

第三章 软件需求分析

软件需求分析是软件定义阶段的最后一个步骤，它的基本任务是要准确地回答“系统必须做什么？”这个问题，即对目标系统提出完整、准确、清晰、具体的要求。需求分析的结果是系统开发的基础，直接影响软件产品及工程的质量。

软件需求分析是一个不断进行揭示和判断的过程。在此过程中我们将对在软件可行性研究阶段确定的软件范围加以提炼使之具体化，并分析各软件部件可能采用的解决办法。在软件需求分析阶段，软件的开发者和软件需求者起着同样的重要作用。软件需求者设法把有关软件功能和性能的一些模糊的概念加以重述，使之成为具体的细节，而软件开发者则起着询问、顾问和问题解决者的作用。在需求分析中需要大量地交换意见，这其间充满着传错信息和发生误解的可能性，而我们的任务就是面对各种矛盾，协调各种人与人、人与物，物与物之间的关系。

3.1 需求分析的任务

1. 确定系统的综合要求

系统的综合要求包括下面几个方面。

- (1) 确定系统的功能要求。提出系统必须完成的全部所有功能。
- (2) 确定系统的性能要求。包括系统的响应时间、系统需要的存储容量、后援存储器容量、系统重新启动、系统的安全性和可靠性等方面性能要求。
- (3) 确定系统的运行要求。主要是指系统运行时所处的环境要求，包括支持系统运行的软件环境，工具软件和系统软件；支持系统运行的硬件环境，外存储器、通信接口、输入和输出等外部设备。

(4) 系统的扩充要求。不属于当前系统的开发范围，是将来有可能提出的要求，目的是使在

现有的设计中为将来的扩充做准备。

2. 分析系统的数据要求

任何一个软件系统其本质上都是一个信息处理系统，系统必须处理的信息和系统应该产生的信息在很大程度上决定了系统的面貌，同时也对软件设计有着深远的影响。因此，分析系统的数据要求，是软件需求分析的任务之一。

系统的数据来源和去处一般含如下几个方面。

- (1) 从系统以外来，再到系统以外去；
- (2) 从系统以外来，再到系统内部去；
- (3) 从系统内部来，再到系统内部去；
- (4) 从系统内部来，再到系统外部去。

复杂的数据是由许多基本数据元素组成的，数据元素之间的逻辑关系形成了数据结构。我们一般用图形工具辅助描绘数据结构，常用的有层次方框图和 Warnier 图，将在本章第三节中介绍这两种工具。

3. 建立系统的逻辑模型

以上述综合要求和数据要求的结果为基础，我们可以导出系统的逻辑模型，并通过数据流图、数据字典和主要处理算法来描述这个逻辑模型。具体过程如图 3-1 所示。



图 3-1 系统逻辑模型的导出过程

4. 修正系统开发计划

由分析过程而获得对系统的深层了解之后，我们可以准确地估计系统的成本及进度，修正以我们所制定的开发计划。

5. 开发模型系统

开发模型系统是指在需求分析阶段建造软件样机。它的目的主要是检验关键设计方案的正确性及系统是否能真正满足用户的需求。

在软件开发中采用样机策略的主要困难是成本问题。对于一次设计后大量生产的产品，设计样机的费用可分摊到每件产品上，因此每件产品的成本增加很少。而某些应用软件，通常一次只开发一件产品，采用样机策略则成本增加很多。近年来主张采用样机策略的人逐渐多起来，样机法已逐渐成为开发软件的一种重要方法，有的书也称此为原型法。

3.2 需求分析的方法

在软件工程的需求分析阶段，通常采用结构化分析技术、面向对象分析技术、原型（样机）开发技术等。下面对这三种分析技术加以介绍。

3.2.1 结构化分析技术

结构化分析技术是七十年代中期由 E. Yourdon 等人倡导的一种面向数据流的分析方法。按照 T. DeMarco 的定义，“结构化分析就是用数据流图、数据字典、结构化英语、判定表和判定树等工具，来建立一种新的、称为结构化说明书的目标文档。”其中结构化说明书就是需求规格说明书。

结构化分析技术将软件系统抽象为一系列的逻辑加工单元，各单元之间以数据流发生联系。关于数据流图的细化、定义、加工、小说明的描述前面已经介绍过，在此不再赘述。下面我们通过房产管理系统这个实例，具体看一看结构化分析的过程。

1. 项目说明

房产计算机管理系统包括住房分析、调整和计租等。用户可以查询住房情况和房租金额，房产部门也可以对房产进行统计，输出需要的统计表。

在房产计算机管理系统中我们把住户的要求分为三类，即分房要求，调房要求，退房要求。把查询要求分为：查询住房情况，查询房租和查询全局住房情况三种。对以上要求又具体规定如下：

分房要求：可根据分房单进行住房分配，分配住房要从房产文件中读出相应的空房信息。如房号、面积、单位面积房租等。然后把相应的住房信息，如户主姓名、部门、住房分数、家庭人口等再写回房产文件中去，同时还要写入到住房文件中去。最后输出分配后的住房单。与此项要求有联系和影响的工作是房租计算，计算好新分配房的房租后写入到房租文件中去。

