“十一五”国家级规划教材

教育部“高等学校教学质量与教学改革工程”立项项目

宋　雨 编著

软件工程实践教程

计算机科学与技术专业实践系列教材

清华大学出版社

普通高等教育“十一五”国家级规划教材

计算机科学与技术专业实践系列教材

教育部“高等学校教学质量与教学改革工程”立项项目

软件工程
实践教程

宋　雨 编著

清华大学出版社
北京

内 容 简 介

本书共 3 章，第 1 章系统综述了软件工程课程的核心内容，读者通读该章可达到提纲挈领的学习目的，该章的内容包括软件需求分析、软件设计、软件编码、软件测试、软件复用、面向对象的软件工程、软件维护、软件管理、应用 Web 工程、软件工程标准和软件文档。第 2 章给出了软件工程课程设计的内容及考核方式，这一章列出了精选的 100 个课题供读者选用，这些课题涉及很多应用领域，全部具有实际意义，有些就是实际的工程项目。课题中给出了系统应达到的功能要求、目标、性能指标、两种考核方式和具体的量化考核标准。第 3 章简要地列出了软件工程课程设计应交付文档的格式、各种文档应包含的主要内容及基本要求。附录中给出了软件工程课程设计任务书及软件工程课程设计文档评分表。

本书旨在为软件工程实践教学提供有价值的教材、参考文献和指导。本书可作为大学生或研究生进行软件类综合实验、课程设计、毕业设计或相关课题的教学用书或参考书，也可供想快速学习软件工程学科的读者阅读。

图书在版编目（CIP）数据

软件工程实践教程/宋雨编著．—北京：清华大学出版社，2011.3
（计算机科学与技术专业实践系列教材）
ISBN 978-7-302-23962-8

Ⅰ．①软…　Ⅱ．①宋…　Ⅲ．①软件工程－高等学校－教材　Ⅳ．①TP311.5

中国版本图书馆 CIP 数据核字(2010)第 202401 号

责任编辑：汪汉友　李玮琪
责任校对：时翠兰
责任印制：杨　艳

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62795954，jsjfc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185×260　　**印　张**：7.5　　**字　数**：168 千字
版　　次：2011 年 3 月第 1 版　　**印　次**：2011 年 3 月第 1 次印刷
印　　数：1～3000
定　　价：16.00 元

产品编号：036721-01

普通高等教育“十一五”国家级规划教材
计算机科学与技术专业实践系列教材

编 委 会

前　言

软件工程是一门实践性很强的课程，很多学校不但设置了软件工程这门课程，而且还设置了软件工程课程设计作为必修课，单算学分，可见软件工程实践教学的地位是很高的，本书就是为配合软件工程的实践教学而编写的。

本书的指导思想是篇幅尽量小、实用，不给读者造成负担。本书共3章，第1章系统综述了软件工程课程最主要的内容，读者只要通读该章就可用较短的时间回顾课程的全部内容，该章内容包括软件需求分析、软件设计、软件编码、软件测试、软件复用、面向对象的软件工程、软件维护、软件管理、应用Web工程、软件工程标准和软件文档，这些内容涵盖了教学大纲，既有传统的内容，也有软件工程新技术。这次重编时，在面向对象、应用Web工程、统一建模语言UML、软件测试等方面增加了笔墨，既突出了重点又不赘述；第2章给出了软件工程课程设计的内容及考核方式，这一章列出了精选的100个课题供师生选用，这些课题涉及很多应用领域，全部具有实际意义，有些就是实际的工程项目，课题中给出了系统应达到的功能要求、目标、性能指标、两种考核方式和具体的量化考核标准；第3章简要地列出了学生完成课程设计应交付文档的格式、各种文档应包含的主要内容及基本要求，这一章列出的文档包括可行性研究报告，软件计划，风险缓解、监测和管理计划，软件需求规格说明书，软件设计说明书，软件测试计划，测试分析报告，开发进度月报，用户手册，操作手册和项目开发总结报告。附录A是软件工程课程设计任务书，给出了成绩的评定方式。附录B给出了学生完成课程设计交付文档后的评分参考标准，该标准主要从规范性、原创性、工作量及逻辑性四个方面考察文档的质量。

本书旨在为软件工程课程设计提供有价值的教材、参考文献和实践指导，本书可作为大学生或研究生进行软件类综合实验、课程设计、毕业设计或相关课题的教学用书或参考书，也可供想快速学习软件工程学科的读者阅读。

