第3章 结构化分析

与所有的软件开发过程一样，软件开发团队在开发软件之前必须对软件的需求进行分析，并严格按照用户给出的需求内容来完成软件的设计和实现。

高级语言的出现让开发计算机程序变得越来越方便，软件开发人员可以根据自己的思维模式来创作软件。与结构化分析相关的概念最早出现于 20 世纪 60 年代后期，Douglas Ross 阐述了如何使用结构化思想来分析软件需求。但是，直到 20 世纪 70 年代末 Demarco 设计了结构化分析的建模方法和描述工具之后，结构化分析方法才被逐渐用于软件开发中。

结构化分析就是指采用结构化的方法对目标软件需求进行描述的过程。由于结构化程序设计主要跟踪数据的处理过程和处理流程，结构化分析中通常采用软件的功能模型、数据模型和行为模型来建模用户需求，即采用数据流图（Data Flow Diagram，DFD）、状态转换图（State Transition Diagram，STD）和数据字典（Data Dictionary，DD）等方式对软件需求进行模型化描述。

在结构化分析中，功能模型用于定义目标软件系统应当完成的功能；行为模型表示目标软件系统对外部事件响应的系统行为；数据模型用于理解和表示问题的信息域。

3.1 结构化需求描述方法

在结构化程序设计中，软件系统根据业务流程对输入软件系统的数据进行处理。输入数据经过多个函数的“处理”以后，再输出系统。可以发现，在结构化程序设计中，软件开发人员可以将数据处理功能封装为“函数”，将复杂的处理分解为更小的函数，通过业务功能组合来实现要求的数据处理过程。

因此，在使用结构化方法描述用户需求时，需求分析人员必须关注数据在系统内部的处理过程，即数据从输入到输出的过程中经过了哪些“处理”以及处理的对象是什么，对象的内容是什么。

根据人类语言常识，可以认为：关于处理的动作一般采用“**动词短语**”来表示，例如，均分苹果，打印文件，显示结果，休息等，且动词短语可以由“动词+宾语”和“动词”两种情况组成。

（1）动词+宾语

“动词+宾语”表示了一个完整的动作及动作处理的对象，即动词表示需要对宾语处理的动作，宾语表示被处理的对象。在进行需求分析时，需求分析人员可以将动词理解为数据处理的功能，宾语表示被处理的数据。

（2）动词

单一的动词可以表示目标软件系统在处理该动作时，仅需要进行一些内部处理，或者进行一些不需要输入操作的动作。

在结构化需求分析中，为了准确地描述数据在业务系统中的处理流程，需求分析人员必

须重点关注用户需求描述中的动词以及动词处理的数据对象。

本书以一个简单的、类似火车订票系统的软件为例来介绍结构化需求分析过程。为了简单示例，假设该系统仅有一个用户操作订票软件，暂不考虑软件并发需求。与此同时，该案例屏蔽了软件与其他软件系统进行通信的细节。

假设需求分析人员在与用户进行沟通时获得了以下的需求描述内容。

用户进入订票系统后，系统将向用户显示订票软件的操作界面。

软件界面中显示了软件名称、系统软件 Logo、系统通知信息、当前时间等。同时，系统提供输入用户名和密码的登录方式，允许注册用户填写注册信息进行注册。

未注册用户可以在软件界面上选择注册功能，进行注册。用户选择注册功能以后，系统将进入软件注册界面。用户在注册界面上填写合法的注册信息后，提交系统检验。检验合格后，完成注册过程。

订票系统提供车次录入功能。管理员可以登录系统后台，录入车次信息。车次信息包括车次、始发地、目的地、实发时间、车票数量信息等。系统将保存录入的车次信息。

用户进入系统后，系统将显示订票提示信息。同时，界面提供用户输入想要订票的车次信息和出行日期。用户也可以输入始发站和终点站信息来查询满足要求的所有列车。

用户确认订票车次和日期后，系统将从后台数据库中寻找满足用户需求的车次信息，并通过列表方式进行显示。列表由多个列车信息项组成。列车信息项中显示了车次信息、出发时间、到站时间、可订座位数量信息和票价。

用户选择好想要预订的车次后，单击“预订”按钮进行车票订购。此时，订票客户端将向系统提交订票请求。系统验证订票信息以及车次信息有效性后，向用户反馈订票结果。如果订单有效，系统将要求用户进行在线支付；如果订单无效，系统将向用户提示订票失败。

网络支付页面显示当前需要支付的订单信息。订单信息包括车次信息、出发时间、到站时间、订单价格。同时，为了方便用户支付，系统将显示当前支持的多种支付方式。用户选择特定的支付方式进行支付，系统将提供订单支付页面。用户在支付页面填写支付信息后，系统将支付信息提交给银行审核。银行审核支付信息合法后，系统提示订票成功，系统修改后台的车次、坐席、数量信息，同时提醒用户订票成功；银行审核支付信息无效时，系统提示当前订票支付失败。

从上述需求可以得出：该软件涉及的数据主要包括系统软件、用户身份信息、车次信息、订票信息和银行卡信息五个方面。

依据前面论述的需求分析方法可以知道，在采用结构化分析方法建模软件需求时，需求分析人员需要重点把握最初的输入数据，并跟随输入数据寻找数据在软件系统中被处理的过程，即需求分析人员需要重点对输入数据的处理过程进行关注。

需求分析人员首先从需求描述中挑选出所有的“动词+宾语”和“动词”；其次，由于此时的需求描述只是对目标软件系统的需求进行一般性概述，在需求描述中可能包含了大量与目标软件系统无关的内容。因此，需求分析人员可以对选取的动词短语进行分析，去除所有与本项目无关的动词；最后，需求分析人员可以对剩余的动词短语进行过程排序，追踪数据进入系统后被处理的过程。

通过以上处理过程，需求分析人员对文字描述的需求进行了再次整理，可以得到案例需求中对上述五种数据的主要处理动作如下。

（1）系统

显示订票软件操作界面、保存车次信息、显示软件信息、提供登录信息、提供注册信息、

显示订票提示信息、验证订票信息、显示支付方式、提示订票结果。

（2）用户身份信息

用户注册、用户登录、用户身份验证。

（3）车次信息

录入车次信息、保存车次信息、查询车次、列举车次信息。

（4）订票信息

订购车票。

（5）银行卡信息

审核支付信息。

除了获取需求分析中所有的“处理”和“处理对象”以外，需求分析人员还需要关注需求描述中各个“处理”过程的次序和先后关系，即数据在系统中被处理的过程。

与此同时，在撰写需求内容时，需求分析人员还需要对动词处理的对象“宾语”进行重点关注，即对各个数据项目的内容组成和内容格式进行详细描述。同样以上述订票软件为例，需求分析人员必须对系统中涉及的五种数据对象的内容和格式进行记录。例如，用户身份信息由姓名、出生日期、性别、身份证号、用户名、密码等信息组成；其中，姓名由 2～10 个汉字组成。

