面向对象软件工程

陈世鸿　彭　幕　等编著
华　宏　　审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL:http://www.phei.com.cn

面向对象软件工程

陈世鸿　彭蓉　等编著

华　宏　审校

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

本书从工程化角度讨论了面向对象方法在软件开发全过程中的基本原理及相应技术，即OOSE(Object-Oriented Software Engineering)。重点讨论了OOSE的思维简式、简求模型、分析模型、设计模型、实现模型、测试模型及管理技术等诸方面的问题。

全书共分16章，第1章讨论软件开发过程中存在的关键问题；第2章从模型化角度阐明了传统软件工程方法及存在的问题；第3～6章讨论了面向对象方法的基本原理；第7～12章以同一个实例论述了从需求到测试各模型的建立方法和步骤，并在第13、14两章中以两个实际应用系统详细实践了OOSE的开发过程，以便给读者具体的实战指导；第15章讨论了面向对象软件工程的管理问题；第16章介绍了其他面向对象方法及与OOSE相关概念的比较。可以认为OOSE是最具体的，可实际操作的两向对象的工程化软件开发方法。本书既可用于计算机专业本科以上学生的软件工程方法学教材和教学参考书，也是软件开发工作者的实用指导教程。

图书在版编目(CIP)数据

面向对象软件工程/陈世鸿等编著．－北京：电子工业出版社，1999.3
ISBN7-5053-5094-3

Ⅰ．面…　Ⅱ．陈…　Ⅲ．面向对象语言－软件开发　Ⅳ．TP311.52

中国版本图书馆CIP数据核字(98)第40759号

JS199/17

书　　名：面向对象软件工程
编 著 者：陈世鸿　彭蓉　等
审 校 者：华　宏
责任编辑：潘海
印 刷 者：北京兴华印刷厂
装 订 者：三河市双峰装订厂
出版发行：电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：17.25　字数：436.8千字
版　　次：1999年5月第1版　1999年5月第1次印刷
书　　号：ISBN 7-5053-5094-3/TP·2534
定　　价：34.00元

凡购买电子工业出版社的图书，如有缺页、倒页、脱页、所附磁盘或光盘有问题者，请向购买书店调换。
若书店售缺，请与本社发行部联系调换。电话　68279077

前　言

人们普遍认为八十年代兴起的面向对象软件开发技术将取代六十年代末的结构化软件开发技术，是软件开发方法的第二次革命。近十年来，不仅面向对象程序设计思想被广大软件工作者所认识和接受，而且已广泛地应用于各个领域的程序设计之中。

近年来，虽然在面向对象程序设计基础上研究了多种面向对象软件开发方法，但尚缺乏一种协调的支持软件开发全生命期的、实际可操作的工程化面向对象方法，本书论述的 OOSE 方法正好弥补了这一空白。纵观近十年来面向对象方法的研究，所见到的对象式方法大多只能用于软件开发的某个阶段，有的方法仅为某种设计原则。如 BOOCH(1991)的面向对象设计(OOD)适用于软件的设计阶段，Shlaer 和 Mellor(1988)的面向对象系统分析(OOSA)实质是基于数据模型的信息分析，Coad 和 Yourdon(1991)的面向对象分析(OOA)仅适用于软件系统的分析阶段，面 Wasserman 等人的面向对象结构设计(HOOD)仅为一种面向对象设计记法和对结构图的扩充。

OOSE 是一种支持软件开发全生命期的面向对象的工程化方法，从需求分析到测试的各个阶段分别建立一个规范协调模型，即 RM、AM、DM、IM 和 TM。各模型间具有可跟踪性，这就是说，任一对象在前后模型中可找到其对应关系，OOSE 的宗旨是所建系统的可靠性和可维护性。

本书是作者在多年软件开发方法的研究、实践和教学基础上参考国内外著作撰写而成。全书共分 16 章，第 1 章讨论了软件开发过程中存在的关键问题；第 2 章从模型化角度阐明了多种传统软件开发方法及其存在的问题；第 3～6 章形式化地讨论了面向对象的基本概念、设计原理和计算系统；第 7～12 章以同一个实例论述了从需求到测试各模型的建立方法和步骤，并在第 13、14 两章中以两个不同类的实际应用系统详细实践了 OOSE 开发过程，以给读者具体实际指导；第 15 章讨论了面向对象软件工程的管理问题；第 16 章介绍了其他面向对象方法及与 OOSE 相关概念的比较。可以认为 OOSE 是最具体、可实际操作的面向对象的工程化软件开发方法。本书既可用于计算机专业本科以上学生软件方法学教材和教学参考书，也是软件开发工作者的实用指导教程。

