现代软件工程

主摇编摇成奋华摇摇摇摇
副主编摇曾凡秩摇何海江
编摇委摇陆海霞摇陈云志
摇摇摇摇刘铁祥摇李玉菲

中南大学出版社

前摇 言

近年来，以计算机为中心的信息产业飞速发展，极大地推动社会的进步。与此同时这种进步也促进了计算机本身的发展。人们看到各种新的硬件、各种开发工具、分析工具不断推出，令人眼花缭乱，但这只是发展的表面现象，发展最重要的不是推出几种产品，更重要的是有关思想的发展、认识的提高，比如：从面向过程的思想到面向对象的思想发展，循此思想人们推出了一系列的软件产品（[illegible]）。软件工程即是研究软件开发和软件管理的一门工程学科，是计算机应用及软件工程相关专业的主干课，它能培养那些从事软件开发、应用、系统分析等的人学会思考、学会分析。

近年来，有些人将高职高专教育完全等同于培训，对此作者持有不同见解。我们认为要注意教育与培训的区别，教育是培养人可持续发展能力，而培训仅局限在培养人的某种技能，高职高专是教育不是培训，如果把《软件工程》、《操作系统》等课程都取消，那么我们将教给学生什么，这是值得思考的问题。当然，高职高专是为社会培养生产管理一线的高级技工，那么从这种角度看，高职高专培养学生掌握几门工具，掌握生产操作技能也同样重要。不过我们必须明白，学会思想比掌握一种工具更重要。

参与本教材编写的是一批长期工作在高职高专一线的有丰富教学经验、熟悉高职高专教学模式的教师，他们是：长沙民政职业技术学院成奋华、湖南工程职业技术学院曾凡秩、湖南经济职业技术学院何海江、湖南保险职业技术学院李玉菲、常德职业技术学院陈云志、长沙民政职业技术学院陆海霞、湖南环保职业技术学院刘铁祥。在整部教材的编写过程中，不少的领导和同仁给予了关心和指导。我们特别要感谢长沙民政职业技术学院邹文开副院长，他一直关注整个教材的编写过程，并对教材的编撰作了不少重要指示。中南大学博士任胜兵老师对教材提出了很多宝贵意见，作者的一些同事也给予了宝贵的支持，我们在此一并致谢。

虽然我们想尽量使教材完善一些，但终因能力、学识的局限，加之时间仓促，所以书中的错误和缺陷在所难免，欢迎广大高校师生和[illegible]软件工作者批评指正。站在高职高专特色教材或更高的角度来看本书，肯定会发现许多不如意的地方，作为编者我们会虚心收集各方面反馈意见，在适当的时候进一步完善本教材。

成奋华

[illegible]年 [illegible]月

内容提要

本书吸取了国内外大量同类书刊的精华，并总结了编者多年从事软件工程教学和研究的经验和体会，本书的特点：讲解深入浅出，讲透基本的概念、原理、技术和方法；注重科学性和现代性；既有原理性论述，又有丰富、完整的实例配合，有利于读者学习。

本书由九章组成，第一章绪论，第二章软件的需求分析，第三章系统设计与实现，第四章面向对象设计方法，第五章软件编码，第六章软件测试，第七章软件维护，第八章软件管理，第九章软件开发环境与工具。

本书可作为大专院校“软件工程”课程的教材或教学参考书，可供有一定实际经验的软件工作人员和需要开发应用软件的广大计算机用户阅读参考，也可作为大专学生自学用书。

目 录

第 1 章摇绪摇论

教学目标

灵郾理解摇软件工程的定义、性质、特点、目标；软件生存期各阶段的特点和内容；软件危机产生的原因。

圆郾应用摇软件生存周期模型。

猿郾了解摇软件生产发展的三个阶段及各阶段的特点；软件危机的产生及其表现形式。

源郾关注摇软件工程的三种开发方法：面向过程的方法、面向数据的方法、面向对象的方法。

灵援灵摇软件工程简述

灵援灵援灵摇软件的发展

灵郾软件发展的三个阶段

灵怨源远年世界上的第一台计算机 耘晕陨粤悦诞生以后，就有了程序的概念。可以认为它是软件的前身。经历了几十年的发展，使人们得以对软件有了更为深刻的认识。在这几十年中，计算机软件经历了四个发展阶段：

