软件工程

Software Engineering

软件工程介绍

软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。 它涉及程序设计语言、数据库、软件开发工具、系统平台、标准、设计模式等方面。 在现代社会中，软件应用于多个方面。 典型的软件有电子邮件、嵌入式系统、人机界面、办公套件、操作系统、编译器、数据库、游戏等。

软件工程学科

(083500) (085405)

序号	学校名称	评选结果
1	北京航空航天大学（985 211）	A+
2	浙江大学（985 211）	A+
3	国防科技大学（军 985 211）	A+
4	北京大学（985 211）	A
5	清华大学（985 211）	A
6	华东师范大学（985 211）	A
7	南京大学（985 211）	A
8	武汉大学（985 211）	A
9	天津大学（985 211）	A-
10	东北大学（985 211）	A-
11	哈尔滨工业大学（985 211）	A-
12	同济大学（985 211）	A-
13	上海交通大学（985 211）	A-
14	苏州大学（211）	A-
15	中国科学技术大学（985 211）	A-
16	四川大学（985 211）	A-

- 北航 软件学院-991-软件工程基础综合
- 《软件工程基础综合》考试内容包括数据结构与算法、**软件工程**和操作系统三部分，试卷满分150分，各部分占比均三分之一。

考试大纲
一、软件工程概述
二、软件工程过程
三、软件需求分析
四、软件设计
五、软件构造与测试
六、软件项目管理基础

引论：

1. 我们已经学习过：计算机组成原理、数据结构、操作系统、程序设计等课程。
2. 是否具备完成一般软件目标能力呢？例如：⑴编写计算y=sin(x)的源码；⑵手机芯片操作系统(COS)设计；⑶大学课表编制系统。等等。
3. 要达到完成上述目标，我们仅有计算机的基础技术是不行的。

4. 一般情况下，一个软件的完成，除了要掌握计算机的基础技术外，还要有数学、物理、电子技术等基础科学与领域知识。
5. 有了上述技术还不够。还要有软件开发的专门技术。例如，我们要编写问题2中的程序时：
   ① 我们发现要知道他的设计是怎样的；
   ② 当设计该问题时，我们发现对该问题要进行分析。

③ 我们求解该问题的程序对吗？

④ 如果软件开发周期比较长，如何应付开发对象的变化。

⑤ 参加的人员多的时候。对参加人员的分工、布局

⑥ 软件质量怎么管理？

6. 一个软件的开发用什么工具。

7. 开发中如何与用户交流与合作，与同伴合作。

8. 每一个软件技术人员都希望有软件项目开发。那么是不是有投资就可以开发呢？这就是所谓的可行性问题。
9. 如何计算一个软件工程的费用与开发周期。
10. 工程中个人荣誉与利益。

……

只有学习《软件工程》，才能回答上述问题。

平常作业

- ACM Transactions on Software Engineering and Methodology
- http://dblp.uni-trier.de/db/journals/tosem/
- IEEE Transactions on Software Engineering
- http://dblp.uni-trier.de/db/journals/tse/

International Conference on Software Engineering

- https://dblp.uni-trier.de/db/conf/icse/icse2022.html

Oral ppt　在场所有同学一起打分

poster　所有同学共同打分

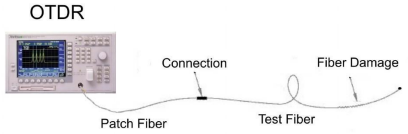
Fiber optic characterization using a simulated Optical Time-Domain Reflectometer (OTDR)

Robb P. Merrill

Department of Electrical and Computer Engineering - University of Utah

Introduction

Optical Time Domain Reflectometry (OTDR) is a common technique for detecting damage in fiber optic cables. The process involves transmitting a pulse of light down the optical fiber, analyzing the amount of light reflected back to the source, and displaying the reflection patterns on the OTDR screen.



Abnormalities in the fiber, such as bends, cracks, connectors, and other abrupt changes in the refractive index create reflection spikes called Fresnel ("Fre'-nel") reflections [2]. After a spike is detected, a significant delay occurs when the reflectometer 'settles down' from its saturated state. This delay is called a Fresnel tail (Figure 1).

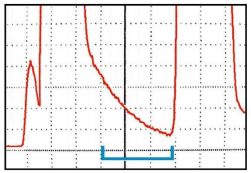


Figure 1: OTDR screenshot showing reflection spike from cable connector, and resulting Fresnel tail (area marked by bracket)

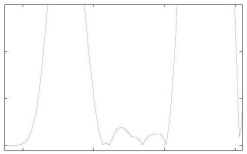


Figure 2: Simulated ideal response showing fiber damage (small reflection bumps). Damage is visible because no Fresnel tail is present.

During characterization of short fiber optic cables of approximately 1 meter, Fresnel reflections pose a serious challenge to accurate damage detection. The Fresnel tail obliterates any small reflections that are produced by damaged sections of cable, and the damage is overlooked.

Simulation Method

The Finite Difference Time Domain method [1] was implemented in MATLAB to simulate a pulse of light traveling through the patch and test fibers. The following parameters used in the simulation were obtained from an actual OTDR system: *Index of refraction (n) of test fiber* = 1.4525, *Wavelength (λ) of light pulse* = 850 nanometers [3] .