在描述了系统需求涉及的“动词”和“宾语”后，如果软件系统在处理数据过程中涉及状态变化，则需求分析人员还应该对系统在运行中的处理状态进行区别，对系统在不同状态下的行为进行描述。同样以上述订票软件为例，在订票环节中就涉及是否有剩余车票这个状态。如果选定车次的车票还有剩余，则系统进入订单支付状态；相反，如果选定车次的车票已经售罄，则系统将提示订票失败。

可以发现，在以结构化方法对目标软件系统进行需求描述时，需求分析人员可以从“动词短语”“宾语的内容及组成”“动词的处理顺序”以及“系统状态”四个方面对业务内容进行说明，对用户给出的处理需求进行准确、完整的描述。

3.2 结构化需求建模

尽管 3.1 节已经对如何描述用户的需求进行了介绍，但是，由于自然语言在描述事物时往往存在随意性和二义性等问题，文字描述的软件需求可能会对后期的软件设计带来障碍，影响软件设计人员对需求内容的理解。

为了准确地描述用户需求，便于软件开发团队理解问题，需求分析人员可以采用模型化方式对目标软件的需求内容进行无歧义的描述。所谓模型是指人们为了理解事物而对事物做出的抽象，即模型是人们对于某个实际问题或客观事物、规律进行抽象后的一种形式化表达方式。通常，模型由一组图形符号和组织这些符号的规则组成。

结合上述需求分析所要描述的内容，需求分析人员在进行结构化需求建模时，可以分别使用数据流图、数据模型、数据字典、处理/加工逻辑说明和状态转换图等建模工具对需求中数据处理的次序、数据处理的对象和对象内容、处理对象的内容组成、处理数据的过程，以及系统在处理数据时的状态进行建模。

3.2.1 数据流图

数据流图，顾名思义，是对数据的流向进行描述的建模工具。数据流图以图形方式来表

达数据在系统内部的逻辑流向和逻辑变化过程，可用于对系统内部的数据处理过程建模。

在绘制数据流图时，需求分析人员可以根据对需求的理解程度，采用分层方式描述数据处理的细节，即在需求分析初期，需求分析人员可以采用概述方式对数据流进行建模；随着需求分析人员对问题理解的深入，顶层数据流图中的处理可以被分解，并进一步拆分为更细致的数据处理过程，形成细化后的下层数据流图。通过分层数据流图，需求分析人员可以从整体到局部对目标软件系统处理数据的流程进行建模，准确描述数据在系统内部的处理过程，帮助软件开发团队对数据的处理过程有一致的理解。

因此，对于根据数据处理过程来设计程序的结构化方法而言，数据流图除了是一种描述系统处理数据逻辑的图形工具以外，还可以清楚地说明系统的组成部分以及各个组成部分之间的数据联系。

1. 数据流图表示

由于数据流图采用抽象的图形符号来表现数据在软件中的流动和处理的过程，如何定义数据流图中的符号至关重要。目前，主流的数据流图图示符号有 Yourdon 和 GaneSarson 两种体系，且每种体系都由处理、数据存储、外部实体和数据流四种符号组成。

尽管 Yourdon 和 GaneSarson 两种体系非常成熟，但是，目前支持这两种体系的绘图工具较少。因此，本书以主流的矢量图绘图软件 Microsoft Office Visio 来介绍数据流图。数据流图的主要符号如图 3-1 所示。

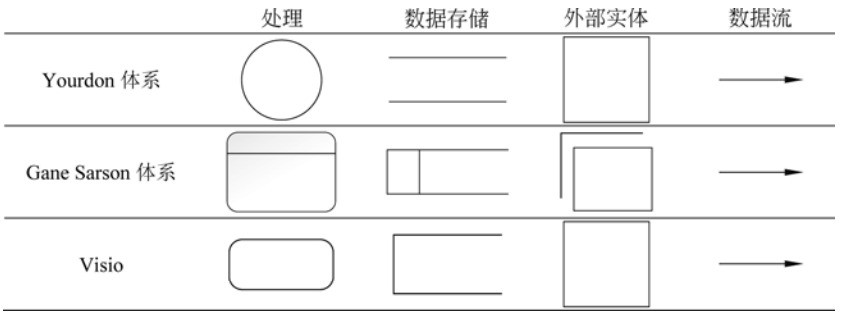


图 3-1　数据流图符号

在数据流图中，处理也称为加工，即数据的处理或变换。处理可用于表示目标软件系统中对数据进行处理的逻辑单元。一般而言，处理的名字对应着与之相关的“动词短语”，表述处理的动作。

外部实体一般简称为实体，也可以称为数据源点或终点、外部对象等。外部实体用于表示数据的外部来源和输出，也可以是软件之外的其他系统。

数据存储在数据流图中表示对数据的存储。数据存储可以是系统中的文件、数据库等，也可以是系统中的表单、账本等。数据存储一般用“名词”或“名词短语”命名，可以在备注中说明数据存储的介质或者物理设备，便于软件开发人员理解。

数据流图采用带标识的箭头来表示数据流，表示系统处理数据对象的流动方向。一般而言，数据流由一个或者一组确定的数据项组成。数据流可以连接两个处理，也可以连接处理和数据存储，或者连接处理和外部实体，表示数据在系统中的传递过程。

除此以外，需求分析人员在绘制数据流图时应遵循以下原则。

（1）数据流的实体和存储之间至少需要一个处理过程。

数据流不能从外部实体直接流向其他外部实体，也不能直接流向数据存储；同样，数据

流也不能直接从数据存储流向其他数据存储或外部实体。

（2）数据流必须标出名字，且名字必须反应出数据流中的数据内容。

由于数据流表示数据进入处理环节或离开处理环节，数据流的命名必须体现进入或离开时的细节。

（3）数据流的名字不允许重复。

从（2）中可以知道，数据流用于建模参与处理的数据。即使不同数据流在结构上是相同的，但是由于数据流处于不同的处理阶段，其内容和状态已经发生了变化。因此，在绘制数据流图时，需求分析人员必须为数据流设定不同的名字来表述具体的数据内容，对数据的内容进行准确定义。