在本书的编著与出版过程中得到刘汉斌、许莲英、朱福喜、陈版芳、黄宝贵、代芳等同志的大力帮助与支持，在此表示衷心感谢。由于作者学识水平有限，书中谬误与欠妥之处，恳请读者批评指正。

作者

1998 年 6 月

引　言

如今，对从事计算机科学的人们来说，“软件工程”一词恐怕不会陌生，大概也不会再有人怀疑按工程方法开发软件（特别是大型软件）系统的必要性与重要性。

自 1968 年北大西洋公约组织成员国的软件工作者提出软件工程的概念以来，已经历了 30 年的发展期，软件工作者在软件工程学方面作出了许多卓有成效的研究，所提出的各种工程化方法和技术在软件系统开发中起到了不可忽视的作用。然而，在软件系统的实际开发过程中，严格按某种工程化方法的标准规范和设计步骤实施的系统尚不多见，不少大型软件系统的开发仍然处在高强度、低效率、难以保障质量的困境中，这与硬件技术及其产品的高速发展形成了鲜明对照。

引起如上所述问题的因素众多，但其根本原因在于软件产品的特殊性导致工程化方法实施的复杂性和难度（当然，不按工程化方法开发大型软件系统更难，甚至不可能）。

近 30 年来，广大软件科学工作者虽然经过不懈努力，先后提出了结构化分析与设计（SA/SD）、面向数据的设计（Jackson）、原型化设计（Prototyping）等多种软件设计方法，并强烈呼吁软件生产的工程化思想。但由于这些方法都存在各种不同程度的缺陷，没有哪种方法能根本奏效，“软件危机”的状况依然存在。

1979 年，美国财政部对为美国联邦政府开发的九个软件项目进行了调查，虽然这九个项目的工程规模并不大，但调查结果是令人忧虑的（见图 1）。调查表明 47% 的资金花费在从未使用过的软件上。更糟糕的是，另外 29% 的资金花费在那些交付后需再开发以及交付后在 GAO 调查时已被遗弃的软件上。总之，成功的项目仅占 3～4%。该调查虽已过近二十年，但软件开发的现状并没有得到根本性的改变。我国软件生产现状更是如此，除少数中小规模软件有成功的例子外，成功的大型软件开发的实例实属罕见。因此，改进传统软件开发方法和研究新的软件开发方法仍然是软件产业的当务之急。

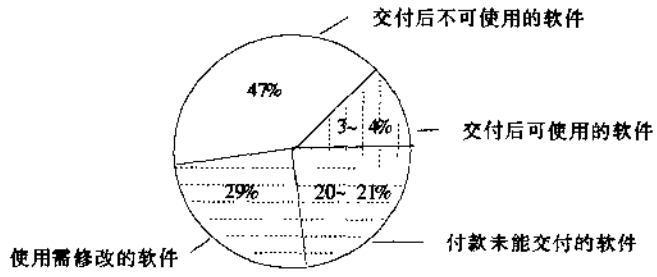

图 1　成功与非成功软件项目比

80 年代初开始引起人们广泛关注的面向对象程序设计语言与随之发展起来的面向对象软件设计方法被视为一种全新的软件开发方法，其基本思想是：对问题领域进行自然分割，以更接近人类通常思维的方式来建立问题领域的模型，便于对现实世界的客观实体进行结构模拟和行为模拟，从而使设计出的软件尽可能直接地表现出问题求解过程。

十多年来，软件工作者对面向对象的程序设计及其在各领域中的应用进行了广泛研究，已

成功地设计出多种面向对象的程序设计语言，如 Simula、Smalltalk、Eiffel、C ++ 等。但其基本理论、设计规范及实现技术等还有待深入研究，尤其是用面向对象方法实施软件工程化设计尚不完善。本书重点是讨论用面向对象技术实现工程化软件系统开发的方法和基本原理，面不是一本单纯的面向对象程序设计的书。我们关心的是大系统采用面向对象方法在开发过程中的技术及相关步骤，即面向对象的软件工程（Object-Oriented Software Engineering，简写为 OOSE），强调 OOSE 在软件系统开发中可实际操作的步骤，即详细讨论了其思维模型、需求分析模型、设计模型、实现模型、测试技术和管理等有关问题，以及它们与传统软件开发方法的差异。

