CHAPTER

第5章 软件设计

软件设计以需求分析为依据，是软件研发的关键环节，将直接影响软件项目的质量乃至成败。在软件分析阶段主要确定各种需求指标的具体要求，确定新软件系统具体“做什么”，还必须通过软件设计解决“怎么做”的问题（具体实现方案）的表示与描述，以便为后续软件系统的总体实现提供重要依据和方案。

教学目标

（1）掌握软件设计的概念、目标、阶段和过程。
（2）熟悉软件总体设计及详细设计的任务和原则。
（3）掌握数据库设计、网络设计和界面设计要点。
（4）掌握软件设计工具使用及设计文档编写方法。

教学课件
第5章　软件设计

5.1 软件设计概述

【案例 5-1】 在运动服销售手机 App 研发项目中，软件设计在整个研发过程中至关重要。某网络信息有限公司的软件研发部在进行需求分析的基础上，进行软件总体设计和详细设计，由于在设计网络查询与统计等细节方面出现疏忽，导致研发的软件部分功能及性能出现问题，不能很好地满足企业用户的需求而返工修改，延误了交付时间，对研发公司的信誉造成了不良影响和重大损失。

5.1.1 软件设计的概念和目标

软件设计也称为**系统设计**，是利用软件研发技术、工具和方法，确定新软件系统的具体物理实现方式、方法和方案的过程。其**总体目标**是：将软件需求分析阶段确定的各种功能对应的逻辑模型，转换为具体的物理模型（描述新软件“如何做”，即“实现方案”的物理过程），确定一个合理的软件系统的体系结构，包括划分组成系统的模块与功能、模块间的调用关系及接口、软件系统所用的数据结构等，最后完成“软件设计说明（书）”，提高软

件性能及可靠性、可维护性以及质量与效率。

软件设计分为两大阶段：总体设计和详细设计。其中，**总体设计**又称为**概要设计**，在此阶段应确定软件系统的总体实现方案、给出软件的模块结构、编写总体设计文档等。**详细设计**也称为**过程设计**，是对总体设计(概要设计)的一个具体细化的过程，确定组成模块及联系、处理过程、数据库设计、网络设计、界面设计、软件设计文档和实现具体方案等，为后续软件的具体实现提供详细方案及依据。

知识拓展

软件设计与软件分析的关系

5.1.2 软件设计的过程

软件工程与建筑工程等有很多类似之处，在实际施工前都需要进行认真细致的设计工作。软件设计需要先进行总体设计(即概要设计)，从总体上进行宏观概要架构设计，将软件实际的需求转化为软件的系统结构和数据结构。在经过“复审”得到可接受的总体设计方案后，进入“详细设计”，进一步进行“模块描述”，最后还要经过“复审”，完成“设计说明(书)”。**软件设计的工作过程**如图 5-1 所示。

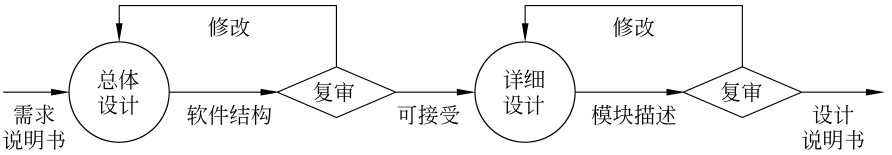


图 5-1 软件设计的工作过程

软件设计建立在软件需求分析的基础上，需求分析的结果为设计提供了最基本的输入要求和信息。分析模型的每个元素都提供了创建设计模型时所需的信息。图 5-2 描绘了软件设计过程中的信息流。由数据模型、功能模型和行为模型清楚表示的软件需求被传送给软件设计者，以适当的设计方法完成体系结构设计、过程设计、数据(库)设计和接口设计。手机软件开发流程和步骤略有差异。

知识拓展

手机软件开发流程和步骤

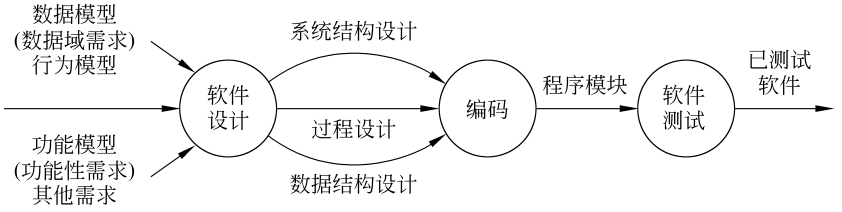


图 5-2 软件设计过程中的信息流

讨论思考

(1) 什么是软件设计？其总体目标具体是什么？

(2) 软件设计主要分为哪两个阶段？具体做什么？

(3) 软件设计的工作过程具体是什么?