调房要求和退房要求与分房要求大体类似，这里不再叙述。

查询住房情况要求：根据住户名从住房文件中读出该住户的住房情况并打印。

查询房租要求：从房租文件读出该住户的信息并输出。

查询全局住房情况要求：根据统计要求作统计处理后输出报表。

2. 分层细化数据流图，见图 3-2, 3-3, 3-4, 3-5 所示。



图 3-2 第一层数据流图

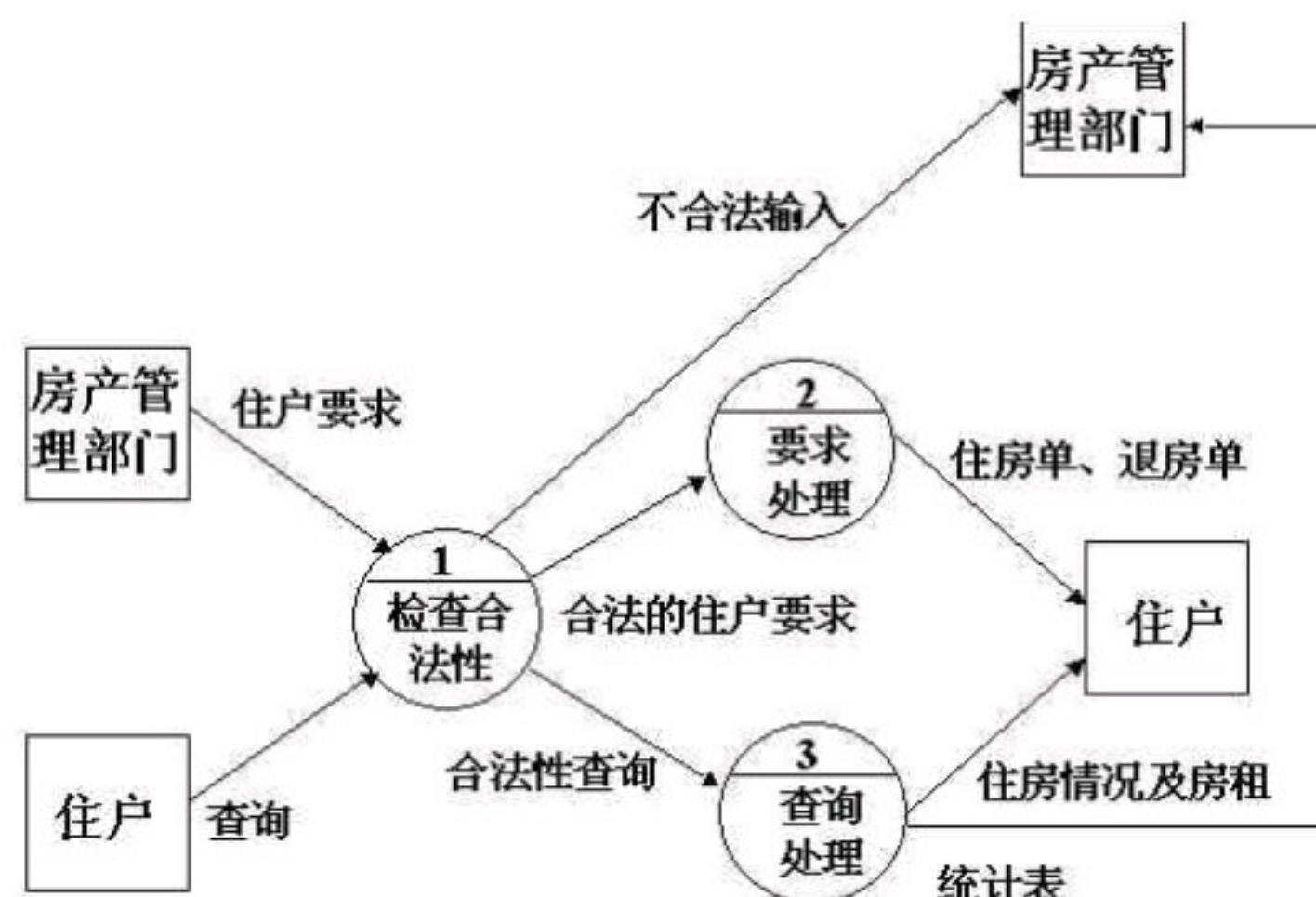


图 3-3 第二层数据流图

3. 数据字典

(1) 数据流

住户要求=户主+〔分房要求|调房要求|退房要求〕

查询=户主+〔住房情况查询|房租查询|统计要求〕

统计表={住房面积+已分住房数|空房数}

住房情况=部门+职称+户主+家庭人口+住房面积+房租

分房要求=部门+职称+家庭人口+住房分数+要求住房面积

调房要求=部门+职称+家庭人口+住房分数+原居住面积+要求调房面积

退房要求=部门+房号

分房单=部门+房主+职称+住房分数+要求住房面积

调房单=部门+户主+职称+住房分数+原住房面积+原房号+要求调房面积

退房单=户主+房号+部门

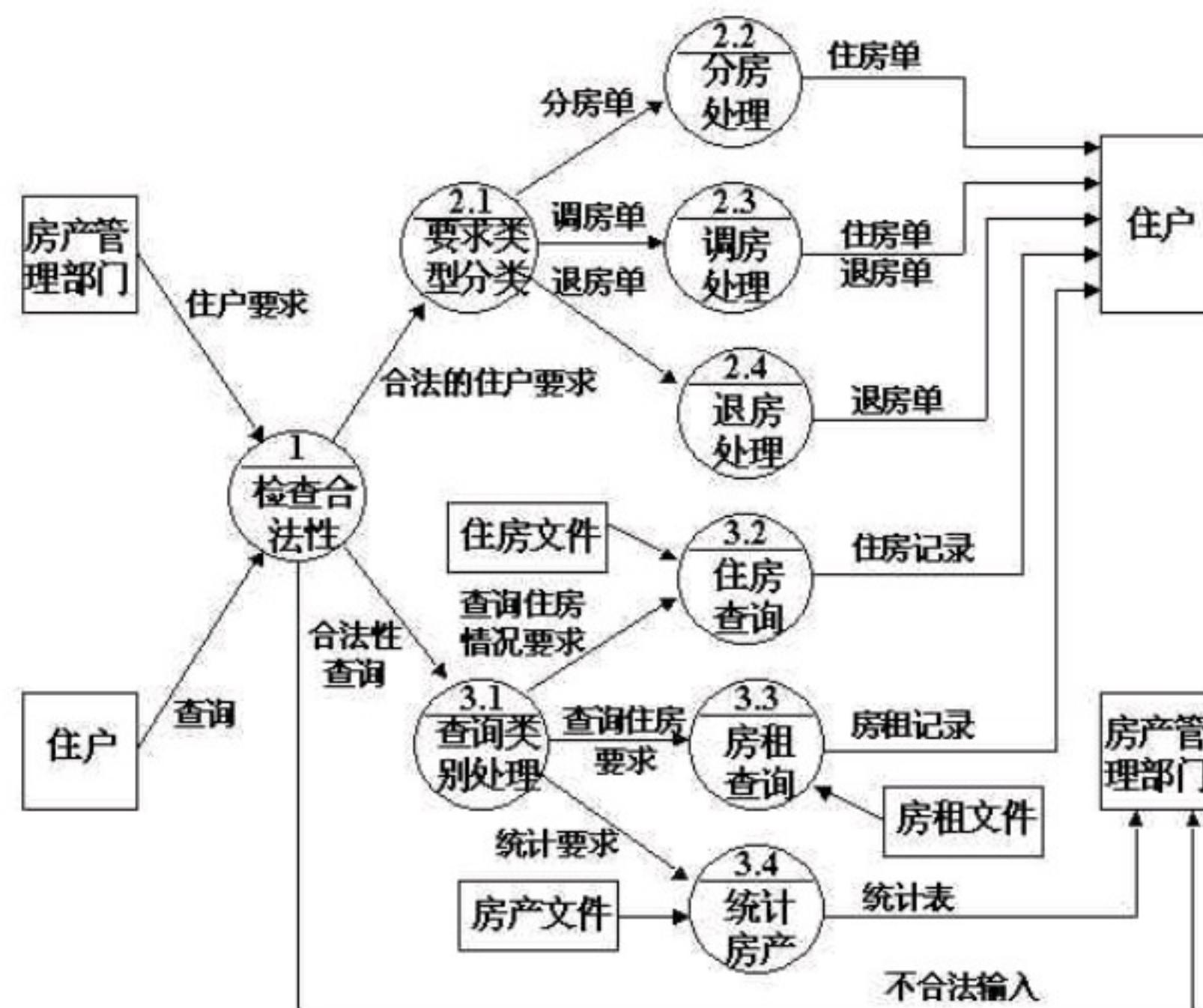


图 3-4 第三层数据流图

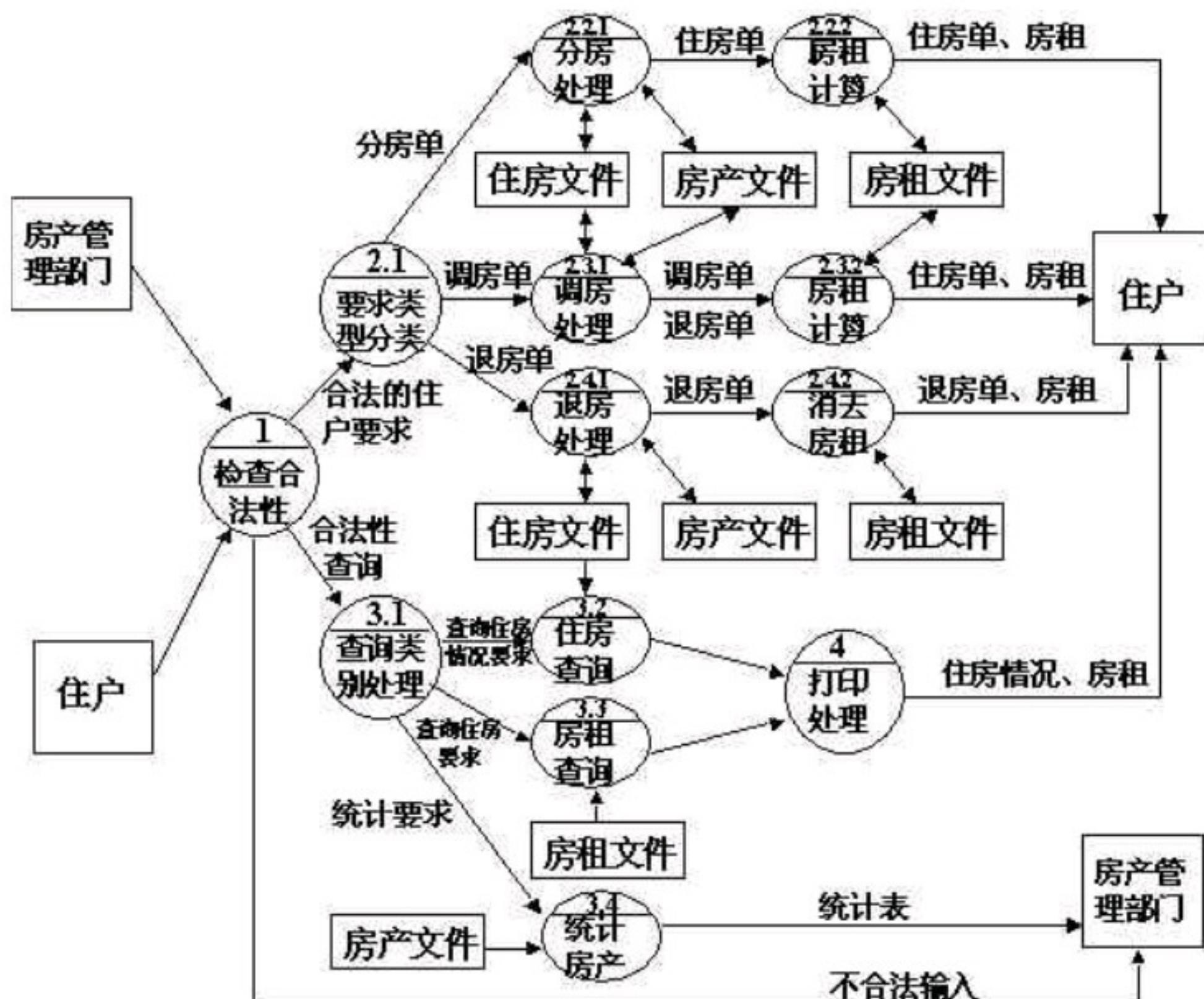


图 3-5 第四层数据流图

住房单=户主+房号+部门+住房面积+租金

房号=楼号+房号

房租=住房面积×单位租金

(2) 文件

房产文件={房号+住房面积+分配标志+单位租金}按房号为关键字排序

住房文件={部门+户主+职称+家庭人口+房号+住房面积}以户主名为关键字排序

房租文件={房号+户主+住房面积+租金+缴纳情况}以户主名为关键字排序

(3) 加工说明

加工编号：1

加工名：检查合法性

加工逻辑：检查住户要求和住房查询的合法性，对不合法的要求或查询给予拒绝。

有关信息：有输入时执行此加工。

加工编号：2.1

加工名：要求类型分类

加工逻辑：根据住户要求，选择分房、调房、退房处理。

有关信息：住户要求合法时执行此加工，处理结果输出分房单或调房单或退房单。

加工编号：2.2.1

加工名：分配住房

加工逻辑：从房产文件中读出合理记录，把分房单有关信息拼成住房文件记录写入到住房文件中去，在房产文件的相应记录中填入已分标志到分配标志字段中。

有关信息：收到分房单时执行此加工，输出住房单。

加工编号：2.2.2

加工名：房租计算

加工逻辑：依住房计算房租写入房租文件。

有关信息：收到住房单时执行此加工。

加工编号：2.3.1

加工名：调房处理

加工逻辑：对住房、房产文件进行读、写操作，修改有关字段内容和有关记录内容。

有关信息：收到调房单时执行此加工，输出住房单和退房单。

加工编号：2.3.2

加工名：房租核计

加工逻辑：依据住房单和退房单进行房租的核算写入房租文件。

有关信息：收到住房单和退房单时执行此加工。

加工编号：2.4.1

加工名：退房处理

加工逻辑：从住房文件读出有关记录，输出退房单，删除该记录，对房产文件中的相应记录修改。

有关信息：收到退房单执行此加工。

加工编号：2.4.2

加工名：消去房租

加工逻辑：由退房单对房租文件进行修改删除。

有关信息：收到退房单时执行此加工。

加工编号：3.1

加工名：查询类别处理

加工逻辑：根据查询要求，选择住房查询或房租查询或统计房产要求。

有关信息：当有查询要求时执行此加工，处理结果输出查询住房情况要求或查询房租要

求或统计要求。

加工编号：3.2

加工名：住房查询

加工逻辑：由查询要求从住房文件中读出相应记录。

有关信息：有查询住房情况要求时执行此加工，输出住房记录。

加工编号：3.3

加工名：房租查询

