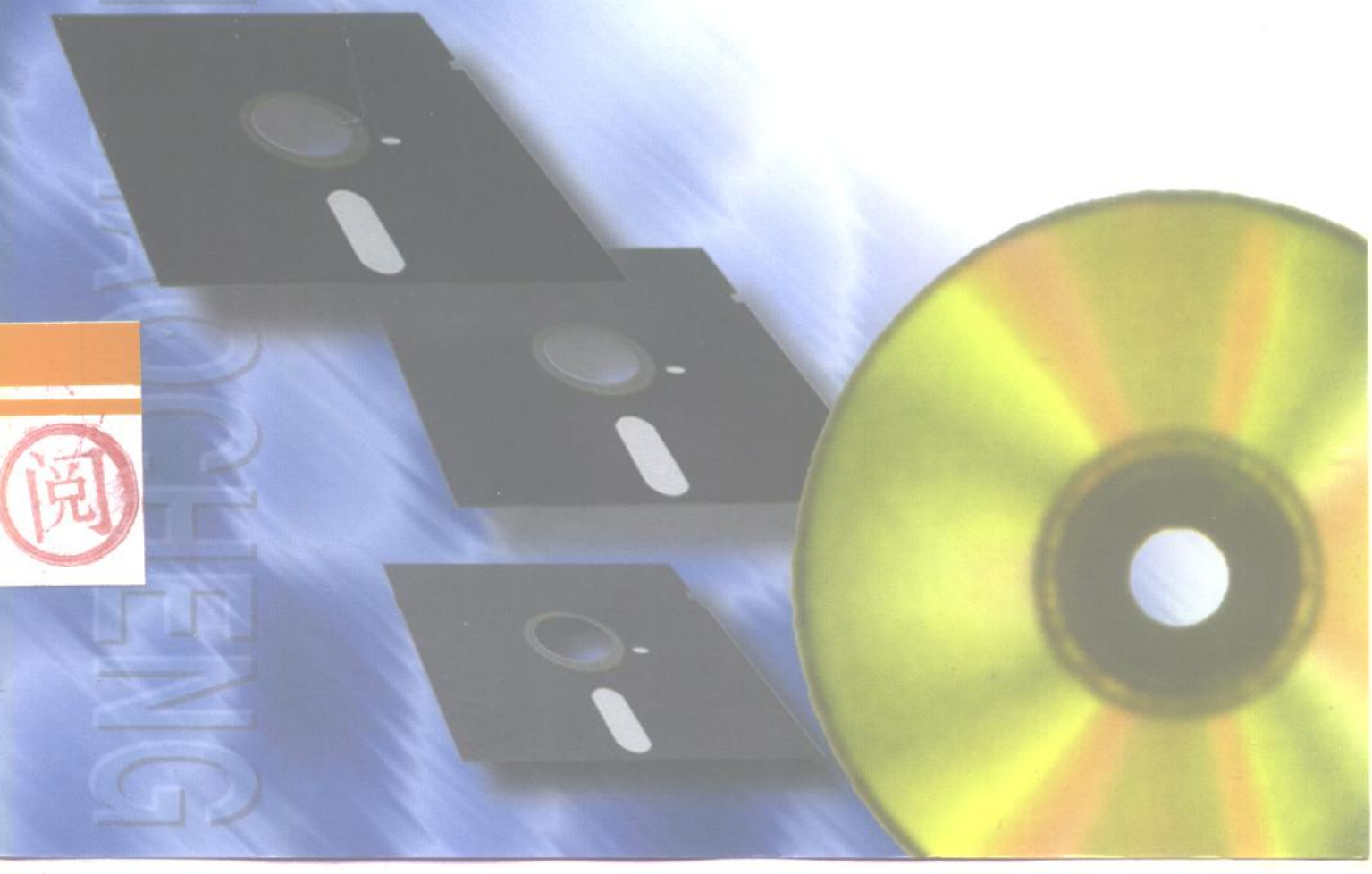
RUANJIAN GONGCHENG JIAOCHENG

软件工程教程

主 编 师素娟 副主编 韩林山 陈根生

黄 河 水 利 出 版 社



软件工程教程

主　编　师素娟
副主编　韩林山　陈根生

黄河水利出版社

图书在版编目(CIP)数据

软件工程教程/师素娟主编.—郑州:黄河水利出版社,1999.5

ISBN 7-80621-308-2

Ⅰ.软… Ⅱ.师… Ⅲ.软件工程 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(1999)第 13235 号

