层次式面向对象设计方法 HOOD 的研究与改进

郑明春 张家重 高 波 (济南山东师范大学计算机系 250014)

摘要:本文详细介绍了层次式面向对象设计方法 HOOD 的基本概念和设计过程,以及其图形和文本描述语言,然后与其他面向对象设计方法进行了比较,指出了 HOOD 方法的优点与不足,进而,提出了一种改进 HOOD 的方案。

关键词:对象 面向对象设计 HOOD

一、引言

层次式面向对象设计方法 HOOD(Hierarchical Object Oriented Design)方法是由 ESA(European Space Agency)提出的,用于软件系统的结构设计方法,即主要支持概要设计阶段,它适合大型复杂软件系统的结构设计,并支持实时系统开发,具有层次结构清晰、易于集成和测试、简单实用等优点。在国外,HOOD 方法已被许多大公司采用,但在国内还未见到有关研究 HOOD 方法的报道,有鉴于此,本文较详细地介绍 HOOD 方法的具体内容,以及我们对该方法的研究情况和已开展的有关工作。

二、HOOD 方法设计背景

HOOD 方法最初是面向 Ada 语言的,第 1 版于 1987 年推出。它借鉴了抽象机 AM(Abstract Machine)[3]和 Booch 的 OOD[4]的概念和经验,AM 与 OOD 概念之间不仅没有矛盾和冲突,而且 AM 中 machine 的概念与 OOD 中 object 的概念非常类似,此外,AM 能够增强对象的层次结构,而这正是一些 OOD 方法所缺乏的,因此 HOOD 引进了层次的概念。

HOOD 推出以后,许多公司接受并使用了这种方法,在 1988 年,Columbus Space Station 项目在分析和设计(AD)阶段都使用了 HOOD 方法;1989 年,Hermes Space-place 项目也选择了 HOOD 方法。HOOD 方法之所以能够获得成功并被广泛使用,主要得益于两个突出优点:

- 层次结构清晰;
- 易于集成和测试。

HOOD 工作小组也于 1989 年修改和更新了 HOOD 参考手册,从而使得 HOOD 逐渐完善起来。

三、基本概念

1. 对象

HOOD 将对象分为两大类,被动对象和主动对象。对被动对象来说,控制从一对象传递到另一对象,操作立即被执行,相当于顺序处理;而对主动对象来说,控制不被传递,它能够接受刺激,并根据其内部状态进行响应。

2. 关系

HOOD 有如下三种关系:

(1) 包含关系。如果对象 A 被分解为若干子对象 A1, A2, ..., An(相应地, A 被称为父对象),它们共同实现了父对象的功能,则对象 A 与其子对象之间的关系就是包含关系。依据这种分解所形成的层次就是父子层次,整个系统被分解为一棵层次树。

(2) 使用关系。若对象 A 使用对象 B 的操作实现其功能,那么就称对象 A 使用了对象 B。对象间的这种使用关系形成了对象间的高低层次。高层的对象可以使用和控制低层的对象,低层的对象为高层的对象提供功能,反之不然。

(3) 实现关系。若父对象提供的操作(功能)直接由某一子对象的操作实现,那么父、子对象操作之间的关系就是实现关系,这种实现关系仅存在于对象的操作之间。

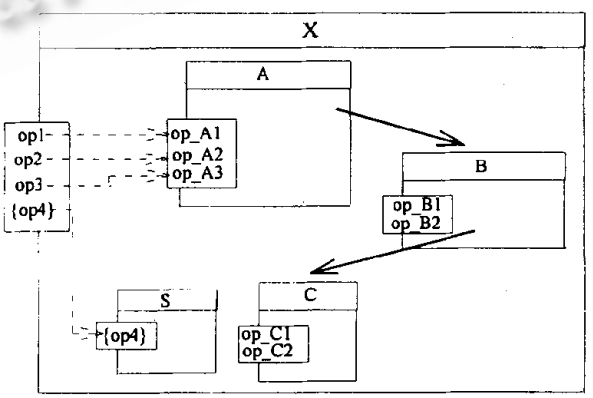

图 1 三种关系图示

图 1 清楚地说明了上述三种关系。对象 X 包含对象 A、B、C 和 S，A 使用 B，B 使用 C；操作 op1、op2、op3 分别由子操作 op-A1、op-A2、op-A3 实现，操作集 op4 由子对象 S 的操作集 op4 实现。

