高等学校教材·计算机教学丛书

软件工程

宋广军 黎 明
杜 鹃 王 崇 编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

高等学校教材·计算机教学丛书

软件工程

宋广军　黎　明
杜　鹃　王　崇　编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

内容简介

面对无穷无尽的计算机应用需求，软件开发已成为软件开发人员面临的主要任务。“软件工程”已成为计算机教学一门重要的专业课。本书以软件的生命周期为主线，重点讨论结构化的软件开发方法，包括结构化分析、结构化设计、编码、测试。通过对基本概念、基本原理、基本技术、基本方法的学习，使读者能很快运用工程的方法与技术开发软件。近些年来面向对象软件开发方法和技术不断普及，用最后两章的篇幅介绍面向对象的基本概念，面向对象的分析和设计方法。书中内容尽量做到通俗易懂，图文并茂，原理、方法与实例相结合。可作为高等学校计算机专业教材，也可供计算机软件人员和计算机用户参考。

图书在版编目(CIP)数据

软件工程 / 宋广军，黎明，杜鹃，王崇编著. —北京 ：北京航空航天大学出版社，2011.6

ISBN 978-7-5124-0448-9

Ⅰ. ①软… Ⅱ. ①宋… ②黎… ③杜…④王… Ⅲ. ①软件工程 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2011)第 091380 号

软件工程

宋广军 黎 明 杜 鹃 王 崇 编著

责任编辑 许传安

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：bhpress@263.net 邮购电话：(010)82316936

北京市松源印刷有限公司印装 各地书店经销

*

开本：787×1092 1/16 印张：13.5 字数：346 千字

2011 年 6 月第 1 版 2011 年 6 月第 1 次印刷 印数：3 000 册

ISBN 978-7-5124-0448-9 定价：25.00 元

总 前 言

随着科学技术、文化、教育、经济和社会的发展，计算机教学进入了我国历史上最火热的年代，欣欣向荣。就计算机专业而言，全国开办计算机本科专业的院校在2004年之初有505所，到2006年已经发展到771所。另外，在全国高校中的非计算机专业，包括理工农医以及文科（文史哲法教、经管、文艺）等专业，按各自专业的培养目标都融入了计算机课程的教学。过去出版界出版了一大批计算机教学方面的各类教材，满足了一定时期的需求，但是还不能完全适应计算机教学深化改革的要求。

面对《国家科学技术中长期发展纲要（2006年—2020年）》制订的信息技术发展目标，计算机教学也要随之进行改革，以便提高培养质量。教学要改革，教材建设必须跟上。面对各层次、各类型的学校和各类型的专业都要开设计算机课程，就应有多样化的教材，以适应各专业教学的需要。北京航空航天大学出版社是以出版高等教育教材为主的，愿对计算机教学的教材建设做出贡献。

为计算机类教材的出版，北京航空航天大学出版社成立了“高等学校教材·计算机教学丛书”编审委员会。出版计算机教材，得到了北京航空航天大学计算机学院的大力支持。该院有三位教育部高等学校计算机科学与技术教学指导委员会（下称教指委）的成员参加编审委员会的工作。其他成员是北京航空航天大学、北京交通大学等6所院校和中科院计算技术研究所对计算机教育有研究的教指委成员、专家、学者和出版社的领导。

我们组织编写、出版计算机课程教材，以大多数高校实际状况为基点，使其在现有基础上能提高一步，追求符合大多数高校本科教学适用为目标。按照教指委制订的计算机科学与技术本科专业规范和计算机基础课教学基本要求的精神，我们组织身居教学第一线，具有教学实践经验的教师进行编写。在出书品种和内容上，面对两个方面的教学。一是计算机专业本科教学，包括计算机导论、计算机专业技术基础课、计算机专业课等；二是非计算机专业的计算机基础课程的本科教学，包括理工农医类、文史哲法教类、经管类、艺术类等的计算机课程。

