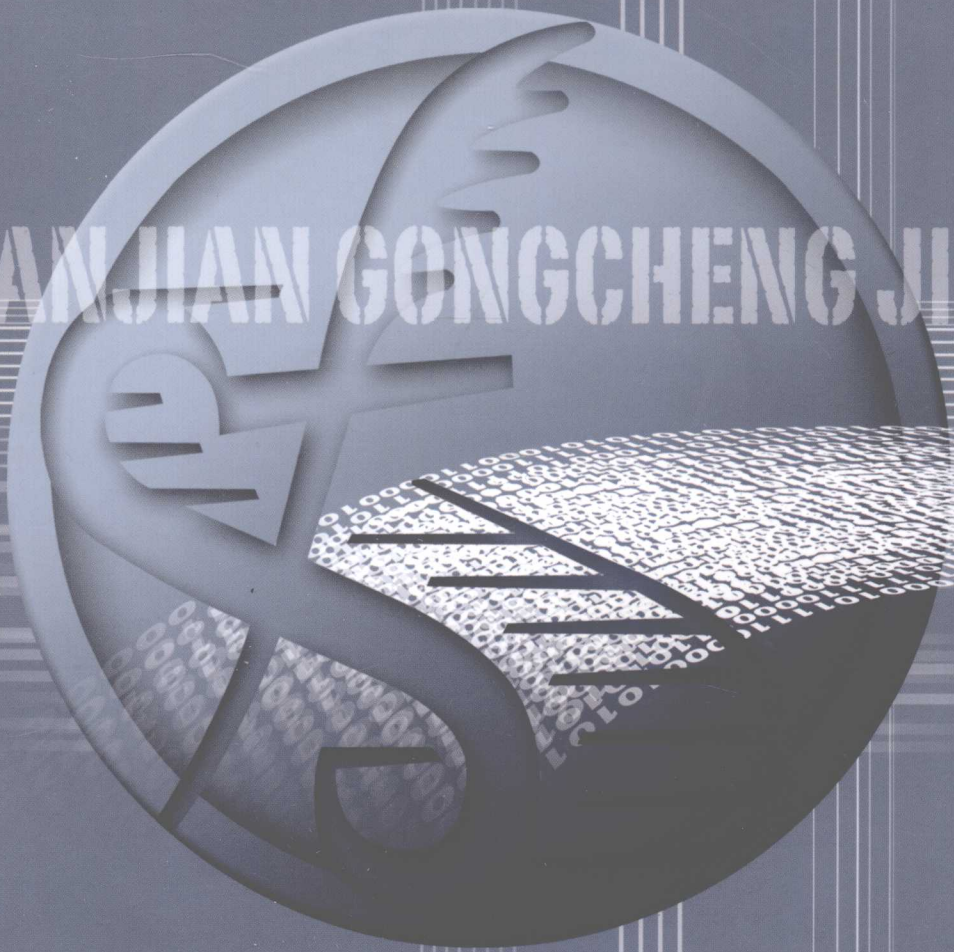
浙江省高等教育重点教材

软件工程基础

RUANJIAN GONGCHENG JICHU



编著 周苏 王文 吴艳

浙江科学技术出版社

浙江省高等教育重点教材

软件工程基础

编著 周苏 王文 吴艳

RUANJIAN GONGCHENG JICHU

浙江科学技术出版社

图书在版编目(CIP)数据

软件工程基础/周苏，王文，吴艳编著. —杭州：浙江科学技术出版社，2008.8
（浙江省高等教育重点教材）
ISBN 978-7-5341-3371-8

Ⅰ.软… Ⅱ.①周…②王…③吴… Ⅲ.软件工程—高等学校—教材 Ⅳ.TP311.5

中国版本图书馆CIP数据核字(2008)第110456号

丛 书 名 浙江省高等教育重点教材
书　　名 **软件工程基础**
编　　著 周　苏　王　文　吴　艳

出版发行 浙江科学技术出版社
杭州市体育场路347号　邮政编码：310006
联系电话：0571-85170300-61712
E-mail：cl@zkpress.com
排　　版 杭州大漠照排印刷有限公司制作
印　　刷 杭州丰源印刷有限公司

