JSD 面向对象 信息系统 (13)

63-69

计算机科学1993 Vol. 20№. 4

TP31

JSD与面向对象的信息系统开发

诸葛海 （中国科学院软件研究所 北京100080）

摘 要

This paper disscusses how JSD modelling technique is incorporated to create an Object- Oriented MIS analysis model.

一、引言

面向对象的程序设计（OOP）已在过去十年中引起了软件行业的广泛兴趣，但目前仍局限于程序设计实践和一些基本概念应用，并未形成较成熟的面向对象方法学（OOM），究其主要原因，是缺乏一个面向对象的针对问题领域的系统模拟技术，因而一个面向对象分析（OOA）模式的构造对于OOM来说是非常迫切需要的。当然，一个很自然的想法就是借鉴那些较为成熟的系统开发方法。

JSD（Jackson System Development）是八十年代初以Jackson为代表的学者在JSP（Jackson Structure Programming）的基础上发展起来的。JSD提供了系统开发的框架和一整套的详细技术，是一种较成熟的系统开发方法，并得到广泛应用，特别是在欧洲。

JSD主要有三个技术阶段：模拟、网络和实现，其中模拟阶段主要涉及"所要开发的系统是什么"的问题，网络阶段主要涉及"系统要做什么"的问题，实现阶段是"如何运行已定义的网络"的问题。

一般来说，OOM也应涉及三个阶段：分析、设计和实现，更具体地说，包括六个步骤：定义问题、标识对象、标识操作、建立视图（可视性）、建立界面以及每个对象的实现。

JSD与OOM都有一个所共同关心的实质，那就是反映问题本身结构的问题解结构之软件质量问题。事实上，现实领域问题的各部分和解的软件部件之间有着简单的对应关系，而这种对应恰恰是JSD和OOM所共同关心的方法学基础。

本文论述如何借用JSD的模拟技术来构成OOA的基本内容和方法，主要包括两个内容：一是对象模型的定义，二是功能需求的定义。

二、JSD的模拟技术

JSD模拟阶段的核心思想可概括成"分离"和"对应"，所谓分离就是将功能需求与对象模型分开，旨在提供一个关于问题空间的稳定模型。例如，在图书馆环境中必然包括像图书、读者这样在若干年内相对稳定的对象，而功能需求则不然，它会经常变化。分离对象模型与功能需求就是分离了构成系统的变化因素和稳定因素，也减少功能需求的变化对全局系统规范的影响。显然，对软件系统的维护也带来好处。所谓对应就是现实世界问题的结构与问题解的软件结构之间存在着对应一致的关系，现实世界问题中的对象与问题解结构中的软件构件之间存在对应关系。功能需求与对象模型的分离有助于建立现实世界问题与其解的软件结构之间的稳定对应。

三、定义对象模型

运用面向对象概念（OOC）来开发系统的关键是对象的"标识"。这里所说的"标识"

收到日期：92-10-14

一词有识别和标记两层含义。标识对象的一种方法就是首先从问题陈述中找出名词和动词（分别包括短语）构成词表，由于并非所有描述问题的名词和行为都是目标对象和行为，所以要通过构造启发式方法来选择适当的名词作为目标对象，选择适当的动词作为对象，这是一种可纳的方法，其中最重要的是启发式知识的设计。当然，从问题定义的文字描述中来标识对象还需要开发更有效的方法。

与此相对应，JSD开发系统的模拟过程也具有类似的过程，也采用相似的方法。因而，构造恰当的映射将JSD模拟方法映射为OOC下的分析方法就成为本文的重要目的。

假定OOM=(OOC,OOA,OOD,OOP)，OOD是面向对象的设计，JSD=(JC,JSP,JSD)，我们构造如下从JSD到OOM的映射：

f：JSD→OOM，对任意jc∈JC，都存在oc∈OOC，使得f(jc)=oc，且如果对于jc_1、jc_2∈JC，$jc_1 \neq jc_2$，那么：$f(jc_1) \neq f(jc_2)$。

然后，着重讨论f(JSD)和f(JSD)+OOC的内容。

1. **对象与对象的标识**

JSD所定义的"实体"和OOC中所定义的"对象"相比较，两者的涵义有所不同。JSD中的实体的唯一意义，就是使现实世界中的对象直接映射为计算模型的结构，由此得到问题域的较为准确的视图，而OOC下的对象则具有更为丰富的内容。