- 程序设计阶段，约为 圆园世纪 缘园至 远园年代；
- 程序系统阶段，约为 远园至 苑园年代；
- 软件工程阶段，约为 苑园年代以后；
- 现代软件阶段，约为 怨园年代后。

圆郾人们对软件的认识

从 圆园世纪 缘园年代到 远园年代，人们曾经把程序设计看做是一种任人发挥创造才能的技术领域。当时一般认为，写出的程序只要能在计算机上得出正确的结果，程序的写法可以不受任何约束；而且只有那些通篇充满了程序技巧的程序才是高水平的好程序，尽管这些程序很难为别人看懂。然而随着计算机的广泛使用，人们逐渐抛弃了这种观点。因为对于小的程

序，仅供极小范围使用（例如只是程序设计者本人或只有几个人），尚可“孤芳自赏”。对于稍大的程序，并需要较长时间为许多人使用的程序，情况就完全不同了。人们要求这些程序容易看懂、容易使用，并且容易修改和扩充。于是，程序便从个人按自己意图创造的“艺术品”转变为能为广大用户接受的工程化产品。这时程序中难以理解的技巧成了有害的东西。

软件的定义

一般认为，程序是计算机为完成特定任务而执行的指令的有序集合。站在应用的角度可以更通俗理解为：

程序 = 算法 + 数据结构

对象 = 对象 + 消息

构件 = 构件 + 构架

通常，软件可以定义为：

软件 = 程序 + 数据 + 文档

软件危机

在软件技术发展的第二阶段，随着计算机硬件技术的进步，计算机的容量、速度和可靠性有明显提高，生产硬件的成本降低了。计算机价格的下跌为它的广泛应用创造了极好的条件。在这一形势下，要求软件能与之相适应。一些开发复杂的、大型的软件项目提了出来。然而软件技术的进步一直未能满足形势发展提出的要求。在软件开发中遇到的问题找不到解决的方法，致使问题积累起来，形成了日益尖锐的矛盾。

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。

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（[illegible]）”或“软件困扰（[illegible]）”。不过“软件危机”这个词强调了问题的严重性，而且也已为绝大多数软件工作者所熟悉，所以本书仍将沿用它。

具体来说，软件危机主要有以下一些典型表现。

（1）软件开发无计划性

由于缺乏软件开发的经验和有关软件开发数据的积累，使得开发工作的计划很难制定。主观盲目地制定计划，执行起来和实际情况有很大差距，致使常常突破经费预算。对于工作量估计不准确，进度计划无法遵循，开发工作完成的期限一拖再拖。已经拖延了的项目，为了加快进度而增加人力，结果适得其反，不仅未加快，反而更加延误了。在这种情况下，软件开发的投资者和软件的用户对软件开发工作既不满意，也不信任。

（2）软件需求不充分

用户对“已完成的”软件系统不满意的现象经常发生。软件开发人员常常在对用户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，就仓促上阵匆忙着手编写程序。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。

（猿） 软件开发过程无规范

开发过程没有统一的、公认的方法论和规范指导，参加的人员各行其是。加之不重视文字资料工作，使设计和实现过程的资料很不完整；或是忽视了每个人的工作与其他人的接口部分，发现了问题就修修补补，这样的软件很难维护。

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，这些文档资料更是至关重要、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。

