普通高等教育“十一五”国家级规划教材

高等学校软件工程系列教材

软件设计与体系结构

齐治昌　主编

董　威　文艳军　陈振邦　编著

高等教育出版社
Higher Education Press

普通高等教育“十一五”国家级规划教材
高等学校软件工程系列教材

软件设计与体系结构

RUANJIAN SHEJI YU TIXI JIEGOU

齐治昌　主编
董　威　文艳军　陈振邦　编著

高等教育出版社·北京
HIGHER EDUCATION PRESS　BEIJING

内容提要

软件工程强调以工程化思想和方法开发软件，而软件设计作为软件开发过程中的核心活动之一，对开发出满足需要的高质量软件起关键作用。本书对软件设计以及软件体系结构的相关思想、理论与方法进行了系统的介绍，包括软件设计与软件体系结构在软件工程中的地位和作用、软件设计的基本方法与原则、统一建模语言UML 2.0、面向对象的软件设计方法、面向数据流的软件设计方法、人机界面设计、软件体系结构风格与设计模式、基于构件的软件体系结构、软件体系结构评估、软件设计的进化等内容。本书包含了作者多年来在软件开发实践、软件工程教学和科研活动中的认识与体会，并结合了大量的案例分析，力求全书内容与组织结构的系统性、先进性、基础性和实用性。

本书可作为高等院校计算机科学与技术专业、软件工程专业或信息类相关专业的本科生和研究生教材，以培养学生的软件设计思维能力以及方法和技术的运用能力，同时也适用于开发人员和项目管理人员在软件开发实践中参考。

图书在版编目(CIP)数据

软件设计与体系结构/齐治昌主编. —北京：高等教育出版社，2010.2

ISBN 978-7-04-028408-9

Ⅰ.软… Ⅱ.齐… Ⅲ.软件设计-高等学校-教材

Ⅳ.TP311.5

中国版本图书馆CIP数据核字(2009)第220738号

策划编辑 倪文慧 **责任编辑** 俞丽莎 **封面设计** 于文燕 **版式设计** 王艳红

责任校对 杨凤玲 **责任印制** 毛斯璐

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街4号	**咨询电话**	400-810-0598
邮政编码	100120	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	唐山市润丰印务有限公司	**畅想教育**	http://www.widedu.com
开　　本	787×1092　1/16	**版　　次**	2010年2月第1版
印　　张	19	**印　　次**	2010年2月第1次印刷
字　　数	430 000	**定　　价**	27.00元

物料号　28408-00

前　言

随着计算机技术、微电子技术、网络技术和多媒体技术的迅速发展和广泛应用，今天的社会进入了以计算机为核心的信息社会。软件被认为是信息化的灵魂，已被用于政治、经济、文化、科技、教育、国防、生活等各个领域。随着软件在社会中的地位和作用越来越显著，人们对软件的质量、成本和开发周期等方面提出的要求也越来越高。软件工程强调以工程化思想和方法开发软件，而软件设计作为软件开发过程中的核心活动之一，对开发出满足需要的高质量软件起关键作用。软件设计在软件工程中所体现的重要性包括：软件设计是对软件需求的直接体现；软件设计为软件实现提供直接依据；软件设计将综合考虑软件系统的各种约束条件并给出相应方案；软件设计的质量很大程度上将决定最终软件系统的质量；及早发现软件设计中存在的错误将极大地减少软件修复和维护所需的成本；等等。在软件工程的发展过程中，出现了大量与软件设计相关的方法与技术，而软件体系结构作为软件设计过程中控制软件复杂性、提高软件系统质量、支持软件开发和复用的重要手段之一，自提出以来日益受到软件研究者和实践者的关注。