加工逻辑：由房租查询要求，从房租文件中读出相应记录信息。

有关信息：有查询房租要求时，执行此加工。

加工编号：3.4

加工名：统计房产

加工逻辑：读房产文件，统计房屋分配情况，输出统计表。

有关信息：有统计要求时执行此加工。

加工编号：4

加工名：打印处理

加工逻辑：把住房记录或房租记录打印出来。

有关信息：收到住房记录和房租记录时执行此加工。

4. 复审

分析工作的最后一步是按照结束标准对分析阶段的工作成果进行正式的技术审查，以数据流图作为基本文档，在数据字典、算法描述及其他有关文档的辅助下，仔细分析研究需求分析阶段的结果，目的是发现错误和遗漏。

审查小组通常由四人组成，组长由一名没有参加这个项目的有经验的系统分析员担任，组员由本系统的分析员和两名用户代表构成。若审查合格，那么审查小组成员应该在正式的审查表上签字，若有问题应提出并限期修改，改正后再进行审查，直到合格为止。但要注意，在进入下阶段工作之前，要进行管理复审，只有在使用部门的负责人审查修正后的成本和进度是可接受的，开发工程才能继续进行。

3.2.2 面向对象的分析(0OA)技术

面向对象的概念是在七十年代程序设计方法学的抽象数据类型中产生的。它在软件工程中的应用是即美国XEROX公司于1980年研制出面向对象的程序设计语言Smalltalk-80之后。面向对象的分析技术以模块封装和内部信息隐蔽为主要特征。面向对象语言具有易编程、易修改、易维护，能大幅度提高软件生产率和质量等特点，二者的结合是软件产业中的一次革命。

面向对象的分析，是抽取和整理用户需求并建立问题精确模型的过程。通常，面向对象分析过程从分析陈述用户需求的文件开始，发现和改正原始陈述中的二义性和不一致性，补充遗漏的内容，使需求变得完整准确。接下来分析员要深入理解用户需求，抽象出目标系统的本质属性，并用模型准确地表示出来。

面向对象建模得到的模型包括对象的三个要素，即静态结构(对象模型)——表示静态的、结构化的系统的“数据”性质，它是对模拟客观世界实体的对象以及对象彼此间的关系的映射；交互次序(动态模型)——表示瞬时的、行为化的系统的“控制”性质，它规定了对象模型中的对象的合法变化序列；数据变换(功能模型)——表示变化的系统的“功能”性质，它指

明了系统应该“做什么”，更直接地反映了用户对目标系统的需求。

复杂问题的对象模型由五个层次组成，即主题层、对象层、结构层、属性层和服务层。这五个层次一层比一层显现出对象模型的更多细节，而且这五个层次对应着在面向对象分析

过程中建立对象模型的五项主要活动，即标识对象(类)，标识结构，标识主题，定义属性，定义服务。下面我们以实时空运系统为例，介绍面向对象分析技术的步骤——五个主要活动的内容。

1. 标识对象(类)

对象是所有数据及可对这些数据施加的操作结合在一起所构成的独立单位的总称。

类是对一组具有相同数据结构和相同操作的对象的描述。

为了标识对象，我们应从问题空间的结构、与其他系统的相互作用、设备、记住的事件、发挥的作用、地点和组织单位等方面进行寻找。一旦分析员发现了一个候选对象，应该考虑需要的记忆，需要的服务，多于一个属性，公共属性，公共服务，基本要求等。对确定的对象要设立一个对象名，一般用名词或形容词+名词来表示。

实时空运系统包括 7 个对象。如图 3-6 所示。其中对象 Aircraft(飞机)和 Radar(雷达)为其他系统，对象 Mission(飞行任务)、Flight(航班)、Cargo Item(货物)和 Aircraft Failure(飞机故障)为需记忆的事件，对象 Passenger(乘客)为扮演的角色。

2. 标识结构

面向对象分析方法中有两种结构类型——分类结构和组装结构。其中分类结构表示一般和特殊的关系，这一关系描述了一个对象类是另一对象类的子类/超类。组装结构表示组装、容纳、包含关系，它描述了一个对象是由另外一个或多个称之为成分的对象组成的。