开　　本 787×1092　1/16　　印张　22.75
字　　数 505 000
版　　次 2008年8月第1版　2008年8月第1次印刷
书　　号 ISBN 978-7-5341-3371-8　　定价：39.50元

策划组稿 张祝娟　　**封面设计** 金　晖
责任校对 张　宁　　**责任印务** 李　静

前　言

软件工程是一门理论性和实践性都很强的学科，它采用工程化的概念、理论、技术和方法来指导计算机软件的开发与维护，它主要研究软件结构、软件设计方法、软件工具、软件工程标准和规范以及软件工程的有关理论。采用工程化的概念、原理、技术和方法来开发与维护软件，把经过时间考验证明是正确的管理技术和当前能够得到的最好的开发方法结合起来，这就是软件工程。

高等教育的新形势需要我们积极进行教学改革，研究和探索新的教学方法。在长期的教学实践中，我们体会到，坚持“因材施教”的重要原则，把实验实践环节与理论教学相融合，抓实验实践教学促进学科理论知识的学习，是有效地改善教学效果和提高教学水平的重要方法之一。

本书是浙江省高等教育重点教材，是全新设计编写的具有较强实践性的高等院校“软件工程”课程教材。本书共分 14 章，针对计算机和其他 IT 专业学生的发展需求，系统、全面地介绍了软件工程的概念、原理、方法及其应用，详细介绍了软件生存周期、面向对象软件过程和软件过程工程的思想和实现方法，力图反映软件工程领域的最新发展，具有较强的系统性和可读性。

本书的主要特色是：理论联系实际，把软件工程的概念、理论和技术知识融入到实践当中，使学生保持浓厚的学习热情，加深对软件工程知识的认识、理解和掌握；按照一系列软件工程国家标准来表达和描述软件工程的知识，使软件工程技术具有很强的可操作性。

在本书的设计编写中，包含了以下几个愿望：

- 通过主要基于因特网的实验活动，培养学生在网络环境中自主学习的能力。
- 通过针对 Visio、WinRunner、PowerDesigner、Project 和 Visual SourceSafe 等常用软件工具的学习和实验活动，培养学生的动手能力。
- 通过广泛的专业文章和教材阅读，培养学生探究性学习、理性思考和创新思维的能力。

本书由周苏、王文、吴艳老师编著，参加本书编写的还有朱勇、魏金岭等老师。本书的编撰和出版得到了浙江省教育厅的扶持和资助，得到了浙江科学技术出版社、浙江大学城市学院、浙江工业大学之江学院、浙江商业职业技术学院等单位的大力支持，在此一并表示感谢！欢迎教师索取与本书教学配套的相关资料。邮箱地址：zs@mail.hz.zj.cn；QQ：81505050；个人教学网站：www.zhousu.net。

周　苏

2008 年新春于西子湖畔

目　录

第一章　软件工程概述

随着计算机系统的迅速发展和应用范围的日益广泛，计算机软件的规模越来越大，其复杂程度也在不断增加。进入20世纪60年代以来，人们开始逐渐认识到确实存在着“软件危机”这样一个事实。例如：