(4) 画出软件设计过程中的信息流。

5.2 软件总体设计

5.2.1 软件总体设计的任务

根据软件需求分析的指标要求,在总体设计阶段,利用合适的设计方法和工具,将一个复杂的软件系统按功能划分成模块,概要设计其功能、性能、可靠性和接口等要素“怎么做”,以及模块之间的调用关系、数据结构与处理流程,最后完成文档及评审。

软件总体设计的任务主要包括 9 方面。

(1) 软件系统总体结构设计和模块结构设计。主要确定软件系统的总体层次(逻辑)结构及系统的体系结构(如 B/S 结构等),并将系统划分为功能模块及子模块,确定模块及子模块的功能结构和联系,并画出相应的软件总体功能结构图。

(2) 软件处理流程设计。确定软件系统的功能模块之间、子模块之间传输数据与处理流向和实际的调用关系及顺序。

(3) 确定软件的功能及调用关系。包括确定各功能与程序结构的功能及调用关系。

(4) 数据库及数据结构总体设计。确定软件系统与模块之间的逻辑结构设计、物理结构设计、数据结构与程序的关系。

(5) 网络及接口概要设计。对软件系统有关的交互网络、用户界面、软件接口、硬件接口和模块之间的内部接口进行概要性设计。

(6) 确定软件系统的实现总体方案。主要对上述各项概要设计,总结出软件系统总体的具体实现方案,还应考虑软件运行模块的组合、运行控制及时间等设计。

(7) 出错问题概要设计。包括出错输出信息、出错处理对策等概要设计。

(8) 性能、可靠性及安全性概要设计。特别是对一些特殊行业及特殊需求应用方面。

(9) 文档及维护概要设计。总体设计文档并进行阶段评审,以及后续维护修改。

5.2.2 总体设计的原则和过程

总体设计的总原则和过程是:由宏观到微观、逐步求精的原则,定性与定量分析相结合、分解与协调相结合和模块化方法,并兼顾软件的通用性、关联性、整体性和层次性。根据软件的总体结构、功能、任务和目标的要求进行分解,使各组成模块之间互相协调配合,实现软件的整体优化。**软件设计的基本原则**是:模块化、抽象、内聚和耦合、子系统及模块划分、信息隐藏等。

1. 软件工程模块化

模块(Module)是构成软件系统的基本构件,主要由数据说明、相关的执行语句等程序对象构成。模块在程序中单独命名,并通过名字实现对指定模块的访问、调用和运行。

例如,过程、函数、子程序,以及面向对象程序设计中的对象、方法等都可以作为模块。通常软件系统都由模块和子模块(可能多层)构成。

模块化(Modular)是将复杂的软件划分为功能相对独立且易于处理的模块的过程,软件的层次结构正是模块化的具体体现。模块化是软件的一个重要属性,其特性提供了处理复杂问题的一种方法,同时也有利于软件的有效实现和管理。

知识拓展
模块化的作用和意义

2. 抽象和逐步求精

抽象是指提取事物的本质特性而暂时不考虑其细节的方法。在现实中,一些事物、状态或过程之间总存在着某些相似的共性,将其集中和概括,暂时忽略其他较小差异有利于认识事物的本质特征。抽象是在人类认识复杂问题的过程中使用的最强有力的思维分析及解决问题的工具。在软件开发中,从系统定义到实现,每步进展都可看作是对软件解决方案抽象化过程的一次细化,将其相似方面集中和概括,暂时忽略其他差异,有助于理解和设计。