Pulse Duration

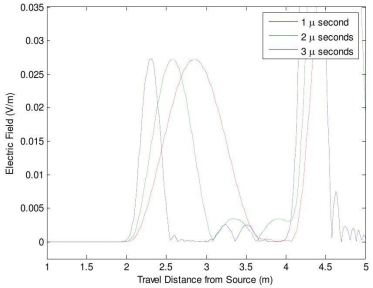


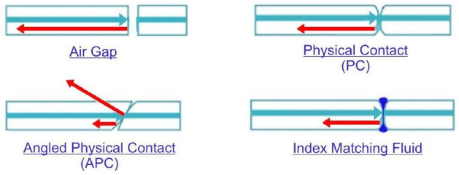
Figure 3: Simulated Fresnel Tail skews, then obliterates, the damage reflection at larger durations

To determine the effect of the light pulse duration on the saturation level of the OTDR unit, one period of a raised cosine pulse was transmitted through the fiber at various frequencies. A pulse duration of 1 microsecond proved to be the most favorably responsive for the parameters of the simulation (see Figure 3). In real-world application, however, the duration must actually be smaller due to the relatively slow simulation speed vs. the physical speed of light.

Connector Type

The index of refraction of the patch vs. the test fiber was allowed differ by up to 10%, which created a mismatch at the junction of the two fibers. Four types of connectors were simulated to determine which produced the lowest reflection magnitude.

Figure 4: Common types of fiber optic connectors with relative reflection magnitudes shown



Plotting the reflection response patterns from all four connection types shows that the Angled Physical Contact connector produced the lowest reflection (see Figure 6). Though much less expensive, Index Matching Fluid only has a lifetime of 2 years. Most optical fiber applications require 10 years life or more [3].

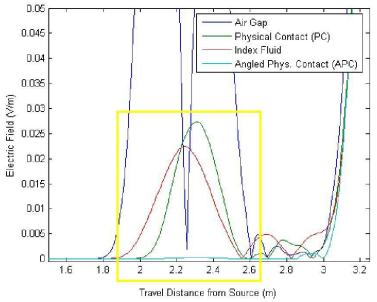


Figure 5: Reflection patterns using various connectors (reduced Fresnel magnitudes inside yellow box)

Summary

Short fiber optic cables present many challenges that must be overcome in order to accurately detect fiber damage using OTDR. Pulse durations shorter than 1 microsecond, and Angled Physical Contact (APC) fiber connectors are recommended to provide the greatest reduction in Fresnel reflection. By performing OTDR simulations, an optical systems engineer could understand the behavior of a fiber network and detect potential problems before actual production.

References

[1] Sadiku, N.O. Matthew. Numerical Techniques in Electromagnetics
[2] Newton, Steven A. Novel Approaches to Optical Reflectometry
[3] Knapp, John. Characterization of Fiber-Optic Cables Using an Optical Time Domain Reflectometer (OTDR)

论文题目	idea	方法策略-怎么做	其他对比方法	相关工作	Applications	实验平台	指标效果	前提假设	缺点启示	行文逻辑	code source
Imbalanced Deep Learning by Minority Class Incremental Rectification	Class imbalanced deep learning, Multi-label learning	Class Rectification Loss, by batch-wise incremental minority class rectification with a scalable hard mining principle.) CifarNet [34], (2 ResNet32 [36], and (3) DenseNet [82].	Over-Sampling, Down-Sampling, Cost-Sensitive,Threshold-Adjustment,**LMLE**	imbalance：data,algorithm,hybird	CelebA [12], X-Domain [11], and CIFAR-100	NVIDIA Tesla K40 GPU and 20 E5-2680 @ 2.70GHz CPUs	mean Class-balanced accuracy ,0.8042		看能否进一步提升		

读论文的目的：了解相关工作进展

- 论文的背景与目标
- 目标想解决什么问题-这个问题是怎么来的？
- 当前解决此问题的主要技术路线：相关工作及相关工作之间的关系
- 论文观察到了什么，或者发现了什么
- 如何利用这个发现，遇到了什么困难，设计了什么方法/策略，
- 效果如何？对比的指标是什么？对比是否公平？是否全面？
- 论文的前提假设是什么？是否有局限性？
- 有什么缺点

2022年信息与计算机学院研究生国家奖学金评审细则

4.科研成果的加分标准为（60分）：
科研成果主要对研究生在读期间的论文、专利、研究进展、实践成果等进行评价。
（一）论文
（1）论文被SCI（中科院一区，限定为电子信息类分区）期刊录用或发表，一篇60分（前三名比例为60%：30%：10%）；
（2）论文被SCI（JCR一区，限定为电子信息类分区）期刊录用或发表，一篇30分；
（3）论文被SCI（JCR二区，限定为电子信息类分区）期刊录用或发表、CCF A类会议录用或发表，一篇25分；
（4）论文被SCI（JCR其他分区）期刊录用或发表、CCF B类会议录用或发表，一篇20分；
（5）论文被EI期刊录用或发表，一篇15分；
（6）论文被CSCD核心版录用或发表，一篇10分；
（7）论文被CSCD扩展版录用或发表，一篇4分。