在当前我国软件产业的发展过程中，基础扎实、知识全面、经验丰富的高水平软件设计人员仍然非常缺乏，这成为我国软件产业发展的制约因素。在当前软件工程专业方向的教育中，把软件设计和软件体系结构相关理论与方法作为单独一门课程进行系统地讲述，对培养大量所需的高水平软件设计人员至关重要。而在教育部高等学校计算机科学与技术教学指导委员会制定的《软件工程(本科)专业规范》中，“软件设计与体系结构”已经作为一门核心课程单独列出，并有相应的知识单元和知识点，但亟需相应的教材以便于高等院校教学实施。针对以上背景和需求，本书对软件设计以及软件体系结构的相关思想、理论与方法进行了系统的介绍，包括软件设计与软件体系结构在软件工程中的地位和作用、软件设计的基本方法与原则、统一建模语言 UML 2.0、面向对象的软件设计方法、面向数据流的软件设计方法、人机界面设计、软件体系结构风格与设计模式、基于构件的软件体系结构、软件体系结构评估、软件设计的进化等内容。

在教学计划中，建议对第 1～10 章采用自然顺序讲授。如果课时有限，对于本科生，第 9、10 章可进行略讲或略去，第 5 章也可以根据情况省略，其他各章按照顺序讲述。本书内容曾多次在国防科技大学计算机专业本科生和硕士研究生的相关课程教学中讲授。

本书由齐治昌主编，董威撰写了第 1、2、3、4、5、10 章，文艳军撰写了第 7、8 章，陈振邦撰写了第 6、9 章。北京大学郁莲博士认真审阅了本书的全部初稿，并提出了许多中肯的修改意见。借此机会，作者谨向为本书付出辛勤劳动和智慧的老师、同学和软件工程师表示诚挚的谢意。

最后，诚恳欢迎读者和专家对本书提出批评意见和改进建议。

作　者

2009 年 9 月

目　录

第 1 章

软件工程与软件设计

随着微电子技术、计算机技术、网络技术和多媒体技术的迅速发展和广泛应用，今天的社会已进入了以计算机为核心的信息社会，计算机已广泛应用于航空航天、工业控制、办公自动化、商业信息处理、家用电器等领域，成为人们工作和生活中不可缺少的工具。软件被认为是信息化的灵魂，它在硬件的支持下，肩负着信息采集、存储、处理、加工、传输、显示、人机交互、控制、应用等任务。目前，形形色色的软件被用于政治、经济、文化、科技、教育、军事、生活的各个领域，人们对软件的依赖越来越紧密。由于软件在社会中的地位和作用越来越显著，人们对软件的功能、质量、成本和开发周期等方面提出的要求也越来越高。

然而，软件的功能越强、使用越方便，其规模和复杂程度就越高，如果软件开发水平跟不上，就会大量出现软件开发计划一拖再拖、成本失去控制、软件质量得不到保证等现象。因此，数十年来，人们十分重视软件开发方法、工具和环境的研究，并在这些领域取得了重要的成果，使得软件工程取得了长足的进步，并逐渐成熟。软件工程强调以工程化思想和方法开发软件，而软件设计作为软件开发过程中的核心活动之一，对开发满足需要的高质量软件起到关键作用。本章首先从全局的观点对软件工程的概念、软件生存周期、开发过程模型进行概述，然后阐述软件设计在软件工程和软件开发中的地位、作用及其相关特征和组成要素。在软件工程的发展过程中，出现了大量与软件设计相关的方法与技术，而软件体系结构作为软件设计过程中控制软件复杂性、提高软件系统质量、支持软件开发和复用的重要手段之一，自提出以来日益受到软件研究者和实践者的关注。另外，本章对软件体系结构的主要概念、发展过程和所关注的内容也进行了概要描述。

1.1 软 件 工 程

软件工程是计算机软件发展到一定阶段后，为了应对软件危机而出现的。本节首先对软件的定义和特点进行简要介绍，然后对软件危机、软件工程的发展、相关概念、目标与原则等方面进行回顾与总结。

