普通高等教育“十三五”规划教材（软件工程专业）

软件工程

（第二版）

Software Engineering

王　伟　杜文洁　英　皓
赵志刚　张明林　王树彬　等　编著

- 内容丰富、先进——反映软件开发技术的最新成果
- 结构系统、合理——章节设计循序渐进、深入浅出、突出实用性
- 讲解详细、全面——阐述了软件工程的基础理论知识及相关的实用技术

配套微课资源

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十三五”规划教材（软件工程专业）

软件工程（第二版）

王　伟　杜文洁　英　皓
赵志刚　张明林　王树彬　等　编著

中国水利水电出版社
www.waterpub.com.cn
·北京·

内 容 提 要

本书详细阐述了软件工程的基础知识及相关实用技术，内容包括软件工程概述、软件可行性研究、软件需求分析、软件总体设计、软件详细设计、面向对象技术、统一建模语言（UML）、程序编码、软件测试、软件实施与维护、软件重用技术、软件项目计划与管理、软件开发工具与软件工程环境、综合实例——超市在线销售系统。

本书注重软件工程基础知识和基本概念的形象表述，注重理论与实践的结合，力求做到通俗易懂、突出实用性。本书通过对软件工程常用方法的介绍，展现软件设计的实际运作过程，帮助读者掌握相关知识并在实践中加以运用。

本书可作为高等院校、高职高专院校计算机和软件相关专业的教材，也可作为社会上软件开发技术的培训教材，同时还可以供从事软件开发工作的技术人员参考。

图书在版编目（CIP）数据

软件工程 / 王伟等编著. -- 2版. -- 北京 : 中国水利水电出版社, 2018.11
普通高等教育“十三五”规划教材. 软件工程专业
ISBN 978-7-5170-7043-6

Ⅰ. ①软… Ⅱ. ①王… Ⅲ. ①软件工程－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2018)第251863号

策划编辑：石永峰　责任编辑：张玉玲　加工编辑：张青月　封面设计：李　佳

书　名	普通高等教育“十三五”规划教材（软件工程专业） 软件工程（第二版） RUANJIAN GONGCHENG
作　者	王　伟　杜文洁　英　皓　赵志刚　张明林　王树彬　等　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路1号D座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16开本　18印张　440千字
版　次	2010年6月第1版　2010年6月第1次印刷 2018年11月第2版　2018年11月第1次印刷
印　数	0001—3000册
定　价	45.00元

前　言

软件工程是一门运用工程学的原理和方法来组织和管理软件的开发、运行和维护，力求以高效率生产出高质量的软件产品的学科。软件工程是计算机科学技术领域的一个重要分支，在软件开发实践中起到基础指导的作用。

当今，国内软件工程教材较多，相关内容的深浅度、侧重点各有不同。我们依据高等院校本科生“软件工程”学科教学大纲所规定的教学要求编写了本书，同时把多年来软件工程的教学经验和教学实践成果融入到书中。本书介绍了软件工程的基础理论、软件开发流程以及相关实践操作，从而指导学生进行软件开发活动。本书适合高职高专计算机和软件技术及相关专业的学生使用。

本书在编写上，本着深入浅出的原则，注重内容的先进性、系统性和实用性，力求反映软件开发技术的最新成果；在结构安排上，通俗易懂地阐述软件工程的基础理论知识，循序渐进，注重结合实践内容，在每章内容后面基本都附有小结和课后习题。