https://computer.ahau.edu.cn/info/1082/6564.htm

阶梯选题

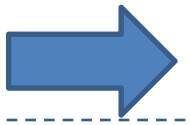
Introdution
Methods
Results

IEEE software in 2022
软件学报
专业领域期刊与软件工程相关

读书报告

Sure title before Oct. 18
Final submit time before Nov. 3

课程主页
http://81.70.163.81/software



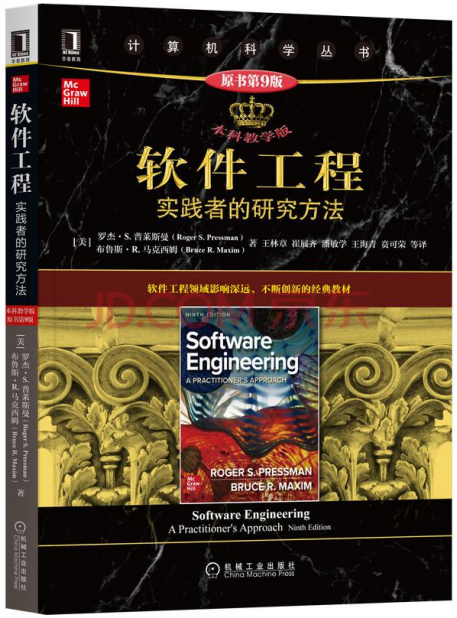
Poster 标准

评分部分	评分要素	评分标准	得分
内容	主体突出，内容完整；作品内容能够清晰的表达论文的精要。	很好50-40	
		好40-30	
		一般30	
论文技术理解	对论文中的文字、图片、表格、图表、代码等理解并能够有效的展示。	很好20-15	
		好15-10	
		一般10	
美观	整体界面美观布局合理，层次分明， 视觉效果好，表现力和感染力强。字体设计恰当，文字清晰，风格引人入胜。	很好15-10	
		好 10-7	
		一般7	
格式	立意新颖，构思独特，设计巧妙，具有想象力和去现力，具有较大的创新，原创成分高，具有鲜明的个性。	很好15-10	
		好10-7	
		一般7	

ppt 标准

评分部分	评分要素	评分标准	得分
内容	主体突出，内容完整；作品内容能够清晰的表达论文的精要。结构合理，逻辑顺畅，幻灯片之间具有层次性和连贯性、顺畅。	很好50-40	
		好40-30	
		一般30	
论文技术理解	对论文中的文字、图片、表格、图表、代码等理解并能够有效的展示。	很好20-15	
		好15-10	
		一般10	
美观	PPT背景图片、音乐、视频等各种运用符合当选主题。层次分明， 视觉效果好，表现力和感染力强。字体设计恰当，文字清晰，风格引人入胜。	很好15-10	
		好 10-7	
		一般7	
演讲技巧	音量合适，发音标准。语速合适，表达清晰，逻辑完整。	很好15-10	
		好10-7	
		一般7	


"十二五"普通高等教育本科国家级规划教材
全国优秀畅销书
全国高校出版社优秀畅销书
21世纪软件工程专业规划教材
软件工程导论（第6版）
清华大学出版社


计 算 机 科 学 丛 书
原书第9版
本科教学版
软件工程
实践者的研究方法
[美] 罗杰·S. 普莱斯曼（Roger S. Pressman） 著 王林章 祖延齐 潘敏学 王海青 贲可荣 等译
布鲁斯·R. 马克西姆（Bruce R. Maxim）
软件工程领域影响深远、不断创新的经典教材
Software Engineering
A PRACTITIONER'S APPROACH
ROGER S. PRESSMAN
BRUCE R. MAXIM
Software Engineering
A Practitioner's Approach Ninth Edition
机械工业出版社
China Machine Press


软件工程
第5版
清华大学出版社

绪 论

软件工程是指导计算机软件开发和维护的一种工程科学，它涉及的知识相当广泛。在学习软件工程之前，必须对软件工程领域的一些基本概念有所了解，对软件工程有一个初步的认识，这样才能顺利地进入后面章节的学习。

