中创软件丛书

An Introduction to Technology of Software Engineering

软件工程技术概论

朱三元　钱乐秋　宿为民　编著

科学出版社
Science Press

中创软件丛书

软件工程技术概论

朱三元　钱乐秋　宿为民 编著

科学出版社

2002

内 容 简 介

本书全面系统地阐述软件工程所涉及到的各种新技术.

本书共分九章.第一章概述20世纪70年代到80年代软件工程的基本概念和方法.第二、三章概要介绍面向对象的分析和设计技术,并着重介绍统一建模语言UML的技术,包括用例建模、类和对象建模、动态建模和物理体系结构建模等.第四、五章主要介绍软件过程中的基本技术,包括过程建模、过程度量和过程改进以及近期的轻载方法.第六至八章概要介绍了软件复用技术、构件生产技术、构件组装技术、构件接口技术等.第九章主要介绍软件评审、生存周期软件开发V模型、软件测试自动化技术以及配置管理.每章末均给出相关的参考文献.附录中给出了软件工程职业道德规范和实践要求(5.2中文版),可供读者参考.

本书可作为高等学校计算机专业及相关专业高年级学生和研究生的教材或教学参考书,也可供科研机构和软件企业的技术人员参考阅读.

图书在版编目(CIP)数据

软件工程技术概论/朱三元等编者.—北京:科学出版社,2002
(中创软件丛书)
ISBN 7-03-009940-0

Ⅰ.软… Ⅱ.朱… Ⅲ.软件工程-新技术-概论 Ⅳ.TP311.5

中国版本图书馆CIP数据核字(2001)第086703号

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com
新蕾印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2002年1月第 一 版 开本:720×1000 1/16
2002年1月第一次印刷 印张:19 1/2
印数:1—4 000 字数:370 000
定价:29.00元
(如有印装质量问题,我社负责调换(环伟))

中创软件丛书编辑委员会

中创软件丛书序言

软件乃程序及其文档之总称.细言之,有个体含义、总体含义、学科含义之别.软件与硬件犹如阴与阳,天与地,二者互为依存,缺一不可,同为计算机系统之基本成分.就其学科含义而论,二者之发展构成计算机科学技术之发展.

软件之作用有三:一为用户与硬件之接口界面;二为计算机系统之指挥机构;三为计算机体系设计之重要依据.

软件一词出现于20世纪60年代初.至60年代中期,其规模与复杂度日益增大,软件开发难以控制,出现"软件危机".1968年北大西洋公约一次学术会议上提出"软件工程",视软件开发为工程,以工程方法开发软件,研究软件开发范型、模型与方法.80年代以来,结构构件化、开发自动化、表示形式化、接口自然化等逐渐成为研究热点.研究软件开发环境与工具,探讨软件体系结构,企图从总体上解决软件问题.然而,鉴于软件开发之复杂度,迄今工作虽多,鲜有突破.

40年来,我国学者在系统软件、支撑软件、应用软件诸方面均有建树.惜乎反映研究成果之著作,问世者甚鲜.山东中创软件工程股份有限公司高瞻远瞩,深感软件人才与教育之重要,高质量著作为二者所必需.因此,设立了"中创软件丛书基金",奖励杰出软件人才,出版"中创软件丛书",并成立"中创软件丛书编委会",负责选题、审议、编纂等事.此举蒙国家科委、中国计算机学会软件专业委员会、各高等院校与科研机构以及广大软件工作者之大力支持.愿本丛书遵"质量第一"之原则,自1997年起每年均有一至二部著作问世,冀收促进发展、嘉惠来学之效.

中创软件丛书编委会

序

肇始于1968年之软件工程乃软件领域之中流砥柱.简言之,软件工程之含义有二:一为工程含义,即软件开发、维护、管理等活动之总体;二为学科含义,即工程含义下软件工程所涉及之理论、原理、方法、技术所构成之学科.