本书共 14 章，针对计算机专业和软件技术专业学生的实际特点设计了知识结构，全面系统地阐述了软件工程的知识体系、软件开发的相关流程，具体如下：

第 1 章，软件工程概述。本章主要介绍了软件工程的概念、软件生存周期和软件开发模型。

第 2 章，软件可行性研究。本章概括描述了可行性研究的内容、可行性研究的步骤和可行性研究报告。

第 3 章，软件需求分析。本章系统介绍了需求分析、面向数据流的分析方法、需求分析方法与图形工具、实体—联系图和需求规格说明与评审。

第 4 章，软件总体设计。本章介绍了软件设计、总体设计的图形描述工具、模块化设计和面向数据流的设计方法。

第 5 章，软件详细设计。本章介绍了详细设计、详细设计的图形描述工具、Jackson 设计方法和 Warnier 设计方法。

第 6 章，面向对象技术。本章介绍了面向对象技术、面向对象的开发模型、面向对象的分析、面向对象的系统设计、面向对象的实现。

第 7 章，统一建模语言（UML）。本章介绍了 UML 概况、UML 的概念模型、UML 的静态建模机制、UML 的动态建模机制和 UML 的物理架构建模。

第 8 章，程序编码。本章介绍了程序设计语言、结构化程序设计、程序设计风格、程序设计效率和程序复杂性度量。

第 9 章，软件测试。本章介绍了软件测试的基本概念、软件测试方法、软件测试流程、测试用例的设计、面向对象软件测试和软件测试相关文档。

第 10 章，软件实施与维护。本章介绍了软件产品的实施、软件产品的维护活动、软件维护过程、软件维护文档、软件可维护性和软件维护的深化——软件再工程。

第 11 章，软件重用技术。本章介绍了软件重用技术、基于构件的软件开发和面向对象的软件重用技术。

第 12 章，软件项目计划与管理。本章介绍了软件项目的计划与组织、软件成本估算及控制、软件工程标准与软件文档。

第 13 章，软件开发工具与软件工程环境。本章介绍了软件开发工具、软件工程环境和 CASE 技术。

第 14 章，综合实例——超市在线销售系统。

本书由王伟、杜文洁、英皓、赵志刚、张明林、王树彬等编著，周功、尹叔杰、王若鹏、刘玲也参与了部分内容的编写。全书统稿工作由王伟完成。

由于时间仓促及作者水平有限，书中难免存在错误和不妥之处，恳请广大读者批评指正。

编者

2018 年 7 月

目　　录

第 1 章　软件工程概述

【本章引言】

自从 1946 年第一台电子计算机诞生以来，计算机的研究、生产和应用都得到了迅猛的发展。计算机系统已经经历了四个不同的发展阶段，计算机科学也成为当今世界上发展最快和应用最广的学科之一。然而，我们仍然没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的关键因素。为了克服这种困扰，软件工作者在不断地研究消除软件危机的方法，从而逐步形成了计算机科学技术领域中的一门新兴的工程学科——软件工程。

本章将对软件概念、软件危机、软件工程概念、软件生存周期及软件开发模型等做简要的介绍。通过本章学习，可为后续章节的深入学习打下基础。

【本章重点】

- 软件工程的概念
- 软件生存周期
- 软件开发模型

【学习目标】

- 理解软件工程的基本概念和软件生存周期
- 了解软件开发的几种模型

1.1　软件危机

计算机系统已经从原来的电子管阶段发展到现在的大规模集成电路阶段，但人们仍没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的瓶颈。

1.1.1　软件的概念及特点

1. 软件的概念

计算机软件（Computer Software）是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲，软件被划分为编程语言、系统软件、应用软件和介于这两者之间的中间件。软件并不只是包括可以在计算机（这里的计算机是指广义的计算机）上运行的计算机程序，与这些计算机程序相关的文档一般也被认为是软件的一部分。简单地说软件就是程序加文档的集合体。

目前对软件比较公认的解释为：软件是计算机系统中与硬件相互依存的另一部分，包括程序、相关数据及其说明文档。其中程序是按照事先设计的功能和性能要求执行的指令序列；数据是程序运行过程中处理的对象；文档是与程序开发、维护和使用有关的各种图文资料。

2. 软件的特点

（1）软件不同于硬件。它是计算机系统中的逻辑实体而不是物理实体，具有抽象性。人们无法看到软件本身的形态，只有通过观察、分析、思考、判断等方式才能了解它的功能和性能。

