WUTP

普通高等学校计算机科学与技术专业新编系列教材

Software Engineering

软件工程

曾建潮　主编

```
#include <stdio.h>
#include <stdio.h>
void main()
void main()
{
{
   void swap(int *ptr1,int *ptr2);
   void swap(int *ptr1,int *ptr2);
  int x,y,*ptr1,*ptr2;
  int x,y,*ptr1,*ptr2;
  printf("input x,y:");scanf("%d,%d",&
  printf("input x,y:");scanf("%d,%d",&
  printf("%d\t%d\n",x,y);ptr1=&x;ptr2
  printf("%d\t%d\n",x,y);ptr1=&x;ptr2
  if(x<y)
  if(x<y)
  swap(ptr1,ptr2);
  swap(ptr1,ptr2);
  printf("%d\t%d\n",x,y);
  printf("%d\t%d\n",x,y);
}
}
void swap(int *ptr1,int *ptr2)
void swap(int *ptr1,int *ptr2)
```

武汉理工大学出版社
Wuhan University of Technology Press

普通高等学校计算机科学与技术专业新编系列教材

Software Engineering

软 件 工 程

曾建潮 主编
崔尚森 副主编

武汉理工大学出版社
Wuhan University of Technology Press

内 容 提 要

本书在吸取了国内外大量同类书刊精华的基础上，总结了编者多年来从事软件工程教学研究的经验和体会，全面系统地介绍了软件工程的概念、原理和典型的技术方法。本书的特点是讲解深入浅出，着重讲透基本的概念、原理、技术和方法；既注重系统性、科学性和先进性，又特别注重实用性；既有原理性论述，又有丰富、完整的实例与之配合，利于读者理解和掌握，特别是书后的附录，详细介绍了软件开发文档编写指南，是软件开发人员必备的资料。

本书正文共 12 章，第 1 章是概论，第 2 章讲述可行性分析，第 3 章讲述结构化分析方法，第 4 章讲述总体设计，第 5 章讲述详细设计，第 6 章讲述编码实现，第 7 章介绍了面向对象的方法学、面向对象的分析方法和建模技术，第 8 章介绍面向对象的设计和实现技术，第 9 章讲述软件测试技术，第 10 章讲述软件维护，第 11 章介绍软件工程项目管理技术，第 12 章介绍软件质量保证技术。

本书内容新颖、实例丰富，即可作为高等院校“软件工程”课程的教材或教学参考书，也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。

图书在版编目(CIP)数据

软件工程/曾建潮主编. —武汉：武汉理工大学出版社，2003.8
普通高等学校计算机科学与技术专业(本科)新编系列教材
ISBN 7-5629-1954-2

Ⅰ.软… Ⅱ.①曾… 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(2002)第 106863 号

出版发行：武汉理工大学出版社(武汉市武昌珞狮路 122 号 邮编：430070)
HTTP://cbs.whut.edu.cn
E-mail:wutp02@163.com wutp@public.wh.hb.cn
经 销 者：各地新华书店
印 刷 者：湖北省荆州市翔羚印刷实业公司
开 本：787×960 1/16
印 张：18.75
字 数：367 千字
版 次：2003 年 8 月第 1 版
印 次：2003 年 8 月第 1 次印刷
印 数：1—5000 册
定 价：25.00 元

普 通 高 等 学 校
计算机科学与技术专业新编系列教材
编 审 委 员 会

出版说明

当今世界已经跨入了信息时代，计算机科学与技术正在迅猛发展。尤其是以计算机为核心的信息技术正在改变整个社会的生产方式、生活方式和学习方式，推动整个人类社会进入信息化社会。为了顺应时代潮流，适应计算机专业调整及深化教学改革的要求，充分考虑到不同层次高校的教学现状，满足广大高校的教学需求，武汉理工大学出版社经过广泛调研，与国内近30所高等院校的计算机专家进行探讨，决定组织编写“普通高等学校计算机科学与技术专业新编系列教材”。

我们在组织编写新编本套系列教材时，**以培养现代化高级人才为重任，以提高学生综合素质、培养学生应用能力和创新能力为目的，以面向现代化、面向世界、面向未来为准绳，注重系列教材的特色和实用性，反映最新的教学与科研成果，体现本专业的时代特征。**同时，面对教育改革的需要、人才的需要和社会的需要，在编写本教材时，借鉴、学习国外一流大学的先进教学体系，结合国内的实际需要，吸取具有先进性、实用性和权威性的国外教材的精华，以更好地促进国内教材改革顺利进行。从时代和国际竞争要求的高度来思考，为打造一套高起点、高水平、高质量的系列教材而努力。

