现代软件工程

上

管理技术篇

周之英 编著

科学出版社

现代软件工程（上）

管理技术篇

周之英 编著

科学出版社

2001

内 容 简 介

本书分上、中、下三册，每册独立成篇，上册为管理技术篇，中册为基本方法篇，下册为新技术篇。上册从宏观的角度讨论软件工程的管理技术。首先从分析和讨论软件开发技术的发展史开始，建立软件工程技术的发展的观点和变化的观点，以此作为全套书的线索，进而讨论了软件工程的管理技术：软件风险管理、软件生命期管理和软件工程技术的基本原则。本书的三个专题涉及软件工程的标准和软件度量，它们提供了技术管理层决策判断的工具和准则，是软件工程技术方法的重要组成部分。

上册可作为学习计算机软件工程和信息系统工程的大学生、研究生的教材或参考资料。对从事软件工程管理人员来说，本书内容是提高管理水平的重要参考资料。它也可帮助从事技术工作的人员进一步提高软件工程能力。

图书在版编目（CIP）数据

现代软件工程（上）：管理技术篇/周之英编著．-北京：科学出版社，1999

ISBN 7-03-007677-X

Ⅰ．现… Ⅱ．周… Ⅲ．软件工程 Ⅳ．TP311.5

中国版本图书馆 CIP 数据核字（1999）第 23974 号

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

新蕾印刷厂印刷

科学出版社发行 各地新华书店经销

*

1999 年 9 月第 一 版 开本：787×1092 1/16

2001 年 6 月第三次印刷 印张：13 3/4

印数：6 001—8 000 字数：306 000

定价：18.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

前　言

我们即将迎来的21世纪信息社会将高度地依赖于信息系统。事实上，在现代化的社会中已经很难想象没有“计算机”、没有“软件”会是怎样，面对着那无穷无尽的现实的和潜在的计算机应用需求，研究如何更快、更好、更多、更方便地开发出各种不同类型、不同目的的软件，这就是软件开发技术和软件工程技术所要解决的一个问题。

软件开发技术一直是软件工作者的主要研究方向。50多年来，随着计算机系统的发展，软件开发技术也发生着变化。软件工程首先是为了解决软件危机而提出的。其目的是用成功的、卓越的开发经验来指导，通过类似于工业化的管理，把一般程度的开发人员的水平提高到优秀水平。软件开发技术的巨大成就，已经使软件开发不再是少数逻辑天才或专家的专利，而是广大用户可以参与和直接开发自己的应用项目。因此，软件工程技术和设计方法将会受到更多人的关注。

90年代以来，软件工程不仅从方法论的角度为管理人员和开发人员提供可见的结构和有序的思考的方式，而且大量的成功软件总结出的设计经验，使软件开发人员在面对新的项目时，不必从头做起，而可以充分利用设计模式、框架、部件库等等。网络计算环境提供了经济全球化的新倾向，硬件技术以每6个月为周期的速度在发展，这些新的需求要求软件能具有充分的适应能力，去适应各种不同类型的连通和变动要求。

因此，老的软件工程教学体系已基本不能反映这些新技术和新需求的现状。从1996年我开始逐步对清华大学研究生校级学位课“软件工程技术与设计”的教学内容进行更新。教学的基本方针强调软件技术发展的变动性。我认为在急速变化的技术与社会环境中，无法想象还坚持不折不扣地照搬以前的成功经验会在新环境下继续成功，也无法想象不了解过去条件下的成功与失败，就可以迅速地创造出全新的方法与产品。因此，当前“软件工程技术与设计”的新的教学内容是以软件开发技术的发展史为纲，试图说明每种软件工程技术发展的原因、解决的问题及局限性，以使学生不仅学到技术知识，更强调能根据具体情况灵活应用，甚至创造性地推动技术的更进一步发展。目前“软件工程技术与设计”课程讲课大纲如下：

- 软件开发技术发展史
- 软件危机及风险研究的重要性
- 软件工程技术方法的基本原则
- 软件过程的CMM模型
- 软件过程改进的实例分析
- 软件过程改进现状
- 需求工程的意义及现状
- 重要的需求分析及规格说明技术（功能性为主，结构化方法）
- 重要的需求分析及规格说明技术（控制为主，状态机，Petri Net）
- 重要的需求分析及规格说明技术（形式方法）

