21 世纪高职高专计算机专业教材

软 件 工 程

刘欣怡　周跃东　田秀丽　编著

清 华 大 学 出 版 社
北京交通大学出版社
·北京·

内 容 简 介

软件工程是一门指导软件开发和维护的新兴工程学科。软件工程不仅是计算机有关专业的必修课程，也是从事计算机软件开发及应用人员所必备的知识和技能。

本书是编者多年来在大学讲授软件工程和从事软件工程项目开发时教学和科研实践经验的总结。全书共分 10 章，包括软件工程的基本概念、结构化分析与设计、原型化开发方法、面向对象分析与设计、统一建模语言 UML 基础、软件测试、软件维护、软件工程环境与工具、软件质量保证与软件质量度量、软件管理。

本书可作为高职高专计算机相关专业教材，也可作为各类大专院校师生的参考书。考虑到使用本教材参加全国计算机四级等级考试的读者的需要，各章内容及习题皆参考有关要求编写。

图书在版编目 (CIP) 数据

软件工程/刘欣怡，周跃东，田秀丽编著. —北京：清华大学出版社；北京交通大学出版社，2007.10
(21 世纪高职高专计算机专业教材)
ISBN 978－7－81123－142－7

Ⅰ. 软… Ⅱ. ①刘… ②周… ③田… Ⅲ. 软件工程－高等学校：技术学校－教材
Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字 (2007) 第 117910 号

责任编辑：谭文芳　　特邀编辑：高振宇
出版发行：清 华 大 学 出 版 社　　邮编：100084　　电话：010－62776969
　　北京交通大学出版社　　邮编：100044　　电话：010－51686414
印 刷 者：
经　　销：全国新华书店
开　　本：185×260　　印张：15　　字数：384 千字
版　　次：2007 年 10 月第 1 版　　2007 年 10 月第 1 次印刷
书　　号：ISBN 978－7－81123－142－7/TP·372
印　　数：1～5000 册　　定价：25.00 元

前　言

计算机软件工程学（简称软件工程）是一门指导软件开发和维护的新兴工程学科，研究如何用工程化的方式有效地管理软件开发，以期“高产优质”地获得软件产品。作为 IT 产业的一个支柱，软件工程这一学科已逐渐为人们所熟悉和广泛应用。现在公认：如果哪个软件项目不遵循软件工程原则来进行开发，则必定会受到实践的惩罚。相当多的 IT 专业的毕业生认为，软件工程课是他们参加工作后最能直接应用的一门专业课。软件工程不仅是计算机有关专业的必修课程，也是从事计算机软件开发及应用人员所必备的知识和技能。

本书是编者多年来在大学讲授软件工程和从事软件工程项目开发时教学和科研实践经验的总结，并参阅了不少有关软件工程的著作和教材。在编写中，努力做到概念引出自然、内涵与外延适中、深入浅出、通俗易懂，在内容取舍、文字描述、习题选择方面力求“实际、实用、实践”和“必需、够用、先进”。本书可作为高职高专计算机相关专业教材，也可作为各类大专院校师生的参考书。考虑到使用本教材参加全国计算机四级等级考试读者的需要，各章内容及习题皆参考有关要求编写。

学以致用，学用结合，这既是学习软件工程的目的，也是学好软件工程的主要方法。因此，本书中采用了两大教学方法：

⑴ 任务驱动法　提出问题，解决问题（方法），归纳分析（必要的结论和概念）；

⑵ 案例教学法　从实际到理论，从具体到抽象，从个别到一般，从零散到系统。

全书各重要部分都贯穿一个完整的软件项目开发过程，让读者带着任务学习，通过案例来掌握概念、方法、工具和过程。在各个小部分穿插小练习题，使读者能更主动地学习并及时掌握有关知识。在各大部分学完后，通过马上布置类似案例的大练习题，让读者自己动手来进行相应的开发工作，最终达到让读者“会用软件工程的方法开发软件项目”的目的（而不仅仅是知道一些概念而已）。

本书共 10 章，内容涵盖了软件工程的基本内容，并使软件工程的基本原理与实践相结合。包括软件工程的基本概念，结构化分析与设计，原型化开发方法，面向对象分析与设计，统一建模语言 UML 基础，软件测试，软件维护，软件工程环境与工具，软件质量保证与软件质量度量，软件管理。