3. 设计过程

HOOD 采用自顶向下分解的方法将一个系统分解为一棵树，如图 2 所示。

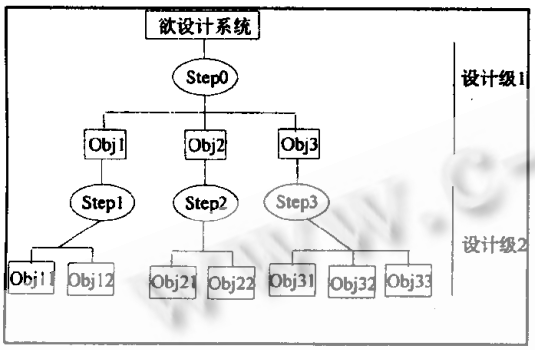

图 2

树的根，被称之为根对象，对应着欲设计的系统，其接口对应着欲设计系统的功能；树的叶子，被称之为终端对象，直接被描述，不能被进一步分解；树枝是通过对象间包含关系而得到的。

HOOD 由一系列的基本设计步骤组成，在每一步中，将一对象分解为若干子对象，设计的结果就形成了设计过程树，每个设计级对应着树的一个深度层次。

每个基本设计步包括如下四个阶段：

(1) 问题定义和分析阶段。主要任务是标识出需由对象处理的问题，而不是对它们进行求解。问题的定义可以参考软件需求文档(SRD)，也可以借助数据流图(DFD)和状态转换图(STD)，将它们作为对文本描述的补充。

(2) 非形式化求解策略的阐述。用自然语言描述非形式化求解策略，以便能够在下一阶段中标识出对象和操作，这些对象或者是现实世界中的对象，或者是与数据实体相联系的软件对象。

(3) 求解策略的结构化描述。此阶段的任务是将对象初步分解为若干子对象，可分为四个子阶段：

①标识对象和操作。一般认为，对象和操作分别来自非形式化求解策略描述的名词和动词，但是仍有一些其它考虑。

②根据非形式化求解策略的阐述，罗列、描述并将相关的对象和操作联系起来，为 HOOD 图的开发作准备。

③根据 HOOD 方法的两种层次和对象间的三种关系，开发 HOOD 图。

④对设计策略进行适当的调整。

(4) 解的详细描述。对象描述框架 ODS 采用文本形式描述对象，是对 HOOD 图的补充细化。ODS 从六个方面详细地描述了对象，能够用于检查完整性、内部一致性，并使得将设计从一种平台转移到另一种平台或工具箱更容易些。

四、对象模型的描述

HOOD 对象模型的描述有图形和文本两种语言，图形描述形式一般用于描述系统的整体结构，亦称为 HOOD 图。HOOD 图具有清晰直观的特点，能够使用户对于对象的分解一目了然；文本形式是对象的详细描述，用于为 HOOD 图的对象增加必要的信息。

1. HOOD 图

用于构造 HOOD 图的基本构件(图元)有很多，以下介绍最常用的 8 种构件，并藉此说明 HOOD 的特点。

(1) 被动对象。结构设计中的对象多为被动对象，其图形表示与 Booch 等人所采用的对象图元类似，如图 3 所示。

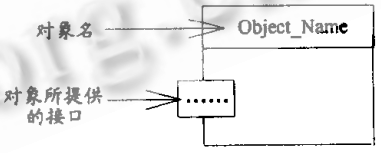

图 3　被动对象图标

(2) 主动对象。如果在对象图的左上角有一个标记"A"，就说明该图形所表示的对象是主动对象。被动对象中的所有操作都是不受约束的，而对主动对象来说，只有一部分操作是不受约束的，另一部分是受约束的。一个操作是否为受约束的，取决于下述两个方面：

①对象的内部状态。即该操作的执行依赖于对象内运行的其他操作的状态。换句话说，操作的语义依赖于内部逻辑条件，而这些内部逻辑条件是通过对象内其他操作的运行确定的。

②运行请求的类型。被使用对象接收到一个请求或刺激后，在其内部将产生新的控制流，以作为对该请求的响应。对某一操作来说，有 HSER、LSER、ASER、TOER 四种类型运行请求，它们可以用来表示对操作的约束，其中，HSER 表示高级同步执行请求，LSER 表示松散同步执行请求，ASER 表示异步执行请求，TOER 表示定时执行请求，如果在其接口所提供的操作前面冠以"Z"，就表明该操作是受约束操作。如图 4 所示。