教材的编写注重以下几点。

1. 基础性。具有基础知识和基本理论，以使学生在专业发展上具有潜力，便于适应社会的需求。

2. 先进性。融入计算机科学与技术发展的新成果；瞄准计算机科学与技术发展的新方向，内容应具有前瞻性。这样，以使学生扩展视野，以便与科技、社会发展的脉络同步。

3. 实用性。一是适应教学的需求；二是理论与实践相结合，以使学生掌握实用技术。

编写、出版的教材能否适应教学改革的需求，只有师生在教与学的实践中做出评价，我们期望得到师生的批评和指正。

“高等学校教材·计算机教学丛书”编审委员会

“高等学校教材·计算机教学丛书”编审委员会成员

软件工程是计算机学科中指导计算机软件开发的工程科学，然而长期以来，随着微电子技术的发展，计算机硬件性能不断提高，人们开发优质软件的能力远远落后于硬件技术的发展和应用计算机软件的需求。从20世纪60年代末期开始，为了克服“软件危机”，人们在软件开发领域做了大量的工作，积累了一定的经验，逐渐形成系统的软件开发理论、方法和技术，即软件工程学。采用软件工程的方法、技术开发软件，可以提高软件的质量和数量。软件运行期间的维护工作量大大减少了。

进入21世纪信息社会，面对无穷无尽的计算机应用需求，如何更好、更快、更多、更方便地开发软件，已成为软件开发人员面临的主要任务。软件工程的方法和技术越来越受到人们的关注。它已经成为计算机学科中一个非常有价值，并具有广阔发展空间的研究领域，有良好的发展前景。

软件工程是计算机科学的一个重要分支，所涉及的范围非常广泛，包括软件开发技术、软件工程环境、软件经济学、软件心理学和软件工程管理等诸多方面的知识。本书以软件的生命周期作为主线，重点讨论结构化的软件开发方法和技术，包括结构化分析、结构化设计、编码、测试。在软件工程的入门阶段，结构化软件开发方法是最基本、实用的技术。通过对基本概念、基本原理、基本技术、基本方法的学习，使读者能很快运用工程的方法和技术开发软件。近年来，由于面向对象软件开发方法和技术的研究及应用不断普及，本书利用一定篇幅介绍了面向对象的基本概念、面向对象的分析和设计方法。面向对象方法与人类习惯的思维方式一致，符合人们认识客观世界、解决复杂问题的渐进过程；用面向对象方法设计的软件，其稳定性和可重用性好，并且易于维护，是当今比较流行的软件开发方法之一。由于软件工程是一门实践性很强的学科，书中提供了大量的范例供读者参考。书中内容尽量做到通俗易懂，图文并茂，原理、方法与实例相结合。

本书共分11章：第1章软件工程概述；第2章软件计划；第3章软件需求分析；第4章总体设计；第5章详细设计；第6章程序编码；第7章软件测试；第8章软件实施与维护；第9章软件项目管理；第10章面向对象方法学与建模；第11章面向对象设计与实现。其中，第3,9章由宋广军编写；第4,10,11章由黎明编写；第1,2,7章由杜鹃编写；第5,6,8章由王崇编写，全书由宋广军修改定稿。使用本教材的参考讲授学时为36学时，另外可安排10～20学时组织学生针对具体课题进行设计实训，以便加深对软件工程课程内容的理解和掌握。

本书可作为高等院校计算机专业教材，也可供计算机软件人员和计算机用户参考。

由于软件工程所涉及的知识面广泛，内容复杂，加上时间仓促，作者水平有限，书中不足之处恳请广大读者和专家批评指正。

编 者

2011年3月16日

第 1 章

软件工程概述

1.1　软件工程与软件危机

1.1.1　软件的发展阶段

任何一种计算机系统都包括硬件和软件两大部分。硬件只是提供了计算机的可能性，还必须有支持和管理计算机的软件，系统才能实现计算。因此，软件的发展是与硬件的发展相联系的。