本书旨在把实践教学系统化，读者在进行实践时，可先通读第1章的内容，巩固所学知识，同时也可起到集中辅导的作用，使理论基础和概念进一步增强。对于未曾接触过这些内容的读者，可达到事半功倍的学习效果，因为这一章是该课程内容的精华。之后根据第2章给出的实践内容，读者可选择一个课题，完成软件计划、需求分析、软件设计、编码、软件测试及软件维护等软件工程工作并按要求编写出相应的文档。

北京师范大学心理学院宋一辰帮助整理并录入书稿，书中实践课题参考了华北电力大学计算机系本科生的毕业设计，在此深表感谢。特别感谢清华大学出版社汪汉友老师，在他的积极策划和鼓励下，本书才得以问世。

作　者

2010年7月28日于华北电力大学

目　录

第 1 章　软件工程的主要内容

1.1　概述

软件是由计算机程序、数据及相关文档组成的，其中，程序是让计算机执行的指令序列，文档则是与程序开发、维护和使用相关的图文资料。软件和硬件共同构成完整的计算机系统，软件和硬件互相依存，两者缺一不可，软件有以下特征：

(1) 软件是逻辑产品，是脑力劳动的结晶，具有抽象性。

(2) 软件产品不会磨损和用坏，因而对软件产品的维护与硬件产品的维护有很大不同。

(3) 软件产品的成本高，软件的生产数量与成本基本无关，而且软件的维护费用大于开发费用。

若按功能对软件进行分类，则可将软件划分为系统软件、支撑软件和应用软件三大类；若按规模进行划分，可将软件分为六类，如表 1-1 所示；若按软件工作方式进行划分，则可分为实时处理软件、分时软件、交互式软件和批处理软件四类；若按软件服务对象的范围进行划分，则可分为项目软件和产品软件两大类；也可按其他方式对软件进行划分，例如按软件的使用频度进行划分，按软件失效的影响进行划分等。

表 1-1　按规模对软件的分类

类别	参加人员数	研制期限	产品规模(源程序行数)
微型	1	1～4 周	0.5 千行
小型	1	1～6 月	1～2 千行
中型	2～5	1～2 年	5～50 千行
大型	5～20	2～3 年	50～100 千行
甚大型	100～1000	4～5 年	1 兆行(＝1000 千行)
极大型	2000～5000	5～10 年	1～10 兆行

软件是伴随着电子计算机的诞生而出现的，到目前为止，软件发展大致可分为以下四个阶段。

(1) 程序设计阶段(1946 年—20 世纪 50 年代末)，以手工编程方式生产软件，使用的工具是机器语言和汇编语言。

(2) 程序系统阶段(20 世纪 60 年代)，以作坊式的小集团合作生产软件，生产工具是高级语言。

(3) 软件工程阶段(20 世纪 70 年代以后)，以工程化的方式生产软件，使用数据库、开发工具、开发环境、网络、分布式和面向对象技术等开发软件。

(4) 现代软件工程阶段(20 世纪 80 年代末至今),伴随着网络技术的发展,软件工程也进入了快速发展时期,网络环境下的软件工程规模更大、系统更复杂,并且系统间相互作用,在网络环境下软件工程的关注域转向需求,软件将以"服务"作为基本模块,软件的演化比测试更重要,问题的形式化向着本体描述发展。网构软件是在互联网开放、动态和多变环境下软件系统基本形态的一种抽象,它既是传统软件结构的自然延伸,又具有区别于在集中封闭环境下发展起来的传统软件形态的独有的基本特征:自主性、协同性、反应性、演化性和多态性。传统的软件理论、方法和技术等在处理网构软件时都遇到了一系列的挑战。

在软件发展的第二阶段,硬件技术的迅速进步导致软件技术的发展不能满足要求,从而出现了软件危机。软件危机是指在计算机软件开发和维护过程中所遇到的一系列严重的问题。软件危机的表现形式多种多样,造成软件危机的原因是软件产品本身的特点以及开发软件的方式、方法、技术和人员所引起的。

为了克服软件危机,在 1968 年北大西洋公约组织召开的学术会议上首先提出了"软件工程"的概念,提出要用工程化的思想来开发软件,按工程化的原则和方法组织软件开发是摆脱软件危机的重要出路。软件工程是一门用科学知识和技术原理来定义、开发和维护软件的学科,它目前已成为计算机科学中的一个重要分支。