朱君三元、钱君乐秋二教授潜心研究软件工程多年,造诣颇深,偕其高足宿为民博士合著《软件工程技术概论》(下称《概论》)一书,览读再四,数善存焉.

一、源流清晰,脉络分明

任何理论或技术之发展均有其源流,软件工程亦复如此.《概论》一书上溯企图克服"软件危机"、提高软件质量与生产率之源,下及数十年来逐步发展之各项工程,俾读者能窥其发展之来龙去脉,谙其发展之源流,全书组织以"工程"为纲,可谓源流清晰,脉络分明.

二、突出本质,取材新颖

《概论》一书由传统工程、对象工程、过程工程、构件工程、产品工程组成,突出"工程"之本质,纲举目张.此诸工程并非彼此排斥,实为不同视面之反映,互为补充,形成统一整体.内容源于国际新近之代表著作及作者多年之研究心得,可谓突出本质,取材新颖.

三、理例并举,注重实用

本书名为《概论》,意即论其梗概,不涉繁文缛节.其内容陈述,理论联系实际,既有原理、方法、技术,又有实例.俾读者便于理解,学为所用,为自行开发软件工程项目奠定基础,可谓理例并举,注重实用.

该书问世,定能嘉惠来学,促进"软件工程"课程之改进,促进软件产业、软件科学技术之发展.余有幸先睹书稿,得益良多,故乐而为之序.

徐家福

辛巳仲夏于金陵

不阿斋

前　言

每当我们回顾20世纪的技术进展，都会一致认同信息技术是发展最快的技术之一，特别是信息技术应用的渗透性，几乎在各个领域中都可以看到它的身影，从而使我们的世界变得更加精彩.软件作为信息技术的灵魂，更是扮演了极其重要的角色，因为在现代社会中已经很难想象没有软件会是怎样?所以软件产业也正在全球经济中占据越来越重要的地位.而软件工程已成为软件产业健康发展的关键技术，那么它的发展又是如何呢?

自1968年提出“软件工程”这个词以来，已经历了30多年的时间.那时为了解决软件危机，人们希望通过其他工程的技术方法和管理手段，将软件的开发纳入工程化的轨道.在70年代取得了大量的研究成果的基础上，已基本形成了软件工程的概念、框架、方法和手段，成为软件工程的第一代，我们称之为传统软件工程.

20世纪80年代以来，面向对象的方法与技术已受到广泛的重视，80年代出现的Smalltalk-80标志着面向对象程序设计进入了实用阶段，80年代中到90年代，研究的重点转移到面向对象的分析与设计，从而演化成一种完整的软件开发方法和系统的技术体系，成为软件工程的第二代，有不少人称之为对象工程.

到了80年代中期，人们在研究和实践中发现，为了提高软件生产率，并使软件质量得到保证，其关键在于软件开发和维护中的管理和支持能力，并认识到最关键的是“软件过程”，因此，1984年开始掀起了“软件过程运动”，从而逐步形成软件过程工程，并成为软件工程的第三代.

进入90年代之后，软件工程的一个重要进展就是基于构件的开发方法.为了提高软件生产力，不草率地开发应用程序，则要尽可能地利用可复用的构件，组装成新的应用软件系统.随着Internet技术的飞速发展，大量的分布式处理系统需要开发，这种方法的重要性也日益显露出来，从而成为软件工程的第四代，也有不少人称之为构件工程.

任何事物只要停滞不前就是消亡的开始.软件工程至今还在不断发展，无论是构件工程、过程工程以及对象工程都有不少新的进展，即使是传统软件工程中的一些基本概念、框架，随着技术的进步也在发生不少演变.总之，软件工程代与代之间并没有鸿沟，它们不仅有交叉重叠，也有携手并进.这种说法只是为了阐述软件工程的迅速发展，从不同的角度来研究和实践而已，并没有严格的含义.不管怎样论述，软件工程是一门处于前沿地位的重要学科，需要我们认真地研究和细心地学

习，也需要我们在技术实践中不断创新，并做出应有贡献.