（4）数据流描述被处理的数据，不应体现具体的处理控制内容。

数据流中应当体现处理的数据内容细节，即流入、流出处理的内容细节。对数据进行的处理应该封装到处理中，数据流中无须体现。

2. 数据流与处理之间的关系

在复杂的数据流图中，处理之间可能不仅仅只有一条数据流，即处理可能会产生多个数据流输出，或者处理需要多个数据流的输入。

为了表示处理之间多个输入数据流或者输出数据流之间的逻辑关系，数据流图中增加了额外的标记符号来表达处理与数据流以及数据流之间的关系。在数据流图中，符号“*”表示相邻数据流之间存在“与”关系，即流入或流出处理的多个数据流必须同时存在；符号“+”表示相邻数据流之间存在“或”关系，即流入或流出处理的数据流可以存在其中一个，或者多个数据流同时存在；符号“⊕”表示相邻数据流之间存在“异或”关系，即流入、流出处理的数据流中只能取其中一个，多个数据流不能同时存在，如图 3-2 所示。

图 3-2　数据流和处理之间关系的符号表示

3. 数据流图分层

对于大型的软件系统而言，由于涉及的业务逻辑比较复杂，需求分析人员不可能马上明确系统需求，也不可能一次性将全部的处理和数据流绘制到一张完整的数据流图中。此时，

需求分析人员可以采用分层数据流图对系统的业务流程进行抽象、建模。

一般而言，分层数据流图包括顶层数据流图、中间层数据流图和底层数据流图三层，如图 3-3 所示。图中，L0 表示顶层数据流图，L1 表示中间层数据流图，L2 表示底层数据流图。可以看到，下层数据流图是对上层数据流图的进一步细化。

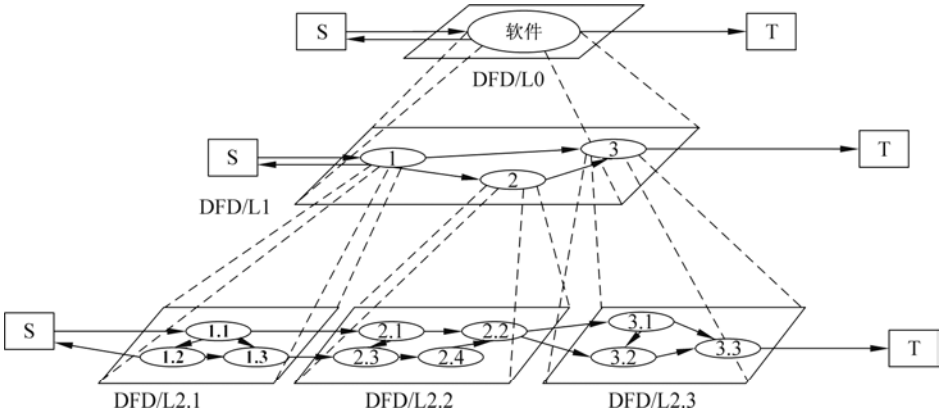


图 3-3　分层数据流图案例

在分层数据流图中，顶层数据流图用于描述整个软件系统的作用范围。此时，需求分析人员可以将整个目标软件系统看成顶层数据流图的处理，而处理的输入和输出描述了系统与外部环境的接口。

中间层数据流图是对顶层数据流图的分解，可以认为是顶层数据流图的功能划分。

底层数据流图是对中层数据流图的再次分解、细化。底层数据流图由一些不能再分解的处理和简单数据流组成。当然，如果底层数据流图仍然复杂，需求分析人员可以对底层数据流图进行再次细分。底层数据流图的处理应当功能独立，简单明确，且数据流被严格定义。

同时，为了清晰地表明处理之间的层次关系，需求分析人员可以采用多级编号方式来索引数据流图中的各个处理。通常，分层数据流图的顶层称为第 0 层，它是第 1 层的父图；而第 1 层既是第 0 层的子图，又是第 2 层的父图。此时，第 1 层中的处理可以采用编号 1、2、3…来进行标记；由于第 2 层中的处理为第 1 层中数据处理的分解和细化，需求分析人员可以用下一层的编号来标记拆分后的子处理。例如，第 1 层中处理 1 的子处理在第 2 层中可以用编号 1.1、1.2、1.3…来标记，以此类推。

随着需求分析人员对问题理解的不断深入，需求分析人员可以逐层绘制数据处理的具体细节。通过对目标软件系统中的“处理”进行分层描述，分层数据流图可以采用由外至内、自顶向下、逐层细化的方式对复杂软件系统的数据处理关系和流程进行建模。

可以发现，数据流图的分层思想体现了需求建模的抽象和信息隐藏，即上层处理不需要考虑下层处理的细节，暂时掩盖了下层加工的功能以及加工之间的复杂关系。

值得注意的是，在绘制分层数据流图时，需求分析人员还必须注意父图与子图的平衡，即保持父图与子图的输入数据流和输出数据流一致，确保父图中的输入、输出数据流也是子图的输入、输出数据流。如果数据流在子图中得到了细化，需求分析人员需要在数据字典中对分解的数据流进行准确描述。

同样以上述订票软件为例，在进行首次需求调研时，需求分析人员可以将其业务过程抽象为图 3-4 所示的数据流图。

图 3-4　L0 级车票订购业务数据流

随着需求分析人员对订票流程的进一步理解，可以发现图 3-4 中的“订票”处理可以分解为“查询车次”和“订购车票”两个子处理，如图 3-5 所示。

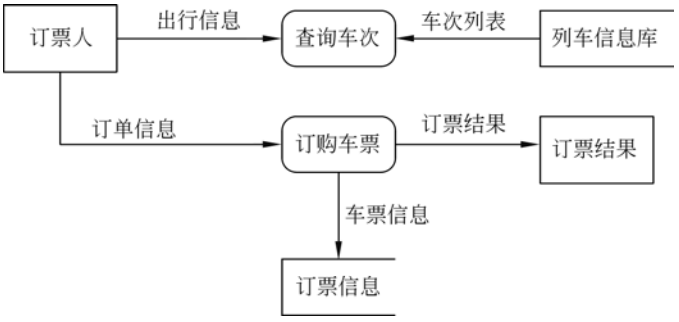


图 3-5　L1 级车票订购业务数据流

接着，需求分析人员可以根据业务流程对“订购车票”处理进行继续细化，分为“查询余票”与“生成订单”两个子处理。其中，“生成订单”处理的输出中，“成功信息”和“失败信息”两个数据流为异或关系，如图 3-6 所示。

