工业和信息化部“十二五”规划教材

高等学校规划教材

面向对象技术与系统建模

◎孙玉山　徐汉川　主编
◎梁永先　周丽娜　朱东杰　副主编
◎王宇颖　主审

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校规划教材
工业和信息化部“十二五”规划教材

面向对象技术与系统建模

孙玉山　徐汉川　主编
梁永先　周丽娜　朱东杰　副主编
王宇颖　主审

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

内 容 简 介

本书阐述编程技术的结构化分析、设计的基本步骤与面向对象理论的发展，重点讲述面向对象的基本理论、面向对象设计原则，以及使用面向对象技术进行分析、设计、实现的完整软件开发流程。本书在面向对象系统分析与设计中重点强调软件作为系统与面向对象系统建模的概念。在系统分析、系统设计与系统实现中，使用统一建模语言 UML 建立文档。本书力求包括软件工程与面向对象研究的最新进展，如统一过程模型等。

书中大多数设计例子都使用纯面向对象语言 Java 实现，每章都有练习题，大多数章还提供课下设计——编程作业，部分 Java 源代码可以登录华信教育资源网（www. hxedu. com. cn）下载或者直接向作者（电子邮箱 mikesun725@ aliyun. com）索取。

本书适用对象为研究生，也可以在精选内容的基础上，用于高年级本科生教学，还可以作为软件工程师的学习参考书。

图书在版编目（CIP）数据

面向对象技术与系统建模/孙玉山，徐汉川主编．—北京：电子工业出版社，2015. 9
ISBN 978-7-121-26995-0

Ⅰ．①面… Ⅱ．①孙… ②徐… Ⅲ．①面向对象语言－程序设计－高等学校－教材 Ⅳ．①TP312

中国版本图书馆 CIP 数据核字（2015）第 195646 号

策划编辑：章海涛
责任编辑：冉 哲 特约编辑：李春雷
印 刷：北京中新伟业印刷有限公司
装 订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×1092 1/16 印张：20. 25 字数：518 千字
版 次：2015 年 9 月第 1 版
印 次：2015 年 9 月第 1 次印刷
定 价：52. 00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@ phei. com. cn，盗版侵权举报请发邮件至 dbqq@ phei. com. cn。

服务热线：（010）88258888。

前　言

面向对象分析与设计概念、理论及技术是在20世纪60年代末提出的并且经历了70年代至90年代的逐步发展与完善的过程。由于该技术已经成为当今软件开发的主流技术，在进入21世纪至今的10余年中，人们对面向对象技术的研究仍然在继续。在20世纪90年代产生的统一建模语言对面向对象分析与设计提供了建模工具方面的有力支持，从而使得面向对象软件开发具有了统一的规范化的文档，也使得面向对象软件设计与开发的生产率得到了很大的提高。

面向对象分析与设计方面的课程是计算机科学与软件工程专业本科生及研究生的最重要的专业基础课程之一。然而，虽然有大量的面向对象分析与设计方面的专著出版，但是要进行有效的面向对象分析与设计的教学仍然是比较困难的。原因是：① 面向对象概念本身就比较难以理解；② 缺乏合适的既有较深入的理论阐述，又有生动实例与练习题的教材。

本教材主要面向大学计算机与软件工程相关专业高年级本科生或者研究生。针对面向对象分析与设计的理论较为抽象，而在校学生往往无实践经验的特点，本教材特别注意将抽象的面向对象理论与实例结合起来，通过精心设计的实例，引导学生掌握课程相关内容，这正是本教材的特色所在。

本教材共分为8章。

第1章从程序设计发展历史的角度，讲述了非结构化设计、结构化设计和面向对象设计的发展阶段与本质区别。

第2~4章为面向对象设计的基础理论部分。

第5章讲述面向对象分析与设计应该遵循的原则。

第6章讲述统一建模语言UML，重点给出了作为面向对象开发工具的统一建模语言在实际软件开发建模方面的应用。

第7章重点讲述统一过程开发模型中重要的需求分析、领域建模与对象设计技术。需求分析与用例建模技术是面向对象软件开发中的重要环节。创建良好的用例模型与领域模型可以为后续的系统分析、系统设计、系统实现和系统测试与系统维护等环节打下良好的基础。

第8章通过一个较为完整的面向对象分析与设计案例，说明从系统的需求分析、领域建模、系统分析、系统设计到系统实现各个阶段所需要的开发技术与产生的文档。教师可以让学生在课程结束的时候，以第8章的项目例子为基础，完成一个完整的软件项目的分析、设计以及编程的全过程（使用Java语言编程）。