- 软件体系结构研究的意义及现状
- 基本软件体系结构
- 软件设计方法的发展概述
- 面向对象设计方法概述
- Use Case OOD
- Design Pattern
- 软件过程标准化问题
- CORBA（软件部件接口标准）
- DCOM/COM（OLE）（软件部件接口标准）
- 软件度量
- 实例分析（实时系统的设计）
- 自选（必须与教师协商）

当然，现在学生上计算机课已经不满足于单单在课堂上学习理论。理论如果不与实际应用相结合，是无法深刻地理解和应用的，更难于发扬创造精神。但由于条件的限制，这一点还有待于今后的改进。

本书是我多年从事软件工程教学的总结。全书共分为五部分：第一部分是总论，涉及一些全局性的问题，如软件开发技术发展史、软件风险、软件工程技术的基本原则和软件生命期过程及改进问题。第二部分讨论与需求工程有关的各种问题。第三部分是与软件系统的设计与构造有关的问题。第四部分涉及分布系统，它对网络计算环境有特殊意义。第五部分是三个专题，其中有关标准和软件度量是更高层次上的问题。我们说，大部分软件技术和方法提供了软件的可见性，标准和度量提供了判断上的工具和准则，因此正文部分和专题部分是相辅相成的，但传统上更强调方法论。本书没有涉及编码和测试问题。这虽然也很重要，但前者有其他课程讨论，后者更需要针对性地讨论，所以暂时不纳入本书范围内。

本书分上、中、下三册出版。除了要全面学习软件工程的学生和从事软件工作的人员外，分册出版可以帮助他们（她们）更方便地找到最需要的内容。上册包括第一部分和三个专题。软件开发的管理层次的人员会对这些内容有兴趣。中册是需求工程（第二部分）和设计方法（第三部分的大部分内容）。软件系统的开发技术人员可以在其中找到灵感，获取进行软件需求分析和设计方面的可选用的方法。下册则是现代软件工程成就的集中体现。包括第三部分中代表了优秀的软件设计经验复用的重要途径——设计模式方面的内容和第四部分——分布系统方面的成果。对于在基本软件工程理论和方法方面有相当造诣，需要了解 90 年代以来最新发展的工程技术人员来说，一定会感兴趣。

选修本课程的清华大学计算机系和其他系的近百名研究生和进修教师——姚诚伟、裴亚民、韩松、曾伟林、刘波、肖奔放、张义、宋晖、徐伟华、黄扬清等，在学习本课程后收集了大量资料和报告。在本书的内容中，吸纳了其中一部分内容。没有他们的积极参与和共同讨论，本书是不可能现在就面世的。协助本书编写工作的还有清华大学计算机系软件教研组的谢若阳老师和计算机系的部分大学生。在此一并表示感谢。

由于目前国内有关软件工程技术与设计方面的资料比较缺乏，因此本课程新的教学内容大都参考国外的书籍和资料。当前的技术发展可说是日新月异，而教学任务要求尽

快把教学内容整理出版，以供急需。因时间和水平所限，一定有许多不周到和不准确之处，恳切希望读者提出批评和建议。

清华大学计算机科学与技术系　周之英

1999 年 4 月 5 日于清华园

目　录

第一章 软件开发技术发展史

1.1 信息社会与计算机

我们即将迎来的21世纪将是信息社会。农业社会里，生存资源是土地，个体的农业劳动和手工业劳动是社会生产的基本形式，劳动生产率很低。在工业社会中，生存资源是资本，大规模的机械化与电气化的工业生产是社会生产的主要形式。随着科学技术的进一步发展，特别是“3C”，即Communication（通信）、Control（控制）和Computer（计算机）的高度发展，将把人类由工业化社会带入了21世纪的信息社会。

一、信息社会的基本特征

- 战略资源是信息：信息的发掘与增加、信息的管理、信息的开发、流通和利用以及更新将是提高社会劳动生产率的主要手段。
- 知识爆炸：在信息社会里，知识更新快、老化快，呈现“知识爆炸”的现象。据有关资料估计，在信息社会里，科学技术信息的年增长率将达到40%以上，每5～8年增长10倍。
- 脑力劳动为主体：在信息社会里，大多数人是从事信息的管理与生产工作，主要进行脑力劳动。
- 核心技术与计算机有关：在信息社会里，无论个人，团体或企业；无论地区，国家或全球；无论文化，经济或军事；无论生产，管理，教育或休闲的进展，计算机都会或明或暗地起中枢神经作用。