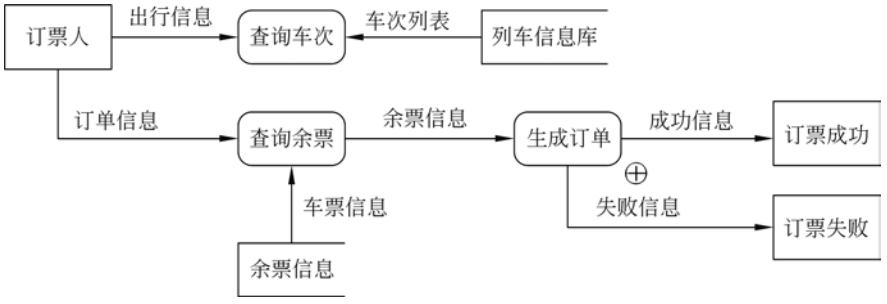


图 3-6　L2 级车票订购业务数据流

通过分层数据流图，需求分析人员可以将火车订票业务逐层细化，对订票的业务处理流程进行详细建模。

可以发现，需求分析人员可以借助数据流图，对需求描述中的数据处理顺序和处理之间传递的数据内容（数据输入和数据输出）进行准确建模。同时，需求分析人员可以采用分层数据流图来建模复杂的业务处理过程和数据内容，实现对复杂业务需求内容的精准建模。

3.2.2　处理/加工逻辑说明

在需求分析过程中，需求分析人员可以借助数据流图对目标软件处理数据的过程进行建模。然而，即使需求分析人员将数据流图分解到了最低层次，“处理”的概念仍然很抽象，无法具体到数据的实际操作过程。

处理/加工逻辑说明对数据流图中出现的“处理或加工”进行详细描述，即描述处理是“做什么”和“怎么做”的。具体而言，处理/加工逻辑说明对指定处理的数据处理过程、处理逻辑进行说明，描述处理如何将输入数据转变为输出数据的加工规则。同时，处理/加工逻辑说

明还包括其他与处理相关的信息，如处理的执行条件、优先级、执行频率和出错处理等。

一般而言，数据流图中的每一个处理都应有一个处理/加工逻辑说明。处理/加工逻辑说明应当完整、严密、易于理解。

目前，需求分析人员主要采用结构化语言、判定树和判定表三种形式描述处理的业务逻辑。

1. 结构化语言

结构化语言（Structured Language），也称为问题描述语言（Problem Describe Language，PDL），是一种专门用于描述功能单元逻辑要求的语言。结构化语言不同于自然语言，也区别于任何特定的程序设计语言。它是在自然语言的基础上增加一些控制方式而得到的一种介于自然语言和形式化语言之间的半形式化语言。

在结构化语言中出现的词汇由命令动词、数据字典中定义的名字、有限的自定义词和逻辑关系词等内容组成。

同时，结构化语言采用内层和外层两种语法来描述处理的内部加工逻辑。

1）外层语法

结构化语言的外层语法用于描述操作的控制结构，如顺序、选择和循环等。

（1）顺序结构：采用简短的语句对多个连续的业务处理进行描述，避免使用复合语句。

（2）判定结构：采用 IF THEN ELSE 或 CASE OF 来组织多个可选动作。

（3）循环结构：采用 WHILE DO 或 REPEAT UNTIL 表示动作的多次重复。

结构化语言通过控制结构将处理/加工中的多个操作连接起来，用于对业务处理过程进行宏观说明。

2）内层语法

结构化语言的内层语法一般采用自然语言来表述具体的处理逻辑。值得注意的是，为了减少结构化语言描述处理/加工时可能导致的歧义，需求分析人员在内层语法中使用的动词必须具体（避免使用抽象的动词），并且，动作描述中尽量不使用形容词和副词。

以一个评估学生期末成绩的处理为案例，该处理对应的结构化语言描述可能如下：

```
IF 成绩 >= 85 THEN
   该生成绩等级为 A
ELSE
   IF 期末分 >= 75 THEN
       该生成绩等级为 B
  ELSE
      IF 期末分>= 64 THEN
          该生成绩等级为 C
      ELSE
          IF 期末分>= 60 THEN
             该生成绩等级为 D
          ELSE
             该生成绩等级为 F
          ENDIF
      ENDIF
   ENDIF
ENDIF
```

2. 判定树

当处理中存在复杂的判断逻辑时，需求分析人员可以借助判定树（Decision Tree）来表达处理在不同条件下的行为方式。

判定树的左边为树根，从左向右依次排列各个条件；在设计判定树时，需求分析人员可以根据条件的取值不同产生各类分支。判定树各分支的最右端（即树梢）为处理在不同条件取值状态下所采取的行动（也称为策略），判定树优先考虑左边的条件，如图 3-7 所示。

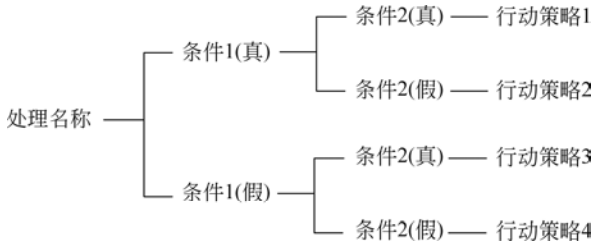


图 3-7　判定树结构

以一个假定的行李托运费计算处理为例，其判定树可能如图 3-8 所示。

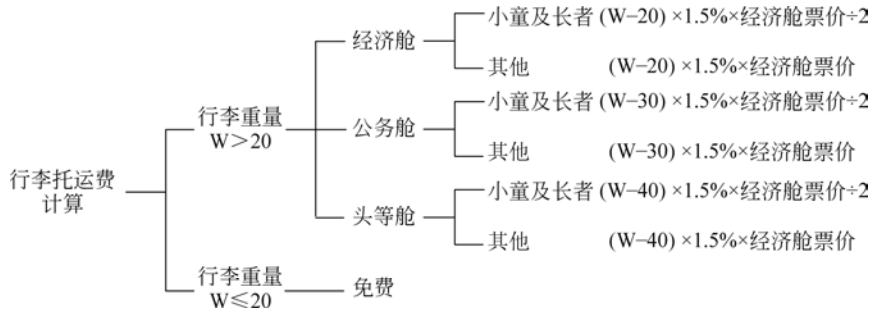


图 3-8　行李托运费计算判定树案例

判定树的优点在于形式简单，易于掌握和使用。然而，判定树的缺点在于简洁性较差，相同的行为方式会在判定树中多次重复，且越接近叶端，重复的次数越多。与此同时，判定树分枝选择的条件次序可能会对最终完成的判定树的简洁程度造成较大影响。