本套教材具有以下特色：

与时俱进，内容科学先进——充分体现计算机学科知识更新快的特点，及时更新知识，确保教材处于学科前沿，以拓宽学生知识面，培养学生的创新能力。

紧跟教学改革步伐，体现教学改革的阶段性成果——符合全国高校计算机专业教学指导委员会、中国计算机学会教育委员会制订的“计算机学科教学计划2000”的内容要求。

实现立体化出版，适应教育方式的变革——本套教材努力使用和推广现代化的教学手段，凡有条件的课程都准备组织编写、制作和出版配合教材使用的实验、习题、课件、电子教案及相应的程序设计素材库。

本套教材首批25种预定在2003年秋季全部出齐。我们的编审者、出版者决不敢稍有懈怠，一定高度重视，兢兢业业，按最高的质量标准工作。教材建设是我们共同的事业和追求，也是我们共同的责任和义务，我们诚恳地希望大家积极选用本套教材，并在使用过程中给我们多提意见和建议，以便我们不断修订、完善全套教材。

武汉理工大学出版社

2002年10月

前　言

随着计算机的日益普及，计算机软件应用领域越来越广泛，但是由于人们在软件开发过程中形成的一些错误概念和做法，严重影响了软件的开发，使得软件开发至今还受到“软件危机”的困扰。人们开发优质软件的能力大大落后于计算机硬件的发展水平和社会对计算机软件增长的需求，这种情况严重地阻碍计算机技术的发展。

为了摆脱软件危机的困扰，一门研究软件开发与维护的普遍原理和技术的工程学科——软件工程学，从20世纪60年代末期开始迅速发展起来了，现在它已经成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。严格遵循软件工程方法论可以大大提高软件开发的成功率，能够显著减少软件开发和维护中的问题。

软件工程学研究的范围非常广泛，包括技术方法、工具和管理等许多方面，软件工程又是一门不断发展的学科，新的技术方法和工具不断涌现，因此，在一本书中不可能包含软件工程的全部内容。本书作为软件工程课程的基础教材，强调了软件工程的基本概念，特别强调了软件工程的技术与方法，对基于数据流的结构化方法和面向对象方法作了比较详细的介绍。希望本书既能对实际的软件开发工作有所帮助，又能为读者在今后深入研究这门学科奠定良好的基础。

本书由曾建潮教授主编，共含12章，具体编写分工如下：第1章由曾建潮（太原重型机械学院）编写，第2、4、6章由王宏刚（太原重型机械学院）编写，第3、5章由刘冠蓉（武汉理工大学）编写，第7、8章由乔钢柱（太原重型机械学院）编写，第9、10章由谷岩（广州大学）编写，第11、12章及附录由崔尚森（长安大学）编写。

本书很适合于有一定实践经验的软件工作者和广大计算机用户参考或自学；对于高等院校计算机系高年级本科生和研究生来说，本书可以做为软件工程课程的教材。

感谢本书的责任编辑，他细致耐心的工作使本书的质量和进度得到了保证。

由于时间和水平的限制，难免出现疏忽和谬误之处，敬请读者指正。

编　者

2003年4月

目　录

1　软件工程概论 …… (1)

1.1　软件危机引发的思考 …… (1)

1.1.1　软件的发展与软件危机 …… (1)

1.1.2　产生软件危机的根源 …… (3)

1.1.3　软件产品的特征 …… (4)

1.1.4　软件产品的生产过程与软件生存期 …… (4)

1.1.5　解决软件危机的途径 …… (6)

1.2　软件工程的概念与原理 …… (6)

1.2.1　软件工程的概念 …… (6)

1.2.2　软件工程项目的基本目标 …… (7)

1.2.3　软件工程与传统工程的区别 …… (7)

1.2.4　软件工程的基本原理 …… (8)

1.2.5　软件质量评价 …… (10)

1.3　软件生命周期模型 …… (10)

1.3.1　瀑布模型 …… (10)

1.3.2　演化模型 …… (13)

1.3.3　螺旋模型 …… (14)

1.3.4　增量模型 …… (14)

1.3.5　喷泉模型 …… (18)

1.4　软件开发方法 …… (19)

1.4.1　结构化分析与设计方法 …… (19)

1.4.2　面向对象的分析与设计 …… (22)

1.4.3　软件工具与软件开发环境 …… (22)

习题与思考题 …… (24)