二、计算机的特点与应用

计算机的特点主要有：

(1) 运算速度快。

(2) 精确度高。

(3) 具有“记忆”和逻辑判断能力。

(4) 程序控制自动完成操作。

计算机的应用非常广泛，已从数值计算领域发展到几乎深入到人类生产和生活的一切领域。计算机的应用领域还将随着计算机的进一步发展而扩大。可以预料到21世纪时，计算机将从目前基本上还是一种自成体系的设备，发展成一种到处都存在，但又看不见摸不着的“隐藏”在其他设备中的部件，使计算机完全融合在人类生活之中。

三、计算机硬件的发展——计算机的过去，现在与将来

软件离不开计算机这个硬件载体，所以，当我们来谈论计算机软件时，自然离不开计算机硬件本身的发展。

(1) 简单的计算工具及机械。在很久以前，人类进行计算是出自于现实的农业生产和简单的产品交换和分配的需要，为此人们需要记录某些数字和进行简单的计算。在公元前1000年前，人们通过在粘土上刻痕来记录数字。到了公元13世纪，中国人在算筹的基础上发明了算盘，通过一些串在轴上的算珠可以完成加、减、乘、除运算。可以说，这是世界上最早的较为成熟的计算工具。算盘一直沿用至今。而中国对算法的研究也早有记载，宋代数学家秦九韶算法是利用递归公式来计算多项式的值。

随着人类商业、制造业以及科学实验的发展，计算工作越来越频繁。在1612～1614年，苏格兰数学家John Napier（1550～1617）发明了一种帮助计算乘法的骨质拼条，它是一种辅助的计算工具。在1642年，法国科学家Blaise Pascal（1623～1662）发明了齿轮式加减法器，称为Pascalene。当时他曾制造了50台。在此之后，德国的Cottfried Wilhelm von leibniz、法国的Joseph Marie Jacpuard和Charles de Colmar都先后对机械式计算机进行了改进。1822年开始，英国剑桥大学Charies Babbage在穿孔卡织布机的启迪下，开始了分析机（analytical engine）的设计，首创了包括现代计算机都具有的五大装置：输入、处理、存储、控制和输出，开创了近代机械式计算机研究的先河。1854年，乔治·布尔出版了名著《思想法则的研究》。该书阐述了一种符号逻辑推理系统。即开创了逻辑代数——布尔代数（Boolean algebra）。在布尔代数中建立了两种逻辑值“0”、“1”以及三种运算“与（and）”、“或（or）”、“非（not）”，为现代电子计算机的发展奠定了数学基础。以后又相继出现了手摇式机械计算机和电动式机械计算机。

直到19世纪40年代，所有的机械式或其他形式的计算机都没有硬件和软件之分。所谓算法，或者说是计算原理都是通过机械的制造模型来确定的。如果想修改算法，那就只有重新制造一台新的机器了。好在当时计算机只用来进行固定而简单的计算工作，所以当时虽没有什么软件也可将就了。直到1945图灵在它的ENIAC报告中描述了存储程序在计算机上的应用。这样，从概念上，将用于算法的程序从硬件中分离出来，“程序”就成为了“软件”的雏形。

(2) 第一台电子计算机的诞生。世界上第一台真正意义上的电子计算机是诞生在1946年美国宾西法尼亚大学的摩尔学院。字长为12位，每秒可进行5000次加法运算，有18800个电子管，1500个继电器，占地150平方米，重30吨，耗电达150千瓦。从此，计算机技术以人类历史上空前的速度飞速发展。短短的50年，计算机发展速度之快，就连人类自己都不敢相信。

(3) 1946～1958年——电子管计算机时代。主要逻辑元件是电子管。形成存储程序计算机的架构。这类工作模式一直延续至今。此一时期的存储设备已发展为主要使用磁芯及磁鼓。

(4) 1958～1968年——晶体管计算机时代。主要逻辑元件是半导体晶体管。使计算机的体积大大缩小，出现了以屏幕监视器及键盘输入方式的商用计算机。

(5) 1968～1971 年——中小规模集成电路计算机。集成电路的发展引发了超级计算机的研究。例如，1968 年由 SEYMOUR CRAY 设计的 CDC7600 超级计算机的性能达到每秒 4000 万次浮点运算。