自 1985 年正式出版第一本软件工程的书籍(即《软件工程指南》，朱三元等编著，上海翻译出版公司出版)以来，还未能找到较为全面地反映整个软件工程技术进展的教学参考书或教材，因此，在本书撰写中，就我们所了解和掌握的技术，力求系统性、新颖性、实用性和简明性，将软件工程所涉及到的技术进行概要阐述. 系统性是指将 30 多年来软件工程积累的技术作了较为系统的叙述；新颖性是指将截止到 2000 年不断涌现的新技术、新概念作了较为扼要地介绍；实用性是指便于博采各种软件工程技术之长，并结合自身的特点加以应用；简明性是指对各种软件工程技术从概念上阐述清楚，而不作太多细节的描述.

本书由九章组成. 第一章概述 20 世纪 70 年代到 80 年代软件工程的基本概念和方法. 第二、三章概要介绍面向对象的分析和设计技术，并着重介绍统一建模语言 UML 的技术，包括用例建模、类和对象建模、动态建模和物理体系结构建模等. 第四、五章主要介绍软件过程中的基本技术，包括过程建模、过程度量和过程改进，以及近期的轻载方法. 第六至八章概要介绍软件复用技术、构件生产技术、构件组装技术、构件接口技术等. 第九章主要介绍软件评审、生存周期软件开发 V 模型、软件测试自动化技术以及配置管理. 另外，在每章的章末均列出参考文献，可以从中检索得到相关的软件工程技术细节. 此外附录中给出了软件工程职业道德规范和实践要求(5.2 中文版)，可供读者参考.

本书第二章、第三章、第六章、第七章由钱乐秋教授撰写，第四章、第五章由宿为民博士撰写，第 5.4 节是根据居德华教授在上海软件行业协会会员大会(2001 年 6 月 27 日)上的报告，由本人改编. 第八章由饶若楠副教授撰写. 第一章和第九章由本人撰写. 全书由本人统一筹划，并进行统稿.

本书承山东中创软件工程股份有限公司大力支持，南京大学徐家福教授亲自细心审阅本书稿，两次莅临上海与作者商讨书稿中的有关问题，同时提出非常有益的宝贵意见，并欣然作序. 科学出版社鞠丽娜编辑精心审校，在此一并深致谢忱.

由于本书撰写时间断断续续，时间、精力不够集中，限于水平，疏漏、欠妥、谬误之处在所难免，恳请读者指正.

朱三元

辛巳孟夏于沪上

目　录

中创软件丛书序言
序
前言
第一章　传统软件工程概述 …… **1**
1.1　引言 …… 1
1.2　软件工程史前期与软件危机 …… 2
1.3　软件工程定义 …… 3
1.4　软件工程与一般工程的差异 …… 5
1.5　软件生存周期 …… 6
1.6　程序设计方法 …… 8
1.6.1　结构化程序设计 …… 9
1.6.2　模块化与信息隐蔽 …… 10
1.6.3　面向对象程序设计 …… 13
1.7　软件开发模型 …… 15
1.7.1　瀑布模型(waterfall model) …… 16
1.7.2　渐增模型(incremental model) …… 17
1.7.3　演化模型(evolutionary model) …… 18
1.7.4　螺旋模型(spiral model) …… 19
1.7.5　喷泉模型(fountain model) …… 21
1.7.6　智能模型(intelligent model) …… 21
1.8　软件开发方法 …… 22
1.8.1　模块化方法(modular method) …… 23
1.8.2　结构化方法 …… 23
1.8.3　面向数据结构方法 …… 25
1.8.4　面向对象方法 …… 26
参考文献 …… 27
第二章　面向对象的分析和设计 …… **29**
2.1　面向对象的基本概念 …… 30
2.1.1　对象 …… 30