2　可行性分析 …… (25)

2.1　可行性分析的主要任务 …… (25)

2.1.1　分析和澄清问题定义 …… (25)

2.1.2 确定问题是否值得去解 …… (26)
2.2 可行性分析的步骤 …… (26)
2.2.1 复查系统的规模和目标 …… (26)
2.2.2 通过对现实环境的调查研究,获得更多的信息 …… (26)
2.2.3 确定新系统的高层逻辑模型 …… (27)
2.2.4 对新系统的逻辑模型进行验证并重新定义问题 …… (27)
2.2.5 导出可供选择的方案并进行评价 …… (27)
2.2.6 向决策人员提交行动建议 …… (28)
2.2.7 书写文档提交审查 …… (28)
2.3 可行性分析的技术方法 …… (28)
2.3.1 系统流程图 …… (28)
2.3.2 数据流图 …… (31)
2.3.3 数据字典 …… (34)
2.4 成本/效益分析 …… (36)
2.4.1 成本估计 …… (36)
2.4.2 成本/效益分析 …… (37)
习题与思考题 …… (38)

3 需求分析 …… (40)

3.1 需求分析的任务 …… (40)
3.1.1 确定目标系统的综合要求 …… (41)
3.1.2 分析目标系统的数据要求 …… (41)
3.1.3 导出目标系统的逻辑模型 …… (41)
3.1.4 修正软件项目开发计划 …… (42)
3.1.5 开发原型系统 …… (42)
3.1.6 编写软件需求规格说明书 …… (42)
3.2 需求分析的过程 …… (42)
3.3 数据驱动的分析方法 …… (44)
3.3.1 数据流图 …… (44)
3.3.2 数据字典 …… (47)
3.4 功能驱动的分析方法 …… (49)
3.4.1 状态迁移图 …… (49)
3.4.2 Petri 网 …… (51)
3.5 快速原型驱动的分析方法 …… (53)
3.6 数据与数据库需求 …… (54)

3.6.1 E-R模型 …… (54)
3.6.2 数据结构的规范化 …… (56)
3.7 需求验证 …… (57)
3.7.1 如何验证软件需求的正确性 …… (57)
3.7.2 软件需求验证的方法 …… (57)
习题与思考题 …… (58)

4 总体设计 …… (60)

4.1 总体设计的目标与任务 …… (60)
4.2 软件设计的基本原理 …… (62)
4.2.1 抽象化 …… (62)
4.2.2 模块化 …… (62)
4.2.3 信息隐蔽原理 …… (63)
4.2.4 模块独立性(模块的内聚性,模块间的耦合性) …… (63)
4.3 软件结构准则 …… (66)
4.3.1 软件结构图 …… (66)
4.3.2 软件结构设计的优化准则 …… (68)
4.4 软件设计的图形工具 …… (70)
4.4.1 IPO图 …… (70)
4.4.2 HIPO图 …… (71)
4.5 结构化设计方法 …… (72)
4.5.1 变换流分析 …… (72)
4.5.2 事务流分析 …… (74)
习题与思考题 …… (75)

5 详细设计 …… (77)

5.1 详细设计的目标和任务 …… (77)
5.2 程序的基本结构 …… (78)
5.3 详细设计工具 …… (79)
5.3.1 程序流程图 …… (79)
5.3.2 N-S图 …… (79)
5.3.3 PAD图 …… (81)
5.3.4 判定表和判定树 …… (82)
5.3.5 伪码 …… (84)
5.4 Jackson方法 …… (85)

5.4.1 Jackson 图 …… (86)
5.4.2 JSP 方法 …… (88)
5.4.3 JSD 方法 …… (89)
5.5 Warnier 方法 …… (89)
习题与思考题 …… (91)

6 编码实现 …… (93)

6.1 对源程序质量的要求 …… (93)
6.2 程序设计的风格 …… (94)
6.3 程序设计语言的选择 …… (96)
6.4 程序复杂性度量 …… (97)
6.4.1 代码行度量法 …… (97)
6.4.2 McCabe 度量法 …… (97)
6.4.3 综合度量 …… (99)
习题与思考题 …… (99)

7 面向对象的方法学 …… (102)