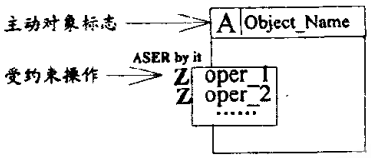

图 4 主动对象图示

(3) 三种关系。图 1 已经表明了三种关系的描述，其中，实线箭头表示使用关系，虚线箭头表示实现关系，对象 X 框中的子对象框表示父对象与子对象之间的包含关系。需要说明的是，对象之间的使用关系形成了对象间的控制流，而所有使用关系的箭头都应当是从高层指向低层的，如果在图中出现了环，那么，这种使用关系就是不合法的。如果对象 A 使用对象 B，则对象 B 对 A 的子对象是可见的，并且称对象 B 为对象 A 的子对象的叔父对象。为避免形成环，HOOD 中不允许一个子对象使用其父对象。包含关系应当遵循对象的主动和被动性质，如果在分解的过程中没有违反父对象的被动性质，一个被动对象所包含的子对象可以是主动对象。但是，一个父操作不能直接由受约束的操作实现。实现关系是 1:1 映射关系，而且一个受约束操作应由一个受约束的子操作实现。当一个非受约束操作由多个子对象的非受约束操作实现时，通常引入类型为 OP-CONTROL 的专用对象来表示一对多的映射关系。OP-CONTROL 对象的命名形式为：<父对象名>.<子对象名>，通常用一个没有提供接口的对象符号表示。OP-CONTROL 对象总是终端对象，不能被分解，在其提供的接口中仅有异常操作。此外，父对象的操作也可以由子对象的一个操作集实现，操作集是为多个操作提供的速记名，通常由花括号({})内的一个名字表示，如{operation-set-name}。

(4) 主动对象的控制结构分解。主动对象控制流的交互是通过对象的控制结构来描述的，它们可以按照如下两种方式进行分解：

· 对象控制结构由一个专门的子对象来处理，并命名为 CTRL-Object-Name。由于控制流的交互作用是在对象内部，因此这种专门对象是主动对象。

· 对象控制结构由多个专门的对象处理。

(5) 异常流。ADA 中的一个重要概念就是异常，异常是一种用于处理程序运行过程中所出现的错误或非预期事件的设施。当程序处于非正常状态时，就导致一个异常，然后由称为异常处理程序的另一部分程序处理。在 HOOD 图中的控制流上加上一条与其垂直的短线就表示存在异常流，并在一侧标上异常名，其方向与控制流的方向相反。

(6) 环境对象。早期 HOOD 的用户发现他们需要描述与系统外的对象或软件的接口，因为需要将系统被分解为多个部分，或者需要使用另一已有软件，这就导致环境对象的出现，它不是系统的一部分，仅有一对外提供资源的接口。

(7) 类对象。类对象是一个专门设计用于复用的对象，其操作的参数类型或数据没有完全定义，当这些类型和数据被完全定义之后，就得到了一个类对象的实例，即为设计创建了一个实例对象。类对象的类型可以是主动对象、被动对象或虚拟结点对象。类对象可以被分解为主动对象或被动对象，可以使用环境对象，也可以包含其他类对象的实例，在欲设计系统的每层都可列出若干类对象，但类对象不得使用欲设计系统的对象。

在 HOOD 图中，类对象用包含实例名和类名的单个对象符号来表示，若实例对象使用了环境对象，在创建实例对象时，应当把环境对象增加到要设计系统的环境中，但在图形表示中，只要给出实例即可。

(8) 虚拟结点对象。为了适应分布式软件的设计，HOOD 方法中引入了虚拟结点对象的概念。一个分布式应用可被看作一个由相互通信的虚拟结点构成的网络。在对象符号的左上角标上一个字母"V"，就表示该对象为虚拟结点对象。

此外，HOOD 图还能反映出对象间的数据流，通常以小圆圈表示数据，箭头表示数据流向，并将它们画在控制流的一侧，然后标上数据名，数据名可以是一个或多个数据项，也可以是一个表，但它们都是非形式化的。