2.1.2 类 …… 31
2.1.3 继承 …… 32
2.1.4 消息 …… 33
2.1.5 多态性(polymorphism)和动态绑定(dynamic binding) …… 33
2.2 面向对象分析(Object-Oriented Analysis,OOA) …… 33
2.2.1 OOA 的目标和步骤 …… 33
2.2.2 分析过程 …… 34
2.2.3 建造对象-关系模型 …… 38
2.2.4 建立对象-行为模型 …… 38
2.3 面向对象设计(Object-Oriented Design, OOD) …… 39
2.3.1 OOD 的步骤 …… 39
2.3.2 系统设计 …… 39
2.3.3 对象设计 …… 41
2.3.4 设计模式 …… 42
2.4 几种典型的面向对象方法简介 …… 42
2.4.1 Coad & Yourdon 方法 …… 42
2.4.2 OMT 方法 …… 44
2.4.3 Booch 方法 …… 46
2.4.4 OOSE 方法 …… 46
参考文献 …… 47
第三章 统一的建模语言(UML) …… 49
3.1 UML 概述 …… 49
3.1.1 发展历史 …… 49
3.1.2 UML 简介 …… 50
3.1.3 图 …… 51
3.1.4 视图 …… 53
3.2 用例建模 …… 54
3.2.1 用例图 …… 55
3.2.2 确定行为者 …… 55
3.2.3 确定用例 …… 56
3.2.4 用例之间的关系 …… 58
3.3 类和对象建模 …… 58
3.3.1 类图和对象图 …… 58
3.3.2 确定类 …… 60

3.3.3 UML 中类之间的关系 …… 62
3.3.4 包 …… 68
3.3.5 模板(templates) …… 69
3.3.6 一个类图的实例 …… 70
3.4 动态建模 …… 71
3.4.1 消息 …… 71
3.4.2 状态图 …… 71
3.4.3 时序图 …… 81
3.4.4 协作图 …… 83
3.4.5 活动图 …… 85
3.5 物理体系结构建模 …… 88
3.5.1 逻辑体系结构和物理体系结构 …… 88
3.5.2 构件图 …… 90
3.5.3 部署图 …… 91
3.6 使用 UML 的过程 …… 94
3.6.1 UML 过程的基础 …… 94
3.6.2 面向对象方法的一般过程 …… 95
3.6.3 Rational Objectory 过程 …… 98
参考文献 …… 99
第四章 软件过程工程 …… 101
4.1 软件过程概念及软件过程工程框架 …… 102
4.1.1 软件过程 …… 102
4.1.2 软件过程工程 …… 106
4.1.3 软件过程周期 …… 109
4.2 软件过程模型及其构造方法 …… 113
4.2.1 软件过程模型 …… 113
4.2.2 软件过程建模方法 …… 115
4.2.3 软件过程建模语言 …… 119
4.3 软件过程的实施机制 …… 122
4.3.1 过程实施概述 …… 122
4.3.2 软件过程的例化 …… 124
4.3.3 过程运作 …… 128
4.3.4 过程的模拟 …… 130
参考文献 …… 132

