普通高等院校计算机类专业系列教材

Software Engineering

软件工程

江开耀　张俊兰　李　晔　编著
陆丽娜　主审

西安电子科技大学出版社
http://www.xduph.com

普通高等院校计算机类专业系列教材

软件工程

Software Engineering

江开耀　张俊兰　李　晔　编著

陆丽娜　主审

西安电子科技大学出版社

2003

内 容 简 介

本书从实用角度介绍了软件工程的基础知识和软件工程技术方法。全书分三部分，共 17 章。第一部分介绍软件工程基础知识与传统的软件工程方法，主要内容是软件工程的基本概念和基于结构化方法的软件工程技术，包括结构化的分析、设计、编码与测试；第二部分讲述了面向对象技术的基本概念和面向对象的分析、设计和实现技术；第三部分综合介绍了软件工程项目管理方法，主要内容包括工程估算、软件度量、风险防范、软件质量保证和软件配置管理等方面的知识。

本书主要供初学软件工程的读者使用，可以作为高等院校计算机科学与技术专业本科教材，也可作为专科学生的参考教材及软件工程师的参考书。建议学时为 50 课时。

★本书配有电子教案，有需要的老师可与出版社联系，免费索取。

图书在版编目（CIP）数据

软件工程 = Software Engineering / 江开耀等编著. —西安：西安电子科技大学出版社，2003.8
(普通高等学校计算机类专业系列教材)
ISBN 7 - 5606 - 1272 - 5
Ⅰ. 软… Ⅱ. 江… 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（2003）第 063790 号

责任编辑　马晓娟　张 友
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话：(029)8242885　8201467　邮　编　710071
http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com
经　　销　新华书店
印刷单位　西安文化彩印厂
版　　次　2003 年 8 月第 1 版　2003 年 8 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 17.875
字　　数　420 千字
印　　数　1～4 000 册
定　　价　20.00 元
ISBN 7-5606-1272-5/TP・0670(课)

XDUP 1543001-1

普通高等院校计算机类专业系列教材

编审专家委员会名单

前　言

随着微电子技术的飞速发展，硬件设备的功能急剧提升、价格大幅度下降。据统计，硬件系统的性能价格比平均每十年提高两个数量级，质量也在稳步提高。在计算机应用领域的不断拓展和深入的过程中，对软件产品的数量、种类、功能、性能的需求也在不断攀升。但是，由于软件生产过程和软件产品固有的不可视特征以及缺乏工程化的软件生产模式，导致了软件系统的开发成本逐年上升、产品质量难以控制、成品软件不便于维护、生产率远远跟不上实际需求等。这样，软件的生产与维护就成为了限制计算机应用系统继续快速发展的关键因素。西方计算机科学家把在软件开发和维护过程中遇到的一系列严重问题统称为“软件危机”，并从20世纪60年代末开始认真研究解决软件危机的方法，从而逐步形成了一门新兴的学科——计算机软件工程学。

经过40年的发展，软件工程学已经取得了长足的进步。对软件工程过程、软件开发技术和软件工程工具的研究与改进正引导着我们逐渐克服软件危机的影响，从传统的手工业生产模式向大工业化的软件生产模式转化。由传统的软件工程到计算机辅助软件工程，从结构化技术到面向对象技术，从只注重产品到同时关注过程与产品，软件工程学本身也在实践中不断发展并走向成熟，已经成为计算机科学技术的一个重要分支学科。

本书注重从实用角度介绍软件工程的基础知识和实用的软件工程技术方法，希望能够使读者对现代软件工程有一个较为全面的初步理解，并对实际的软件工程活动有所帮助。全书共三部分，17章。第一部分介绍软件工程基础知识与传统的软件工程方法，主要内容是软件工程的基本概念和基于结构化方法的软件工程技术，包括结构化的分析、设计、编码与测试等；第二部分讲述了面向对象技术的基本概念和面向对象的分析、设计和实现技术；第三部分综合介绍了软件工程项目管理方法，主要内容包括工程估算、软件度量、风险防范、软件质量保证和软件配置管理等方面的知识。