自 1946 年世界上第一台计算机问世以来，计算机软件大致经历了三个发展阶段。它们是：程序设计阶段、软件系统阶段、软件工程阶段。

程序设计阶段：在 20 世纪 50～60 年代，出现了百万次每秒的大型计算机，主要用于科学研究机构。这期间各大公司主要致力于硬件的设计和生产，而软件是为某一专门的应用领域设计的。人们认为软件就是程序，主要由设计者本人利用机器语言或汇编语言，运用子程序、过程、函数等技术手段，开发的小型源程序。这类程序大多结构简单，功能单一，工作可靠性差，由设计者自行维护。此阶段的程序设计活动完全是个人程序设计技术的体现。

软件系统阶段：从 20 世纪 60 年代至 70 年代，随着硬件技术的发展，有了微机，人们可以在 PC 机上进行各种信息的处理。为了信息共享和互通信息，还发展了局域网和广域网，计算机的应用面拓宽，用户增加。在这期间，有了专门的软件公司，软件成为产品。软件的功能、规模日益增大，出现了大量的高级语言，软件的开发工作由开发小组所承担，不再是个人的艺术，满足少数用户的需求，采用的技术手段主要是结构化程序设计。在这一阶段，软件规模的增长，带来了它的复杂度的增加。软件可靠性往往随规模的增长而下降，质量保证也越来越困难。软件的发展速度远不能满足用户的需求，出现了软件危机；而硬件的迅速发展，又要求以“大程序系统”为特征的软件支持。随着计算机应用的增加，尽管提出了软件工程的方法来解决软件的开发和维护问题，但这一问题至今仍然严重地影响着软件的发展。

软件工程阶段：大约在 20 世纪 70 年代以后，以 PC 机为主的计算机已渗透到人类活动的各个领域。这期间出现了大量的兼容机和软件生产企业，除系统软件外，工具软件和应用软件门类繁多，无所不有。许多软件厂商为了使自己的软件能够适应不同的应用，就不断地增加功能，从而使软件规模越来越大，结构越来越复杂。随着软件需求的规模、数量剧增和交付要求迫切，大型程序的设计已成为工程项目。其开发工作一般由开发小组及大中型软件开发机构来完成；开发手段也日益丰富，主要包括数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络和分布式开发、对象技术等，并具备了专业维护人员，面向广大市场用户的需求。虽然开发技术有了很大的进步，但始终未有突破性进展，价格昂贵，并未完全摆脱软件危机。

1.1.2 软件危机

从软件发展的第二个阶段开始，就出现了软件危机。软件的生产不能满足日益增长的软件需求。随着计算机应用的增长，硬件技术迅速发展，出现软件供不应求的局面。更严重的是，软件的生产率随软件规模的增加和复杂性的提高而下降，导致软件的成本在计算机系统成本构成中所占比例急剧上升。庞大的软件费用，加上软件质量的下降，对计算机应用的继续扩大构成巨大的威胁。面对这种严峻的形势，有识之士发出了软件危机的警告。软件危机主要有下述一些表现。

1）系统实际功能与实际需求不符。软件开发人员缺乏对用户需求的深入了解。具体实现的功能与用户需求相差太远，导致程序上机运行时出现错误。由于软件人员和用户未能及时交换意见，使得一些问题不能及时解决，而隐蔽下来，造成开发后期矛盾的集中暴露，给将来的调试和维护工作带来更大的困难。

2）软件的维护费用急剧上升。软件的费用不仅花费在开发上，尤其要花费在维护上。由于开发阶段存在一定的隐患，因而不能保证软件运行中不再发现错误。维护最重要的事就是纠正软件中遗留的错误，此外，还要进行完善性维护和适应性维护。不言而喻，软件的规模愈大，维护的成本就愈高。

3）对软件文档配置没有足够的重视。由于开发过程没有统一的、公认的方法和规范作指导，软件文档不规范、不健全，参加的人员各行其是，忽视人与人的接口部分。发现了问题修修补补，这样的软件很难维护。提交给用户的软件质量较差，软件文档主要是开发过程各个阶段的说明书、数据字典、程序清单、软件使用、维护手册、软件测试报告及测试用例。这些文档的不完整，是造成软件开发进程，成本不可控、软件维护、管理困难的重要因素。

