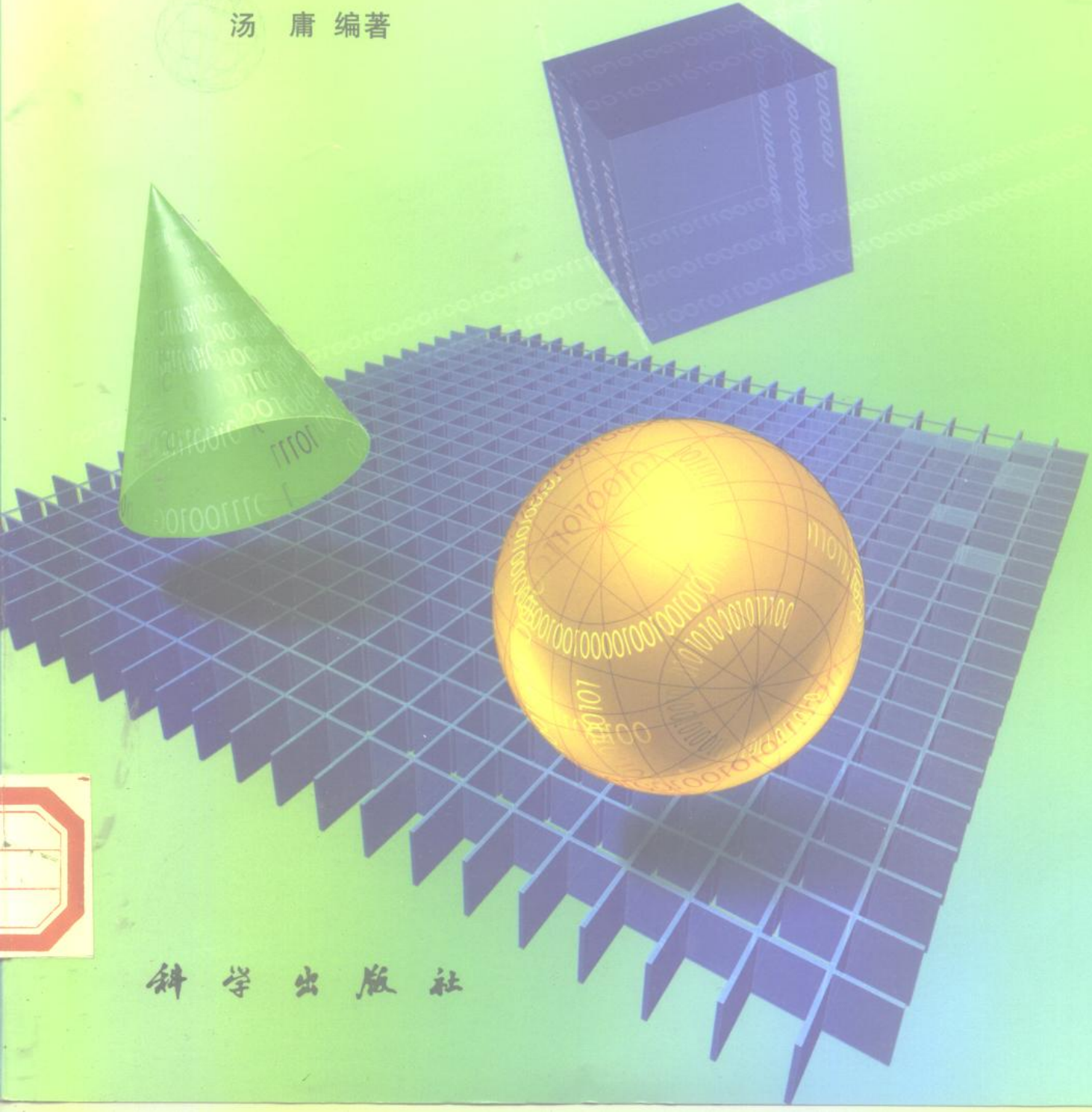
结构化与面向对象软件方法

Software Methodologies: Structure and Object-Oriented

汤 庸 编著



科学出版社

结构化与面向对象软件方法

汤 庸 编著

科学出版社
1998

内 容 介 绍

结构化与面向对象方法是当前两种最具代表性的软件工程与方法学。本书系统地介绍结构化和面向对象方法的基本概念、基础理论和主要技术。全书分八章。第一章介绍软件方法学的形成及发展、软件危机、软件生存周期、软件开发模型及方法。第二章到第四章介绍结构化程序、程序形式化推导及变换技术、结构化分析与设计方法、程序测试与正确性证明。第五章至第七章介绍面向对象基本概念、基本特征，面向对象分析与设计方法、面向对象程序设计等。最后一章讨论软件质量模型和软件维护。

本书适合于计算机研究和软件开发人员阅读，也可用作高等院校高年级学生和研究生教材或参考书。

图书在版编目（CIP）数据

结构化与面向对象软件方法/汤庸编著. -北京：科学出版社，1998.8
ISBN 7-03-006846-7

Ⅰ.结… Ⅱ.汤… Ⅲ.①结构化程序设计②面向对象语言-程序设计
Ⅳ.TP311.11

中国版本图书馆 CIP 数据核字（98）第 18283 号

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
中国科学院印刷厂印刷
新华书店北京发行所发行　　各地新华书店经售
*
1998 年 9 月第 一 版　　开本：787×1092　1/16
1998 年 9 月第一次印刷　　印张：13 1/2
印数：1-4000　　字数：305，000

定价：20.00 元

前　言

结构化方法与面向对象方法是当前最为重要的两种软件方法学。结构化是软件开发与维护的最基础方法，其它现代软件方法学都是在结构化方法基础上发展和演绎而来，而且遵循基本的结构化思想；面向对象方法是近十年来最为流行的软件方法，被称为当代的“结构化方法”。可以这样说，当前从事软件开发、应用和管理的人员都应该了解结构化和面向对象方法。本书介绍软件工程与方法学的基本概念和一般方法，较系统地讨论了结构化和面向对象软件开发的基本概念、理论、方法与技术。

