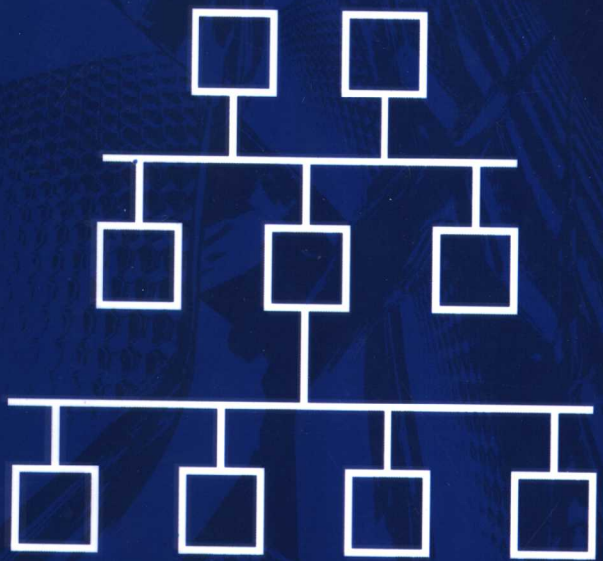
21 世纪高等学校规划教材

软件体系结构

刘 真 编著



中国电力出版社
www.infopower.com.cn

21 世纪高等学校规划教材

软件体系结构

刘 真 编著

中国电力出版社
www.infopower.com.cn

内容提要

本书系统地介绍了软件体系结构的基本概念、主要构成及有关理论和方法。在此基础上，着重介绍了目前广泛应用的几种软件体系结构的风格和模式，并深入分析了它们的系统结构、功能和非功能特性以及它们的设计实现。全书共分 6 章。第 1 章主要讨论软件开发的特点和问题。第 2 章主要介绍软件体系结构的概念和研究的重要性、软件体系结构的风格和模式。第 3 章介绍 10 种系统级体系结构风格模式。第 4 章介绍中等规模的设计模式。第 5 章介绍基础结构模式。第 6 章简要介绍软件体系结构的其他研究领域，如软件体系结构的描述语言、软件体系结构的形式化及软件体系结构的工具环境等。

本书适合高等院校高年级学生和研究生使用，也可作为从事软件工程、软件设计开发、软件应用及软件体系结构研究人员的参考书。

图书在版编目（CIP）数据

软件体系结构 / 刘真编著. —北京：中国电力出版社，2004.8

21 世纪高等学校规划教材

ISBN 7-5083-2292-4

I.软... II.刘... III.软件－系统结构－高等学校－教材 IV.TP311.5

中国版本图书馆 CIP 数据核字（2004）第 079195 号

责任编辑：李萌

丛 书 名：21 世纪高等学校规划教材

书　　名：软件体系结构

出版发行：中国电力出版社

地址：北京市三里河路 6 号　　邮政编码：100044

电话：（010）88515918　　传　　真：（010）88518169

本书如有印装质量问题，我社负责退换

印　　刷：北京丰源印刷厂

开本尺寸：185×233　　**印　　张**：13　　**字　　数**：282 千字

书　　号：ISBN 7-5083-2292-4

版　　次：2004 年 9 月北京第 1 版

印　　次：2004 年 9 月第 1 次印刷

印　　数：0001—3000 册

定　　价：20.00 元

前　言

随着计算机和网络应用的迅速普及和发展，软件已成为影响网络和计算机发展的重要因素。目前，软件业受到世界各国前所未有的关注。软件的开发和研制能力已成为一个国家科技、经济和国防实力的重要标志。

由于计算机和网络的发展，软件的规模不断扩大。软件的生产成本也随着软件的复杂程度而急剧增加。软件是人类抽象思维的产物，它的复杂性给它的大规模发展和维护带来了困难，从而使软件技术的发展远远落后于软件的需求。但是人们在开发研制软件的长期实践中也积累总结了许多成功的经验。如果能坚持不懈地总结和使用这些经过实践检验的方法和经验，软件业是可以快速健康发展的。

软件体系结构是软件设计过程中的一个层次。这个新的层次超越了计算过程中的算法设计和数据结构设计。良好的体系结构设计是决定软件系统成功的必要因素。它为软件项目的长期稳定性、可靠性提供了保证。软件体系结构作为软件设计的一个高层次，为软件工作者提供了一个重要的规范，为他们提供更好的方法去分析和理解更大、更复杂的软件系统。