为获得软件产品,在软件工具的支持下由软件工程师完成的一系列软件工程活动称为软件工程过程。软件工程过程的基本活动有 P(Plan——软件规格说明)、D(Do——软件开发)、C(Check——软件确认)和 A(Action——软件演进)。因此,软件工程过程可看作是针对某类软件产品而规定的工作步骤。软件工程的基本活动可展开成制定计划、需求分析、设计、编码、测试、运行和维护六个阶段,这六个阶段称为软件的生存期,描述软件开发过程中各种活动如何执行的模型称为软件生存期模型,软件生存期模型主要有以下几种。

(1) 瀑布模型　该模型将各种软件工程活动按次序固定,自上而下相互衔接,如同瀑布流水。

(2) 演化模型　该模型先开发一个软件"原型",在此基础上逐步完善。

(3) 螺旋模型　该模型将上述两种模型结合起来,并增加了两种模型均忽略的风险分析,每经过一个螺旋周期,便都开发出一个功能更完善的、新的软件版本。

(4) 喷泉模型　该模型适合于面向对象的开发方法,开发工程具有迭代性和无间隙性,无间隙是指在分析、设计和编码等活动之间不存在明显的边界。

(5) 智能模型　该模型把专家系统与上述若干模型结合起来,采用归纳和推理机制,因而这种模型也称为基于知识的软件开发模型。

(6) 增量模型　该模型融合了瀑布模型的基本成分(重复的应用)和原型实现的迭代特征,它以小的、但可用的片段(称为增量)来交付软件。通常,每个增量的建造都是基于那些已经交付的增量而进行的,任何增量均可以按原型开发模型来实现。

(7) 并发过程模型　该模型也称为并发工程,它被表示为一系列的主要技术活动、任务及它们的相关状态。并发过程模型定义了一个活动网络,网络中的每一个活动均可与其他活动同时发生,在一个给定的活动或活动网络中,其他活动中产生的事件触发一个活

动中状态的变迁。

(8) 基于构件的开发模型　该模型是在面向对象技术的基础上发展起来的，它融合了螺旋模型的许多特征，利用预先包装好的软件构件(有时称为类)来构造应用系统。

(9) 形式化方法模型　该模型是一种严格的软件工程方法，它是一种强调正确性的数学验证和软件可控性认证的软件过程模型。

(10) 第四代技术模型　该模型包含了一种组件工具，它们具有共同点，能使开发人员在较高的级别上规约软件的某些特征，并把这些特征自动生成源代码。

(11) 混合模型　每种软件生存期模型都不是十全十美的，要让它们适应各种项目的开发和各种情况的需要也是很困难的。开发混合模型的目的是为了发挥各个模型的优势，对于具体开发组织也可使用不同的模型组成一个较实用的混合模型，以便获得最大的效益。

软件工程项目的基本目标是以较低的开发成本达到预期的软件功能和良好的软件性能，能按时完成开发工作并及时交付使用。开发出的软件具有便于移植、可靠性高、需要的维护费用低的优点。为达到这些目标，在软件开发过程中必须遵循抽象、信息隐蔽、模块化、局部化、确定性、一致性、完备性和可检验性的原则，这八个原则也称为软件工程原则。

1.2　软件需求分析

在软件需求分析之前应进行软件的可行性研究，并制定软件项目计划。可行性研究包括经济可行性、技术可行性、法律可行性及方案的选择四项内容，工作结束后提交可行性研究报告。软件项目计划包括确定软件作用范围、确定资源、估算开发成本和安排开发进度四项工作。软件需求分析的目标是深入描述软件计划中所确定的功能和性能，确定软件设计的约束、软件与其他系统元素的接口细节，定义软件其他的有效性需求。需求分析的任务是借助当前系统的逻辑模型导出目标系统的逻辑模型，解决目标系统“做什么”的问题，具体可包括问题识别、分析和综合、建模、编写文档和对需求分析进行评审五项工作。

软件需求获取技术包括建立获取用户需求的方法框架，支持和监控需求获取的过程两个方面。获取用户需求的主要方法是调查研究，它具体包括了解系统的需求、进行市场调查、访问用户及用户领域的专家和考察现场四个方面。在做调查研究时，可以采用如下的方式。

