高职高专计算机类专业“十二五”规划教材

软件工程

RUANJIAN GONGCHENG

杨志宏　主　编
庄晋林　杨雅军　副主编

化学工业出版社

软件工程

RUANJIAN GONGCHENG

ISBN 978-7-122-16121-5
9 787122 161215 >
定价：39.00元

销售分类建议：计算机

高职高专计算机类专业“十二五”规划教材

软 件 工 程

杨志宏　　主　编
庄晋林　杨雅军　副主编

化学工业出版社
·北京·

本书从软件开发、维护和管理等方面，系统地介绍了软件工程的概念、原理、过程及主要方法，按照软件生存周期依次讲述了软件开发的可行性分析、项目计划、需求分析、系统设计、软件实现、软件测试与调试、软件运行与维护，对数据库、面向对象技术以及软件项目的管理进行了介绍。

本书采用案例式教学，理论与实践紧密结合，内容翔实，既注重基本知识的表述，又注重内容的先进性、科学性和系统性，反映软件工程、软件开发技术发展的最新成果，实用性、可操作性强。

本书可作为高职高专计算机类人才培养的专业教材，也可作为本科生的教学、参考用书，还可作为计算机爱好者的自学用书。

图书在版编目（CIP）数据

软件工程/杨志宏主编. —北京：化学工业出版社，2013.1
高职高专计算机类专业“十二五”规划教材
ISBN 978-7-122-16121-5

Ⅰ.①软… Ⅱ.①杨… Ⅲ.①软件工程-高等职业教育-教材 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字（2012）第 304320 号

责任编辑：刘 哲　　装帧设计：刘丽华
责任校对：宋 玮

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号 邮政编码 100011）
印　　刷：北京永鑫印刷有限责任公司
装　　订：三河市万龙印装有限公司
787mm×1092mm 1/16 印张 17¾ 字数 447 千字 2013 年 3 月北京第 1 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899
网　　址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：39.00 元

前　言

2012年4月，工业和信息化部发布《软件和信息技术服务业“十二五”发展规划》。规划中提出：软件和信息技术服务业是关系国民经济和社会发展全局的基础性、战略性、先导性产业，具有技术更新快、产品附加值高、应用领域广、渗透能力强、资源消耗低、人力资源利用充分等突出特点，对经济社会发展具有重要的支撑和引领作用。发展和提升软件和信息技术服务业，对于推动信息化和工业化深度融合，培育和发展战略性新兴产业，建设创新型国家，加快经济发展方式转变和产业结构调整，提高国家信息安全保障能力和国际竞争力，具有重要意义。

目前，软件产业已经成为国际竞争的焦点和各国竞相发展的产业。要适应从工业社会到信息社会的转化，需要有一大批高素质、高水平的软件人才。软件开发的技术水平是决定整个软件产业发展的关键因素，软件生产至关重要。软件工程学是计算机学科领域中研究软件开发的一个重要分支学科，软件工程利用工程学的原理和方法来组织和管理软件生产，以保证软件产品的质量，提高软件生产率。我们编写本教材就是按照规划所提出的要求，在多年从事软件工程教学和软件工程科研实践的基础上进行的。

计算机软件技术发展非常快，目前越来越多的实用软件具有不同程度的自动编程功能。随着计算机硬件、软件技术的发展，软件工程各阶段的自动化程度也将越来越高。如何正确安排软件的结构，合理组织、管理软件的生产，不仅仅是专业从事软件开发人员的事，广大计算机应用人员也需掌握这方面的知识。本书吸取了国内外同类教材的优点，抓住软件工程理论中最基础的知识点组织编写，从实用、够用的角度出发，以医院门诊取药管理系统为主线，详细讲述了软件工程的基本原理、概念、技术和方法。

本书共分10章，内容包括：软件工程的时代背景、理论基础及软件开发的可行性分析；软件项目的需求分析；软件项目的总体设计和详细设计方法；软件项目的实现；软件项目的测试技术；软件的维护和软件项目管理等；以及如何书写整个软件项目的开发总结性材料。

全书编写工作分配如下：第1章由张瑞霞、杨志宏编写，第2章和第6章由杨雅军编写，第3章由张晓红、张瑞霞合编，第4章由张晓红、郭改文编写，第5章和第10章由杨志宏编写，第7章由庄晋林、石燕合编，第8章、第9章由郭改文编写，由杨志宏、庄晋林负责统稿。

