# 实用软件工程

第二版

郑人杰 殷人昆 陶永雷

清华大学出版社

# 实用软件工程

（第二版）

郑人杰　殷人昆　陶永雷

清华大学出版社

(京)新登字 158 号

## 内 容 简 介

本书是《实用软件工程》的第二版，本书第一版出版后，被许多学校和培训班作为软件工程课程的教材而受到欢迎。第二版在保持第一版的易懂和实用的风格外，增加了软件工程近期发展的新资料。本书从软件的开发、维护和软件管理等方面系统地阐述了软件工程的基本概念和常用的方法。各章节均结合实例讲解，使读者易于理解和掌握。书后附有软件产品开发文档编制指南，是软件开发人员必备的资料。

本书可作为大专院校的教材，计算机软件人员资格(或水平)考试的培训教材，也可供计算机软件开发人员和用户阅读。



**图书在版编目(CIP)数据**

实用软件工程/郑人杰等编著．-2 版．-北京：清华大学出版社，1997.4
ISBN 7-302-02520-7

Ⅰ．实… Ⅱ．郑… Ⅲ．软件工程 Ⅳ．TP311.5

中国版本图书馆 CIP 数据核字（97）第 4824 号

出 版 者：清华大学出版社（北京清华大学校内，邮编：100084）
责任编辑：徐培忠
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京科技发行所
开　　本：787×1092 1/16　印张：31.5　字数：651 千字
版　　次：1997 年 4 月第 2 版　1997 年 4 月第 1 次印刷
书　　号：ISBN 7-302-02520-7/TP・1277
印　　数：0001—5000
定　　价：29.50 元

# 再版前言

随着计算机的日益普及，计算机软件无处不在。以软件的说明、开发、维护和管理为内容，作为信息产业的一个支柱，软件工程这一学科已逐渐为人们所熟悉和广泛应用。这和前些年有很大的不同。现在大家都认识到，如果有哪个项目不遵循软件工程原则必定会受到实践的惩罚。一些清华大学计算机系的毕业生认为：软件工程课是他们参加工作以后最能直接应用的一门专业课。

本书能为数万读者普及软件工程知识起到微薄的作用。作者自然感到十分欣慰。从第一版开始发行至今已有十几年了，在此期间软件技术已有了迅速的发展。当第一版印刷多次又告罄而决定再版时，必须考虑技术更新的因素。在对一些章节做了修改和增加了新内容后，发现增加得过多，于是不得不再削减。不过全书毕竟是在第一版框架的基础上改写的，仍然尽可能坚持简明、实用的原则。希望第二版能够继续为培养我国软件专业人才发挥作用。

有关CASE等方面的内容，请参阅本人与彭春龙先生编著的《计算机辅助软件工程CASE技术》一书。

在第二版文稿的撰写过程中得到了美国Grand Valley State University陶永雷教授的热情关心和帮助，书中有关面向对象技术的许多内容就是他编写的，或根据他提供的资料完成的。在此深表谢意。必须说明的是，清华大学计算机系殷人昆副教授在繁忙的教学、实验室工作中，为文稿材料的组织、编写，甚至录入付出了大量的劳动，没有他的参与，第二版何时能与读者见面是难于估计的。

最后，还要提到我的同事蒋维杜、张素琴，北京太平洋技术软件有限公司刘建国，美国Prentice-Hall出版公司中国公司姜峰、武卫东等诸位先生，以及高峻等研究生，对他们的帮助表示感谢。

郑人杰

1996年10月

# 目 录

# 第 1 章　软件工程概述

在近代技术发展的历史上，工程学科的进步一直是产业发展的巨大动力。传统的工程学科走过的道路已为人们所熟知。水利工程、建筑工程、机械工程、电力工程等对工农业、商业、交通业的影响是极为明显的。人们在认识和征服自然的长征中继续前进，近年来人们开始对气象工程、生物工程、计算机工程等有了新的认识。然而，对工程学科家族的另一新成员——软件工程却不很熟悉。事实上，软件工程的地位非常重要，它对软件产业的形成和发展起着决定性的推动作用。在计算机的发展和应用中至关重要，在人类进入信息化社会时成为新兴信息产业的支柱。而人们对软件工程不了解，其根本原因是对软件本身认识不清。

本章将对软件的地位和作用、软件的特点、软件的发展和软件危机、软件工程学科的形成、软件生存期及软件工程过程等方面的问题和基本概念给出简要的介绍，以便使读者在进入软件工程专题以前，对总体框架获得一般性的理解。