1.1.3 软件工程

软件工程学作为指导软件开发与维护实践的理论，是在人们为解决 20 世纪 60 年代开始出现的软件危机中逐步形成和发展起来的。“软件工程”一词，是 1968 年北大西洋公约组织(NATO)在联邦德国召开的一次会议上首次提出的。软件工程是大型软件开发所必须采用的一种重要手段。它采用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验而证明是正确的管理技术和当前能够得到的最好技术方法结合起来，称之为软件工程。

软件工程是指导计算机软件开发与维护的工程学科。其目标是追求软件产品的正确性、可用性和软件生产的效率。一般情况下，采用生命期方法和结构系统分析、结构系统设计技术来研究软件工程。

利用生命期方法学，从时间角度把软件开发和维护的复杂问题，划分成若干个阶段，每个阶段均有相对独立的任务，然后再逐步完成每个任务，并且每个阶段都要有技术和管理审查，避免反复返工。每个阶段还应有高质量的文档资料，保证软件工程结束后，交付给用户一个完整准确的软件。每一个设计任务还要严格依照结构分析设计技术来完成，以保证软件的质量，提高软件的可维护性和可靠性。

1.2　软件开发模型

1.2.1　软件生命周期

软件生命周期是一个软件系统从目标提出到最后丢弃的整个过程。生命周期是软件工程的一个重要概念。把整个生存期划分为较小的阶段，是实现软件生产工程化的重要步骤。阶段的划分使得人员分工职责清楚，项目进度控制和软件质量得到确认。原则上，前一阶段任务的完成是后一阶段工作的前提和基础；而后一阶段的任务是对于前一阶段问题求解方法的具体化。给每个阶段赋予确定而有限的任务，就能够简化每一步的工作内容，使软件复杂性变得较易控制和管理。

一般来说，软件生命周期包括计划、开发和运行三个阶段。各阶段的划分如图 1.1 所示。

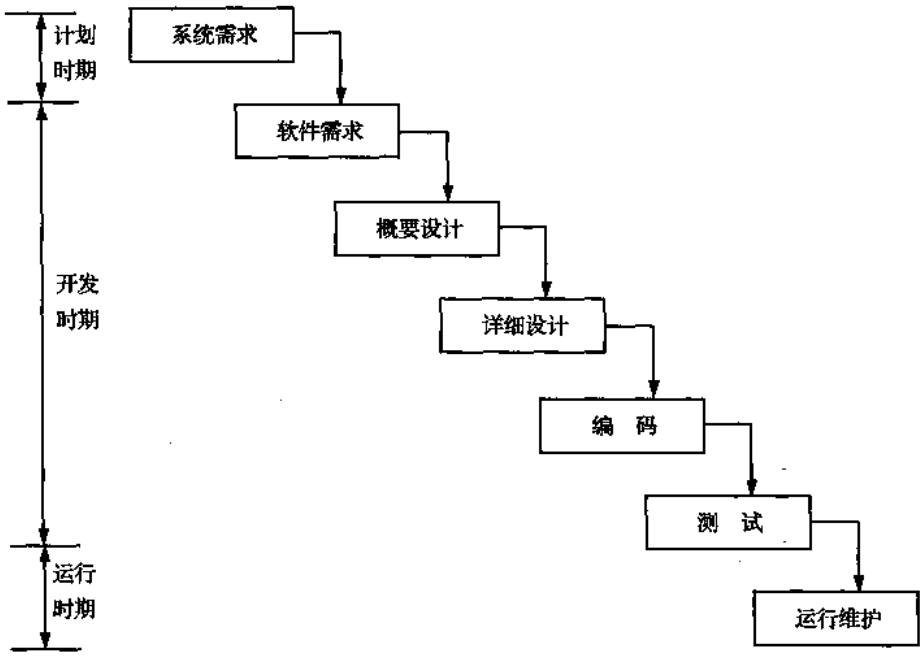

图 1.1　软件生命周期的阶段划分

1. 计划时期

计划时期的主要任务是分析用户的要求，确定软件开发的总目标，给出系统功能、性能结构、可靠性以及接口等方面的要求，由分析员和用户合作，研究完成该项软件任务的可行性，制定软件开发计划，并对可利用的资源、成本、效益、开发的进度作出估计，制定出完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查，为软件设计提供依据。因此，软件定义进一步分为问题定义和可行性研究。