由于编者水平有限，书中难免有不妥之处，恳请读者批评指正。

编　者

2012年11月

目　录

第 1 章　软件及其可行性分析

为了高效率地开发一个软件项目，并确保软件使用和维护的便捷性，应该采用工程方法对其进行开发管理，即采用软件工程的系统思想、方法进行开发和管理。本章首先介绍软件的特点、软件的生命周期、软件过程的模型及软件工程的基本概念，使读者对软件工程的相关知识有一个初步了解。通过本书的项目案例“医院门诊取药管理系统”的可行性调研分析，使读者了解软件项目开发的第一个主要环节。

1.1　软件与软件危机

自第一台电子计算机问世以来，计算机技术日趋成熟，运算速度不断提高，存储容量不断扩大，从单机运行已经发展到网络环境。计算机硬件技术的每次突破，都为软件技术的发展提供了更加广阔的空间，开拓了更新、更广阔的应用领域。随着新的电子元器件的出现，计算机硬件的性能和质量逐年提高，并且价格大幅度下降，使得计算机的应用几乎遍及社会的各个领域，成为人们日常工作和生活不可缺少的工具。人们对软件的需求也急剧增加，软件系统也从简单发展到复杂，从小型发展到大型，由封闭系统发展到开放系统。在软件应用需求发展的过程中，软件开发方法也从注意技巧发展为注重管理，力图在可接受的性价比条件下，不断改进个人和软件开发组织的开发过程，强调在各自条件下追求软件过程的改进。

1.1.1　软件的特点

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序能正常操纵信息的数据结构；文档是与程序开发、维护和使用有关的图文材料。

软件具有以下特点。

① 软件是一种逻辑实体，而不是具体的物理实体，因而它具有抽象性。这个特点使它与计算机硬件、或其他工程对象有着明显的差别。人们可以把它记录在介质上，但却无法看到软件的形态，而必须通过测试、分析、思考、判断等行为去了解它的功能、性能及其他特性。

② 软件与硬件的生产方式不同，它没有明显的制造过程。对软件的质量控制，必须着重在软件开发方面下工夫。一旦某一软件项目研制成功，以后就可以大量地复制同一内容的副本。即其研制成本远远大于其生产成本。软件故障往往是在开发时产生而在测试时没有被发现的问题。所以要保证软件的质量，必须着重于软件开发过程，加强管理和减少故障。

③ 在软件的运行和使用期间，没有像硬件那样的机械磨损、老化问题。

④ 软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性，在软件的开发和运行中必须以硬件提供的条件为基础。为了消除这种依赖关系，在软件开发中提出了软件移植的问题，并且把软件的可移植性作为衡量软件质量的因素之一。

⑤ 软件的开发尚未完全摆脱手工的开发方式。由于传统的手工开发方式仍然占据统治地位，软件开发的效率受到很大的限制。因此，应促进软件技术发展，提出和采用新的开发方法。例如近年来出现的充分利用现有软件的复用技术、自动生成技术和其他一些有效的软

件开发工具或软件开发环境，既方便了软件开发的质量控制，也提高了软件的开发效率。

⑥ 软件本身是复杂的。软件的复杂性可能来自它所反映的实际问题的复杂性，也可能来自程序逻辑结构的复杂性。

⑦ 软件成本相当昂贵。软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它的成本是比较高的。

⑧ 相当多的软件工作涉及到社会因素。许多软件的开发和运行涉及机构、体制及管理方式等问题，甚至涉及到人们的观念和心理。它直接影响到项目的成败。

注意 软件不仅包括程序，还包括文档。所以软件的开发不仅仅是编程序，还包括编写相关的文档。下面先看一下软件发展的历史和一些相关的数据。

1.1.2 软件发展简史

(1) 个体手工方式时期 这时的软件通常是规模较小的程序，编写者和使用者往往是同一个（或同一组）人，几乎没有什么系统化的标准方法可遵循。对软件的开发没有任何管理方法，一旦任务超时或者成本提高，程序员才开始弥补。这种个体化的软件开发方式，使得程序的开发结果，只有程序框图和源程序清单可以保留下来。同时因为硬件体积大，存储容量小，运算速度慢，所以特别讲究编程技巧，以解决计算机内存容量不足和运算速度太低的矛盾。由于过分追求编程技巧，开发出的程序除作者之外，其他人很难读懂。所以这个时期，也称为个体手工方式时期。