本书主要特点如下：

（1）对面向对象技术与系统建模的理论与技术做深入、详尽的阐述。

（2）全书贯彻“教你怎样做”的原则。书中给出了大量的设计实例，每个实例都试图解释面向对象的基本与重要的概念及设计技巧。

（3）为了培养学生的动手能力，在第8章中以一个软件设计项目——宾馆预订与入住系统的例子贯穿始终，讲述在统一过程模型中的一次迭代所要进行的具体软件开发活动与产

生的文档。

（4）为了体现面向对象分析与设计的特点，大多数设计例子都使用纯面向对象语言 Java 实现，大部分实例都给出了 Java 源代码，可以登录华信教育资源网（www. hxedu. com. cn）下载或者直接向作者（电子邮箱 mikesun725@ aliyun. com）索取。

（5）全书精心设计了许多趣味性较强的练习题与课下设计——编程作业。这些实例、练习题与课下设计——编程作业正是本书的精髓。如果没有这些实战性的练习，大量晦涩难懂的抽象概念会使本课程的教学效果大打折扣。

（6）书中的内容力求反映最近的软件分析和设计理论与技术。

建议按 28 学时或者 36 学时讲授这门课程，课程应该至少覆盖第 1 ~ 6 章。如果学时紧张，第 1 章可以不讲，第 7 章可以略讲，第 8 章作为面向对象项目开发实例可以略讲，然后让学生自己阅读。建议本课程的课下设计——编程作业（或者完成一个简单的软件项目）部分占总成绩的 40% 或者 50%，期末考试部分占 60% 或者 50%，并进行开卷考试。

本书作者之一孙玉山在 1993 年任哈尔滨工业大学（简称哈工大）数学系副教授，1995 年赴美留学。获得计算机科学专业学位以后，在美国的 IBM 公司担任软件工程师的两年时间中，获得了一些商业软件设计与开发的实际经验。2002 年夏天回国以后，在哈工大软件学院开设 Java 语言、软件工程、面向对象设计与 UML 及软件体系结构等课程，现任教授。

本书是作者对近 10 年来相关教学经验的凝练，可作为高年级本科生或者研究生的教材使用。作者诚挚希望，本书不但能够对相关课程的教学工作有所帮助，而且还能够为已经在软件公司就职的软件工程师们提供借鉴。总之，作者希望本书成为一本很好的教材，使学习面向对象分析与设计理论和技术不再困难，变成一件比较轻松的事情！对于本书的不足之处，也希望读者能够及时指正，作者将不胜感激。

本书由孙玉山教授策划并编写了其中的大部分章节，包括所有的练习题以及相关的代码实现。徐汉川老师参与编写了大部分章节，独立编写了第 7 章与第 8 章的大部分内容，并对全书进行了修改与校对。梁永先老师参与编写了本书的大部分章节，并且完成了本书第 8 章所涉及项目的全部 Java 源代码的编写与调试。周丽娜老师参与了大部分章节的编写。朱东杰老师除了参与部分章节的编写以外，还负责书中的所有练习题与课下设计——编程作业的源代码的组织、验证、注释工作。王宇颖教授对本书的撰写提出了原则性、架构性的建议，并审阅了书稿，提出了很多宝贵的具体修改建议。张廷斌教授和潘景昌教授对本书的撰写提出了宝贵的建议。在此一并表示感谢。

编　者

于哈尔滨工业大学

目　录

第 1 章　编程语言的发展与程序结构 …… 1
1.1　编程语言发展简史 …… 1
1.1.1　机械计算机时代的“编程” …… 1
1.1.2　编程语言的发展历程 …… 2
1.2　非结构化编程简介 …… 5
1.3　结构化系统分析与设计方法简史 …… 6
1.3.1　结构化分析 …… 6
1.3.2　结构化设计 …… 8
1.3.3　结构化编程 …… 9
1.4　非结构化程序设计与结构化程序设计的区别 …… 11
1.4.1　非结构化程序的特点 …… 11
1.4.2　结构化程序的特点 …… 12
1.5　面向对象编程中对象与类的初步概念 …… 13
1.6　面向对象程序与结构化程序的区别 …… 15
1.7　本章总结 …… 16
1.8　练习题 …… 16
第 2 章　对象模型的基本概念 …… 17
2.1　对象模型基础 …… 17
2.1.1　面向对象系统的基本概念 …… 18
2.1.2　面向对象分析的基本概念 …… 19
2.1.3　面向对象设计的基本概念 …… 19
2.1.4　面向对象编程 …… 21
2.1.5　面向对象分析、设计与编程之间的关系 …… 22
2.2　对象模型元素 …… 22
2.2.1　抽象的概念 …… 23
2.2.2　抽象的例子 …… 25
2.2.3　封装的概念 …… 26
2.2.4　封装的例子 …… 27
2.2.5　模块化的概念 …… 28
2.2.6　Java 语言对模块化的支持 …… 32
2.2.7　抽象的层次化 …… 33
2.2.8　由“is－a”关系所产生的抽象层次的例子 …… 36
2.2.9　由“part of”关系所产生的抽象层次的例子 …… 37