本书由西安工程科技学院、延安大学、西安理工大学、宝鸡文理学院相关同志通力合作完成，并由西安交通大学陆丽娜教授主审。西安工程科技学院江开耀教授担任本书主编，负责编写了第1～3章和第13章，并负责统稿；延安大学张俊兰教授担任副主编，编写了第8～11章；西安理工大学李晔老师编写了第4～6章；宝鸡文理学院吴莉霞老师编写了第7章；西安工程科技学院王继川副教授编写了第12章、马柯副教授编写了第14章、陈建斌老师编写了第15章、江山老师编写了第16、17章。另外，西安理工大学李长河教授参加了编写大纲的讨论，提出了很好的建议。

在本书写作过程中，得到了西安电子科技大学出版社马晓娟老师的多方指导和大力协助，提出了许多很好的建议，在此一并致谢。

编　者

2003.4

目　录

第一部分　传统的软件工程

第二部分 面向对象的软件工程

第三部分 软件工程项目管理

第一部分 传统的软件工程

软件工程引论

第 1 章

任何一个实际的计算机应用系统都由硬件和软件两大部分组成。随着微电子技术的飞速发展，硬件设备的功能急剧提升、价格大幅度下降，设备生产能力有了很大的发展。据统计，硬件系统的性能价格比平均每十年提高两个数量级，质量也在稳步提高。在计算机应用领域的不断拓展和深入的过程中，对软件产品的数量、种类、功能、性能的需求在不断攀升。但是，由于软件生产的人工成本居高不下，导致了软件系统的开发成本逐年上升、人工产品的质量难以控制、成品软件不便于维护、生产率也远远跟不上实际需求。这样，软件的生产与维护就成为了限制计算机应用系统继续快速发展的关键因素。西方计算机科学家把在软件开发和维护过程中遇到的一系列严重问题统称为“软件危机”，并从 20 世纪 60 年代末开始认真研究解决软件危机的方法，从而逐步形成了一门新兴的学科——计算机软件工程学。

1.1 软件产品的概念与特征

1.1.1 软件产品的概念与分类

就本质而言，软件就是一个信息转换器，它的功能不外是产生、管理、获取、修改、显示或转换信息。它担任着双重角色，首先，它是一种产品，表达了由计算机硬件体现的计算潜能；其次，它又是开发和运行产品的载体，是计算机控制(操作系统)、信息通信(网络)的基础，也是创建和控制其他软件(软件工具和开发环境)的基础。

对于软件的一种公认的解释是：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、数据及其相关文档的完整集合。其中，程序是为实现设计的功能和性能要求而编写的指令序列；数据是使指令能够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资料。

根据用途划分，软件可以大致划分成如下类别：

(1) 系统软件：就一般情况来说，系统软件是为其他软件服务的软件。系统软件与计算机硬件交互频繁，处理大量的确定或不确定的复杂数据，往往需要具有多用户支持、资源

精细调度、并发操作管理、多种外部设备接口支持等项功能。

(2) 实时软件：管理、分析、控制现实世界中所发生的事件的软件称为实时软件。它一般有数据采集、数据分析、输出控制等三方面的功能。实时软件需要保持一个现实任务可以接受的响应时间，即必须保证能够在严格限定的时间范围内对输入做出响应。

(3) 商业管理软件：商业信息处理是最大的软件应用领域，包括常规的数据处理软件和一些交互式的计算处理(如 POS 软件)软件。它的基本功能是将已有的数据重新构造，变换成一种可以辅助商业操作和管理决策的形式。在这个过程中，几乎都要涉及到对于大型数据库的访问。各类管理信息系统(MIS)、企业资源计划(ERP)、客户关系管理(CRM)等都是典型的商业管理软件。

(4) 工程与科学计算软件：此类软件的特征是要实现特定的“数值分析”算法。例如离散傅立叶变换、有限元分析、演化计算等等。CAD/CAM 软件一般也可以归属到这一类型中来。