（源） 软件产品无测评手段

未能在测试阶段充分做好检测工作，提交给用户的软件质量差，在运行中暴露出大量的问题。在应用领域工作的不可靠软件，轻者影响系统的正常工作，重者发生事故。

这些矛盾表现在具体的软件开发项目上，最突出的实例就是美国 陨月酝 公司在 员怨远猿 年至 员怨远远 年开发的 陨月酝猿远园 机的操作系统。这一项目花了 缘园园园 人一年的工作量，最多时有 员园园园 人投入开发工作，写出了近 员园园 万行源程序。尽管投入了这样多的人力和物力，得到的结果却是非常糟的。据统计，这个操作系统每次发行的新版本都是从前一版本中找出 员园园园 个程序错误而修正的结果。可以设想，这样的软件质量糟到什么地步。难怪这个项目的负责人 云援阅援月则燥燥噪泽 事后总结他在组织开发过程中的沉痛教训时说：“……正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深。最后无法逃脱灭顶的灾难……程序设计工作正像这样一个泥潭……一批批程序员被迫在泥潭中拼命挣扎……谁也没有料到问题竟会陷入这样的困境……”陨月酝猿远园 操作系统的历史教训成为软件开发项目的典型事例为人们所汲取。

以上这些矛盾多少描绘了软件危机的某些侧面，如果不能突破这些障碍，进而摆脱困境，软件的发展是没有出路的。

圆援 消除软件危机的途径

必须充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、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所积累的行之有效的原理、技术和方法，特别要吸取几十年来人类从事计算机硬件研究和开发的经验教训。

应该开发和使用更好的软件工具。正如机械工具可以“放大”人类的体力一样，软件工具可以“放大”人类的智力。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。

总之，为了消除软件危机，既要有技术措施（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新兴学科。

1.1 软件工程

1.1.1 什么是软件工程

概括地说，软件工程是指导计算机软件开发和维护的工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明是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，以便经济地开发出高质量的软件并有效地维护它，这就是软件工程。

下面给出软件工程的几个定义。

1983 年 IEEE 给软件工程下的定义是："软件工程是开发、运行、维护和修复软件的系统方法。"这个定义相当概括，它主要强调软件工程是系统方法而不是某种神秘的个人技巧。

Fritz Bauer 认为："软件工程学是为了在成本限额以内按时完成开发和修改软件产品所需要的系统生产和维护技术及管理学科。"这个定义明确指出了软件工程的目标是在成本限额内按时完成开发和修改软件的工作，同时也指出了软件工程包含技术和管理两方面的内容。

Barry Boehm 给出了下述定义："软件工程是为了经济地获得可靠的且能在实际机器上有效地运行的软件，而建立和使用的完善的工程化原则。"这个定义不仅指出软件工程的目标是经济地开发出高质量的软件，而且强调了软件工程是一门工程学科，应该建立并使用完善的工程化原则。

1993 年 IEEE 进一步给出了一个更全面的定义。

软件工程是：①把系统化的、规范的、可度量的途径应用于软件开发、运行和维护的过程，也就是把工程化应用于软件中；②研究①中提到的途径。

认真研究上述这些关于软件工程的定义，有助于我们建立起对软件工程这门工程学科的全面的整体认识。

1.1.2 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了"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 Boehm 综合这些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原理的最小集合。这 7 条原理是互相独立的，其中任意 1 条原理的组合都不能代替另一条原理，因此，它们是缺一不可的最小集合；然而这 7 条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) 用分阶段的生命周期计划严格管理

有人经统计发现，在不成功的软件项目中有一半左右是由于计划不周造成的。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的生命周期中，需要完成许多性质各异的工作。这条基本原理意味着，应该把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。Barry Boehm 认为，在软件的整个生命周期中应该

制定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。

(圆) 坚持进行阶段评审

当时已经认识到，软件的质量保证工作不能等到编码阶段结束之后再进行。这样认为至少有两个理由：第一，大部分错误是在编码之前造成的。例如，根据 月[illegible]等人的统计，设计错误占软件错误的 远[illegible]，编码错误仅占 猿[illegible]；第二，错误发现与改正得越晚所需付出的代价就越高。因此，在每个阶段都进行严格的评审，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。