软件体系结构是随着描述大型复杂系统结构的需要而兴起并发展的。为了避免大型软件开发的盲目性，提高软件的质量，在软件开发中需要有效的理论作为指导。软件体系结构作为软件开发中的设计指导思想，对软件开发的成败起着至关重要的作用。

虽然软件体系结构在软件工程中已有了很广泛的应用，但是由于有关的研究还刚刚起步，因而对它的理解还没有达到共识。许多研究人员基于自己的经验从不同角度、不同侧面对软件体系结构进行了不同的刻画和定义。不同的认识反映了软件体系结构概念的广度和复杂性。

软件体系结构研究中很重要的成就之一就是抽象出许多常见的系统构建模式，形成了软件体系结构的某些风格。这些都是软件设计人员多年工作的经验积累。他们在长期工作中摸索到一些规律性的东西，经过提炼总结，得到具有普遍意义的构建模式和构建风格。使用软件体系结构的风格和模式对软件开发具有重要的应用价值和经济效益。本书将着重介绍目前应用较广泛的软件体系结构的几种风格和模式，讨论它们的系统结构、功能特性、应用范围和优缺点等。

本书共分 6 章。第 1 章主要讨论软件开发的特点和问题。第 2 章主要介绍软件体系结构的概念和研究的重要性、软件体系结构的风格和模式。第 3 章介绍 10 种软件系统级体系结构风格模式。第 4 章介绍中等规模的设计模式。第 5 章介绍基础结构模式。第 6 章简要介绍软件体系结构的其他研究领域，如软件体系结构的描述语言、软件体系结构的形式化及软件体系结构的工具环境等。

本书在资料收集和编写过程中得到许多同事和研究生的帮助。王焱同志参加了本书第 3、4 章的校核和绘图工作。2000 级研究生参加了部分调试工作。

由于软件体系结构是一个新课题，加上作者水平有限，书中错误在所难免，希望大家批评指正。

编著者

目　录

第1章 软件危机

计算机系统由计算机硬件系统和计算机软件系统两大部分组成。随着计算机和网络应用日益普及和深化，对于软件在计算机系统和网络中所占据的重要地位已经很少有人怀疑了。现在，正在不断投入使用的计算机和网络软件的数量在急速地增长，而且现代计算机和网络软件的规模也越来越大。耗资几亿美元，有几百万行代码，耗费几千人年编制的程序已并不罕见。

随着电子工业技术的发展，计算机硬件的性能/价格比以每10年升高两个数量级的速度在飞速发展，而且质量十分稳定。与此同时，计算机软件的成本却在逐年升高，并且质量没有确定的保证。软件开发的速度也越来越赶不上网络和计算机应用的需要。软件已经成为影响网络和计算机发展的一个重要因素。目前，软件业受到世界各国前所未有的关注。各国政府都在竞相发展本国的软件业。软件的开发和研制能力已成为一个国家科技、经济和国防实力的重要标志。

1.1 软　件

软件是相对硬件而言的。网络和计算机硬件是指组成网络和计算机的具体物理实体，是看得见、摸得着的几何组件。而软件却是一种非物理实体，它是一种逻辑实体。写出的程序代码在计算机上运行之前，人们很难感觉到它的存在。

最初，人们认为软件是为每个具体应用而开发的小程序，应用者为完成一个任务而编写和执行一段程序。由于这种个体化的编制和使用过程，使软件的设计成为一个脑力劳动的隐含过程，除了程序清单，没有任何其他的资料保存下来。

但是随着软件的日益发展，软件的功能规模不断增长，人们为了软件的设计合作和维护的需要，还需要更多的文档说明，数据描述等要求也日益增长。人们逐渐认识到程序只是完整的软件产品的一个组成部分，而不是全部。B. Boekm 曾说过："计算机程序，开发使用和维护此程序的一切文档都是软件。"因此，软件是一个包括程序、数据和相关文档的完整集合。

软件具有以下特点：

（1）软件是一个逻辑，而不是一个具体的物理实体，人们可以把它记录在纸面上，存储在计算机内存储器中或保存在磁盘或光盘上，但是却无法看到软件的具体形态。人们只能通过观察它的执行行为来分析、判断、思考、了解它的功能及表现等，但却看不见、摸不着软件本身。

（2）软件是人类思维抽象的产物，因而，软件是十分复杂的。有人认为，计算机软件是人类能够创造的最复杂的产物。软件的复杂性来自它反映的现实问题的复杂性。因为软件要反映的自然规律或人类社会的事物都是十分复杂的。人类社会是一个不断进化的复杂多变的环境。而人们越来越多地要求计算机软件能够反映和处理各种可能的情况，从而使得许多计算机软件变得十分庞大而复杂。软件的开发，尤其是应用软件的开发常常会涉及到许多特定领域的专门