3. 判定表

判定表（Decision Table）是需求分析中的另一种用于表达复杂逻辑判断的工具。当处理中包含多重嵌套的条件判断时，判定表可以清晰地表示复杂的条件组合与应做动作之间的对应关系。

判定表由条件桩、动作桩、条件项和动作项四个部分组成，如图 3-9 所示。

条件桩	条件项
动作桩	动作项

图 3-9　判定表结构

（1）条件桩（Condition Stub）列出问题的所有判断条件，条件的先后次序无关紧要。

（2）动作桩（Action Stub）列出处理所有可能采取的操作行为。同样，操作行为的排列顺序没有约束。

（3）条件项（Condition Entry）列出处理可能会遇到的多个条件取值的组合，即多个条件组合得到的真假值列表。

（4）动作项（Action Entry）列出处理在某种条件取值组合情况下采取的操作行为。

同样以计算行李托运费为例，可以得到该处理对应的判定表如图 3-10 所示。

	1	2	3	4	5	6	7
小童及长者		T	F	T	F	T	F
经济舱		T	T	F	F	F	F
公务舱		F	F	T	T	F	F
头等舱		F	F	F	F	T	T
W≤20	T	F	F	F	F	F	F
免费	√						
(W－20)×1.5%×T÷2		√					
(W－20)×1.5%×T			√				
(W－30)×1.5%×T÷2				√			
(W－30)×1.5%×T					√		
(W－40)×1.5%×T÷2						√	
(W－50)×1.5%×T							√

图 3-10　行李托运费计算判定表案例

与结构化语言和判定树相比，判定表的优点在于能够将所有条件组合充分地表达出来；判定表的缺点是建立过程比较繁杂，且表达方式不如结构化语言和判定树简便。

在对处理的加工逻辑进行描述时，需求分析人员可以综合利用结构化语言、判定树和判定表，争取对业务的处理动作和逻辑进行无歧义建模。

3.2.3　状态转换图

在日常生活中，状态是指某个物体所处的情况。而在数据处理过程中及在软件系统内部，状态则是指决定程序分支走向和循环量的关键变量值。当软件系统处于不同的状态时，或者处理中的状态发生改变时，软件对数据的处理和动作也都将随之改变。因此，在对目标软件系统开展需求分析时，需求分析人员除了对数据的处理流程、处理的内部细节进行建模以外，还必须对软件系统的状态以及软件系统在不同状态下对外部事件、数据输入的响应行为进行建模。

状态转换图（State Transform Diagram）是一种用于描述系统对内部或者外部事件响应的行为模型，它描述了系统的各种行为模式（称为“状态”）以及系统在事件作用下的状态转换，即状态转换图是针对系统状态以及系统状态变化的描述。

状态转换图由状态和事件两种元素组成。

1. 状态

状态是指软件系统在运行中的不同阶段，或者软件系统任何可以被观察到的系统行为。直观而言，状态就是指影响软件系统数据流图走向，或者影响处理内部动作流程的关键变量取值。

状态转换图中设置了初态（初始状态）、终态（最终状态）和中间状态三种符号，且每节点表示一个系统状态，如图 3-11 所示。初态采用实心圆或者空心圆表示；终态采用双圈表示；而中间状态使用圆角矩形表示。状态可以标注状态的名称、状态的变量标志和动作。

同时，需求分析人员可以利用中间状态中提供的进入（entry）、执行（do）和退出（exit）三种内部事件来细化状态行为，如图 3-12 所示。进入事件用于指定软件系统在进入该状态时需要执行的动作；执行事件表示系统在该状态下的行为；而输出事件用于指定软件系统退出

该状态时需要执行的动作。

图 3-11　状态转换图符号

图 3-12　中间状态标准事件

值得注意的是，软件系统对应的各个状态转换图中有且仅有一个初始状态。然而，根据软件系统的运行结果不同，各个状态转换图中可以设置 0 到多个终态。

2. 事件

通常而言，事件是对引起软件系统动作或系统状态发生改变的外部动作的抽象，即事件是引起系统做动作或（和）状态转换的控制信息。

事件的表达式语法如下：

事件名(参数表)［守卫条件］/ 动作表达式

其中，事件名是用户与需求分析人员对事件的命名；参数表中列出了事件在触发或者运行时接收的参数；守卫条件表明事件触发时必须满足的条件；而动作表达式则给出了事件在运行过程中伴随的动作。

在状态转换图中，事件采用箭头表示，用于连接状态转换图中的两个状态。事件的箭头指明了状态的转换方向，表示前一个状态经过事件以后转换为后一个状态。事件上可以标记出事件说明、守卫条件和动作表达式等，如图 3-13 所示。

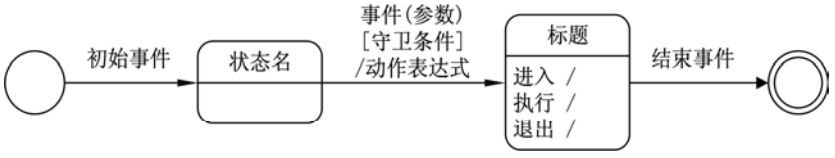


图 3-13　状态图案例

通常，软件系统的状态发生转换可以表示为软件系统对一个动作或者一系列动作的响应，也可以是软件系统本身状态的改变，还可以是动作和状态改变的结合。如果软件系统的状态转换是由事件触发的，需求分析人员必须在引起状态发生转换的事件箭头上标出相应的事件表达式；如果箭头上未标明事件表达式，则意味着软件系统在原状态的内部活动执行完成之后，自动触发转换为后一个状态。