全书共八章。第一章介绍软件工程与方法学的形成与发展；提出软件生存周期五个基本时期的划分，并讨论了软件生存周期各个时期的基本任务；概述当前主要软件开发模型和开发方法。第二章到第四章讨论结构化方法的基本理论和方法：介绍结构化程序、结构化定理、程序形式化推导及变换技术、结构化分析与设计方法、模块化设计方法与模块优化准则；讨论程序测试与正确性证明技术。第五章至第七章介绍面向对象基本概念和基本特征：介绍一种面向对象分析与设计方法，讨论面向对象程序设计基本思想和面向对象程序设计语言的一般特征；并以两种最具代表性的面向对象程序设计语言 Smalltalk 和 C++ 为例，介绍面向对象编程的基本概念和方法。软件方法学研究的目的就是为了提高软件的质量、增强软件的可靠性和可维护性、降低软件开发与维护成本，因此，本书最后在第八章讨论了软件质量模型和软件维护的一般方法和概念。

本书结合了传统的“软件工程”、“程序设计方法学”和还处于发展阶段的“面向对象分析与设计”、“面向对象程序设计”等理论与方法，试图从软件方法学角度系统介绍结构化和面向对象软件方法的基本概念和思想。本书是在作者编著的《软件工程与方法学》研究生讲义的基础上完成的，书中主要取材于国内外已发表的经典著作和最新文献，也有作者提出的新观点与方法，如软件生存周期的五个阶段划分、结构化程序的封闭结构概念、面向对象软件质量与维护课题等，不足和错误之处恳请读者批评指正。

本书注重基础理论和基本概念，也强调应用技术与方法，适合于软件开发、应用与管理的人员阅读参考，也可用作高等院校高年级学生和研究生的教材和参考书。

本书在编写过程中得到了作者所在课题组和研究室的同事和研究生的大力支持，参考和引用了大量文献资料，还得到了广东省自然科学基金和广东省重点学科的资助，科学出版社徐一帆老师认真地阅读和校改了书稿，在此一并表示衷心的感谢。