（2）软件的生产不同于硬件。它没有明显的制作过程，一旦开发成功，就可以大量复制同一内容的副本。

（3）软件在运行过程中不会因为使用时间过长而出现磨损、老化以及用坏的问题，但可能要不断地对软件的设计和编码进行改动以适应硬件和系统环境以及使用者需求的变化。

（4）软件的开发、运行在很大程度上依赖于计算机系统。受计算机系统的限制，在客观上出现了软件移植问题。

（5）软件开发复杂性高，开发周期长，成本较大。至今软件开发尚未摆脱手工方式，虽然市场上出现了一些辅助开发工具，但是最终的核心代码仍然要程序员手工编写和组织。

1.1.2 软件危机的产生

1. 软件危机的含义

软件危机（Software Crisis）是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。概括地说，主要包含两方面的问题：如何开发软件，怎样满足对软件日益增长的需求；如何维护数量不断增长的已有软件。

鉴于软件危机的长期性和症状不明显的特征，近年来有人建议把软件危机更名为“软件萧条”或“软件困扰”。不过“软件危机”这个词强调了问题的严重性，而且也已被绝大多数软件工作者所熟悉，所以本书仍沿用它。

2. 软件危机产生的原因

软件危机爆发于 20 世纪 60 年代中期。软件规模的扩大、复杂性的增加和功能的增强，使得高质量的软件开发变得越来越困难。在软件开发的过程中，经常会出现不能按时完成任务、产品质量得不到保证、工作效率低下和开发经费严重超支等情况。

软件危机的出现及其日益严重的趋势，促使人们去探究其产生的根本原因。最终，人们发现软件危机的产生有两方面的因素：一方面与软件本身的抽象性和复杂性有关，这是客观原因；另一方面则与软件开发和维护过程中使用的技术和方法有关，这是主观原因。其根本原因是软件开发过程不成熟，主要表现为：

（1）忽视软件开发前期的调研和分析工作。只有用户最能了解他们自己的需求，但许多用户开始并不能正确具体地描述他们的需求，这就要求软件开发人员要做大量的深入细致的调查工作，反复多次地与用户进行交流，才能全面、真实、具体地了解用户的需求。实践证明，在没有准确完整地了解用户需求和定义问题的前提下就急于编程，是导致软件开发工程失败的主要原因之一。

（2）开发过程缺乏统一的规范。规模日益增大的软件往往需要很多人合作开发，这就需要在用户和软件开发人员之间，以及软件开发人员之间进行有效及时的沟通，以消除开发过程中对问题理解的差异，从而防止后续错误的发生。然而多年的“手工作坊”式的工作环境，使得软件开发人员更习惯于独自工作。开发过程中所采用的技术没有一个统一的规范和标准，每个人按照自己的习惯来决定要做什么。这种情况势必会妨碍整个项目开发的团队合作，这也是

导致软件危机产生的一个重要原因。

（3）软件开发管理困难。大型软件项目需要规模较大的团队来共同完成，多数管理人员缺乏大型软件系统的管理经验，而多数软件开发人员又缺乏管理方面的经验。两方面不能进行及时准确的信息交流，甚至还会产生误解。软件开发人员不能有效、独立地处理软件开发过程中的各种工作流程和工作关系，因此容易产生疏漏和错误。另外，开发过程中忽视撰写和保存相关文档的工作，使文档缺乏一致性和完整性，从而导致开发者失去工作的基础、管理者失去管理的依据。

（4）没有完善的质量保证体系。建立完善的质量保证体系需要有严格的评审制度，同时还需要有科学的软件测试技术及质量维护技术。软件的质量得不到保证，开发出来的软件产品往往不能满足需求，同时还可能需要花费大量的时间、资金和精力去修复软件的缺陷，从而导致软件质量下降和开发预算超支等后果。

3. 软件危机的案例

1995 年，Standish Group 研究机构以美国境内 8000 个软件项目作为调查样本进行调查。调查结果显示，有 84%的软件计划无法于既定时间、经费中完成，超过 30%的项目于运行中被取消，项目预算平均超出 189%。

IBM OS/360 操作系统被认为是一个典型的案例。到现在为止，它仍然被使用在 360 系列主机中。这个经历了数十年、极度复杂的软件项目甚至产生了一套不包括在原始设计方案之中的工作系统。OS/360 是第一个超大型的软件项目，使用了 1000 左右的程序员。Frederick P.Brooks 在他的大作《人月神话》中承认，他在管理这个项目的时候犯了一个价值数百万美元的错误。