(6) 1971 年开始——大规模集成电路计算机。第一个微处理机（即完全放置于一个芯片上的计算机由 Intel 开发出来了。

(7) 1980 年——个人计算机 。网络开始迅速地扩大到人类生活各个领域。

(8) 1990 年——计算机、网络与通信高度接合所代表的信息高速公路。

(9) 2000 年以后。微处理机将向微体系结构（micro-architectures）多处理机芯片（CMP）发展。其性能大大突破由数十亿晶体管构成的单一处理机。按照摩尔法则：微处理器的性能每 18 个月将翻番。在芯片上的晶体管的数目每两年翻番。预计到 2010 年每个芯片上将有 800 亿个晶体管，时钟频率可达到 2GHz。一个芯片上的功率将达到 180W。芯片要进一步提高性能的主要限制可能将是散热问题。

四、软件、硬件与计算机应用之间的关系

从第一台电子计算机问世以满足计算方面的日益增长的需要之后，每次硬件技术的突破，都为软件技术新发展提供了更为广阔的空间，开拓了新的更广阔应用领域。计算机的应用领域从单纯的科学计算发展到军事、经济、科学、文化直到社会生活的各个方面；进而要求计算机的运算速度不断提高，存储容量不断扩大，体积微型化。而计算机数量的巨增，使软件系统从简单发展到复杂，从小型发展到大型，由封闭的自动化孤岛发展成为一种开放的系统。在开发方面：从注意技巧发展为注意管理；由单独设计发展为注意复用，由少数天才的编程艺术发展为广大用户直接参与开发应用。这种由应用驱动而相互促进、激励的发展过程，使人类社会文明进入了信息社会新的高度发展的阶段。

1.2 软件开发技术发展史

正如计算机硬件在 50 年内发生了极大变化，同样，对计算机软件的开发技术来说，在观念及目标等方面都发生了很大变化。我们可以大致以 10 年左右来划分软件开发技术的各个阶段。分析与讨论每个时期的特点，即每个时期的软件开发技术处理的对象、用途、目的、开发方法、认识、发展状况、突破、理论成就及目标。

一、40～50 年代

1. 软件开发技术处理的对象

软件开发技术处理的对象，以处理机的机器码占据主要地位。机器码一般由指令操作码及数据码组成。即是由“0”及“1”组成的序列。软件开发称为程序编写。由于每个计算机的指令系统都单独设计，没有规律，“0”与“1”组成序列的含义难于辨别，难于记忆，因此编写、阅读、调试程序都非常困难。

2. 用途

当时计算机的用途主要是科学计算或与军事有关的计算问题(如导弹表的计算)。世界上第一个电子计算机在宾西法尼亚大学摩尔学院运行几个月后，就拆迁到马里兰州阿伯丁武器试验场工作，直到1955年退役。

3. 目的

在这时期的计算机没有装入任何软件。即我们现在称之为“裸机”的机器。裸机只能识别二进制代码。程序人员编写机器能识别的机器码程序的目的是“确定计算机硬件动作的序列”。使计算机能自动地执行程序编写人员要求计算机完成的计算任务。

4. 开发方法

由于当时每台计算机都是单独设计，机器的指令系统没有规律，计算机价格昂贵，懂得计算机的人很少，会用计算机的人更少。人们认为，只有逻辑天才才有能力施展高技巧，写出让计算机完成计算任务的程序。普通人是不敢问津的，也根本没有想到需要开发方法。到50年代，美国总共也只有500名程序员。

5. 认识

认为计算机的用途就是快速计算，用途很窄。当时认为计算机的主要用途只要计算一些数据表就行，一旦计算完成之后，可将结果印发给全世界使用。所以全世界只需有5台计算机就足够了。

6. 发展状况

随着计算机体系结构成熟，商业上可以大批量地生产计算机，价格下降，应用范围很快扩大。特别，为了使大众及商业界能接受及理解计算机，人们在计算机应用方面作了许多尝试，以扩大计算机的影响。如：利用计算机预测美国总统选举结果；在日本，利用计算机进行照相机镜头的设计；甚至，开始设计交通管制系统等等。这些软件是用机器语言编写的，程序员将指令和数据直接写到计算机的存储体中。当程序员在原有的程序中加入一条新的指令的时候，程序员必须亲自对整个程序作检查，以确定所有相关的指令和操作经过变动后仍然是正确的。当这些应用变得更为复杂时，编写程序十分困难。渐渐地人们发现机器指令和存储体地址可以用一些便于记忆的符号来代替，为了摆脱用机器码编程的困难，使程序人员熟记计算机的全部指令，出现了用指令符号来编制程序的办法。用指令符号编制的程序称为符号程序，在编制程序时，只要记住用英文名称缩写的指令助记符就可以了。例如，取数用LDA、加法用ADD等。在符号语言的基础上，进一步发展就是出现了汇编语言。用汇编语言编制程序要比用机器的指令代码编写方便得多，大大便于检查和修改错误。而且，指令、原始数据和结果数据的存放单元可以由机器自动分配。