汤　庸

1998 年 6 月

Software Methodologies: Structure and Object-Oriented

TangYong
(Department of Computer Science and Engineering, GuangDong University of Technology)

Summary

Structure and Object-Oriented both are the most typical and most popular software methodologies. This book is an introduction to the principles and basic techniques of structure and Object-Oriented Software methodologies. Firstly, this book briefly introduces software engineering, programming methodology and software life circle. Chapter 2, 3 and 4 discuss the structure program concepts and constructing, structure analysis and design, program verification and testing. Chapter 5,6 and 7 discuss the Object-Oriented concepts, Object-Oriented analysis and design, Object-Oriented implementation and Object-Oriented programming languages. Finally, chapter 8 discusses software quality models and maintenance methods. In this book, the author presents some new ideas, e.g. the five periods of software life circle, close-structure concept, Object-Oriented software quality and maintenance.

目　录

第一章　绪　论.....**1**

1.1　形成与发展.....1

1.1.1 程序设计方法学.....1

1.1.2 软件工程.....2

1.1.3 软件方法学.....2

1.1.4 结构化方法.....3

1.1.5 面向对象方法.....4

1.2　软件生存周期.....5

1.2.1 五个软件时期的划分.....6

1.2.2 软件定义与计划.....7

1.2.3 软件分析.....10

1.2.4 软件设计.....11

1.2.5 软件实现.....11

1.2.6 软件运行与维护.....15

1.3　软件开发过程模型.....15

1.3.1 瀑布模型.....16

1.3.2 原型模型.....16

1.3.3 其它模型.....17

第二章　结构化程序及构造方法.....**18**

2.1　结构化程序.....18

2.1.1 结构化程序设计.....18

2.1.2 关于 GOTO.....18

2.1.3 控制结构.....19

2.1.4 正规程序.....20

2.1.5 基本程序.....22

2.1.6 结构化程序.....23

2.2　结构化定理.....23

2.2.1 程序函数.....23

2.2.2 结构化定理.....24

2.2.3 非结构化程序到结构化程序转换.....26

2.3　逐步求精方法.....29

2.3.1 什么是逐步求精.....29

2.3.2 逐步求精技术 30
2.4 程序形式推导技术 32
2.4.1 语句的HOARE表示 32
2.4.2 谓词变换器 32
2.4.3 程序的形式语义 34
2.4.4 面向目标的推导 37
2.4.5 不变式推导 40
2.5 程序变换技术 44
2.5.1 程序变换的基本思想 44
2.5.2 程序变换的基本规则 45
2.5.3 程序的生成方法 47
2.5.4 递归消去法 48
第三章 结构化分析与设计 51
3.1 结构化分析与设计的一般步骤 51
3.2 数据流分析技术 52
3.2.1 数据流分析 52
3.2.2 数据流图 54
3.2.3 数据字典 55
3.3 逻辑分析工具 58
3.3.1 结构化语言 58
3.3.2 判定树 59
3.3.3 判定表 60
3.4 结构化设计的图表工具 61
3.4.1 IPO 图 61
3.4.2 结构图 63
3.4.3 程序流程图 64
3.4.4 盒图 65
3.4.5 PAD 图 66
3.4.6 过程设计语言 PDL 68
3.5 面向数据流的设计 69
3.5.1 变换流与事务流 69
3.5.2 设计步骤 70
3.5.3 变换设计 72
3.5.4 事务设计 73
3.6 面向数据结构的设计 74
3.6.1 Jackson 图 74
3.6.2 Jackson 方法 74