美国银行 1982 年进入信托商业领域，并规划发展信托软件系统。项目原订预算 2 千万美元，开发时程 9 个月，预计于 1984 年 12 月 31 日以前完成。后来至 1987 年 3 月都未能完成该系统，期间已投入 6 千万美元。美国银行最终因为此系统不稳定而不得不放弃，并将 340 亿美元的信托账户转移出去，而且失去了 6 亿美元的信托生意商机。

1.1.3　解决软件危机的方法

为了摆脱软件危机造成的困境，北大西洋公约组织（NATO）的科学委员会于 1968 年在联邦德国召开的有关研讨会上，第一次提出了“软件工程”（Software Engineering）的概念。它作为一个新兴的工程学科，主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本、改进软件产品质量、提高软件生产率水平的目标。软件工程学从硬件工程和其他人类工程中吸收了许多成功的经验，明确提出了软件生存周期的模型，发展了许多软件开发与维护阶段适用的技术和方法，并应用于软件工程实践，取得了良好的效果。

在软件开发过程中人们开始研制和使用软件工具，用以辅助软件项目管理与技术生产。人们还将软件生存周期各阶段使用的软件工具有机地集合成为一个整体，形成能够连续支持软件开发与维护全过程的集成化软件支援环境，以期从管理和技术两方面解决软件危机问题。

此外，人工智能与软件工程的结合成为 20 世纪 80 年代末期活跃的研究领域。基于程序变换、自动生成和可重用软件等软件新技术研究也取得了一定的进展，把程序设计自动化的进

程向前推进一步。在软件工程理论的指导下，发达国家已经建立起较为完备的软件工业化生产体系，形成了强大的软件生产能力。软件标准化与可重用性得到了工业界的高度重视，在避免重复劳动、缓解软件危机方面起到了重要作用。

1.2 软件工程的概念

软件工程是一门指导计算机软件开发和维护的工程学科。它运用工程学中的概念、原理、方法和技术来指导软件开发和维护工作。

1.2.1 软件工程的定义及原理

1. 软件工程的定义

关于软件工程，不同的学者和组织机构给出了不同的定义。

1993 年，电气电子工程师学会（Institute of Electrical and Electronics Engineers，IEEE）给出的定义是：软件工程是将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护过程，也就是将工程化应用于软件开发和管理之中。

2001 年，软件工程大师 Roger S.Pressman 对软件工程的定义是：软件工程是一个过程、一组方法和一系列工具。

2006 年我国的国家标准《软件工程术语》（GB/T 11457-2006）中对软件工程的定义为："应用计算机科学理论和技术以及工程管理原则和方法，按预算和进度，实现满足用户要求的软件产品的定义、开发、发布和维护的工程或进行研究的科学。"

目前人们比较认可的一种定义认为：软件工程是研究和应用如何以系统性的、规范化的、可定量的过程化方法去开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来。

2. 软件工程的基本原理

自从 1968 年在联邦德国召开的国际会议上正式提出并使用"软件工程"这个术语以来，研究软件工程的专家学者们陆续提出了 100 多条关于软件工程的准则或"信条"。著名的软件工程专家 Barry W.Boehm 综合这些专家学者们的意见并总结了 TRW 公司多年开发软件的经验，于 1983 年在一篇论文中提出了软件工程的 7 条基本原理。他认为这 7 条原理是确保软件产品质量和开发效率的最小集合。这 7 条基本原理是互相独立的，其中任意 6 条基本原理的组合都不能代替另一条基本原理。因此，它们是缺一不可的最小集合。然而这 7 条基本原理又是相当完备的，人们虽然不能用数学方法严格证明它们是一个完备的集合，但是可以证明在此之前已经提出的 100 多条软件工程原理都可以由这 7 条原理的任意组合所蕴含或派生。

下面简要介绍软件工程的 7 条基本原理。

（1）用分阶段的生存周期计划严格管理。经统计发现，在不成功的软件项目中有一半左右是由计划不周造成的。可见把建立完善的计划作为第一条基本原理是吸取了前人的教训而提出来的。