1.1 软件概述

计算机系统由硬件和软件两大部分组成。电子计算机自1946年诞生后到20世纪60年代中期是计算机发展的早期。

在早期，计算机系统还是以硬件为主，软件费用是总费用的20%左右。

到了中期（20世纪60年代中期到20世纪80年代初期），软件费用迅速上升到总费用的60%，软件不再只是技巧性和高度专业化的神秘机器代码。

1985年以后到今天，软件费用已上升80%以上。软件相对硬件的费用比例在不断提高。

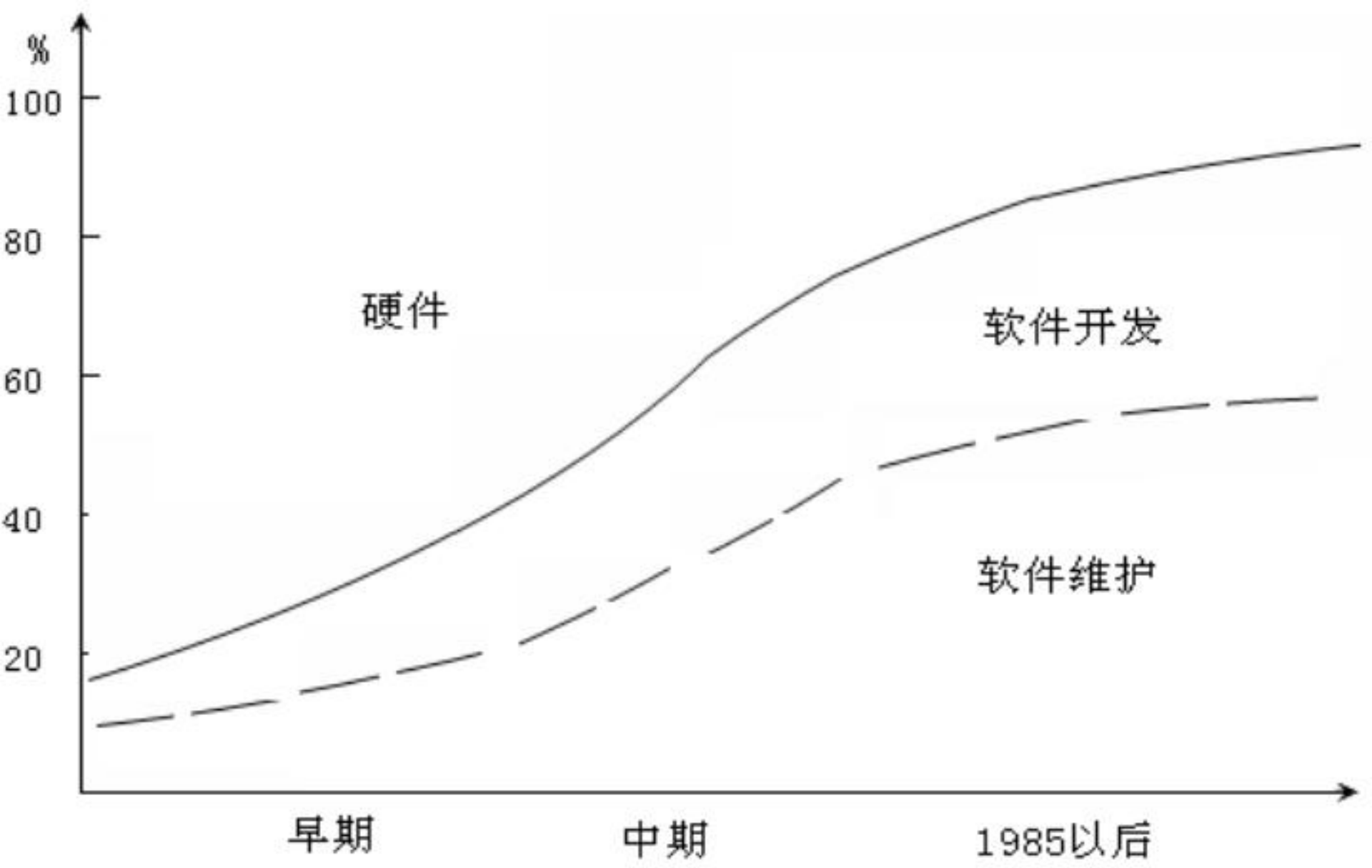


图1-1　硬件和软件费用比例变化示意图

1.1.1 什么是计算机软件

计算机软件的定义如下：

[定义1-1]：计算机运行所需要的各种程序和数据的总称，包括操作系统、汇编程序、编译程序、数据库、文字编辑及维护使用手册等。软件是计算机系统的重要组成部分。

软件是当代计算机行业中的重要产品，但它不是一种有形的物质，它表示的仅仅是一种思想，必须以某种形式表达，通常存储在磁盘（带）介质上或者以文本方式提供，也可以存储在ROM中。

1.1.2 软件的特点

软件是对客观世界中问题空间与解空间的具体描述，是客观事物的一种反映，是知识的提炼和“固化”。客观世界是不断变化的，因此，构造性和演化性是软件的本质特征。如何使软件模型具有更强的表达能力、更符合人类的思维模式，即如何提升计算环境的抽象层次，在一定意义上来讲，这紧紧围绕了软件的本质特征——构造性和演化性。

计算机系统中的软件与硬件是相互依存的，缺一不可。而软件与其他产品的特点不同。它是一种特殊的产品，具有下列特殊性质：

（1）软件产品的生产主要是脑力劳动，还未完全摆脱手工开发方式，大部分产品是“定做”的。

（2）软件是一种逻辑产品，它与物质产品有很大的区别，它是脑力劳动的结晶。软件产品是看不见摸不着的，因而具有无形性。它以程序和文档的形式出现，保存在存储介质上，通过计算机的运行才能体现它的功能和作用。

（3）软件产品不会用坏，不存在磨损、消耗问题。

（4）软件产品的生产主要是研制。其成本主要体现在软件的开发和研制上，软件开发研制完成后，通过复制就产生了大量软件产品。

（5）软件费用不断增加，软件成本相当昂贵。软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它的成本非常高。

1.1.3 软件的分类

20世纪计算机产生以来，人们围绕着它开发了大量的软件，广泛应用于科学研究、教育、工农业生产、事务处理、国防和家庭等众多领域，积累了丰富的软件资源。然而，在软件的品种质量和价格方面仍然满足不了人们日益增长的需要。计算机软件产业是一项年轻的、充满活力的飞速发展的产业。因此，关于其分类方法不同，所分类型差别也很大。

这里简单地介绍计算机软件在计算机系统、实时系统、嵌入式系统、科学和工程计算、事务处理、人工智能、个人计算机和计算机辅助软件工程（CASE）等方面的应用。

按照计算机的控制层次，计算机软件分为系统软件和应用软件两大类。

1. 系统软件
2. 应用软件

系统软件

计算机系统软件是计算机管理自身资源(如CPU、内存空间、外存、外部设备等)，提高计算机的使用效率并为计算机用户提供各种服务的基础软件。系统软件依赖于机器的指令系统、中断系统以及运算、控制、存储部件和外部设备。

系统软件包括操作系统、网络软件、各种语言的编译程序、数据库管理系统、文件编辑系统、系统检查与诊断软件等。

（1）操作系统。

（2）语言处理程序。

（3）数据库管理系统。

（4）实用程序与软件工具。

（1）操作系统。DOS是基于字符界面的单用户单任务的操作系统；Windows：是基于图形界面的单用户多任务的操作系统；UNIX：是一个通用的交互式的分时操作系统，用于各种计算机；NetWare：是基于文件服务和目录服务的网络操作系统；Windows NT：是基于图形界面32位多任务、对等的网络操作系统。

（2）语言处理程序。机器语言：计算机能直接执行的、由一串“0”或“1”所组成的二进制程序或指令代码，是一种低级语言。
汇编语言：一种用符号表示的、面向机器的低级程序设计语言，需经汇编程序翻译成机器语言程序才能被计算机执行。
高级语言：按照一定的“语法规则”、由表达各种意义的“词”和“数学公式”组成的、易被人们理解的程序设计语言，需经编译程序翻译成目标程序（机器语言）才能被计算机执行。如：FORTRAN、C、BASIC等。