7.1 “面向对象”的概念 …… (102)
7.1.1 对象 …… (103)
7.1.2 类 …… (103)
7.1.3 属性、操作和方法 …… (104)
7.1.4 消息 …… (104)
7.1.5 封装、继承、多态和重载 …… (104)
7.2 面向对象的方法学简介 …… (105)
7.2.1 面向对象方法的基本活动 …… (106)
7.2.2 Coad & Yourdon 方法 …… (106)
7.2.3 OMT 方法 …… (107)
7.2.4 Booch 方法 …… (108)
7.2.5 UML 技术 …… (109)
7.3 面向对象的分析 …… (110)
7.3.1 面向对象分析概述 …… (110)
7.3.2 面向对象分析的原则 …… (110)
7.3.3 面向对象分析的基本过程 …… (111)
7.4 面向对象建模 …… (123)
7.4.1 面向对象建模概述 …… (123)

7.4.2 基于 UML 语言的模型类型 …… (123)
7.4.3 基于 UML 语言的建模过程 …… (124)
7.4.4 建模实例 …… (125)
习题与思考题 …… (131)

8 面向对象的设计与实现 …… (132)

8.1 面向对象的设计方法 …… (132)
8.1.1 面向对象的设计概述 …… (133)
8.1.2 面向对象设计的任务 …… (134)
8.1.3 面向对象设计的过程 …… (135)
8.2 基于 UML 的系统设计 …… (144)
8.2.1 基于 UML 的设计模型 …… (144)
8.2.2 设计模型的结构 …… (144)
8.2.3 基于 UML 的设计实例 …… (145)
8.3 面向对象的实现 …… (147)
8.3.1 面向对象实现概述 …… (147)
8.3.2 面向对象实现的基本准则 …… (147)
8.3.3 面向对象语言的选择 …… (148)
8.3.4 实现的工作流程 …… (148)
习题与思考题 …… (150)

9 软件测试 …… (151)

9.1 软件测试的基本概念 …… (151)
9.2 软件测试的目的和原则 …… (152)
9.2.1 软件测试的目的 …… (152)
9.2.2 软件测试的原则 …… (152)
9.2.3 测试与软件开发各阶段的关系 …… (154)
9.3 软件测试的方法 …… (154)
9.3.1 静态测试与动态测试 …… (155)
9.3.2 黑盒测试与白盒测试 …… (157)
9.4 白盒测试的测试用例设计 …… (158)
9.4.1 逻辑覆盖 …… (158)
9.4.2 基本路径覆盖 …… (163)
9.5 黑盒测试的测试用例设计 …… (167)
9.5.1 等价类划分 …… (167)

9.5.2 边界值分析 …… (170)
9.5.3 错误推测法 …… (172)
9.5.4 因果图 …… (172)
9.6 软件测试的策略(过程/步骤)…… (175)
9.6.1 单元测试 …… (176)
9.6.2 组装集成测试 …… (178)
9.6.3 确认测试 …… (182)
9.6.4 系统测试 …… (183)
9.7 测试终止标准 …… (183)
9.8 调试 …… (184)
9.8.1 调试的目的 …… (184)
9.8.2 调试的技术策略 …… (184)
习题与思考题 …… (188)

10 软件维护 …… (189)

10.1 维护的概念与内容 …… (189)
10.1.1 软件维护的定义 …… (189)
10.1.2 软件维护的内容 …… (190)
10.2 软件维护的过程 …… (192)
10.2.1 维护机构与维护申请报告 …… (192)
10.2.2 软件维护工作流程 …… (193)
10.2.3 维护档案记录 …… (195)
10.2.4 维护活动评价 …… (195)
10.3 软件的可维护性 …… (196)
10.3.1 软件可维护性的定义与度量 …… (196)
10.3.2 提高可维护性的方法 …… (201)
习题与思考题 …… (202)

11 软件工程项目管理 …… (204)

11.1 软件项目管理概述 …… (204)
11.1.1 软件项目失控原因分析 …… (205)
11.1.2 软件项目管理的特点 …… (205)
11.1.3 软件项目管理的内容 …… (206)
11.2 软件规模度量 …… (208)
11.2.1 软件度量的分类 …… (208)