在软件开发与维护的漫长生存周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生存周期划分成若干个阶段，并相应地制订出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Barry W.Boehm 认为，在软件的整个生存周期中应

该制订并严格执行六类计划，它们是项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划和运行维护计划。

不同层次的管理人员都必须严格按照计划各尽其职地管理软件开发与维护工作，绝不能受客户或上级人员的影响而擅自背离预定计划。

（2）坚持进行阶段评审。人们已经认识到，软件的质量保证工作不能等到编码阶段结束之后再进行。这样说至少有两个理由：第一，大部分错误是在编码之前造成的。例如，根据 Barry W.Boehm 等人的统计，设计错误占软件错误的 63%，编码错误仅占 37%；第二，错误发现与改正得越晚，所需付出的代价也越高。因此，在每个阶段都进行严格评审，以便尽早发现在软件开发过程中所犯的错误，是一条必须遵循的重要原则。

（3）实行严格的产品控制。在软件开发过程中不应随意改变需求，因为改变一项需求往往需要付出较高的代价。但是，在软件开发过程中改变需求又是难免的。由于外部环境的变化，相应地改变用户需求是一种客观需要，显然不能硬性禁止用户提出改变需求的要求，而只能依靠科学的产品控制技术来顺应这种要求。也就是说，当改变用户需求时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。

所谓基准配置又称为基线配置，是经过阶段评审后的软件配置成分（各个阶段产生的文档或程序代码）。基准配置管理也称为变动控制：一切有关修改软件的建议，特别是涉及对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准后才能实施修改。绝不能谁想修改软件（包括尚在开发过程中的软件）就随意进行修改。

（4）采用现代程序设计技术。从提出软件工程的概念开始，人们一直把主要精力用于研究各种新的程序设计技术。20 世纪 60 年代末提出的结构化程序设计技术已经成为绝大多数人公认的程序设计技术。以后又进一步发展出各种结构分析（SA）与结构设计（SD）技术。近年来，面向对象技术已经在许多领域中迅速地取代了传统的结构化开发方法。实践证明，采用先进的技术不仅可以提高软件开发和维护的效率，而且可以提高软件产品的质量。

（5）结果应能清楚地被审查。软件产品不同于一般的物理产品。它是看不见摸不着的逻辑产品。软件开发人员（或开发小组）的工作进展情况可见性差，难以准确度量，从而使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的可见性，更好地进行管理，应该根据软件开发项目的总目标及完成期限，规定开发组织的责任和产品标准，从而使得所得到的结果能够清楚地被审查。

（6）开发小组的人员应该少而精。这条基本原理的含义是，软件开发小组的组成人员要有较高的素质，而且人数不宜过多。开发小组人员的素质和数量是影响软件产品质量和开发效率的重要因素。素质高的人员的开发效率比素质低的人员的开发效率可能高几倍甚至几十倍，而且素质高的人员开发的软件中的错误明显少于素质低的人员开发的软件中的错误。此外，随着开发小组人员数量的增加，因为交流情况、讨论问题而造成的通信开销也急剧增加。当开发小组人数为 N 时，可能的通信路径有 N(N-1)/2 条。可见随着人数 N 的增大，通信开销将急剧增加。因此，组成少而精的开发小组是软件工程的一条基本原理。

（7）承认不断改进软件工程实践的必要性。虽然遵循上述六条基本原理就能够按照当代软件工程基本原理实现软件的工程化生产。但是，仅依靠上述六条基本原理并不能保证软件开发与维护的过程能赶上时代前进的步伐和技术的不断进步。所以，Barry W.Boehm 提出应把承认不断改进软件工程实践的必要性作为软件工程的第七条基本原理。按照这条基本原理，不仅

要积极主动地采纳新的软件技术，而且要注意不断总结经验，如收集进度和资源耗费数据、出错类型和问题报告数据等。这些数据不但可以用来评价新的软件技术的效果，而且可以用来指明必须着重开发的软件工具和应该优先研究的技术。