## 1.1　软件的概念、特点和分类

### 1.1.1　软件的概念与特点

“软件”这一名词在 60 年代初从国外传来，当时许多人说不清它的确切含意。software 一词确是 soft 和 ware 两字组合而成。有人译为“软制品”，也有人译为“软体”。现在应该统一称它为软件。对于它的一种公认的解释为，软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序能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材料。尽管这个说法并不是计算机软件的精确定义，然而却有助于与扩充了含意的广义软件相区别。因为当前在产业界的经济活动中，相对于机器设备、车辆、原材料这样的有形实体以外，则可以把技术条件、管理法规以及人员素质这样的无形因素称为广义的软件。

为了能全面、正确地理解计算机和软件，必需了解软件的特点。

(1) 软件是一种逻辑实体，而不是具体的物理实体。因而它具有抽象性。这个特点使它和计算机硬件，或是其他工程对象有着明显的差别。人们可以把它记录在纸面上，保存在计算机的存储器内部，也可以保留在磁盘、磁带和可重写光盘上，但却无法看到软件的形态，而必须通过观察、分析、思考、判断，去了解它的功能、性能及其他特性。

(2) 软件的生产与硬件不同。在软件的开发过程中没有明显的制造过程。也不像硬件那样，一旦研制成功，可以重复制造，在制造过程中进行质量控制，以保证产品的质量。而软件是通过人们的智力活动，把知识与技术转化成信息的一种产品。一旦某一软件项目研制成功，以后就可以大量地复制同一内容的副本。所以对软件的质量控制，必须着重在软件开发方面下功夫。由于软件的复制是件非常容易的事情，因此出现了软件产品的保护问题。为了使软件开发的复杂劳动受到社会的承认和尊重，必须在技术上和法律上采取有力的措施，对于任意复制软

件的行为加以严格的限制。

虽然近年来国内外也都有建立“软件工厂”的说法，但软件工厂毕竟只是为软件开发创造更优越的环境和条件，提供更有效的手段，以利于高效地开发软件，它并不意味着要按硬件生产的模式生产软件。

(3) 在软件的运行和使用期间，没有硬件那样的机械磨损，老化问题。任何机械、电子设备在运行和使用中，其失效率大都遵循如图 1.1(a)所示的 U 型曲线。因为在刚刚投入使用时，各部件尚未做到配合良好、运转灵活，常常容易出现问题。经过一段时间运行，就可以稳定下来。而当设备经历了相当长时间的运转，就会出现磨损、老化等问题，使失效率越来越大。当失效率达到一定程度，就到达了寿命的终点。而软件的情况与此不同，它没有 U 型曲线的右半翼，因为它不存在磨损和老化问题。然而它存在退化问题。在软件的生存期中，为了使它能够克服以前没有发现的故障、使它能够适应硬件、软件环境的变化以及用户新的要求，必须要多次修改(维护)软件，而每次修改必不可免地引入新的错误，这样一次次修改，导致软件失效率升高，如图 1.1(b)所示，从而使得软件退化。究其原因，新的错误多在设计或程序编码阶段产生。因此，软件维护比硬件维护要复杂得多，与硬件的维修有着本质的差别。

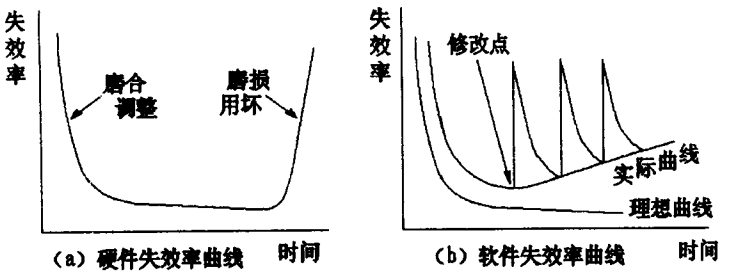


图 1.1　失效率曲线

(4) 软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。软件不能完全摆脱硬件单独活动。在开发和运行中必须以硬件提供的条件为依据。有的软件这种依赖性大些，常常为某个型号的计算机所专用，这对使用会带来许多不便。有的软件依赖于某个操作系统。为了解除这种依赖性，在软件开发中提出了软件移植的问题，并且把软件的可移植性作为衡量软件质量的因素之一。