2.3 本章总结 …… 39
2.4 练习题 …… 39
2.5 课下设计——编程作业 …… 41
第3章 对象 …… 42
3.1 对象的基本知识 …… 42
3.1.1 对象的概念 …… 42
3.1.2 对象的状态 …… 43
3.1.3 对象的行为 …… 45
3.1.4 对象操作的类型 …… 46
3.1.5 对象的角色与职责 …… 46
3.1.6 对象就像机器一样 …… 47
3.1.7 对象的标识 …… 50
3.2 对象之间的关系与协作 …… 51
3.2.1 对象之间的链接 …… 51
3.2.2 对象之间职责的分配 …… 53
3.2.3 聚合关系 …… 56
3.3 本章总结 …… 56
3.4 练习题 …… 57
3.5 课下设计——编程作业 …… 59
第4章 类的基本概念 …… 60
4.1 类的基本知识 …… 60
4.1.1 类的定义 …… 60
4.1.2 类的接口与实现 …… 61
4.2 类之间的关系 …… 63
4.2.1 关联关系 …… 64
4.2.2 类的继承关系 …… 66
4.2.3 多态的概念与例子 …… 72
4.2.4 类的聚合关系 …… 75
4.2.5 类的依赖关系 …… 76
4.2.6 关联的表达形式 …… 76
4.2.7 关联的实现方法 …… 78
4.3 类与对象之间的关系 …… 89
4.4 本章总结 …… 91
4.5 练习题 …… 92
4.6 课下设计——编程作业 …… 95
第5章 面向对象设计原则 …… 99
5.1 类的设计应遵循的基本原则 …… 99

5.1.1 面向对象设计中类与模块设计质量的度量 …… 99
5.1.2 操作设计的基本原则 …… 104
5.1.3 类之间关系确定的基本原则 …… 107
5.1.4 类的内部视图 …… 112
5.2 面向对象设计的 SOLID 原则 …… 113
5.2.1 单一职责原则 …… 113
5.2.2 开闭原则 …… 115
5.2.3 里氏代换原则 …… 117
5.2.4 接口隔离原则 …… 118
5.2.5 依赖倒转原则 …… 120
5.3 本章总结 …… 126
5.4 练习题 …… 127
5.5 课下设计——编程作业 …… 129
第 6 章 统一建模语言 UML …… 133
6.1 结构图 …… 135
6.1.1 类图 …… 135
6.1.2 对象图 …… 143
6.1.3 包图 …… 144
6.1.4 组件图 …… 151
6.1.5 部署图 …… 155
6.2 行为图 …… 159
6.2.1 用例图 …… 159
6.2.2 活动图 …… 163
6.2.3 时序图 …… 167
6.2.4 通信图 …… 170
6.2.5 状态机图 …… 172
6.3 本章总结 …… 180
6.4 练习题 …… 180
6.5 课下设计——编程作业 …… 185
第 7 章 需求分析、领域建模与对象设计 …… 186
7.1 统一过程模型 …… 186
7.1.1 统一过程模型的概念 …… 186
7.1.2 统一过程模型的迭代性质 …… 189
7.2 需求分析的概念与方法 …… 190
7.2.1 软件需求的三个不同层次的含义 …… 190
7.2.2 需求的获取与分析 …… 191
7.2.3 需求获取方法 …… 192

7.3 用例图与用例建模 …… 194
7.3.1 用例图 …… 194
7.3.2 用例建模 …… 195
7.4 用例建模实例 …… 198
7.5 领域模型的概念与创建方法 …… 211
7.6 领域模型实例 …… 213
7.7 对象设计与职责分配模式 …… 216
7.7.1 信息专家模式 …… 217
7.7.2 创造者模式 …… 218
7.7.3 低耦合模式 …… 221
7.7.4 高内聚模式 …… 224
7.7.5 控制器模式 …… 225
7.7.6 多态模式 …… 227
7.7.7 纯虚构模式 …… 229
7.7.8 间接模式 …… 232
7.7.9 受保护变化模式 …… 233
7.8 本章总结 …… 235
7.9 练习题 …… 235
第8章 面向对象分析与设计案例——宾馆预订与入住系统 …… 239
8.1 宾馆预订与入住系统需求描述 …… 239
8.1.1 中文描述 …… 239
8.1.2 英语描述 …… 240
8.2 宾馆预订与入住系统的领域模型 …… 241
8.3 宾馆预订与入住软件系统分析 …… 243
8.3.1 宾馆预订与入住软件系统用例建模 …… 243
8.3.2 系统分析的目的 …… 250
8.3.3 系统分析中对象的设计 …… 252
8.3.4 软件体系结构的选择 …… 253
8.3.5 宾馆预订与入住系统的用例分析与实现 …… 256
8.4 宾馆预订与入住系统的设计 …… 268
8.4.1 用户输入/输出设计 …… 269
8.4.2 宾馆预订与入住系统数据库设计 …… 275
8.5 类的详细设计 …… 284
8.5.1 控制类 Controller 的详细设计 …… 284
8.5.2 宾馆类 Hotel 的详细设计 …… 286
8.5.3 订单类 Booking 的详细设计 …… 287
8.5.4 客户类 Customer 的详细设计 …… 288