(5) 嵌入式软件：驻留在专用智能产品的内存中，用于控制这些产品进行正常工作，完成很有限、很专业的功能的软件。例如各类智能检测仪表、数码相机、移动电话、微波炉等智能产品都必须在嵌入式软件的支持下才能正常工作。

(6) 人工智能软件：利用非数值算法去解决复杂问题的软件。各类专家系统、模式识别软件、人工神经网络软件都属于人工智能软件。

(7) 个人计算机软件：文字处理系统、电子表格、游戏娱乐软件等等。

此外，还可以根据软件的规模(代码行及开发工作量，如表 1.1)、软件的工作方式、使用频度、失效后造成的影响等对软件产品进行分类。

表 1.1　根据规模进行软件分类

软件规模类别	参加人员数	开发期限	产品规模(源代码行数)
微型	1	1～4 周	0.5 k
小型	1	1～6 月	1～2 k
中型	2～5	1～2 年	5～50 k
大型	5～20	2～3 年	50～100 k
甚大型	100～1000	4～5 年	1 M
极大型	2000～5000	5～10 年	1～10 M

1.1.2　软件产品的特征

在制造硬件时，人的创造性的劳动过程(分析、设计、建造、测试)能够完全转换成物理的形式，但软件是逻辑的而不是物理的产品，因此软件具有和硬件完全不同的特征：

(1) 软件是一种逻辑实体，具有抽象性。我们可以把软件保存在媒体介质上，但却无法直接看到软件的形态，因而必须通过运行、观察、分析、思考、判断才能够了解软件的功能、性能及其他特性。换句话说，软件产品具有明显的非可视特征。

(2) 软件的生产与硬件不同。软件是由开发或工程化而形成的，不是由传统意义上的制造过程生产的。虽然软件开发和硬件制造之间有一些相似之处，可是两者在本质上是不同的。这两者都能够通过良好的设计获得高质量的产品，但即使有了良好的设计和优秀的样

品，硬件在批量制造过程中仍然可能引入质量问题，这种情况对于软件而言几乎不存在。软件在开发完毕，形成为产品之后，其批量制造过程只是简单的拷贝/复制；软件的开发和硬件的制造都依赖于人，但参与者和他们完成的工作之间的关系不同；两者的终极目的都是建造产品，但方法不同；软件的成本集中在开发过程上，而硬件生产的成本更多地表现在原材料消耗上。因此，软件项目开发过程不能完全像硬件制造过程那样来管理。

(3) 软件产品不会“磨损”。和硬件产品类似，软件产品也会出现故障。所不同的是，硬件产品的故障多来自外在条件导致的“磨损”或“老化”，而软件产品如果发生故障，无一例外的是在设计开发过程中留有隐患。因此，硬件的故障可以通过简单的更换部件解决，而软件的故障必须通过全面的软件维护活动才有望克服。同时，不完善的维护活动又可能在软件中注入新的故障，导致软件质量的“退化”。也就是说，软件故障的修复要比硬件故障的修复复杂得多。因此，衡量软件产品质量的一个重要指标就是它的“可维护性”。图 1.1 是软、硬件产品的失效率曲线。