值得指出的是，OOSE 是一种新的软件开发方法与技术，面决不是成功开发软件系统的灵丹妙药。软件系统从手工艺技术型的设计转变为工程化的生产不仅仅是某种新技术的引进，同时还依赖于开发人员思想意识的转变和整个开发过程的组织与管理。读者虽可从本书中学到 OOSE 的实际方法，但切不可忽视软件系统开发的各个方面。

目　录

第 1 章　绪论 …… (1)
1.1　软件及其发展的三个阶级 …… (1)
1.2　软件工程 …… (2)
1.3　软件工程面临的问题 …… (3)
1.3.1　软件可靠性 …… (3)
1.3.2　软件生产率 …… (4)
1.3.3　软件再应用 …… (4)
1.3.4　软件维护 …… (5)
1.4　软件的生命期 …… (6)
第 2 章　软件开发技术 …… (8)
2.1　结构化分析和设计技术 …… (8)
2.1.1　SADT 的图形语言 …… (8)
2.1.2　SADT 模型 …… (8)
2.1.3　系统开发过程与 SADT 的缺陷 …… (10)
2.2　Jackson 软件设计技术 …… (11)
2.2.1　数据结构表示法 …… (12)
2.2.2　系统开发过程与缺陷 …… (13)
2.3　快速原型设计技术 …… (14)
2.3.1　原型模型与建模步骤 …… (15)
2.3.2　原型技术的特点 …… (17)
2.4　面向对象设计技术 …… (17)
第 3 章　什么是面向对象 …… (19)
3.1　对象 …… (20)
3.2　类和实例 …… (23)
3.3　多形 …… (25)
3.4　继承性 …… (26)
3.5　建立合理的继承性结构 …… (29)
3.6　多继承性 …… (31)
第 4 章　面向对象的程序设计 …… (33)
4.1　几种典型 OOPL …… (33)
4.2　对象 …… (37)
4.3　类和实例 …… (37)
4.4　类作为对象 …… (39)
4.5　继承性 …… (41)
4.6　多形 …… (43)
4.7　实例 …… (45)
4.8　OOP 计算模型 …… (47)

第5章 面向对象的开发技术 …… (49)
5.1 面向对象分析 …… (49)
5.2 面向对象的设计 …… (50)
5.3 面向对象测试 …… (51)
第6章 系统的组成与模型 …… (53)
6.1 系统开始是构造模型 …… (53)
6.1.1 模型 …… (53)
6.1.2 软件系统结构 …… (55)
6.1.3 开发过程 …… (56)
6.1.4 过程和模型 …… (57)
6.2 需求模型 …… (58)
6.2.1 使用者和使用事件 …… (58)
6.2.2 使用事件驱动设计 …… (60)
6.3 分析模型 …… (61)
6.3.1 分析模型的对象 …… (61)
6.3.2 由 RM 构造 AM …… (66)
6.4 设计模型 …… (67)
6.4.1 设计模型的对象 …… (67)
6.4.2 用 DM 工作 …… (69)
6.5 实现模型 …… (70)
6.6 测试模型 …… (70)
第7章 分析 …… (71)
7.1 引论 …… (71)
7.1.1 为什么要有分析过程 …… (71)
7.1.2 分析过程中做什么 …… (71)
7.1.3 实例 …… (72)
7.2 RM …… (73)
7.2.1 用户的需求定义是系统开发的基础 …… (73)
7.2.2 使用者 …… (73)
7.2.3 使用事件 …… (74)
7.2.4 界面描述 …… (78)
7.2.5 问题领域对象 …… (78)
7.2.6 RM 的进一步修正 …… (79)
7.2.7 讨论 …… (81)
7.3 AM …… (82)
7.3.1 界面对象 …… (83)
7.3.2 命名关联 …… (84)
7.3.3 实体对象 …… (87)
7.3.4 控制对象 …… (90)
7.3.5 子系统 …… (93)