8.5.5 房间类 Room 的详细设计 …… 289
8.5.6 服务类 Service 的详细设计 …… 289
8.5.7 费用类 Fee 的详细设计 …… 289
8.6 宾馆预订与入住系统的实现 …… 290
8.6.1 实现图 …… 290
8.6.2 实现策略 …… 291
8.6.3 类的实现 …… 291
8.6.4 关联的实现 …… 296
8.6.5 操作的实现 …… 299
8.7 本章总结 …… 308
8.8 练习题 …… 309
8.9 课下设计——编程作业 …… 310
8.10 课程项目：一个面向对象分析与设计的软件项目 …… 311
参考文献 …… 314

第1章

编程语言的发展与程序结构

本章简述计算机编程语言的发展历史，分析在编程语言发展历史阶段中的非结构化、结构化与面向对象编程语言的特点以及所编写的应用程序的典型结构，从而使读者能够清晰地了解面向对象语言与其他编程语言的联系与区别。重要的是，让读者了解面向对象编程继承了结构化编程的某些优点，而又与结构化编程截然不同。与非结构化和结构化编程模式相比，面向对象编程模式产生了革命性的变化。

1.1 编程语言发展简史

本节从计算机及其编程语言发展历史的时间顺序角度出发，简述计算机编程语言的发展历史。其目的是让读者能够通过本节的讲述，粗略地了解各个阶段的编程语言特征，从而对后面章节要讲述的结构化编程、设计和分析与面向对象编程、设计和分析有一个连贯的理解。期望读者能够理解面向对象的概念、编程、设计和分析的出现并不是突然的，而是在结构化编程、设计和分析的基础上发展起来的。然而，其又是与结构化概念根本不同的革命性的概念。

1.1.1 机械计算机时代的“编程”

编程语言有很多定义，其中一个定义为：“编程语言是一种被设计为利用指令与机器（尤其是计算机）通信的人造语言。编程语言可以被用来创建程序，以控制机器的行为或者表示算法”。

按照以上定义，可以认为，最早的编程语言在计算机被发明之前就存在了。例如，在19世纪初，被用来控制、操纵提花织机的穿孔纸带所承载的内容即可理解为一种编程语言。提花织机（Jacquard Looms）是一种机械织机，由约瑟夫·玛丽·雅卡尔（Joseph Marie Jacquard）发明。1801年，首次证实了该织机简化了在纺织品生产过程中复杂图案的生产过程。该织机是由一些排列成连续序列的穿孔卡片所控制的。每张卡片上带有多行穿孔。每个完整的卡片对应（织物）设计的一行。

另外一个例子是在电子计算机出现100年之前出现的机械计算机的设计。由英国数学家Charles Babbage在1837年设计的命名为分析引擎（Analytical Engine）的通用机械计算机与当代电子计算机原理非常类似。该分析引擎包括：① 算术逻辑单元；② 以条件分支和循环形式存在的控制流；③ 集成内存（memory）。该设计成为第一个符合当今意义下的图灵完

整性（Turing complete）通用目的的计算机。该分析引擎本身的设计逻辑非常先进，是电子通用计算机的先驱。

该分析引擎使用穿孔卡给计算机提供程序和数据的输入，该方法与曾被用来控制、操纵提花织机的穿孔纸带的原理是相同的。在输出方面，该分析引擎设计包括了打印机和曲线画图仪。该机器也可以在卡上打孔，代表数字，以便将来读出。该分析引擎采用十进制数计数。

在该分析引擎设计中，包括一个能存储 1000 个数字的“仓库”（store）。算术单元（mill）可以被用来进行四则运算、比较和开平方运算。和当今的 CPU 一样，mill 将依赖它内部的存储过程，以便执行用户程序中复杂的指令。

用户采用的编程语言有点像当代的汇编语言。该语言使用了三种不同类型的打孔卡：第一种用于进行算术运算；第二种是数值常数；第三种用于进行存储运算，将数字存储到算术单元中或者从算术单元中取回数字。该分析引擎使用三种不同的针对不同类型卡片的“读卡器”。