3.6.3 设计实例 75
3.7 模块化技术 78
3.7.1 模块与模块化 78
3.7.2 模块的特征与独立性 79
3.7.3 模块的藕合 80
3.7.4 模块的内聚 82
3.7.5 模块设计一般准则 84
3.7.6 模块的作用域与控制域 86
3.8 应用实践 88
第四章 程序测试与正确性证明 89
4.1 程序正确性定义 89
4.2 HOARE 公理化方法 90
4.2.1 HOARE 基本法则 90
4.2.2 简单语句证明法则 91
4.2.3 循环语句证明法则 93
4.3 FLOYD 方法 95
4.3.1 不变式断言法 95
4.3.2 良序集法 100
4.4 递归程序正确性证明 103
4.4.1 递归与迭代 103
4.4.2 递归的程序模式 104
4.4.3 递归计算规则 105
4.4.4 结构归纳法 106
4.4.5 良序归纳法 107
4.5 测试的基本概念 109
4.5.1 软件测试 109
4.5.2 黑盒测试与白盒测试 109
4.6 测试过程与步骤 111
4.6.1 测试过程 111
4.6.2 单元测试 112
4.6.3 集成测试 113
4.6.4 验收测试 116
4.7 测试方案设计 117
4.7.1 测试方案设计的基本原则 117
4.7.2 逻辑覆盖 117
4.7.3 等价类划分 120
4.7.4 边界值分析 121

4.8 调试 121
4.8.1 静态查找 122
4.8.2 消去法 122
4.8.3 回溯法 123
第五章 面向对象的概念 124
5.1 对象与类 124
5.1.1 对象 125
5.1.2 消息与方法 127
5.1.3 类 129
5.2 基本特征 131
5.2.1 协议与封装 131
5.2.2 继承性 132
5.2.3 多态性与动态联编 135
5.3 软件生存周期与开发模型 136
5.3.1 面向对象的软件生存周期 136
5.3.2 面向对象方法与快速原型技术 137
第六章 面向对象分析与设计 139
6.1 前 言 139
6.1.1 OOA 形成 139
6.1.2 多层次多组成模型 140
6.1.3 OOA 步骤 141
6.2 标识类/对象 145
6.2.1 为什么要标识类/对象 145
6.2.2 如何确定对象 146
6.2.3 实例 147
6.3 标识结构 149
6.3.1 什么是结构 149
6.3.2 为什么要定义结构 149
6.3.3 如何定义结构 149
6.4 标识主题 150
6.4.1 什么是主题 150
6.4.2 为什么要引入主题 151
6.4.3 如何定义主题 151
6.5 定义属性 152
6.5.1 为什么要定义属性 152
6.5.2 如何定义属性 152

6.6 定义服务 157
6.6.1 什么是服务 157
6.6.2 如何定义服务 157
6.6.3 表示所需要的服务 158
6.6.4 标识消息连接 158
6.6.5 说明服务 159
6.7 面向对象设计 160
6.7.1 转向面向对象的设计 160
6.7.2 OOD 准则与步骤 160
6.7.3 汇集 OOA 文档资料 161
第七章 面向对象程序设计 162
7.1 前 言 162
7.1.1 OOD 与实现语言 162
7.1.2 面向对象程序设计语言 163
7.1.3 面向对象程序的特点 164
7.2 Smalltalk 面向对象编程 165
7.2.1 对象、类、实例及继承性 165
7.2.2 变量名与文字句法 166
7.2.3 消息表达式 167
7.2.4 块表达式 168
7.2.5 基本控制结构 168
7.2.6 方法 170
7.3 C++面向对象编程 172
7.3.1 C++面向对象概念 172
7.3.2 C++类 172
7.3.3 成员函数 173
7.3.4 对象 175
7.3.5 构造函数与析构函数 176
7.3.6 继承性与派生类 177
7.3.7 多态性与虚拟函数 180
7.3.8 友元 182
7.3.9 一个例子 183
第八章 软件质量与维护 185
8.1 软件质量 185
8.1.1 软件质量标准 185
8.1.2 软件质量保证 186

8.2 软件质量度量模型 187
8.2.1 Boehm 模型 187
8.2.2 McCall 模型 187
8.2.3 ISO 建议模型 188
8.2.4 软件质量因素 188
8.3 软件质量评价方法 192
8.3.1 McCabe 软件复杂性度量 192
8.3.2 Halstad 软件复杂性度量方法 194
8.3.3 软件可靠性度量方法 195
8.4 软件维护的概念 196
8.4.1 软件维护的类型 197
8.4.2 软件的可维护性 197
8.4.3 软件维护工作量模型 198
8.4.4 软件维护的典型问题 198
8.4.5 维护的代价与副作用 199
8.5 维护过程与方法 200
8.5.1 维护组织 200
8.5.2 维护模型 201
8.5.3 维护报告 202
8.5.4 维护的记录与评价 202
8.6 面向对象软件质量与维护 203
8.6.1 软件质量保证和可维护性 203
8.6.2 面向对象软件质量与维护的新课题 204
参考文献 205