第 8 章　构造 …… (96)
8.1　引论 …… (96)
8.1.1　为什么要有构造过程 …… (96)
8.1.2　构造阶段做什么 …… (97)
8.2　DM …… (97)
8.2.1　可跟踪性 …… (97)
8.2.2　实现环境 …… (99)
8.2.3　关联图 …… (102)
8.2.4　消息定义 …… (104)
8.2.5　关联图的结构 …… (106)
8.2.6　扩展使用事件 …… (108)
8.2.7　一致化 …… (108)
8.3　块设计 …… (109)
8.3.1　块的界面设计 …… (109)
8.3.2　对象行为 …… (112)
8.3.3　块的内部结构 …… (118)
8.3.4　实现 …… (118)
8.3.5　探针的实现 …… (122)
8.4　结构工作 …… (123)
8.4.1　现存的产品 …… (124)
8.4.2　抽象 …… (124)
8.4.3　逐渐式开发 …… (125)
8.4.4　进一步讨论的问题 …… (125)
第 9 章　实时系统规模描述 …… (126)
9.1　实时系统的分类 …… (126)
9.2　基本问题 …… (127)
9.3　分析 …… (127)
9.4　构造 …… (128)
9.5　测试和验证 …… (133)
第 10 章　数据库规格描述 …… (134)
10.1　引言 …… (134)
10.2　关系 DBMS …… (135)
10.2.1　问题的提出 …… (135)
10.2.2　从对象到表 …… (135)
10.2.3　继承性问题 …… (136)
10.2.4　模型化持久性对象 …… (138)
10.3　面向对象 DBMS …… (140)
10.4　讨论 …… (141)
第 11 章　部件 …… (142)
11.1　引言 …… (142)

11.1.1 软件工程中的可重用性 …… (142)
11.1.2 部件作为增强机制 …… (142)
11.1.3 现在为什么还没有好部件 …… (143)
11.1.4 新的探讨 …… (144)
11.2 软件部件 …… (144)
11.3 部件的使用 …… (146)
11.3.1 找出使用部件的位置 …… (146)
11.3.2 用部件实现 …… (147)
11.4 部件的管理 …… (148)
11.4.1 部件的构造 …… (148)
11.4.2 部件系统 …… (149)
11.4.3 部件文档 …… (151)
11.5 小结 …… (152)
第 12 章 测试 …… (154)
12.1 引言 …… (154)
12.2 测试中的一般问题 …… (155)
12.2.1 测试的目的 …… (155)
12.2.2 测试策略 …… (156)
12.2.3 等价集 …… (157)
12.2.4 自动测试 …… (158)
12.3 单元测试 …… (159)
12.3.1 需求规格测试 …… (159)
12.3.2 基于状态的测试 …… (160)
12.3.3 结构测试 …… (161)
12.4 综合测试 …… (163)
12.5 系统测试 …… (164)
12.6 测试过程 …… (165)
12.6.1 测试计划 …… (165)
12.6.2 测试标记 …… (166)
12.6.3 测试规格说明书 …… (166)
12.6.4 执行测试 …… (166)
12.6.5 错误分析 …… (167)
第 13 章 示例学习之一:仓库管理系统 …… (168)
13.1 例子概述 …… (168)
13.2 ACME 仓库管理有限公司 …… (168)
13.3 需求模型 …… (169)
13.3.1 仓库间人为测配 …… (171)
13.3.2 客户要求提货 …… (173)
13.3.3 抽象事件 …… (174)
13.4 分析模型 …… (175)

13.4.1 分析对象 …… (175)
13.4.2 子系统 …… (184)
13.5 构造 …… (186)
13.5.1 确定实现环境 …… (186)
13.5.2 块结构 …… (187)
13.5.3 使用事件设计 …… (190)
13.5.4 块设计 …… (193)
第 14 章 示例学习之二:通信 …… (195)
14.1 通信转接系统 …… (195)
14.2 需求模型 …… (197)
14.2.1 局域通话使用事件 …… (199)
14.2.2 变更用户实例 …… (200)
14.2.3 抽象使用事件 …… (201)
14.3 分析模型 …… (202)
14.3.1 局域通话使用事件 …… (202)
14.3.2 进一步细化 …… (204)
14.3.3 用户变更使用事件 …… (205)
14.3.4 子系统 …… (205)
14.4 设计模型 …… (206)
14.4.1 块结构 …… (206)
14.4.2 进程构造 …… (209)
14.4.3 使用实例设计 …… (209)
14.4.4 块接口 …… (212)
14.5 实现模型 …… (214)
14.5.1 用 If-语句实现 …… (214)
14.5.2 执行:分支实现 …… (216)
第 15 章 面向对象软件工程的管理 …… (218)
15.1 概述 …… (218)
15.2 项目选择和准备工作 …… (218)
15.2.1 引进新的开发方法 …… (218)
15.2.2 如何选择第 1 个项目 …… (219)
15.2.3 教育和培训 …… (220)
15.2.4 风险性分析 …… (221)
15.3 产品开发组 …… (223)
15.4 项目组织与管理 …… (226)
15.4.1 逐步增长的开发方法 …… (228)
15.5 项目开发人员 …… (230)
15.6 软件质量保证 …… (232)
15.7 软件度量 …… (234)
第 16 章 其他 OOM …… (238)