(5) 软件的开发至今尚未完全摆脱手工艺的开发方式。软件产品大多是“定做”的，很少能做到利用现成的部件组装成所需的软件。近年来软件技术虽然取得了不少进展，提出了许多新的开发方法，例如充分利用现成软件的复用技术、自动生成技术，也研制了一些有效的软件开发工具或软件开发环境。但在软件项目中采用的比率仍然很低。由于传统的手工艺开发方式仍然占据统治地位，开发的效率自然受到很大的限制。对于软件人员来说，开发工作是一种高强度的脑力劳动，没有哪一个软件人员认为这是一项轻松的工作。

(6) 软件是复杂的。有人认为，人类能够创造的最复杂的产物是计算机软件。软件的复杂性可能来自它所反映的实际问题的复杂性，例如，它所反映的自然规律，或是人类社会的事务，都具有一定的复杂性；另一方面，也可能来自程序逻辑结构的复杂性，例如，一个系统软件要能处理各种可能出现的情况。软件开发，特别是应用软件的开发常常涉及到其他领域的专门知

识，这对软件人员提出了很高的要求。软件的复杂性与软件技术的发展不相适应的状况越来越明显。图 1.2 示出软件技术的发展落后于复杂的软件需求，并且随着时间的推移，这个差距日益加大。

(7) 软件成本相当昂贵。软件的研制工作须要投入大量的、复杂的、高强度的脑力劳动，它的成本是比较高的。问题不仅于此，值得注意的是硬件软件的成本 40 年来发生了戏剧性的变化。无论研制也好，向厂家购买也好，在 50 年代末，软件的开销大约占总开销的百分之十几，大部分成本要花在硬件上；但 80 年代这个比例完全颠倒过来，软件的开销大大超过硬件的开销，如图 1.3 所示。今天——90 年代的情况更是这样。美国每年投入软件开发的经费要有几百亿美元。然而，也并非在所有软件开发上的花费都能获得成果。

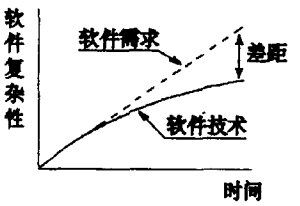


图 1.2 软件技术的发展落后于需求

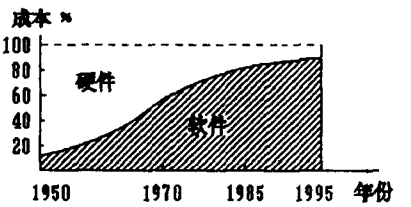


图 1.3 计算机系统硬、软件成本比例的变化

(8) 相当多的软件工作涉及到社会因素。类似于企业管理类型的软件自然是不言而喻的。许多软件的开发和运行涉及机构、体制及管理方式等问题，甚至涉及到人的观念和人们的心理。对于这些人的因素重视得不够，常常是软件工作遇到的问题之一。即使是对软件的看法不同也会有很大的影响。例如，由于主管部门对正在开发的软件不够理解，因而软件开发得不到应有的重视和必要的支持，造成人力和资金上的困难，它直接影响到项目的成败。

## 1.1.2 软件的分类

以上讨论的是区别于计算机硬件或其他工程对象的各种软件的共同特点。那么究竟软件有哪些类型？事实上，要给计算机软件做出科学的分类是很难的，但鉴于不同类型的工程对象，对其进行开发和维护有着不同的要求和处理方法，因此仍然需要对软件的类型进行必要的划分。既然找不到一个统一的严格分类标准，那么从不同角度来做出分类是比较符合实际情况的。

(1) 按软件的功能进行划分

· 系统软件：能与计算机硬件紧密配合在一起，使计算机系统各个部件、相关的软件和数据协调、高效地工作的软件。例如，操作系统、数据库管理系统、设备驱动程序以及通信处理程序等。系统软件的工作通常伴随着：频繁地与硬件交往、大量地为用户服务、资源的共享与复杂的进程管理，以及复杂数据结构的处理。系统软件是计算机系统必不可少的一个组成部分。

· 支撑软件：是协助用户开发软件的工具性软件，其中包括帮助程序人员开发软件产品的工具，也包括帮助管理人员控制开发的进程的工具。表 1.1 给出了一些支撑软件的实例。