本课程参考教学学时为 48～64 学时。前期课程为高级语言程序设计、数据结构、操作系统、数据库技术等。

本书由刘欣怡、周跃东和田秀丽编著，由刘欣怡负责全书的策划、统稿与定稿工作。

刘欣怡编写第 1 章、第 2 章、第 6 章和第 7 章，周跃东编写第 4 章、第 5 章、第 8 章、第 9 章和第 10 章，田秀丽编写第 3 章。

由于编者水平有限，时间仓促，错误与不妥之处在所难免，恳请读者与专家批评指正。

目　　录

第 1 章　软件工程的基本概念

本章对软件工程作一个简短的概述，从而对软件工程的基本思想、原理、方法等有概括的本质的认识。其中，应着重理解软件工程过程的实质，它的基本思想是：系统地有条不紊地从抽象的逻辑概念逐步发展到具体的物理实现，这是软件工程的关键。

知识点

(1) 软件；
(2) 软件危机；
(3) 软件工程；
(4) 软件工程管理；
(5) 软件工程方法；
(6) 软件工程技术；
(7) 软件工程工具；
(8) 软件工程环境；
(9) 软件生命周期；
(10) 软件过程模型。

重点

(1) 软件工程；
(2) 软件过程模型。

难点

软件过程模型。

要求

掌握：
(1) 软件危机的产生、表现及原因；
(2) 软件工程的概念、软件生命周期与软件过程模型。

了解：
(1) 软件概念的产生与发展、软件的分类与特点；
(2) 软件工程管理；
(3) 软件工程的方法、技术、工具与环境。

1.1　软件与软件危机

1.1.1　软件

1. 软件概念的产生

自世界上第一台电子计算机 (ENIAC,1946 年 2 月 14 日,美国宾夕法尼亚大学) 诞生时起,就有了程序的概念:人们通过在计算机上运行自己编写的机器指令程序或汇编语言程序,指挥计算机硬件做有关工作,从而完成有关任务,得到期望的结果。这时,程序的生产方式是自给自足的个体手工方式 (类似个人自己建造一个“茅草屋”自用)。程序设计是一种任人发挥才能的技术领域,程序设计的过程是在一个人的头脑中完成的,程序的质量完全取决于个人的编程技术,程序的写法可以不受任何约束;只有通篇充满了编程技巧,使用了许多窍门,最终达到运行速度快、存储容量小的目标的程序才是高水平的好程序;这样的程序常常很难被别人看懂,也不要求易被别人看懂,因为都是些小程序,使用范围也极小 (只限于编程者本人或有关的少数人),修改完善它也是编程者自己的事情。

这时的程序就是软件,软件中只包含程序,除程序清单外,无其他文档资料。

2. 软件的发展

在经历了几十年的发展后,人们对软件有了更为深刻的认识。计算机软件经历了 3 个发展时期:

⑴ 程序设计时期,约为 20 世纪 50 年代至 20 世纪 60 年代;

⑵ 程序系统时期,约为 20 世纪 60 年代至 20 世纪 70 年代;

⑶ 软件工程时期,约为 20 世纪 70 年代至今。

有关具体内容见表 1－1。

表 1－1　计算机软件发展的 3 个时期及其特点

时期 / 特点	程序设计 (20 世纪 50—60 年代)	程序系统 (20 世纪 60—70 年代)	软件工程 (20 世纪 70 年代至今)
软件内容	程序	程序、程序说明书	程序、文档、数据
主要程序设计语言	汇编语言、机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括程序设计和测试	软件生存期
软件需求者	程序设计者本人	少数用户	市场用户
开发软件者	个人	开发小组	开发小组、大中型软件开发机构
软件规模	小型	中、小型	大、中、小型
决定软件质量的因素	个人编程技术水平	开发小组技术水平	软件工程管理水平
软件开发方法和技术等	子程序 程序库	结构化程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术等
软件维护者	程序设计者	开发小组	专职维护人员