知识和特殊的需求。这就对软件开发人员提出了很高的要求。这也是软件技术的发展远远落后于软件需求的原因。

(3) 软件的开发过程与许多工业产品的生产过程不同。例如我们可以看到硬件产品的生产过程：新产品一旦研制成功就可以重复生产，它们的质量是在制作过程中控制的。而软件开发过程是许多开发人员智力劳动的结晶，因此，与硬件生产不同，软件生产没有明显的制作过程。它是把人类的知识和技术转化为信息的一种产品，在软件的开发阶段中要注意加强软件的质量控制。

(4) 软件的生产成本随着软件的复杂程度急剧增加。软件的研制工作需要投入大量的高强度的脑力劳动，而且软件开发人员的培训也是一笔相当大的开支。目前的软件规模越来越庞大，如美国 IV 代宇宙飞船的软件规模呈指数型增长，因而同一项目中硬软件成本之比发生了根本的变化。在 20 世纪 50 年代，软件的成本仅占总成本的百分之十几，而到 20 世纪 80 年代，由于硬件成本的下降，及软件复杂度的增加，软件的成本大大超过了硬件。而且，软件的投入具有很大的风险性。

(5) 软件是非常容易进行复制和传播的。软件一旦研制成功，可以大量复制同一内容的副本。为了使软件开发这一复杂劳动得到承认和尊重，维护软件开发公司和人员的合法权益，必须在技术上和法律上对软件产品进行保护。对任意复制、盗版软件的行为进行有效的防范和打击，才能促进和维护软件业的正常、规范的发展。

(6) 软件的维护——软件的运行与使用硬件设备的情况不同，它没有机械磨损和电气老化问题，但是软件的维护却是一个不可忽视的问题。在软件的生存期间，不仅需要不断改正使用过程中发现的每个错误，而且需要不断地修改软件来适应硬件、软件环境的变化以及用户新的要求；而每次修改又可能引入新的错误，又需要修改。软件的维护工作是相当复杂的，需要花费很大的代价。

(7) 软件的开发和运行常常受到计算机硬件环境的限制，对计算机系统有着不同程度的依赖性。软件不能完全摆脱硬件独立活动，在开发和运行软件过程中必须以可提供的硬件为依据。有些专用的计算机软件，完全依赖于某种计算机环境，对软件的使用和推广很不利。有些软件依赖于某种操作系统，如有的软件只可运行于 Windows 98，而到 Windows XP 即不能运行。为了解决这种依赖性，软件开发中把可移植性作为一个衡量软件的重要指标之一。这对软件的商品化、市场化都是十分重要的。

(8) 软件开发过程中传统的手工艺开发方法仍然占据统治地位，因而软件开发的效率很低。尽管目前软件工程和软件技术取得较大进展，提出了许多新的开发方法，如充分利用现有软件的自动化生成技术、软件复用技术，研制了一些软件开发工具或开发环境。但是它们在目前软件项目开发中占有的比例仍然是很低的。

1.2 软件的发展阶段

“程序存储”仍是目前计算机最基本的设计原理之一。从世界上第一台计算机诞生以来，就有了程序的概念。程序是软件的前身。经历了计算机应用几十年的发展，人们对软件有了越

来越深入的认识。在这几十年的过程中，计算机软件经历了 3 个发展阶段。

1. 程序设计阶段

20 世纪 50 年代到 60 年代中期，是计算机系统发展的早期阶段。这时，通用硬件相当普遍，而软件却是为每一个具体应用而编写的，通常是规模较小的程序。当时认为，写出的程序只要能在计算机上得出正确的结果，程序的写法可以不受任何拘束，而且为了节省内存和 CPU 时间等硬件资源，人们使用了许多程序技巧。由于程序设计中人们任意发挥创造性的技巧，写成的程序往往很难被别人看懂。但是随着计算机应用的逐渐普及，人们逐渐改变了对程序设计的看法。因为对于规模较大、在较长时间内供许多人使用的程序，人们就要求这些程序容易被看懂，便于使用，而且容易被修改和扩充。因此要求程序设计按照一定的方法并具有足够的注释，同时由于硬件资源价格的下降，许多技巧逐渐失去了价值。由此程序便从创造者的“艺术品”转变为广大用户接受的“工业产品”。