(1) 软件生产不能满足日益增长的需要。

(2) 对软件开发成本和开发进度的估计往往不准确，实际成本有时高出预计成本好几倍，预计完工的时间往往推迟几个月，甚至更长时间。

(3) 软件开发人员和用户之间信息交流不充分，用户对完成的软件满意度很低。

(4) 软件价格昂贵，软件成本在整个计算机系统中所占的比例急剧上升，软件已成为许多计算机系统中花钱最多的项目。

(5) 软件质量难以保证，质量保证技术还没有真正应用到软件开发的全过程。

(6) 软件可维护性差，程序中的错误很难改正，或者当硬件环境发生变化时，想要进行适应性或完善性维护极其困难。

导致这一系列问题产生的一个重要原因就是，由于软件的研制和维护本身是工程性的任务，而软件人员所采取的方式却未能工程化。

为了克服“软件危机”，人们开始考虑采用工程化方法和工程途径来研制和维护软件。20世纪60年代末至70年代初开始，逐渐发展起一组总称为“软件工程”的技术。这些技术把软件作为一个工程产品来处理，它需要计划、分析、设计、实现、测试以及维护。

第一节　计算机系统及其软件的发展

硬件工程和软件工程都可以看成是一门更广义的学科——计算机系统工程的一部分。

一、计算机系统工程

用于计算机硬件的工程技术是由电子设计技术发展起来的。今天，硬件设计技术已经达到了比较成熟的水平。虽然制造方法仍在不断地改进，但可靠性已是一种可以期待的现实，而不再是一种愿望了。

但是，计算机软件工程却还处于某种困境之中。一方面，在以计算机为基础的系统中，软

件已经取代硬件成为系统中设计起来最困难、最不容易成功(按时完成和不超过预计的成本),而且是最不易管理的部分。另一方面,随着以计算机为基础的系统在数量、复杂程度和应用范围上的不断增长,对软件的需求却仍然有增无减。

计算机系统工程的主要内容是对系统所要求的功能加以揭示、分析,并把它们分配给系统的各个部分,如图 1-1 所示。

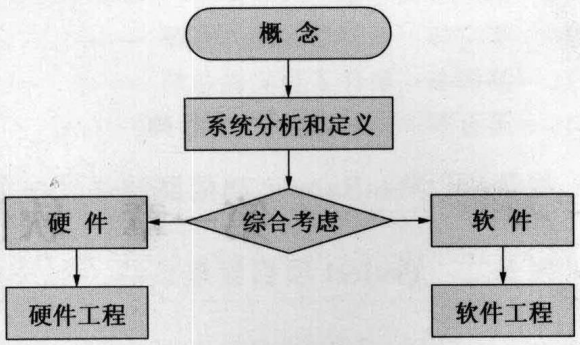


图 1-1 计算机系统工程

在大多数新系统创建时,对系统所要求的功能往往只有模糊的概念。系统分析和定义的目的在于明确项目的范围,这就要对需要进行处理的信息、所要求的功能、所期望的性能以及设计的约束和检验的标准等进行系统、详细的分析。

在范围确定之后,计算机系统工程师必须考虑多种能潜在的满足项目范围的、可供选择的配置。在综合考虑各项因素之后,选择其中的一种配置,并将系统的功能分配给系统的各个部分(例如硬件和软件)。

二、计算机软件的发展

软件发展的历史是与计算机系统发展紧密相关的。

在计算机系统发展的初期(20 世纪 50～60 年代),硬件经历了不断的变化,而软件则被多数人作为一种事后的工作来看待,几乎没有什么系统的方法可以遵循。在此期间,多数系统采用批处理工作方式,硬件通常仅用来执行一个单一的为某个特定的应用目的而编制的程序。

当通用硬件已经成为平常的事情时,软件则仍然是为每一种用途而分别设计的,它们的通用性有限。大多数软件是由使用该软件的人或机构研制的,软件设计是在某个人的头脑中完成的一种隐含的过程。

计算机系统发展的第二个时期跨越了从 20 世纪 60 年代中期到 70 年代中期这 10 年,这个时期以产品化软件的使用和“软件车间”的出现为特征。多道程序设计、多用户系统引入了人机对话的新概念。人机对话技术为计算机的应用开辟了新的天地,并使硬件和软件提高到新的水平;实时系统能够在几毫秒内收集、分析和变换来自多个信息源的数据,进而控制处理过程并产生输出;联机辅助存储设备的发展导致了数据库管理系统的出现;人们开发软件以广泛销售为目的。