1.1.2 编程语言的发展历程

本节简述按照时间发展顺序、在各个阶段出现的编程语言特点，这些阶段包括：早期的编程语言阶段，计算机语言基本模式的建立阶段，面向对象编程大发展的年代，互联网年代——大规模软件开发阶段。

1. 早期的编程语言（20 世纪 50 年代初至 60 年代中期）

编程语言是用来编写程序的符号，被用来表示某个计算或算法的详细步骤。一些（不是所有的）作者将编程语言限制为那些能够表达所有可能的算法的语言。

第一批使用指令与计算机交互的编程语言出现于 20 世纪 50 年代。

于 1949 年发布（运行在 BINAC 计算机上），并且于 1950 年（运行在 UNIVAC I 计算机上）和 1952 年（运行在 UNIVAC II 计算机上）完善的 Short Code 编程语言，是有史以来开发的有影响力的早期电子计算机高级语言之一。与使用机器代码直接运行程序的做法不同，Short Code 语言以一般公众可以理解的形式表示数学表达式。因为没有现代概念上的编译器，所以，使用该语言编写的程序必须在每次运行之前被手工翻译成机器代码。这种手工翻译的做法很快被今天所熟知的编译器所替代。

另外一种具有开创意义的编程语言是 1952 年在美国曼彻斯特大学开发完成的 Autocode 语言。该语言的重要意义在于该语言带有编译器，程序员使用编译器将该语言写成的程序自动转换成机器代码。对于生活在今天的计算机和软件专业的师生和计算机工作者来讲，编译器已经习以为常。但是在当时，能想到利用一个程序（编译器）对另外一个程序进行自动（而非手动）编译，即自动将程序代码转换为机器代码，是具有划时代的意义的。

于 1955 年年初发布的 Flow-matic 编程语言的特点是，使用英语语句代替了数学符号。其目的是使得商业人士也会使用该语言编程。Flow-matic 对 COBOL 的设计产生了重大影响。

FORTRAN 编程语言 FORTRAN I 是 1956 年由 IBM 公司开发的，并成为第一个被广泛使用的高级通用编程语言。该语言中包含大量的表示数学公式的语句，主要应用于科学计算。其优点是，具有很强的数学计算能力，使得程序员能够更加容易地将数学公式写入程序，从

而使得程序员再也不用使用汇编语言或者机器语言编写烦琐的程序。其缺点是，缺乏用于商业输入与输出的字符串操作。该语言不断更新的后续版本一直被持续使用到今天。

随着商业的发展，急需可以用于商业、银行业与金融业方面的计算机编程语言，因此，作为世界上第一个商用语言的 COBOL 语言应运而生。COBOL 语言是 1959 年发布的专门用于商业的语言。该语言的特点是面向数据处理、面向文件与面向过程。该语言适用于数据描述与文件处理。其主要应用领域为文件处理与商业应用，例如银行业、金融业软件的编写。该语言直接使用英语语句，而不是数学公式。在该语言中，为了商业上的输入与输出，使用了字符串数据类型。COBOL 语言成为当今还在使用的早期语言之一（当然，增加了很多特点）。

1958 年发布的 FORTRAN II，引入了可以独立编译的子程序的概念，其应用领域为科学与工程计算。

1959 年发布的 ALGOL 60 语言中引入了局部性、动态、递归、BNF 范式、块状结构与数据类型的概念，其应用领域为科学与工程计算。

1960 年发表的 Lisp 语言包含了列表处理、指针与垃圾处理，其特点是在编程中大量使用递归，其应用领域为人工智能。该语言中引入了算法抽象（子程序）的概念。

1964 年发布的 PL/1 语言综合了 FORTRAN、ALGOL 与 COBOL 语言的特点，支持数据抽象、面向过程编程、结构化编程与命令式编程。该语言既支持科学与工程计算，同时也支持商业应用。

从 20 世纪 50 年代末期开始到 60 年代末期出现的编程语言强调使用子程序或者过程进行算法抽象。

2. 计算机语言基本模式的建立（20 世纪 60 年代末期至 70 年代末期）

在 20 世纪 60 年代，尤其是随着半导体与集成电路技术的成熟，计算机硬件的价格大幅度降低，而处理能力却以几何形式大大增强，为使用计算机解决较大与较复杂的问题提供了可能性。然而，这需要对更多种的数据类型进行操作。所以，支持数据抽象的语言诞生了，例如，ALGOL 68 以及随后在 1970 年出现的 Pascal 语言。这些语言支持数据抽象、命令式与结构化编程。

在这个阶段，程序员可以描述相关种类的数据（它们的类型）并且让编程语言强制执行（带有数据结构的）设计决策。这一阶段出现的编程语言的共同特点是支持数据抽象（抽象数据类型）。