16.1 概述 …… (238)
16.2 OOM 简介 …… (239)
16.3 面向对象分析(OOA/Coad-Yourdon) …… (240)
16.3.1 构造 …… (240)
16.3.2 构造方法 …… (241)
16.3.3 交付 …… (241)
16.3.4 讨论 …… (242)
16.4 面向对象设计(OOD/Booch) …… (243)
16.4.1 构造 …… (243)
16.4.2 方法 …… (244)
16.4.3 交付 …… (245)
16.4.4 讨论 …… (245)
16.5 层次化面向对象设计(HOOD) …… (246)
16.5.1 构造 …… (246)
16.5.2 方法 …… (247)
16.5.3 交付 …… (248)
16.5.4 讨论 …… (248)
16.6 对象模型化技术(OMT) …… (249)
16.6.1 构造 …… (249)
16.6.2 方法 …… (250)
16.6.3 交付 …… (250)
16.6.4 讨论 …… (250)
16.7 功能驱动设计 …… (251)
16.7.1 构造 …… (251)
16.7.2 方法 …… (252)
16.7.3 交付 …… (252)
16.7.4 讨论 …… (253)
16.8 小结 …… (253)
附录 A:对象方法学的发展 …… (255)
A.1 简介 …… (255)
A.2 对象方法 …… (256)
A.2.1 开发事件 …… (256)
A.2.2 开发实体对象 …… (257)
A.2.3 进程 …… (259)
A.2.4 根据开发实体对象和进程来研究开发事件 …… (259)
A.2.5 企业构造 …… (260)
A.2.6 根据活动块得到的开发事件 …… (260)
A.2.7 活动块的实现 …… (261)
A.3 对象方法学 …… (261)
参考文献 …… (262)

第1章 绪 论

1.1 软件及其发展的三个阶段

人们常说的计算机系统是由硬件和软件两大部分组成的。用计算机求解问题，程序是不可缺少的，程序被称为软件的实体部分。程序只有在硬件载体上运行才可获得所求问题的解，因而硬件和程序是求解问题的最基本条件。然而，仅有程序也会给使用者带来诸多不便，好的程序应有相应的文字资料，如各种规格说明书、设计说明书、用户手册等。我们称这些文字资料为文档。文档不仅对使用者是必要的，而且对程序开发者更是至关重要的。特别是由多个人经过多年才能完成开发任务的大型程序，人与人之间、开发者与使用者之间都需要有规范的书面文档规定程序的功能、使用环境、使用方法等。所以严格地说，程序和软件是两个不同的概念。程序是指计算机可识别的源程序代码或机器可直接执行的程序代码。而软件是指程序加上开发、使用和维护该程序所需要的全部文档。程序是软件的实体，而文档是软件的重要组成部分。这个公认的浅显认识囊括了软件发展的三个阶段。

软件的发展大致可分为三个时期：即程序时期(1947～60 年代初)、软件＝程序＋说明时期(50 年代末～70 年代初)、软件＝程序＋文档时期(70 年代初～至今，也称软件工程时期)。

我们这里所说的程序是指在计算机上可执行的代码序列。在程序时期，使用者需直接操作计算机硬件，因而一开始就形成了计算机硬件就是计算机这一概念。这一时期，计算机工作者致力于改善硬件结构和可靠性研究，而仅把程序作为机器运行时必须进行的准备工作。每个程序都是为求解某个特定问题而专门设计的，不考虑程序的通用性，更没意识到程序同计算机硬件一样，也是计算机系统不可分割的部分。这一时期的程序设计工作全凭设计者个人经验和技艺独立进行，是一种典型的手工艺智力劳动。程序中出现的错误只能由设计者本人才能修改，面面也无需向他人做任何交待和说明，当时人们脑中只有程序概念而无软件概念。

50 年代末至 70 年代初这段时期，由于硬件技术的飞速发展，与此同时相继问世了一批高级程序设计语言(如 FORTRAN、ALGOL60、COBOL、BASIC 等)的编译程序、解释程序，以及操作系统等支持程序(也称系统程序)和具有较强通用性的应用程序，使得计算机应用从单一的科学计算，迅速发展到数据处理和实时控制等各个应用领域。为使用户方便地使用这些支持程序和通用应用程序，必须提供相应的技术指南、用户手册等说明性文字资料。因而出现了“软件＝程序＋说明”这一概念。这个时期的特征主要表现在三个方面：