2. 程序系统阶段

20 世纪 60 年代中期到 70 年代中期，这个时期的重要特征是出现了“软件作坊”。此时的软件需求成为软件发展的动力。软件的开发成果逐渐具有了社会属性。软件逐渐成为一种商品。软件业也逐渐发展成为一个新兴的产业。在软件产品市场化的发展协作中，软件工作的范围从只考虑程序的编写扩展到涉及整个软件的生存期。此时由于计算机硬件的发展，计算机的速度、容量和可靠性明显地提高，硬件的成本很快地下降。计算机价格的降低为计算机的迅速普及提供了极好的条件，因而对软件的需求也急剧增长，因此一些复杂、大型的软件项目也应运而生。但是，软件的开发仍然沿用个体化软件的开发方法，软件技术已无法满足形势发展对它提出的要求。首先程序运行时发现的错误需要及时改正，用户提出新的需求时需要相应地修改程序；硬件或操作系统更新时，程序需要适应新的硬件环境，软件的维护工作大幅度地增加，甚至某些遇到的问题找不到解决的办法，必须重新进行软件设计，这形成了非常尖锐的矛盾。“软件危机”就这样开始了。1968 年北大西洋公约组织的计算机专家在德国召开国际会议，讨论“软件危机”问题，并第一次提出了“软件工程”的概念。从此软件设计进入了第三个阶段，即“软件工程”阶段。

3. 软件工程阶段

20 世纪 70 年代中期以后软件的规模逐渐扩大。软件开发的过程，都是由多个软件人员分工合作，共同完成的。开发过程中，各个人员之间的工作应很好地衔接。协调开发完成后，软件成果要面向用户，在实践中接受用户的检验。因此软件开发过程不是某种个体劳动的神秘技巧，而应是一种组织良好，管理严密，各类人员协同配合，共同完成的工程项目。它应该充分吸收和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法。经过实践，人们认识到按照工程化的原则和方法组织软件开发工作是十分有效的。

软件工程方法是指导计算机软件开发和维护的工程方法。它采用工程的概念和方法来开发与维护软件。把经过实践证明的管理技术和当前可以达到的最好技术方法结合起来，就是软件工程方法。它是从管理和技术两个方面研究如何更好地开发和维护计算机和网络软件的一门技

术和方法。

自计算机专家国际会议提出软件工程概念以后，人们陆续提出了100多条关于软件工程的准则。著名软件工程专家B. Boekm综合了专家们的意见，于1983年提出了软件工程的7条基本原理。它们是：

（1）用分阶段的生命周期计划严格管理；

（2）坚持进行阶段评审；

（3）采用现代程序设计技术；

（4）实行严格的产品控制；

（5）软件产品应能清楚地审查；

（6）软件开发人员应该少而精；

（7）承认不断改进软件工程实践的必要性。

1.3 软件危机

什么是软件危机？软件危机是指在计算机软件开发和维护过程中所遇到的一系列严重问题。这些问题在几乎所有的软件上都不同程度地存在着，主要有下述表现。

1. 对软件开发的成本和时间估计常常不准确

随着软件规模的逐步增大，软件开发成本和时间往往呈指数曲线上升，这样，软件开发的实际成本往往高出估计成本一个数量级，实际进度比预定进度拖延数月至数年。进度计划根本无法执行。为了加快进度和节约成本所采取的一些权宜之计往往又损害了软件产品的质量，这些现象严重损害了软件开发组织的信誉，引起了软件投资者和用户的不满。

2. 用户对完成的软件产品常常不满意

作为软件设计依据的用户需求，在软件开发初期用户很难准确具体地进行叙述和表达，软件开发人员常常对用户的要求只有模糊的了解，对要解决的问题缺乏全面认识。而在软件开发工作开始之后，软件人员与用户之间又未能及时交换意见，因而，完成的软件产品常常不符合用户的实际需要。

3. 软件的质量常常不可靠

由于缺乏标准的软件评测手段，软件的测试工作往往不能顺利进行。软件的质量保证技术没有坚持应用到软件开发的全过程中，因而提交给用户的软件产品常发生质量问题。在应用领域工作的不可靠的软件，给系统的正常工作带来极大危害。有些软件中的错误是非常难以改正的。许多软件不能适应新的硬软件环境，也不能跟随用户的需要追加新的功能，即许多软件常是不可维护的。