1.1.1 软件概述

计算机软件是与计算机系统操作有关的程序、规程、规则及任何与之有关的文档及数据[1-1]，即：

计算机软件＝程序＋数据＋文档

软件由两部分组成：其一是机器可执行的程序及有关数据；其二是机器不可执行的，与软件开发、运行、维护、使用、培训有关的文档。程序(Program)是用程序设计语言描述的、适合计算机处理的语句序列，它是软件开发人员根据用户需求开发出来的。程序设计语言编译器可以将程序翻译成机器可执行的指令，这组指令亦称机器语言程序，它将根据用户的需求控制计算机硬件的运行，处理用户提供或机器运行过程中产生的各类数据并输出结果。曾经出现过的程序设计语言有几百种，但广泛使用的高级语言不过十余种，例如，用于科学计算的 FORTRAN 语言；用于事务处理的 COBOL 语言；支持结构化程序设计的 Pascal、C、Ada 语言；支持面向对象程序设计的 C++、Java 语言等。此外，还有一些专用的高级语言，例如用于数据库查询的 SQL 语言。文档(Document)是一种数据媒体和其上所记录的数据，文档记录软件开发的活动和阶段成果，它具有永久性并能供人或机器阅读。它不仅可以用于专业人员和用户之间的通信和交流，还可以用于软件开发过程中管理和运行阶段的维护。在现代软件工程中，文档起着非常重要的作用。

软件是逻辑产品而不是物理产品，因此软件在开发、生产、维护和使用等方面与硬件相比均存在明显的差异。软件开发与硬件开发相比，更依赖于开发人员的业务水平、智力、人员的组织、合作和管理。在大多数场合，软件的开发、设计几乎都是从头开始的，开发的成本和进度很难估计。软件在提交使用以前，尽管经过了严格的测试和试用，但仍不能保证软件没有潜在的错误。软件开发成功之后，一般只需对原版软件进行复制即可使用。但是，软件在使用过程中的维护工作却比硬件复杂得多，包括对错误进行修改的“纠错性维护”、对软件性能和功能进行完善和改进的“完善性维护”、运行环境发生变化后的“适应性维护”等。由于软件内部的逻辑关系复杂，软件在维护过程中还可能产生新的错误，因此，软件产品在使用过程中的维护工作远比硬件产品的维护复杂。

经过数十年的发展，计算机软件已经广泛应用于各个领域，下面根据软件应用的目标和特点简单地介绍计算机软件的应用领域和类型。

(1) 系统软件。计算机系统软件是计算机管理自身资源(如 CPU、存储器、外部设备等)、提高计算机的使用效率并为计算机用户提供各种服务的基础软件。系统软件要为各类用户提供尽可能标准、方便的服务，尽量隐藏计算机系统的某些低级特征或实现细节。系统软件包括操作系统、编译器、数据库管理系统、系统检查与诊断软件等。

(2) 实时软件。监视、分析和控制现实世界发生的事件，能以足够快的速度对输入信息进行处理并在规定的时间内做出反应的软件，称之为实时软件。实时软件依赖于处理机系统的物理特性，如计算速度和精度、I/O 信息处理与中断响应方式、数据传输效率等。支持实时软件的操作系统称为实时操作系统。实时软件必须有很高的可靠性和安全性。

(3) 嵌入式软件。嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分，控制该系统的运行，进而实现一个特定的物理过程。用于嵌入式计算机系统的软件称为嵌入式软件。嵌入式计算机系统一般都要和各种仪器、仪表、传感器连接在一起，因此，嵌入式软件一般需要具有实时地采集、处理、输出数据的能力。

(4) 科学和工程计算软件。它们以数值算法为基础，对数值量进行处理和计算，主要用于科学和工程计算，例如数值天气预报、计算机系统仿真、计算机辅助设计(CAD)等。从20世纪50年代起，有经验的程序员就用程序设计语言把许多常用算法编制成标准程序，如今已经积累了大量的科学和工程计算软件，为计算机在科学和工程上的应用作出了重要贡献。

(5) 事务处理软件。用于处理事务信息，特别是商务信息的计算机软件。事务信息处理是软件最大的应用领域，它已由初期零散的、小规模的软件系统，如工资管理系统等发展成为管理信息系统(MIS)，如世界范围内的飞机订票系统等。事务处理软件需要访问、查询、存放有关事务信息的一个或几个数据库，经常按某种方式和要求重构存放在数据库中的数据，能有效按照一定要求和格式生成各种报表。它们往往具有良好的人机界面，在大多数场合采用交互工作方式。