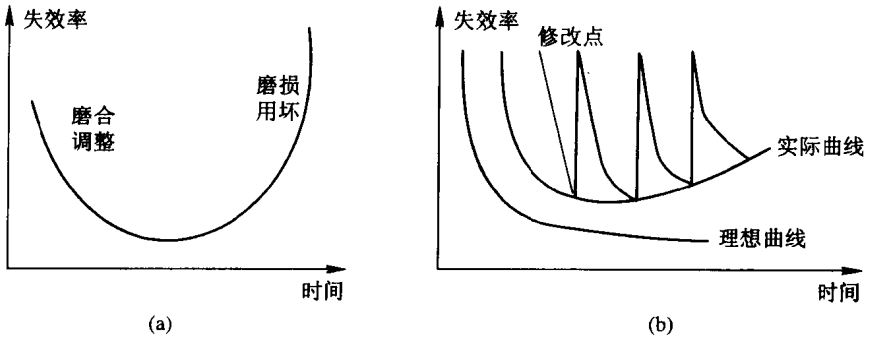

图 1.1　软件/硬件产品失效率曲线

1.1.3　软件发展的阶段划分

自从 20 世纪 40 年代第一台计算机问世以来，就有了“程序”的概念，可以认为它是软件的前身。经过了几十年的发展，人们对软件有了更为深刻的认识，在这几十年中，软件开发经历了三个发展阶段：20 世纪 50～60 年代属于程序设计阶段；20 世纪 60～70 年代为程序系统阶段；20 世纪 70 年代之后进入软件工程阶段。各阶段的特点与区别见表 1.2。

表 1.2　计算机软件发展的三个阶段及其特点

阶　段 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、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	设计和测试	整个软件生命周期
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、中型开发机构
软件规模	小型	中、小型	大、中、小型

续表

阶段 / 特点	程序设计	程序系统	软件工程
决定质量的因素	个人技术	小组技术水平	技术与管理水平
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、集成开发环境、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、计算机辅助软件工程
维护责任者	程序设计者	开发小组	专职维护人员
硬件的特征	高价、存储量小、可靠性差	降价，速度、容量和可靠性明显提高	向超高速、大容量、网络化、微型化方向发展
软件的特征	完全不受重视	软件的技术发展不能满足需求，出现软件危机	开发技术有进步，但仍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。

1.2 软件危机

1.2.1 软件危机及其表现

现代计算机应用系统中，软件的地位日益重要和突出。如何满足日益增长的软件需求，如何维护应用中的大量已有软件，已经成为了计算机应用系统进一步发展的瓶颈。1968 年，北大西洋公约组织的计算机科学家们在联邦德国召开的国际会议上讨论了软件危机问题，同时也是在这个会议上提出了“软件工程”这个名词，导致了一门新的工程学科的正式诞生。

简单地说，所谓软件危机，就是指在软件开发和软件维护过程中所存在的一系列严重问题。具体地说，软件危机具有如下一些表现：

(1) 软件开发没有真正的计划性，对软件开发进度和软件开发成本的估计常常很不准确，计划的制定带有很大的盲目因素，因此工期超出、成本失控的现象经常困扰着软件开发者。

(2) 对于软件需求信息的获取常常不充分，软件产品往往不能真正地满足用户的实际需求。

(3) 缺乏良好的软件质量评测手段，从而导致软件产品的质量常常得不到保证。

(4) 对于软件的可理解性、可维护性认识不够；软件的可复用性、可维护性不如人意。有些软件因为过于“个性化”，甚至是难以理解的，更谈不上进行维护。缺乏可复用性引起的大量重复性劳动极大地降低了软件的开发效率。