逐步求精是指集中精力解决主要问题而尽量推迟并逐步考虑细节问题的方法,是人类解决复杂问题时采用的一种基本策略,也是软件工程技术的基础,如规格说明技术、设计和实现技术、测试和集成技术等。求精是个细化过程,从高度抽象级别定义的功能(或数据)描述开始,仅概念性地描述了功能或数据,而没有提供功能的内部工作情况或数据的内部结构。求精要求细化原始描述,随后续求精(细化)步骤的完成会提供越来越多的细节。

知识拓展
逐步求精的由来及意义

注意:抽象与求精是互补的。抽象使得设计者注重过程和数据,而忽略了低层细节。可将抽象看作是一种通过忽略多余的细节而突出主要环节,实现逐步求精的方法。求精有助于设计时揭示出低层细节。两者结合有助于在设计演化过程中创造出完整的设计模型。

3. 模块的内聚和耦合

在设计模块时既要考虑“模块的具体功能”,又要考虑“应怎样与其他模块交流信息”。兼顾模块的内聚性和耦合性并保持相对的“功能独立性”很重要,以便提高效能并降低开发、测试、维护成本。但并非要求模块之间绝对孤立,软件系统要完成某项任务,需要多个模块相互配合,以及模块之间的信息交流。

(1) **内聚**。内聚表示一个模块内部各个组成元素之间相互结合的紧密程度,是信息隐藏和局部化概念的自然扩展。模块中组成元素结合得越紧密,模块的内聚性就越高,模块的独立性也就越强。设计时应该力求做到高内聚,理想内聚的模块只做一件事情。内聚按强度从低到高有 7 种类型:偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚和功能内聚,设计时应该力求做到功能高内聚,并将功能相近的子模块集成在一个模块中。

知识拓展
模块内聚的具体种类

(2) **耦合**。耦合是指在软件模块之间(外部)互相依赖的方式和程度,是影响软件复杂程度及性能的一个重要因素。模块间的耦合程度将影响系统的可理解性、可测试性、可靠性和可维护性。其强弱取决于模块间接口的复杂程度、进入或访问一个模块的点以及通过接口的数据。其强度所依赖的主要因素有模块间的调用、传递的数据量、施加的控制、接口的复杂程度。

知识拓展

模块耦合的具体种类

注意:内聚和耦合密切相关,同其他模块存在强耦合的模块通常可能弱内聚,而强内聚的模块也可能与其他模块之间存在弱耦合。在实际研发中,很难做到理想模块内部都是功能内聚,而模块之间都是数据耦合,但模块的设计应最大限度地追求高内聚、低耦合。

4. 子系统及模块的划分

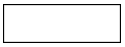
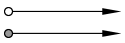
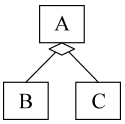
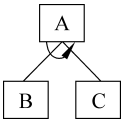
软件体系结构设计的三要素是程序构件(模块)的层次结构、构件之间交互的方式和数据的结构。子系统及模块划分除了上述要求模块高内聚、低耦合外,还应重点考虑以下因素。

(1) 模块大小适当。实际上,当模块大小块数增加时,模块间的联系也会增加,对这些模块进行连接的工作量也随之增加。

(2) 模块的层次结构。对需求中每一个功能,用哪一层,哪个模块、类和对象来实现;反之,应说明将要创建的系统每一层、模块、对象、类“做什么”和具体实现的功能。

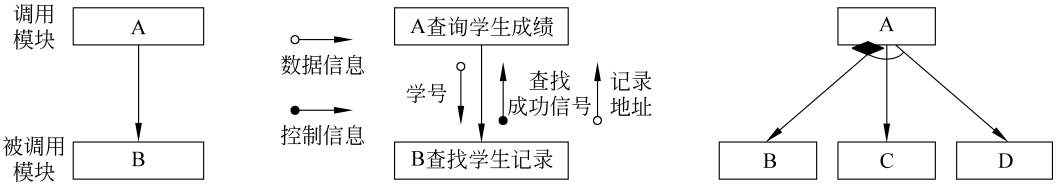
结构图(Structure Chart,SC)是准确描述、表达软件结构的图形表示方法,可反映模块之间的层次调用关系和联系。模块**结构图常由特定的组成符号**组成,如表 5-1 所示。