HOOD 图的构造以一个对象所包含的子对象及其关系的描述为基本单位，图 1 所示即为对象 X 所包含的子对象以及它们之间的三种关系情况。对象的分解以操作

的实现和对象描述的抽象程度的降低为主要制导线。若干对象图构成了一棵设计树，反映了欲设计的系统。

2. **对象描述框架** ODS

ODS 是 HOOD 提供的文本形式的规约语言，用 ODS 描述的对象具有一定的框架结构，因而可以进行完整性和一致性检查。限于篇幅，仅对 ODS 所描述的 6 个方面进行简单说明。

(1) 对象级描述。对象级描述部分不仅说明了对象名，而且用自然语言描述了对象的主要功能。此外，在一个被称为实现或同步约束的域中，有一些动态方面的描述作为补充，包括同步、控制顺序、和其他特别的设计要求。与硬件中断、定时器、死锁预防机制相联系的描述也被包括在这一部分。

(2) 对象所提供的接口。对象所提供的接口部分描述了对象能够向其他对象所提供的全部资源，是对外的接口，一般采用结构化英语形式描述，其内容包括：①类型：封装在对象内而能被其他对象使用的数据类型。②常量：能够被其他对象使用的常量。③操作：比 HOOD 图中的操作更具体，增加了操作的参数。④操作集：操作集部分列出了操作集的名称。⑤异常：包括了所有的由异常流标识的异常。

(3) 对象所需接口。对象所需要的接口部分列出了对象所需要的一切外部资源，包括：对象、环境对象、类对象、类型、操作、异常等。

(4) 对象控制结构 OBCS。OBCS 是专为主动对象设计的，以自然语言的形式描述了操作的实时约束，以及与受约束操作相联系的外部事件，并用伪码描述了对象交互。此外，OBCS 还描述了一个父对象的 OBCS 如何由多个子对象的 OBCS 实现。

(5) 内部描述。内部描述说明了出现在对象内的实体，如子对象、需要定义的类型和数据、一些内部的操作或公共操作等。定义内部描述需要较多的信息，只有当对象被进一步分解后才能定义。由于对象的分解不会在一次完成，因此内部描述的定义很难一步完成。

(6) 操作控制结构 OPCS。操作控制结构部分描述了每个输出操作的参数、所使用的其他对象的操作、实现算法等。具体包括如下几个部分：①声明：声明部分定义了操作的参数名、模式(IN, IN OUT, OUT)、类型、描述以及函数的返回值类型。② 描述：以自然语言形式对操作的描述，并扩展了设计细节。③ 接口：说明了接口的定义，该操作所用到的其他对象的操作也被列出和定义，还描述由此操作所引起的异常。④代码：代码部分以伪码的形式描述了操作逻辑。

五、评价与改进

OOD 通常使用抽象、信息隐蔽、继承等原则控制复杂性。目前，常采用评价标准有：耦合性、内聚性、可复用性、设计的清晰度、一般 - - 特殊结构的深度、服务的简单性、设计中公认的优雅风格等。当前，比较具有代表性的 OOD 方法有 Booch 的 OOD、GOOD (General Object Oriented Development)，Coad - Yourdon 的 OOD，OOSD (Object Oriented Structure Design) 等。与上述方法相比，HOOD 方法具有下述优点：

(1)层次性好，它采用自顶向下的分解方法，确定了软件结构之间的低耦合、高内聚的特点，因此易于测试和集成。

(2)HOOD 图清晰，其设计过程简单明了，因此易于维护和扩充。

(3)环境对象和类对象的引入，使其具有一定的复用能力。

(4)HOOD 较好地结合了结构化设计方法和面向对象方法，在很大程度上克服了许多对象方法对与系统功能刻划不足的问题。

其不足之处体现在 HOOD 对复用的支持比较有限，尽管 HOOD 的类对象概念支持复用，但 HOOD 不支持继承，这可能主要是因为 HOOD 的设计背景是面向 Ada 的缘故，而 Ada 被公认为是基于对象的程序设计语言。此外，HOOD 方法的形式化程度不够高，它使用图形化语言和结构化英语描述软件设计，属于半形式化形式，这有可能导致规约的二义性。

参考文献

[1] 蔡希尧，面向对象技术述评，计算机应用，1996，4

[2] HOOD Working Group, European Space Agency, HOOD Reference Manual, WME/89 - 173/JB, September 1989, Issue 3.0

[3] P Robinson, Object - Oriented Design, Chapman & Hall, 1992.

(来稿时间：1997 年 12 月)