(5) 软件开发过程没有实现“规范化”，缺乏必要的文档资料或者文档资料不合格、不准确，难以进行专业维护。

(6) 软件开发的人力成本持续上升，如美国在 1995 年的软件开发成本已经占到了计算

机系统成本的90%(如图1.2所示)。

(7) 缺乏自动化的软件开发技术，软件开发的生产率依然低下，远远满足不了急剧增长的软件需求(如图1.3所示)。

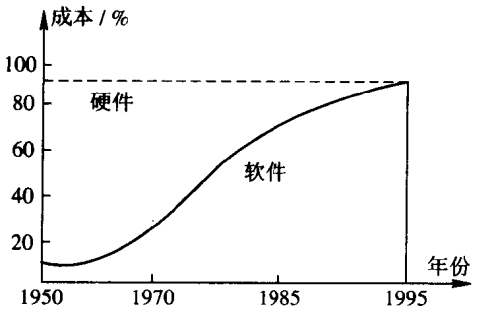

图1.2 计算机系统硬件、软件成本比例变化

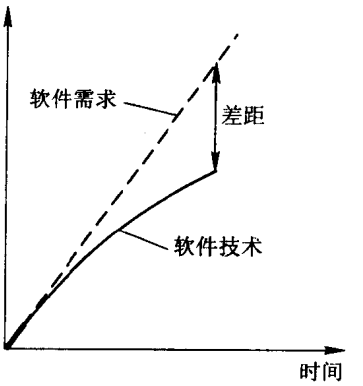

图1.3 软件技术的发展落后于需求

软件危机曾经是历史上的阴影，目前软件工程界也仍然在一定程度上受到它的影响。软件工程概念的提出，正是为了克服软件危机。自1968年以来，随着软件工程学的不断发展，软件危机得到了一定程度的遏制，但还远远没有被彻底解决。《The Standish Group. Chaos. 1995.》一文报告了20世纪90年代中期美国商用软件产业的情况：1995年美国公司取消了810亿美元的软件项目；在所考察的软件项目中，在完成前就取消了其中的31%；53%的软件项目进度拖延，通常拖延的时间超过预定工期50%以上；只有9%的大型软件项目能够及时交付且费用不超支(对中型和小型软件公司来说这一数据为16%)。

从上面的统计数据不难看出，在软件开发过程中，危机的影响依然存在。再回顾一下世纪之交时，为防治软件“千年虫”问题所投入的巨大人力与物力，也反映了软件维护工作的艰巨与困难。

1.2.2 产生软件危机的原因

软件危机的存在是不争的事实。产生软件危机的原因可以归纳为主、客观两个方面。

从客观上来看，软件不同于硬件，它的生产过程和产品都具有明显的“不可视”特征，这就导致在完成编码并且上机运行之前，对于软件开发过程的进展情况较难衡量，软件产品的质量也较难进行先期评价，因此，对于开发软件的过程进行管理和控制比较困难。在软件工程的早期，制定详细的开发计划并且进行全程跟踪调控，对于所有的阶段产品和阶段工作进展进行技术审查和管理复审，可望在一定程度上克服“开发过程不可视”造成的消极影响。

此外，软件运行过程中如果发现了错误，那么必然是遇到了在开发时期(分析、设计、编码过程)引入的，在检测过程中没有能够检查出来的故障。对于此类故障的维护，通常意味着要修改早期的分析结果、设计结果并调整编码。由于软件产品的不可视特征，维护过程不像硬件产品维护时只要简单的更换损坏部件那样容易，这在客观上造成了软件难以维护的结果。利用足够的文档资料使不可视的产品可视化，有助于提升软件产品的可理解性

和可维护性。

从主观上分析，导致软件危机发生的另一大原因，可以归于在计算机系统发展的早期，软件开发的“个体化”特点，主要表现为忽视软件需求分析的重要性、忽视软件的可理解性、文档不完备、轻视软件的可维护性、过分强调编码技巧等等方面。

只有软件的用户才真正了解他们自己的需求。而且应当承认，用户一开始并不见得能够清晰、准确、无二意地表达自己的需求。软件开发人员需要做大量的、深入细致的调研工作，引导用户逐步准确、具体地描述软件的需求，才能够得到对问题、目标的正确认识，从而获得解决问题的恰当出发点，有望开发出真正能够满足用户需求的软件产品。在对用户的需求没有清楚的认识时就仓促进行程序编写，最终必然会导致开发工作的失败。

一般来说，软件产品从策划、定义、开发、使用与维护直到最后废弃，要经过一个漫长的时期，通常把这个时期称为软件的“生命周期”。可以将生命周期分作“软件定义”、“软件开发”和“运行与维护”三个阶段。

在软件定义阶段中，主要进行软件目标的策划、可行性研究和软件的需求分析工作，通过和用户多次交流，在所要开发的软件必须“作什么”方面和用户达成一致(当然在开发过程中也允许在严格的控制下进行需求变更)。

软件被定义之后，进入开发阶段，主要对软件的体系架构、数据结构和主要算法进行设计和编码实现。对于编码结果，还要按照规范进行测试后，才能最终交付使用。如前所述，在开发阶段也可能对于此前不够准确的软件定义结果进行调整。统计数据表明，在典型的软件工程过程中，编码工作量大约只占软件开发全部工作量的 15%～20%。