表 5-1 模块结构图的组成符号

图 例	说 明
	表示模块,方框内注明模块的名称
	表示模块之间的调用关系
	表示模块调用过程中传递的信息,箭头上标明信息的名称。箭头尾为○表示传递数据信息,若为●则表示传递的是控制信息
	表示模块 A 选择调用模块 B 或模块 C
	表示模块 A 循环调用模块 B 和模块 C

在图 5-3 中,图 5-3(a)表示模块 A 和模块 B 的调用关系;图 5-3(b)中一个模块调用

另一模块时，调用模块将数据或控制信息(指令)传送给被调用模块使其运行，而被调用模块在执行中又将它产生的数据或控制信息回传给调用模块；图 5-3(c)中，在模块 A 的箭头尾部标一个菱形符号，表示模块 A 可以“有条件地调用模块 B”。调用箭头尾部标一个弧形符号，表示模块 A 可以“反复(循环)调用模块 C 和模块 D”。



(a) 模块调用关系　(b) 模块间的接口表示　(c) 模块的条件及循环调用

图 5-3　模块结构图

注意：有的模块结构图对“数据信息”和“控制信息”不加以区别，一律用注有信息名的短箭头→统一表示调用关系。

【案例 5-2】　应用软件中常用的一个打印报告的模块结构图。通常的系统调用次序为上层调用下层，同层按照数据传递关系确定，一般从左到右执行。执行过程即按照数据流向进行，如图 5-4 所示。

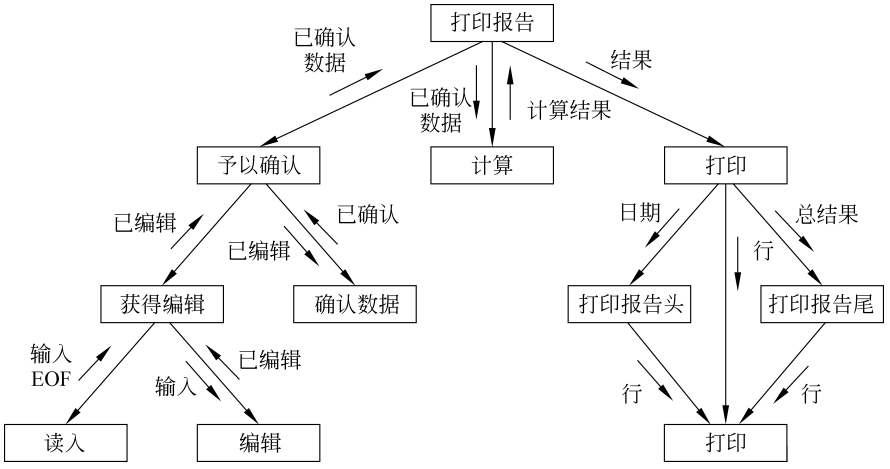


图 5-4　打印报告模块结构图

此层次结构表明：当较复杂事务无法一次完成时，应分解为多层，逐层完成。

(3) 软件层次结构。软件结构图表示软件的系统结构，是软件模块间关系的表示，软件之间的各种关系，均可表示为层次结构。常用层次结构如图 5-5 所示。有关指标如下。