(1) 问题定义

问题定义阶段是计划时期的第一步。根据用户或者市场需求，提出软件项目目标和规模，即确定“用户要计算机解决什么问题”。由系统分析员根据对问题的理解，提交关于系统目标

与范围的说明。

(2) 可行性研究

可行性研究是问题求解目标一经提出，分析员必须对它进行可行性研究。目的是为前一步提出的问题寻求一种或数种在技术上可行，且在经济上有较高效益的解决方法。为此，系统分析员应在高层次上简化需求分析和概要设计，并写出可行性论证报告。对建议实施的软件项目进行成本效益分析是可行性研究的主要任务之一。

同时，在计划时间还应制订出人力、资源及进度计划。

2. 开发时期

开发时期要完成设计和实现两大任务。其中设计任务用需求分析、概要设计和详细设计三个阶段完成。实现任务用编码和测试两个阶段完成。把设计和实现分两步，目的是在开发初期让程序人员集中全力搞好软件的逻辑结构，避免过早地为实现的细节分散精力。

(1) 需求分析

需求分析的任务是完整定义系统必须"做什么?"，并用开发人员与用户均能准确理解的语言表达出来。需求分析是软件开发的基础性工作，必须高度重视，谨慎实施。需求分析文档描述了经过用户确认的系统逻辑模型。它既是软件设计实现的依据，同时也是项目最后验收交付的依据。当采用结构化分析方法，需求规格说明书通常由数据流图、数据字典和加工说明等一整套文档组成。软件工程使用的结构分析设计的方法为每个阶段都规定了特定的结束标准。需求分析阶段必须提出完整准确的系统逻辑模型，经过用户确认之后才能进入下一个阶段。这就可以有效地防止和克服急于着手进行具体设计的倾向。

(2) 概要设计

主要任务是建立软件的总体结构，包括系统功能设计和系统结构设计。

系统功能设计的任务是确定系统的外部规格与内部规格。所谓外部规格包括：系统运行环境，用户可见功能、性能一览表，系统输入/输出数据格式。内部规格是指各主要处理的基本策略、系统文档种类与规格、系统测试总体方案。系统结构设计的任务是确定系统模块结构、确立各模块功能划分和接口规范、调用关系、确定主要模块算法和主要数据结构。所以，系统设计员应选择有经验的高级程序员担任，或直接由系统分析员兼任。总体设计说明书中系统功能设计可以采用表格形式给出，而系统结构设计通常由软件结构图或者高层 IPO 图给出。

(3) 详细设计

详细设计是针对单个模块的设计。目的是确定模块内部的过程结构，详细说明实现该模块功能的算法和数据结构，所以有时也称为算法设计。详细设计由高级程序员和程序员担任，按照系统结构将各模块分解到人。详细设计的完成标志是用图形或者伪码描述的模块设计说明书。

(4) 编 码

编码的任务是根据模块设计说明书，用指定的程序设计语言把模块的过程性描述翻译成源程序。与"需求分析"或"设计"相比，"编码"要简单得多，所以通常由编码员或初级程序员担任。系统编码的完成标志是可运行代码和完整的模块内部文档(源程序清单)。前面产生的都属于软件的文档。

(5) 测 试

测试是开发时期的最后一个阶段。其任务是通过各种类型的测试使软件达到预期的要

求。按照不同的层次，又可细分为单元测试、综合测试、确认测试和系统测试等步骤。测试是保证软件质量的重要手段。大型软件的测试通常由独立的部门和人员进行，通过测试结果的分析，要求建立系统可靠性模型，对系统可以达到的各项功能、性能指标进行量化确认。因此，测试阶段的文档称为测试报告，包括测试计划、测试用例与测试结果等内容。