4. 软件开发常常缺乏适当的文档资料

计算机软件开发过程中需要统一的、公认的方法与规范作为指导，尤其在大型软件开发项

目中，不仅仅需要产生程序清单，而且需要一整套文档资料。这些文档资料随着软件开发过程的进行，不断地产生和更新。软件开发人员可依赖这些文档作为交流的工具。软件开发的管理人员可依据此文档资料来管理和控制软件开发的进展状况。对于软件维护人员而言，这些文档资料更是至关重要的依据。缺乏适当的、合格的文档资料，必然给软件开发和维护带来严重的困难。

5. 软件技术发展的速度远远赶不上形势的需要

由于电子技术的发展，计算机硬件的成本逐年下降，计算机和网络的应用迅速普及。但是，软件的开发和维护需要大量的人力，软件成本随着软件应用规模和数量的扩大而不断扩大。在20世纪90年代，美国软件的成本已占计算机系统总成本的90%，[张海藩 P2]这极大地限制了计算机和网络应用的深入发展。另一方面，软件产品供不应求的现象也限制了人们对于充分利用计算机和网络巨大潜力的需要。

软件危机产生的原因，除了与软件本身的特点有关，更重要的是与软件开发和维护方法不正确有关。

在软件开发与维护中长期存在着某些糊涂观念，在软件开发过程中仍采用着一些错误的方法和技术，其主要原因是在软件开发早期阶段的个体化特点所形成的。其主要表现为忽视软件需求分析的重要性，认为软件开发就是写程序并设法使之运行；在软件开发过程中既无计划、无规范，在软件产品产出后，又无评测和维护手段等。

从软件危机的现象和发生危机的原因可以看出，摆脱软件危机不是一件简单的事，不能只从一两个方面着手解决，也并非一朝一夕能够做到。在不断改进过程中，应排除人们的一些传统观念和某些错误认识。只有端正了认识，真正抓住软件的特点和发展趋势，才能使软件发展走上正确的道路。

软件本身的特点确实给开发和维护带来一些困难，但是人们在开发和使用计算机系统的长期实践中，也确实积累和总结了许多成功的经验。如果能坚持总结并使用经过实践检验证明是正确的方法和经验，软件危机是完全可以克服的。

1.4 软件生命周期

如同任何事物一样，软件也有孕育、诞生、成长、衰亡的生存过程，这称为软件的生命周期。如果按照软件的生命周期的时间顺序把软件的开发和维护的许多问题进行分解，将它们划分成若干个阶段。每个阶段都有其相对独立的任务。前一个阶段任务的完成是后一阶段工作开始的前提和基础，而后一阶段任务的完成是使前一阶段提出的问题给予解决的保证。将软件的生命周期划分为相对独立的若干个阶段，每个阶段的任务相对独立，大大降低了整个软件开发工程的困难程度，也有利于不同人员的分工合作。同时，在生命周期的每个阶段都采用科学的管理技术和良好的技术方法，在每个阶段结束前都从技术和管理两方面严把质量关，从而保证了整个软件的质量，提高了软件的可维护性。

软件生命周期的划分方法有许多种，但各种方法都应遵守一条基本原则：即各阶段的任务

应尽可能地相对独立，从而降低各阶段任务的复杂程度，简化各不同阶段之间的联系，有利于软件开发工程的组织管理。

一般软件的生命周期分为软件定义、软件开发和软件维护3个基本时期，而按上述基本时期的活动展开，又可得到软件生命周期的6个主要步骤，它们是：制定计划、需求分析、系统设计、程序编码、系统测试以及软件的运行维护。

1. 制定计划

制定计划阶段分为问题分析和可行性研究两个阶段。首先根据用户的要求，系统分析员对于要解决的问题的性质、工程的目标和规模进行研究。通过与用户或使用单位负责人的访问调查，系统分析员扼要地写出问题的要求、目标和规模的书面报告。系统分析员与用户和负责人认真讨论该报告，澄清含混不清之处，改正理解不正确之处，最后得出双方均认可的报告。

第二阶段进行可行性研究，系统分析员与用户合作，研究该项软件任务的可行性，探讨解决该问题的可能方案。可行性研究阶段应该导出建议系统的高层逻辑模型（如数据流图），并且在此基础上更准确、更具体地确定工程的规模和目标。然后，系统分析员对建议系统进行仔细的成本和效益估算。通过对可利用资源（硬件、软件、人力等）的开发，对进度成本和效益作出估计，作出可行性报告，提交管理部门审查。可行性研究的结果是使用单位负责人决定是否实行此项工程的主要依据。一般情况只有可能取得较大效益的系统才值得继续。