同样以订票系统为例，其订票过程的局部状态转换图可能如图 3-14 所示。订票系统将根据是否有余票对订票结果进行区分处理。

如果软件系统的状态数量较多，系统对应的状态转换图的复杂度也将急剧增加。此时，需求分析人员可以在状态转换图中引入复合状态和子状态，采用分层、嵌套方式来建模复杂软件系统的状态转换关系。复合状态中嵌套了两个或者两个以上的子状态，可用于表示软件系统的宏观状态，如图 3-15 所示；而子状态则可用于对软件系统的微观状态进行建模。

可以发现，状态转换图可以用来标识系统分析与设计中的关键变量信息，有助于软件设计人员结合软件系统的状态变化来设计程序的运行路径。

在需求分析过程中，需求分析人员可以采用以下步骤来绘制系统的状态转换图。

步骤 1：列出产品/系统的所有状态。

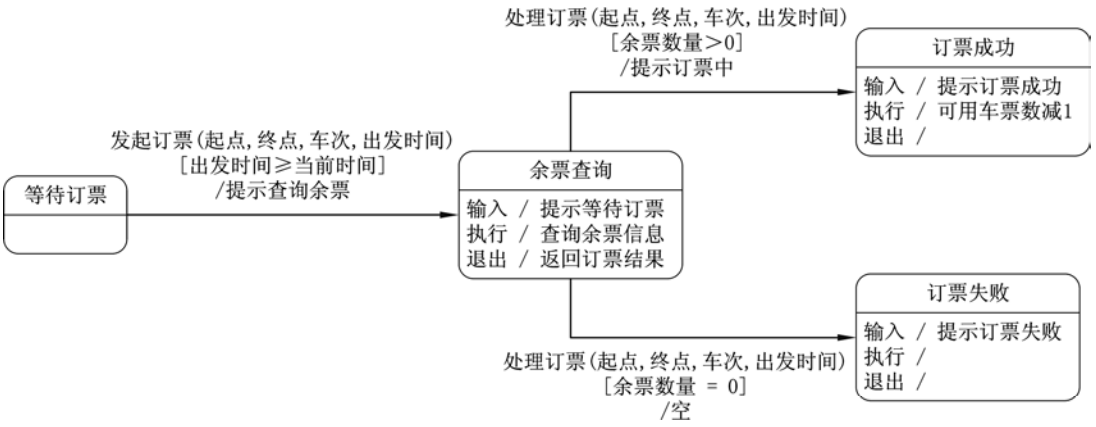


图 3-14　订票系统的状态转换图局部案例

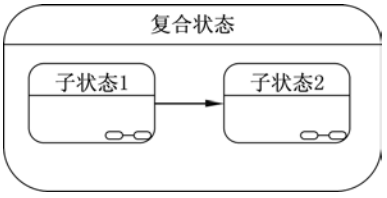


图 3-15　复合状态与子状态

步骤 2：列出每个状态须执行的动作。

步骤 3：确认并绘制出引起状态发生转换的事件。

步骤 4：标注初态和终态，并细化状态转换图。

由于软件系统中的关键变量直接决定了软件处理数据的过程和程序走向，需求分析人员必须对软件系统中的关键变量的变化过程进行跟踪、建模。通过分析触发软件系统关键变量值发生变化的事件以及观察软件系统在特定情况下的行为，需求分析人员可以借助状态转换图对软件系统的状态和行为进行精确建模，确保需求分析的完整性。

3.2.4　数据模型

这里所谓的“数据”是狭义的数据，并非大数据。数据是指输入、输出系统以及在系统各个处理之间传递的信息及其格式。

可以发现，尽管需求分析人员可以借助数据流图对目标软件系统的数据处理过程进行建模，但是，数据流图仅用数据流对待处理的数据进行了笼统表述，缺少对数据流的明确定义。因此，为了准确定义各个处理的输入/输出数据，需求分析人员必须对数据流图中的数据流进行建模，得到目标软件系统的数据模型。

在数据模型中，需求分析人员通过对数据流图中存在的数据内容、数据关系等信息进行描述，确保数据模型的内容能够对各个处理的输入/输出数据以及设计的数据结构进行准确表述。因此可以认为，数据模型通过无歧义的、系统化的表达方式，从含义、结构和组成上对处理的输入/输出数据、数据存储及数据加工进行准确定义。

通常而言，数据模型由数据的结构、数据之间的关系和数据项的组成三部分构成。

1. 数据的结构

在结构化程序设计中，处理可以理解为完成具体功能的函数。而输入/输出处理的数据流

即对应着函数的输入形参和输出结果。

以 C 语言中的函数为例，函数的输入/输出信息中除了简单的原子型数据以外，在涉及数据处理、数据存储或者数据输出等软件功能时，往往存在着特定的数据结构，如结构型数据、文件格式等。

1）结构型数据

结构型数据是指数据中包含了若干属性信息，并且各属性数据依次排列，形成一个整体。数据的各个属性是数据信息在软件处理范围内的具体描述。

为了表达和记录需求中各个实体的相关信息，人们在计算机系统中将需求中的实体抽象为特定的数据结构，对目标软件系统涉及的实体属性进行抽象。此时，需求分析人员仅需关心目标软件系统涉及的信息。例如，“人”在不同的软件系统中存在着不同的属性。在学籍管理系统中，人的属性可能有学号、姓名、性别、出生日期、所属学院、绩点等；在医保系统中，人的属性可能有社保号、姓名、性别、出生日期、社保类型、社保余额等；同时，在订票系统中，人的属性可能有身份证号、姓名、登录号等。需求分析人员可以将实体涉及的属性信息封装为结构型数据。

如果目标软件系统处理的数据涉及用户定义的表格，需求分析人员可以与用户协商，咨询是否存在已有的数据表格，或者与用户共同设计合适的数据表格来表示相应的数据内容。

当然，如果目标软件系统中涉及的数据呈现出特定的结构，如通信信令、机器之间的通信数据等，需求分析人员可以采用表格或者框图形式对目标数据结构进行描述。例如图 3-16 对 ICMP 数据包的结构进行了建模。

图 3-16　ICMP 包结构

2）文件数据格式

当软件系统涉及文件数据时，需求分析人员还必须对文件的数据格式进行建模。

从操作系统角度而言，文件是指计算机为了存储信息而采用的信息编码格式。然而，从程序角度来看，文件是计算机程序按照指定的文件格式或者文件标准保存处理数据的结果。程序可以通过文件存储数据，或者与其他程序交换数据。