(6) 人工智能软件。支持计算机系统产生类似人类某些智能的软件。它们不是用传统的计算或分析方法求解复杂问题，而是采用诸如基于规则的演绎推理技术和算法，在很多场合还需要知识库的支持。迄今为止，在专家系统、模式识别、自然语言理解、人工神经网络、自动程序设计、机器人学等领域开发了许多人工智能应用软件，用于医疗诊断、图像和语音自动识别、语言翻译等。

(7) 个人计算机软件。个人计算机上使用的软件也可包括系统软件和应用软件两类。在个人计算机上已出现大量的文字和表格处理软件、图形和多媒体信息处理软件、个人和商业上的财务处理软件、网络软件等，并采用多窗口、多媒体等技术，使个人计算机具有用文字、图形、图像、声音进行人机交互的能力，为个人计算机的普及创造了必要条件。目前，个人计算机普遍与计算机网络相结合，使得通信、网络资源共享等更加方便，加速了人类社会信息化的进程。

1.1.2 软件危机

20世纪60年代末期开始，“软件危机”一词在计算机界广为流传。事实上，软件危机几乎从计算机诞生之日就出现了。当时训练有素的程序员大多数还处于“技艺式”的工作状态，他们不用框图和注释就能够熟练编写出几百行程序代码并很快在机器上调试通过。他们熟悉程序的全部操作和结构，为了减少计算机的时/空开销，他们创造并使用了许多令人惊叹的技巧。然而，客观上，工业、商业、科学技术和国防等部门对软件功能的需求很高，软件规模很大，往往有几万、几十万、甚至几百万行代码。在当时的条件下，要完成这样一个系统，在一定的时间内依靠一个人或几个人的智力和体力是无法实现的。由于软件是逻辑和智力产品，盲目增加软件开发人员并不能成比例地提高软件开发效率。相反，随着人员数量的增加，人员的组织、协调、通信、培训和管理等方面出现的问题将更为严重。人们在大型软件项目开发面前显得力不从心，一些公司或

团体承担的大型软件开发项目经常出现预算超支、软件交货时间延迟、软件质量差、维护困难、在软件维护过程中很容易产生新的错误、软件的可移植性差、软件很少能够复用等问题，工业界为维护软件支付的费用甚至占全部硬件和软件费用的40%～75%。许多重要的大型软件开发项目在耗费了大量的人力和财力之后，由于离预定目标相差甚远不得不宣告失败。这些都使得软件危机达到了令人难以容忍的地步。

从软件危机的种种表现和软件作为逻辑产品的特殊性可以发现软件危机的原因有以下几点：

(1) 用户对软件需求的描述不精确，可能存在遗漏、二义性、错误等。在软件开发过程中，用户甚至还提出修改软件功能、界面、支撑环境等方面的要求，导致需求不断变化。

(2) 软件开发人员对用户需求的理解与用户的期望有差异，这种差异必然导致开发出来的软件产品与用户要求不一致。

(3) 大型软件项目需要组织一定的人力共同完成，但多数管理人员缺乏开发大型软件系统的经验，而多数软件开发人员又缺乏管理方面的经验。各类人员的信息交流不及时、不准确，有时还会产生误解。

(4) 软件项目开发人员不能有效地、独立自主地处理大型软件的全部关系和各个分支，因此容易产生疏漏和错误。

(5) 缺乏有力的方法学和工具方面的支持，过分地依靠程序设计人员在软件开发过程中的技巧和创造性，加剧了软件产品的个性化。

(6) 软件产品的特殊性和人类智力的局限性，导致人们无力处理“复杂问题”。一旦人们采用先进的组织形式、开发方法和工具提高了软件的开发效率和能力，新的、更大、更复杂的问题又出现在人们面前。

在认真地分析了软件危机的原因之后，人们开始用工程的方法探索软件生产的可能性，即用现代工程的概念、原理、技术和方法进行计算机软件的开发、管理、维护和更新。于是，计算机科学技术的一个新领域——“软件工程”诞生了。40多年来，软件工程的研究与应用已经取得很大成就，包括软件开发方法、工具、管理等多个方面，大大缓解了软件危机造成的被动局面。