第一章 绪 论

软件方法学是研究软件工程理论、技术与方法，指导软件开发的一门计算机学科，采用良好的软件开发方法是成功开发软件的根本保障。本章介绍软件方法学的形成和发展；主要介绍软件危机、软件生存周期等基本概念，介绍软件工程、程序设计方法学产生和发展，讨论软件生存周期的五个基本时期的主要任务，介绍软件开发基本模型和方法。

1.1 形成与发展

电子计算机自 1946 年诞生后到 60 年代中期计算机发展的早期阶段，计算机系统还是以硬件为主，软件费用是总费用的 20%左右。到了计算机中期（60 年代中期到 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。而到 1985 年以后，软件费用已上升到今天 80%以上。软件相对硬件的费用比例还在不断提高。

事实上，60 年代中期，随着计算机技术迅速发展和应用领域迅速拓宽，软件需求迅速增长，软件数量急剧膨胀，软件系统空前庞大与复杂。而当时的程序设计与软件开发技术却远远落后于这种发展。人们没有认识到从宏观上对程序设计方法进行研究的重要性，许多人只满足于写出可以运行的程序，而“不拘一格”或一味炫耀“编程技巧”。因此，许多大型软件常常质量低劣、可靠性不高、可维护性差，却又价格昂贵、供不应求。这种情况严重阻碍了计算机和计算机应用的发展，这一系列严重的问题就是所谓的“软件危机”（Software Crisis）。

软件危机出现了，如何进行软件开发呢？为了解决这个问题，程序设计方法学和软件工程学就逐渐形成了。

1.1.1 程序设计方法学

1967 年 Floyd 提出的断言方法证明流程图程序的正确性；1968 年 Dijkstra 的“GOTO”有害论；1969 年 Hoare 在 Folyd 断言法基础上提出的程序公理方法；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。

程序设计方法学是运用数学方法研究程序的性质以及程序设计的理论和方法的一门

学科。程序设计方法学经典内容主要包括结构化程序理论、程序的正确性证明、程序形式推导、程序变换技术等。

1.1.2 软件工程

“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。

软件工程学是主要应用工程的方法和技术，研究软件开发与维护的方法、工具和管理的一门计算机科学与工程学交叉的学科。

软件工程的基本出发点是以软件生命周期为基础，吸取工程的方法和技术，将软件开发和维护过程规范化、科学化。传统的软件工程技术主要以结构化思想为基础。

1.1.3 软件方法学

在初期，程序设计方法学和软件工程是从两种不同的角度和应用不同的方法研究软件开发技术的两种紧密相关、相辅相成又各有侧重的学科。前者是以数学理论为基础的理论性学科；后者是以工程方法为基础的工程学科。

软件工程学和程序设计方法学都是研究软件开发和程序设计的学科，它们的研究对象、研究内容、出发点和目标都是一致的。它们的根本目标是以较低的成本开发高质量的软件和程序。主要目的包括：提高软件的质量与可靠性；提高软件的可维护性；提高软件生产率，降低软件开发成本等。

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，主要差异有：

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。这两条研究途径的界限逐渐地模糊化、一体化了。

一方面，程序设计方法学研究已发生了较大的变化，逐渐从“纯粹的程序”正确性证明等较老的课题转向“软件”的结构化、正确性、可靠性及软件设计方法方面的研究。例如，现在程序设计方法学及软件工程学都将面向对象的方法做为其重要的新的研究方向，“程序设计方法学”正逐渐发展成“软件设计方法学”。

另一方面，软件工程从一开始就是以程序设计方法学为基础的一门工程学科，而且还在不断吸收程序设计方法学和计算机科学理论新成果和新技术。从某种意义上，可以说，软件工程学实际上就是“应用设计方法学”。