当目标软件系统涉及文件读取或者写入时，需求分析人员必须与用户沟通并获取对应的文件格式，或者根据用户要求寻找满足需求的标准文件格式。如果目标文件为特殊文件格式，需求分析人员还需要向用户索取样例文件，便于软件开发人员对文件格式进行解析。

2. 数据之间的关系

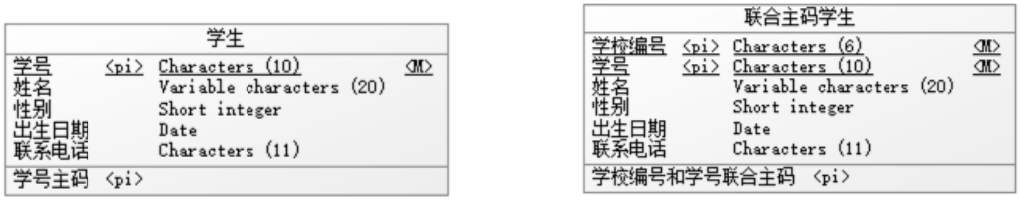
在进行软件设计时，软件设计人员除了需要明确数据的内容组成以外，还必须依赖数据之间的关系来设计目标软件系统获取、处理和存储数据的方式。因此，当目标软件系统涉及处理大量数据，以及数据之间的关系时，例如数据存储、数据查找等，需求分析人员就必须对如何区别数据，以及对数据之间的关系进行建模。

1）数据标识

为了标识大量结构相同的同类数据内容，需求分析人员可以结合实际的软件应用场景，

在数据内容中选择一个或者多个属性的组合来唯一标识数据记录。此时，被选择用于标识数据的属性或者属性组合被称为该数据的标识符（标识符在数据库存储中也可以称为数据的主码、主键）。

例如，如果目标软件系统需要使用一个特定的数据结构来存储仅有少数几个人的小团体人员数据，需求分析人员可以选择人员数据中的“姓名”属性作为该数据结构的标识符；但是，如果目标软件系统需要用某个数据结构来存储人数较多的学校或班级学生人员数据，此时多个学生的“姓名”属性可能会出现重复。此时，需求分析人员可以选择学生信息中的“学号”属性作为该数据结构的标识符。当然，如果目标软件系统需要用一个特定的数据结构来存储国内多所高校的学生数据，且不同高校采用了相同的“学号”编码方式，则学生数据结构中的“学号”属性可能会出现重复。为了唯一标识这些学生数据，需求分析人员可以将数据结构中的“学校编码”与“学号”两个属性组合为标识符，对来自多所学校的学生信息进行唯一标识，如图 3-17 所示。



（a）学号为主码　　（b）学校编码与学号作为联合主码

图 3-17　数据标识案例

数据标识的建立为建模数据之间的关系奠定了基础。

2）数据之间的关系

所谓数据之间的关系，即数据之间的联系，是指某一类数据与另外一类数据的对应关系，用于描述数据之间彼此相互连接的方式。常见的数据关联关系有一对一（1∶1）、一对多（1∶N）和多对多（M∶N）三种。

（1）一对一关联（1∶1）

如果两类数据的记录是一一对应的，则这两类数据之间存在着一对一的关联关系。此时，需求分析人员可以将这两类数据中表示主要内容的数据当成主数据，把另外一类数据作为主数据的扩展。例如，学生数据与紧急联系人数据，一个学生可以设置一个紧急联系人，如图 3-18 所示。学生信息为主数据，而紧急联系人为主数据的扩展。

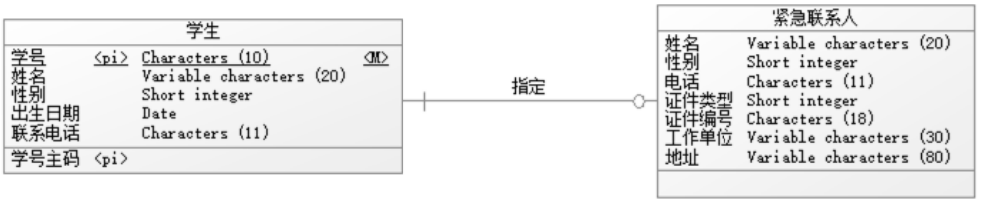


图 3-18　1∶1 案例

（2）一对多关联（1∶N）

如果需求中涉及的一类数据中的记录与另外一类数据中的零条或者多条记录相关联，则可以认为这两类数据之间存在着一对多的关联关系。在这两类数据中，处于 1 的数据内容可以认为是主数据，另一方处于 0 或者多的数据为关联数据。例如，学生数据与书籍数据，一

个学生可以有多本书籍，如图 3-19 所示。此时，学生数据为主数据，书籍数据为关联数据。

图 3-19　1∶*N* 案例

（3）多对多关联（*M*∶*N*）。

如果两类数据中的任意一条记录均可以与另一类数据中的零条或者多条记录相关联，则可以认为这两类数据之间存在着多对多的关联关系。此时，这两类数据均为独立数据，不存在主数据和关联数据之分。例如，学生数据和课程数据，一个学生可以选择多门课程，而一门课程也可以被多个学生选择，如图 3-20 所示。

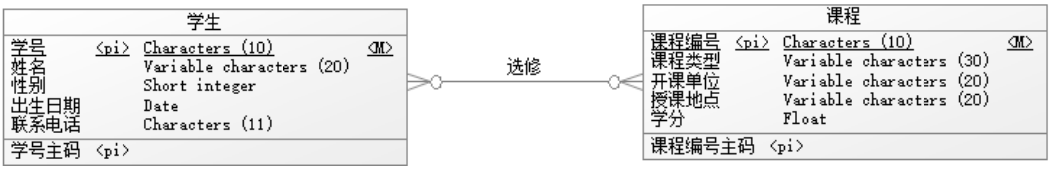


图 3-20　*M*∶*N* 案例

除了建模数据之间的关联重数以外，需求分析人员在表达数据关系时还需要对数据关系的约束情况进行建模，如可选、强制等，如图 3-21 所示。

（a）0 或者 1　　（b）必须有 1 个　　（c）0 个或者多个　　（d）1 个或者多个

图 3-21　数据之间的关系约束

其中，图 3-21（a）表示数据 A 中的 1 条记录与数据 B 中的 0 条或者 1 条记录存在着对应关系；图 3-21（b）表示数据 A 中的 1 条记录必须对应着数据 B 中的 1 条记录；图 3-21（c）表示数据 A 中的 1 条记录对应着数据 B 中的 0 条或者多条记录；图 3-21（d）表示数据 A 中的 1 条记录对应着数据 B 中的 1 条或者多条记录。