责任编辑:雷元静　　封面设计:朱　鹏

责任校对:赵宏伟　　责任印制:温红建

出版发行:黄河水利出版社

地址:河南省郑州市顺河路黄委会综合楼 12 层　邮编:450003

印　刷:黄委会设计院印刷厂

开　本:787 mm×1092 mm　1/16　　印　张:13.25

版　别:1999 年 5 月 第 1 版　　印　数:1－2000

印　次:1999 年 5 月 郑州第 1 次印刷　　字　数:314 千字

定　价:23.00 元

前　言

随着计算机技术的普及和发展，各个行业对专业 CAD 应用软件的开发要求越来越高、越来越迫切。提高软件质量，缩短软件开发周期，降低软件开发成本，已经成为开发 CAD 应用软件的当务之急。软件工程学就是解决这一问题的最佳途径。

软件工程学以工程化的思想作指导，研究软件开发、维护与管理的普遍原理与技术。本书结合工程专业，从实用的角度讲述软件工程的基本概念、基本原理及常用的技术、方法、工具等。

全书内容共分十四章。第一章，概括介绍软件工程学产生的历史背景及软件工程学的基本概念、原理、方法和发展现状。第二章，介绍软件设计中的科学方法。第三章，介绍工程 CAD 软件的特点、开发模式、开发方法、质量评价等。第四章到第十三章，介绍软件生命周期中各个阶段的任务、目标、过程、方法和开发工具等。第十四章，介绍软件项目的特点，软件管理的职能、工具，成本估计，质量保证等。

全书由师素娟主编，韩林山、陈根生为副主编。其中，第一章、第四章、第十三章由韩林山编写；第二章、第七章、第九章由师素娟编写；第三章、第十章由王铁生编写；第五章、第十二章由张瑞珠编写；第六章、第十一章由陈根生编写；第八章、第十四章由武兰英编写。

由于时间仓促，作者知识水平有限，书中难免存在不当或错误之处，恳请广大读者和有关专家指正。

编　者

1999 年 5 月

目　录

第一篇　总　论

第二篇　系统分析

第三篇　系统设计

第四篇　编码与测试

第五篇　维护与管理

第一篇 总 论

在人类社会发展的历史上，工程技术的进步一直是产业发展的巨大推动力。特别是1946年世界上第一台电子计算机的诞生，标志着人类由工业化社会进入了信息化社会，以计算机产业和计算机应用服务业为支柱的信息工业，成了信息化社会的主要基础之一。

20世纪60年代以来，由于新的微电子元器件的出现，计算机硬件的性能和质量逐年提高，计算机硬件的价格大幅度下降，使得计算机的应用几乎遍及社会的各个领域和部门，成为人们日常工作和生活中不可缺少的工具。随着计算机应用的日益普及和深入，人们对软件的需求量急剧增加。但此时计算机软件的开发技术却远远没有跟上硬件技术的发展，使得软件开发的成本逐年剧增，更为严重的是，软件的质量没有可靠的保证。软件开发的速度与计算机普及的速度不相适应，软件开发技术已经成为影响计算机系统发展的“瓶颈”。

计算机软件开发早期所形成的错误的开发方式，严重阻碍了计算机软件的开发，导致了20世纪60年代软件危机的发生。60年代后期，西方的计算机科学家开始认真研究解决软件危机的方法，提出借鉴工程界严密完整的工程设计思想来指导软件的开发与维护，并取得了可喜成果，从而一门新的学科——软件工程学(Software Engineering)诞生了。

第一章 概 论

20世纪60年代的软件危机提高了人们对软件开发重要性的认识。随着社会对软件需求的增长，计算机软件专家加强了对软件开发和维护的规律性、理论、方法和技术的研究，从而形成了一门介于软件科学、系统工程和工程管理学之间的边缘性学科，称之为软件工程学。随着软件的发展和商品化，这门学科的研究范围越来越广，分别形成了软件工程经济、软件工程方法、软件工程标准与规范、软件工程工具与环境等分支学科。对软件设计人员来说，了解和掌握软件工程方法是非常重要的。

第一节 计算机软件

一、软件(Software)