续表

特点 \ 时期	程序设计 (20 世纪 50—60 年代)	程序系统 (20 世纪 60—70 年代)	软件工程 (20 世纪 70 年代至今)
硬件特征	价格高 存储容量小 工作可靠性差	降价、容量、速度及工作可靠性有明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视,在计算机系统中处于从属地位	软件开发方法、技术的发展不能满足社会发展需要,出现软件危机	软件开发方法、技术有进步,但未获突破性进展,价格高,未完全摆脱软件危机

* 注:软件语言包括需求定义语言、软件功能语言、软件设计语言、程序设计语言等。

由表 1-1 可以看出,软件发展到现在,其最根本的变化体现在:人们对软件有了新的认识。随着计算机应用的逐步扩大,软件在计算机系统中的比重越来越大,软件需求量迅速增加,软件规模也日益扩大,长达数万行、数十万行乃至百万行以上的软件,已不鲜见。例如,一个大型电话交换机软件超过四百万行,一个宇宙飞船软件多达两千万行。

显然,程序已从个人按自己意图搭建的"茅草屋"转变为工程队建造的能为广大用户接受的"住宅楼"。而对于这些不再微小的并在较长时间内为许多人使用的程序,绝不可能再任由编程者"孤芳自赏",而是要求这些程序易懂、易用、易修改和扩充。这时,程序中难以理解的部分(特别是技巧)成了有害的东西。为消除它们,就要求软件的内容决不能再仅仅是程序,必须包含程序、文档、数据 3 大部分。其含义如下。

⑴ 程序:是为完成预定的功能和性能,按既定算法,用某种计算机语言编写的指令(语句)序列。

⑵ 文档:是程序开发、使用和维护时所必需的阐述性的图文资料。

⑶ 数据:是使程序能正常操纵信息的数据结构。

现在,对软件的正确理解应该是:计算机软件是计算机系统中与硬件相互依存的部分,是软件工程师设计和建造的产品;它由在任意规模和体系结构的计算机系统中执行的程序、文档及表示数字、文本、图形、图像、视频和音频等的数据组成。

3. 软件的分类

在软件发展过程中,有各种分类法,从应用角度,现在通常可把软件分为以下 8 类。

⑴ 系统软件:如操作系统、数据库管理系统、设备驱动程序、通信处理程序等。

⑵ 工程和科学计算软件:如材料计算软件、数学计算软件、建模软件、信号处理软件等。

⑶ 实时软件:如实时操作系统、卫星实时监控软件、锅炉控制系统软件、外汇实时行情软件等。

⑷ 管理信息处理软件:如人力资源管理软件、财务信息管理软件、民航售票系统软件、医院信息管理软件等。

⑸ 嵌入式软件:就是嵌入在硬件中的操作系统和开发工具软件,可细分成系统软件、支撑软件、应用软件 3 类,是嵌入式系统的重要组成部分。嵌入式软件广泛应用于国防、工控、家用、商用、办公、医疗等领域,如移动电话、掌上电脑、数码相机、机顶盒、MP4 等,都是用嵌入式软件技术对传统产品进行智能化改造的结果。

⑹ 个人计算机应用软件:如办公自动化软件、计算机辅助教学软件、防/反计算机病毒软件、游戏软件等。

⑺ 基于 Web 的软件 如 Web 服务器软件、网络游戏软件、在线考试系统、网络银行等。

⑻ 人工智能软件 如专家系统、机器博弈软件、模式识别软件、智能机器人软件等。

4. 软件的特点

软件与硬件有完全不同的特征,表现在如下 5 个方面:

- 软件是逻辑实体,而不是有形的物理元件;
- 软件是设计和开发的,而不是传统意义上被制造的;
- 软件在运行和使用中不会磨损,但它存在退化问题;
- 软件特别是大型软件系统非常复杂,因此软件的研制工作要投入大量、复杂、高强度的脑力劳动,它的开发成本相当昂贵;
- 虽然软件产业目前正在向基于构件组装 (类似于搭积木) 的方向发展,但软件开发至今仍未摆脱手工生产方式,大多数软件产品必须得"定做"。

1.1.2　软件危机

由前述的软件发展历程可以看出,20 世纪 60 年代,软件进入第二个发展时期,软件在计算机系统中已不再处于从属地位。随着计算机的广泛使用,对软件的需求越来越大,软件规模也越来越大。软件在整个计算机系统中所占份额逐年上升,产品软件已被广泛使用;但软件生产方法仍是"软件作坊"式的个体化开发方法。人们发现,研制软件系统需要投入大量的人力和物力;但系统的质量却难以保证,也就是说,软件开发所需的高成本同产品的低质量之间存在尖锐的矛盾。