在实时空运系统的 7 个对象中，Shipment Item 为分类结构，如图 3—6 中的半圆所示，它是由两个子类 Passenger 和 Cargo Item 组成的一个超类。对象 Mission 和 Flight 为组装结构，对象 Flight 和 Shipment Item 为组装结构。图 3—6 中三角形指出了上述组装结构，两端标记指出了各对象之间的实例约束。一项 Mission 可以没有任何 Flight 的情况下存在，而一个 Flight 至多是一个 Mission 的组成部分。Flight 和 Shipment 可以独立存在或以任何数量的整体与部分关系存在。

3. 标识主题

主题的个数一般是 7 个左右，统计数据表明，通常人们认为在一个时间内短期记忆限制在 7±2 个事件内，称为“7±2”规则。

定义主题的方法是：对每个结构增加一个相应的主题，对每个对象增加一个相应的主题。当由此产生的主题个数超过 7 个左右时，需要进一步提炼主题(依据主题的耦合情况)，以得到一个更好的模型概观。最后列出主题及主题层上各主题之间的消息连接。

4. 定义属性

属性是描述对象或分类结构实例的数据单元，定义属性分以下几步。

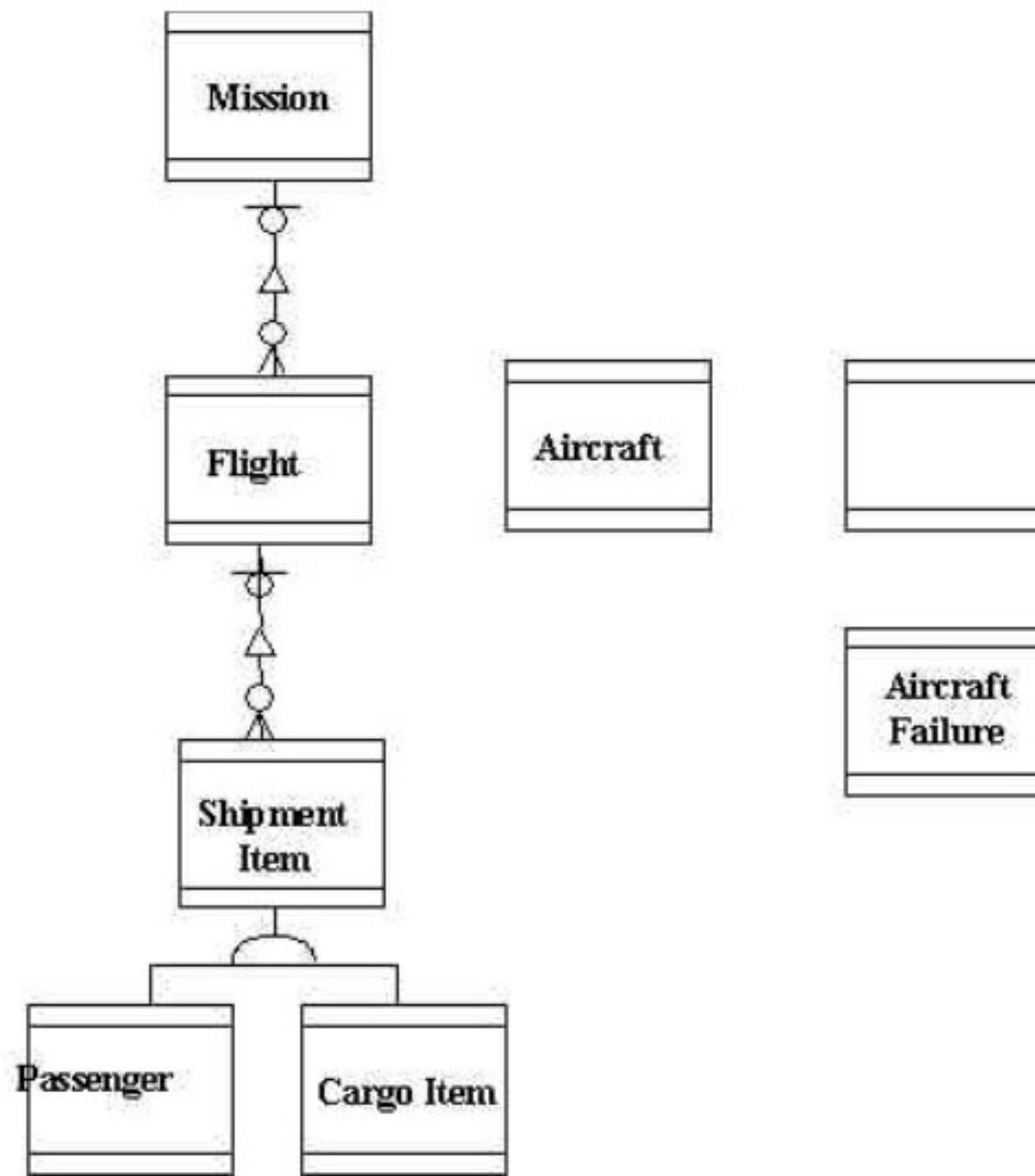


图 3-6 实时空运系统中的对象及结构关系

(1) 标识属性

分析员要与用户交流，在问题空间中找出适用于这个对象或分类结构的所有实例的属性。

(2) 属性定位

利用分类结构中的继承机制确定属性的位置。把通用属性放在结构的高层，特殊属性放在低层。如果某个属性可适用于绝大多数的特殊情况，则可将其放在通用的位置上，然后在不需要它的特殊地方覆盖掉。如果某个属性常常有“不可适用的”值出现，则进一步找出其附加结构。

(3) 标识和定义实例连接

实例连接就是一个实例与另一个实例的映射关系，它反映问题空间的对应性，以获得最少的必要的连接集合。如果连接适合于所有实例，则在分类结构的通用层相连接，否则在特殊的专用层连接。

定义多重性：在每一个方向上建立单重或多大的连接。

定义参与性：在每一个方向上定义连接是任选的还是强制性的，特殊情况作特殊处理。

(4) 修订对象

随着属性的增加，需要重新修订一些对象或分类结构，通常从以下几个角度来检验

1) 带有“非法”值的属性。如果某些属性不适合于一个对象的所有实例或分类结构的某个特定的实例，则应考虑引入附加的分类结构。

2) 单个属性。如果对象或分类结构的实例只有一个属性，则应修改模型以反映更高一层的抽象，将单个属性直接放入与该属性所描述的对象相关连的对象，然后删除这个多余的对象。