如果目标软件系统涉及多类数据的处理和存储，需求分析人员首先需要对目标软件系统涉及的数据进行分析，并对数据的标识和数据之间的关系进行建模。通过对数据进行建模，需求分析结果可以帮助软件设计人员理清目标软件系统中需要处理的各项数据，并帮助软件设计人员根据数据之间的关系来选择合适的数据结构。如果目标软件系统中需要处理的数据记录较多时，数据模型中的数据标识和数据之间的关系也是将来软件设计人员开展数据库设计的依据。

3. 数据项的组成

除了对数据组成、数据之间的关系进行建模以外，需求分析人员还必须对数据的各个数据项内容进行描述。在需求分析中，对数据项的内容描述被称为数据字典。

1）数据字典的组成

为了准确地描述数据项，数据字典必须从名字、内容、补充信息和使用方式等多个角度

来定义与数据项相关的内容。名字是指用户和需求分析人员对系统中出现的各种数据、控制项、数据存储和外部实体的无歧义命名；而内容是指数据项或者控制项的内容和组成；补充信息对数据项的数据类型、预置值、约束条件、取值范围等进行描述；使用方式对如何使用数据项以及哪些处理可以使用数据项等内容进行约束。

数据字典作为软件需求分析阶段的重要内容，对需求中涉及的数据项、数据流、处理/加工逻辑、数据存储和外部实体等信息进行精确、严格的定义；同时，数据字典也对各种数据内容的预设值、约束条件、取值范围、出现时间、频率以及最大值等内容进行描述。

2）数据字典的定义和编写格式

在现实世界中，数据项是指对客观世界中实体的属性描述，而描述信息往往可以由简单的原子数据元素组成。因此，需求分析人员在定义数据字典中的内容项时可以采用自顶向下、逐步分解的方式来表达数据项的内容组成，做到对数据内容的精确、无歧义定义。

通常，数据项的内容组成方式有顺序、选择、重复和可选四种方式。

（1）顺序：目标数据项由两个或多个数据元素按照先后顺序排列组成。

（2）选择：目标数据项由两个或多个可能的数据元素中的一个组成。

（3）重复：目标数据项由零个或者多个指定的数据元素重复组成。

（4）可选：可选作为重复的特殊形式，表示选择的数据元素对于目标数据项而言可有可无，即重复零次或一次。

为了更加清晰、简洁地表达数据项内容，需求分析人员可以采用表 3-1 所示的符号来定义各个数据项的内容。

表 3-1　数据定义符号

符　　号	含　　义	案　　例
=	被定义为/等价于	
+	顺序	日期 = 年＋月＋日
[…,…]或[…|…]	选择	性别 = [男,女]，年级 = [一|二|三|四]
{…}或 m{…}n	重复	文件 = {记录}，密码 = 3{[字母,数字]}8
(…)	可选	区号 = (0755)
“…”	基本数据元素	字母 = "A"
…	连接符	数字 = "1"…"9"，字母="A"…"Z"

以火车票订票系统为例，火车票的车次信息数据项可以定义如下：

车次信息文件＝{车次类型＋车次号＋起点＋终点＋日期＋出发时间＋到站时间}

车次类型＝[D | G | Z | T | K]

车次号 = 2{十进制数字}4

十进制数字＝"0"…"9"

起点＝终点＝1{汉字}10

日期＝年＋月＋日

年＝4{十进制数字}4

月＝"01"…"12"

日＝"01"…"31"

出发时间＝到站时间＝时＋分

时＝"00"…"23"

分＝"00"…"59"

可以看到，车次信息数据项中存在以下组合项、重复项、选择项和原始数据项。

组合项：车次信息=车次类型＋车次号＋起点＋终点＋日期＋出发时间＋到站时间

日期＝年＋月＋日

出发时间＝到站时间＝时＋分

重复项：车次号 ＝2{十进制数字}4

起点＝终点＝1{汉字}10

年＝4{十进制数字}4

选择项：车次类型＝[D | G | Z | T | K]

原始数据项：十进制数字＝"0"…"9"

时＝"00"…"23"

分＝"00"…"59"

月＝"01"…"12"

日＝"01"…"31"

除了对数据内容进行严格定义以外，需求分析人员也可以在数据项的描述中对数据的约束条件、补充信息和使用方式进行说明。

通过对目标软件系统中出现的各种数据进行准确定义，数据字典使用户和软件开发团队对于软件系统的输入/输出、数据存储和中间计算结果的数据内容有了共同的理解，有助于改进软件开发团队与用户之间的通信。

3.3 小　　结

由于结构化方法主要是通过跟踪数据的处理过程来完成程序设计，需求分析人员在结构化分析过程中必须对用户需求中的“动词短语”进行重点关注，围绕动词短语开展软件的需求分析工作。

为了准确地描述用户需求，需求分析人员可以借助数据流图来建模数据的处理过程，并使用处理/加工逻辑说明对数据流图中各个处理的内部动作流程进行详细描述。与此同时，由于系统状态往往会影响软件系统处理数据的流程，需求分析人员在分析过程中可以借助状态转换图对目标软件系统的工作状态变迁进行建模。

数据模型对目标软件系统中出现的数据组成、数据之间的关系进行建模。

作为需求分析中的核心内容，数据字典主要对数据流图、处理/加工逻辑说明、状态图以及数据模型中出现的各种数据内容进行准确定义。

在需求分析过程中，需求分析人员必须综合利用多种工具来准确描述用户需求，为将来的软件设计工作奠定基础。

3.4 习　　题

1. 结构化程序的主要组成元素是什么？如何根据结构化程序设计的特点来优化需求分析？
2. 数据流图在结构化需求分析中的作用是什么？
3. 在结构化需求分析中，处理/加工逻辑说明的意义是什么？
4. 状态转换图主要用于建模什么信息？在结构化程序设计中起到什么作用？
5. 数据模型包括哪些内容？这些内容分别有什么作用？