① 深度。表明模块之间控制的层数，软件的控制程度越深，软件越复杂。

② 宽度。表示同一层次上模块的总数，宽度越宽，表示软件越复杂。

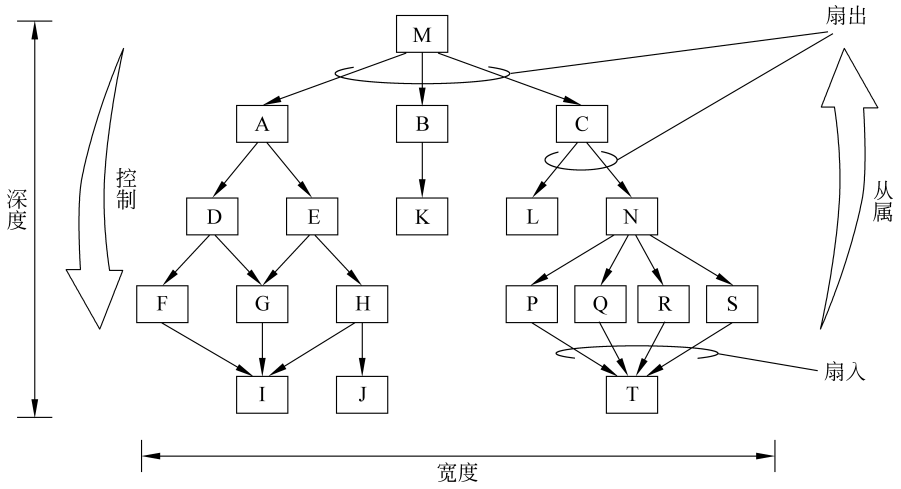
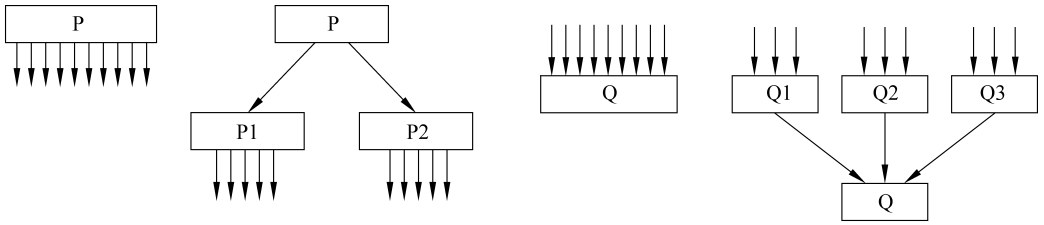


图 5-5　常用层次结构

③ 扇出。表示模块直接控制其他模块的数量。扇出数太大，表明此模块需要协调的下级模块过多，控制过于复杂，可适当增加中间层次的控制模块，如图 5-6(a)和图 5-6(b)所示。模块划分时，一般扇出数平均为 3～4，其上限为 5～8。



(a) 过多的模块扇出　(b) 改善后的模块扇出　(c) 过多的模块扇入　(d) 改善后的模块扇入

图 5-6　软件扇出和扇入控制

④ 扇入。表示模块直接受到其他模块的所控制个数，扇入数越大表明共享该模块的上级模块数越多，虽有一定好处，但不宜片面追求高扇入，如图 5-6(c)和图 5-6(d)所示。好的软件结构形态准则是：顶部宽度小，中部宽度大，底部宽度次之；顶部扇出数较高，底部扇入数较高。

注意：分解一个软件系统得到最佳的模块组合很重要。通常设计好的模块结构，其顶层扇出数较高，中层扇出数较少，底层模块高扇入。

5. 信息隐藏

信息隐藏主要是指模块所包含的“过程及数据”信息，对于其他模块需要隐藏。**模块规定和设计应遵从**：使包含在模块中的“过程或数据”信息，对于其他不需要这些信息的模块，不能访问或“不可见”。其原理有助于软件的设计、实现、测试和维护，由于对于软件的其他部分，绝大多数数据和过程都是隐藏的，可使编程或修改期间因疏忽所造成的影响

只局限于一个或几个模块内，不至于扩大到软件的其他部分。但信息隐藏并非模块内的信息不可用，而是指模块之间的信息传递只能通过合理的调用接口进行实现。

知识拓展
信息隐藏的作用及应用

5.2.3 软件总体设计的方法

软件设计方法从系统设计的角度出发可以**分为三大类**：一是面向数据流的设计，也称为结构化设计方法或过程驱动设计；二是面向数据结构的设计，也称为数据驱动的设计；三是面向对象的设计，具体内容已在第 3 章单独介绍。

1. 结构化设计方法

1）结构化设计方法概述

结构化设计(Structured Design，SD)方法是一种典型的结构化开发方法，是面向数据流的设计方法的核心和关键，主要完成软件系统的总体结构设计。

软件具有层次性和过程性特征：软件的层次性反映了其整体层次结构性质，常用结构图表示。而过程性则反映了其局部性质，常用框图等表示。SD 法**分为总体设计和详细设计两个阶段**。