1963 年至 1966 年,美国 IBM 公司在开发 IBM360 机的操作系统时这些矛盾最为突出。这一项目的工作量是 5 000 人/年,有时 1 000 人投入开发工作,共约 100 万条指令,结果却非常糟糕。每次发行的新版本都是从前一版本中找出 1 000 个程序错误,在花费了上千人/年的开发成本及不断修正后,该操作系统终因错误过多、性能不稳定而被放弃。该项目负责人 F. D. Brook在总结时沉痛地说:"……像巨兽在泥潭中作垂死挣扎,挣扎得越猛,泥浆就沾得越多,最后没有一只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……"

显然,这时软件生产的复杂性和高成本已导致软件的生产和维护出现了很大的困难,即出现了软件危机 (software crisis)。其典型表现是 软件交付延期、费用超支、质量无法保证。

软件危机的具体表现如下:

- 软件需求增长得不到满足;
- 软件生产成本高,价格昂贵;
- 软件生产进度无法控制;
- 软件需求定义不准确,易偏离用户需求;
- 软件质量不易保证;
- 软件可维护性差。

归结起来,软件危机主要表现在两方面:一方面是无法满足对软件日益增长的需求;另一方面是难以满足对已有的软件系统的维护需要。

软件危机的出现,使得人们去寻找产生危机的内在原因。其原因可归纳为两方面:一方面是由于软件生产本身存在着复杂性,这是由软件的特点决定的;另一方面是与软件开发所使用

的方法和技术有关,“软件作坊”式的软件开发方法及技术已不能满足社会发展需要。

通过实践人们发现,将传统工程学(如建筑工程学)的原理、技术和方法应用于软件开发,可以起到使软件生产规范化的作用,有利于组织软件生产、提高开发质量、降低成本和控制进度。由此产生了“软件生产工程化”的思想——软件工程(1968 年,北大西洋公约组织,计算机科学国际会议)。

软件工程正是为克服软件危机而提出的一种概念,并在实践中不断地探索其原理、技术和方法。在此过程中,人们研究和借鉴了工程学的某些原理和方法,形成了一门新的学科——软件工程学。但时至今日,人们并没有完全克服软件危机。

课堂练习(单选题)

1. 软件危机是软件产业化过程中出现的一种现象,通常是指在计算机软件开发和维护中所产生的一系列严重的问题,这些问题中相对次要的因素是(　　)。

(A) 软件功能　　(B) 文档质量　　(C) 开发效率　　(D) 软件性能

2. 软件生产的复杂性和高成本,使软件的生产出现危机,下述哪些是软件危机的主要表现(　　)。

(1) 需求增长难以满足　(2) 生产成本过高　(3) 进度难以控制　(4) 质量难以保证

(A) (1) 和 (2)　　(B) (4)　　(C) (2) 和 (3)　　(D) 全部

3. 造成软件危机的主要原因是(　　)。

(1) 用户使用不当　(2) 软件本身特点　(3) 硬件不可靠　(4) 对软件的错误认识

(5) 缺乏好的开发方法和手段

(A) (1) 和 (3)　　(B) (1)、(2) 和 (4)　　(C) (3) 和 (4)　　(D) (2) 和 (5)

1.2 软件工程

软件工程(学)是一门交叉学科,是应用计算机科学、数学、工程科学及管理科学等原理,进行软件开发的工程学科,它借鉴传统工程的原则、方法,以提高软件质量、降低软件开发成本为目的。其中,计算机科学、数学用于构造模型和算法;工程科学用于制定规范、设计范型、评估成本及确定权衡;管理科学用于计划、资源、质量、成本等管理。

行业标准化组织美国电气和电子工程师协会(Institute Electrical and Electronics Engineers,IEEE)在(IEE93)中给出了软件工程的定义:

(1) 将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护,即将工程化应用于软件;