2. 需求分析

该阶段主要确定目标系统必须具备的功能。

一般软件的用户了解他们所做的工作，但是通常不能完整、准确地表达他们的要求，也不知道如何用计算机去完成他们的要求。而软件开发人员知道怎样用计算机去实现各种要求，但对不同用户的具体要求却并不完全清楚。所以在系统需求分析阶段，系统分析员必须与用户密切配合，充分交流，共同讨论决定：哪些需求是真正需要并可能完成的，并对其加以确切描述。最好使用数据流图、数据字典和简要的算法工具表示用户确认的系统逻辑模型。

在需求分析阶段确定的系统逻辑模型以及软件需求说明书（或系统功能说明书）是以后设计和实现目标系统的基础。因此必须比较准确、完整地体现用户的要求。经用户确认和管理机构评审后，才能进入下一阶段。

3. 软件系统设计

系统设计是软件工程的技术核心，一般可分为总体设计和详细设计两个步骤进行。

首先，进行总体设计，在这个阶段概要地设计要解决的问题。应该考虑几种可能的解决方案。例如是用批处理输入，还是用人机交互；用文件系统存储，还是用数据库存储……，通常可以考虑高、中、低3种可能解决的方案。

软件人员可使用系统流程图或其他工具描述每种可能实现的系统，估计各个方案的成本和效益，并在充分权衡各种方案利弊的基础上，推荐一个最佳方案，并制定出推荐方案的详细计划。

总体设计的一项重要任务就是设计人员把已确定的需求转换成相应的体系结构。结构中的每一组成部分都是意义明确的模块，每个模块都与某些需求相对应。通常可用层次图或结构图

描绘软件体系结构系统。

设计的第 2 个阶段是详细设计，详细设计的任务是对每个模块要完成的工作进行具体的描绘，设计出程序的详细规格说明。所有的设计考虑都应写入，即包含所有必要的设计细节。为编写程序打下基础，并提交评审。

4. 程序编码

本阶段的任务是根据上面的详细设计，写出正确及容易理解和维护的程序代码。

软件人员根据目标系统要求和实际环境，选取一种适当的程序设计语言，把详细设计中的每一个功能模块编写成选定语言表示的“源程序清单”，这一步工作也叫做编码（coding）。写出的程序应结构良好，清晰易读，与详细设计一致的代码。

5. 软件测试

测试是保证软件质量的重要手段。

首先进行单元测试，查找各个模块在功能上和结构上存在的问题，并立即给予纠正。

其次，进行综合测试，通过各种类型测试和调试，使软件达到预定要求。最基本的测试是集成测试和验收测试。集成测试是根据设计的软件体系结构，把经过单元测试的模块按选定的策略装配起来，按规定的各项需求，逐项进行有效性测试。最后的验收测试是按照规格说明书，由用户或在用户参加的情况下，对目标系统进行验收。通过对软件测试结果的分析，可以预测此软件的可靠性，决定此软件是否合格，能否交用户使用。

6. 软件维护

已交付的软件投入正常使用，进入运行阶段。这一阶段可能要持续几年甚至十几年。在软件运行阶段，需要通过各种必要的维护活动使软件系统持久地满足用户使用的要求。

实际上每进行一项维护活动，都可能会影响到软件的全局或局部，因此需要认真仔细地考虑。较大变动的维护活动更应该提出维护要求，分析维护要求，提出维护方案，审批维护方案，确定维护计划，修改软件设计，修改程序，测试程序，复查验收等一系列步骤。每一项维护活动都应该准确地记录下来，作为正式的文档资料予以保存。表 1.1 采用表格形式列出了软件生命周期各阶段的名称和任务。

表 1.1　软件生命周期各阶段的名称和任务

阶段名称	关键问题	完成任务
问题定义	问题是什么	关于规模和目标的报告书
可行性研究	有可行的解吗	系统的高层逻辑模型：数据流图、成本/效益分析
需求分析	系统必须做什么	系统的逻辑模型：数据流图、数据字典、算法描述
总体设计	概括地说，应该如何解决这个问题	可能的解法：系统流程图、成本/效益分析 推荐的系统体系结构：层次图或结构图

续表

阶段名称	关键问题	完成任务
详细设计	怎样具体地实现这个系统	编码规格说明：HIPO 图或 PDL
编码	正确的程序模块	源程序清单；单元测试方案和结果
综合测试	符合要求的软件	综合测试方案和结果；完整一致的软件配置
维护	持久地满足用户需要的软件	完整准确的维护记录