软件的运行与维护阶段在软件生命周期中占据的比例最大。在软件运行过程中，分析和设计阶段的一些遗留缺陷可能会逐步暴露；运行环境的演变也会对运行中的软件提出变更要求；用户新需求的提出则常常要求扩充现有软件的功能或者改进其性能，所有这些要求与问题都必须通过“软件维护”工作去解决。在维护过程中，必须注意保持所有软件工作产品之间的一致性。针对不同的需求，维护工作一般可以分为纠错性维护、适应性维护、扩充性维护和预防性维护等不同类型。

作为软件，应当有一个完整的配置。Boehm(美国著名的软件工程专家，加州州立大学教授)指出，“软件是程序以及开发、使用、维护程序所需要的所有文档”。所以，软件产品除包括程序之外，应当包括完整、准确、翔实的文档资料。主要的文档应当包括“需求规格说明书”、“体系结构设计说明书”、“详细设计说明书”、“安装手册”、“操作手册”、“系统管理员手册”等。缺乏必要的配置文档，将严重影响软件的可理解性，从而给软件的维护造成严重障碍。

做好包括项目策划、可行性研究、需求分析三项内容的软件定义工作，是提高软件质量、降低软件成本、保证开发进度的关键环节。

值得注意的严重问题是，在软件开发的不同阶段进行修改所付出的代价是极其不同的。在早期引入变动，涉及的面比较小，因而代价也比较低；在开发的中期，因为许多配置项(被标识的工作产品)已经完成，所以引入一个变动，就要对它所涉及的所有已经完成的配置项进行变更，不仅工作量大，而且逻辑上也更复杂，因此付出的代价剧增；如果在软件“已经完成”时再引入变更，更是要付出高得多的代价。根据美国一些软件公司的统计资料，软件开发后期引入一个变动比在早期引入相同变动所需付出的代价高 2～3 个数量级。图

1.4 定性地描绘了在不同时期引入一个变动需要付出的代价的变动趋势。图 1.5 是美国贝尔实验室统计得出的定量结果。

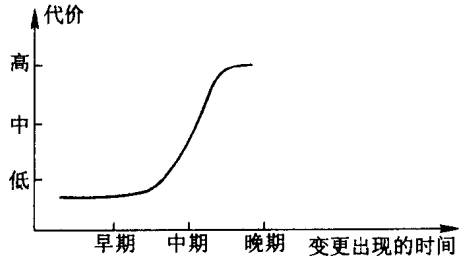

图 1.4 变更代价随时间变化的趋势示意

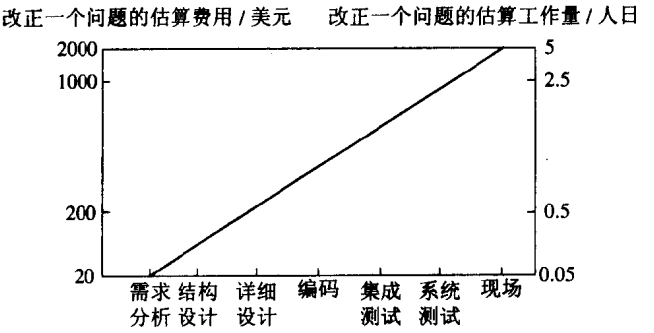

图 1.5 改正一个问题需要付出的代价

1.2.3 解决软件危机的途径

可以借鉴其他工程领域的成功经验，基于软件危机产生的主、客观原因，从软件工程技术和软件工程管理两方面来采取措施，防范软件危机的发生。

软件开发不是某种个体劳动的神秘技巧，而应当是一种组织良好、管理严密，分析、设计、编码、测试、品保等各类人员协同配合、共同完成的工程项目。在软件开发过程中，必须充分吸收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、概念、技术和方法，特别要注意吸收几十年来在计算机硬件研究和开发中积累的经验、教训。