第五章　软件过程改进 …… **133**
5.1　软件过程的度量及改进 …… 133
5.1.1　过程度量的基本概念 …… 133
5.1.2　过程度量的通用模式 …… 137
5.1.3　软件过程的度量模型 …… 139
5.1.4　过程改进 …… 142
5.2　CMM 软件过程成熟度模型及其过程改进模式 …… 145
5.2.1　CMM 概述 …… 145
5.2.2　CMM 的内容和组成部分 …… 146
5.2.3　基于 CMM 的过程改进 …… 152
5.3　SPICE 软件过程改进模式 …… 153
5.3.1　SPICE 概述 …… 153
5.3.2　软件过程评价标准的框架 …… 153
5.3.3　SPICE 过程改进模式 …… 159
5.4　AGILE 开发方法和过程 …… 161
5.4.1　极值程序设计(eXtreme Programming,XP) …… 162
5.4.2　SCRUM 软件开发过程 …… 162
5.4.3　自适应软件开发(Adaptive Software Development,ASD) …… 165
5.4.4　Crystal 方法族 …… 165
5.4.5　瑞理统一过程(Rational Unified Process,RUP) …… 166
参考文献 …… 166
第六章　软件复用和构件技术 …… **168**
6.1　软件复用概述 …… 168
6.1.1　软件复用的定义 …… 168
6.1.2　软件复用的过程 …… 169
6.1.3　软件复用的粒度 …… 170
6.1.4　软件复用的形式 …… 171
6.2　生产者复用和消费者复用 …… 172
6.2.1　生产者复用(producter reuse) …… 172
6.2.2　消费者复用(consumer reuse) …… 173
6.3　软件复用经济学 …… 174
6.3.1　软件复用对质量、生产率和成本的影响 …… 174
6.3.2　复用成本估计 …… 176
6.4　构件与体系结构 …… 177

6.4.1 什么是构件和体系结构 …… 177
6.4.2 基于构件的软件体系结构风格 …… 178
6.5 构件与构件系统 …… 181
6.5.1 对可复用构件的要求 …… 181
6.5.2 构件模型 …… 182
6.5.3 构件系统 …… 183
6.5.4 构件的分类 …… 186
6.5.5 构件库管理 …… 188
6.6 建造构件 …… 189
6.6.1 建造可复用构件 …… 189
6.6.2 可变性机制 …… 191
参考文献 …… 193
第七章 软件复用的实施和组织 …… 194
7.1 基于复用的软件开发过程重组 …… 194
7.1.1 以往的软件开发技术缺乏对软件复用的支持 …… 194
7.1.2 为复用改变软件开发过程 …… 195
7.1.3 软件复用的组织结构 …… 197
7.2 软件复用工程的过程 …… 198
7.2.1 应用族工程(AFE) …… 198
7.2.2 构件系统工程(CSE) …… 200
7.2.3 应用系统工程(ASE) …… 201
7.3 渐增地、系统地实施软件复用 …… 202
7.3.1 向复用业务过渡的关键要素 …… 202
7.3.2 渐增地系统地采用复用技术 …… 202
7.3.3 实施系统复用需遵循的原则 …… 208
参考文献 …… 208
第八章 构件接口技术 …… 210
8.1 概述 …… 210
8.1.1 基本概念 …… 210
8.1.2 企业级构件系统的构件接口技术 …… 211
8.2 EJB/J2EE 技术 …… 212
8.2.1 J2EE 概述 …… 212
8.2.2 EJB 技术 …… 215
8.3 COM＋技术 …… 222

8.3.1 COM+技术概述 …… 222
8.3.2 构件对象模型 COM/DCOM 基础 …… 224
8.3.3 COM+构件新特性 …… 229
8.3.4 COM+系统及其服务 …… 232
8.4 CORBA …… 235
8.4.1 CORBA 概述 …… 235
8.4.2 CORBA 的核心概念和体系结构 …… 236
8.4.3 CORBA 应用开发 …… 242
8.4.4 CORBA 构件模型 …… 246
参考文献 …… 250
第九章 产品化技术 …… 251
9.1 软件评审 …… 251
9.1.1 软件评审任务 …… 252
9.1.2 软件评审方法 …… 253
9.1.3 软件评审的特点 …… 254
9.2 生存周期软件开发 V 模型 …… 255
9.2.1 测试案例设计原则 …… 256
9.2.2 软件测试基本技术 …… 257
9.2.3 软件测试自动化技术 …… 260
9.3 软件配置管理 …… 264
9.3.1 基本概念 …… 264
9.3.2 软件配置管理的任务 …… 265
9.3.3 配置管理计划编制大纲 …… 271
9.3.4 配置数据库 …… 272
参考文献 …… 272
附录 1 软件工程职业道德规范和实践要求 …… 274
附录 2 中英名词对照表 …… 282
附录 3 缩略词表 …… 293