所以实际上“软件工程学”或“程序设计方法学”术语都已难以准确表达它们的研究内涵和含义了。也许采用“软件工程与方法学”或“软件方法学”更能概括当前软件工程学和方法学研究的内涵。

可以这样描述：软件工程与方法学是指应用计算机科学理论和工程方法相结合的研究方法，研究软件生存周期一切活动（包括软件定义、分析、设计、编码、测试与正确性证明、维护与评价等）的方法、工具和管理的学科。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。

现在已有一些较成熟的软件开发方法，例如：适用于实时事务处理系统的有限状态机方法（FSM）；适用于并发软件系统的 PETRI 网方法，以数学概念和理论为基础的形式化方法（流行有 SDC 公司的形式化开发方法：Formal Development Methodology：FDM；IBM 公司的维也纳开发方法：Vienna Development Method：VDM）等。但是，目前最重要的的软件方法是结构化方法和面向对象方法。结构化方法是最基础的方法，是其它软件技术的基础。本书主要介绍结构化和面向对象软件方法的基本概念、基本理论、主要方法和主要技术。

1.1.4 结构化方法

结构化方法（Structure Method）是最早的、最传统的软件开发方法。60 年代初，就提出了用于编写程序的结构化程序设计方法，而后发展到用于设计的结构化设计（SD）方法、用于分析结构化分析（SA）方法；以及结构化分析与设计技术（SADT）等；面向数据结构的 JACKSON 方法，WARNIER 方法等。

常见结构化技术如下：

1）Yourdon 方法，即通常使用的结构化分析与结构化设计（合称结构化分析与设计方法），它适用于一般数据处理系统，是一种较流行的软件开发方法。在实际软件开发中使用的许多方法都是基于结构化分析与设计的改进方法。

2）JACKSON 方法也是一种适用于一般数据处理系统的结构化方法。

3）WARNIER 方法，又称逻辑构造程序的方法，简称 LCP，也是一种面向数据结构的方法。

4）SADT（Structure Analysis and Design Technique）是 D.T.Ross 于 1973 年提出来的，后来经过美国 Seffech 公司的改进。SADT 以模块图式表示系统构成、系统设计方案，适合于分析和设计大型复杂系统。其基础是自顶向下、模块化、层次化等结构化思想。

结构化方法的基本思想可以概括为：自顶向下、逐步求精；采用模块化技术、分而治之的方法，将系统按功能分解成若干模块；模块内部由顺序、分支、循环基本控制结构组成；应用子程序实现模块化。

结构化方法强调功能抽象和模块性，将问题求解看作是一个处理过程。结构化方法

由于采用了模块分解和功能抽象，自顶向下、分而治之的手段，从而可以有效将一个较复杂的系统分成若干易于控制和处理的子系统、子系统又可以分解成更小的子任务，最后的子任务都可以独立编写成子程序模块。这些模块功能相对独立、接口简明、界面清晰，使用和维护起来非常方便。所以，结构化方法是一种非常有用的软件方法，也是其它软件方法学的基础。

但是，由于结构化方法将过程和数据分离为相互独立的实体，程序员在编程时必须时刻要考虑所要处理的数据的格式。对于不同的数据格式做同样的处理或对于相同的数据格式做不同的处理都需要编写不同的程序，所以结构化程序的可重用性不好。另一方面，当数据与过程相互独立时，总存在错误的数据调用正确的程序模块或用正确的数据调用错误的程序模块的可能性。因此，要使数据与程序始终保持相容，已成为程序员一个沉重的负担。以上这些问题，用面向对象方法就可以得到很好地解决。