1.1.3 软件工程的概念

1968年，在德国Garmish召开的NATO(北大西洋公约组织)计算机科学会议上首先提出了“软件工程”的概念，试图建立并使用正确的工程方法开发出低成本、高可靠性并能高效运行的软件，从而解决或缓解软件危机。

软件工程的定义有不同的表述方式，典型的定义包括：

(1) 软件工程是将系统的、规范的、可度量的方法应用于软件的开发、运行和维护过程，以及对上述方法的研究。

(2) 软件工程是用工程、科学和数学的原则与方法，研制、维护计算机软件的有关技术及管理方法。

一般认为，软件工程由方法、工具和过程三个要素组成，如图 1-1 所示。在三个要素中，方法支撑过程和工具，而过程和工具促进方法学的研究。

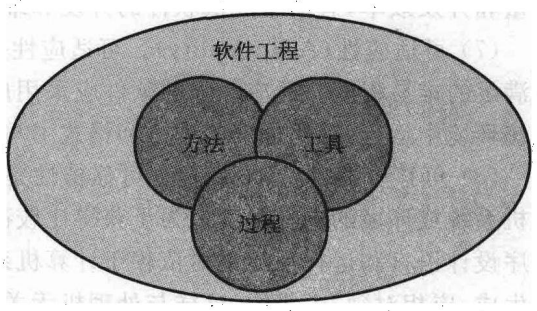

图 1-1　软件工程的组成要素

软件工程方法是完成软件工程项目的技术手段，它支持项目计划和估算、系统和软件需求分析、软件设计、编码、测试和维护等活动。软件工程使用的软件工具是人类在软件开发活动中智力和体力的扩展和延伸，它自动或半自动地支持软件的开发和管理，支持各种软件文档的生成。软件工具最初是零散的，不系统、不配套，后来根据不同类型软件项目的要求建立了各种软件工具箱或集成环境，支持软件开发的全过程。软件工程中的过程贯穿于软件开发的各个环节。管理者在软件工程的过程中，要对软件开发的质量、进度、成本进行评估、管理和控制，包括人员组织、计划跟踪与控制、成本估算、质量保证、配置管理等。

1.1.4　软件工程的目标与原则

软件工程的目标是：在给定成本、进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可复用性、可适应性、可移植性和可追踪性并满足用户需求的软件产品。追求这些目标有助于提高软件产品的质量和开发效率，减少维护的困难[1-2]。

(1) 可修改性(Modifiability)。容许对系统进行修改而不增加原系统的复杂性。它支持软件的调试与维护，但度量起来比较困难。

(2) 有效性(Efficiency)。软件系统能最有效地利用计算机的时间资源和空间资源，一般将系统的时/空开销作为衡量软件质量的一项重要技术指标。很多场合，在追求时间有效性和空间有效性方面会发生矛盾，这时不得不牺牲时间效率换取空间有效性或牺牲空间效率换取时间有效性，因此时/空折中是经常出现的。

(3) 可靠性(Reliability)。软件在给定环境和时间下不发生故障的概率。对于实时嵌入式计算机系统，可靠性是一个非常重要的目标，因为软件要实时地控制一个物理过程，如宇宙飞船的导航、核电站的运行等。如果可靠性得不到保证，一旦出现问题可能会导致灾难性的结果，后果不堪设想。

(4) 可理解性(Understandability)。系统具有清晰的结构，能直接反映问题的需求。可理解性有助于控制软件系统的复杂性，并支持软件的维护、移植或复用。

(5) 可维护性(Maintainability)。软件产品交付用户使用后，能够方便地对它进行修改，以改正潜在的错误以及改进性能和其他属性，使软件产品适应环境的变化等。软件的可理解性和可修改性有利于软件的可维护性。