(猿) 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难度的代价。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。显然不能硬性禁止客户提出改变需求的要求，而只能采用科学的产品控制技术来顺应这种要求。也就是说，当改变需求时，为了保持软件各个配置万分的一致性，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，它们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：一切有关个性软件的建议，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，获得批准以后才能实施修改。绝对不能谁想修改软件 (包括尚在开发过程中的软件)，就随意进行修改。

(源) 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。圆[illegible]世纪 远[illegible]年代末提出的结构程序设计技术，已经成为绝大多数人公认的先进的程序设计技术。以后又进一步发展出各种结构分析 ([illegible]) 与结构设计 ([illegible]) 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。实践表明，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。

(缘) 结果应能清楚地审查

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，从而使软件产品的开发过程比平时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，从而使所得到的结果能够被清楚地审查。

(远) 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的素质应该好，而人数则不宜过多。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至几十倍，而且素质高的人员所开发的软件中的错误明显少于素质低的人员所开发的软件中的错误。此外，随着开发小组人员数目的增加，因为交流情况讨论问题而造成的通信开销也急剧增加。当开发小组人员数为 晕时，可能

的通信路径有 晕(晕原员 辕圆)条，可见随着人数 晕的增大，通信开销将急剧增加。因此，组成人员少而精的开发小组是软件工程的一条基本原理。

(苑) 承认不断改进软件工程实践的必要性

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，仅有上述六条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，能跟上技术的不断进步。因此，[illegible]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，例如，收集进度和资源耗费数据，收集出错类型和问题报告数据等。这些数据不仅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。

猿.圆 软件工程方法学

正如前面已经讲过的，软件工程包含技术和管理两方面的内容，是管理与技术的紧密结合。

所谓管理就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。

通常把在软件生命周期全过程中使用的一整套技术的集合称为方法学 ([illegible])，也称为范型 ([illegible])。在软件工程范畴中，这两个词的含义基本相同。

软件工程方法学包括三个要素，这就是方法、工具和过程。其中，方法是完成软件开发的各项任务的技术方法，回答“如何做”的问题；工具是为方法的运用提供自动的或半自动的软件支撑环境，回答“用什么做”的问题；过程是为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤，回答“做了什么”的问题。

目前使用得最广泛的软件工程方法学，分别是传统方法学和面向对象方法学。传统方法学也称为生命周期方法学或结构化范型。它采用结构化技术 (结构化分析、结构化设计、结构化程序设计和结构化测试) 来完成软件开发的各项任务并使用适当的软件工具或软件工程环境来支持结构化技术的运用。这种方法学把软件生命周期的全过程依次划分成若干个阶段，然后顺序地逐步完成各个阶段的任务。采用这种方法学开发软件的时候，从对任务的抽象逻辑分析开始，一个阶段一个阶段地进行开发。前一个阶段任务的完成是开始进行后一个阶段工作的前提和基础，而后一阶段任务的完成通常是使前一阶段提出的解法更进一步具体化，加进了更多的实现细节。每一个阶段的开始和结束都有严格标准，对于任何两个相邻的阶段而言，前一阶段的结束标准就是后一阶段的开始标准。在每一个阶段结束之前都必须通过确认后这个阶段才算结束；如果检查通不过，则必须进行必要的返工，并且返工后还要再经过审查。审查的一条主要标准就是每个阶段都应该交出“最新式的”(即和所开发的软件完全一致的) 高质量的文档资料，从而保证在软件开发工程结束时有一个完整准确的软件配置交付使用。文档是通信的工具，它们清楚准确地说明了到这个时候为止，关于该项工程已经知道了什么，同时确立了下一步工作的基础。此外，文档也起备忘录的作用，如果文档不完整，那么一定是某些工作忘记做了，在进入生命周期的下一阶段之前，必须实践这些遗漏的细节。在完成生命周期每个阶段的任务时，应该采用适合该阶段任务特点的系统的技术方法——结构化分析、结构化设计、结构程序设计或结构化测试技术。