(1) 制定调查提纲，向不同层次的用户发调查表。

(2) 召开分层次的用户调查会，听取各层次用户对待开发系统的想法和建议。

(3) 向用户领域的专家及关键岗位上的工作人员进行咨询。

(4) 实地考察，跟踪现场实际业务的流程。

(5) 查阅有关资料。

(6) 使用建模工具，如数据流图、任务分解图、网络图、用例图、用 UML 表示的状态图和活动图等。

软件需求分析的方法主要有结构化分析方法、动态分析技术及支持需求分析的原型化方法。

需求分析阶段结束后应交出经复审通过的软件需求规格说明(Software Requirements Specification,SRS),它是需求分析的最终产物,通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准,给出对目标软件的各种需求。为保证软件需求定义的质量,复审应由专门指定的人员按规程严格进行,并给出结论性的意见。除分析员之外,用户/需求者、开发部门的管理者以及软件设计、实现、测试人员都应当参加 SRS 的复审工作。复审结果一般要包含一些修改意见,待修改完成后再经复审通过,才可进入设计阶段。

1.2.1 结构化分析方法

结构化分析(Structured Analysis,SA)是面向数据流进行需求分析的方法,是一种建模技术,它建立的分析模型如图 1-1 所示。

分析模型的核心是数据词典,它描述了所有在目标系统中使用和生成的数据对象。围绕这个核心有三种图:实体—关系图(Entity Relation Diagram,ERD)描述数据对象之间的关系;数据流图(Data Flow Diagram,DFD)描述数据在系统中如何被传送或变换,并描述对数据流进行变换的功能;状态—迁移图(Status Transfer Diagram,STD)描述系统对外部事件如何响应,如何动作。因此,ERD 用于数据建模,DFD 用于功能建模,而 STD 用于行为建模。

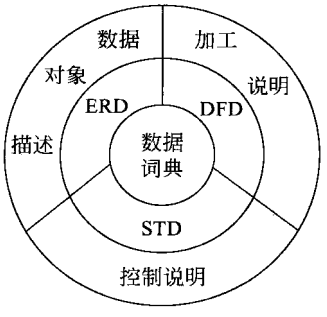


图 1-1 分析模型的结构

(1) 数据建模 数据模型包括三种互相关联的信息,即数据对象、描述对象的属性和描述对象间相互连接的关系。

(2) 功能建模 使用 DFD 来表达系统内数据的运动情况,数据流的变换可用结构化英语、判定表或判定树来描述。画 DFD 可按由外向内,自顶向下,分层描述,逐步细化、求精和完善的步骤进行。

(3) 行为建模 使用状态—迁移图和状态-迁移表来描述系统或对象的状态、导致系统或对象状态改变,从而描述系统的行为,或使用 Petri 网描述相互独立、并发执行的处理系统。

(4) 数据词典 准确、严格地定义与系统相关的每个数据元素、数据结构、数据流及数据文件的具体含义,以字典顺序将它们组织起来,使用户和分析员对所有的输入、输出、存储数据及中间计算都有共同的理解。

1.2.2 动态分析技术

这是面向对象的技术,用动态模型描述系统中与时间和操作序列有关的内容,即标识改变的事件和事件序列,定义事件上下文状态及事件和状态的组织。动态模型着眼于“控制”,即描述系统中发生的操作序列,而不考虑操作做些什么、对什么进行操作以及如何实

现这些操作。

动态分析技术常用的工具是状态图，每一个状态图都展示了系统中对象类所允许的状态和事件序列。动态分析技术的第一步是编写典型交互行为的脚本，尽管脚本中不可能包括每个偶然事件，但至少必须保证不遗漏常见的交互行为。第二步是从脚本中提取出事件，确定触发每个事件的动作对象以及接受事件的目标对象。第三步是排列事件发生的次序，确定每个对象可能有的状态及状态间的转换关系，并用状态图描述它们。最后，比较各个对象的状态图，检查它们之间的一致性，确保事件之间的匹配。

1.2.3 支持需求分析的原型化方法

在软件开发中，原型是指软件的最初可运行的版本，它反映了最终的系统的部分重要特性。在了解了基本需求后，通过快速分析构造出一个小型的软件系统，它能满足用户最基本的要求，用户可在试用原型系统的过程中亲身感受并受到启发，对系统做出反映和评价。然后开发人员根据用户的意见对原型改进，经过不断试验、纠错、使用、评价和修改，获得新的原型版本。如此周而复始，逐步确定出各种需求细节并适应需求的变更，从而提高最终的软件产品的质量。