然而，计算机的内部结构是根据指令代码设计的。也就是说，它只能识别和理解用二进制代码表示的指令，不能识别和理解指令助记符。因此，人们用汇编语言编出程序

后，必须将此程序翻译为机器语言程序，称为目的程序（目标程序），机器才能执行。这个翻译工作十分繁琐，几乎完全是机械式的一一对应的翻译。因此，人们开发出专门的程序来完成机械式的翻译。这种程序称“汇编程序”。汇编程序是一种进行“编辑与翻译”的程序，计算机有了它，才允许用户在该计算机上使用汇编语言编制程序，称为汇编语言程序或汇编源程序。汇编程序是当时计算机必不可少的软件。

用汇编语言写的程序和用机器语言写的程序有相同之处，又有不同之处。相同之处在于程序主体部分几乎是一一对应的，不同的是0、1数码换成了符号，地址换成了可读的名字，另外还增加了关于工作单元和常数单元的成分。这些不同之处也正是汇编语言的优点，使得用汇编语言编写的程序好写、好读、好改。

由于汇编是依赖于机器的，因此称它为面向机器的语言。使用时必须了解机器的某些细节，如累加器的个数、每条指令的执行速度、内存容量等等。但也正由于它依赖于机器就可以与机器语言程序一样结合机器特点编出短小、高质量、执行速度快的程序。所以，时至今日，汇编语言仍起着重要作用。在一些计算机公司中仍用汇编语言编写系统软件，以保证高质量软件的功效。

机器码书写冗长且易出错，而汇编语言有一定改进，但仍依赖于机器，仍为“面向机器的语言”，使用时需了解机器的某些细节，从而使计算机的应用普及以及普通人学习计算机产生了巨大的困难。随着技术的增长以及程序开发任务的复杂化，单靠机器码与汇编语言已越来越难以适应程序开发的需要。同时，程序内部的控制日趋复杂。到50年代后期，人们发现按照某种规则来组织描述程序的助记符将会有助于程序的理解。例如使用一系列数学符号来表示数学运算等措施比单纯的汇编程序要容易理解。这种想法导致了一系列早期的高级语言的出现，其中包括现在我们仍在使用的Fortran语言。高级语言的出现和数据类型的使用使得编制复杂的软件成为可能。因此，面向应用的高级语言及相应编译系统的研究成为重要发展方向。

7. 重要技术突破

• 对时间-空间关系的认识有了提高，即认识到：可通过使用便宜的存储器来代替计算机硬件的逻辑功能。在电子计算机出现之前，人们设计专用的硬件来解决特定的问题，大部分问题都是通过直接的方式来实现的。后来，发现可以通过组合方式来处理许多问题。因此，经费决定了所能解决问题的复杂程度及规模。当时许多人认为只要有足够的钱来构造硬件，就可解决几乎任何问题。而有了计算机，带来了各种层次的存储设施，如磁芯、磁鼓、磁带、磁盘、纸带、穿空卡片等等。其价格又便宜，数量又充足。通过把信息放在存储介质（空间资源）上，再多次利用，这样就代替了部分昂贵的硬件逻辑功能。

• 迭代-反复使用的子程序的利用。从此，计算机解决问题的方式不仅仅是意味着可以利用各种功能的组合，而且还可以通过反复利用子程序，来解决规模巨大的问题。而如果只依靠过去那种硬件组合的方式，想解决这类问题是无论如何无法承受相应的巨大经济压力的。

• 因此，人们思想观念发生了巨大变化。从只要有“足够的资金”就能解决困难而复杂的问题转变为只要能容忍时间上的“等待”就可以解决几乎任何问题。从此，为了

减少“等待”时间，人们努力的方向就是不断突破计算机的运算速度及存储容量的限制。

8. 理论成就

冯·诺依曼提出了存储程序的概念。图灵进一步提出了存储程序在计算机中的应用。程序员可以不必知道计算机的细节，就可以用程序来实现某些运算。他也预言了高级语言的开发，甚至在当时就提出“利用电话线路来控制远距离计算机是完全可能的”。