3) 属性值的冗余。当一个对象实例的某个属性可能有多个重复值时，应考虑增加新的对象。

4) 适应性参数。对尚无着落的适应性变化参数可作为属性。

5) 说明属性和实例连接的约束。用名字和描述来说明属性，还可以增加一定的属性约束，而且每个属性都可以归类成描述性的、定义性的、通常可导出的、偶尔可导出的。属性作

为一个对象(类)的另一重要部分,它连同对象(类)名字和服务组成一个完整的对象(类)表示。实时空运系统各对象(类)的属性说明如图 3-7 所示。

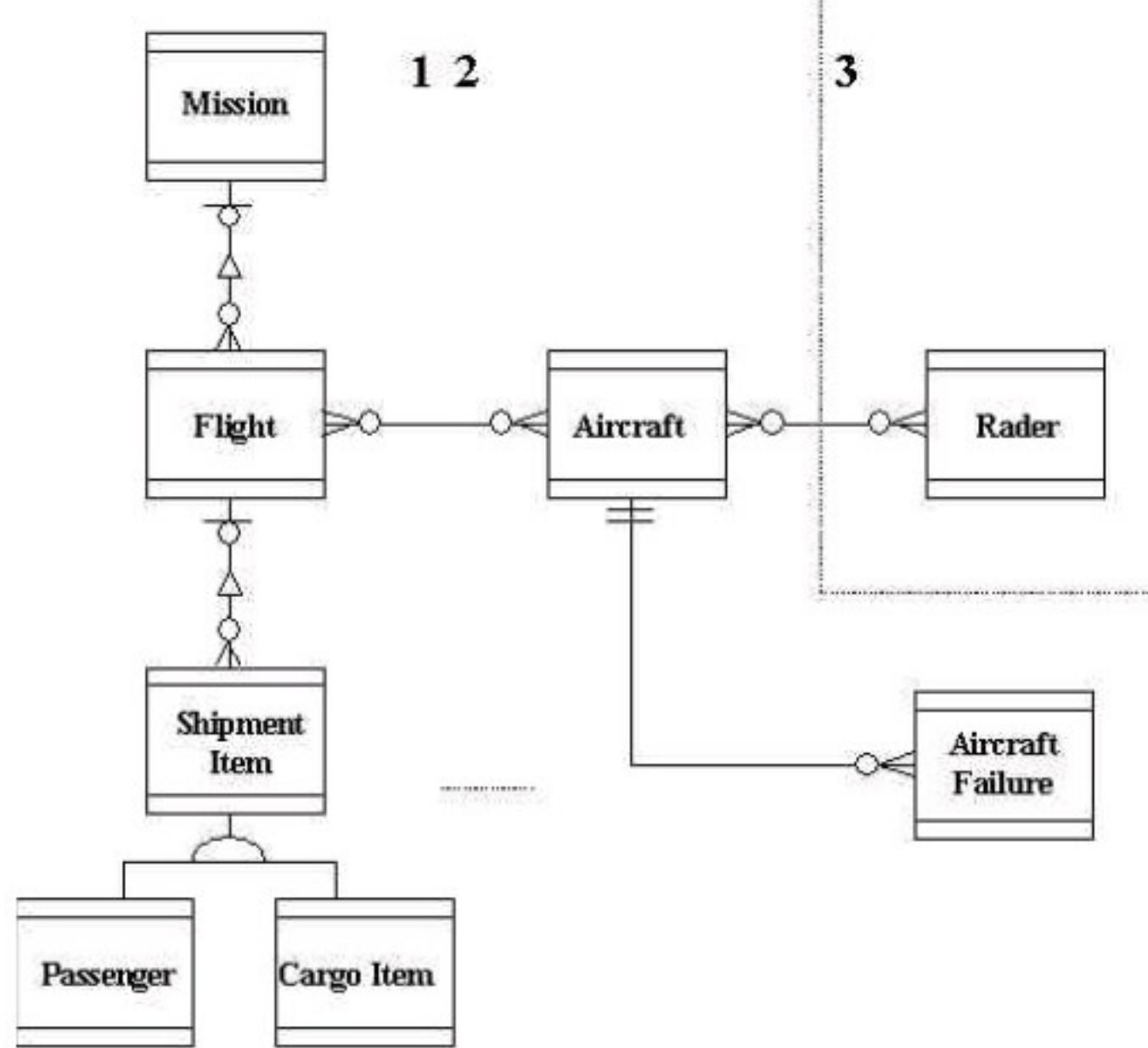


图 3-7 实时空运系统各对象(类)的属性说明

5. 定义服务

一个服务就是收到一条消息之后所执行的处理。定义服务的中心问题是为每个对象和分类结构定义所需要的行为。定义服务的第二个问题是确定对象实例之间必要的通讯。

定义服务分四步：

(1) 标识服务——基本策略

对每一个对象或分类结构考虑三类基本服务：

OCCUR 服务为隐含服务, 完成实例的增加、修改、删除和选择。

CALCULATE 服务为某个实例或代表另一个实例的计算结果。

MONITOR 服务为执行对外界系统、设备或用户的运行监控。

(2) 标识服务——辅助策略

通过对对象或分类结构以后发生的事件, 由基本服务出发, 检查每一步的演变, 增加基本序列, 增加服务, 把这些附加的服务名放入模型图中。

(3) 标识消息连接

消息连接用于适应服务的需要, 表示发出一条消息, 也表示接收到一个响应, 在考虑消息连接时, 首先在已存在实例连接的对象和分类结构之间增加消息连接, 然后再对属性寻找服务。消息连接在图形中用箭头表示, 它从发送者指向接收者, 如图 3-8 所示。

(4) 详细说明服务

为了建立所有服务的细节文档, 对服务的规格说明分为以下几部分:

- 1) 集中于所需要的外部可见的行为;
- 2) 使用一个模板;
- 3) 增加图示以简化服务说明;
- 4) 增加支持的表;
- 5) 建立服务的文字叙述。

实时空系统的服务层如图 3-8 所示。

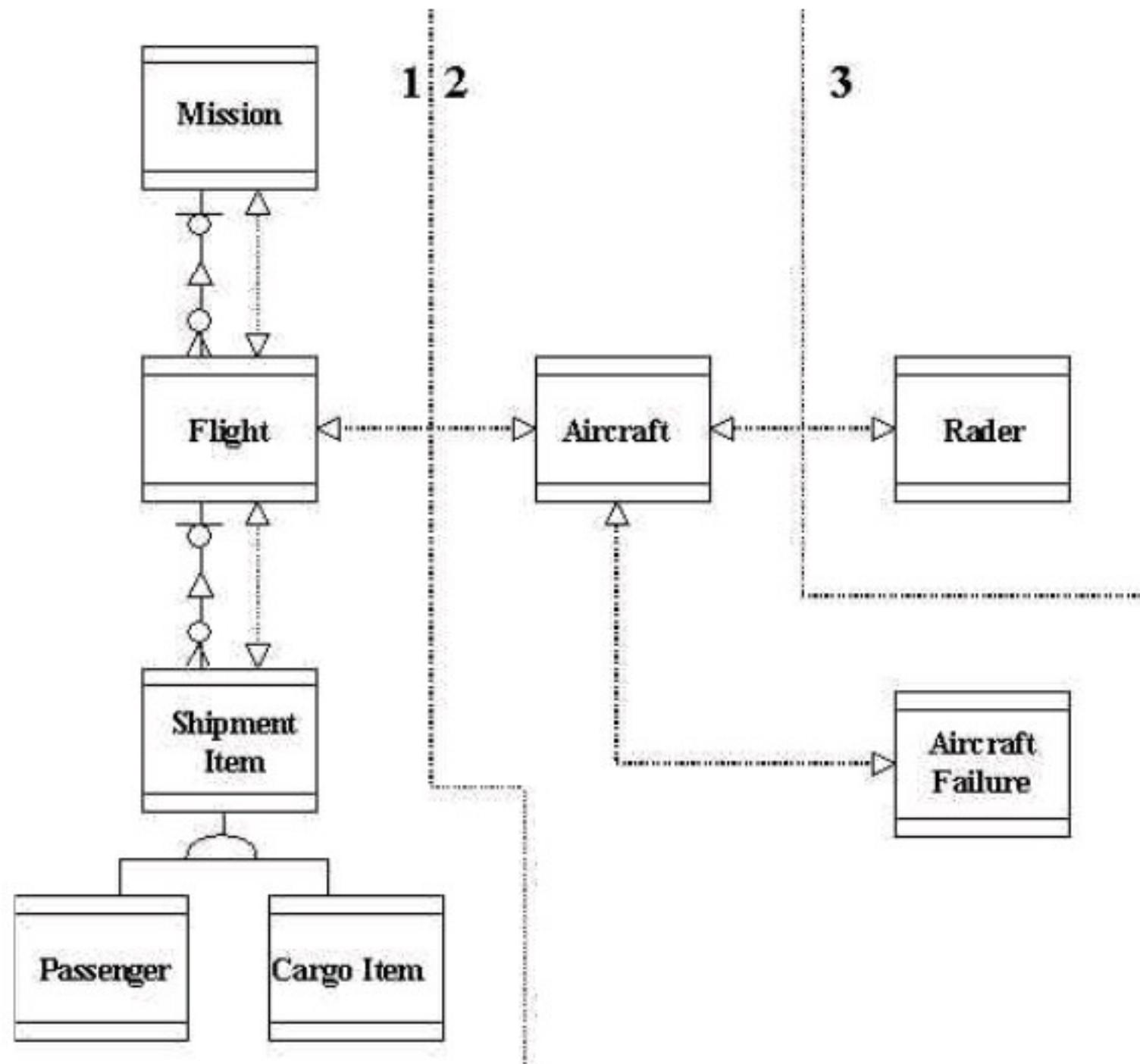


图 3-8 实时空运系统服务层

3.2.3 原型开发技术

传统的生存周期方法学强调自顶向下分段开发，在进入实际的开发时期之前必须预先对需求严格定义。但是，在系统建立起来之前很难仅仅依靠分析就确定出一套完整、一致、有效的应用需求，特别是它不适应用户需求不断变化的情况。原型开发技术打破了传统的自顶向下开发模式，是目前比较流行的实用开发方法之一。