(1) **f(JSD)的对象**

f（实体）="f(JSD)的对象"。在f(JSD)中，对象与所需模拟的"行为"的有序集相联系。所谓行为就是在特定时刻发生的事件。一个行为涉及一个或多个对象。每个行为的结束改变了对象的状态。描述对象特点的是属性，它是检测行为需求的根据。由此，f(JSD)所定义的对象可看作是行为和属性的封装。f(JSD)中的对象也具有丰富的内容，它应同时存在于现实世界和信息系统之中，一个对象只有一个名，但一个对象却可能具有多种类型。这里所说的f(JSD)中的对象类型与OOC模式中的类非常相似。

f(类型)="f(JSD)的对象类型"。在f(JSD)中，对象类型的选择限于两种考虑：其一是根据类型实例的行为和这些行为的有序集来确定对象类型（区别于OOC中的类）。例如在银行环境中，根据开户、存款、取款和消账的某种有序集来描述账务，如开户行为应在所有其它三个行为之前。行为的次序在f(JSD)中是非常重要的，它通常用f(JSD)的过程结构图（如图1所示）来描述。其二是确定对象在模型边界内的整个生命期。图书的生命期是从订购到报废这段时间。对象的生命期一般是较长的，如图书的生命期一般都在几十年以上。一旦某对象类型在模型中出现，那么它在该对象的整个生命期内是确定不变的。

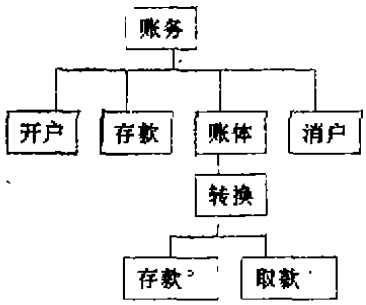

图1 账对象的结构

f(JSD)的对象模型是由非常接近现实世界的对象组成，是现实世界问题的映像，现实世界中对象的任何变化都能相应地在模型中得到反映。由于这些变化发生在现实的时序空间里，所以在相应对象模型的时序空间里模拟这些行为就提供了问题的更加实际的视图。建立逼近现实世界的精确模型有两个优点：一是便于用户根据现实世界的变化来检查和分析模型；二是为适应功能需求的变化提供一个更加稳定的平台。因此，f(JSD)标识对象的方法可以引入OOA中用以标识对象。

(2) **角色**。对象的实例往往可以扮演多个角色，例如，对于同一个人可能要制订两种不同的类型来描述其不同阶段的生活历

的父子关系中得到标识。父对象与多个子对象相连，而一个子对象仅与一个特定的父对象相连。子对象的行为集是其父对象行为集的子集。袋对象的概念提出了一个重要的观点：通过构造对象行为的有序集，来标识隐含在其它对象结构中的对象。

2． **关系**

除了分析对象的结构之外，还必须了解一对象与其它对象之间的关系，就像它们在现实世界中所呈现的那样。模拟这种关系除了表明对象之间的连接关系之外还应表明施加在连接关系上的静态约束。理解这种静态约束将有助于表明对象实例的创建和消亡以及对其它对象存在的影响。

（1）全关系、部分关系和弱关系。全关系（无条件关系）定义为对象类型的每个实例都与另一个对象类型的实例相联系。例如，“国家由一个政府来执政”（如图4所示）是一个一对一的全关系。

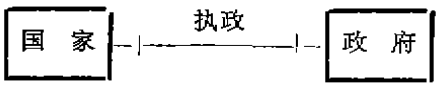

图4 全关系

部分关系（条件关系）与全关系所不同的是不要求对象类型的所有实例都与另一个对象相联系。如图5所示，学生并不一定能填满教室座位。

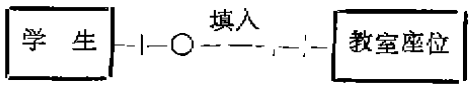

图5 部分关系

弱关系是一个对象类型的实例与另一个对象类型的特殊实例之间的关系，如图6所示。

图6 弱关系

读者与书的一对多关系如图7所示。

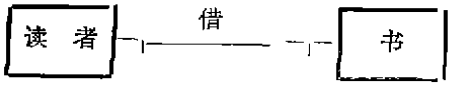

图7 读者与书的一对多关系

（2）共同的行为和关系。当两个对象参与同一个事件时，在这些对象之间就会存在共同的行为。共同的行为表明了两对象之间的一种对应关系，一个对象是行为的发出者，另一个是行为的接受者。对象关系和共同行为关系是互补的，关系描述了对象之间的静态约束，而共同行为则模拟了事件的动态约束关系。共同行为的模拟有助于验证关系的正确性，由此可得到现实世界领域问题的更精确的模型。