习 题 一

1. 计算机软件有哪些特点？举例说明。
2. 什么是软件危机？分析软件危机产生的原因。
3. 以一个软件系统为例，说明软件生命周期的各个阶段及各阶段的任务。

第 2 章　软件体系结构概念

经历了软件危机后，人们认识到软件开发也需要使用工程化的方法和管理。为了避免软件开发的盲目性，提高软件的质量，在软件开发中需要有效的理论作为指导。软件体系结构作为软件开发中的设计指导思想，无疑对软件开发的成败起着至关重要的作用。软件体系结构作为软件工程中一个新兴的研究课题，是随着描述大型复杂系统结构的需要，以及开发人员和计算机科学家在大型软件系统研制过程中对软件系统理解的逐步深入而发展起来的。目前，国际计算机界对软件体系结构的研究都有着极高的热情，并给予高度的重视。随着计算机网络和软件业的发展，软件体系结构的研究必然会越来越深入地开展下去。

2.1　软件体系结构的基本概念

目前软件体系结构在软件工程中已有了很广泛的应用，但是由于有关的研究还刚刚兴起，因而对它的理解还没有达到共识。许多研究人员基于自己的经验从不同角度、不同侧面对软件体系结构进行了刻画，但至今为止还没有一个能被广泛接受的、标准化的关于软件体系结构的定义。下面按时间顺序列出一些重要文献中有关软件体系结构的定义，由此可以看出其概念的演化发展过程。

1992 年 Perry 和 Wolf 在他们早期关于软件体系结构的论文中指出：软件体系结构是一组具有一定形式的结构化元素或称为设计元素组成。它们分为 3 类，分别是：处理元素、数据元素和连接元素。

1993 年 Shaw 和 Garlan 在一篇被誉为软件体系结构研究里程碑的重要论文中指出：软件体系结构是软件设计过程中的一个层次。这个新的层次超越了计算过程中的算法设计和数据结构设计。他们提出：软件体系结构由元素和连接以及对它们的约束组成。由此，设计和说明系统的总体结构作为一个新问题被提出来了。他们还指出：软件体系结构问题包括总体组织和全局控制，通信协议，同步，数据存取，给设计元件分配特定功能，设计元素的组织规模和性能，以及在各设计方案中进行选择等方面。

1994 年 Hayes-Roth 在 ARPA 特定软件体系结构的报告中指出：软件体系结构是一个抽象的系统规范，主要包括用其行为来描述的功能组件、组件和组件之间的相互连接，接口和关系。

1994 年 Bass 等人在关于软件体系的品质和属性方面的研究中指出，一个软件系统的体系结构设计可以由 3 个方面来描述：功能划分、结构和功能到结构的分配。

1995 年 Garlan 和 Perry 在 IEEE 软件工程学报上修正了他们原先对软件体系结构的定义：软件体系结构是一个系统各组件的结构，它们之间的相互关系，以及进行设计的原则规范和随时间演进的指导方针。

1995 年 Soni、Nord 和 Hofmeister 这 3 位西门子公司的员工通过对工业界普遍使用和流行

的开发设计环境进行研究后指出：至少可以从4个不同的角度对软件体系结构进行研究。这就是后来发展的软件体系结构四视图观点。四视图观点反映的体系结构是：概念上的体系结构、模块体系结构、代码体系结构和运行体系结构。

概念上的体系结构：描述系统的主要成分及它们之间的关系。主要包括：组件、连接器、性能等。影响它们的主要因素是应用问题的分解和划分。

模块体系结构：描述功能分解和层次结构两个正交的结构。主要包括：子系统、模块、引入、引出、模块的界面管理、控制和一致性等。它们受到软件的设计原则、组织结构和软件技术的影响。

代码体系结构：描述源程序，二进制文件和库文件在系统开发环境中的组织。主要包括：文件、目录、库、软件的配置管理、系统建造等。它受到语言、工具和外部系统等因素的影响。

运行体系结构：描述系统的动态行为。它主要包括：任务、过程、进程、性能、调度、动态分配和不同执行系统之间的接口等。它主要受到硬件体系结构、运行环境、性能和通信机制因素的影响。

另外，还有把软件体系结构分为逻辑视图、处理视图、物理视图和开发视图的四视图观点。逻辑视图是设计的概念模型。处理视图包括并发、同步、异步等。物理视图是软件向硬件及分布配置的映射。开发视图是软件在开发环境中的静态组织结构。

软件体系结构的第一次国际会议在1995年召开。这次会议对软件体系结构的命名进行了探讨和论述。会议论文提出如下观点：

（1）所有关于软件体系结构的概念模式都认为，软件体系结构包括软件组件、组件间的联系以及系统构造、方式、约束、语义、分析、属性、基本原理和系统需求。这一领域的研究可从结构模式语言（ADL）的发展中得到体现。