原型开发技术要求在获得一组基本需求说明后，就快速地使其“实现”，通过原型反馈加深对系统的理解，并对需求说明进行补充和精化，原型开发技术一般包括以下五个方面的内容。

1. 实现原型的途径

建立原型的目的不同，实现原型的途径也有所不同，通常有以下三种类型。

(1) 用于验证软件需求的原型

确定了软件需求之后，从中选择某些应着重验证的功能和性能，用适当的工具快速构造出可运行的原型系统，由用户来试用和评价。

这类原型往往用后就丢掉，因此构造它们所用的工具不必与目标系统的生产环境集成在一起，通常使用简洁而易于修改的超高级语言对原型进行编码。

(2) 用于验证设计方案的原型

为确保软件产品质量，在总体设计或详细设计过程中，用原型来验证总体结构或某些关键算法。若验证完设计方案之后就弃掉，则构造原型所用的工具不必与目标系统的生产环境集成在一起。反之，如果想把原型作为最终产品的一部分，则必须把原型的生产环境与目标系统的生产环境集成在一起。原型和目标系统也可以使用同样的程序设计语言编写。

(3) 用于演进出目标系统的原型

不经过实践不能预先定义所有需求，看来比较合理的办法是经过初步分析获得一组基本需求之后，就迅速地用原型加以实现，作为沟通各方的基础和实践场所。随着用户和开发人员对系统理解逐渐加深，不断对原型进行修改和扩充，直到用户感到满意为止。力图用正常的迭代来避免不正常的反复。如果用户希望从满意的原型直接转成实用的目标系统，则必须把

原型的生产环境和目标系统的生产环境集成在一起，当原型与目标系统使用不同的语言时，要改进原型，进行翻译。

2. 功能选择

要恰当选择原型实现的功能。原型它不同于最终的软件系统，两者在功能范围上是有区别的，主要表现在：

- (1) 最终系统是软件需求全部功能的实现，而原型只实现所选择的部分功能；
- (2) 最终系统对每个软件需求都要求详细实现，而原型仅仅是为试验和演示用的，部分功能需求可以忽略，或者模拟实现。

3. 构造原型

在构造一个原型时，应强调着眼于预期的评估，而不是为了正规的长期使用。一般采用超高级语言来实现原型系统，可以大大减少系统原型的开发成本。

4. 评价和确认

通过运行原型，对软件需求规格说明进行评价和确认。在用户参与的评价活动中，要注意来自用户的反馈信息。

5. 进一步使用

根据原型实现的特点和环境，原型既可以作为试验的工具，也可以全部或部分地成为最终系统的组成部分。原型开发与原型运行和评价两者间需要反复进行多次，才能得到经过确认的需求规格说明系统的原型开发，因此原型系统必占总开销成本的一部分。

一个软件是否要开发原型系统，应视软件规模与复杂程度而定。原型开发技术的开发过程如图 3-9 所示。

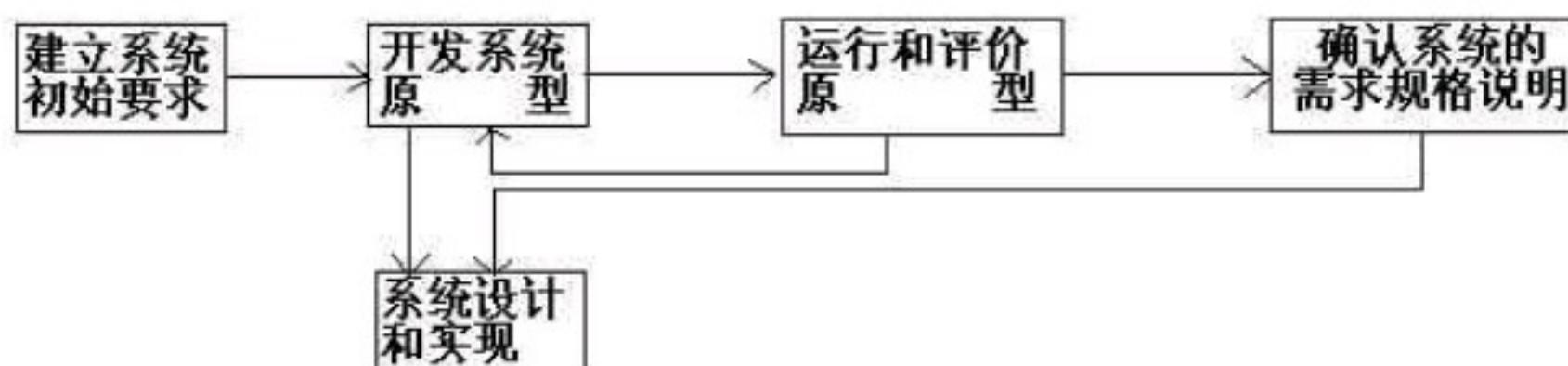


图 3-9 原型开发过程

3.3 需求分析阶段的图形工具

用图形工具来描述数据结构，比文字叙述更直观，本节介绍三种需求分析阶段所使用的图形工具。

3.3.1 层次方框图

层次方框图是用树形结构的一系列多层次的矩形框描绘数据的层次结构。树形结构的顶层是一个单独的矩形框，它代表完整的数据结构。下面各层的矩形框代表这个数据的子集，最低层的各个框代表组成这个数据的实际数据元素(不可再分割)。