(1) 总体设计。总体设计过程要解决系统的模块结构问题，确定系统模块的层次结构。在软件设计过程中，总体设计是关键，决定了系统结构、数据结构以及软件的质量，反映了系统的概貌。SD 法的**总体设计步骤**如下。

① 从 DFD 图(数据流图)导出初始的模块结构图。DFD 图描述了系统对数据的处理过程，确定 DFD 图与结构图之间的关系并将其转换为初始的模块结构图。

② 改进初始的模块结构图。为了将系统设计成由相对独立、单一功能的模块组成的结构，应按“高内聚、低耦合”的设计总则，改进初始模块结构图，获得新系统最终的模块结构图。

(2) 详细设计。此阶段的主要任务是：对模块图中每个模块的过程进行描述。常用的描述方式有流程图、N-S 图、PAD 图等。

2）结构化开发方法的设计过程

结构化开发方法的目标是确定设计软件结构的一个系统化的途径和过程。在软件工程的需求分析阶段，数据流是描述的一个关键因素，通常用数据流图描绘数据在系统中的加工和流动情况。面向数据流的设计方法定义了一些不同的“映射”，利用这些映射可将数据流图变换成软件结构。由于任何软件系统都可用数据流图表示，所以面向数据流的设计方法理论上可设计任何软件的结构。面向数据流方法的设计过程如图 5-7 所示。

SD 方法总体设计过程需要从 DFD 图导出初始的模块结构图，先要分析 DFD 图的类型，对不同类型的 DFD 图，采用不同的技术将其转换为初始的**模块结构图**(Structured Chart，SC)。常将 DFD 图分为两种典型的类型：**中心变换型和事务处理型**。

(1) 中心变换型(Transform Center)。这类数据流图可看成是对输入数据进行转换而得到输出数据的处理过程。如图 5-8 所示，该类图的特点是：DFD 图可以明显分为“输