1.1.5 面向对象方法

面向对象方法（Objected -Oriented）是当前软件方法学的主要方向，也是目前最有效、最实用和流行的软件开发方法之一。

面向对象（OO）的概念和思想却由来已久。有人认为，可以将 Dahl 与 Nygard 在 1967 年推出的程序设计语言 Simula-67 作为面向对象的诞生标志。Simula-67 首先在程序中引入了对象概念。但是，面向对象真正的第一个里程碑应该是 1980 年 Smalltalk-80 的出现。smalltalk-80 发展了 Simula-67 的对象和类的概念，并引入方法、消息、元类及协议等概念，所以有人将 smalltalk-80 称为第一个面向对象语言。但是最后使面向对象广泛流行的则是面向对象的程序设计语言 C++。

面向对象的方法认为：客观世界是由许多各种各样的对象组成，每个对象都有各自的内部状态和运动规律，不同对象之间的相互作用和联系就构成了各种各样不同的系统，构成了我们所面对的客观世界。

面向对象吸取了结构化的基本思想和主要优点。面向对象方法将数据与操作放在一起，作为一个相互依存、不可分割的整体来处理。面向对象综合了功能抽象和数据抽象，采用数据抽象和信息隐蔽技术，将问题求解看作是一个分类演绎过程。与结构化方法相比，面向对象更接近人们的认识事物和解决问题的过程和思维方法。

早在 1982 年，Rentsch 就曾预言“80 年代的面向对象程序设计就象 70 年代的结构化程序设计一样，每个人都喜欢用它，每个软件商都开发他们的软件支持它，每个管理员都要付出代价应用它，每个程序员都要以不同的方式实践它，但是没有人能清楚地讲清楚它”。事实已经证明，80 年代面向对象的研究热潮比 70 年代结构化研究热潮有过之而无不及，所以有人称面向对象是“80 年代的结构化”。到 90 年代的今天，面向对象的方法和技术已经真正到达了 Rentsch 所预言的那样一种应用情景，而且，人们正在力图较清楚地描述“面向对象到底是什么”。

面向对象与结构化在概念上主要区别如下：

（1）模块与对象

结构化方法中模块是对功能的抽象，每个模块是一个处理单位，它有输入和输出。

而面向对象方法的对象也具有模块性，但它是包括数据和操作的整体，是对数据和功能的抽象和统一。所以，可以说对象包含了模块的概念。

（2）过程调用与消息传递

在结构化程序设计中，过程是一个独立实体，显式地为其使用者所见。而在面向对象程序设计中，方法是隶属于对象的，是对象的功能的体现，不能独立存在的实体。消息传递机制很自然地与分布式并行程序、多机系统和网络通讯模型取得一致。在结构化设计中，同一个实参的调用，其结果是相同的，例如，设 max(x,y)是求 x 和 y 最大值的过程，那么，无论什么时候调用 max(60,100)，其结果都是 100。但在面向对象中的消息传递则不同，同一消息的多次传递可能产生不同的结果。例如，设消息 go: aPoint 一个从发送点到点 aPoint 的一条直线，那么，消息 go: {50,100}发给发送点{50,50}的结果是画一条从坐标点{50，50}到坐标点{50，100}的竖线，而发给发送点{10，100}的结果则是画一条从坐标点{10，100}到坐标点{50，100}的横线。

（3）类型与类

类型与类都是对数据和操作的抽象，即定义了一组具有共同特征的数据和可以作用于其上的一组操作。但是，类型仍然是偏重于操作抽象，类则集成了数据抽象和操作抽象，二者缺一不可。此外，类引入了继承机制，实现了可扩充性。

（4）静态连接与动态连接。

在面向对象系统中，通过消息的激活机制，把对象之间的动态联系连接在一起，使整个机体运转起来，实现系统的动态连接。

相对传统的结构化方法来说，面向对象方法具有更多的优势。当然，面向对象并不是十全十美和唯一的软件方法，结构化思想和方法是基础，面向对象是在吸取结构化思想和优点的基础上发展起来的，是对结构化方法的进一步发展和扩充。所以，在实际软件开发中，常常需要综合应用结构化思想和面向对象方法。

本书主要介绍结构化技术和面向对象方法。第二章至第四章介绍结构化方法的基本概念、基本理论，介绍结构化软件分析与设计基本方法。第五章至第七章介绍面向对象方法的基本概念和软件分析与设计方法，并介绍面向对象的程序设计基本模式。