(6) 可复用性(Reusability)。概念或功能相对独立的一个或一组相关模块定义为一个软部

件，软部件可以在多种场合应用的程度称为部件的可复用性。可复用性有助于提高软件产品的质量和开发效率，有助于降低软件的开发和维护费用。

(7) 可适应性(Adaptability)。可适应性是指软件在不同的系统约束条件下，使用户需求得到满足的难易程度。适应性强的软件应采用广为流行的程序设计语言编码，在广为流行的操作系统环境中运行，采用标准的术语和格式书写文档。适应性强的软件较容易推广和使用。

(8) 可移植性(Portability)。可移植性是指软件从一个计算机系统或环境移植到另一个计算机系统或环境的难易程度。为了获得比较高的可移植性，在软件设计过程中通常采用通用的程序设计语言和运行环境。对依赖于计算机系统的低级(物理)特征部分，如编译系统的目标代码生成，应相对独立、集中，这样与处理机无关的部分就可以移植到其他系统上使用。可移植性支持软件的可复用性和可适应性。

(9) 可追踪性(Traceability)。根据软件需求对软件设计、程序进行正向追踪，或根据程序、软件设计对软件需求进行逆向追踪的能力。软件可追踪性依赖于软件开发各个阶段文档和程序的完整性、一致性、可理解性，降低系统的复杂性会提高软件的可追踪性。

上述的软件开发目标适用于所有的软件系统开发。为了达到这些目标，在软件开发过程中必须遵循下列软件工程原则：抽象、信息隐藏、模块化、局部化、一致性、完全性和可验证性。

(1) 抽象(Abstraction)。抽取事物最基本的特性和行为，忽略非基本的细节。采用分层次抽象的办法可以控制软件开发过程的复杂性，有利于软件的可理解性和开发过程的管理。

(2) 信息隐藏(Information Hiding)。将模块中的软件设计决策封装起来的技术。系统中的模块应设计成“黑箱”，模块接口应尽量简洁，模块外部只能使用模块接口说明中给出的信息，如操作、数据类型等。由于实现细节隐藏，软件开发人员便能够将注意力集中于更高层次的抽象上。

(3) 模块化(Modularity)。模块是程序中逻辑上相对独立的成分，它是一个独立的编程单位，应有良好的接口定义。模块化有助于信息隐藏和抽象，有助于表示复杂的软件系统。模块的大小要适中，模块过大会导致模块内部复杂性的增加，不利于模块的调试和重用，也不利于对模块的理解和修改。模块太小会导致整个系统的表示过于复杂，不利于控制复杂性。

(4) 局部化(Localization)。要求在一个物理模块内集中逻辑上相互关联的计算资源，从物理和逻辑两个方面保证系统中模块之间具有松散的耦合关系，而在模块内部有较强的内聚性，这样有助于控制设计的复杂性。

(5) 一致性(Consistency)。整个软件系统(包括文档和程序)的各个模块均应使用一致的概念、符号和术语；程序内部接口应保持一致；软件与硬件接口应保持一致；系统规格说明与系统行为应保持一致；用于形式化规格说明的公理系统应保持一致等。

(6) 完全性(Completeness)。软件系统不丢失任何重要成分，达到完全实现系统所需功能的程度；当系统处于出错或非预期状态时，系统行为保持正常的能力。完全性要求人们开发必要且充分的模块。为了保证软件系统的完全性，软件在开发和运行过程中需要软件管理工具的支持。

(7) 可验证性(Verifiability)。开发大型软件系统需要对系统逐步分解,系统分解应该遵循系统容易检查、测试、评审的原则,以便保证系统的正确性。使用一致性、完全性、可验证性等原则可以帮助人们实现一个正确的系统。

1.2 软件的生存周期

软件产品从形成概念开始,经过开发、使用和维护,直到最后退役的全过程称为软件生存周期(Software Life Cycle)。软件生存周期根据软件所处的状态、特征以及软件开发活动的目的和任务,可以划分为若干个阶段。目前各阶段的划分尚不统一,但无论采用哪种划分方式,从较高的层面看,软件生存周期都包括软件定义、软件开发、软件使用与维护、退役 4 个部分。如果再进一步细化,软件生存周期一般应包括软件系统可行性研究、需求分析、概要设计、详细设计、软件构造、单元测试、集成测试、确认测试、使用与维护、退役等多个阶段,如图 1-2 所示。可以看到,细化也主要对软件开发阶段中的活动进行具体划分,因为软件开发前和投入使用后所面临的活动相对比较单一。