软件是软件工程学的一个重要概念。许多人认为“软件就是程序”。那么，软件是不是程序呢？

程序是计算机用户使用计算机，为完成某项特定任务而编写的一个有序的命令和数据的集合。这些命令可以是低级语言的指令，也可以是某种高级语言的语句。特定的任务可以是计算某个具体问题，控制某一制作的工艺流程或处理某件日常事务。一般的用

户程序属研制者本人所有，即程序的研制者、用户、维护者是同一个人或同一批人。

软件是程序的完善和发展，是经过严格的正确性检验和实际试用，并具有相对稳定的文本和完整的文档资料的程序。这些文档资料包括功能说明、算法说明、结构说明、使用说明和维护说明等。

一般说来，软件的用户大多数不是开发者本人。软件可以作为商品在市场上出售。

大、中型软件常常称为“系统”。系统是指为了完成某项任务或论述某个实体，能够方便地同其它事物相区别，而被独立研究或讨论的对象。如计算机软件系统、机械 CAD 系统、土木 CAD 系统、水利 CAD 系统等等。

软件工程学认为，软件是与计算机系统操作有关的计算机程序、规程、规则及相关的文件和数据。根据这一定义，软件可分为可执行部分和不可执行部分。

1. 可执行部分

软件的可执行部分，包括操作系统（如 DOS、Windows、UNIX 等）、语言编译系统（如 BASIC、FORTRAN、C 语言等）、支撑程序（如 AutoCAD 绘图系统、窗口管理系统、工程数据库及其管理系统、三维几何造型系统等）和专业的应用程序（如有限元分析程序、优化设计方法包、用户的专业设计程序）。它们都是以编码信息存放在存储介质上的程序与过程。

2. 不可执行部分

软件的不可执行部分，包括面向开发者的文档和面向用户的文档两部分。这两部分文档虽不可执行，但却是可执行部分开发与维护的重要依据，是设计、制作、了解、使用、维护程序的资料和说明。

正如机械设计一样，除了生产出的机械产品与设备之外，机械设计还应该包括方案设计、总体设计、用户手册、维护手册等生产技术性文件。这些非产品性文件是技术人员和生产者、技术人员和用户之间沟通的桥梁，是在设计、制造、使用与维护中必不可少的。

人们对软件的上述认识是在经历了一定的挫折之后得出的，是从 20 世纪 60 年代发生的“软件危机”的教训中总结出来的。

二、计算机软件的发展

自 1946 年世界上第一台电子计算机诞生以来，计算机软件的发展经历了三个历史时期。

1. 程序设计时期

从 1947 年到 60 年代初，是计算机软件发展的初期。这个时期，人们最关心的是计算机能否可靠、持续地运行等问题。对于软件，还没有充分认识到它在发挥计算机效能上所占的重要位置，仅仅是把它当作在计算机上求解某一问题而必须进行的准备工作而已。因此，此时的程序设计很少考虑通用性。

到 20 世纪 60 年代初，硬件已经相当通用化，但程序设计仍是工程技术人员为解决某个实际问题而专门编写的。程序的规模很小，程序的开发者和使用者又往往是同一个人，无须向其他人作任何的交待和解释。这种个体化的软件开发环境，使得程序的开发只是开发者头脑中的一个隐含过程。程序开发的结果，除了程序流程图和源程序清单可以留下来之外，没有任何其它形式的文档资料保留下来。此时只有程序的概念，没有软件的概

念。

这个时期，由于硬件体积大，存储容量小，运算速度慢，因此十分讲究编程技巧，以解决计算机内存容量不够和运算速度太低的矛盾。由于过分追求编程技巧，因此程序除程序作者本人之外，其他人很难读懂。

该时期称为程序设计时期，其生产方式为个体手工方式，程序设计被视为某个人的神秘技巧。

2. 程序系统时期

从20世纪60年代初到70年代初，是计算机软件发展的第二个时期。这个时期，计算机硬件技术有了较大的发展，稳定性与可靠性也有了极大的提高。随着通道技术、中断技术的出现，外存储设备、人机交互设备的改进，为计算机应用领域的扩大奠定了基础。计算机从单一的科学计算，扩展到数据处理、实时控制等方面，工程界对CAD应用软件的制作要求也越来越迫切。与此同时，人们为摆脱机器码编程的困难，相继研制出了一批高级程序设计语言(如ALGOL、FORTRAN、BASIC等)。这些高级程序设计语言的出现，大大加速了计算机应用普及的步伐，各种应用程序相继出现。另外，计算机公司为了扩大系统的功能，方便用户使用，合理调度计算机资源，提高系统运行效率，也投入了大量人力、物力从事系统软件和支撑软件的开发研究。此时，无论是应用软件还是系统软件，软件的规模都比较大，各个软件成分之间的关系也比较复杂，软件的通用性也很强。因此，提出了“软件”这一概念。“软件”的概念虽然提出了，但人们对软件的认识仅仅局限于“软件＝程序＋说明”。