1.2 软件生存周期

软件生存周期（Software Life Cycle）是软件工程与方法学最基础的概念。软件工程的方法、工具和管理都是以软件生命周期为基础的活动。软件工程强调的是使用软件生存周期方法学和使用成熟的技术和方法来开发软件。本书介绍的结构化与面向对象方法都是以软件生存周期为基本特征的软件开发方法。

软件生存周期的基本思想是：任何一个软件都是提出开始、通过开发、交付使用、到最终被淘汰为止，有一个存在期。软件生存周期的概念并不是说软件同硬件一样，存在“被用坏”和“老化”问题，而是指其有无存在价值。

1.2.1 五个软件时期的划分

在人类生命周期内划分成若干阶段（如幼年、少年、青年、中年、老年等），类似地，软件生存周期也可以划分成若干阶段，每个阶段有较明显的特征，有相对独立的任务，有其特定的方法和工具。

软件规模、种类、开发方式、开发环境与工具、开发使用的模型和方法都影响软件生命周期阶段的划分。软件生命周期阶段的划分应遵循一条基本原则，即：要使每个阶段的任务尽可能相对独立，同一阶段各项任务的性质应尽可能相同。这样降低每个阶段任务的复杂程度，简化不同阶段之间的联系，有利于软件开发的管理。

目前，软件生存周期的阶段划分有多种方法。一种典型的阶段划分为：问题定义、可行性研究、需求分析、概要设计（总体设计）、详细设计、编码与单元测试、综合测试、维护等八个阶段。但是这种软件生存周期的划分只适合于早期“理想的”软件工程项目，在实际软件工程项目中较难操作。

我们提出活动时期的软件生存周期划分思想，将软件生存周期划分成：软件定义与计划时期、软件分析时期、软件设计时期、软件实现时期、软件运行与维护时期等五个大的时期。其中软件分析、软件设计和软件实现就是通常所说的“软件开发”（图 1.1）。

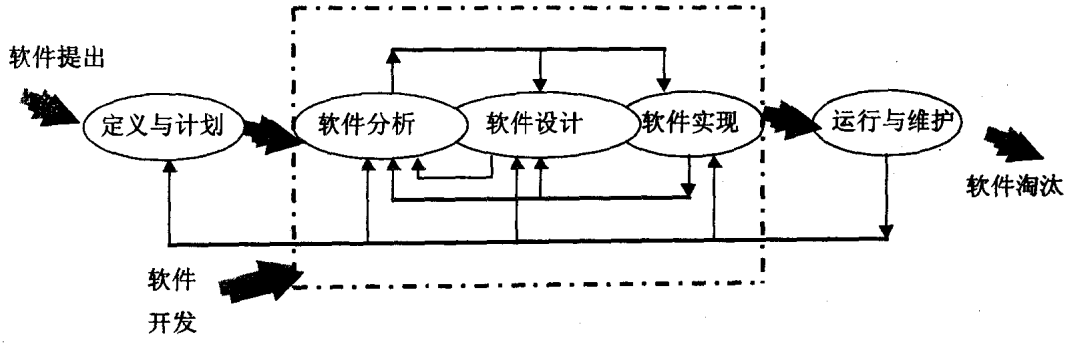


图 1.1 软件生存周期的时期划分

1）软件定义与计划，简称软件计划。这个时期的主要任务是确定软件的目标、规模和基本任务，论证项目的可行性，估算软件成本和经费预算，制定软件开发计划和进度表等。软件定义评审通过后，软件项目才真正立项，才能进入软件开发阶段。

2）软件分析时期的主要任务是需求分析，确定软件的具体功能与性能要求，通常也称为需求分析。

3）软件设计时期的目标是设计软件的结构、算法和实现方案，为软件编码提供设计依据和算法。

4）软件实现时期就是用某种（些）编程语言或工具将软件设计方案实现成软件产品，交付用户使用。这个时期的基本任务包括程序实现和测试两个方面，所以也称为软件编码与测试。

5）软件运行与维护时期是软件生存周期的最后一个时期，也是最长的时期，包含