图灵提出的自动装置的计算模型——图灵抽象机模型讨论了图灵机的停机问题，成为现代计算机理论的基础。他还指出如果完全无法区别一台机器对于询问的响应与人类响应有什么不同，那么这台机器就具有智能。这一论断称为图灵测试，从而奠定了人工智能的理论基础。

9. 目标

这一时期工作目标是：要求编出的程序能执行，执行快，最后能产生结果，其结果可接受。程序的质量完全依赖于程序员个人的技巧。希望用最少资源来获得最大运算能力。

二、60 年代

1. 软件开发技术处理的对象

不再直接用机器码编程，主要是使用各种符号语言来编程，包括面向机器语言的汇编语言，面向应用的高级语言。高级语言可以独立于机器。高级语言是按一定的“语法规则”，由表达各种意义的“词”或“数学公式”组成。这种程序设计方法比较接近人的习惯，人们不必考虑计算机内部的构造和不同机器的特点，只要考虑解题步骤，写出程序，经过编译程序翻译成机器能执行的机器指令，计算机就能执行，从而给编程工作提供极大方便，提高和扩大了解决问题的规模及难度。

2. 用途

计算机应用不再局限于计算问题，应用范围大大扩大。计算机大规模进入商业，银行等领域。开发了一批规模相当大的系统，如 IBM 公司用了 6000 人年开发了操作系统 OS 360，费用高达 1 亿美元。美国航空订座系统经历了长达 7 年的开发周期，于 1964 年投入使用。著名的美国半自动化防空系统 SAGE 由 112 万行的后援指令和 10 万行执行指令组成。

3. 目的

这一时期编程的目的不再是关心计算机硬件的动作，而是要确定程序人员定义的动作序列。这种动作不是代表计算机的一条指令，而是代表了一系列的计算机指令的执行。即程序员控制的动作粒度可大大增加。它们由高级程序设计语言的语句，或宏指令，宏语句等等所表示，往往与计算机的内部结构和指令系统无关。它们更直接地接近人们所要处理的问题，因此解决问题的规模与复杂性大为增加。

4. 开发方法

我们可以用“功能性程序设计”一词来表征这一时期开发技术的特点。针对特定问题，根据所需功能，制定特定方法。甚至要考虑是否需要制造特定机器去解决该问题。每种方案要设计出特定的程序符号信息与结构。编程无章法，类似于智力游戏，依赖于才智与技巧。检查错误的手段是仅仅依赖于输出的全部存储信息（dump）的分析。缺乏软件开发技术方法与理论。编程的随意性很大，一个人写的程序，另一个人很难看懂或理解。为了扩大待解决问题的规模和复杂性，开始把一组小程序链接成大程序，以完成单一综合的系统功能。当问题变得复杂，系统变大时，这种完全靠个人想象力构成的系统中所产生的问题就很难解决。

5. 认识

高级编程语言的发展，使程序的编写与执行程序的计算机硬件无关。因而开始认识到软件应独立于硬件。

为了扩大应用规模，必定会受程序的规模与复杂性的限制。复杂性直接与程序的控制流有关。从此，开始认识到程序内部控制流的影响是不可忽视的。正因为子程序的返回控制和程序内部不规则而多变的控制流，使程序变得复杂而难于理解。因此一些计算机科学家提出必须规范这种“控制流”。但是，在这一时期对软件开发中应重视的“管理问题”尚无认识。

6. 发展状况

这一阶段主要的发展是助记忆符以及适用于各种应用目的的高级编程语言，如FORTRAN，COBOL，ALGOL，BASIC，SIMULA 及 LISP 等等。它们对计算机硬件有更大的控制粒度。高级编程语言与计算机独立，使编程工作可以脱离对计算机本身的结构或指令系统的认识而独立进行。进而使计算机应用的编程工作不必只能由专职的程序设计人员担任。大大促进了计算机应用的扩大。同时，为了使计算机能理解高级语言，发展编译系统技术是极为必要的。计算机的非数值计算方面的能力也随之得到愈来愈大的认同。非数值的商业应用得到极大发展。但开发这些应用时，仅停留在使用程序内部精确注释及散文式的规格说明方式来联系软件的用户与开发人员。大型应用系统的开发情况所呈现的“软件危机”出现了。