（2）框架模式观点与概念模式有一定的相似之处，但更强调整个系统的连贯性结构。框架模式更多针对特定的应用领域问题。该领域的研究包括CORBA（公共对象请求代理体系结构）及基于CORBA的系统结构模型等。

（3）动态模式强调系统的行为品质。这里的动态可以指系统总体配置的变化，建立或禁止预定义的通信和互联通道，以及计算的进展，如数据值的变化。

（4）进程模式强调软件体系结构的构造或构造过程中的步骤和进程。在此观点下，软件体系结构是一个进程描述结果。

（5）上述4方面观点彼此并不互相排斥，也不表示在软件体系结构基本问题上看法的冲突。它们只是总结了观察分析软件体系结构研究领域的不同角度：软件体系结构的组成成分，结构整体，已经形成和正在形成的行为。

软件研究者从不同的角度出发，对软件体系结构的定义有所不同。不同的认识反映了软件体系结构概念的广度。对于初学者而言，从软件设计的角度看体系结构的本意：它是指构建系统时的构造范型、构造风格和构造模式。软件体系结构对软件系统的构造起着指导性作用。它回避了软件系统的功能细节，着重于讨论软件系统的总体构架。

软件体系结构涉及多方面的内容：

（1）软件的组成组件及系统构架；

（2）软件各组件的选择，各组件之间的相互作用，软件组件的进一步复合以及指导软件复合过程的总体模式。

（3）系统的功能、性能、设计以及从多种方案中进行选择的决策。

可见，软件体系结构关注的是系统结构及其组成组件。软件体系结构开始于系统的早期设计。它主要描述以下属性：系统的组件，包括功能组件和数据组件；系统组件间的连接，包括数据流和控制流；组件和连接的约束，包括组件间的通信协议，组件间的同步等；以及用组件和连接表示的系统整体结构的拓扑关系。

根据软件生命周期的标准，Perry 和 Wolf 将软件开发过程划分为需求分析、体系结构设计、详细设计和实现。

（1）需求分析：根据用户的需求，决定软件系统的功能。

（2）体系结构设计：选择模式，选择组件，组件之间的关系以及它们之间的约束。以此为框架，为详细设计奠定基础。

（3）详细设计：主要对系统进行模块化和描述各个组件间的详细接口，算法和数据结构类型等。

（4）实现：使用程序设计语言实现设计方案的要求。

2.2 研究软件体系结构的重要性

良好的体系结构设计是决定软件系统成功的重要因素。

软件的设计如同一幢大楼的设计，是一个系统工程，应该在建造大楼之前在总体结构上考虑周到。这种总体设计的工作很多人往往掌握得不是很好，而软件体系结构的出现给软件工程师提供了一个重要的规范，软件体系结构作为软件设计的一个高层次的抽象，给工程师提供了更好的方法来理解软件以及寻找出新的方法来构造更大、更复杂的软件系统。

在以往的软件设计实践中，我们曾看到不少成功的和失败的案例。例如美国 IBM 公司在 1963 年～1966 年开发的 IBM 360 的操作系统就是一个典型的失败。这个项目共花费了 5000 个人年的工作量，写了近 100 万行的源程序，结果却令人失望。这个项目的负责人 E·Brooks 事后在总结他在组织开发过程中的沉痛教训时说：“正像一只逃亡的野兽落到泥潭中做垂死的挣扎。越是挣扎，陷得越深，最后仍无法逃脱灭顶的灾难。程序设计工作正像这样一个泥潭，一批批程序员被迫在泥潭中拼命挣扎，谁也没有料到问题会陷入这样的困境……”IBM 360 操作系统开发的历史教训成为软件开发项目引以为戒的典型事例。

由历史经验可以得出，在软件开发项目的整个设计过程中，选择一个良好的体系结构和构造规则是至关重要的。对于一些工程项目，如建筑高楼，许多自然规则和历史经验给设计者留下选择结构的参考，使设计者有一定的规律可遵循。但是软件开发却一直被认为是个人创造力的产物。尽管也有一些成功的体系结构例子，但是其描述也是不成熟、不正规的，因此，软件开发体系结构设计可遵循的规律和限制很少，十分灵活，从而导致了许多开发者无规可循，难以把握。另外，软件为了适应主观和客观环境的变化，在它的整个生命周期中，需要经历多次