从管理层面上考虑，应当注意推广和使用在实践中总结出来的开发软件的成功的技术和方法，并且探索更好的、更有效的技术和方法，注意积累软件开发过程中的经验数据财富，逐步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。建立适合于本组织的软件工程规范；制定软件开发中各个工作环节的流程文件、工作指南和阶段工作产品模板；实施针对软件开发全过程的计划跟踪和品质管理活动；为每一项工程开发活动建立配置管理库；实施严格的产品基线管理并建立组织的软件过程数据库和软件财富库；为各类员工及时提供必要的培训等等都是加强软件开发活动管理工作的有效手段。

从技术角度考虑，应当开发和使用更好的软件开发工具，提高软件开发效率和开发工作过程的规范化程度。在计算机软件开发的各个阶段，都有大量的繁琐重复的工作要做，在适当的软件工具的辅助下，开发人员可以把这类工作做的既快又好。目前广为使用的统一建模语言(UML)、各种配置管理工具、缺陷管理工具和自动测试工具都在软件工程活动中发挥了很好的作用。计算机辅助软件工程(CASE)更是目前备受重视的一个旨在实现软件开发自动化的新的领域。

1.3 软件工程的产生及其发展

1968 年，北大西洋公约组织的计算机科学家们在原联邦德国召开的国际会议上，针对软件危机的严峻形势，提出了把在其他工程领域中行之有效的一些工程学知识运用到软件

开发过程中来，从管理和技术两个方面研究如何更好地开发和维护计算机软件的设想。这也就是软件工程的基本思路。在这次会议上首次提出并使用了“软件工程”这一术语。

简单地说，软件工程是指导软件开发和维护的工程学科。它的核心思想是采用工程的概念、原理、技术和方法来开发和维护软件，把经过实践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，从而大大提高软件开发的成功率和生产率。

许多计算机专家都曾经描述过“软件工程”的定义。

Boehm 曾为软件工程下过定义：“运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料”。

1983 年，IEEE(电气和电子工程师协会)给出的软件工程定义为：“软件工程是开发、运行、维护和修复软件的系统方法”。

Fritz Bauer(美国著名的软件工程专家)则给出了另一个关于软件工程学的定义：“建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法”。

后来又有一些从事软件工程方法学研究的人陆续提出了许多更为完善的软件工程的定义，但主要思想都是强调软件开发过程中需要应用工程化原则的重要性。

IEEE 给出了关于软件工程的一个更加综合的定义：

(1) 将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护过程。即将工程化方法应用于软件开发与维护过程中。

(2) 对上述方法的研究。

就内容来看，软件工程应当包括三个要素：方法、工具和过程。

软件工程方法为软件开发提供了“如何做某项工作”的技术指南。它包括了多方面的任务。例如项目策划和估算方法、软件需求分析方法、体系结构的设计方法、详细设计方法、软件测试方法等等。使得整个开发过程的每一种阶段任务都能够“有章可循”。

软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境。目前这样的工具已经有许多种，而且已经有人把诸多软件工程工具集成起来，使得一种工具产生的信息可以为其他工具所使用，形成了一种称之为计算机辅助软件工程(CASE)的软件开发支撑环境。CASE 把各种软件工具、开发机器和一个存放开发过程信息的工程数据库组合起来，形成了一个完整的软件工程环境。

软件工程中的“过程”是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。可以将软件工程过程理解为软件工程的工艺路线。过程定义了各种方法使用的顺序、各阶段要求交付的文档资料、为保证质量和控制软件变更所需要的管理环节和在软件开发各个阶段完成的里程碑。

针对软件工程的基本要件，有许多计算机科学家进行了诠释，先后提出了 100 多条有关软件工程的相关原则。著名软件工程专家 B.W.Boehm 集众家所长，并总结了 TRW 公司多年开发软件的经验，在 1983 年提出了软件工程的七项基本原则，作为保证软件产品质量和开发效率的最小集合。具体包括：

(1) 用分阶段的生命周期计划严格管理软件工程过程。

(2) 坚持在软件工程过程中进行阶段评审。

(3) 实行严格的产品控制。