把软件生命周期划分成若干个阶段，每个阶段的任务相对独立，而且比较简单，便于不同人员分工协作，从而降低了整个软件开发工程的困难程度；在软件生命周期的每个阶段都采用科学的管理技术和良好的技术方法，而且在每个阶段结束之前都从技术和管理两个角度进行严格的审查，合格之后才开始下一阶段的工作，这就使软件开发工程的全过程以一种有条不紊的方式进行，保证了软件的质量，特别是提高了软件的可维护性。总之，采用生命周期方法学可以大大提高软件开发的成功率，软件的生产率也能明显提高。

目前，生命周期方法学仍然是人们在开发软件过程中使用得非常广泛的软件工程方法学。由于这种方法学历史悠久，为广大软件工程师所熟悉，而且在开发某些类型的软件时也比较有效，因此，在相当长一段时间内这种方法学还会有生命力。此外，如果没有完全理解传统方法以及这种传统方法与面向对象方法的区别也就不可能理解面向对象方法学为何优于传统方法学。因此，本书不仅讲述面向对象方法学，也讲述传统方法学。

当软件规模较大，或者对软件的需求是模糊的或随时间变化的时候，使用结构化范型开发软件往往不成功。

结构化范型只能获得有限成功的一个重要原因是，这种技术要么面向行为（即对数据的操作），要么面向数据，没有既面向数据又面向行为的结构化技术。众所周知，软件系统本质上是信息处理系统。离开了操作便无法更改数据，而脱离了数据的操作是毫无意义的。数据和对数据的处理原本是密切相关的，把数据和处理人为地分离成两个独立的部分，自然会增加软件开发与维护的难度。与传统方法相反，面向对象方法把数据和行为看成同等重要，它是一种以数据为主线，把数据和对数据的操作紧密地结合在一起的方法。

概括地说，面向对象方法具有下述四个要点。

• 把对象（韵[illegible]）作为融合了数据及在数据上的操作行为的统一的软件构件。面向对象程序是由对象组成的，程序中任何元素都是对象，复杂对象由比较简单的对象组合而成。

• 把所有对象都划分成类（悦[illegible]）。每个类都定义了一组数据和一组操作，类是对具有相同数据和相同操作的一组相似对象的定义。数据用于表示对象的静态属性，是对象的状态信息，而施加于数据之上的操作用于实现对象的动态行为。

• 按照父类（或称为基类）与子类（或称为派生类）的关系，把若干个相关类组成一个层次结构的系统（也称为类等级）。在类等级中，下层派生类自动拥有上层基类中定义的数据和操作，这种现象称为继承。

• 对象彼此之间仅能通过发送消息互相联系。对象与传统数据有本质区别，它不是被动地等待外界对它施加操作，相反，它是进行处理的主体，必须向它发消息请求它执行它的某个操作以处理它的数据，而不能从外界直接对它的数据进行处理。也就是说，对象的所有私有信息都被封装在该对象内，不能从外界直接访问，这就是通常所说的封装性。

面向对象方法学的出发点和基本原则，是尽可能模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界解决问题的方法与过程，从而使描述问题的问题空间（也称为问题域）与实现解法的解空间（也称为求解域）在结构上尽可能一致。

传统方法学强调自顶向下顺序地完成软件开发的各阶段任务。事实上，人类认识客观世界解决现实问题的过程，是一个渐进的过程，人的认识需要在继承以前的有关知识的基础上，经过多次反复才能逐步深化。在人的认识深化的过程中，既包括了从一般到特殊的演绎