3. 运行时期

运行时期是软件生命周期的最后一个时期。其主要工作是作好软件维护。维护的目的是使软件在整个生命周期内保证满足用户的需求和延长软件的使用寿命。软件维护的具体活动包括纠错维护、适应性维护、功能性维护和预防性维护。所谓纠错性维护就是改正在软件运行过程中暴露出来的系统遗留的各种错误。这种维护活动在系统交付初期比较频繁，但当系统进入稳定运行期后应该很少发生。适应性维护是指当系统运行环境发生变化以后，为适应这种改变必须对软件进行的修改。功能性维护是指在软件过程中为满足用户需求的变化与扩充对软件所做的修改。预防性维护则是指为改善软件将来的可维护性所做的准备工作。软件运行稳定以后，维护的主要活动应该是适应性和功能性维护。

虽然没有把维护阶段进一步划分成更小的阶段，但是实际上每一项维护活动都应该经过提出维护要求、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，因此实质上是经历了一次压缩和简化了的软件定义和开发的全过程。

1.2.2　软件开发的瀑布模型

瀑布(waterfall)模型也称软件生存周期模型，由 W. Royce 于 1970 年首先提出。根据软件生存周期各个阶段的任务，瀑布模型从可行性研究(或称系统分析)，逐步进行阶段性变换，直至通过确认测试，并得到用户确认的软件产品为止。瀑布模型上一阶段的变换结果是下一阶段变换的输入，相邻两个阶段具有因果关系，紧密相连。一个阶段工作的失误将蔓延到以后的各个阶段。为了保障软件开发的正确性，每个阶段任务完成后，都必须对它的阶段性产品进行评审，确认之后再转入下一阶段的工作。评审过程发现错误和疏漏后，应该反馈到前面的有关阶段修正错误、弥补疏漏，然后再重复前面的工作，直至某一阶段通过评审后再进入下一阶段。这种形式的瀑布模型是带有反馈的瀑布模型(见图 1.2 所示)。

严格按照软件生命周期的阶段划分，顺序执行各阶段构成软件开发的瀑布型模型。它是传统的软件工程生存期模式，如图 1.2 所示。

1. 瀑布模型的特点

(1) 段间具有顺序性和依赖性

顺序性要求每个阶段工作开始的前提是其上一阶段工作结束。前一阶段的输出文档，是后一阶段的输入文档。依赖性是指各阶段工作正确性依赖与上一阶段工作的正确性。因此，如果在生命周期某一阶段出现了问题，往往要追朔到在它之前的一些阶段，必要时还要修改前面已经完成的文档。

(2) 推迟实现的观点

软件开发人员接受任务后，往往急于求成，总想早一点开始编写程序。编码开始得愈早，完成所需的时间反而愈长。这是因为过早地考虑程序的实现，常常导致大量的返工，有时甚至给开发人员带来灾难性的后果。

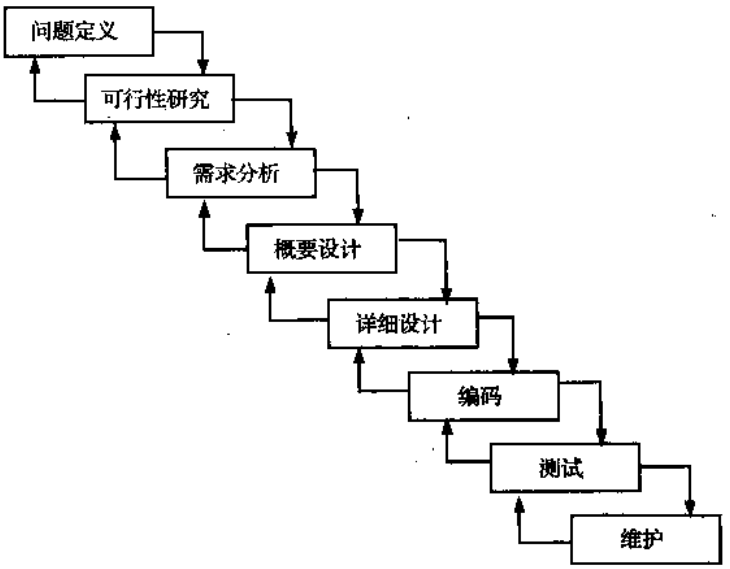

图 1.2 软件开发的瀑布型模型

瀑布模型明确要求在项目前期，即分析阶段和设计阶段只考虑系统的逻辑模型，不涉及系统的物理实现。因此，直到设计结束，设计员主要考虑系统的逻辑模型。将逻辑设计和物理设计清楚地划分开来，尽可能推迟程序的物理实现，是瀑布型软件的一条重要的指导思想。