(2) 对(1)中所述方法的研究。

软件工程的目标就是研制与生产出具有良好质量和费用合理的软件产品。要达到此目标,软件工程应包括过程、管理和技术方法、工具及环境。

(1) 软件工程的过程:定义了一组关键过程域(KPA)的框架,它们构成了软件项目管理的基础,并规定了技术方法的采用、工程产品(模型、文档、数据、报告、表格等)的产生、里程碑的建立、质量保证及变更的管理。

(2) 软件工程的方法:提供了软件在技术上应该“如何做”,包括项目计划与估算、需求分析、设计、编程、测试和维护等一系列任务。

⑶ 软件工程的工具：对过程和方法提供了自动或半自动的支持。

⑷ 软件工程的环境：当把多个软件工具集成起来，使得工具之间产生的信息可以共享利用，就建立起一种称之为计算机辅助软件工程（CASE）的软件开发支撑环境。

上述的过程、方法和工具常称为软件工程三要素，即：要掌握任何软件工程技能，都必须从这三方面依次着手进行学习。

课堂练习（单选题）

1．软件工程学的提出是由于软件生产中的软件危机引起的，软件工程学的目的应该是最终解决软件生产的什么问题（　　）。

Ⓐ 消除软件的生产危机　　Ⓑ 加强软件的质量保证

Ⓒ 提高软件的开发效率　　Ⓓ 使软件生产工程化

2．软件工程三要素包括（　　）。

Ⓐ 过程、方法和管理　　Ⓑ 过程、管理和工具

Ⓒ 过程、方法和工具　　Ⓓ 管理、技术和方法

1.3　软件生命周期

软件生命周期（Software life cycle）是人们在研究软件生产时所发现的一种规律性的事实。

一切工业产品都有自己的生命周期，软件（产品）也不例外。与工业产品生产一样，软件产品或软件系统也有一个从生产、使用到被淘汰的过程，被称为软件的生命周期，即一个计算机软件从功能确定、设计，到开发成功投入使用，并在使用中不断地修改、增补和完善，直至被新的需要所替代而停止该软件的使用为止的全过程。

软件生命周期是软件工程的一个重要概念。而把整个软件生命周期划分为较小的阶段，是实现软件生产工程化的重要步骤。给每个阶段赋予确定然而有限的任务，就能够简化每一步的工作内容，使因为软件规模增长而大大增加了的软件复杂性变得较易控制和管理。

从总体上分析，软件生命周期包括分析、设计、实现和维护等一系列过程。显然，软件生命周期的划分应该适应软件生产工程化的需要，而不应该是千篇一律的。而对于软件生命周期的不同划分方法就会形成不同的软件生命周期模型。