第一章　传统软件工程概述

1.1　引　　言

1962年6月，美国飞向金星的第一个空间探测器(水手I号)，因其飞舱中的计算机导航程序之一的一个语句的语义出错，致使偏离航线无法取得成功. 这个语句的错误并不是语法错，而是具有完全不同于程序员所期望的意思. 可以称得上是世界上最精心设计，并花费了巨额投资的美国阿波罗登月飞行计划的软件，仍然没有避免出错，例如阿波罗8号太空飞船的一个计算机软件错误，造成存储器的一部分信息丢失；阿波罗14号在飞行的10天中，出现了18个软件错误. 当时，软件系统的可靠性得不到保证，几乎没有不存在错误的软件系统.

那时，计算机已有近20年的历史，一方面硬件成本每隔2～3年降低一半，内存和外存的成本每年降低40%左右，硬件性能价格比每十年提高一个数量级，但所需的软件很少能在成本、时间进度、功能规模、维护能力等方面达到要求，特别是可靠性难以符合人们的需要，正如E. E. David指出的那样，大型系统的软件生产已经成为管理人员担惊受怕的项目，于是，人们在20世纪60年代后期惊呼发生了软件危机(software crisis).

自1968年由NATO(北大西洋公约组织)在德国格密斯(Garmish)举行的学术会议上正式提出“软件工程(software engineering)”这一术语以来，软件工程已有三十多年的历史. 应该说，在这期间软件工程发展极快. 如果说20世纪20年代为电学时代，40年代为战争时代，60年代为太空时代的话，那么，毫无疑问，20世纪80年代是计算机及其软件的时代. 没有什么领域能够像计算机及其软件那样成就显赫，计算机及其软件已向全人类展示出一幅全新的认知画面，人们能够通过它看到未来世界的行为方式，其前景令人痴迷而神往. 而这一切，决不能忘记新兴的软件工程所起的推动作用. 软件工程作为工程学科家族中的新成员，它指导人们科学地开发软件、制作软件产品、集成计算机系统，以期降低成本、提高质量、符合进度要求. 如果说软件是计算机及其应用的灵魂的话，那么可以毫不夸张地说，软件工程是计算机和信息产业的支柱，或不可分割的组成部分，其重要性于此可见一斑.

30多年来，软件工程与计算机硬件、应用需求相互影响、相互促进、共同发展，

特别是在近十多年中，软件工程发展神速. 在软件工程的理念、方法、模型、管理等方面均取得了长足的进步，因此很有必要进行一定程度的归纳和总结. 在论述近代软件工程时，我们有必要对 20 世纪 80 年代所形成的软件工程的概念、方法、技术、管理进行回顾，同时亦需对那时的软件工程概念，从近代的软件工程理念进行必要的修正和补充.

1.2　软件工程史前期与软件危机

从 40 年代开始，人们从在 MARK-I 和 ENIAC 计算机上编制程序，到软件工程术语提出时为止的二十多年的时间里，对软件开发的理解就是编程序，且编程是在一种无序的、崇尚个人技巧的状态中完成的. 同今天的软件开发相比，那时的编程有以下特点：

1) 软件规模相对较小. 原因有二：其一，人们对软件可能达到的功能认识有限，那时最为关心的是计算机硬件的发展. 作为一个计算机专业人员，他不太关心软件问题(只有为数不多的专业人员才去关心软件)，但他必须懂得计算机的结构. 作为一个机构，其大量资金也是用于计算机硬件开销上，软件只是作为展现其硬件性能的一种手段而投入少量资金. 其二，硬件性能从某种意义上说左右着人们对软件的需求，速度低、存储量小的硬件只能支持简单、单一的软件应用. 专业人员在考虑软件需求时，总是不自觉地在心中核算着硬件支持的可能性，这种现象从根本上阻碍了软件的广泛应用，从而限制了软件规模.