表 1.1 支撑软件举例

| 一 般 类 型 | 支持需求分析 |
|---|---|
| 文本编辑程序 | PSL/PSA 问题描述语言、问题描述分析器 |
| 文件格式化程序 | 关系数据库系统 |
| 磁盘向磁带做数据传输程序 | 一致性检验程序 |
| 程序库系统 | CARA 计算机辅助需求分析器 |
| 支 持 设 计 | 支 持 实 现 |
| 图形软件包 | 编辑程序 |
| 结构化流程图绘图程序 | 交叉编辑程序 |
| 设计分析程序 | 预编译程序 |
| 程序结构图编辑程序 | 连接编辑程序 |
| 支 持 测 试 | 支 持 管 理 |
| 静态分析程序 | PERT 进度计划评审方法绘图程序 |
| 符号执行程序 | 标准检验程序 |
| 模拟程序 | 库管理程序 |
| 测试覆盖检验程序 | |

• 应用软件：是在特定领域内开发，为特定目的服务的一类软件。现在几乎所有的国民经济领域都使用了计算机，为这些计算机应用领域服务的应用软件种类繁多。其中商业数据处理软件是所占比例最大的一类，工程与科学计算软件大多属于数值计算问题。此外，应用软件在计算机辅助设计/制造(CAD/CAM)、系统仿真、智能产品嵌入软件(如汽车油耗控制、仪表盘数字显示、刹车系统)，以及人工智能软件(如专家系统、模式识别)等方面大显神通，使得传统的产业部门面目一新，带给我们的是惊人的生产效率和巨大的经济效益。而在事务管理、办公自动化方面的软件也在企事业机关迅速推广，中文信息处理、计算机辅助教学(CAI)等软件使得计算机向家庭普及，甚至连娃娃也能在计算机上学习和游戏。

(2) 按软件规模进行划分

按开发软件所需的人力、时间以及完成的源程序行数，可确定六种不同规模的软件，如表 1.2 所示。

表 1.2 软件规模的分类

| 类 别 | 参 加 人 员 数 | 研 制 期 限 | 产品规模(源程序行数) |
|---|---|---|---|
| 微型 | 1 | 1～4 周 | 0.5k |
| 小型 | 1 | 1～6 月 | 1～2k |
| 中型 | 2～5 | 1～2 年 | 5k～50k |
| 大型 | 5～20 | 2～3 年 | 50k～100k |
| 甚大型 | 100～1000 | 4～5 年 | 1M＝1000k |
| 极大型 | 2000～5000 | 5～10 年 | 1M～10M |

• 微型：只是一个人，甚至是半时，在几天之内完成的软件。写出的程序不到 5 百行语句，仅供个人专用。通常这种小题目无须做严格的分析，也不必要有一套完整的设计、测试资料。不过这并不是说可以随便地不讲任何方法地做。事实说明，即使这样小的题目，如果经过一定的

分析、系统设计、结构化编码以及有步骤地测试，肯定也是非常有益的。

• 小型：一个人半年之内完成的 2 千行以内的程序。例如，数值计算问题或是数据处理问题就是这种规模的课题。这种程序通常没有与其他程序的接口。但须要按一定的标准化技术、正规的资料书写以及定期的系统审查。只是没有大题目那样严格。

• 中型：5 个人以内在一年多时间里完成的 5 千到 5 万行的程序。这种课题开始出现了软件人员之间、软件人员与用户之间的联系、协调的配合关系问题。因而计划、资料书写以及技术审查需要比较严格地进行。这类软件课题比较普遍，许多应用程序和系统程序就是这样的规模。在开发中使用系统的软件工程方法是完全必要的，这对提高软件产品质量和程序人员的工作效率起着重要的作用。

• 大型：5 至 10 个人在两年多的时间里完成的 5 万到 10 万行的程序。例如编译程序、小型分时系统、应用软件包、实时控制系统等很可能是这种软件。参加工作的软件人员需要按二级管理，例如划分成若干小组，每组 5 人以下为好。在任务完成过程中，人员调整往往不可避免。因此会出现对新手的培训和逐步熟悉工作的问题。对于这样规模的软件，采用统一的标准，实行严格的审查是绝对必要的。由于软件的规模庞大以及问题的复杂性，往往会在开发的过程中出现一些事先难于做出估计的不测事件。

• 甚大型：100 至 1000 人参加用 4 到 5 年时间完成的具有 100 万行程序的软件项目。这种甚大型项目可能会划分成若干个子项目，每一个子项目都是一个大型软件。子项目之间具有复杂的接口。例如，实时处理系统、远程通信系统、多任务系统、大型操作系统、大型数据库管理系统、军事指挥系统通常有这样的规模。很显然，这类问题没有软件工程方法的支持，它的开发工作是不可想象的。