20 世纪 70 年代，B．W．Boehm 提出了软件生命周期的瀑布模型（Waterfall model），它较典型地刻画了软件生命周期的阶段划分。

瀑布模型将软件生命周期划分为 8 个阶段，每个阶段的任务分别是：问题定义、可行性研究、需求分析、概要设计、详细设计、编码、测试、运行和维护。

8 个阶段又可以归纳为 3 个大的阶段，即计划定义阶段、开发阶段和运行维护阶段。

其各阶段的工作按顺序开展，形如自上而下的瀑布，故人们将这种开发模式称之为瀑布模型，如图 1－1 所示。

瀑布模型的计划定义阶段包括问题定义、可行性研究；开发阶段包括需求分析、概要设计、详细设计、编码和测试；运行维护阶段包括系统的运行和维护。

瀑布模型 8 个阶段的主要任务如下。

⑴ 问题定义：标识要解决的特定问题。

⑵ 可行性研究：从技术、经济、开发方案、运行等方面研究其可行性。

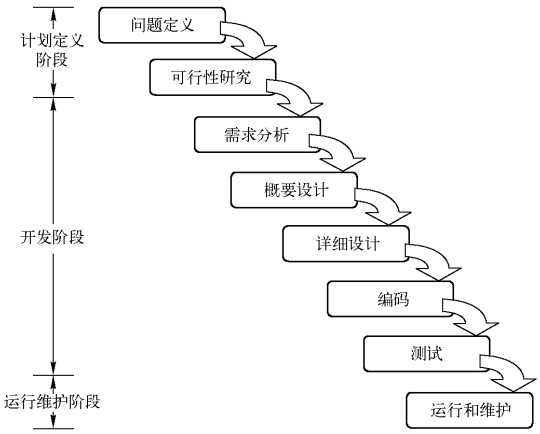

图 1－1 瀑布模型

⑶ 需求分析 :弄清用户要求,解决“做什么”,生成软件的功能和性能规约。

⑷ 概要设计 :建立整个软件体系结构,包括子系统、模块及相关层次的说明、每一模块的接口定义。

⑸ 详细设计 :产生程序员可用的模块说明,即数据结构说明及加工描述。

⑹ 编码 :把设计结果转换为可执行的程序代码。

⑺ 测试 :包括单元测试、组装测试和确认测试,这些测试活动的目的是使软件系统达到需求分析时提出的各项要求。

⑻ 运行和维护 :是对投入运行的软件进行修改,使软件系统能适应外界环境的变化、实现功能扩充和质量改善。

课堂练习 (单选题)

1. 瀑布模型的软件生命周期有三大阶段,即计划定义、开发及 (　　)。

⑷ 可行性研究　　⑻ 详细设计

⑹ 测试　　⑽ 运行维护

2. 在瀑布模型中,以下哪个环节出错,对软件的影响最大 (　　)。

⑷ 问题定义　　⑻ 需求分析

⑹ 概要设计与详细设计　　⑽ 编码与测试

1.4 软件过程模型

软件过程模型是软件开发全部过程、活动和任务的结构框架。软件开发包括分析、设计、编码和测试等阶段,有时也包括维护阶段。

软件过程模型能清晰、直观地表达软件开发全过程,能明确规定要完成的主要活动和任务 ;因此软件过程模型被用来作为软件项目工作的基础。软件过程模型应允许对于不同的应用系统采用不同的开发手段和方法,使用不同的程序设计语言及不同技能的人员参与工作 ;它

还应允许采用不同的软件工程工具或不同的软件工程环境。对于这所有的不同,模型都应该是稳定有效和普遍适用的。

到目前为止,已经提出了多种软件过程模型。目前常见的有以下 4 种。

1.4.1　瀑布模型

最早出现的软件过程模型是图 1－1 所示的瀑布模型。该模型给出了固定的顺序,将软件生命周期活动从上一阶段向下一阶段逐级过渡,如同流水下泻,最终得到所开发的软件产品,投入使用。

但实践表明,各个阶段间的关系并非如此简单。可能某阶段的工作结果无法通过阶段评审,则会返回前项甚至更前项的活动进行返工,即出现向前阶段的反馈,致使在各阶段间产生环路,即瀑布流水出现上流,如图 1－2 所示。

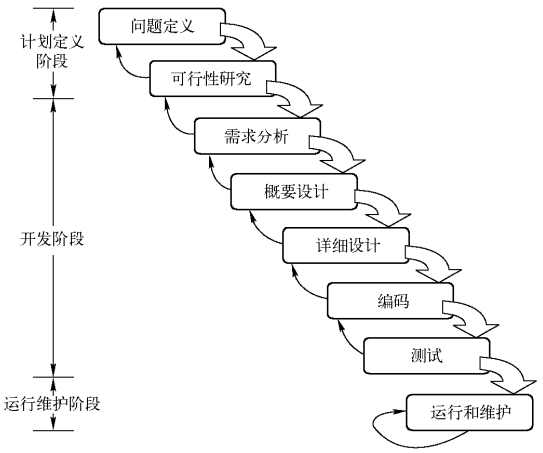

图 1－2　改进的瀑布模型

瀑布模型的优点是：

- 符合软件开发的思维过程并且容易掌握和运用；
- 为软件开发与维护提供了一种有效的管理模式,根据这一模式制订开发计划、进行成本预算、组织开发人员,以阶段评审和文档控制为手段有效地对整个开发过程进行指导,从而保证了软件产品的质量；
- 在支持开发结构化软件、控制软件的开发复杂度、促进软件开发工程化方面起了不少作用；
- 适合于在软件需求比较明确、开发技术比较成熟、工程管理比较严格的场合下使用。

因此,几十年来瀑布模型广为流行,在实际项目中,瀑布模型是非常有效的,需要去掌握。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,其中最为突出的缺点如下：

⑴ 该模型缺乏灵活性,无法在开发过程中再更改软件需求；

⑵ 该模型缺乏演化性,无法通过开发活动澄清本来不够确切的软件需求。