该时期软件开发的特征主要表现在以下三个方面：

(1) 由于程序的规模增大，程序的设计已不可能由一个人独立完成，而需要多人分工协作。软件的开发方式由“个体生产”发展到“软件作坊”。

(2) 程序的运行、维护也不再由一个人来承担。

(3) 程序已不再是计算机硬件的附属成分，而是计算机系统中与硬件相互依存、共同发挥作用的不可缺少的部分。在计算机系统的开发过程中，起主导作用的已不仅仅是硬件工程师，同时也包括软件工程师。

基于上述特征，在软件的开发过程中，所有参与人员之间的通讯和理解就成为一个主要问题。对于一个大型软件的开发，设计者、制作者、维护者与用户之间，必须有一种能达到共同理解的约定和说明；硬件工程师和软件工程师之间的配合，也需要有一个相互制约的要求和说明。

这个时期，软件规模相当大，有的软件已达到几十万甚至上百万条指令组成。软件产业已经萌芽，其中一个重要特征就是出现了“软件作坊”；软件产品广泛销售，软件数量急剧增加。

“软件作坊”就相当于“手工作坊”，它基本上沿用了软件发展早期所形成的个体化的开发方式。该时期软件规模增大，逻辑关系复杂，软件开发与维护难度很大。软件运行出错，必须修改软件；用户的需求改变，必须修改软件；硬件或操作系统更新，也应该修改软件以适应新的环境。上述种种情况，使得软件的开发与维护费用以惊人的速度递增。更为严重的是，由于程序的个体化开发特征，使得许多软件产品不可维护，最终导致出现了严重的“软件危机”。这个时期，称为程序系统时期。

3. 软件工程时期

从20世纪70年代中期至今，是计算机软件发展的第三个时期。这个时期软件产业已经兴起，软件作坊已经发展为软件公司，甚至是跨国公司。软件的开发不再是“个体化”或“手工作坊”式的开发方式，而是以工程化的思想作指导，用工程化的原则、方法和标准来开发和维护软件。软件开发的成功率大大提高，软件的质量也有了很大的保证。软件也已经产品化、系列化、标准化、工程化。这个时期，称为软件工程时期。

第二节　软件危机

一、软件危机(Software Crisis)的概念及其表现

20世纪60年代，随着计算机硬件的发展，计算机的应用领域越来越广，几乎涉及到社会生活的各个方面，如工厂管理、银行事务、学校档案、图书馆流通、旅馆预订等。这些系统的软件规模都很庞大，逻辑上也相当复杂，并且功能上要求能够不断地更改和扩充。

由于软件本身是一个逻辑实体，而非一个物理实体，因此软件是非实物性与不可见的。而软件开发本身又是一个“思考”的过程，很难进行管理。开发人员以“手工作坊”的开发方式来开发软件，完全是按照各自的爱好和习惯进行的，没有统一的标准和规范可以遵循。因而，在软件的开发过程中，人们遇到了许多困难。有的软件开发彻底失败了，有的软件虽然开发出来了，但运行的结果极不理想，如程序中包含着许多错误，每次错误修改之后又会有一批新的错误取而代之。这些软件有的因无法维护而不能满足用户的新要求，最终失败了；有的虽然完成了，但比原计划推迟了好几年，而且成本上大大超出了预算。

IBM公司开发的OS/360系统就是一个很好的例子。该系统由4 000多个模块组成，约100万条指令，人工为5 000人年(一个人年为一个人工作一年的工作量)，耗费达数亿美元。该系统投入运行后发现了2 000多个错误，而以后每个版本的更新均有1 000多个大大小小的错误存在。系统开发陷入了僵局。OS/360系统的负责人F. D. Brooms曾这样形象地描述了开发过程中的困难和混乱：“……像巨兽在泥潭中作垂死挣扎，挣扎得越猛，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……程序设计就像是这样一个泥潭……一批批程序员在泥潭中挣扎……没人料到问题竟会这样棘手……”