由于运用原型的目的和方式不同，原型可分为废弃型和演化型两种类型。采用废弃型原型是先构造一个功能简单、质量要求不高的模型系统，然后针对这个模型系统反复地进行分析修改，形成比较好的设计思路，据此设计出更加完整、准确、一致和可靠的系统。系统构造完成后，原来的模型系统被废弃。废弃型原型还可再分为探索型和实验型两种。探索型原型主要针对开发目标模糊，用户和开发者对项目都缺乏经验的情况，目的是要弄清楚对目标系统的要求，确定所希望的特性，并探讨多种方案的可行性。而实验型原型用于大规模开发和实现之前、考核方案是否合适以及规格说明是否可靠的情形。演化型也称为追加型，采用这种原型是先构造一个功能简单且质量要求不高的模型系统作为最终系统的核心，然后通过不断地扩充修改，逐步追加新要求，发展成为最终系统。

若在需求分析阶段使用原型化方法，则必须从系统结构、逻辑结构、用户特征、应用约束、项目管理和项目环境等多方面来考虑，以决定是否采用原型化方法。

原型的开发和使用过程称为原型生存期，它包括快速分析、构造原型、运行和评价原型、修正和改进、判定原型是否完成、判定原型细部是否需要严格地说明、对原型细部的说明、判定原型效果和整理原型并提供文档九个阶段。

构造原型的技术通常包括可执行规格说明、基于场景的设计、自动程序设计、专用语言、可复用的软件构件和简化假设等。

1.3 软件设计

软件设计是软件开发阶段中最重要的步骤，也是软件开发过程中用以保证质量的关键步骤。软件设计就是把软件需求变换成软件表示的过程，它可分两步完成。首先，概要设计，即将软件需求转化为数据结构和软件的系统结构，并建立接口描述；其次，详细设计，即过程设计，通过对结构的表示进行细化，得到软件各功能块的详细数据结构和算法

描述。

1.3.1 软件设计的原则

(1) 抽象化，包括过程抽象、数据抽象以及控制抽象。
(2) 自顶向下、逐步细化。
(3) 模块化。
(4) 信息隐蔽。

1.3.2 软件体系结构设计

体系结构设计的主要目标是开发一个模块化的程序结构，并表示出模块间的控制关系。其次，体系结构设计将程序结构和数据结构结合，为数据在程序中的流动定义接口。软件系统的体系结构经历了一个由低级到高级的发展过程，其间出现过的体系结构的风格可归纳为以下几种。

(1) 以数据为中心的体系结构。
(2) 数据流体系结构。
(3) 调用—返回体系结构。
(4) 面向对象体系结构。
(5) 层次体系结构。

1.3.3 模块独立性

模块独立性指软件系统中每个模块只涉及系统要求的具体的子功能，与其他的模块联系最少且接口简单。模块独立性概念是模块化、抽象和信息隐蔽这些原理的直接产物。模块独立性可用两个定性准则：耦合和内聚来度量。耦合是模块之间相互联系程度的一种度量，它从强到弱可分为内容耦合、公共耦合、外部耦合、控制耦合、标记耦合、数据耦合以及非直接耦合七种方式。内聚是对模块功能强度的度量，它反映了一个模块内部各个元素彼此结合的紧密程度，它从强到弱也分为七种：功能内聚、信息内聚、通信内聚、过程内聚、时间内聚、逻辑内聚和巧合内聚。耦合和内聚其实是一个问题的两个方面，模块之间的连接越紧密，联系越多，耦合度就越高，其模块独立性就越弱。一个模块内部各元素之间的联系越紧密，则它的内聚性就越高，相对地，它与其他的模块之间的耦合度就越低，而模块独立性就越强。因此，独立性比较强的模块应是高内聚、低耦合的模块。

1.3.4 结构化设计方法

结构化设计(Structured Design，SD)方法是以 DFD 为基础建立软件的结构，因此又称为面向数据流的设计。

DFD 有两种典型的类型，一种是变换型，一种是事务型。变换型的 DFD 是一种线形状结构，由输入、变换和输出三部分组成。变换是系统的主加工，变换输入端的数据流称为系统的逻辑输入，变换输出端的数据流称为系统的逻辑输出，系统输入端的数据流称为物理输入，系统输出端的数据流称为物理输出。外部输入的数据(即物理输入)一般要经