(2) 软件作坊时期 计算机系统发展的第二阶段（20 世纪 60 年代中期到 70 年代末期）为软件作坊时期。这个时期，软件规模相当大，软件产业已经萌芽，软件产品广泛销售，软件数量急剧增加。

(3) 软件工程时期 计算机系统发展的第三阶段始于 20 世纪 70 年代末期，并跨越了近 20 年，称为软件工程阶段。在这一阶段，以软件的产品化、系列化、工程化、标准化为特征的软件产业发展起来，打破了软件生产的个体化特征，有了软件工程化的设计原则、方法、标准可以遵循。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。

(4) 现代软件工程时期 随着互联网的应用与普及，软件开发已经不再着重于单台计算机系统和程序，而是面向计算机和软件的综合影响。由复杂的操作系统控制的强大的桌面机、广域网络和局域网络，配以先进的软件应用已成为标准。计算机体系结构迅速地从集中的主机环境转变为分布的客户/服务器环境。随着第四阶段的发展，一些新技术开始出现，面向对象技术将在许多领域中迅速取代传统软件开发方法。

1.1.3 软件危机

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这些问题绝不仅仅是不能正常运行的软件才具有的，实际上，几乎所有软件都不同程度地存在这些问题。这是因为软件本身是一个逻辑实体，开发过程是一个“渐进的”思考过程，很难进行管理。开发者根据自己的喜好和习惯来编写程序，没有统一的标准和规范可以遵循。

(1) 软件危机的主要表现

① 软件的复杂度越来越高，传统的软件开发方式已无法满足要求。计算机软件系统的规模和复杂程度，已经不是程序员各自为战的“手工作坊”式的生产方式可以解决的。

② 对软件开发成本和进度的估计常常很不准确。实际成本比估计成本有可能高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质

量，从而不可避免地会引起用户的不满。

③ 用户对软件的满意度差。软件开发人员常常在对用户的要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”必然导致最终的产品不符合用户的实际需要。

④ 软件产品的质量往往靠不住。软件可靠性和质量保证的确切的定量概念刚刚出现不久，软件质量保证技术（审查、复审和测试）还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

⑤ 软件常常是不可维护的。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的硬件环境，也不能根据用户的需要在原有程序中增加一些新的功能。“可重用（或复用）的软件”还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

⑥ 软件通常没有适当的文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新式的”（即和程序代码完全一致的）。软件开发组织的管理人员可以使用这些文档资料作为“里程碑”，来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

⑦ 软件成本在计算机系统总成本中所占比例持续上升。由于微电子学技术的进步和生产自动化程度不断提高，硬件成本逐年下降，然而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。

⑧ 软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

（2）软件危机产生的原因　在软件开发和维护的过程中存在这么多严重问题，一方面与软件本身的特点有关，另一方面也和软件开发与维护的方法不正确有关。

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；软件不会因使用时间过长而“老化”或“用坏”；软件具有可运行的行为特性，在写出程序代码并在计算机上试运行之前，软件开发过程的进展情况较难衡量，软件质量也较难评价，因此管理和控制软件开发过程十分困难；在运行时所出现的软件错误几乎都是在开发时期就存在而一直未被发现的，改正这类错误通常意味着改正或修改原来的设计，这就在客观上使得软件维护远比硬件维护困难；随着计算机应用领域的扩大，99％的软件应用需求已不再是定义良好的数值计算问题，而是难以精确描述且富于变化的非数值型应用问题。因此，当人们的应用需求变化发展的时候，往往要求通过改变软件来使计算机系统满足新的需求，维护用户业务的延续性。