软件开发的高成本与软件产品的低质量之间的尖锐矛盾，终于导致了软件危机的发生。具体地说，软件危机主要有以下几方面的表现：

1. 软件的复杂性越来越高，“手工作坊”式的软件开发方式已无法满足要求

计算机软件系统的规模越来越大，逻辑功能越来越复杂。因此，程序员各自为营的“手工作坊”式的开发方式已无法完成大型软件系统的开发。

2. 软件开发的成本与进度严重估计不足

常常是实际成本远远高出估计成本，实际进度远远落后于预期进度。这种情况的存在，大大降低了软件公司的信誉。与此同时，软件开发者为了赶进度与节约成本所采取的权宜之计，又常常会降低软件产品的质量。所有这一切，引起了用户的极大不满。

3. **软件开发周期长**

计算机软件开发所需的生产周期与程序代码行不是直线递增关系，而是指数递增关系，程序代码行与所需人年的关系见表 1-1。

4. **软件成本在计算机系统总成本中所占的比例逐年上升**

由于微电子技术的进步和发展，计算机硬件的价格逐年下降，计算机软件的价格却随着软件规模和数量的增加在逐年持续地上升。美国曾为此作过一个统计（见表 1-2），1985 年，计算机软件的投资已达计算机系统总投资的 90%。

表 1-1

代码行(万行)	所需人年
10	8.3
20	83
30	170
40	250
50	380
60	500

表 1-2

年　份	软件投资占计算机系统总投资的百分比
1955	20
1970	60
1980	85
1985	90

5. **软件的维护工作量大**

软件成本在计算机系统中所占的比例越来越大，而软件成本中的 70%～80%是用于软件维护的。软件中的错误发现得越晚，修改错误所耗费的代价就越高。测试阶段改正一个错误所需的费用，大约是需求分析阶段的 100 倍。美国贝尔实验室对改正软件中的一个错误所需的费用统计见表 1-3。

表 1-3

问题修改的阶段	改正一个问题所需的费用(美元)
需求分析	20
详细设计	200
集成测试	1 000
系统测试	2 000

6. **软件没有足够的文档资料**

计算机软件不仅应有源程序，而且还应该有一套完备的文档资料。这些文档资料描述了软件开发的全过程。软件开发的组织人员和管理人员，用这些资料来管理和评价软件开发工程的进度情况，并完成与用户的交流；软件开发人员之间，则用它作为通讯工具，以达到准确地交流信息；软件维护人员，则需要用这些资料进行正确维护。

7. **软件的开发速度远远跟不上计算机应用普及的速度**

软件产品的供不应求，使人们不能充分利用计算机硬件提供的巨大潜力。

8. **软件产品质量难以保证**

软件产品质量低劣、出错率高、可靠性差，无法真正满足用户的需要。

9. **用户对“已完成的”软件系统极不满意**

一般用户是不懂计算机的专业人员，软件开发人员是不懂专业的计算机人员，用户和软件开发人员之间缺乏准确的信息交流。开发人员在没有完全理解用户的真正需求的前提下开发出的软件，又无法按照用户的实际需要进行修改，最终导致软件开发的失败。

二、软件危机产生的根源

软件危机的产生，一方面和软件本身的特性有关，另一方面和软件开发与维护的方法不当有关。

软件是计算机系统中的逻辑部分，在程序完成之前，是无法掌握进度，无法评价质量的。因此，管理和控制软件的开发极为困难。

软件开发主要涉及两方面的人员：用户和软件开发人员。用户提出问题，软件开发人员进行问题的解答。这里存在的问题是，一般情况下，用户熟悉本专业业务但不熟悉计算机，软件开发人员熟悉计算机但不了解用户的专业，这两方面的人员缺乏共同的交流语言。软件开发人员习惯用数据结构、程序结构、编程语言等方式来讨论问题，但用户却不能确切地理解这些概念。所以，双方交流时存在着隔阂。更严重的问题是，用户本身也常常不知道他究竟要计算机做些什么。这就更增加了交流的困难。而开发人员往往急于求成，在未明确软件系统应该“做什么”的情况下，就开始进行设计、编程。用户却不清楚软件人员在设计一个怎样的系统，直到系统完成交付使用之后，才发现它不符合用户想像中的要求。这一类教训，国内外都不少见。用户与软件人员之间交流困难，是造成软件危机的重要原因之一。