随着以计算机为基础的系统日益增多,计算机软件库开始膨胀。所有这些程序,当检测出故障、用户要求改变时,或者要使软件在新的硬件上运行时,都必须加以维护和修改,软件维护开销越来越大。更坏的是,许多软件所带有的个人色彩使得它们实际上是不可维护的。“软件危机”开始了。

计算机系统发展的第三个时期从 20 世纪 70 年代初期开始,分布式系统(多个计算机、各机器并行执行和相互通信)极大地增加了以计算机为基础的系统的复杂性。由于微处理机和

有关部件的功能越来越强而价格越来越低，因此在最常用的计算机应用领域中，具有“嵌入智能”的产品取代了较大的计算机。此外，微处理机的出现也使得人们可能以极低的成本实现复杂的逻辑功能。

硬件的迅速发展超过了人们提供支持软件的能力。在第三个发展时期，软件危机日益严重。维护软件的费用迅速增长，而软件开发的生产率又跟不上对软件需求的步伐。为了对付不断增长的软件危机，软件工程开始得到认真的对待。如今，计算机系统的发展正在向第四个时期过渡，即从技术性应用向消费性市场过渡。计算机软件发展的3个时期详见表1-1。

表1-1 计算机软件发展的3个时期

时期 / 特点	程序设计	程序系统	软件工程
软件所指	程序	程序及说明书	程序、文档及数据
主要程序设计语言	汇编及机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括设计和测试	软件生存周期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大、中型软件开发机构
软件规模	小型	中、小型	大、中、小型
决定质量的因素	个人程序设计技术	小组技术水平	管理水平
开发技术和手段	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、网络和分布式开发、面向对象技术、软件过程与过程改进
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高、存储容量小、工作可靠性差	降价，速度、存储容量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	软件技术的发展不能满足需求，出现软件危机	开发技术有进步，但未获突破性进展，价格高，未完全摆脱软件危机

* 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。

第二节 软件、软件生存周期和软件生存周期过程

2006年出版的《中国大百科全书》给软件下的定义是：软件是“计算机系统中的程序和有关的文件。程序是计算任务的处理对象和处理规则的描述；文件是为了便于了解程序所需的资料说明。程序必须装入机器内部才能工作，文件一般是给人看的，不一定装入机器。程序作为一种具有逻辑结构的信息，精确而完整地描述计算任务中的处理对象和处理规则。这一描述还必须通过相应的实体才能体现”。

也就是说，软件不仅仅是指程序，在软件研制过程中按一定规格产生的各种文件也是软件不可缺少的组成部分。

《软件工程术语》(GB/T11457—1995)定义了软件生存周期，即：从设计软件产品开始到产品不能再使用时为止的时间周期。亦即：一个计算机软件，从出现一个构思之日起，经过开发成功投入使用，在使用中不断增补修订，直到最后决定停止使用，并被另一款软件替代时止，被认为是该软件的一个生存周期(或称生命周期、生存期)。

一个软件产品的生存周期可以划分成若干个互相区别而又有联系的阶段，每个阶段中的工作均以上一阶段工作的结果为依据，并为下一阶段的工作提供了前提。经验表明，失误造成的差错越是发生在生存周期的前期，在系统交付使用时造成的影响和损失就越大，要纠正它所花费的代价也越高。因此，在前一阶段工作没有做好之前，决不要草率地进入下一阶段。

《软件生存周期过程》(GB/T8566—2001)则根据软件工程的实践和软件工程学科的发展，进一步完善了软件生存周期的定义，即：软件生存周期是从概念形成直到退役，并且由获取和供应软件产品及服务的各个过程所组成。该标准把软件生存周期中可以开展的活动分为5个基本过程(获取过程、供应过程、开发过程、运作过程、维护过程)、8个支持过程(文档编制过程、配置管理过程、质量保证过程、验证过程、确认过程、联合评审过程、审核过程、问题解决过程)和4个组织过程(管理过程、基础设施过程、改进过程、培训过程)。