1.2.2 软件工程的原则及目标

1. 软件工程的原则

过去，软件工程的基本原则是抽象、模块化、清晰的结构、精准的设计规格说明。但现在的认识已经发生了很大的变化，现已提出的软件工程的四条基本原则为：

（1）必须认识软件需求的变动性，以便采取适当措施来保证软件产品能更好地满足用户需求。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）稳妥的实际方案将大大方便软件开发，以达到软件工程的目标。软件工具与环境对软件设计的支持来讲颇为重要。

（3）软件工程项目的质量与经济开销取决于对它所提出的支撑质量与效用。

（4）只有在强调对软件过程进行有效管理的情况下，才能实现有效的软件工程。

2. 软件工程的目标

软件工程的目标是在给定成本、进度的前提下，开发出具有适用性、有效性、可修改性、可靠性、可理解性、可维护性、可重用性、可移植性、可追踪性、可互操作性和满足用户需求的软件产品。追求这些目标有助于提高软件产品的质量和开发效率，减少软件维护的困难。

（1）适用性：软件在不同的系统约束条件下，使用户需求容易得到满足。

（2）有效性：软件系统能最有效地利用计算机的时间和空间资源。各种软件无不把系统的时/空开销作为衡量软件质量的一项重要技术指标。很多情况下，在追求时间有效性和空间有效性时会发生矛盾，这时不得不牺牲时间有效性换取空间有效性或牺牲空间有效性换取时间有效性。时/空折衷是经常采用的技巧。

（3）可修改性：允许对系统进行修改而不增加原系统的复杂性。支持软件的调试和维护有时是一个难以达到的目标。

（4）可靠性：能防止因概念、设计和结构等方面的不完善造成的软件系统失效，具有挽回因操作不当造成软件系统失效的能力。

（5）可理解性：系统具有清晰的结构，能直接反映问题的需求。可理解性有助于控制系统软件复杂性，并支持软件的维护、移植或重用。

（6）可维护性：软件交付使用后，能够对它进行修改，以改正潜伏的错误，改进性能和其他属性，使软件产品适应环境的变化等。软件维护费用在软件开发费用中占有很大的比重。可维护性是软件工程中一项十分重要的目标。

（7）可重用性：把概念或功能相对独立的一个或一组相关模块定义为一个软部件。可组装在系统的任何位置，降低工作量。

（8）可移植性：软件从一个计算机系统或环境移到另一个计算机系统或环境的难易程度。

（9）可追踪性：根据软件需求对软件设计、程序的正向追踪能力；或根据软件设计、程序对软件需求的逆向追踪的能力。

（10）可互操作性：多个软件元素相互通信并协同完成任务的能力。

1.3　软件生存周期

任何事物都有一个发生、发展、成熟，直到衰亡的过程，系统和软件产品也一样，有着一个定义、开放、运行维护，直至被淘汰的过程，一般称其为计算机软件的生存周期。

1.3.1　软件生存周期的含义

软件生存周期是指从构思软件产品开始到产品不能再使用时为止的时间。典型的软件生存周期包括需求分析阶段、设计阶段、实现阶段、测试阶段、安装和验收阶段、运行和维护阶段，有时还包括引退阶段。在引退阶段内对软件产品的支持将被终止。

相关的一个概念是软件开发周期。它是指从决定开发一个软件产品开始到产品交付为止的时间间隔。

软件工程的生存周期方法学从时间进程的角度把软件的开发和维护过程划分为若干个阶段。每一个阶段的工作任务相对独立，工作性质相近，每一个阶段工作的结束均经技术审查和管理复审，前一阶段工作的结束和确认是后一阶段工作开始的前提。作为阶段间的通信手段和阶段工作的审查依据，每个阶段均应提交高质量的文档资料，从而在软件开发结束时，可以提交一个完整而准确的软件配置，作为运行维护阶段管理的依据。

粗略来看，软件生存周期由计划、开发和运行三个阶段组成，每一阶段可再细分一些工作阶段。在不同的软件开发规范中，尽管对阶段的划分表面看来略有差别，但实质上是完全一致的。把整个开发问题分解为一个个子问题，便于各类人员分工协作、各个击破。在完成每个阶段的任务时，仍强调使用结构化的系统技术，从而降低整个开发工程的难度。随着开发周期的时间进程，也就是在有限的步骤内，把需要解决的问题从抽象逻辑概念逐步转化为具体的物理实现，直到最终的源程序为止。