软件规模的增大，使程序的复杂程度大大增加，软件开发的难度难以衡量。大型软件的开发，要求人与人之间必须能够准确地进行通讯与交流，而实际情况是，用户、领域专家、软件开发人员之间没有一个合适的通讯方式。这就使得软件开发的难度更大，开发出的软件难以满足用户的需要，必须进行修改。

软件危机产生的主要原因，是软件的开发采用了“手工作坊”式的开发模式。

“手工作坊”式的开发模式，是20世纪50年代大多数软件开发所采用的模式。当时的软件规模很小，多为一个人或一组人开发，开发人员没有统一的规范标准可以遵循，只是按照开发者各自的爱好和习惯进行。软件开发的结果是开发者把方案构想、总体设计等重要设计过程完全装在大脑里，只有程序流程图和可执行的源程序作为文档留下来，缺少了必要的、不可执行的、面向开发者的文档，如软件需求说明书、结构图、程序代码中的注释、测试用例设计及测试报告等。

同机械产品和土木工程设计一样，软件作为一种产品，如果缺少必要的文档，就无法进行维护。无法进行维护的软件就不能作为产品生存下来，软件开发失败是必然的。

三、对软件危机的挽救

为摆脱软件危机，北大西洋公约组织成员国在1968年和1969年两度召开会议，商讨解决“软件危机”的对策。会议总结了软件开发中失败的经验与教训，吸收了机械工程和土木工程设计中成熟而严密的工程设计思想，首次提出了“软件工程”的概念，认为计算机软件的开发，也应像工程设计一样，进行规范性的开发，走“工程化”的道路。

当“软件工程学”这一名词刚刚问世，还处于学术研究阶段时，工程化的设计思想就对软件开发产生了巨大的影响。1971 年，IBM 公司首先运用软件工程技术，成功地开发出了纽约时报情报库系统和空间实验室的飞行模拟系统。这两个系统规模都很大，开发过程中用户又提出了许多新的要求，但在减少了人力、物力，削减了经费的情况下，这两个软件系统成功地、高质量地完成了。

从此，软件工程学这门学科很快地发展起来了。到目前为止，软件工程学已成为计算机学科中的一个重要分支。

第三节　软件工程学概述

软件工程学是指导计算机软件开发和维护的工程学科。它运用工程的概念、原理、方法、技术来开发和维护软件，把经过时间考验而证明是正确的管理技术和当前能够用到的最好的技术方法结合起来。

软件工程学强调使用软件生命周期方法学和各种先进的分析方法、设计技术。

软件工程学包括的面很广，有基础理论研究、应用研究，也有实际开发；除了技术问题之外，它还涉及到与软件开发有关的所有活动，如管理学、经济学等。本书只讨论软件工程中的技术问题，重点是软件产业近年来较流行的实用技术。

一、软件工程学的研究对象

软件工程学研究如何应用一些科学理论和工程技术来指导软件系统的开发与维护，使其成为一门严格的工程学科。

二、软件工程学的基本目标

软件工程学的基本目标在于研究一套科学的工程方法，设计一套方便实用的工具系统，以达到在软件研制生产中投资少、效率高、质量优的目的。

三、软件工程学的三要素

软件工程学的三个基本要素是方法、工具和管理。

四、软件生命周期(Software life cycle)

软件工程是研究软件的研制和维护的规律、方法和技术的学科。贯穿于这一学科的基本线索是软件生命周期学说(也叫软件生存周期)，它将告诉软件研制者与维护者“什么时候做什么、怎样做”。

一个软件项目从问题提出、定义、开发、使用、维护，直至被废弃，要经历一个漫长的时期，通常把这个时期称为软件生命周期。

如同机械产品一样，一台机器的生命周期(从开始研制到机器最终被废弃)要经过问题定义、可行性论证、方案设计、总体设计、装配图设计、零件图设计、制造、安装、测试、运行、维护等几个阶段。伴随着机器的设计制造，设计人员需要按步骤及时、认真地建立一整套图纸与设计资料，例如机器的总体方案、装配图、零件图、使用手册、维护手册等，这些

图纸与资料是研制、使用和维护机器必不可少的。

软件的开发也应如此。软件工程学，将软件的生命周期分解为问题定义、可行性研究、需求分析、总体设计、详细设计、编码与单元测试、综合测试、运行与维护几个阶段，每个阶段的任务都相对独立、简单，便于不同的人员分工协作，从而降低软件开发的难度。