思维过程，也包括了从特殊到一般的归纳思维过程。

用面向对象方法学开发软件的过程，是一个主动地多次反复迭代的深化过程。面向对象方法在概念和表示方法上的一致性，保证了在各项开发活动之间的平滑（无缝）过渡。面向对象方法普遍进行的对象分类过程，支持从特殊到一般的归纳思维过程；通过建立类等级而获得的继承性，支持从一般到特殊的演绎思维过程。

正确运用面向对象方法学开发软件，则最终的软件产品由许多较小的尽可能独立的对象组成，而且大多数对象都与现实世界中的实体相对应，因此，降低了软件产品的复杂性，提高了软件产品的可理解性，简化了软件的开发和维护工作。由于对象是相对独立的实体，容易在以后的软件产品中重复使用，因此，面向对象范型的另一个重要优点是促进了软件重用。面向对象方法特有的继承性，进一步提高了面向对象软件的可重用性。

员圆摇软件的生存周期及其开发模型

员圆员摇软件生存周期

软件生存周期是借用工程中产品生存周期的概念而得来的。引入软件生存周期概念对于软件生产的管理、进度控制有着非常重要的意义，使得软件生产有相应的模式、相应的流程、相应的工序和步骤。

软件生存周期是指一个软件从提出开发要求开始直到该软件报废为止的整个时期。把整个生存周期划分为若干阶段，使得每个阶段有明确的任务，使规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。

软件生存周期的各阶段有不同的划分。软件规模、种类、开发方式、开发环境以及开发使用方法都影响软件周期的划分。在划分软件生存周期阶段时，应遵循的一条基本原则是各阶段的任务应尽可能相对独立，同一阶段各项任务的性质尽可能相同，从而降低每个阶段任务的复杂程度，简化不同阶段之间的联系，有利于软件项目开发的组织管理。通常，软件生存周期包括可行性分析和项目开发计划、需求分析、概要设计、详细设计、编码、测试、维护等活动，可以将这些活动以适当方式分配到不同阶段去完成。

员．可行性分析和项目开发计划

这个阶段必须要回答的问题是“要解决的问题是什么”，该问题有可行的解决办法吗？若有解决问题的办法，需要多少费用？需要多少资源？需要多少时间？要回答这些问题，就要进行问题定义、可行性分析，制定项目开发计划。

可行性分析的任务首先需要进行概要的分析研究，初步确定项目的规模和目标，确定项目的约束和限制，把它们清楚地列举出来。然后，分析员进行简要的需求分析，抽象出该项目的逻辑结构，建立逻辑模型。从逻辑模型出发，经过压缩的设计，探索出若干种可供选择的主要解决办法，对每种解决方法都要研究它的可行性。可从以下三个方面分析研究每种解决方法的可行性。

（员）技术可行性

对要开发项目的功能、性能、限制条件进行分析，确定在现有的资源条件下，技术风险

有多大，项目是否能实现。这里的资源包括已有的或可以搞到的硬件、软件资源，现有技术人员的技术水平和已有的工作基础。

技术可行性常常是最难解决的方面，因为项目的目标、功能、性能比较模糊。技术可行性一般要考虑的情况包括：

• 开发的风险：在给出的限制范围内，能否设计出系统并实现必须的功能和性能？

• 资源的有效性：可用于开发的人员是否存在问题？可用于建立系统的其他资源是否具备？

• 技术：相关技术的发展是否支持这个系统？

开发人员在评估技术可行性时，一旦估计错误，将会出现灾难性后果。

(圆 经济可行性

进行开发成本的估算以及取得效益的评估，确定要开发的项目是否值得投资开发。

对于大多数系统，一般衡量经济上是否合算，应考虑一个“底线”，经济可行性研究范围较广，包括成本效益分析、公司经营长期策略、开发所需的成本和资源、潜在的市场前景等。

(猿 社会可行性

要开发的项目是否存在任何侵犯、妨碍等责任问题，要开发项目的运行方式在用户组织内是否行得通，现有管理制度、人员素质、操作方式是否可行。

社会可行性所涉及的范围也比较广，包括合同、责任、侵权、用户组织的管理模式及规范，其他一些技术人员常常不了解的陷阱等。

典型的可行性研究有下列几个步骤：

(员 确定项目规模和目标

分析员对有关人员进行调查访问，仔细阅读和分析有关的材料，对项目的规模和目标进行定义和确认，清晰地描述项目的限制和约束，确保分析员正在解决的问题确实是要解决的问题。