20 世纪 60 年代末到 70 年代期间产生了许多主要编程语言。现在使用的大部分语言模式，是在这一时期发明的。

1968 年出现的 ALGOL 68 支持数据抽象、科学计算与商业应用。

1967 年发布的 Simula 语言引入了对象、类、继承、子类与虚拟方法的概念。该语言支持数据抽象，为第一个面向对象编程语言，被用于模拟与仿真。

C 语言是通用的过程式的编程语言。在使用 C 语言编写的程序中，需要大量地使用指针，因此 C 程序运行速度快，并且可执行程序比较小。C 语言适用于所有领域，尤其适合仿真、电子游戏等需要高效率软件编写的领域。

FORTRAN 77 是在 1977 年发布的 ANSI 标准化的 FORTRAN 版本，主要用于进行科学与工程计算，为当今仍然在使用的语言。

1972 年发布的 Smalltalk 语言为支持自底向上设计的完整的面向对象语言。

值得注意的是，早在 1978 年就发布了作为关系型数据库管理系统的标准语言 SQL，其后在 1986 年 10 月，美国国家标准协会对其进行了规范，于 1987 年在国际标准化组织的支持下成为国际标准。

前面只介绍了少量的在 20 世纪 70 年代发布的编程语言，而实际上该时期是编程语言研究最活跃并且产生编程语言最多的年代。那时，越来越多的商业应用使得程序越来越大，因此需要新的语言机制来应对大应用程序开发的需要。令人难以置信的是，这 10 年左右的时间里共出现了 2 000 多种不同的编程语言。然而，只有少数语言存活了下来。虽然如此，许多语言机制为后来的编程语言提供了宝贵的借鉴之处，大多数的现代编程语言都可以在这些语言中找到其祖先。

20 世纪 70 年代也经历了关于“结构化编程”的优点方面相当多的争论，即是否使用 goto 语句编程的问题。在本章的稍后部分，我们将讨论一些 goto 语句的缺点。几乎所有的现代程序员都认为，即使允许使用提供了 goto 语句的编程语言，除了在极其特殊的情况下以外，使用 goto 语句也是不好的编程风格。

【例 1-1】 C 语言与 UNIX 操作系统的研发故事。C 语言是由在美国的 AT&T 和贝尔实验室（Bell Labs）工作的哈佛大学毕业的数学博士 Dennis Ritchie 在 1969 年到 1973 年之间开发的。UNIX 操作系统是由一个在贝尔实验室工作的包括 Ken Thompson、Dennis Ritchie、Brian Kernighan、Douglas McIlroy、Michael Lesk 和 Joe Ossanna 在内的团队在 1969 年开始研发的，其第一个版本在 1971 年发布。UNIX 操作系统原来是由汇编语言开发的，但是到了 1973 年，当 C 语言研发成功以后，研发组使用 C 语言重新改写了 UNIX 操作系统。由此可见，UNIX 操作系统和 C 语言就像一对孪生兄弟一样。C 语言与 UNIX 操作系统很快成为世界的主导。

1983 年，Ritchie 和 Thompson 一起获得了图灵奖（Turing Award），原因是他们研发了通用的操作系统理论，并且实现了 UNIX 操作系统。1990 年，Ritchie 和 Thompson 接受了 IEEE 研究所颁发的 IEEE Richard W. Hamming 奖牌。1999 年 4 月，Ritchie 和 Thompson 接受了 1998 年度美国国家技术奖牌，由克林顿总统颁奖。

UNIX 操作系统是发达国家已经使用了多年的操作系统，许多缺陷已经得到修改。与 Windows 操作系统相比较，UNIX 操作系统具有如下几方面的优点：

（1）稳定性：更稳定，不会经常死机，因此需要较少的维护和管理；

（2）安全性：具有强大的内置安全性；

（3）处理能力：具有更强大的处理能力。

3. 面向对象编程大发展的年代（20 世纪 80 年代）

在 20 世纪 80 年代，没有创造新的编程语言模式，主要的语言设计方面的工作都集中在巩固、细化过去 10 年发明的语言模式上。

毋庸置疑，整个 20 世纪 80 年代是面向对象理论、设计与编程大发展时期。基于 20 世纪 70 年代的面向对象的研究，在 20 世纪 80 年代初期，出现了在商业软件开发方面有实际应用的面向对象编程语言。

发布于 1980 年的 Smalltalk 80 为纯粹面向对象语言，最初为了教学目的，后来作为实际的商业软件开发。

1983 年发布的 C ++语言，结合了 C 和 Simula 的优点，为通用的面向对象语言。C ++语

言的特点是广泛地使用指针，因此用 C ++语言开发的程序运行速度快，可以用于所有的领域。C ++语言的出现受到了广大软件开发者的欢迎。经过了 30 多年，C ++语言仍然是当今人们青睐的面向对象编程语言，尤其用于模拟、网络电子游戏方面。