（3）数据库管理系统。普及式关系型：FoxPro、Paradox、Access；大型关系型：Oracle、Sybase、SQL Server。

（4）实用程序与软件工具

应用软件是计算机所应用程序的总称，主要用于解决一些实际的应用问题。

按业务、行业，应用软件也可分为：

（1）个人计算机软件。

（2）科学和工程计算软件。

（3）实时软件（FIX、INTouch、Lookout）。

（4）人工智能软件。

（5）嵌入式软件。

（6）事务处理软件。

（7）工具软件。

（1）个人计算机软件。个人计算机上使用的软件也可包括系统软件和应用软件两类。近20年来，个人计算机的处理能力已提高了三个数量级，以前在中小型计算机上运行的系统软件和应用软件，如今已经大量移植到个人计算机上。

（2）科学和工程计算软件。它们以数值算法为基础，对数值量进行处理的计算，主要用于科学和工程计算，例如数值天气预报、弹道计算、石油勘探、地震数据处理计算机系统仿真和计算机辅助设计（CAD）等。
近年来有的也用C语言或Ada语言描述。它是使用最早、最广泛、最为成熟的一类软件。

（3）实时软件（FIX、INTouch、Lookout）。监视、分析和控制现实世界发生的事件，能以足够快的速度对输入信息进行处理并在规定的时间内做出反应的软件，称之为实时软件。

实时软件依赖于处理机系统的物理特性，如计算速度和精度、I/O信息处理与中断响应方式、数据传输效率等。支持实时软件的操作系统称为实时操作系统。实时软件使用的计算机语言有汇编语言、Ada语言等。实时系统的服务经常是连续的，系统在规定的时间内必须处于能够响应的状态，因此，实时软件和计算机系统必须有很高的可靠性和安全性。

（4）人工智能软件。支持计算机系统产生人类某些智能的软件。它们求解复杂问题时，不是采用传统的计算或分析方法，而是采用诸如基于规则的演绎推理技术和算法，在很多场合还需要知识库的支持。

人工智能在专家系统、模式识别、自然语言理解、人工神经网络、程序验证、自动程序设计、机器人学等领域开发了许多人工智能应用软件，用于诊断疾病、产品检测、自动定理证明、图像和语音的自动识别、语言翻译等。

（5）嵌入式软件。嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分控制该系统的运行，进而实现一个特定的物理过程。用于嵌入式计算机系统的软件称为嵌入式软件。

大型的嵌入式计算机系统软件可用于航空航天系统、指挥控制系统和武器系统等。小型的嵌入式计算机系统软件可用于工业的智能化产品之中，这时嵌入式软件驻留在只读存储器内，为该产品提供各种控制功能和仪表的数字或图形显示功能等。

（6）事务处理软件。用于处理事务信息，特别是商务信息的计算机软件。事务信息处理是软件最大的应用领域，它已由初期零散、小规模的软件系统，如工资管理系统、人事档案管理系统等，发展成为管理信息系统（MIS），如世界范围内的飞机订票系统、旅馆管理系统、作战指挥系统等等。

（7）工具软件。计算机辅助软件工程是指软件开发和管理人员在软件工具的帮助下进行软件产品的开发、维护以及开发过程的管理。

1.1.4 软件的发展

自第一台计算机诞生以来，软件的生产就开始了。随着计算机技术的飞快发展和应用领域迅速拓宽，自20世纪60年代中期以后，软件需求迅速增长，软件数量急剧膨胀。这种增长导致了软件的发展，可以将软件生产的发展划分为三个时代。

1. 程序设计时代（1946-1956年）
2. 程序系统时代（1956-1968年）
3. 软件工程时代（1968年至今）

1.　程序设计时代（1946-1956年）

在这一时期，软件的生产主要是个体手工劳动的生产方式。程序设计者使用机器语言、汇编语言作为工具；开发程序的方法上主要是追求编程技巧和程序运行效率。在程序设计中还没有注意其他辅助作用。因此所设计的程序难读、难懂、难修改。这个时期软件特征是只有程序、程序设计概念，不重视程序设计方法。

2.　程序系统时代（1956-1968年）

由于计算机的应用领域不断扩大，软件的需求也不断增长，软件由于处理的问题域扩大而使程序变得复杂，设计者不得不由个体手工劳动组成小集团合作，形成作坊式生产方式小集团合作生产的程序系统时代。生产工具是高级语言。开发方法仍旧靠个人技巧。由于大的程序需要合作，在程序设计中开始提出结构化方法。

3.　软件工程时代（1968年至今）

1968年在联邦德国召开的国际会议上讨论软件危机的问题。在这次会议上正式提出并使用了“软件工程”术语，新的工程科学就此诞生。

软件工程时代的生产方式是采用工程的概念、原理、技术和方法。

软件特征使开发技术有很大进步，但是未能获得突破性进展，软件价格不断上升，没有完全摆脱软件危机。

1.1.5 软件危机

所谓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。这种“严重问题”不仅仅是“不能正常运行”。实际上几乎所有的软件都不同程度地存在问题。

软件危机的表现

（1）对于软件开发的成本和进度的估计很不准确。由于缺乏软件开发的经验和软件开发数据的积累，使得开发工作的计划很难制定。主观盲目制定的计划，执行起来和实际情况有很大差距，使得开发经费一再突破。由于对工作量和开发难度估计不足，进度计划无法按时完成，开发时间一再拖延。

（2）开发的软件产品不能完全满足用户要求，用户对已完成的软件系统不满意的现象常常发生。

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，导致开发的软件产品不能完全满足用户要求。有些软件还因为合作与技术问题而导致失败。

（3）开发的软件可靠性差。由于在开发过程中，没有确保软件质量的体系和措施，在软件测试时，又没有严格的、充分的、完全的测试，提交给用户的软件质量差，在运行中暴露出大量的问题。这种不可靠的软件，轻则会影响系统正常工作，重则会发生事故，造成生命财产的重大损失。