• 极大型：2000 人到 5000 人参加，10 年内完成的 1000 万行以内的程序。这类软件很少见，往往是军事指挥、弹道导弹防御系统。

从以上介绍可知，规模大、时间长、很多人参加的软件项目，其开发工作必须要有软件工程的知识做指导。而规模小、时间短、参加人员少的软件项目也得有软件工程概念，遵循一定的开发规范。其基本原则是一样的，只是对软件工程技术依赖的程度不同而已。

(3) 按软件工作方式划分

• 实时处理软件：指在事件或数据产生时，立即予以处理，并及时反馈信号，控制需要监测和控制的过程的软件。主要包括数据采集、分析、输出三部分，其处理时间是应严格限定的，如果在任何时间超出了这一限制，都将造成事故。

• 分时软件：允许多个联机用户同时使用计算机。系统把处理机时间轮流分配给各联机用户，使各用户都感到只是自己在使用计算机的软件。

• 交互式软件：能实现人机通信的软件。这类软件接收用户给出的信息，但在时间上没有严格的限定。这种工作方式给予用户很大的灵活性。近年来，终端设备更加普及，交互式软件到处可见。一个重要的问题日益显得突出，这就是用户界面设计。良好的用户界面设计将给用户带来极大的方便。

• 批处理软件：把一组输入作业或一批数据以成批处理的方式一次运行，按顺序逐个处理完的软件。这是最传统的工作方式。

(4) 按软件服务对象的范围划分

完成软件工程项目后可以有两种情况提供给用户：

· 项目软件：也称定制软件，是受某个特定客户（或少数客户）的委托，由一个或多个软件开发机构在合同的约束下开发出来的软件。例如军用防空指挥系统、卫星控制系统的软件就属于这一类。这类项目软件中有的软件带有试验研究性质，完成项目后根据需要可在此基础上做进一步开发。为取得客户的委托项目，软件开发机构的质量管理、技术实力、开发经验以及履行合同的信誉成为受到重视的问题。

· 产品软件：是由软件开发机构开发出来直接提供给市场，或是为千百个用户服务的软件。这是一些服务于多个目的及多个用户的软件。例如，文字处理软件、文本处理软件、财务处理软件、人事管理软件等。由于要参与市场竞争，其功能、使用性能以及培训和售后服务显得尤为重要。

（5）按使用的频度进行划分

有的软件开发出来仅供一次使用。例如用于人口普查、工业普查的软件。由于若干年才进行一次普查，前些年开发的软件在若干年以后很难适用。有的统计资料或试验数据须按年度做统计分析，相应的软件每年运行一次。另外有些问题，须要每天及时进行数据处理，如天气预报。这类软件具有较高的使用频度。显然，开发不同使用频度的软件，有不同的要求，不可一律看待。

（6）按软件失效的影响进行划分

工作在不同领域的软件，适应其不同的需求，在运行中对可靠性也有不同的要求。有的软件如果在工作中出现了故障，造成软件失效，可能给软件整个系统带来的影响不大。例如可能带来一些不便，却能勉强工作。但有的软件一旦失效。可能酿成灾难性后果，其严重损失难以挽回。例如控制载人飞行物的软件，如果不能正常工作，可能以人的生命为代价。

事实上，随着计算机进入国民经济和国防的各个重要领域，其软件的可靠性越来越显得重要。如财务金融、交通通信、航空航天等。人们一般称这类软件为关键软件，其特点在于：

1）可靠性等质量要求高。

2）常与完成重要功能的大系统的处理部件相联。

3）含有可能对以下各项造成影响的程序：

· 人员或公众的安全；

· 设备或设施的安全；

· 环境的质量；

· 国家的政务或部队的军务；

· 数据、通信或实体的机密。

## 1.2 软件的发展和软件危机

本世纪40年代中出现了世界上第一台计算机以后，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，计算机软件经历了三个发展阶段：

· 程序设计阶段，约为50至60年代

· 程序系统阶段，约为60至70年代

· 软件工程阶段，约为70年代以后

从表 1.3 中可以看到三个发展时期主要特征的对比。几十年来最根本的变化体现在：

**表 1.3　计算机软件发展的三个时期及其特点**

| 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、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 | 个人程序技术 | 小组技术水平 | 管理水平 |
| 开发技术和手段 | 子程序<br>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 |
| 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<br>存储容量小<br>工作可靠性差 | 降价、速度、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要，出现软件危机 | 开发技术有进步，但未获突破性进展，价高，未完全摆脱软件危机 |

*）表注：这里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。

### 1.2.1　人们对软件有了新的认识

从 50 年代到 60 年代，人们曾经把程序设计看作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，使用了许多窍门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。