过正确性和合理性检查、编辑和格式转换等预处理，变成逻辑输入，输入给主变换，主变换产生的逻辑输出也要经过一系列变换转换成外部形式输出。事务型的 DFD 是一种放射状结构，其特征是 DFD 中有一个事务处理中心，后面是若干个活动通路，根据条件选择一条路径。

大系统的 DFD 一般是分层的，由分层的 DFD 映射的软件结构图也应是分层的，这样便于设计，也便于修改。由于 DFD 的顶层图反映的是系统与外部环境的界面，所以系统的物理输入与物理输出都在顶层图，因而相应的软件结构图的物理输入与输出部分放在主图中较为合适，以便和 DFD 中的 I/O 对照检查。

在设计时要整体地看，客观地看，一上来不要拘泥于细节和局部，在设计当前模块时，先把这个模块的所有下层模块都定义成“黑箱”，并在系统设计中利用它们，暂时不考虑它们的内部结构和实现方法。在这一步中定义的“黑箱”，由于已确定了它的输入、功能和输出，在下一步就可对其进一步进行设计，这样又会导致更多的“黑箱”。最后，全部“黑箱”的内容和结构都应完全被确定。这就是常说的自顶向下、逐步求精的过程。使用黑箱技术的主要好处是设计人员可以集中精力只关心当前的有关问题，暂时不必考虑琐碎的细节。

1. 总体设计的过程

(1) 精化 DFD。

(2) 确定 DFD 的类型，设计软件结构的顶层和第一层。

(3) 分解上层模块，设计中、下层模块。

(4) 根据优化准则对软件结构求精。

(5) 描述模块功能、接口及全局数据结构。

(6) 复审。如果有错，转向(2)修改完善，复审通过后进入详细设计。

2. 变换型 DFD 的设计步骤

(1) 确定 DFD 的变换中心、逻辑输入和逻辑输出。

(2) 设计软件结构的顶层和第一层。顶层模块一般只有一个，它的功能是完成所有模块的控制，该模块的名称就是系统的名称。第一层设计三种功能模块：输入、输出和变换模块，具体地说就是为每个逻辑输入都设计一个输入模块，其功能是向顶层模块提供数据；为每一个逻辑输出都设计一个输出模块，其功能是将顶层模块提供的数据输出；为变换中心设计一个变换模块，其功能是将逻辑输入变换成逻辑输出。

(3) 设计中、下层模块。

① 为每个输入模块都设计两个下属模块，其中一个是输入模块，另一个是变换模块，其中输入模块是向父模块提供数据的，变换模块则是将数据转换成父模块所需要的形式。该步骤可一直进行下去，直至达到系统的物理输入端。

② 为每个输出模块都设计两个下属模块，其中一个是变换模块，另一个是输出模块，变换模块的功能是将父模块提供的数据转换成输出形式，输出模块则是将变换后的数据输出。该步骤也可一直进行下去，直至达到系统的物理输出端。

③ 设计变换模块的下属模块。

(4) 设计优化。

用上述方法设计出的软件结构图与 DFD 是完全对应的，也是 SD 方法所设计出的

标准结构，但在实际中，可根据具体情况灵活掌握，可把 DFD 中的多个变换组成一个模块，也可把一个变换映射成多个模块，其目的都是设计出由具有高内聚、低耦合模块组成的具有良好特性的软件结构。另外，在软件结构图中要标上模块之间数据的传递关系。

3. 事务型 DFD 的设计步骤

(1) 确定 DFD 的事务中心和处理路径。

(2) 设计软件结构的顶层和第一层。顶层模块是主控模块，它的功能一是接收数据，二是根据事务类型调度相应的处理模块。因此，事务型软件结构应包括接收分支和发送分支两个部分。接收分支负责接收数据，发送分支通常包含一个调度模块，当事务类型不多时，调度模块可与主模块合并。

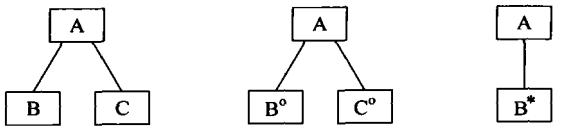
(3) 设计中、下层模块。接收分支下属模块的设计方法与变换型 DFD 的输入部分的设计方法相同，发送分支中的调度模块控制管理所有下层的事务处理模块，每个事务处理模块都还可调用若干操作模块，每个操作模块都还可有若干个细节模块，多个事务处理模块可以共享某些操作模块，多个操作模块也可共享某些细节模块。