（4）软件通常没有适当的文档。开发过程无完整、规范的文档，发现问题后进行杂乱无章的修改。程序结构不好，运行时发现错误也很难修改，导致可维护性差。

（5）软件的可维护性差。由于开发过程没有统一的、公认的规范，软件开发人员按各自的风格工作，各行其是。很多程序中的错误非常难改，实际上不可能使这些程序适应新的硬件环境，也不可能根据用户要求在程序中增加新功能。

（6）软件开发生产率提高的速度，远远跟不上计算机应用普及深入的趋势。软件产品“供不应求”的现象使人类不能充分利用计算机硬件资源提供的巨大潜力。

软件危机的产生

软件发展第二阶段的末期，由于计算机硬件技术的进步，计算机运行速度、容量和可靠性有显著的提高，生产成本有显著下降，为计算机的广泛应用创造了条件。一些复杂的、大型的软件开发项目提了出来。但是，软件开发技术一直未能满足发展的要求。软件开发中遇到的问题因找不到解决的办法，使问题积累起来，形成了尖锐的矛盾，导致了软件危机。

软件危机的原因

在软件的开发和维护过程中存在着这么多的问题，一方面与软件本身的特点有关，另一方面也与软件的开发和维护的方法有关。造成上述软件危机的原因概括起来有以下几方面：

（1）软件的规模愈发庞大。1968年美国航空公司订票系统达到30万条指令；IBM360OS第16版达到100万条指令；1973年美国阿波罗计划达到1000万条指令。这些庞大软件的功能非常复杂，体现在处理功能的多样性和运行环境的多样性。随着计算机应用的日益广泛，需要开发的软件规模日益庞大，软件结构也日益复杂。

（3）软件本身的独有特点确实给开发和维护造成一些客观困难。但是人们在长期的实践中也积累了不少成功的经验。如果坚持使用成功的经验和正确的方法，许多困难是可以克服的。但是相当多的软件开发人员对于软件的开发和维护存在不少糊涂的观念，实践中或多或少地采用错误的方法和技术。这可能是软件危机的主要原因。

（4）软件开发和维护中许多错误认识和方法的形成可以归结与计算机发展早期软件开发的个体化特点。其主要表现在对软件需求分析的重要性认识不够，错误地认为软件开发就是写程序并使之运行，不重视软件需求分析与维护等工作。

（5）软件开发技术落后。在20世纪60年代，人们注重一些计算机理论问题的研究，如编译原理、操作系统原理、数据库原理、人工智能原理、形式语言理论等，不注重软件开发技术的研究，用户要求的软件复杂性与软件技术解决复杂性的能力不相适应，它们之间的差距越来越大。

（6）生产方式落后。软件仍然采用个体手工方式开发，根据个人习惯爱好工作，无章可循，无规范可依据，靠言传身教方式工作。

（7）开发工具落后，生产率提高缓慢。软件开发工具过于原始，没有出现高效率的开发工具，因而软件生产率低下。在1960~1980年期间，计算机硬件的生产由于采用计算机辅助设计、自动生产线等先进工具，使硬件生产率提高了100万倍，而软件生产率只提高了2倍，相差十分悬殊。

1.2 软件工程

软件工程是指导计算机软件开发和维护的工程科学。为了克服软件危机，人们从其他产业的工程化生产得到启示，采用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验而证明正确的管理技术与方法技术结合起来，这就是软件工程。

1. 软件工程的定义

软件工程是用工程、科学和数学的原则与方法研制、维护计算机软件的有关技术及管理法。因此，软件工程的定义是：

[定义1-2]：将系统的、规范的、可度量的工程化方法应用于软件开发、运行和维护的全过程及上述方法的研究。

该定义说明了软件工程是计算机科学中的一个分支，其主要思想是在软件生产中用工程化的方法代替传统手工方法。

工程化的方法借用了传统的工程设计原理的基本思想，采用了若干科学的、现代化的方法技术来开发软件。软件工程由方法、工具和过程三部分组成。

2.　软件工程的性质

软件工程是一门综合性的交叉学科，它涉及哲学、计算机科学、工程科学、管理科学、数学和应用领域知识。计算机科学中的研究成果均可用于软件工程，但计算机科学着重于原理和理论，而软件工程着重于如何建造一个软件系统。

软件工程要用工程科学中的观点来进行费用估算、制定进度、制定计划和方案；要用管理科学中的方法和原理进行软件生产的管理；要用数学的方法建立软件开发中各种模型和各种算法，如可靠性模型，说明用户需求的形式化模型等。

1.2.1 软件工程与方法学

程序设计方法学和软件工程方法学是为了解决软件危机问题而逐渐形成的学科。

1971年Wirth的“自顶而下逐步求精”等对软件工程和程序设计方法学的形成和初期的发展有着深刻的影响。

1969年IFIP（国际信息处理协会）成立了“程序设计方法学工作组”—WG2.3，云集了当时许多著名的计算机科学家，专门研究程序设计方法学，这个国际组织对以后的程序设计方法学的发展起了很大的促进作用。

“软件工程”（Software Engineering）作为一个术语，是在1968年北大西洋公约组织的一次计算机学术会议上，正式提出来的。这个会议专门讨论了软件危机问题。这次会议是软件发展史上一个重要的里程碑。

但是，软件工程学和程序设计方法学研究的途径和侧重点有所差异，主要差异如下。

（1）研究方法和途径不同。

（2）研究对象有所侧重。

（3）软件工程学注重“宏观可用性”；程序设计方法学注重“微观正确性”。

（1）研究方法和途径不同。软件工程学应用的是工程方法；而程序设计方法学依据的是数学方法。软件工程学注重工程方法与工具研究；程序设计方法学则注重算法与逻辑方法研究。