(圆 研究正在运行的系统

正在运行的系统可能是一个人工操作的系统，也可能是旧的计算机系统，要开发一个新的计算机系统来代替现有系统。因此，现有的系统是信息的重要来源，要研究它的基本功能，存在什么问题，运行现有系统需要多少费用，对新系统有什么新的功能要求，新系统运行时能否减少使用费用等等。

应该收集、研究、分析现有系统的文档资料，实地考察现有系统，在考察的基础上，访问有关人员，然后描绘现有系统的高层系统流程图，与有关人员一起审查该系统流程图是否正确。这个系统流程图反映了现有系统的基本功能和处理流程。

(猿 建立新系统的高层逻辑模型

根据对现有系统的分析研究，逐渐明确了新系统的功能（详见第 圆章）、处理流程以及所受的约束，然后使用建立逻辑模型的工具——数据流图和数据字典，来描述数据在系统中的流动和处理情况。注意，现在还不是软件需求分析阶段，不是完整、详细地描述，只是概括地描述高层的数据处理和流动。

(源 导出和评价各种方案

分析员建立了新系统的高层逻辑模型之后，要从技术角度出发，提出实现高层逻辑模型

的不同方案，即导出若干较高层次的物理解法。根据技术可行性、经济可行性、社会可行性对各种方案进行评估，去掉行不通的解法，就得到了可行的解法。

⑷ 推荐可行的方案

根据上述可行性研究的结果，应该决定该项目是否值得去开发的问题。若值得开发，那么可行的解决方案是什么，并且说明该方案可行的原因和理由。该项目是否值得开发的主要因素是从经济上看是否合算，这就要求分析员对推荐的可行方案进行成本—效益分析。

⑸ 编写可行性研究报告

将上述可行性研究过程的结果写成相应的文档，即可行性研究报告，提请用户和使用部门仔细审查，从而决定该项目是否进行开发，是否接受可行的实现方案。

②需求分析

需求分析阶段的任务不是具体地解决问题，而是准确地确定软件系统必须做什么，确定软件系统必须具备哪些功能。

用户了解他们所面对的问题，知道必须做什么，但是通常不能完整、准确地表达出来，也不知道怎样用计算机解决他们的问题。软件开发人员知道怎样用软件完成人们提出的各种功能要求，但是，对用户的具体业务和需求不完全清楚，这是需求分析阶段的困难所在。

系统分析员要和用户密切配合，充分交流各自的理解，充分理解用户的业务流程，完整地、全面地收集、分析用户业务中的信息和处理，从中分析出用户要求的功能和性能，并完整地、准确地表达出来。这一阶段要给出软件需求规格说明书。

③概要设计

在概要设计阶段，开发人员要把确定的各项功能需求转换成需要的体系结构。在该体系结构中，每个成分都是意义明确的模块，即每个模块都和某些功能需求相对应，因此，概要设计就是设计软件的结构，明确该结构由哪些模块组成，这些模块的层次结构是怎样的，这些模块的调用关系是怎样的，每个模块的功能是什么。同时还要设计该项目的应用系统的总体数据结构和数据库结构，即应用系统要存储什么数据，这些数据是什么样的结构，它们之间有什么关系。

④详细设计

详细设计阶段就是为每个模块完成的功能进行具体描述、要把功能描述转变为精确的结构化的过程描述，即该模块的控制结构是怎样的，先做什么，后做什么，有什么样的条件判定，有些什么重复处理等，并用相应的表示工具把这些控制结构表示出来。