### 1.2.2　软件的需求是软件发展的动力。

早期的程序开发者只是为了满足自己的需要，这种自给自足的生产方式仍然是其低级阶段的表现。进入软件工程阶段以后，软件开发的成果具有社会属性，它要在市场中流通以满足

广大用户的需要。软件开发者和用户的分工和责任也是十分清楚的。

### 1.2.3 软件工作的范围从只考虑程序的编写扩展到涉及整个软件生存期

关于软件生存期的概念将在下一节介绍。

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办法，致使问题积累起来，形成了日益尖锐的矛盾。这些问题归结起来有：

(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，为了加快进度赶上去而增加人力，结果适得其反，不仅未能加快，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

作为软件设计依据的需求，在开发的初期阶段提得不够明确，或是未能得到确切的表达。开发工作开始后，软件人员和用户又未能及时交换意见，使得一些问题不能及时解决而隐藏下来，造成开发后期矛盾的集中暴露。然而这时问题既难于分析，也难于挽回。

(3) 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，参加的人员各行其事。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；或是忽视了每个人工作与其他人的接口部分，发现了问题修修补补，这样的软件很难维护。

(4) 软件产品无评测手段

未能在测试阶段充分做好检测工作，提交用户的软件质量差，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，甚至造成生命财产的重大损失。

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 IBM 公司在 1963 年至 1966 年开发的 IBM 360 机的操作系统。这一项目花了 5000 人-年的工作量，最多时有 1000 人投入开发工作，写出了近 100 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 1000 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 F. D. Brooks 事后总结了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……"。IBM 360 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记取。

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

## 1.3 软件工程过程和软件生存期

从上述软件危机的现象和发生危机的原因可以看出，想摆脱危机不是一件很简单的事，不能只从一两个方面着手解决。真正得到满意的解决也并非一朝一夕能够做到的。但在这当中如何针对软件的特点，把它与其他产业部门工作对象的相同和相异之处加以分析，排除人们的一些传统观念和某些错误认识是非常重要的。只有端正了对软件的认识，真正抓住了它的特点和发展趋势，才能逃出危机，走上软件发展的正确道路。

开发一个软件，除去那些规模很小的项目以外，通常要由多个软件人员分工合作、共同完成；开发阶段之间的工作也应有很好的衔接；开发工作完成以后，软件成果要面向用户，在应用中接受用户的检验。所有这些活动都要求人们改变过去那种把软件当作个人才智产物的观点，抛弃那些只按自己工作习惯，不顾与周围其他人员配合关系的做法。在这一点上，软件开发与计算机硬件研制，甚至与高楼建设没有本质的差别。任何参加这些工程项目的人员，它们的才能只有在工程项目的总体要求和技术规范的约束下充分发挥和施展。许多计算机和软件科学家尝试，把其他工程领域中行之有效的工程学知识运用到软件开发工作中来。经过不断实践和总结，最后得出一个结论：按工程化的原则和方法组织软件开发工作是有效的，也是摆脱软件危机的一个主要出路。

### 1.3.1 软件工程过程（Software Engineering Process）

软件工程过程是为获得软件产品，在软件工具支持下由软件工程师完成的一系列软件工程活动。每个软件开发机构都可以规定自己的软件工程过程。针对不同类型的软件产品，同一软件开发机构也可能使用多个不同的软件工程过程。但无论是那种情况，软件工程过程通常包含四种基本的过程活动：

（1）软件规格说明：规定软件的功能及其运行的限制；

（2）软件开发：产生满足规格说明的软件；

（3）软件确认：确认软件能够完成客户提出的要求；

（4）软件演进：为满足客户的变更要求，软件必须在使用的过程中演进。

事实上，软件工程过程是一个软件开发机构针对某一类软件产品为自己规定的工作步骤，它应当是科学的、合理的，否则必将影响到软件产品的质量。因此，有理由要求软件工程过程具有如下的特性：

（1）易理解性。

（2）可见性：每个过程活动均能以取得明确的结果告终，使过程的进展对外可见。

（3）可支持性：易于得到计算机辅助软件工程（Computer Aided Software Engineering，CASE）工具的支持。

（4）可接受性：易于为软件工程师接受和使用。

（5）可靠性：不会出现过程错误，或发现在产品出现故障之前。

（6）健壮性：不受意外发生问题的干扰。

（7）可维护性：过程可随软件机构需求的变更或随认定的过程改进而演进。

（8）速度：从给出规格说明起，就能较快地完成开发而交付。