11.2.2 代码行度量法 …… (210)
11.2.3 功能点度量法 …… (211)
11.2.4 特征点度量法 …… (212)
11.2.5 代码行与功能点度量的比较 …… (213)
11.3 软件开发成本估算 …… (214)
11.3.1 估算方法 …… (214)
11.3.2 分解与类推 …… (215)
11.3.3 基于代码行和功能点的估算 …… (217)
11.3.4 经验估算模型 …… (218)
11.4 软件项目资源管理 …… (221)
11.4.1 人力资源 …… (221)
11.4.2 硬件资源计划 …… (223)
11.4.3 软件资源 …… (224)
11.4.4 软件复用性及软件部件库 …… (225)
11.5 进度计划 …… (226)
11.5.1 进度安排 …… (226)
11.5.2 甘特图 …… (226)
11.5.3 工程网络 …… (227)
11.5.4 软件开发任务的并行性 …… (228)
11.6 风险管理 …… (229)
11.6.1 风险识别 …… (230)
11.6.2 风险估计 …… (230)
11.6.3 风险评价 …… (231)
11.6.4 风险驾驭和监控 …… (232)
11.7 软件工程标准化和软件文档标准化 …… (234)
11.7.1 软件工程标准化的定义 …… (234)
11.7.2 软件工程标准化的层次 …… (235)
11.7.3 文档的作用、分类与编制要求 …… (236)
习题与思考题 …… (239)

12 软件质量保证 …… (240)

12.1 软件质量模型 …… (240)
12.1.1 软件质量的定义 …… (240)
12.1.2 软件质量特性 …… (241)
12.1.3 McCall 软件质量模型 …… (243)

12.2 软件质量度量 …… (245)
12.2.1 软件质量度量概论 …… (245)
12.2.2 软件正确性度量 …… (246)
12.2.3 软件可靠性度量 …… (247)
12.2.4 易使用性和可维护性度量 …… (250)
12.3 软件质量保证 …… (251)
12.3.1 软件质量保证的概念 …… (251)
12.3.2 软件质量保证的任务 …… (252)
12.3.3 提高软件质量的技术途径 …… (252)
12.3.4 软件质量保证体系 …… (253)
12.4 质量检验和评审 …… (254)
12.4.1 各阶段质量检验的项目 …… (254)
12.4.2 软件质量检验方法 …… (256)
12.4.3 软件质量评审 …… (258)
12.5 软件能力成熟度模型(CMM) …… (261)
12.5.1 不成熟的与成熟的软件机构的对比 …… (261)
12.5.2 软件机构能力成熟度模型 …… (262)
12.5.3 成熟度级别的内部结构 …… (265)
12.5.4 关键过程领域 …… (266)
12.5.5 成熟度提问单 …… (266)
12.5.6 利用 CMM 对软件机构进行成熟度评估 …… (268)
习题与思考题 …… (269)

附录 软件开发文档编写指南 …… (271)

A 可行性研究报告 …… (271)
B 项目开发计划 …… (274)
C 需求规格说明书 …… (275)
D 概要设计说明书 …… (277)
E 详细设计说明书 …… (279)
F 测试计划 …… (280)
G 测试分析报告 …… (281)
H 开发进度月报 …… (282)

参考文献 …… (284)

1 软件工程概论

本章提要

随着软件项目的日益复杂，出现了“软件危机”。软件工程则是从技术和组织管理两个方面解决“软件危机”，提高软件开发效率与质量的一门新兴学科。本章从软件的发展与软件危机所引发的思考入手，重点讲述软件工程的基本概念和基本原理，并对常用的软件生命周期模型和软件开发方法与工具进行简要的介绍。

1.1 软件危机引发的思考

1.1.1 软件的发展与软件危机

目前，人们普遍接受的软件的定义是：软件是计算机系统中与硬件相互依存的另一部分，它包括程序、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各种图文资料。

软件对于人类而言是一个全新的东西，其发展历史仅有四五十年。人们对软件的认识也经历了一个由浅到深的过程。软件的发展大致可分为三个时期：即软件＝程序时期(1947 年至 20 世纪 60 年代初)、软件＝程序＋说明时期(20 世纪 50 年代末至 70 年代初)、软件＝程序＋文档时期(20 世纪 70 年代初至今，也称软件工程时期)。

在计算机系统发展的初期，硬件通常用来执行一个单一的程序，而这个程序又是为了一个特定目的而编制的。在通用的硬件已经非常普遍的时候，软件的

通用性却是很有限的。大多数软件是由使用该软件的个人或机构开发的，软件往往带有强烈的个人色彩。早期的软件开发也没有什么系统的方法可以遵循，软件设计是在某个人的头脑中完成的一个隐藏的过程，而且，除了源代码往往没有软件说明书等文档。