7. 突破

计算机体积变小，商业化生产成熟，而价格下降。计算机应用得到极大发展。这一阶段的成就就是通过各种目的的高级编程语言促进了计算机的应用。如 FORTRAN 语言促进了大型计算性的应用。COBOL 语言促进了银行，保险业等行业的计算机的应用。LISP 语言非常有利于表格处理，进而使通过表之间的操作进行符号演算，搏弈等人工智能应用得到发展。这一阶段的软件开发方式可归纳为“功能性程序设计技术”。

8. 理论成就

在1968年10月北大西洋公约组织(NATO)的科学委员会提出了软件危机问题：对于大型软件开发中存在的价格高，开发不易控制，软件开发工作量估计困难，软件质量低，软件项目失败率高，错误率高，无法判断大型系统能否正常工作及软件维护任务重等现象，归结为“软件危机”，认为这将是影响计算机应用发展的瓶颈，从而提出“软件工程”问题。“软件危机”是软件开发的技术落后与软件需求的极大增长这一矛盾的必然产物。软件开发工作通常仅仅依靠单个个人的努力。对于较大的软件系统，单个开发人员就无能为力了，他们通常不能保证软件的质量和开发的成功。然而，失败的系统往往无可挽回，除非再从头做起。但由于时间和经费的限制，又使这成为不可能。在60年代，“软件危机”给软件行业带来了巨大的冲击，常常是大量的人力、物力投入，最终换来的却是失败。人们感到非常沮丧甚至有些恐惧。在IBM公司的OS 360系统之后，有人如此描述当时的困难和混乱：……像巨兽在泥潭中垂死挣扎，挣扎越猛，泥浆就沾得越多，最后没有一个野兽能逃脱淹没在泥潭中的命运……一批批程序员在泥潭中挣扎……没人料到问题竟会这样棘手……。

与此同时，计算机硬件已经从原来的电子管发展到晶体管，以至进入了集成电路时代，硬件性能越来越好，而价格却大幅下降。同硬件相比，软件投资比例上升极快，这就使得软件开发成为计算机技术发展和普及的瓶颈，人们开始意识到了软件开发技术革新的急迫性和重要性。

许多有识之士开始认真研究“软件危机”背后的真正原因，得出了许多具有重要意义的结论。在客观上，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。软件开发实质上是逻辑思维的过程，在写出程序并拿到计算机上运行之前，软件开发的进展情况难于衡量，质量也难于评价，因此管理软件开发过程十分困难。同时，软件规模和复杂度成指数巨增。成百上千人共同开发一个大型系统时，大量的通信、后勤工作成为问题。这常常是造成软件开发的失败多，费用高的重要原因。人们面临的不光是技术问题，更重要的是管理问题。管理不善必然导致失败。另外，软件不能像硬件那样允许误差的存在，软件也不会用坏，发生错误时，只能是在生产现场改正或修改原来的设计。大型系统在开发时，无法看出是否能正常工作，错误率高。为了了解开发中的情况，就要求程序的可读性高，而不再强调编程技巧。为解决软件危机，荷兰科学家E. W. Dijkstra指出：不应该简单地只考虑编写程序，就期望产生一个正确的结果；而应考虑如何把软件进行划分及构造。他在编写操作系统时提出了分层次结构的想法，并付之实现。他提出取消GOTO语句，以提高程序的可读性为目标的结构化程序设计方法。指出：可以从高级语言中取消GOTO语句……程序的质量与程序中所包含的GOTO语句的数量成正比……后来，Jacopini等人证明了只要“顺序”、“选择”与“循环”三种基本控制结构就可以实现任何单入口，单出口的程序——程序结构定理。这就是结构化程序设计的理论基础。只要使用三种基本控制结构的高级程序设计语言，以及只有一个入口及一个出口的程序设计原则，共同形成了新的程序设计思想、方法和风格。使程序变得清晰、易读、易修改——这就是结构化编程方法。

9. 目标

这一时期程序开发技术的目标是如何扩大程序系统的规模，以适应更复杂的应用。程序设计人员不再是计算机发展的早期那样为自己的研究工作需要而编制程序，而是为了他人——用户更好地利用计算机的功能而编制程序。复杂的应用使程序规模变得极为庞大，需要组织大量程序员共同工作。各种人员之间的通信、交流及后勤工作变得极为复杂。为了减少失败，降低费用，减少错误，增加成功率，人们意识到大型软件系统的开发工作与早已为人们熟知的建设工程极为类似。于是，人们就尝试把已经成熟的工程学运用到软件开发上。以“软件工程”——强调组织管理的软件开发方法来解决“软件危机”。这些方法的特点都是强调开发的可见性来支持开发管理。