软件生存周期过程中阶段的划分有助于软件研制管理人员借用传统工程的管理方法(重视工程性文件的编制、采用专业化分工方法、在不同阶段使用不同的人员等)，从而有利于明显提高软件质量、降低成本、合理使用人才，进而提高软件开发的劳动生产率。

由于工作的对象和范围的不同以及经验的不同，对软件生存周期过程中各阶段的划分也不尽相同。但是，这些不同划分中有许多相同之处。《计算机软件开发规范》(GB/T8566—1988)(《软件生存周期过程》GB/T8566—2001的前身)将软件生存周期划分为以下8个阶段，即：可行性研究与计划、需求分析、概要设计(即结构设计)、详细设计、实现(包括单元测试)、组装测试(即集成测试)、确认测试、使用和维护。如图1-2所示为软件生存周期的瀑布模型。

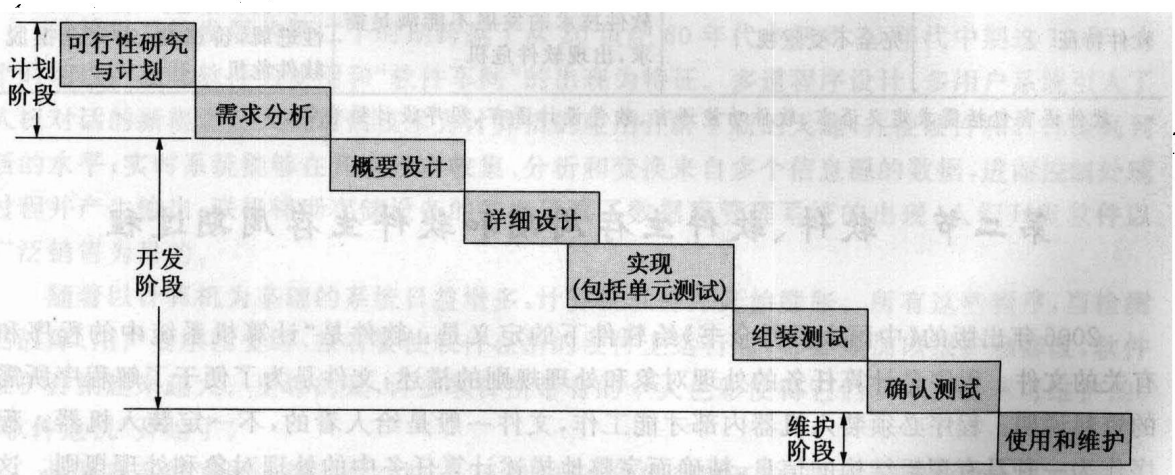


图1-2 软件生存周期的瀑布模型

软件生存周期是对软件的一种长远发展的看法，这种看法把软件开始开发之前和软件交付使用之后的一些活动都包括在软件生存周期之内。

应当注意的是，软件系统的实际开发工作不可能直线地通过分析、设计、编程和测试等阶段，出现各阶段间的回复是不可避免的。

软件生存周期的每个阶段都要产生一定规格的软件文件移交给下一阶段，使下一阶段在所提供的软件文件的基础上继续开展工作。《计算机软件产品开发文件编制指南》(GB/T8567—1988，已升级为GB/T8567—2006《计算机软件文档编制规范》)建议在软件的开发过程中编制下述14种文件，即：可行性研究报告、项目开发计划、软件需求说明书、数据要求说明书、概要设计说明书、详细设计说明书、数据库设计说明书、用户手册、操作手册、模块开发卷宗、测试计划、测试分析报告、开发进度月报以及项目开发总结报告。而《计算机软件需求说明编制指南》(GB/T9385—1988)、《计算机软件测试文件编制规范》(GB/T9386—1988)等有关软件工程的国家标准对软件文件的编制提出了更为详尽的要求，《软件文档管理指南》(GB/T16680—1996)则明确了对软件文件的管理要求。