(3) 质量保证的观点

优质与高产，是软件工程的重要目标。瀑布型软件开发为了保证质量，在各阶段强调：

每一阶段都要完成规定的文档。没有完成文档，就认为没有完成该阶段的任务。完整、准确的文档即是软件开发过程中各类人员之间相互通信的媒介，也是将来软件维护的重要依据。生命周期各阶段与相应文档如图 1.3 所示。

每一阶段都要对已完成的文档进行复审，以便尽早发现问题，消除隐患。愈是早期潜伏的故障，暴露的时间愈晚，排除故障付出的代价也愈高。对每阶段文档的复审，是保证软件质量，降低成本的重要措施。

2. 瀑布模型的缺点

实践证明，这种生存期模型有许多的缺陷，可能对软件项目产生负面影响。而这种模型又不能完全抛弃。在某些领域中，它是最合理的方法，比如嵌入式软件和实时控制系统。但是，对更多的其他应用领域，特别是对商业数据处理，它是不适用的，存在很多缺点，主要有以下各项。

- 它不能对付含糊不清和不完整的用户需求。
- 由于开销的逐步升级问题，它不希望存在早期阶段的反馈。
- 在一个系统完成以前，它无法预测一个新系统引入一个机构的影响。
- 它不能恰当地研究和解决使用系统时人为因素。
- 如果突然地把一个计算机的系统引入一个机构是很危险的，因为用户会抵制这种突然的变革。
- 在用户实际有一个可供使用的系统之前，可能不得不等待很长时间。这可能给用户的

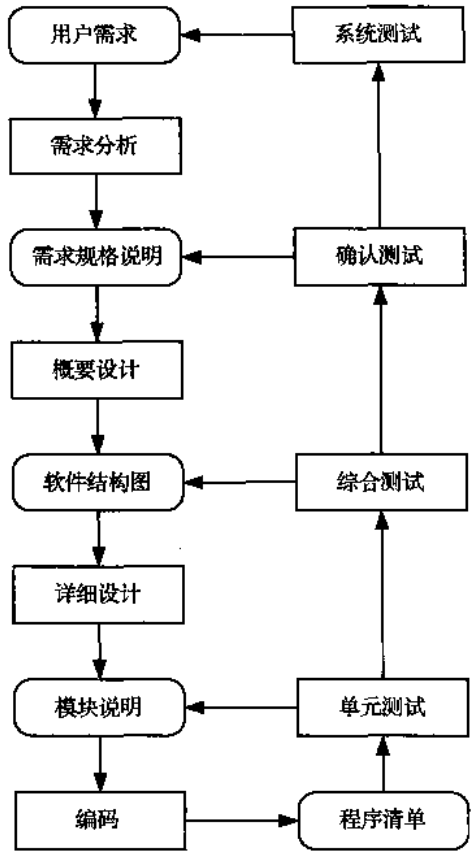

图 1.3　各阶段产生的文档和相互关系

信任程度带来意想不到的影响，也可能导致打击。

- 最终产品将更多的反映用户在项目开始时的需求，而不是最后的需求。
- 一旦用户开始使用最终的系统，并对系统有更多的学习以后，他们的观点和意向会发生很大的变化，用户的这种变化常常是无法预测的。

前面介绍了瀑布型生命周期各个开发阶段的任务，而隐含在瀑布模型各阶段任务后面的指导思想和观点是软件工程的根本性质。因为只有明确了这些观点，才能在软件开发中发挥更大的自觉性和主动性，使软件工程方法得到恰当的应用。

