是如果没有事先准备好的一系列指令，它是不能完成实际任务的。即使由人一条一条地输入指令（通过扳键或光电设备），也只能以人们的输入速度来工作，它的巨大潜力是无法发挥出来的。这里的关键是要有一套事先编好并存入机器的指令，这就是我们现在所说的程序。一台存入了某种程序的计算机与一台没有存入这种程序的计算机，从外表是看不出区别的。然而前者在接到一个启动命令之后可以自动地执行某项任务，而后者却做不到这点。为了区分和描述，人们从已有词汇中借来了 hardware 特指看得见摸得着的硬件。而与之相对地，新造了 software——软件这个新词，用来特指这种看不见、摸不着的，但又发挥着十分重要的作用的、事先编好的指令系列。它们之间的关系，正如人们所说的，硬件是躯体，软件是灵魂，二者缺一不可。

然而，从应用的角度来看，硬件与软件的情况有着极大的差别。硬件提供的是信息存储与处理的基础，这对于任何领域的应用是一样的，没有什么区别的，它不必随应用领域的变化而改变（我们这里主要是指计算机本身，而不包括外围设备。对于外围设备，它的配置完全由应用领域确定）。正如前面指出的，它一头连着计算机硬件，向硬件提供它可以执行的机器指令，另一头面向用户，接受用户提出的要求，提供的算法。从这个意义上说，软件是用户与硬件之间的桥梁。正因为如此，360 行就要有 360 种不同的软件。所以，可以说，为了推广和普及计算机的应用，大量的工作集中在软件领域之中。

从更深一层的意义去理解，软件实际上是人类知识与经验的结晶。所谓事先编好的指令，正是人们在实践中形成的工作规范与步骤。以运筹学和数理统计中的算法为例，每一个程序都是以一定的理论分析与研究为基础的。当人们把这些程序编制出来时，这就是为这些经验或理论知识找到了一种新的载体。这种新的载体与书本纸张作为知识的载体不同，它看不见、摸不着，但是却能在计算机上实施，而且可以对不同的数据反复地使用。不可见是这种载体的一个主要缺点，这导致这些知识或经验在传播与应用中的困难。针对这一点，一些专家提出了软件应当包括程序和文档两个不可缺少的组成部分。这一进步使软件的实质表现出来，作为人类知识财富积累的一种新的手段，它的重要性与地位得到越来越广泛的认可。

如果从人类文明延续的角度看，软件的意义则更为深远。如果说文字的出现是人类文明历史的开端，那么软件这种知识载体的产生将进一步提高人类集中与保存知识与经验的能力。文字只是记录信息，而不包括各种处理方法与技术。而计算机则把人做事的方法与步骤（有时还

只是设想的做法）存储起来，并在任何需要的时候重新执行。从数控机床、自动控制直到许多管理软件都是起着这样的作用。有了计算机软件，再加上方便的通信条件，人与社会的联系就变得更加紧密了，任何人都可以迅速地获取社会的、文化的、技术的最新信息，而他的经验、知识、研究成果（只要他愿意）也可以立即投入人类知识的总汇之中。

因此，对于软件的认识是逐步深入的。人们越来越认识到，单纯的、机械的编程并不是软件开发工作的关键之处，更不是它的全部。在努力提高编程工作的质量与效率的同时，还必须从知识的提取、积累、精确化等方面做大量的工作。

谈到软件开发工作的发展变化，我们把它归纳为四个不同的阶段。最初阶段的工作仅限于把用户已经明确表述出来的算法，用机器语言写成一系列机器指令，供硬件运行使用。这可以说是人们对软件开发工作的最初的认识（见图 1.1）。

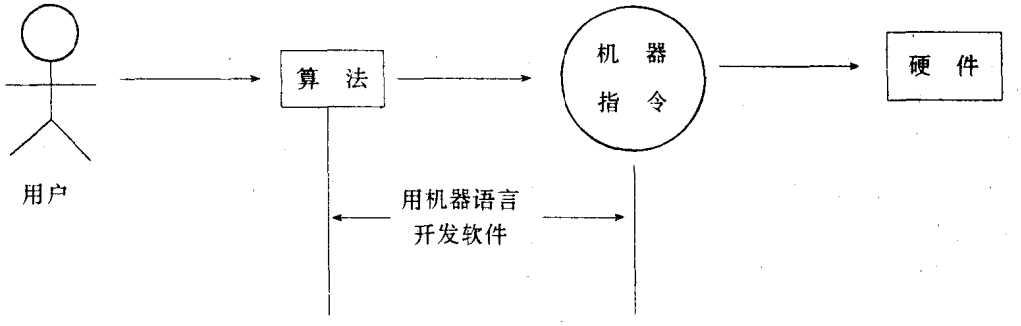

图 1.1 软件开发工作的最初阶段

汇编语言产生之后，情况略有变化。编程工作改为用汇编语言进行，编好的汇编指令由汇编程序转化为机器指令，再交硬件执行。这里的变化可以归纳为三方面。

(1) 能交给机器执行的事情，就通过一定的专用系统去做。可以说，与机器的距离扩大了。(2) 使用的通信方式——语言变了，从机器语言变成了汇编语言。(3) 由于语言的变化，与用户的距离近了，从天书般的机器语言变成了比较接近自然语言的汇编语言了。这可以称之为人类对软件开发工作认识的第二阶段（见图 1.2）。

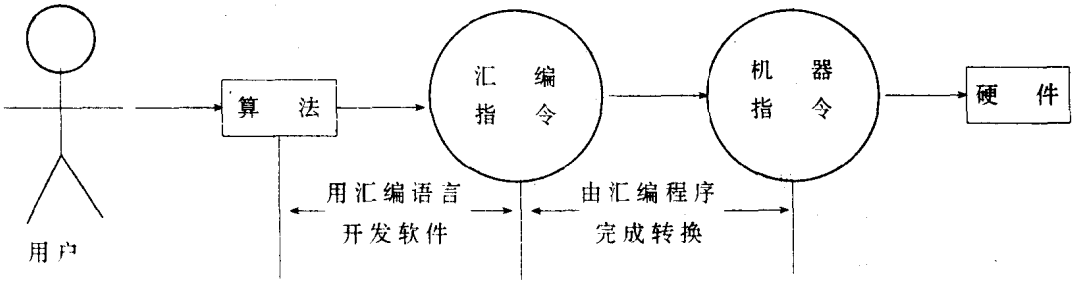

图 1.2 软件开发工作的第二阶段

第三阶段的情况表面上变化不大，只是把汇编语言变成了高级程序设计语言（第三代语言）。然而正如前面所讲过的，高级程序设计语言不再是与机器指令一一对应，而且更加接近人类习惯的自然语言。因此，可以说是离机器更远了，离用户更近了（见图 1.3）。