面向对象 程序设计 软件工程

计算机科学1995 Vol. 22 №. 2

31-34

IDEF4与面向对象设计方法

周之英 王 萍
(清华大学计算机系 北京100084)

TP311.11

摘 要 IDEF方法是由美国空军并行工程信息集成项目的Armstrong实验室研制的，IDEF4是其家族的一员，本文以IDEF4为主探索了各主要面向对象设计方法的特点，并对IDEF4方法的优缺点作了简要分析，旨在本文能作为研制一种实用的面向对象设计方法的基础，为设计面向对象计算机辅助软件设计工具提供参考。

关键词 面向对象设计方法，IDEF4方法，软件工程，并发工程，再造工程，信息集成。

一、引言

面向对象技术是九十年代计算技术关键且热门的技术之一，面向对象方法可以产生据具有良好特性(如模块化、可维护性、可复用性)的代码，也使编制代码更容易。但正因为如此，人们通常错误地使用了强有力的面向对象方法，导致低质量的设计，产生糟糕的代码，从而造成模块性差、难以维护、难以复用的软件。因此研究如何正确地使用面向对象方法，也是近年来在面向对象方法的研究日益高涨情况下，面向对象的分析和设计日益受到重视的原因。

目前国际上对面向对象设计方法研究的著作不少，大多从软件工程的角度对OOA、OOD、OOP作系列研究，试图寻找三者之间的平滑过渡。目前比较有影响的有Grady Booch提出的综合性的OO设计方法；Peter Coad和Ed Yourdan提出的OOD设计方法；Wirfs-Brock等提出的责任驱动设计方法(RDD，即Responsibility Driven Method)；Jacobson提出的应用驱动设计方法以及由Armstrong实验室提出的IDEF4方法等等。

应特别指出的是IDEF4方法，该方法是IDEF家族的一员。IDEF方法由美国空军并行工程信息集成(IICE，即The Air Force Information Integration for Concurrent Engineering)项目的Armstrong实验室研制的。IICE工程的主要任务是发展理论基础、方法和工具以成功地实现和推进信息集成企业。这些技术把信息和知识资源作为得到高质量系统的关键。IDEF方法族现在已广泛用于支持企业集成，作为并发工程、全面质量管理和商业再造工程(Business Re-engineering)的基本组成。

复杂系统的体系结构表示不是单一的，而是一个集合，集合中的成份是相辅相成的。试图寻找单一方法来描述系统所有信息非常困难，而采用多个方法会导致方法之间难以集成，IDEF方法族通过提供特殊机制来集成各种方法，试图在包含所有需要信息的"超级方法"和只能限于描述系统某专一方面的多个方法之间作一权衡。

独立使用时，IDEF方法有良好能力，可以实现对象事实的收集、分析、设计和构造活动；IDEF整体可以看成一个互补的工具箱，在那些整体的竞争能力越来越多地依赖于对企业信息和知识的有效掌握、管理和使用的环境中帮助人们提高工作效率。

IDEF家族现已发展到包括十多个成员，从IDEF0到IDEF14，IDEF0建立系统的功能模型；IDEF1X建立语义数据模型；IDEF2描述系统资源的时变行为；IDEF3是一种场景驱动的过程流建模方法；IDEF4是面向对象的系统设计方法；IDEF5为知识获取方法；IDEF6着重于信息系统设计基本原理的研究等等。其中IDEF0、IDEF1X已有较成熟的产品，并使用广泛；IDEF2很少使用；IDEF3、IDEF4正处于有效性测试阶段。