（2）研究对象有所侧重。软件工程的对象所指的软件，一般是指“大型程序”，是一个系统；而程序设计方法学的研究对象则侧重于一些较小的具体程序模块，早期的程序设计方法学研究重点是某个单独的程序的时空效率、正确性证明等问题。

（3）软件工程学注重“宏观可用性”；程序设计方法学注重“微观正确性”。例如，软件工程学研究软件的“可靠性”的方法是“软件测试”；程序设计方法学研究的方法则是程序的“正确性证明”。

随着软件技术的迅速发展，软件工程学和程序设计方法学的研究内容也都在不断发展，研究的内容和方法相互渗透。事实上，人们已经很少、也没有必要区分什么是软件工程学的范畴，什么是程序设计方法学的范畴了。逐渐地，这两条研究途径的界限又模糊化、一体化了。

软件工程方法学既强调软件（一般指大型软件）开发的工程特征，又强调软件设计方法论的科学性、先进性。其基本内容包括以下几个方面。

（1）结构化理论与方法。

（2）模块技术与数据抽象。

（3）软件测试与程序正确性证明。

（4）软件分析与设计方法、工具及环境。

（5）软件工程管理与质量评价。

1.2.2 软件工程的基本原理

1983年B.Weohm提出了软件工程的七条基本原理。

这七条基本原理是保证软件产品质量和开发效率的最小集合，又是相当完备的。现在虽然不能用数学方法严格证明是一个完备的集合，但是可以证明在此以前的100多条软件工程原理都可以由这七条原理组合与派生。这七条原理如下。

1.　用分阶段的生命周期计划严格管理

统计表明，50%以上的失败项目是由于计划不周而造成的。在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。

在整个软件生命周期中应指定并严格执行六类计划：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、运行维护计划。

2. 坚持进行阶段评审

统计结果显示：大部分错误是在编码之前造成的，大约占63%；错误发现得越晚，改正它要付出的代价就越大，要差2到3个数量级。

因此，软件的质量保证工作不能等到编码结束之后再进行，应坚持进行严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

在软件开发的过程中不应随意改变需求，因为改变一项需求需要付出较高的代价。开发过程中麻烦的事情之一就是改动需求。但是实践告诉我们，需求的改动往往是不可避免的。由于各种客观的需要，不能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适应这种要求。也就是要采用变动控制，又叫基准配置管理。当需求变动时，其他各个阶段的文档或代码随之相应变动，以保证软件的一致性。

4. 采纳现代程序设计技术

从提出软件工程的概念开始，人们主要的精力都用于研究各种新的程序设计技术，20世纪60年代的结构化软件开发技术，随后又发展的结构化分析和结构化设计技术，已成为大多数人认为的先进程序设计技术。后来又提出的面向对象技术，从第一、第二代语言，到第四代语言，人们已经充分认识到：方法大于气力。采用先进的技术即可以提高软件开发的效率，又可以减少软件维护的成本。

5. 结果应能清楚地审查

软件产品不同于一般的物理产品，软件是一种看不见、摸不着的逻辑产品。软件开发小组的工作进展情况可见性差，难于评价和管理。为更好地进行管理，应根据软件开发的总目标及完成期限，尽量明确地规定开发小组的责任和产品标准，从而使所得到的标准能清楚地审查。

6.　开发小组的人员应少而精

开发人员的素质和数量是影响软件质量和开发效率的重要因素，应该少而精。这一条基于两点原因：高素质开发人员的效率比低素质开发人员的效率要高几倍到几十倍，开发工作中犯的错误也要少得多。

当开发小组为N人时，可能的通讯信道为N(N-1)/2，可见随着人数N的增大，通讯开销将急剧增大。

7. 承认不断改进软件工程实践的必要性

遵从上述前六条基本原理，就能够较好地实现软件的工程化生产。但是，它们只是对现有经验的总结和归纳，并不能保证赶上技术不断前进发展的步伐。因此，Weohm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原理。根据这条原理，不仅要积极采纳新的软件开发技术，还要注意不断总结经验，收集进度和消耗等数据，进行出错类型和问题报告统计。

1.2.3 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，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性并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，减少维护的困难。下面分别介绍这些概念。

（1）可修改性（modifiability）。
容许对系统进行修改而不增加原系统的复杂性。它支持软件的调试与维护，是一个难以度量和难以达到的目标。

（2）有效性（efficiency）。

软件系统能最有效地利用计算机的时间资源和空间资源。各种计算机软件无不将系统的时/空开销作为衡量软件质量的一项重要技术指标。很多场合，在追求时间有效性和空间有效性方面会发生矛盾，这时不得不牺牲时间效率换取空间有效性或牺牲空间效率换取时间有效性。时/空折中是经常出现的。有经验的软件设计人员会巧妙地利用折中概念，在具体的物理环境中实现用户的需求和自己的设计。

（3）可靠性（reliability）。

能够防止因概念、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。对于实时嵌入式计算机系统，可靠性是一个非常重要的目标。

（4）可理解性（understandabimy）。
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软件系统的复杂性，并支持软件的维护、移植或重用。

（5）可维护性（maintainability）。
软件产品交付用户使用后，能够对它进行修改，以便改正潜伏的错误，改进性能和其他属性，使软件产品适应环境的变化等等。由于软件是逻辑产品，只要用户需要，它可以无限期地使用下去，因此软件维护是不可避免的。

（6）可重用性（reusebility）。

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。软部件可以在多种场合应用的程度称为部件的可重用性。可重用的软部件有的可以不加修改直接使用，有的需要修改以后再用。可重用软部件应具有清晰的结构和注解，应具有正确的编码和较低的时/空开销。