例 描绘一家计算机公司全部产品的数据结构可用图 3-10 表示。

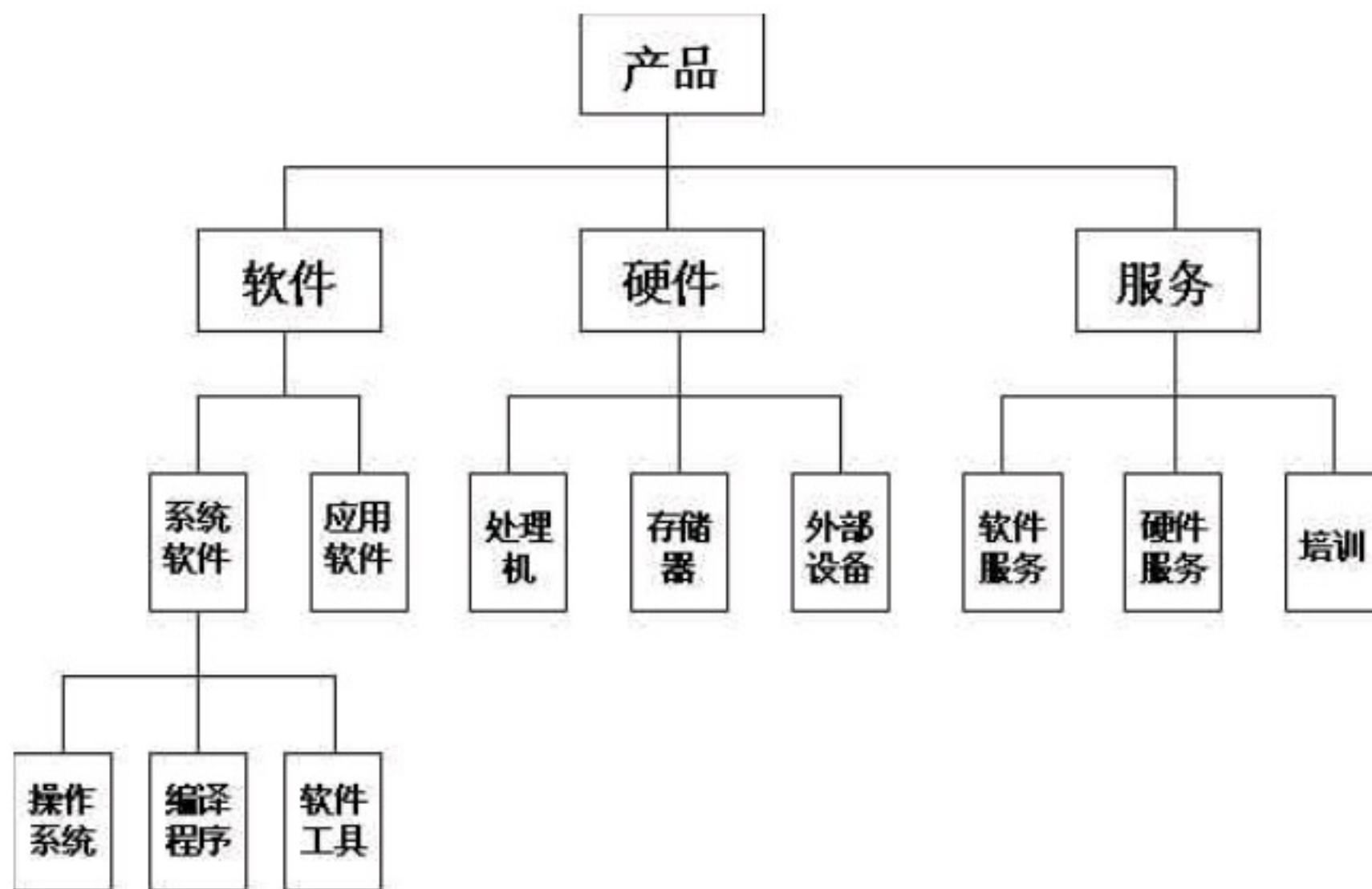


图 3-10 层次方框图实例

随着结构的分解，层次方框图对数据结构的描述也越来越详细。这种方式很适合于结构化分析。

3.3.2 Warnier 图

Warnier 图是由法国计算机科学家 J. D. Warnier 提出的表示信息层次结构的另外一种图形工具。它与层次方框图类似，但比层次方框图更灵活。

用 Warnier 图可以表明信息的逻辑组织，指出一类或一个信息是重复出现的，也可表示特定信息的有条件出现，它很容易变成软件设计的工具。

下面给出用 Warnier 图描述软件产品的一个例子，见图 3-11 所示。

软件产品 系统软件 操作系统(P₁)

编译程序(P₂)

软件工具 编译程序(P₃)

图形生成程序(P₄)

应用软件

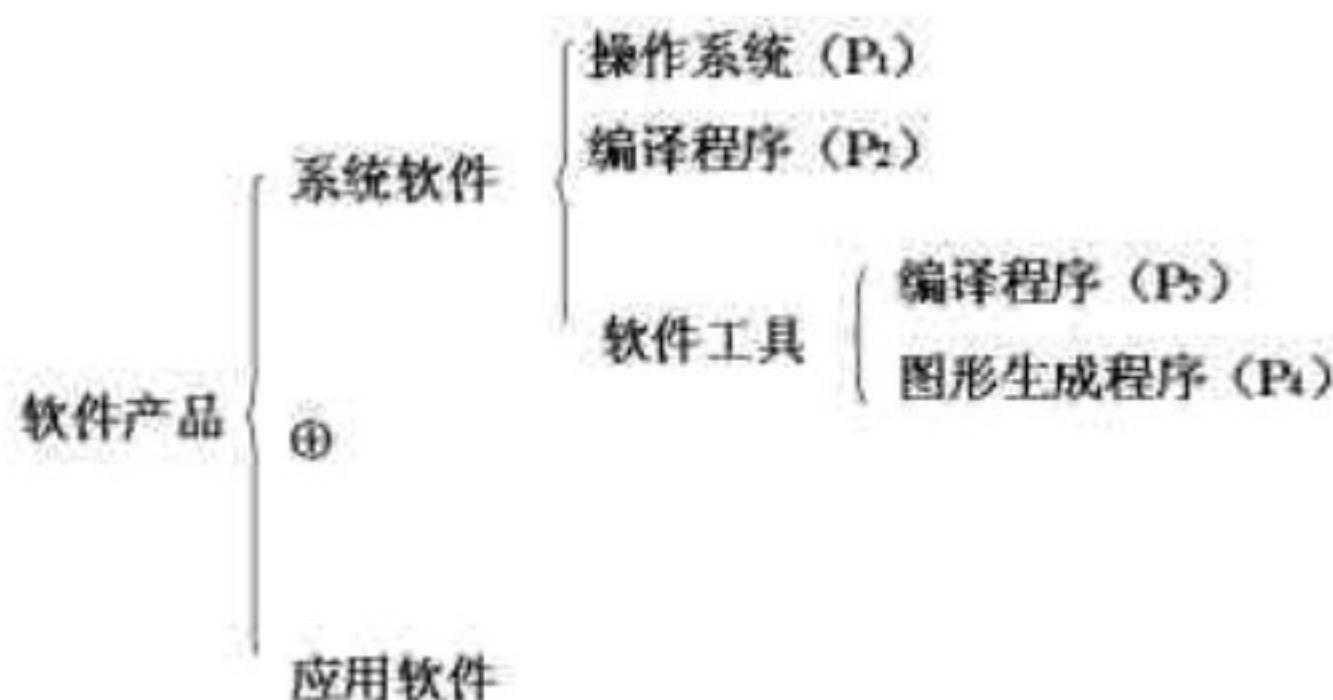


图 3-11 Warnier 图实例

在 Warnier 图中花括号用来区分数据结构的层次，在一个花括号内的所有名字都属于同一类信息；符号 ④ 表示在其上、下方的名字中的一个名字；名字右边圆括号中的符号表示这个名字在信息类中重复出现的次数。

图 3-10 中系统软件和应用软件只能出现一种；系统软件中包括 P₁ 种操作系统，P₂ 种编译程序，软件工具中包括 P₃ 种编译程序，P₄ 种图形生成程序。

3.3.3 IPO 图

IPO 图是输入/处理/输出图的简称，它是由美国 IBM 公司发展完善起来的一种图形工具，可以方便地表示输入数据、数据处理和输出数据三者之间的关系。

IPO 图包括三个矩形框，它的基本形式如图 3-12 所示。左边框列出所有输入数据，中间框列出主要处理及出现的顺序，暗示了执行的顺序，右边框列出输出数据。三个框中间用粗箭头指出数据通信情况。

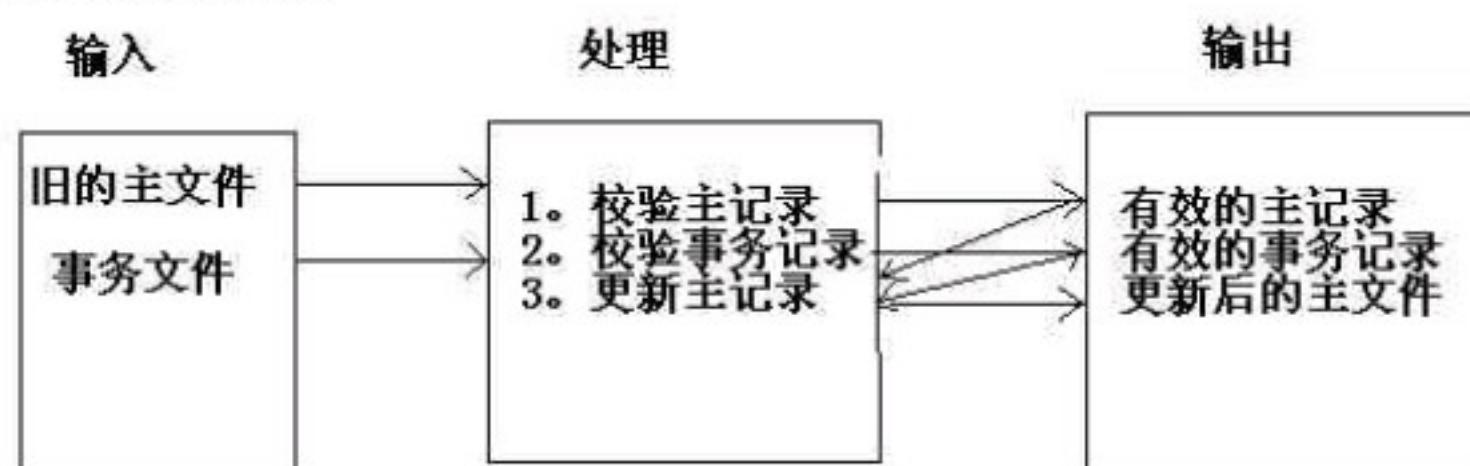


图 3-12 IPO 图

我们在软件设计中比较常用的是另一种改进的 IPO 图，其基本格式如图 3-13 所示。包括了系统名称、作者、日期、模块名、调用与被调用模块清单、注解、以及本模块使用的局部数据元素等。