(4) 设计优化。同变换型结构要求相同。

1.3.5 Jackson 系统开发方法

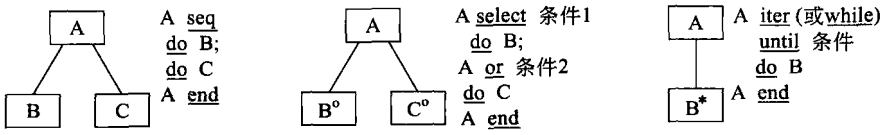
Jackson 系统开发方法(Jackson System Development，JSD)是一种典型的面向数据结构的分析与设计方法，该方法中没有特别强调模块的独立性，模块是作为软件设计的副产品而出现的，用 JSD 方法的最终目标是要得到软件的过程性描述。使用 JSD 方法的步骤是实体动作分析、实体结构分析、定义初始模型、对功能进行描述、决定系统特性以及实现，前三步为需求分析，后三步为软件设计。

Jackson 提出任何数据结构都可用三种基本构造类型按层次组合而成，图 1-2 为用 Jackson 图表示的三种基本构造类型数据结构。程序结构与数据结构完全对应，最后用伪码描述出来，图 1-3 为图 1-2 所对应的程序结构以及伪码表示。



(a) 顺序结构　(b) 选择结构　(c) 重复结构

图 1-2　用 Jackson 图表示的三种基本数据结构



(a) 顺序结构及其伪码　(b) 选择结构及其伪码　(c) 重复结构及其伪码

图 1-3　用 Jackson 图表示的程序结构(与图 1-2 对应)及伪码表示

JSD方法是从输入和输出数据结构中导出程序结构，有些中间处理过程可能在结构图中反映不出来，因而需要对Jackson图表示的程序结构及伪码进行求精和优化，使其完整和易于实现。

在用JSD方法进行设计时，必须找出输入和输出数据在内容、数量和顺序上的对应关系，否则无法将Jackson图表示的数据结构映射成程序结构，这就是所谓的“结构冲突”，结构冲突有顺序冲突、边界冲突和多重穿插冲突三种类型，在有冲突的情况下需进行预处理，在解决了结构冲突后，才能用JSD方法。

1.3.6 数据及文件设计

好的数据设计往往能产生好的软件结构，使模块的独立性增强，程序的复杂性降低。数据设计可分两步进行，第一步，为在需求分析阶段所确定的数据对象选择逻辑表示，对不同的结构进行算法分析。第二步，确定对逻辑数据结构所必须进行操作的程序模块，以便限制或确定各个数据设计决策的影响范围。

文件设计的主要工作是根据使用要求、处理方式、存储的信息量、数据的活动性以及所能提供的设备条件等来确定文件类别，选择文件媒体，决定文件组织方式，设计文件记录格式并估算文件的容量。

文件设计的过程主要分为两个阶段，第一个阶段是文件的逻辑设计，主要在概要设计阶段实施，它包括整理必需的数据元素、分析数据间的关系以及进行文件的逻辑设计三项工作。第二个阶段是文件的物理设计，主要在软件的详细设计阶段实施，它包括理解文件的特性、确定文件的存储媒体、确定文件的组织方式、确定文件的记录格式以及估算存取时间和存储容量五项工作。

1.3.7 软件详细设计

软件详细设计就是对软件过程的描述，给出各模块的具体算法描述和评价，软件详细设计的结果是软件编码的依据。

软件详细设计的工具主要有三大类：图形工具、表格工具和语言工具。常用的图形工具有程序流程图、N-S图、PAD(Problem Analysis Diagram)以及HIPO图(Hierarchy Plus Input Process Output)，N-S图和PAD均是由程序流程图演变而来的。HIPO图的适用范围很广，不限于详细设计。常见的表格工具是判定表，它比较简洁，但有一定的局限性，不能成为通用的设计工具。语言工具即PDL(Program Design Language)，它是一种伪码，与高级语言很接近，因而人们可以方便地使用计算机来完成PDL的书写和编辑工作。

1.3.8 软件设计的复审

软件设计结束后要进行复审。软件设计的最终目标是节省开发费用、降低资源消耗和缩短开发时间，因而要选择能够赢得较高生产率、较高的可靠性和可维护性的方案。复审的目的是及时发现并解决在软件设计中出现的问题，防止把问题遗留到开发的后期阶段，造成更大的损失。经复审通过后所交付的软件设计规格说明是软件设计结束的标志，