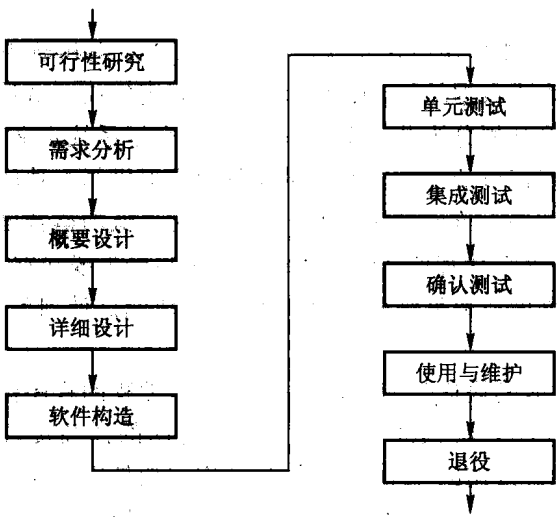

图 1-2 软件生存周期

需要注意的是,图 1-2 只是示意性描述软件生存周期的构成,对于一个具体软件,根据软件特征、开发机构的背景与特点、时间、成本、采用的开发过程模型和技术、应用背景等因素,生存周期中包含的阶段可能或多或少有些不同,并且这些活动的安排并不都是线性顺序的,可能存在迭代、反馈、周期性重复等多种情况。

1. 可行性研究

可行性研究的任务是了解用户的要求及现实环境,从技术、经济和社会等方面研究并论证软件系统的可行性。参与软件系统开发的分析人员应在用户配合下对用户要求及现实环境做深入

细致的调查，并在调查研究的基础上撰写调研报告，根据调研报告及其他有关资料进行可行性论证。可行性论证包括技术可行性、操作可行性和经济可行性三部分。技术可行性是指使用目前可用的开发方法、工具能否支持需求的实现。操作可行性是指用户能否在某一特定的软件运行环境中使用这个软件。经济可行性是指实现和使用软件系统的成本能否被用户接受。

2. 需求分析

需求分析的任务是确定待开发软件的功能需求、性能需求和运行环境约束，编制软件需求规格说明、软件系统的确认测试准则和用户手册概要。软件的功能需求应指明软件必须完成的功能；软件的性能需求包括软件的安全性、可靠性、可维护性、精度、错误处理、适应性、用户培训等；软件系统在运行环境方面的约束指待开发的软件系统必须满足的运行环境方面的要求。软件需求不仅是软件开发的依据，而且也是软件验收的标准。因此，确定软件需求是大型软件开发的关键和难点。系统需求一般由用户提出，由于用户往往缺乏软件开发的知识和经验，系统分析员和软件开发人员不得不与用户反复讨论、协商，使用户需求逐步精确化、一致化、完全化。确定软件需求的过程有时需要反复多次才能得到用户和开发人员的认可。需求分析的一项重要任务是建立面向开发人员的软件需求规格说明，这是软件开发的基础和成败的关键。

3. 概要设计

概要设计根据软件需求规格说明建立软件系统的总体结构和模块间的关系，定义各功能模块的接口，设计全局数据库或数据结构，规定设计约束，制定集成测试计划。对于大型软件系统，应对软件需求进行分解，将其划分为若干个子系统，对每个子系统定义功能模块和各功能模块之间的关系，并给出各子系统接口的定义。概要设计阶段应提供每个功能模块的功能描述、全局数据定义和外部文件定义等，并力争做到功能模块之间有比较低的耦合度，而功能模块内部有较高的内聚度。设计的软件系统应具有良好的总体结构并尽量降低模块接口的复杂性。概要设计应提供概要设计说明书、数据库或数据结构设计说明书、集成测试计划等文件。