(1) 由于程序规模较大，需多人协作才能完成程序编制，我们把这种方式称为“作坊式”生产方式。

(2) 程序的设计与运行维护再也不能由一个人来承担。

(3) 人们已认识到程序再也不是计算机硬件的附属成分，而是计算机系统中与硬件相互依存的不可缺少的部分。

这个时期软件技术取得了程大进展，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、实时系统以及文件管理等。同时出现了更多的通用和专用程序设计语言，在形式语言

理论、编译理论、数据库理论等方面也取得了重大突破，从而诞生了真正的计算机科学。

随着投入市场运行的计算机数量日益巨增，计算机应用领域愈来愈广，软件需求量愈来愈大。因而，美国和西欧一些国家相继建立起庞大的软件公司，专门生产计算机软件，一种新兴产业——软件产业应运而生。

虽然软件需求量不断增大，其复杂度也越来越高，但其生产方式大多停留在手工作坊式生产阶段。这种方式不仅效率低，而且软件质量差，如 IBM 公司 60 年代开发的 OS/360 系统，耗资几千万美元，花费了 5000 多人年，拖延几年才交付使用，交付使用后每年发现近 100 个错误。OS/360 系统开发负责人 Brooks 生动地描述了研制过程中的困难和混乱："……像巨兽陷入泥潭作垂死挣扎，挣扎得越猛，泥浆就沾得越多，最后没有一个野兽能逃脱淹没在泥潭中的命运……。程序设计就象是这样的泥潭，一批批程序员在泥潭中挣扎……。没有料到问题会这样棘手……"。比 OS/360 更糟的软件系统并不少，即花费大量的人力财力，结果半途而废，或完成之日即是遗弃之时。前言中图 1 显示的 GAO 的调查有力地证明了这一事实。这就是人们常说的"软件危机"期。

为了摆脱这一困境，软件工作者一方面研究程序设计方法和程序正确性验证的方法；另一方面寻找工程化的软件系统开发方法。

以 E. W. Dijkstra 提出的"结构化程序设计方法"为代表，引导人们对程序设计方法的广泛研究。在短短几年时间内就提出了模块化、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。虽然有些方法至今仍作为理论上的探讨，但这一时期的研究成果使软件设计者深刻认识到，没有科学的方法作指导，不可能生产出高质量的软件产品，同时还提出了以正确性、结构清晰、便于设计、便于测试和维护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，使程序设计走向更科学的道路，在一定程度上提高了软件可靠性，但设计过程采用的低效手工方式，周期长、代价高，远远不适应软件生产的需要。北大西洋公约组织成员国的软件工作者于 1968，1969 连续两年召开了软件研究会（即 NATO 会议）。集中研究对策，讨论如何摆脱软件危机。考虑到软件系统开发过程同制造一台机器或建造一栋大厦有许多相同之处，于是提出了"软件工程"这个术语，试图用"工程化"的思想作指导来解决软件研究中面临的困难和混乱，从而解决软件危机的困境。这就是我们下节所论述的软件发展第三个时期——软件工程时期。

1.2 软件工程

软件工程的宗旨是为了提高软件生产效率、降低生产成本，以较小的代价（投资）获得高质量的软件产品。虽然这一概念已提出 30 年，但目前对这一概念的定义（或理解）并不统一。下面我们列出了几种具有代表性的解释。

（1）运用现代科学技术来设计并构造计算机程序，以及开发、运行和维护这些程序所必需的相关文件资料（B. W. Boehm，1981）。Boehm 认为软件工程是相关的文件资料，当然他对"设计"一词作了广义的理解。

（2）1983 年 IEEE 给出的定义为：软件工程是开发、运行、维护和修复软件的系统方法。其中"软件"解释为计算机程序、方法、规则以及所要求的文档资料和在计算机上运行时所必需的数据。

(3) 软件工程是应用于计算机软件的定义、开发和维护的整套方法、工具、文档、实践标准和工序。

(4) 软件工程是一组规范化的方法，通过应用这些规范可解决软件开发中产生的问题。

我们还可举出其他定义，但从上述定义中已能归纳出以下几个共同点：

- 软件工程概念的提出是针对计算机软件而言的；
- 软件工程研究的对象涉及到软件的开发和维护过程；
- 软件工程实际上是一些规范化的文档、资料。

由此把软件工程时期定义为软件＝程序＋文档时期就不难理解了。

我们认为软件工程学是研究软件(特别是大型软件)开发的技术、方法、工具、环境和管理的工程学科。上述定义都从不同侧面论述了软件工程所研究的对象、适用范围和所包含的内容。这些定义更符合软件工程学所涉及的内容，但对软件的工程化实施方面的描述都有其自身的缺陷，因而不同程度地脱离了软件工程思想的初衷。