危机原因来自于软件开发人员的如下弱点：其一，软件产品是人的思维结果，因此软件生产水平最终在相当程度上取决于软件人员的教育、训练和经验的积累；其二，对于大型软件，往往需要许多人合作开发，甚至要求软件开发人员深入应用领域的问题研究，这样就需要在用户与软件人员之间以及软件开发人员之间相互通讯，在此过程中难免发生理解的差异，从而导致后续错误的设计或实现，而要消除这些误解和错误往往需要付出巨大的代价；其三，由于计算机技术和应用发展迅速，知识更新周期加快，软件开发人员经常处在变化之中，不仅需要适应硬件更新的变化，而且还要涉及日益扩大的应用领域问题研究，软件开发人员所进行的每一项软件开发几乎都必须调整自身的知识结构以适应新的问题求解的需要，而这种调整是人所固有的学习行为，难以用工具来代替。

软件本身独有的特点确实给开发和维护带来一些客观困难，但是人们在开发和使用计算机系统的长期实践中，也确实积累和总结出了许多成功的经验。如果坚持不懈地使用经过实践检验证明是正确的方法，许多困难是完全可以克服的，过去也确实有一些成功的范例。但是，目前相当多的软件专业人员对软件开发和维护还有不少模糊观念，在实践过程中或多或少地采用了错误的方法和技术，这可能是使软件问题发展成软件危机的主要原因。

(3) 消除软件危机的途径

首先，采用工程化方法和途径来开发与维护软件。软件开发是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目，必须充分吸取和借鉴人类长期以来从事各种工程项目积累的行之有效的原理、概念、技术和方法。应该推广使用在实践中总结出来的开发软件的成功技术和方法，并且研究探索更好更有效的技术和方法，尽快消除在计算机系统早期发展阶段形成的一些错误概念和做法。将软件的生成问题在时间上分成若干阶段以便于分步而有计划地分工合作，在结构上简化成若干逻辑模块。把软件作为工程产品来处理，按计划、分析、设计、实现、测试、维护的周期来进行生产。

其次，应该开发和使用更好的软件工具。在软件开发的每个阶段都有许多繁琐重复的工作需要做，在适当的软件工具辅助下，开发人员可以把这类工作做得既快又好。如果把各个阶段使用的软件工具有机地集合成一个整体，支持软件开发的全过程，则称为软件工程支撑环境。

最后，采取必要的管理措施。软件产品是把思维、概念、算法、组织、流程、效率、质量等多方面问题融为一体的产品。但它本身是无形的，所以有别于一般的工程项目的管理。它必须通过人员组织管理、项目计划管理、配置管理等来保证软件按时高质量完成。

总之，为了解决软件危机，既要有技术措施（包括方法和工具），又要有必要的组织管理措施。软件工程正是从管理和技术两方面研究如何更好地开发和维护计算机软件的一门学科。

1.2 软件生命周期

软件工程采用的生命周期方法就是从时间角度对软件的开发与维护这个复杂的问题进行分解，将软件漫长的生命时期分为若干阶段，每个阶段都有其相对独立的任务，然后逐步完成各个阶段的任务。概括地说，软件生命周期由软件定义、软件开发和软件维护（也称为运行维护）三个时期组成，每个时期又进一步划分成若干个阶段。

1.2.1 软件定义

软件定义时期的任务是：确定软件开发工程必须完成的总目标；确定工程的可行性；导出实现工程目标应该采用的策略及系统必须完成的功能；估计完成该项工程需要的资源和成本，并且制定工程进度表。这个时期的工作通常又称为系统分析，由系统分析员负责完成。软件定义时期通常进一步划分成四个阶段，即问题定义、可行性分析、开发计划和需求分析。

(1) 问题定义　软件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。通过对客户的访问调查，系统分析员扼要地写出关于问题性质、工程目标和工程规模的书面报告，经过讨论和必要的修改之后，这份报告应该得到客户的确认。问题定义阶段是软件生存周期中最短的阶段。

(2) 可行性分析　可行性分析主要研究问题的范围，并探索这个问题是否值得去解决，

以及是否有可行的解决方法。为了回答这些问题，系统分析员需要进行一次大大压缩和简化了的系统分析和设计过程，也就是在较抽象的高层次上进行的分析和设计过程。可行性研究应该比较简短，这个阶段的任务不是具体解决问题。

可行性分析的结果是使用部门负责人乃至高层领导者做出是否继续这项工程的重要依据。

(3) 开发计划　开发计划是在对项目进行可行性分析之后，对项目的开发过程做一个整体规划，也称为实施计划，主要包括项目开发技术计划、资源计划、进度计划及管理计划，其是软件开发工作的方向标。