4. 详细设计

对概要设计产生的功能模块逐步细化，形成若干个可编程的程序模块，用某种设计或建模语言设计模块的内部细节，包括算法、数据结构和各模块之间的详细接口信息，为编写源代码提供必要的说明。软件设计应该遵循的一个原则是：设计应与软件需求保持一致，设计的软件结构应支持模块化、信息隐藏等软件工程的原则。软件设计可以选用的方法和工具是比较多的，如结构化的设计方法、面向对象的设计方法等，软件开发人员可以根据实际情况选用适当的方法。

5. 软件构造

软件构造的主要任务是，根据详细设计文档将详细设计转化为所要求的编程语言或数据库语言的程序，并对这些程序进行调试，该过程可能和软件单元测试交替进行，验证程序模块接口与详细设计文档的一致性。系统分析方法、系统设计方法、编程方法及选用的程序设计语言应该尽可能匹配。如采用结构化的分析方法就应该采用结构化的设计方法和编程技术，例如 C 语言；若采用面向对象的分析和设计方法，就应该选用面向对象的编程技术和编程语言，如 C++、Java 等。在编程过程中不仅要注意程序的正确性，与详细设计文档保持一致，而且还要使程序具有良

好的风格，以利于程序的调试、理解和维护。

6. 单元测试

单元测试的对象是软件设计的最小单位——模块。单元测试的依据是详细设计描述，应对模块内所有重要的控制路径设计测试用例，以便发现模块内部的错误。单元测试任务包括模块接口测试、模块局部数据结构测试、模块边界条件测试、模块中所有独立执行路径测试、模块的各条错误处理路径测试等。一般认为单元测试应紧接在编码之后，当源程序编制完成并通过复审和编译检查，便可开始单元测试。测试用例的设计应与复审工作相结合，根据设计信息选取测试数据将增大发现上述各类错误的可能性。根据需要，可能要为测试模块开发驱动模块或桩模块。

7. 集成测试

根据概要设计中各功能模块的说明及制定的集成测试计划，对通过单元测试的模块逐步进行集成和测试。集成测试应对系统各模块间的连接正确性进行测试；测试软件系统或子系统的输入/输出处理是否达到设计要求；测试软件系统或子系统正确处理能力和承受错误的能力等。通过集成测试的软件应生成满足概要设计要求、可运行的系统源程序清单和集成测试报告。

8. 确认测试

根据软件需求规格说明定义的全部功能和性能要求以及软件确认测试计划对软件系统进行测试，测试系统是否达到了系统需求。确认测试应有用户参加，确认测试阶段应向用户提交最终的用户手册、操作手册、源程序清单及其他软件文档。确认测试结束时应生成确认测试报告、项目开发总结报告。为验证软件产品是否满足软件需求规格说明的要求，必须按照测试计划的要求编制大量的测试用例、采用多种方法和工具、组织专门的测试队伍并严格组织实施。由专家、用户、软件开发人员组成的软件评审小组在对软件确认报告、测试结果和软件进行评审通过后，软件产品正式得到确认，可以交付用户使用。

9. 使用与维护

将软件安装在用户确定的运行环境中，在测试通过后移交用户使用。软件的使用是软件发挥社会和经济效益的重要阶段。在软件使用过程中，客户或维护人员必须认真收集所发现的软件错误，定期或阶段性地撰写“软件问题报告”和“软件修改报告”。软件维护是对软件产品进行修改或对软件需求变化做出响应的过程。当发现软件产品中的潜伏错误，或用户提出要对软件需求进行修改，或软件运行环境发生变化时都需要对软件进行维护。软件维护不仅针对程序代码，而且还针对软件定义、各个开发阶段生成的文档。软件维护直接影响软件的应用和软件的生存周期，而软件的可维护性又与软件的设计密切相关。因此软件在开发过程中应该重视对软件可维护性的支持。

10. 退役

这是软件生存周期中的最后一个阶段，由于软件支持的业务或系统已经不再使用，或者软件已经不能满足当前需要，而对其进行维护的难度和成本可能超过开发一个新系统，因此终止对软件产品的支持，该软件停止使用。

通过总结软件生存周期中各阶段的主要任务，还可以看到该划分阶段的方式还有一个明显