三、70 年代

1. 软件开发技术处理的对象

“程序设计＝数据结构＋算法”这一公式总结了 70 年代程序设计的特征与成就。这一时期基本上可以用“小规模系统的程序设计”（programming in the small）为标志。

2. 用途

这一时期计算机应用主要是大量的各种类型的非数值计算为特征的商业事务应用。计算机应用差不多涉及了各种应用领域，并涉及到大量智能性很强的领域，如能自动巡航的机器人，各种字处理系统，脑瘤病人 X 片的图像识别，电话呼叫的语音生成，象棋比赛，图形界面等等。并开启了计算机网络互联的技术，如电子邮件等。同时，作为应用系统开发的基础设施，如编程语言的编译系统，简化应用系统对计算机资源的“调度与管理”的操作系统及针对数据管理的文件系统，数据库系统等都得到大力发展。

3. 目的

程序员的主要精力集中在选择及发展（确定）主要数据结构及相应算法。达到应用对计算机资源中存储空间/CPU 运行时间间的平衡。

4. 开发方法

这一时期，软件开发技术主要有两大发展。

第一方面是从程序中分离出数据结构与算法。美国科学家 Knuth 在 1971 年发表的三卷“计算机程序设计技巧”已成为数据结构与算法方面的经典著作。数据结构与算法代表了在一些应用系统中反复出现的编程问题的优秀编程方案的经验总结。Knuth 把数据结构与算法问题冠以计算机程序设计技巧是十分恰当的。它们代表了计算机程序设计的制高点。把具有代表性的编程问题从应用程序中分离出来，成为一种独立的研究对象后，就可以集中精力对其进行研究并加以改进，并形成与其有关的理论。利用数据结构与算法的理论，可以优化计算机中的两项重要资源（存储空间与运算控制设备）的利用。

第二方面是进一步把结构化程序设计方法发展成结构化开发方法，包括结构化分析

方法及结构化设计方法。

5. 认识

对计算机程序内部各部分关系的认识有进一步提高：

• 关于空间——时间复杂度的平衡方面：提出了算法复杂性及其度量问题，并提出了 NP 问题，即对于有些问题，当问题规模扩大时，会造成非多项式的计算增长。

• 程序执行的停机问题——由于当时的主要矛盾是在程序执行时，如果发生不结束的错误，就很难办。因此，程序运行后的停机问题十分重要。

• 虽认为软件开发主要靠个人努力，但对复杂问题已认识到应强调程序清晰，而不是过分的技巧。

6. 发展状况

为了解决大型系统的问题，发展了数据独立的概念，即数据（以文件或数据库形式）可脱离系统而存在。亦发展了并发访问的程序执行机制。从此，对软件系统来说，人们不仅要考虑系统的功能，而且开始认识到状态的重要性。软件系统中包含的少量必要的状态，需要新理论方法。换句话说，系统状态比系统终止条件显得更为重要。对大型系统来说，重要的不仅是源程序或可执行程序，同样重要的是复杂的系统规格说明。这种规格说明中包括了系统的功能、性能、可靠性及输入/输出要求等等。

7. 突破

• 数据结构与算法从程序中分离使人们更加容易学习编程经验，不必重新发明创造已有的数据结构和算法。基本解决了编程的方法和逻辑技巧问题。因此，能使用计算机的人群大为扩大。

• 发展了各种程序设计语言，如 PASCAL（不用 GOTO 语句），SMALLTALK（面向对象语言），PROLOG（逻辑语言）……

• 形成系统软件与应用软件的区别。相应的系统软件，有各种编程语言的编译器，操作系统（如 UNIX）及数据库系统等。

• 形成完整的软件系统：不仅是可执行系统，还有独立的数据状态和程序系统的规格说明书。

8. 理论成就

主要涉及三方面：

• 数据结构，算法理论。

• 形式方法——用推理及逻辑断言等对程序正确性进行验证。

• 软件工程方法：开始提出软件开发模型——瀑布模型及相应的结构化分析、设计、编程及测试方法。其中最为成功的是 DeMarco，Yourdon，Jackson 等人提出的分析型或构造型方法及 Parnas 提出的以信息隐藏为特征的抽象数据结构、抽象机等等。