(4) 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，该阶段的主要任务是确定目标系统必须具备的功能。用户了解他们所面对的问题，知道必须做什么，但通常不能完整准确地表达出他们的要求，更不知道怎样利用计算机解决他们的问题；软件开发人员知道怎样用软件实现人们的要求，但是对特定用户的具体要求并不完全清楚。因此，系统分析员在需求分析阶段必须和用户密切配合，充分交流信息，以得出经过用户确认的系统逻辑模型。通常用数据流图、数据字典和简要的算法表示系统的逻辑模型。

这个阶段的一项重要任务，是用正式文档准确地记录对目标系统的需求，这份文档通常称为需求规格说明书。

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问题和建议。

需要注意的是，在实际从事软件开发工作时，软件规模、种类、开发环境及开发时使用的技术方法等因素，都影响阶段的划分。事实上，承担的软件项目不同，应该完成的任务也有差异，没有一个适用于所有软件项目的任务集合。适用于大型复杂项目的任务集合，对于小型简单项目而言往往就过于复杂了。

1.2.2　软件开发

开发时期具体设计和实现在前一个时期定义的软件，它通常由下述四个阶段组成：概要设计，详细设计，编码和单元测试，综合测试。其中，前两个阶段又称为系统设计，后两个阶段又称为系统实现。

(1) 概要设计　概要设计，也叫总体设计或初步设计。概要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，用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。应该设计出实现目标系统的几种可能的方案。通常至少应该设计出低成本、中等成本和高成本等三种方案。软件工程设计师应该用适当的表达工具描述每种方案，分析每种方案的优缺点，并在充分权衡各种方案的利弊的基础上，推荐一个最佳方案。此外，还应该制定出实现最佳方案的详细计划。

软件设计的一条基本原理就是，程序应该模块化，也就是说，一个程序应该由若干个规模适中的模块按合理的层次结构组织而成。因此，概要设计的另一项主要任务就是设计程序的体系结构，也就是确定程序由哪些模块组成以及模块间的关系。概要设计的结果通常以层次图或结构图来表示。

(2) 详细设计　概要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决方法具体化，也就是应该怎样具体地实现这个系统。详细设计也称为模块设计，在这个阶段系统分析师将详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。

这个阶段的任务还不是编写程序，而是设计出程序的详细规格说明。这种规格说明

的作用类似于其他工程领域中工程师经常使用的工程蓝图，它们应该包含必要的细节，通常用程序流程图、盒图、PAD图或PDL语言来描述，程序员可以根据它们写出实际的程序代码。

(3) 编码与单元测试　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，把详细设计的结果翻译成正确的、容易理解的、容易维护的程序的源代码。

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。

(4) 综合测试　模块编码及单元测试完成后，需要根据软件结构进行组装，并进行各种综合测试。通过对软件测试结果的分析可以预测软件的可靠性；反之，根据对软件可靠性的要求，也可以决定测试和调试（纠错）过程什么时候可以结束。

应该用正式的文档资料把测试计划、详细测试方案以及实际测试结果保存下来，作为软件配置的一个组成部分。

1.2.3　软件维护

软件产品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和不完善的地方。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。具体地说，当软件在使用过程中发现错误时应该加以改正；当环境改变时应该修改软件以适应新的环境；当用户有新要求时应该及时改进软件以满足用户的新需要。

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（或报告问题）、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。

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。

1.3　软件过程模型

软件过程是指制作软件产品的一组活动及其结果。这些活动主要由软件工程人员完成。软件过程模型是从一个特定角度提出的软件过程的简化描述，就是对被描述的实际过程的抽象，它包括构成软件过程的各种活动、软件产品以及软件工程参与人员的不同角色。

通用的软件过程模型是以软件生命周期各阶段为基础，为了反映软件生命周期内各种工作应如何组织及各个阶段应如何衔接而给出的直观的图示表达。软件过程模型是软件工程思想的具体化，是实施于过程模型中的软件开发方法和工具，是在软件开发实践中总结出来的软件开发方法和步骤。总的说来，软件过程模型是跨越整个软件生存周期的系统开发、运行、维护所实施的全部工作和任务的结构框架。