1983 年发表的 Ada 83 受 Pascal 语言的影响，属于强类型的面向对象编程语言。美国政府针对 Ada 语言进行了标准化，同时 Ada 语言成为用于国防承包商的一种编程语言。

这些面向对象语言的共同特点是：支持面向对象编程、设计与分析。

4. 互联网年代——大规模软件开发（20 世纪 90 年代至现在）

在 20 世纪 90 年代中期，互联网应用的快速增长对于编程语言产生了重大的影响。通过开放的计算机系统的全新的平台，互联网的发展给创造与之相适应的新语言一个全新的机会。此阶段有重要影响的语言是 Java 与 JavaScript 语言。

Java 是由 Sun Microsystems 公司在 1995 年推出的面向对象程序设计语言（以下简称 Java 语言）和 Java 平台的总称。Java 语言具有跨平台、动态 Web 开发与 Internet 计算的特点。Java 语言自面世后就非常流行，发展迅速。在全球云计算和移动互联网的产业环境下，Java 语言更具备了显著优势和广阔前景。Java 语言是一个纯粹的面向对象的程序设计语言，它继承了 C ++语言面向对象技术的核心。Java 语言舍弃了 C ++语言中容易引起错误的指针（以引用代替）、运算符重载与多重继承（以接口取代）等特性，增加了用于回收不再被引用的对象所占据的内存空间的垃圾回收器功能，使得程序员不用再为内存管理而担忧。

JavaScript 是一种直译式脚本语言，是一种动态类型、弱类型和基于原型的语言。JavaScript广泛用于客户端的脚本语言，最早在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网页增加动态功能。

从 1990 年开始，强调利用 Framework 进行编程，从而导致了 J2EE（现已改为 JavaEE）与 .NET Framework 的产生。Framework 通过组件与服务为程序员提供了大量的支持。

可以说，一个 Framework 就是一个可复用的设计组件，它规定了应用的体系结构，阐明了整个设计与协作构件之间的依赖关系、职责分配和控制流程，表现为一组抽象类以及其实例之间协作的方法，为构件复用提供了上下文（Context）。Framework 的使用使得软件的生产率有了极大的提高。

1990 年以后发布的编程语言与 Framework 的特点是支持大规模软件开发。

1.2 非结构化编程简介

非结构化编程（程序设计）是历史上最早的能够实现图灵完整算法的编程模式。历史上，过程化编程、结构化编程与面向对象编程都是非结构化编程的后继者。

使用非结构化编程思想设计的编程语言既包括低级编程语言，也包括高级编程语言，包括早期的 BASIC、JOSS、FOCAL、MUMPS、TELCOMP 和 COBOL 语言。

通常，利用非结构化的编程语言所写的程序包括依次排列的命令或语句，每行包含一条语句，并且每行都标有编号，以方便程序执行流从一行跳转到任何其他的一行。

非结构化编程引入了基本的控制流的概念，如循环、分支和跳转。虽然在非结构化的范式中没有过程（procedures）的概念，但是子程序是允许的。和过程不同的是，子程序可以有多个入口和出口，直接跳入子程序或者跳出子程序在理论上是允许的。

非结构化的编程语言广泛使用 goto 语句，使得程序的执行可以从一行跳转到任何其他的一行。也就是说，goto 语句是改变程序执行流的重要语句。如果没有 goto 语句，则程序的执行只能按照源程序中的语句自第一行开始，逐行执行到最后一行，中间偶尔会调用某个子程序。可以认为，goto 语句是非结构化编程语言最重要的特征之一。

但问题是，在提供程序执行时跳转便利的同时，goto 语句也给分析用非结构化语言编写的程序，尤其是比较大的程序，带来了很大的混乱。原因是，如果 goto 语句使用过多，程序执行的逻辑流就变得很难跟踪。例如，考虑一个具有 2000 行代码的程序，在代码中使用了 85 次 goto 语句，一会儿从第 280 行跳转到第 1235 行，一会儿又从第 1235 行跳转回第 66 行，等等。正如 20 世纪六七十年代软件业人士批评的那样，这样的程序就像是意大利面条一样，卷曲在一起，要找出一根面条的头和尾在哪里很难，而要找出一碗面条中每根面条的头和尾的位置，就更加困难。关于 goto 语句的辩论进行了很多年才结束。最后，人们普遍认识到，由非结构化语言编写出的程序代码可读性很差，难以理解和维护，很难被用于实现重大的软件项目。