（7）可适应性（adaptability）。
软件在不同的系统约束条件下，使用户需求得到满足的难易程度。适应性强的软件应采用广为流行的程序设计语言编码，在广为流行的操作系统环境中运行，采用标准的术语和格式书写文档。适应性强的软件较容易推广使用。

（8）可移植性（portability）。

软件从一个计算机系统或环境搬到另一个计算机系统或环境的难易程度。为了获得比较高的可移植性，在软件设计过程中通常采用通用的程序设计语言和运行支撑环境。可移植性支持软件的可重用性和可适应性。

（9）可追踪性（tracebility）。
根据软件需求对软件设计、程序进行正向追踪，或根据程序、软件设计对软件需求进行逆向追踪的能力。软件可追踪性依赖于软件开发各个阶段产档和程序的完整性、一致性和可理解性。降低系统的复杂性会提高软件的可追踪性。

（10）可互操作性（interoperability）。多个软件元素相互通信并协同完成任务的能力。为了实现可互操作性，软件开发通常要遵循某种标准，支持这种标准的环境将为软件元素之间的可互操作提供便利。可互操作性在分算环境下尤为重要。

1.2.4 软件工程的内容

软件工程研究的主要内容是指软件开发技术和软件开发管理两个方面。在软件开发技术中，它主要研究软件开发方法、软件开发过程、软件开发工具和环境。在软件开发管理中，它主要研究软件管理学、软件经济学和软件心理学等。

1.2.5 软件工程原则

为了达到软件系统开发目标，在软件开发过程中必须遵循下列软件工程原则：抽象、信息隐藏、模块化、局部化、一致性、完整性和可验证性。

1. 抽象（abstraction）

抽取事物最基本的特性和行为，忽略非基本的细节。采用分层次抽象的办法可以控制软件开发过程的复杂性，有利于软件的可理解性和开发过程的管理。

2. 模块化（modularity）

模块（module）是程序中逻辑上相对独立的成分，它是一个独立的编程单位，应有良好的接口定义。例如，FORTRAN语言中的函数、子程序，Ada语言中的程序包、子程序、任务等等。模块化有助于信息隐藏和抽象，有助于表示复杂的软件系统。

3. 信息隐藏（information hiding）

将模块中的软件设计决策封装起来的技术。模块接口应尽量简洁，不要罗列可有可无的内部操作和对象。按照信息隐藏的原则，系统中的模块应设计成“黑箱”，模块外部只能使用模块接口说明中给出的信息，如操作、数据类型等等。由于对象或操作的实现细节被隐藏，软件开发人员便能够将注意力集中于更高层次的抽象上。

4. 局部化（localization）

要求在一个物理模块内集中逻辑上相互关联的计算资源。从物理和逻辑两个方面保证系统中模块之间具有松散的耦合关系，而在模块内部有较强的内聚性。这样有助于控制解的复杂性。

5. 完整性（completeness）

软件系统不丢失任何重要成分，完全实现系统所需功能的程度；在形式化开发方法中，按照给出的公理系统，描述系统行为的充分性；当系统处于出错或非预期状态时，系统行为保持正常的能力。

6. 一致性（consistency）

整个软件系统（包括文档和程序）的各个模块均使用一致的概念、符号和术语；程序内部接口应保持一致；软件与硬件接口应保持一致；系统规格说明与系统行为应保持一致；用于形式化规格说明的公理系统应保持一致等等。一致性原则支持系统的正确性和可靠性。

7. 可验证性（verifiability）

开发大型软件系统需要对系统逐步分解。系统分解应该遵循系统容易检查、测试、评审的原则，以便保证系统的正确性。采用形式化的开发方法或具有强类型机制的程序设计语言及其软件管理工具可以帮助人们建立一个可验证的软件系统。

1.2.6 软件工程面临的问题

软件工程有许多需要解决的棘手问题，如软件费用、软件可靠性、软件可维护性、软件生产率和软件重用等。

1. 软件费用

由于软件生产基本上仍处于手工状态，软件是知识高度密集的技术的综合产物，人力资源远远不能适应这种迅速增长的社会要求，所以软件费用上升的势头必然还将继续下去。

2. 软件可靠性

软件可靠性是指软件系统能否在特定的环境条件下运行并实现所期望的结果。在软件开发中，通常要花费40%的代价进行测试和排错，即使这样还不能保证以后不再发生错误，为了提高软件可靠性，就要付出足够的代价。

3.　软件可维护性

统计数据表明，软件的维护费用占整个软件系统费用的2/3，而软件开发费用只占1/3。软件维护之所以有如此大的花费，是因为已经运行的软件还需排除隐含的错误，新增加的功能要加入进去，维护工作又是非常困难的，效率又是非常低下的。因此，如何提高软件的可维护性，减少软件维护的工作量，也是软件工程面临的主要问题之一。

4.　软件生产率

计算机的广泛应用使得软件的需求量大幅度上升，而软件的生产又处于手工开发的状态，软件生产率低下，使得各国都感到软件开发人员不足。这种趋势将仍旧继续下去。所以，如何提高软件生产率，是软件工程又一重要问题。

5. 软件重用

提高软件的重用性，对于提高软件生产率、降低软件成本有着重要意义。当前的软件开发存在着大量的、重复的劳动，耗费了不少人力资源。

软件的重用有各种级别，软件规格说明、软件模块、软件代码、软件文档等都可以是软件重用的单位。软件重用是软件工程中的一个重要研究课题，软件重用的理论和技术至今尚未彻底解决。

小结

本章主要介绍了软件工程的一些基本概念与基础知识。在介绍软件发展的过程中，介绍了其发展所经历的程序设计时代、程序系统时代和软件工程时代。还介绍了软件危机的表现、软件危机的产生和软件危机的原因以及软件工程的产生、软件工程的定义、软件工程的性质和基本原理。

谢谢