1.3.1　瀑布模型

瀑布（waterfallmodel）模型也称软件生存周期模型，由W. Royce于1970年首先提出。在20世纪80年代之前，瀑布模型一直是唯一被广泛采用的生命周期模型，现在它仍然是软件工程中应用得最广泛的过程模型。根据软件生存周期各个阶段的任务，瀑布模型从需求分析开始，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，每一阶段任务完

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。传统软件工程方法学的软件过程，基本上可以用瀑布模型来描述。图 1-1 所示为传统的瀑布模型。

传统的瀑布模型过于理想化了，事实上，人在工作过程中不可能不犯错误。在设计阶段可能发现需求规格说明文档中的错误，而设计上的缺陷或错误可能在实现过程中显现出来，在综合测试阶段将发现需求分析、设计或编码阶段的许多错误。当在后续的评审过程中发现错误和疏漏后，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，如图 1-2 所示。当在后面阶段发现前面阶段的错误时，需要沿图中向上的反馈线返回前面的阶段，修正前面阶段的产品之后再回来继续完成后面阶段的任务。

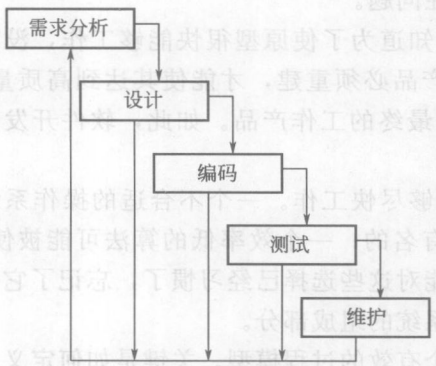

图 1-1　瀑布模型

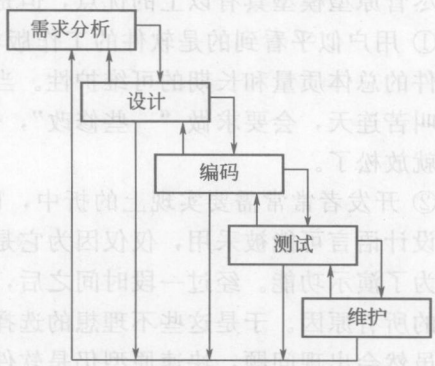

图 1-2　带反馈的瀑布模型

瀑布模型在支持结构化软件开发、控制软件开发的复杂性、促进软件开发工程化等方面起着显著作用，它有利于大型软件开发过程中人员的组织、管理，有利于软件开发方法和工具的研究与使用，从而提高了大型软件项目开发的质量和效率。与此同时，瀑布模型在大量软件开发实践中也逐渐暴露出它的缺点。其中主要缺点有以下两个方面。

① 在软件开发的初始阶段指明软件系统的全部需求是困难的，有时甚至是不现实的。而瀑布模型在需求分析阶段要求客户和系统分析人员必须做到这一点才能开始后续工作。

② 需求确定后，用户和软件项目负责人要等相当长的时间（经过设计、编码、测试、运行）才能得到一份软件的最初版本。如果用户对这个软件提出比较大的修改意见，那么整个软件项目将会蒙受巨大的人力、财力和时间方面的损失。

1.3.2　快速原型模型

常有这种情况，用户定义了软件的一组一般性目标，但不能标识出详细的输入、处理及输出需求；还有一些情况，开发者可能不能确定算法的有效性、操作系统的适应性或人机交互的形式。在这些及很多其他情况下，快速原型模型可能是最好的选择。

快速原型模型如图 1-3 所示，从需求分析开始，软件开发者和用户在一起定义软件的总目标、说明需求并规划出定义的区域，然后用快速设计软件中对用户/客

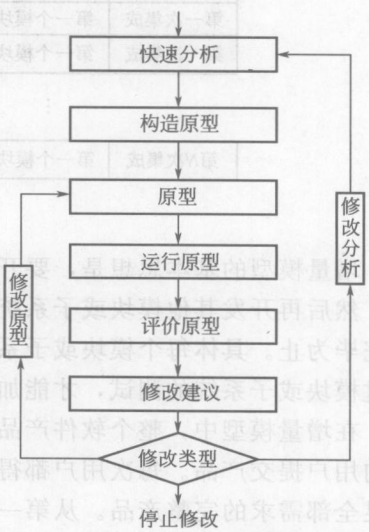

图 1-3　快速原型模型