2) 编程作为一门技艺. 大部分软件技术人员不太关心他人的工作，他们往往陶醉于自己的编程技巧，并且那时也无编程规范与标准，这也是由软件规模所决定的. 决定软件质量的惟一因素就是该编程人员的素质，时间进度亦是如此. 根据 H. Sackman 等人的调查，工作好的人与工作差的人，在软件生产效率上的比例是 10∶1，在程序处理速度和存储容量上的比例是 5∶1.

3) 缺少有效方法与软件工具的支持. 在当时几乎谈不上有效的编程方法，使用最多的亦只是简单的控制流程图(即框图). 软件工具只有子程序库、装入程序、编辑程序(后来有连接程序)、排错程序以及汇编和编译程序. 编程作为一门技艺也是有此背景的.

4) 不可能重视软件开发的管理. 由于人们重视个人的技能，再加上软件开发过程能见度低，许多管理人员甚至根本不知道他们的软件技术人员究竟做得如何，从而造成管理活动是少量的甚至不存在. 直到今天，这种观念在一些机构中仍有残留，为此，我们对软件设计的管理问题必须给予更多的重视.

5) 软件开发后的维护工作很难进行. 由于人们重视个人技能，一旦需要做某

些修改，就要原编程人员进行修改. 如果他已离开本机构，就需要别人去读懂他的程序，此项工作极其辛苦，说不定修改了一处，反而上百处出了漏洞，变成了得不偿失.

进入 20 世纪 60 年代，客观上需要制作一些大型软件，这样就出现了像 1.1 节所描述的例子. 国外在开发一些大型软件系统时，遇到了许多困难，有些系统最终彻底失败了；有些系统虽然完成了，但比原定计划推迟了好几年，而且费用大大超过了预算；有些系统未能圆满地符合用户当初的期望；有些系统则无法进行修改维护. IBM 公司的 OS/360 系统和美国空军某后勤系统都花费了几千人年的努力，历尽艰辛，但结果令人失望.

究其原因，大型软件系统大大地增加了软件复杂性，即技术复杂性和管理复杂性指数地上升. 就拿软件维护来说，也比计算机硬件来得复杂. 美国 Dalg 研究了一个 16 万条汇编语句与 17 万个逻辑门组成的大型实时系统的记录，他发现：

1) 软件维护时修改一条汇编语句的工作量是硬件维护时修理一个逻辑门的四倍；

2) 软件维护费用是硬件维护费用的四倍；

3) 编写一条汇编语句所需工作量是研制硬件一个逻辑门的两倍.

在那时，大型软件系统存在潜在错误已经司空见惯，要开发一个没有错误的软件极其困难，不管你多么警惕，漏洞亦会偷偷地混到程序中去.

20 世纪 60 年代末期所发生的软件危机，体现在软件可靠性没有保障、软件维护费用不断上升、进度无法预测、成本增长无法控制、程序人员无限度地增加等各个方面，以致形成人们难以控制软件开发的局面.

1.3　软件工程定义

1968 年提出的软件工程术语，其概念是模糊的，也没有给出其明确的定义和内涵，犹如尚未出世的婴儿却已起了名字一样，软件工程还只是看不清具体形态与内涵的一个术语.

从此，人们对于软件开发是否已符合工程化思想?软件工程作为一门学科有何自身特点等问题展开了广泛的讨论与研究，从而才对软件工程有了各种各样的定义，我们在此列举一些定义如下：

P. Wegner 和 B. Boehm 认为软件工程可定义为：科学知识在设计和构造计算机程序，以及开发、运作和维护这些程序所要求的有关文档编制中的实际应用.

F. L. Bauer 认为，为了经济地获得软件，这个软件是可靠的并且能在实在的计算机上工作，所需要的健全的工程原理(方法)的确立和使用.