我们所说的软件工程是指软件产品的生产实施过程。就是用软件工程学所研究的软件开发方法、技术、工具和环境，按工业化的生产方式组织和管理软件的生产。因此，一个软件的工业化生产过程应具备以下三个特点：

- 明确的工作步骤
- 详细具体的规范化文档
- 明确的质量评价标准

然而，由于软件产品具有抽象性、逻辑性、非实物性等特点，其生产和维护过程的每一步都是需要经过“思考”而精心制作的智能活动，因此，其设计原理和生产方式与其他工业产品的生产有很大不同。同时，由于软件生产是相当复杂的过程，涉及的因素很多，所以不同软件项目所使用的开发方法、技术、支撑工具与环境是不尽相同的。而且，有些软件项目(或产品)的开发无现存技术，带有不同程度的试探性和风险性，这正是软件产业发展缓慢的根本原因。

本书所讨论的面向对象的软件工程是用面向对象的方法、设计技术和实现工具，按接近于工业化生产方式实施软件生产的全过程。这是因为面向对象的软件工程更有利于解决下节所提出的、多年来一直困挠软件生产的诸多问题。

1.3 软件工程面临的问题

1.3.1 软件可靠性

软件可靠性是指软件系统能否在既定的环境下运行并达到所期望的结果。通常花费在软件测试和排错的代价大约占软件开发代价的40%左右。即便如此，也不能保证经测试的软件就没有错误。好的程序设计技术，如模块化、结构化程序设计能有助于减少程序设计中的错误，提高软件的维护性和可靠性。遗憾的是，测试很难保证一个大的软件系统的正确性。E. W. Dijkstra 对测试的效果讲了一句非常精辟的话：“测试只能说明程序有错，而不能保证程序无错”。但至今为止，工程上还没有找到保证程序正确性的更好方法，所以说软件可靠性是软件工程面临的一个难题。在大型软件系统中，测试是一件很复杂很费时的事情，程序的任何改变都可能会涉及到多个人或软件系统的多个部分，这就使得软件系统花费在测试阶段的代价非常之大。为了提高软件的可靠性，就要付出相应的代价，重要的是在可行性和测试代价之间作出

权衡。

面向对象方法中的封装特点有利于软件系统的测试，在第 12 章的测试中，我们将详细讨论这一问题。

除了软件的正确性之外，在更广泛的意义下，软件的可靠性还应包括安全性和健壮性。

所谓安全性是指对于软件系统的合理输入，系统能给出正确的结果；而对于用户有意或无意的不合理输入，系统应能拒绝这种输入，并视不同情况给出不同提示。如对识别用户身份的非正常输入，应给出非法用户的警告，而大数多场合下的非正常输入，系统应提醒用户注意或提示用户怎样才能正确输入。若对于不合理的输入，系统"束手无策"，甚至导致失败或系统被破坏，这种不安全的系统是不能投入使用的。

健壮性是指软件系统对环境的适应性。当软件系统所处的环境发生变化时，例如存贮量不够，硬件故障等意外事件发生时，系统都能按照某种预定方式作适当处理，有效地控制事故的蔓延，不致丢失重要信息，从而避免灾难性的后果。

对军事、金融、以及许多这类大型应用系统，其安全性、健壮性是至关重要的。

1.3.2 软件生产率

计算机的广泛应用使得软件的需求量急剧上升，世界各国普遍感到软件人力资源的缺乏，这种趋势仍在继续发展。但是，目前的软件生产基本上还处于低效率的手工生产方式，而生产出来的软件质量又很不理想，不适应市场软件的大量需求。如何以较少的投资，获得"高产优质"的软件，这是软件工程最主要的研究目标之一。其途径有二，即良好的软件工程开发方法和方便有效的软件设计工具与环境。

近二十多年来，软件工作者已研究发展了多种软件开发方法，这些方法的适用范围是各不相同的。有的方法适用于数据处理系统，而有的方法适用于实时控制系统；各种方法的风格也迥然不同，有的方法仅仅是一组指导性的原则，而有的方法则有较具体的设计原则；有的方法建立在严密的数学基础之上，而有的方法则是实际经验的总结，因而缺乏一种适应多领域又便于实施的软件开发方法。80 年代中期提出的面向对象的软件开发方法有可能成为满足上述目标的新的软件开发方法，这正是我们撰写 OOSE 一书的目的。