另外，非结构化语言只允许使用基本的数据类型，如数字、字符串和数组。尽管还缺乏结构化数据类型，在非结构化语言中引入数组却是一个显著的进步，使得流数据处理成为可能。

1.3 结构化系统分析与设计方法简史

在整个 20 世纪 50 年代到 60 年代中期，几乎没有较好的程序设计和编程技术方面的指导，也没有记录需求与设计的标准技术。在 20 世纪 60 年代后期，软件在多个行业中有了较为广泛的应用。随着软件系统变得越来越大，越来越复杂，软件设计与开发也变得越来越困难。随之而来的问题是产生了软件危机：软件超预算，不能按时交付，所生产的软件的功能与用户需求相去甚远，从而急需新的软件开发方法论。

在这种大背景下，在 20 世纪 60 年代中期，产生了结构化编程的概念。然后，在 20 世纪 60 年代后期由 Larry Constantine 提出了结构化设计的概念。20 世纪 70 年代《结构化设计》（Structured Design）一书出版之后，才出现了结构化分析的概念与技术。

虽然按照结构化技术发展的历史顺序，结构化编程在前，然后是结构化设计，最后才是结构化分析理论的出现，但是在下面的讲述中，还是遵循现代软件开发流程的正确顺序：即先讲结构化分析，然后是结构化设计，最后才讲述结构化编程。

1.3.1 结构化分析

结构化分析（Structured Analysis，SA）和结构化设计（Structured Design，SD）首先分析业务需求，然后将业务需求转换为软件需求规格说明书，最后获得计算机程序与硬件配置等。

结构化分析从 20 世纪 80 年代开始流行，今天仍然被许多人使用。在结构化分析中，使用数据流图（Data Flow Diagrams，DFD）将问题分解为各个处理步骤。数据流图采用图形方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具。由于它只反映系统必须完成的逻辑功能，所以它是一种功能模型。数

据流图中的每个气泡代表一个包含多个子程序（或者函数）的处理模块；而对于层级较高的数据流图，一个气泡可能仅仅代表一个子程序或者函数。

明显地，数据和控制从一个气泡到数据存储再到另外一个气泡的跟踪是非常困难的，原因是气泡数目可能非常大（设想一个具有200个气泡的数据流图）。一种改进方法是首先定义需要系统反应的外部世界事件，然后用一个气泡代表一个事件，然后将需要交互的那些气泡连接起来，从而定义整个系统。大量的气泡导致系统很难理解，因此，通常将一些气泡组织成更大的（高层的）气泡。

结构化分析方法属于流程驱动。这种方法确定总体功能并且利用迭代的方法将功能（函数）划分成更小的功能（函数），同时保持输入、输出、控制和优化流程所需的机制不变。这种方法也被称为功能分解方法，它侧重于函数的内聚与函数间的耦合。

结构化分析从数据流经系统的视角考虑整个系统。系统的功能通过许许多多的变换数据流的过程（函数）描述。结构化分析利用分而治之（Divide and Conquer）技术，将一个问题分解为若干子问题，然后再次对每个子问题进行类似的分解，依次进行分解，一直到问题细化到比较容易理解与处理的程度才停止。这种技术被称为自顶向下的功能化逐层分解技术。自顶向下的功能化分解活动产生了很多层的DFD。这使得设计者能够将注意力集中在相关的细节而避免纠缠无关的细节。随着细节层次的增加，问题变得越来越容易理解。

在结构化分析中，除了使用数据流图以外，还使用实体关系图（Entity Relationship Diagram，ERD），以表达系统中出现的实体及实体之间的关系。下面粗略地介绍数据流图与实体关系图并且给出两个例子。

（1）数据流图

数据流图是一个数据如何流过某个信息系统的图形表示。通常的做法是，首先画出系统环境图，表示系统和外部实体之间的相互作用。数据流图用来表示系统是如何被划分为更小的部分的，并且突出数据是怎样在这些部分之间流动的。然后，该环境图可以被扩展以便显示系统建模的更多细节。

数据流图对于软件开发者和最终客户都是很重要的。通过绘制数据流图，程序员可以分析、分解一个系统。通过观察数据流图，用户能够以可视化的方式了解系统将如何运作，这个系统将要完成什么以及该系统将怎样被实现。数据流图可以用来为最终用户提供一个比较好理解的物理概念。

【例1-2】 考虑设计一个程序，使用二分搜索算法（Binary Search）对一个整数数组进行搜索。程序要求从一个文件输入一个整数数组，输入的时候要验证数组的每项是否都是整数。然后输入一个待查询的整数，针对此数组进行查询，看是否包含该整数。如果是，就返回该元素所在的数组中的位置；否则，返回值-1。该程序的二层与三层DFD如图1-1和图1-2所示。

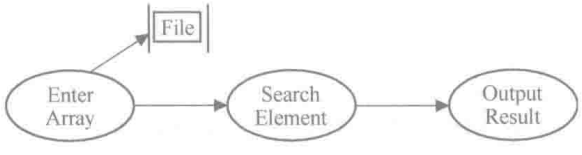

图1-1　使用二分搜索算法实现的程序的二层DFD