3． **分类**

分类是构成面向对象开发模式的重要组成部分，它在“超类-类-子类”关系下，将问题空间描述成子类的层次结构，如图8所示。子类可继承父类的属性和行为。

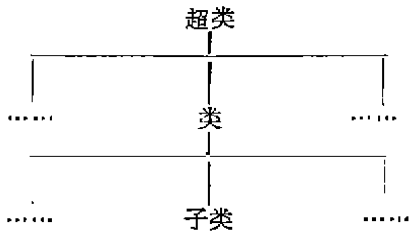

图8 类的层次结构

f(JSD)并不直接支持分类及其继承机制，与之相应的只有类型。事实上，f(JSD)中的对象定义是由类来描述的，而类的出现须确定对象类的规范，换言之，f(JSD)支持的是类型-实例继承而不是子类继承。

在f(JSD)中，所有的子类型都用其上层的类来表示。当f(JSD)中的两个对象在类型或行为的次序上有某些差别时，即使它们具有共同的属性和行为，也可以将它们认为是两个对象类而不是两个子类。例如，在运输公司中，卡车和小汽车是两种实体类型，可以用独立的两个类来模拟，“获取运输许可证”是卡车的行为而不是小汽车的行为，尽管这两种车都有共同的行为和属性，但它们也可归为更高层类的子类，以便两对象可共享共同的属性和行为，但由于缺少分类和继承机制，这在f(JSD)中是不可能的，因而对模拟以这种形式构成的现实世界的对象是一个严重的局限。

(1) **分解对象**。f(JSD) 中的对象都被认为是一个长期运行的稳定过程，每个行为都可认为是过程定义所不可缺少的部分，且行为和属性都局限于过程内部。与此同时，在对象类型中不可能共享行为和属性，因而限制了分类和继承概念的应用。

f(JSD)对象由三个内容所刻画：状态、状态转换和过程。在f(JSD)中行为和状态有明显的区别，行为是一个事件，而状态是事件完成之后对象的情况。Jackson 特别强调不能将状态作为行为。

基于行为的对象结构化显然与对象的状态有关，行为的结束改变了对象的状态，行为是在给定的状态下只允许进行某些行为序列的一种方式。给出了行为和对象状态就定义了对象下一个可能的行为。

对象模型是现实世界领域问题中对象的映像。当对象参与了发生在现实世界的事件时，关于事件的信息也同时在对象模型中得到反映和模拟。在现实世界中发生的每一个行为事件都对应在对象模型中得到模拟运行。

操作作用于对象属性，过程则是操作的有序集合。过程的运行导致了对象计算状态的更新，从而反映了相应的现实世界对象的当前状态。

(2) **引入分类**。对象又可从更加特殊的角度来认识：对象是属性、状态、过程和状态转换的封装（如图9所示）。基于对象的松耦合结构，分类和继承的概念可引入模型之中。将子类的共同属性和行为去掉转而作为高层对象类的一个部分，对于那些在子类中特殊的行为和属性仍保留在子类中。

由以上讨论可得，f(JSD)的对象模拟技术可用来构造现实世界问题的对象模型，它在增加分类和继承的内容之后，可作为面向对象的对象定义模式。

四、定义功能需求

前面讨论了对象模型的结构，对象模型可以看作是现实世界中对象特性的抽象定义，模型中的每个对象都看作是抽象类型，且其属性仅由对象所定义的一组操作来进行操作和更新。每个对象都知道如何操作自己，且只能通过接收对象的标准接口和协议来访问对象的任何数据。因此对象仅包括数据和局限于对象内部的操作，而不包括与其它对象相关的细节。这就与OOM 的信息隐藏和封装的概念相吻合。