1.3.2　软件的计划阶段

软件的计划阶段是软件生存周期的开始阶段。这个阶段的主要任务是确定软件开发任务的总目标；确定是否存在可行的系统解决方案；在确定开发任务可以继续进行的前提下，制订开发工程的人力、资源和进度计划。

可以把计划阶段再细分为几个阶段：

（1）问题定义要确定的是：要求解决的问题是什么？经过初步的调查和访问，即可明确问题的性质、工程的目标和规模。

（2）可行性研究要确定的是：问题有解吗？经过粗略的分析和设计，应该得到若干个系统解决方案，对每一个解决方案都可以从技术上、经济上、社会因素上分析可行性，即能不能做和应不应该做。

（3）制订开发计划：在工程决定正式开发时制订资源和进度计划。

1.3.3　软件的开发阶段

如果在计划阶段中确定软件的开发可行，那么将进入到软件的开发阶段。在开发阶段，通过分析和设计，最终实现软件系统。这一阶段可细分为五个阶段：

（1）需求分析：首先是确定系统的功能、性能和其他方面的要求。也就是确定系统必须做什么。

（2）概要设计：确定系统的事务处理流程，确定软件的总体结构和全局数据结构，以及对输入/输出、安全性控制等方面作出全局性的规划，从而回答系统应该怎么做才能满足需求规定。

（3）详细设计：给出软件的每一个构成元素，如程序模块、存储数据结构、输入/输出数据等的实际方案的详细策划。相当于其他工业产品的制造蓝图设计。

（4）编码：形成软件在预定环境中的源程序形式，即一般意义的“写程序”。

（5）测试和排错：设法找到程序的潜在错误并加以修正，以提高软件的正确性和可靠性。

1.3.4 软件的运行阶段

软件开发完成，投入运行。运行阶段的主要工作是软件维护。维护的目的在于使系统持久地满足用户需求，直到不得不开发新的软件为止。

从上面的简要叙述可以看到软件生存周期的几个主要特点：

（1）阶段的顺序性和依赖性。采用工程化的方法开发软件时，从对任务的抽象逻辑分析开始，要一个阶段一个阶段地顺序工作。前一阶段的完成是后一阶段工作的前提和依据，而后一阶段的完成往往又使前一阶段的成果在实现过程中具体了一个层次。

（2）逐步求精的结构化方法。从时间进程来看，整个软件的开发就是一个从抽象到具体的分层次实现过程。而每一个阶段的工作，亦体现出自顶向下、逐步求精的结构化技术特点。

（3）推迟实现的观点。对于有一定规模的软件，编码越早，完成的时间反而会更长，甚至会导致不可挽回的失败，这是为无数实例所证实了的。工程化方法在软件分析阶段和设计阶段的工作会导致独立于环境的抽象而完备的逻辑结果，这样必然大大缩短编码的实现时间，大大提高软件的正确性、可靠性和可维护性。

（4）质量保证措施。工程方法规定每一阶段都要生成高质量的文档。文档是通信的手段，是开发工作的依据，也是维护阶段的重要支持信息。每一阶段对文档的复审就是对本阶段工作成果的评定，使错误较难传递到下一阶段。越早纠正错误，付出的开销就越少，这是工程化方法的基本观念之一。

1.4 软件开发模型

为了指导软件的开发，用不同的方式将软件生存周期中的所有开发活动组织起来，即形成不同的软件开发模型。软件开发模型是从软件项目需求定义直至软件经使用后废弃为止，跨越整个软件生存周期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。到现在为止，已经提出了多种软件开发模型，如瀑布模型、演化模型、增量模型、螺旋模型、喷泉模型等。

1.4.1 瀑布模型

1970 年 Winston Royce 提出了著名的“瀑布模型”。直到 20 世纪 80 年代早期，它一直是唯一被广泛采用的软件开发模型。瀑布模型是将软件生存周期的各项活动规定为按固定顺序而