1.2.3　原型化开发模型

瀑布型模型的缺陷在于软件开发阶段推进是直线型的。工程实践说明这是一个“理想化”模型，不完全符合人们认识问题的规律。按照这一模型来开发软件，只有当分析员能够作出准确的需求分析时，才能够得到预期的正确结果。不幸的是，由于多数用户不熟悉计算机，系统分析员对用户的专业也往往了解不深，在计划时期定义的用户需求，常常是不完全和不准确的。对于用户和分析员都未经历过的新的系统。这种情况就更加突出。鉴于瀑布型模型的种种缺陷，许多研究人员得出这样的结论：软件开发，特别是开发的早期阶段，应该是一个学习和实践的过程。它的活动应该包括开发人员和用户两个方面。为了使其更有效，不仅要求开发人员要与用户紧密合作，而且还要有一个实际的工作系统，只要这样才能获得成功。尽管用户

在开始时说不清楚所要求的未来软件系统是什么样，但他们却可以对现有系统非常熟练地进行挑剔。

为了克服上述的缺点，提出了原型化的软件开发。它的主要思想是：首先建立一个能够反映用户主要需求的原型，使得用户和开发者可以对目标系统的概貌进行评价、判断。然后对原型进行若干轮反复的扩充、改进、求精，最终建立完全符合用户需求的目标系统。原型开发模型的开发过程如图 1.4 所示。

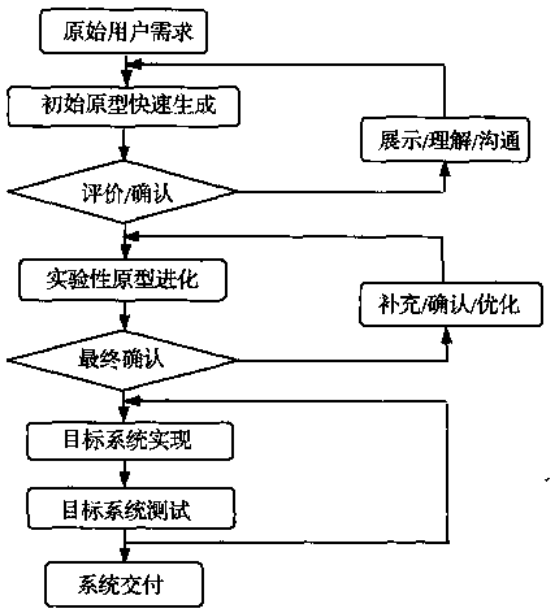

图 1.4　原型开发过程

1. 原型模型

这种开发模型又称“快速原型模型”。它是在开发真实系统之前，构造一个原型，在该原型的基础上，逐渐完成整个系统的开发工作。根据原型的不同作用，有三类原型模型：

(1) 探索型原型

这种类型的原型模型是把原型用于开发的需求分析阶段。目的是要弄清用户的需求，确定所期望的特性，并探索各种方案的可行性。它主要是针对开发目标模糊，用户与开发者对项目缺乏经验的情况，通过对原型的开发来明确用户的需求。

(2) 实验型原型

这种原型主要用于设计阶段，考核现实方案是否合适，能否实现。对于一个大型的系统，若对设计方案没有把握时，可通过对这种原型来证实设计方案的正确性。

(3) 演化型原型

这种原型主要用于及早向用户提交一个原型系统。该原型系统或者包含系统的框架，或者包含系统的主要功能，在得到用户的认可后，将原型系统不断扩充演变为最终的软件系统。它将原型的设想扩展到软件开发的全过程。

2. 原型开发过程

(1) 原型构造要求

原型不同于最终系统。最终系统对每个软件需求都要求详细实现，而原型仅仅是为了试验和演示用的。部分功能需求可以忽略或者模拟实现。

因此，在构造原型时，必须注意功能性能的取舍，忽略一切暂时不关心的部分以加速原型的实现，同时又要充分体现原型的作用，满足评价原型的要求。

(2) 原型的特征分类

根据原型目的和方式的不同，构造原型的内容的取舍不同，体现出原型特征有如下类别：

1) 系统的界面形式，用原型来解决系统的人机交互界面的结构；

2) 系统的总体结构，用原型来确定系统的体系结构；

3) 系统的主要处理功能和性能，用原型来实现系统的主要功能和性能；