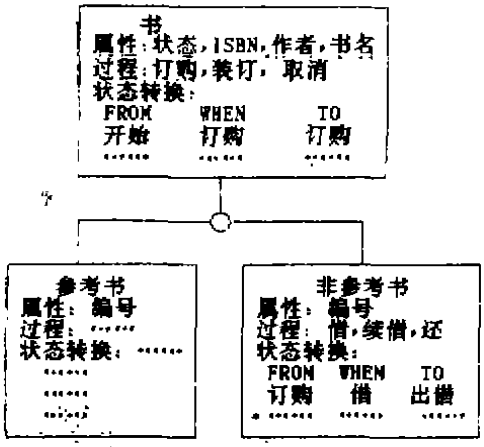

图9 书的分类结构

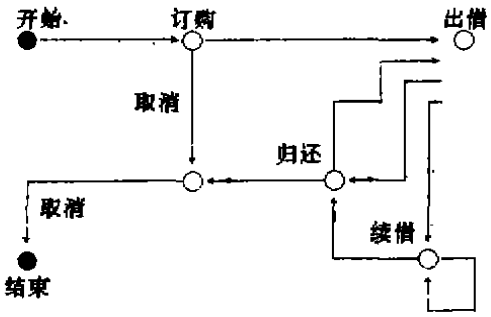

图10 非参考书的状态转换图

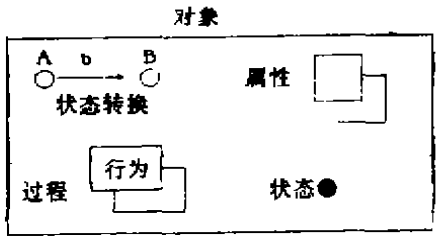

图11 对象结构视图

1. 功能需求

功能需求是对输入、输出和处理的需求，是定义信息系统的另一个主要部分，为了满足用户的功能需求而设置。

我们认为，如果将功能需求作为对象操作来定义，那么必然有以下结果：

(1)违反了作为功能需求的对象的定义，而这种对象又不属于任何一个特定的对象。对象的操作仅仅应用于该对象，也只能用于该对象。所以，功能需求不能定义为对象模型中的操作；

(2) 弱化了功能需求的可视性，使得功能需求的改变非常困难；

(3) 功能需求会经常变化，一个良好的软件开发经验是分离可变部分和稳定部分，这将减少系统一部分的变化对全系统的影响程度。

(4) 将与用户的习惯相矛盾。

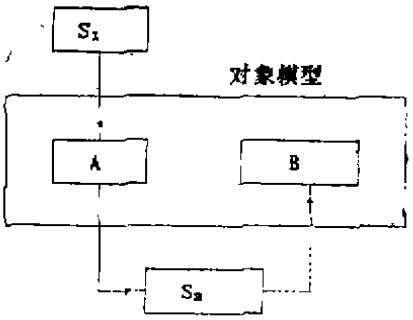

图12 功能需求规范

2. 功能需求规范

基于前面的讨论，有必要分离功能需求和对象模型，这也与f(JSD)的功能需求的定义相吻合。

功能需求规范是由对象模型和与其相连的一组"服务"来表现的。服务是作用在对象模型上操作的过程模型。功能需求可包括多个服务。

服务可用图12描述，图中表示了两个服务S_1和S_2作用在对象模型的两个对象上的功能需求，其规范为"当服务S_1接收到某种信息时，它发送消息给对象A，在完成其操作后，对象A启动服务S_2，而后检查对象类B的实例属性来产生所需要的操作。"，图中的虚线表示检查对象类B的属性，实线表示启动一个对象操作，为了清晰，图中省去了输入和输出。

访问对象的数据是通过被访问对象所提供的标准界面接口和协议来进行的。

系统的功能规范一般是递增的，当要增加所需新的功能时，必须要标识一组相应的服务并将其与对象模型相联。当这种增加过程不断进行下去时，就需要越来越多的服务。例如，当需增加将书发送给读者这种功能时，就需要增加一种新的服务来完成这种功能，即相应地必须要增加与读者和非工具书这样的对象类打交道的服务，其操作可由读者和书两对象用"借"来启动。操作"借"的规范将确定诸如书的发送日期这样相关属性的更新。