在软件生命周期的每个阶段都有明确的要求、严格的标准与规范，以及与开发软件完全一致的高质量的文档资料，从而保证软件开发工程结束时有一个完整准确的软件配置交付使用。

目前划分软件生命周期的方法有很多，软件规模、种类、开发方式、开发环境及开发方法都影响软件生命周期阶段的划分。划分软件生命周期阶段应遵循的一条基本原则是各阶段的任务应尽可能相对独立，以降低每个阶段的复杂程度，简化不同阶段之间的联系，利于软件开发工程管理。一般情况下，软件生命周期由软件定义、软件开发、软件维护三个时期组成。每个时期又分为若干个阶段。

软件定义，又称为系统分析。这个时期的任务，是确定软件开发的总目标，确定软件开发工程的可行性，确定实现工程目标应该采用的策略和必须完成的功能，估计完成该项工程需要的资源和成本，制定出工程进度表。软件定义，可进一步划分为三个阶段，即问题定义、可行性研究和需求分析。

软件开发，是实现前一个时期定义的软件。它包含四个阶段：总体设计、详细设计、编码与单元测试、综合测试。

软件维护的任务，是使软件能够持久地满足用户的需求。具体地说，当软件在使用过程中发现错误，应能及时地改正；当用户在使用过程中提出新要求，应能按要求进行更新；当系统环境改变，应能对软件进行修正，以适应新的环境。

1. **问题定义**

问题定义阶段必须考虑的问题是“做什么”。

正确理解用户的真正需求，是系统开发成功的必要条件。软件开发人员与用户之间的沟通，必须通过系统分析员对用户进行访问调查，扼要地写出对问题的理解，并在有用户参加的会议上认真讨论，澄清含糊不清的地方，改正理解不正确的地方，最后得到一份双方都认可的文档。在文档中，系统分析员要写明问题的性质、工程的预期目标以及工程的规模。

问题定义阶段是软件生命周期中最短的阶段，一般不超过 3 天。

2. **可行性研究**

可行性研究要研究问题的范围，并探索这个问题是否值得去解决，以及是否有可行的解决办法。

可行性论证是分析员在收集资料的基础上，经过分析，明确工程 CAD 软件项目的目标、问题域、主要功能和性能要求，确定应用软件的支撑环境，以及经济、制作和时间限制等方面的约束条件，并用高层逻辑模型（通常用数据流图）对各种可能方案进行可行性分析及成本/效益分析。如果该项目在技术和经济上均可行，可明确地写出开发任务的全面要求和细节，形成软件计划任务书，作为本阶段的工作总结。软件计划任务书，包括工程 CAD 软件项目目标、主要功能、性能、系统的高层逻辑模型、系统界面、可供使用的资源、进度安排和成本预算。

可行性研究的结果是使用部门负责人作出是否继续这项工程决定的重要依据。

3. 需求分析

需求分析即系统分析，通常采用系统模型定义系统。这一阶段的工作相当于机械工程中的“方案设计”，土木工程中的“建筑方案设计”。

需求分析阶段的任务，主要是确定目标系统必须具备的功能。系统分析员和用户密切配合，充分交流信息，得出经过用户确认的系统逻辑模型。系统的逻辑模型，通常是用数据流图、数据词典和简要的描述表示系统的逻辑关系。

需求分析阶段仍不能具体地解决问题，只能在前一阶段的基础上，对几种可行的方案进一步分析，得出经用户确认的系统逻辑模型。根据该系统逻辑模型，准确地回答“为了解决这个问题，目标系统必须做什么”。

需求分析只是原理性方案的设计。在这一阶段的工作中，为清晰地揭示问题的本质，往往略去具体问题中的一些次要因素，只将功能关系抽象为反映该问题的系统模型。这种模型，在机器设计中用机构运动简图来表示，在土木设计中用建筑方案草图来表示。

系统逻辑模型是以后设计和实现目标系统的基础，必须准确而完整地体现用户的要求。这一阶段，特别要注意克服急于着手进行具体设计的倾向。一旦分析员开始谈论程序设计的细节，就意味着他已脱离用户，并妨碍用户继续提出要求和建议。