计算机系统发展的第二个阶段跨越了从 20 世纪 50 年代末到 70 年代初的十余年，这个时期软件技术取得了很大的进展，比较有代表性的是多道程序设计技术、多用户人机交互系统、实时系统及文件管理等，同时，出现了更多的通用和专用程序设计语言，在形式化语言理论、编译理论、数据库理论等方面均取得了重大突破，从而诞生了真正的计算机科学。在这一时期软件开始作为一种产品被广泛使用，并出现了“软件作坊”——专职应别人的需求写软件。但软件开发的方法基本上仍然沿用早期的个体化软件开发方式，而软件的数量却急剧膨胀，软件的需求日益复杂，维护的难度越来越大，开发成本也越来越高，同时，失败的软件开发项目也屡见不鲜，因而出现了“软件危机”。

“软件危机”的出现使得人们开始对软件及其特性、软件开发方法等方面进行更进一步的研究，人们逐渐在改变对软件的不正确看法。早期那些被认为是优秀的程序往往很难被别人看懂，通常充满了程序技巧。现在人们普遍认为优秀的软件除了功能正确、性能优良之外，还应该是容易看懂、容易使用、容易修改和扩充的。

为了摆脱“软件危机”的困境，软件研究人员一方面研究程序设计的方法和程序正确性验证的方法；另一方面研究如何使用工程化的方法进行软件系统开发。

以“结构化程序设计方法”的提出为标志，引发了人们对程序设计方法的广泛关注和研究，先后提出了模块化、结构化、由顶向下逐步求精、程序变换、程序的推理与综合、数据类型抽象、符号测试、程序正确性证明等各种程序设计和验证的方法，这一时期的研究成果使软件研究与设计人员深刻地认识到，没有科学的方法作指导，不可能产生高质量的软件产品，同时提出了以正确性、结构清晰、便于设计、便于测试和维护等作为衡量软件质量优劣的重要标准。

程序设计方法和程序正确性验证方法的研究，使程序设计更加科学化、规范化，并在一定程度上提高了软件的可靠性。但由于设计过程仍采用手工方式，周期长、成本高，很难适应软件生产的需求。对此，北大西洋公约组织的计算机科学家于 1968 年、1969 年连续召开了软件研究会议（即 NATO 会议）。集中研究对策，讨论如何摆脱“软件危机”，提出了“软件工程”的概念，试图用“工程化”的思想作指导来解决软件设计中所面临的困难，从而解决软件危机的困境。

1.1.2 产生软件危机的根源

如前所述，软件危机实际上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困难。“软件危机”(Software Crisis)一词最早是1968年北大西洋公约组织在联邦德国召开的软件国际会议上提出的。

概括来说，软件危机包含两方面问题：一是如何开发软件，以满足不断增长、日趋复杂的需求；二是如何维护数量不断膨胀的软件产品。具体地说，软件危机主要有以下表现：

(1)对软件开发成本和进度的估计常常不准确，开发成本超出预算，实际进度比预定计划一再拖延的现象并不罕见；

(2)用户对“已完成”系统不满意的现象经常发生；

(3)软件产品的质量往往靠不住；

(4)软件的可维护程度非常低；

(5)软件通常没有适当的文档资料；

(6)软件的成本不断提高；

(7)软件开发生产率的提高赶不上硬件的发展和人们需求的增长。

产生软件危机的原因，一方面是与软件本身的特点有关；另一方面是与软件开发和维护的方法不正确有关。

首先，软件不同于硬件，它属于计算机系统中的逻辑部件而不是物理部件。在编制软件并在计算机上试运行之前，软件开发的进展情况很难衡量，软件开发的质量也很难评价，因此，管理和控制软件开发过程相当困难。其次，软件不会因为使用时间过长而被“用坏”，如果在运行中出现错误，往往是由于在开发时期的测试阶段没有能检测出该错误的原因。因此，软件维护通常意味着改正或修改原来的设计，这就在客观上造成了软件较难维护。另外，软件不同于一般程序，其显著特点是规模庞大。这就需要多人共同合作进行软件系统开发，而如何保证每个人完成的工作合在一起确实能构成一个高质量的大型软件系统，则是一个极其复杂困难的问题，这不仅涉及到诸如分析方法、设计方法、形式说明方法、版本控制等许多技术问题，还需要有严格而科学的管理。最后，由于目前相当多的软件开发人员对软件开发和维护还存在不少糊涂观念，在软件开发过程中或多或少地采用了错误的方法和技术，这也是软件问题发展成为软件危机的主要原因。

与软件开发和维护有关的许多错误认识和做法的形成，归因于在计算机系统发展的早期软件开发的个性化特点。软件开发与维护的不正确方法主要表现为忽视软件开发前期的需求分析，开发过程没有统一的、规范的方法论的指导；文档资料不齐全，忽视人与人的交流；忽视测试阶段的工作，提交给用户的软件