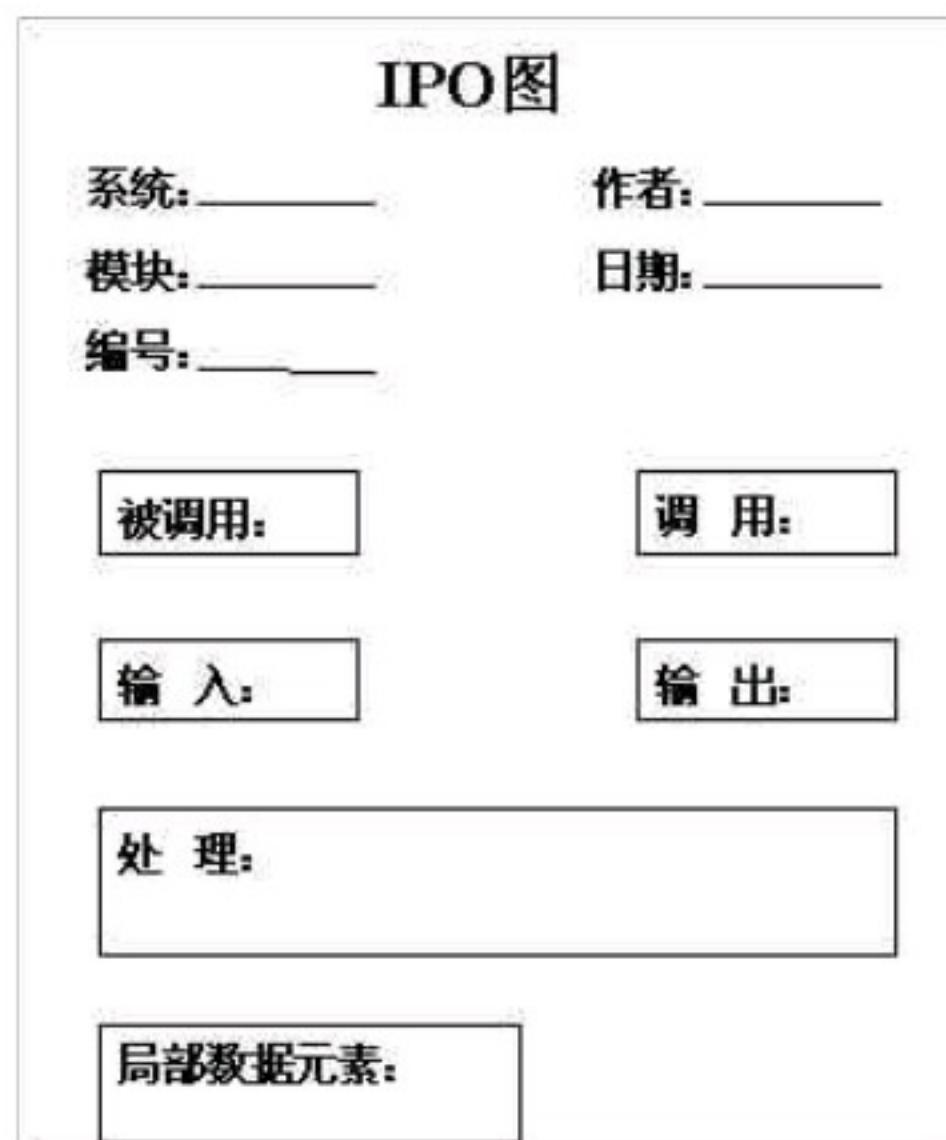


图 3-13 改进的 IPO 图

3.4 验证软件需求

3.4.1 如何验证软件需求的正确性

为了提高软件的质量，确保软件开发成功，降低软件开发成本，我们一般从四个方面进行软件需求的验证。

一致性。在所有需求中，任何一条需求不能和其他需求互相矛盾。

完整性。软件规格说明书必须包括用户需求的每一个功能或性能。

现实性。指定的需求应该是用现有的硬件技术和软件技术基本上可以实现的。

有效性。软件需求确实能解决用户所面对的问题。

3.4.2 用于需求分析的软件工具

用计算机辅助人工需求分析，可使软件开发更趋于工程化和标准化。七十年代末以来，

国外已开发了多种用于需求分析的软件工具，我国在这方面也取得了很大成就，青鸟系统 I 型和 II 型就是这项工作的代表。一般来讲，这些软件工具符合下述标准。

1. 有形式化语法，可用计算机自动处理；
2. 使用这个软件工具能导出详细的文档；
3. 提供分析(测试)规格说明书的不一致性和冗余性的手段，能够产生一组报告指明对完整性分析的结果；
4. 使用软件工具之后，应该能够改进通信状况。

典型的需求分析的工具就是需求描述语言。此语言用来描述用户对系统功能的需求。为了实现计算机对语言的识别和处理，这种语言具有形式化的语法，方便描述系统中各个目标的性质和目标之间存在的联系。1977 年美国密执安大学开发了 PSL/PSA(问题陈述语言/问题陈述分析程序)系统，PSL 语言它的基本结构类似于 RSL 语言，而 PSA 是处理 PSL 描述的分析程序。

PSL/PSA 系统的功能主要有：

1. 描述任何应用领域的信息系统；
2. 创建一个数据库保存对该信息系统的描述符；
3. 对描述符施加增加、删除、修改等操作；
4. 产生格式化的文档和关于规格说明书的各种分析报告。

PSL/PSA 系统用描述符，从系统信息流、系统结构、数据结构、数据导出、系统规模、系统动态、系统性质和项目管理等八个方面来描述信息。

PSL/PSA 系统的主要优点是它改进了文档质量，能保证文档具有完整性、一致性、无二义性，减少管理和维护费用，便于增、删、改操作。

3.4.3 需求分析使用的超高级语言

在需求分析时，必须要开发原型系统，开发原型系统除了可用图形工具表示外，还需要选择超高级语言作为开发工具。超高级语言运行时，需要系统软件的支持，这就增加了存储容量，同时也影响程序的执行速度。因此，超高级语言不适宜于用来开发实际的大型系统。但是对于原型系统来说，通常可以忽略性能方面的要求。相反，因为超高级语言它的功能简洁，可使开发成本大大降低。所以在开发原型系统时，一般都选用超高级语言。本节我们介绍几种常见的超高级语言。

1. APL 是一种典型的超高级语言，提供矩阵运算方面的操作，书写简洁，用 APL 语言开发一个原型系统所需的时间只相当于实现最终软件系统所需时间的一小部分。但 APL 语言运行时开销大，写出的程序结构不理想，不适合于开发实际的大型软件系统。

2. UNIX 操作系统的命令解释语言 Shell，特别是它的第七版既是一个交互式的命令解释程序，又是一个命令级程序设计语言的解释程序。在一个 Shell 程序中可以使用 UNIX 操作系统的全部功能。Shell 语言方便、功能强，开发原型系统时比普通程序设计语言成本低，但它的执行速度较慢。

3. PROLOG 语言，它是以一阶谓词逻辑的 HORN 子集为语法，以消解原理为语义，以深度优先为控制策略的一种交互语言，具有极强的知识表达、推理和查询功能，在表达知识和快速建立软件原型两个方面有明显的优势。PROLOG 程序非常简洁，逻辑性很强。但它计算能力较差，效率也比较低。