服务可以组织在另一个抽象层上而得到标识。这样，相关的服务可被包装在一起，作为一个"伪对象"，所谓伪对象是指在现实世界中并无该对象。它们不参与现实世界问题域的事件，也不受时序的约束。

图13表示了伪对象与对象之间的相互作用，其中虚线表示检测书所处的状态。

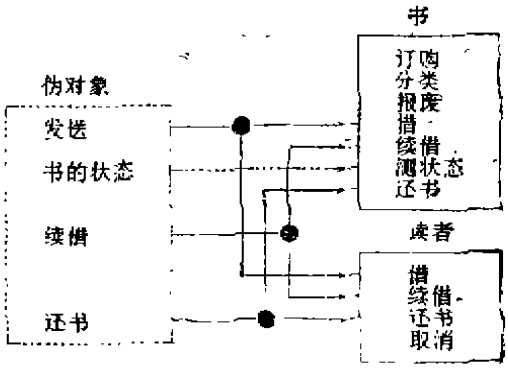

图13 对象之间的作用

五、结论

(1) f(JSD) 中的任何对象都可看作是有序行为和属性的封装，这与OOC中的对象相对应。f(JSD) 中的对象机制提供了在问题空间中标识对象的一个平台，一对象可通过对象发出或接受的行为来标识，对象的属性是通过检查行为的需要来确定的。因而对象可看作是有序行为和属性的封装。

(2) 通过标识现实世界中对象所扮演的不同角色，有可能进一步标识其它的对象类，

软件 规范

计算机科学1993Vol.20№.4

69-72

软件规范

TP31

黄林鹏 孙永强 倪德明 童维勤
（上海交通大学计算机系 上海200030）

摘 要

规范（specification）的目的是提供一个标准，用以引导和评价软件系统的设计、实现和维护。本文从术语"规范"的定义出发，讨论一个良好的软件规范必须具备的性质，分析了形式规范、半形式规范和非形式规范的优缺点，给出了一些典型的规范系统的分类及相互间的关系。

一、引言

Boehm认为，一个软件的开销大概有二分之一甚至三分之二用于维护，因此减少软件开发费用的任何尝试都应注重软件维护的开销。一般降低软件维护需要的主要途径有：a. 降低矫正的需要；b. 降低修改的工作量；c. （通过使用标准构件）减少总的开发规模。一个良好的规范对所有这些途径都有着重要的影响。因此为了改善软件的质量和提高总的生产率，我们不应减少规范工作，而应增加投入，由此在维护上（同时也在设计、实现及集成上）节省更多的开销。

在软件工程领域也许没有其它任何一个词比"规范"有着更多的定义。当"规范"用作一个名词时，它和词"文档"有着相似的含义，如一个需求规范即是一个记录需求的文档；当"规范"用作形容词时，它可修饰软件编码之前的任何工作，如"规范阶段"广泛用于表述关于系统所有模块的外部可视行为的定义和文档活动（产生了一般所称的"模块规范"、"低层设计"或"详细设计"），或者用于表述整个系统的外部可视行为的定义（产生了一般所指的"软件需求规范"）；当然也有人使用"规范"这个词来表示软件的任何（不考虑抽象层次的）形式描述。

尽管"规范"的定义有多种，但这里我们

这使得分析模型得到增强。

（3）嵌入对象的概念使得对象在与其相连的父子关系中得到标识。嵌入对象的概念建议了一个重要的观点：通过构造对象行为的有序集，来使得标识隐含在其它对象结构中的对象得到标识。

（4）通过考虑对象之间的关系，可标识一对象与其它对象之间的组合关系。

（5）在f(JSD)中引入分类的概念以增强其模拟现实领域对象的能力。

（6）不属于任何模型对象定义的功能需求应与对象模型的定义分开，且作为一个服务独立地指定。在高层抽象中，它们可看作是属于伪对象方法的一个伪对象。

（7）最终的系统规范是由模型对象和伪对象的集合组成。

综上所述，f(JSD)加上分类的概念及其继承特性构成对象模型和功能需求模型，两者一起便可作为OOM的第一个步骤——OOA应用于信息系统开发中。

主要参考文献

[1]诸葛海 等著《信息系统与类比方法论》，浙江大学出版社，1992。

[2]诸葛海 编著《信息系统开发方法与环境》，浙江大学出版社，1993。