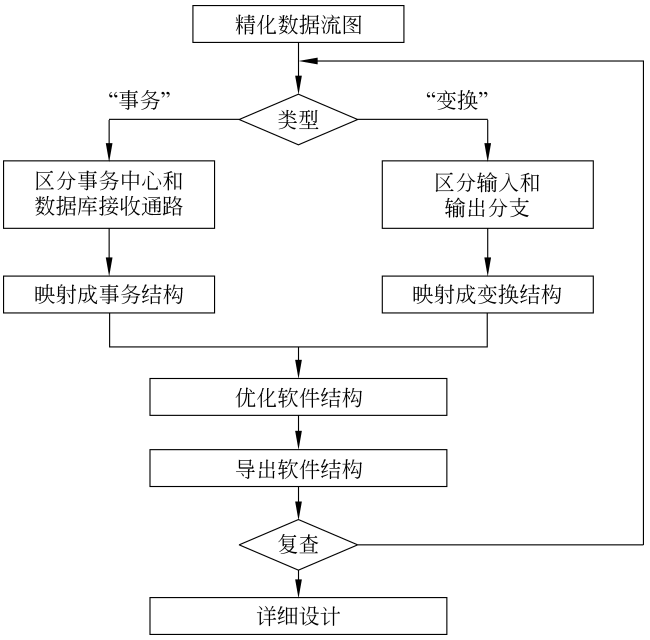


图 5-7　面向数据流方法的设计过程

入—处理—输出”3 部分，对这种类型的 DFD 图的转换采用变换分析技术。

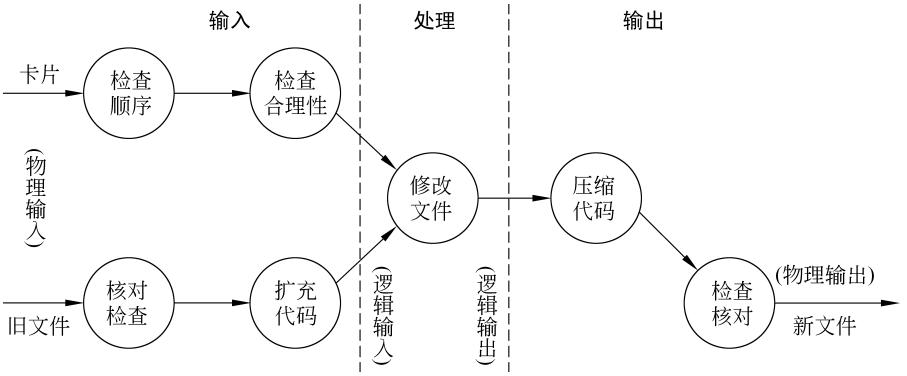


图 5-8　中心变换型

通过这种变换分析技术，可以将中心变换型的 DFD 图进行转换得到结构图(SC 图)，称为变换型的系统结构图。可以得到相应的业务数据、变换数据、给出数据。系统的结构图由输入、中心变换和输出 3 部分组成。

知识拓展

变换分析建立软件结构的步骤

(2) 事务处理型(Transaction)。**事务**是完成作业处理的最小单元。**事务处理型数据流图**如图 5-9 所示，输入流在经过某个“事务中心”时接收输入数据并分析确定其类型，然后根据所确定的类型为数据选择其中的一条加工路径。

对于具有事务型特征的 DFD，采用**事务处理设计方法的步骤**如下。

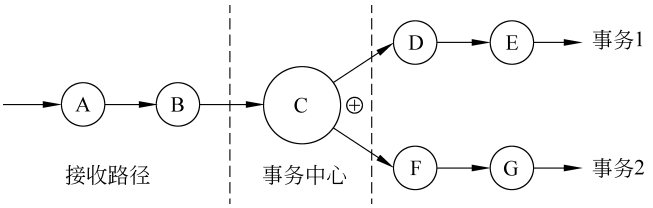


图 5-9　事务处理型数据流图

① 确定 DFD 中的事务中心和加工路径。当 DFD 中的某个加工具有明显地将一个输入数据流分解成多个发散的输出数据流时，该加工就是事务中心。从事务中心辐射出去的数据流为各个加工路径。确定流界时，先从 DFD 中找出事务流、事务处理中心和事务路径。

② 进行一级分析，设计上层模块。对事务中心应设计“事务控制”模块；对事务流应设计“接收事务”模块；对事务路径，应设计“发送控制”模块。

③ 进行二级分解，设计中下层模块。对于接收分支，用类似于转换处理型数据流图中的方法，对输入数据流的方法设计中下层；对于发送分支，在发送控制模块下为每条事务路径设计一个事务处理模块，这一层称为事务层。

图 5-9 对应的事务型软件结构图如图 5-10 所示。

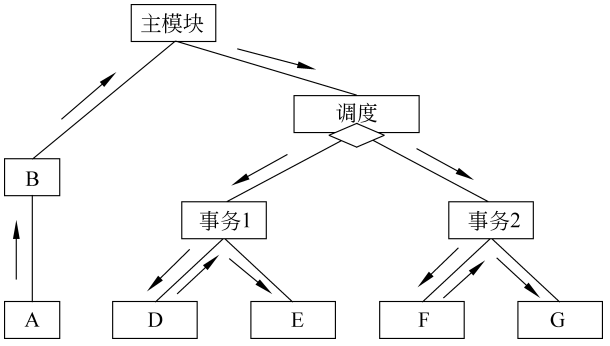


图 5-10　事务型软件结构图

在中型以上系统的 DFD 中，通常既有变换流又有事务流，即混合的 DFD，其软件结构设计方法常用以变换流为主、事务流为辅的方法，其**步骤**为：确定 DFD 整体上的类型；标出局部的 DFD 范围，确定其类型；按整体和局部的 DFD 特征，设计出软件结构。

【案例 5-3】 利用 SD 法将修改贷款文件的 DFD 图转换为模块结构图的过程如下。

(1) 对 DFD 图，确定对应的主加工及逻辑输入和逻辑输出，如图 5-11 所示。

(2) 画出系统模块图的顶层及第一层模块。

(3) 分解中下层模块。对第一层各模块继续向下分解，一直分解到不能再分的功能模块。在图 5-12 中，对输入部分进行分解，不再画出输出和处理部分的分解。