4. 总体设计

总体设计，也叫概要设计或初步设计。这个阶段必须回答的是“概括地说，应该如何解决这个问题”。

总体设计的目标是将需求分析阶段定义的系统模型转换成相应的软件结构，以规定软件的形态及各成分间的层次关系、界面及接口要求。这一阶段的工作，在机械设计中称为装配图设计；在土木设计中称为扩初设计。

总体设计中，由于开发者在划分系统模块时，对模块的功能、模块与模块之间的联系方式、模块内各元素之间的联系等问题处理不同，设计出的软件结构也会不同。机械设计与土木设计也是如此。在机械设计中，根据同一机构运动简图，不同的设计人员会设计出不同结构的机器。正因为如此，才会有风格各异的小轿车及其它一些机器。

总体设计应遵循的一条主要原则，就是程序模块化的原则。总体设计的结果通常以层次图或结构图来表示。

5. 详细设计

总体设计阶段以比较抽象、概括的方式提出了问题的解决方法。详细设计阶段的任务是把解法具体化，也就是回答“应该怎样具体地实现这个系统”。

详细设计亦即模块设计。它是在算法设计和结构设计的基础上，针对每个模块的功能、接口和算法定义，设计模块内部的算法过程及程序的逻辑结构，并编写模块设计说明。

这个阶段的工作还不是编程序，而是设计出程序的工序图。程序工序图通常用程序流程图、盒图、PAD 图、IPO 图或 PDL 语言来描述。它的作用类似于机械工程中的零件图、土木工程中的施工图。它们包含必要的处理细节，程序员根据它可以写出实际的程序代码，正如机械工人根据零件图可以加工出机械零件，建筑工人根据施工图可以进行建筑施工。

程序员可根据详细设计的结果进行编码设计。

6. **编码设计与单元测试**

这个阶段的任务，是根据详细设计的结果，选择一种适合的程序设计语言（必要时可采用汇编语言），把详细设计的结果翻译成程序的源代码。这一步工作就相当于机械设计中根据零件图加工零件，土木设计中根据施工图进行施工。

每编写完一个模块，都要对模块进行测试，即单元测试，以便尽早发现程序中的错误和缺陷。

7. **综合测试**

模块编码及测试完成后，需要根据软件结构进行组装，并进行各种综合测试。这一阶段的工作，相当于机械设计中根据装配图将加工与测试好的零件进行装配，并进行各种各样的综合测试。

软件测试中，测试计划、测试方案、测试用例报告及测试结果，是软件配置的一部分，应以正式的文档形式保存下来。

综合测试的目标，是产生一个可用的软件文本，修订和确认软件的使用手册。

8. **软件运行与维护**

软件产品同机械产品一样，必须经过不断地使用和维护，才能逐步地发现存在于产品中的错误或不完善的地方。只有不断地改正所发现的错误，并完善其功能，产品才能适应社会需求而得以生存和发展。

软件的使用与维护是密不可分的。维护对于确保软件正常使用、延长使用期、充分发挥其经济效益是绝对重要的。对于大型软件项目，维护是一项工程，可安排专门的人员负责，依照一套专门的方法和技术，使用一组专门的维护工具，进行软件维护。维护阶段的任务，是通过各种必要的维护措施使软件系统能持久地满足用户的需要。

维护可分为四类：纠错性维护，适应性维护，完善性维护和预防性维护。纠错性维护是对软件在使用过程发现的错误进行诊断和改正；适应性维护是为了让软件适应新的环境（如操作系统的改变、支撑环境的改变等）而进行的修改；完善性维护是为了改进和扩充软件的功能而进行的修改；预防性维护是为将来的维护活动所作的准备。

每一项维护都要以正式文档的形式记录下来，作为软件配置的一部分。

为便于理解，表 1-4 给出了软件生命周期各阶段的主要工作与机械产品开发各阶段工作的对照。

第四节　软件工程学的发展现状

随着软件工程学的不断研究和广泛应用，出现了许多新的设计思想和方法，如新的软件开发模式、软件重用、计算机辅助软件工程、软件工程的人工智能方法、软件自动生成等。

一、新的软件开发模式

传统的软件开发模式存在着不少弊端。因此，人们一直在探索新的软件开发模式。M. A. Jackson 与 F. P. Brooks 提出了适于大型软件系统的原型开发模式，后来又有人在原型开发 模式的基础上提出了智能原型模式，并预言将进一步提高软件的开发率。J. Martin提