图1-3和图1-4分别说明了在典型的情况下，软件生存周期各阶段的工作量所占的比重。图1-3说明运行维护工作量要占整个生存周期工作量的一半以上，图1-4则说明了测试阶段(组装测试和确认测试)的工作量约占整个开发期工作量的近一半。

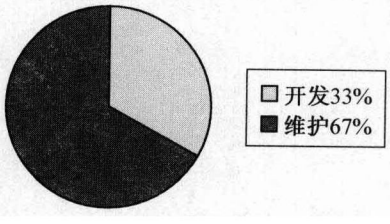


图1-3 软件生存周期工作量分配

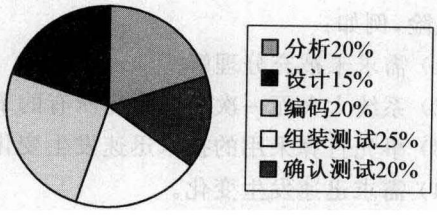


图1-4 开发阶段工作量分配

第三节 软件生存周期模型

软件生存周期模型(又称软件开发模型)是软件工程的一个重要的概念，它可以定义为一个框架，含有遍历系统从确定需求到终止使用这一生存周期的软件产品的开发、运行和维护中需实施的过程、活动和任务。

软件生存周期模型清晰、直观地表达了软件开发的全过程，明确规定了开发工作各阶段所要完成的主要活动和任务，以作为软件项目开发工作的基础。对于不同的软件系统，可以采用不同的开发方法、使用不同的程序设计语言、挑选各种不同技能的人员参与工作、运用不同的管理方法和手段，以及允许采用不同的软件工具和不同的软件工程环境等。软件生存周期模型是稳定有效和普遍适用的。

在软件生存周期过程中，软件生存周期模型仅对软件的开发、运作和维护过程有意义，在信息技术国际标准ISO12207和ISO9000-3中都提到软件生存周期模型，包括瀑布模型、渐增模型、演化模型、螺旋模型、喷泉模型和智能模型等。

一、瀑布模型

瀑布模型(Waterfall Model)是1970年W. Royce提出的最早的软件开发模型(如图1-2所示),它将软件开发过程中的各项活动规定为依固定顺序连接的若干阶段工作,形如瀑布流水,最终得到软件系统或软件产品。换句话说,它将软件开发过程划分成若干个互相区别而又彼此联系的阶段,每个阶段中的工作都以上一个阶段工作的结果为依据,同时为下一个阶段的工作提供了前提。

瀑布模型之所以能广泛流行,一方面,由于它在支持开发结构化软件、控制软件开发复杂度、促进软件开发工程化方面起到了显著作用;另一方面,它为软件开发和维护提供了一种当时较为有效的管理模式,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软件开发过程进行指导,从而使软件质量在一定程度上得到保证。1988年曾依据该开发模型制定并公布了《软件开发规范》(GB8567—1988),对我国软件开发起到了较大的促进作用。

瀑布模型在大量的实践中也暴露出了不足和问题。例如,由于固定顺序,前期阶段工作中所造成的差错,越到后期阶段,所造成的损失和影响也越大,为了纠正它而花费的代价也越高,尽管技术人员小心翼翼,但这种情况还是会经常发生。因此,在评价瀑布模型时,应考虑相关风险,例如:

(1) 需求未被充分理解。

(2) 系统太大而一次不能做完所有的事。

(3) 事先打算采用的技术迅速发生变化。

(4) 需求迅速发生变化。

(5) 有限的资源。

(6) 无法利用某一中间产品。

二、渐增模型

渐增模型(Incremental Model)是指从一组给定的需求开始,通过构造一系列可执行的中间版本来实施开发活动。第一个中间版本纳入一部分需求,下一个中间版本纳入更多的需求,依此类推,直到系统完成。每个中间版本都要执行必要的过程、活动和任务,如需求分析和体系结构设计需要执行一次,而详细设计、编码和测试、软件组装和验收测试在每个中间版本构造过程中都要执行,如图1-5所示。

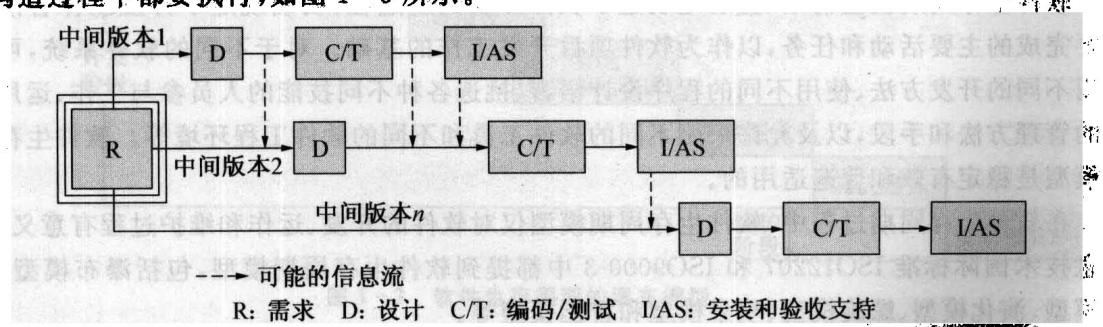


图1-5 渐增模型示意

在开发每个中间版本时，开发过程中的活动和任务顺序地或部分平行地使用。当相继的中间版本在部分并行开发时，开发过程中的活动和任务可以在各中间版本间平行地采用。

在评价该模型时，需要考虑的风险因素是：

(1) 需求未被很好地理解。

(2) 突然提出一些功能。

(3) 事先打算采用的技术迅速发生变化。

(4) 需求迅速发生变化。

(5) 长时期内仅有有限的资源保证(工作人员/资金)。

三、演化模型

演化模型(Evolutionary Model，又称进化模型)主要针对事先不能完整定义需求的软件项目开发。由于人们对软件需求的认识模糊，许多软件开发项目很难一次开发成功，返工再开发也就难以避免。因此，人们对需要开发的软件先给出基本需求，做第一次试验开发，其目标仅在于探索可行性和弄清需求，取得有效的反馈信息，以支持软件的最终设计和实现。通常，我们把第一次试验开发出来的软件称为原型(Prototype)，这种开发模型可以减少由于需求不明给开发工作带来的风险。

演化模型也有多种形式，例如：

(1) 丢弃型。原型开发后，已获得了更为清晰的需求反馈信息，原型无需保留而被丢弃，开发的原型仅以演示为目的，这往往用在软件用户界面的开发上。

(2) 样品型。原型规模与最终产品相似，只是原型仅供研究之用。

(3) 渐增式演化型。原型作为最终产品的一部分，它可满足用户的部分需求，经用户试用后提出精化系统、增强系统能力的需求，开发人员根据反馈信息，实施开发的迭代过程。如果一次迭代过程中有些需求还不能满足用户的要求，可在下一迭代中予以修正，整个过程实现后软件才可最终交付使用。

演化模型(如图 1-6 所示)也是通过构造系统的各个可执行的中间版本来开发系统的，但是它与渐增模型的区别是，承认需求不能被完全了解，且不能在初始时就确定。在该模型中，需求一部分被预先定义，然后在每个相继的中间版本中逐步完善。该模型中，每个中间版本在开发时，开发过程中的活动和任务顺序地或部分重叠平行地被采用。