⑤编码

编码阶段就是把每个模块的控制结构转换成计算机可接受的程序代码，即写成以某种特定程序设计语言表示的“源程序清单”。当然，写出的程序应是结构好、清晰易读，并且与设计相一致的。

⑥测试

测试是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件的各个组成部分。测试分为单元测试、集成测试、确认测试。单元测试是查找各模块在功能和结构上存在的问题。确认测试是将各模块按一定顺序组装起来进行测试，主要是查找各模块之间存在的问题。集成测试是按需求说明书上的功能逐项测试，发现不满足用户需求的问题，

决定开发的软件是否合格，能否交付用户使用。

员.圆.猿 维护

软件维护是软件生存周期中时间最长的阶段。已交付的软件投入正式使用后，便进入软件维护阶段，它可以持续几年甚至几十年。软件运行过程中可能由于各方面的原因，需要对它进行修改。其原因可能是运行中发现了软件隐含的错误而需要修改；也可能是为了适应变化了的软件工作环境而需要做适当变更；也可能是因为用户业务发生变化而需要扩充和增强软件的功能等。

员.猿 软件开发模型

根据软件生产工程化的需要，生存周期的划分也有不同，从而形成了不同的软件生存周期模型（有时也称为软件生存周期模型），或称软件开发模型。软件开发模型总体来说有传统的瀑布模型和后来兴起的快速原型模型，具体可分为瀑布模型、快速模型、软件重用开发模型和螺旋模型，以下分别加以介绍。

员.猿.员 瀑布模型（有时也称为生存周期模型）

瀑布模型遵循软件生存周期的划分，明确规定每个阶段的任务，各个阶段的工作顺序展开，恰如奔流不息拾级而下的瀑布。

瀑布模型把软件生存周期分为计划时期、开发时期、运行时期三个时期。这三个时期又可细分为若干个阶段：计划时期可分为问题定义、可行性研究两个阶段，开发时期分为需求分析、概要设计、详细设计、程序设计、软件测试等阶段，运行时期则边运行、边维护（见图 员.猿.员）。

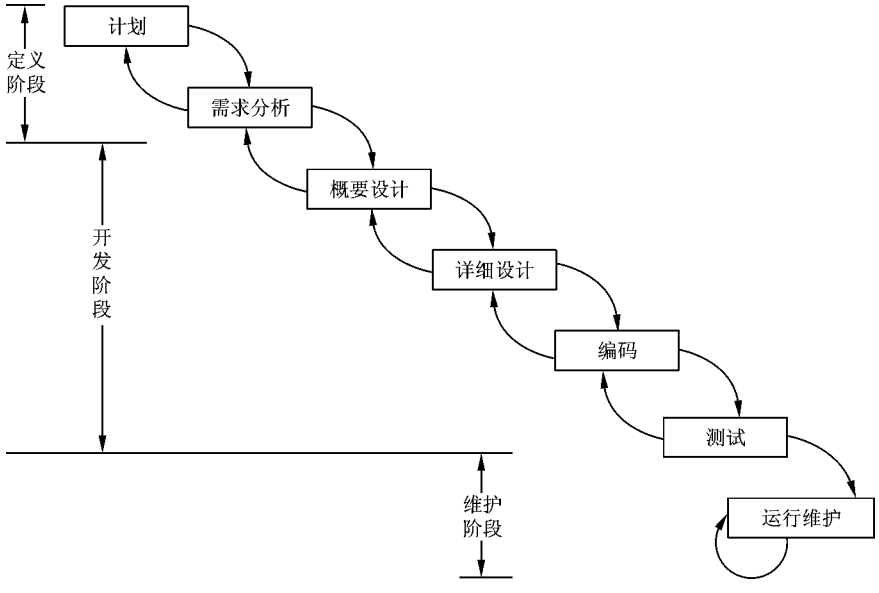

图 员.猿.员 适用于结构化开发技术的典型的瀑布模型