方法和工具、环境之间有着密切的联系，相辅相成。在软件开发和维护的不同阶段，都应该有这样或那样的软件工具和环境，用以帮助开发和维护人员自动地完成许多数据分析和处理工作。这样的工具环境愈完善，软件的设计和维护也就愈方便。这是当前提高软件生产率的一个重要方面。人们希望研究出一套系统的、成龙配套的软件开发工具，构成协调的软件开发环境，从而为软件人员提供一个能覆盖整个软件生存期的良好的工作环境，这样可使软件生产率大为提高。因此，CASE(Computer-Aided Software Engineering)也是软件工程研究的重要课题。现已有许多商品化的 CASE 工具，但为软件人员提供一个能覆盖整个软件生存期、又适用于多种应用领域的良好工作环境的 CASE 尚不多见，要达到上述目标还有许多理论和技术问题有待解决。

1.3.3 软件再应用

软件再应用(或称重用)也是提高软件生产率、降低软件成本的一个重要方面。当人们一次又一次地去设计一些互相雷同的程序时，很多劳动花费在一些基础性的重复工作上，软件再应用的技术旨在减少这种重复。

可再应用的软件单位可以是软件逻辑结构，如软件系统的逻辑框架、软件模块的详细设计表示，也可以是程序代码，如子程序等。

软件再应用方式大体上可分为两种：最简单易行的一种方式是直接组合可再应用的软件单位，这当然会有许多使用上的限制，最成熟、且用得最多的是各种程序库；另一种方式是按模式再应用。有下面几种情况。

(1) 面向问题编程　用面向问题的语言，或非常高级程序语言(Very High-Level Languages)编写程序，这样的程序实质上是问题的一种高级描述(Specifications)，再由此出发，可以得到多种不同语言的机器实现。这种重用技术适用于在不同支持环境上求解同类问题的用户。

(2) 程序变换　可以设计一种源码变换系统，使之能把一种语言的源程序变换成另一种语言的程序。例如，FORTRAN、C 语言有其丰富的应用程序库，只要设计成功从 FORTRAN 语言(或 C 语言)到另一种语言，比如说 Pascal 语言的变换系统，就可立即得到 Pascal 语言的同样丰富的程序库。变换程序的功能完全类似于传统编译程序，只是源语言和目标语言都可能是某种高级语言，这种重用技术是获得资源共享的有效方法。

(3) 程序系统生成　许多程序系统虽然表面上看来很不相同，但它们的基本结构模式是一样的。例如，面向语言的结构化编辑器，不同语言除了语法结构可能不同之外，编辑器其他部分的功能基本上是相同的；另外，高级程序设计语言编译中的语法分析和代码生成也是如此。这些事例表明可以设计出一个关于一定结构模式的系统，给出一些特定的具体说明和要求，这样的系统就能产生出特定的软件来。通常所说的许多程序自动生成系统，就是这种情形的软件再应用的例子。

传统软件设计法和语言缺乏软件重用技术的实现机制，面向对象设计技术和面向对象语言所提供的封装部件和继承性技术为软件重用提供了强有力支持。

与软件重用密切相关的另外两个方面是软件移植和软件规范化，这也是近几年非常受人们重视的研究领域。

1.3.4　软件维护

尽管不少软件人员对软件维护持有不同的看法，但修改与完善已投入运行了的软件是十分必要的。软件的生命力往往取决于对软件的完善性维护，越是通用的软件，维护费用越高，因此，大的软件生产公司都有很强的维护班子。软件维护的内容通常包括完善性维护、正确性维护和适应性维护，即软件功能的修改或增强、故障的排除、性能的提高或环境的变化等都要对软件作大量修改或扩充。除维护本身要花费很大的人力物力外，维护后可能由于副作用而产生新问题，这又要花费精力去解决。因此，如何减少维护的总工作量，也是软件工程面临和要解决的又一主要问题。面向对象设计技术中的"对象"具有独立性，一对象仅通过消息与其他对象联系，这不仅使软件维护容易局部化，也使维护的副作用减少到最低限度，这将大大减少软件维护的总工作量。

以上是软件工程所面临的几个重要问题。可以说，这些问题均未获得完满地解决，面 OOSE 可能是解决以上问题最有希望的方法。