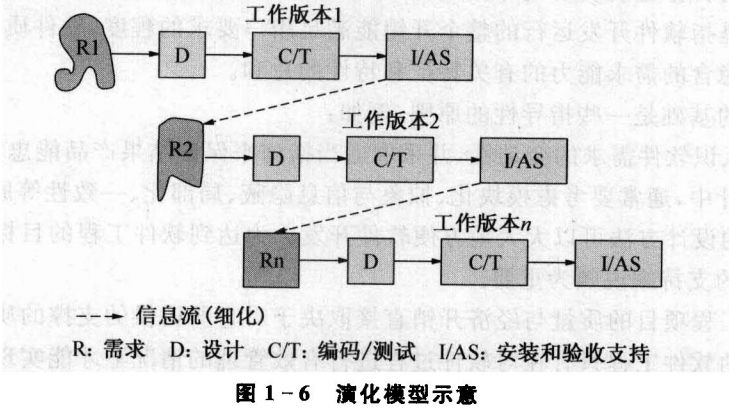


图 1-6 演化模型示意

对所有的中间版本，开发过程中的活动和任务通常按同一顺序被重复使用。维护过程和运作过程可以与开发过程平行地使用，获取过程、供应过程、支持过程和组织过程通常与开发过程平行地使用。

第四节　软件工程定义

人们对软件工程的定义和内涵有一个广泛讨论、研究和认识的过程，从而对软件工程有了各种各样的定义。例如：

(1) P. Wegner 和 B. Boehm 认为软件工程可定义为：科学知识在设计和构造计算机程序，以及开发、运作和维护这些程序所要求的有关文档编制中的实际应用。

(2) F. L. Bauer 认为软件工程的定义是：为了经济地获得软件，这个软件是可靠的并且能在实在的计算机上工作所需要的健全的工程原理(方法)的确立和使用。

(3) 1983 年 IEEE(国际电气与电子工程师协会)的软件工程术语汇编中，将软件工程定义为：对软件开发、运作、维护、退役的系统研究方法。1990 年新版的 IEEE 软件工程术语汇编又将定义更改为：对软件开发、运作、维护的系统化的、有纪律的、可定量的方法之应用，即是对软件的工程化应用。

(4)《软件工程术语》(GB/T11457—1995)则把软件工程定义为：软件开发、运行、维护和引退的系统方法。

从以上对软件工程的不同定义中，我们可以看到对其内容的理解是逐步深入的。发展到今天，软件工程已是一门交叉性学科，它是解决软件问题的工程，对它的理解不应是静止的和孤立的。软件工程是应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则、方法来创建软件，从而达到提高质量、降低成本的目的。其中，计算机科学和数学用于构造模型、分析算法；工程科学用于制定规范、明确风格、评估成本、确定权衡；管理科学用于进度、资源、质量、成本等的管理。

对于软件工程所包含的内容也不是一成不变的。随着人们对于软件系统的研制开发和生产的理解，也应该用发展的眼光来看待它，只有这样，才能客观、公正地反映出它的内涵和外延。

软件工程的目标是明确的，就是研制开发与生产出具有良好的软件质量和费用合算的产品。费用合算是指软件开发运行的整个开销能满足用户要求的程度，软件质量是指该软件能满足明确的和隐含的需求能力的有关特征和特性的总和。

软件工程的基础是一些指导性的原则，例如：

(1) 必须认识软件需求的变动性，并采取适当措施来保证结果产品能忠实地满足用户要求。在软件设计中，通常要考虑模块化、抽象与信息隐蔽、局部化、一致性等原则。

(2) 稳妥的设计方法可以大大地方便软件开发。为达到软件工程的目标，[illegible]具与环境对软件设计的支持来说颇为重要。

(3) 软件工程项目的质量与经济开销直接取决于对它所提供的支撑的[illegible]用。

(4) 有效的软件工程只有在对软件过程进行有效管理的情况下才能[illegible]