二、IDEF4方法

1. 概念

面向对象的方法通过对象及这些对象相互作用的协议来定义系统行为。IDEF4着重以下内容的定义、操作、显示和分析：

(1)对象定义——包括对象的属性及其类型的

定义；

(2)对象结构——包括继承的层次或网格关系、单个对象与其它对象相关的组成；

(3)单个对象行为——包括方法说明与约束条件(前提、后置和进行条件)。

(4)系统运行协议——包括对象例程在对象继承图中的定位、例程的参数情况等。

作为一种设计方法，IDEF4用于面向对象系统的设计，它使用单一的组织机制以保证模型不会因项目规模的增大而变得繁琐和难用。

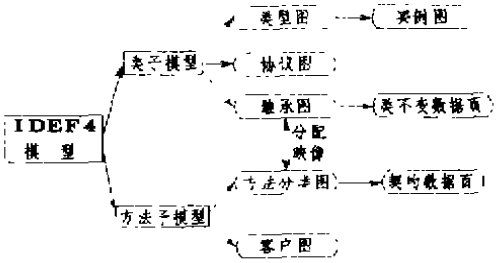

图1 IDEF4模型

从概念上讲，IDEF4方法由建立一系列图形组成的模型与一个开发过程组成，允许开发人员和实现人员从分析、设计阶段开始一直到编码实现为止加强、改正和维护同一个一致的模型。一个 IDEF4模型由两个子模型组成：类子模型和方法子模型(见图1)，这两者通过分配映像相互关联，几乎包含了设计模型的全部信息。子模型是一特定类型信息的各种内容的组合。类子模型由协议图、类型图、继承图、实例图和类不变数据页组成，描述类的继承关系和类的组成结构；方法子模型由客户图、方法分类图和契约数据页组成，用于描述系统的各种特定行为；分配映像用于表示类中的例程与方法分类中的方法之间的对应关系。

2. IDEF4中的图

IDEF4采用了五种图来创建设计模型，它们组成两类子模型，下面对它们作一简要介绍。

类继承图(Class Inheritance Diagram)。以网状结构显示系统中的类继承层次。图中方框(BOX)表示类，盒子之间的箭头表示类之间的继承关系。类盒分为三部分：私有特征、公有特征和类名；箭头从超类指向子类。图中类特征前的符号(如?、$、@等)表示特征的实现方式(函数、过程、槽等)，见图2。

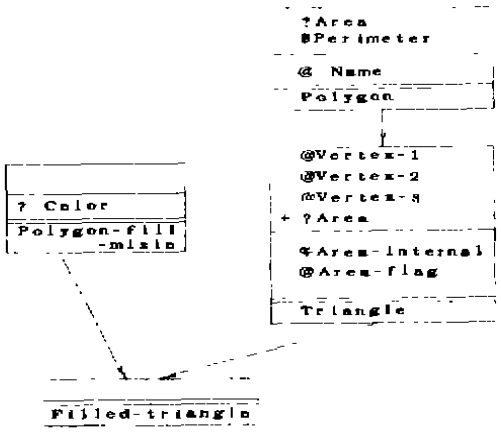

图2 继承图

类继承图中涉及的每一个类都将有一个类不变数据页(Class Invariant Data Sheet)与其相联。CIDS中描述类的定义，包括设计者确定的该类实例所拥有的特征值或对象联系，可以用于维护系统，提供设计文档。

方法分类图(Method Taxonomy Diagram)。从行为相似性来区分各种方法，根据在分类中表示的加于方法集上的约束来确定一类特定的系统行为。如图3所示，各种排序方法以方框表示，方框之间的箭头从较粗略的方法指向较细化的方法。

方法分类图中的每一方法集都有一契约数据页(Contract Data Sheet)与之相联，用于描述该方法集中方法实现时必须满足的限制，它是方法实现的依据。

类型图(Type Diagram)。主要表示类之间属性上的联系，即某类实例可能是另一类实例的一个属性。如图4所示，一辆具体汽车的属性——车轮是轮子这一类的实例，是一种组成关系(A-PART-OF)。

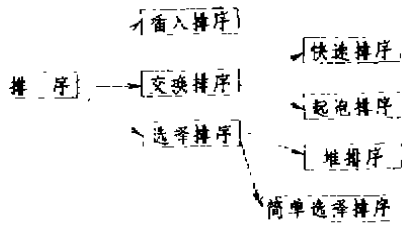

图3 方法分类图

客户图(Client Diagram)。是方法子模型的一部分，用于表示算法分解。它是 IDEF4图中唯一表示方

法内部结构的图，如图5所示，类 Redisplayable_object 的例程 Redisplay 将使用类 Erasable_object 提供的 Erase 和类 Drawable_object 例程 Draw 提供的服务。

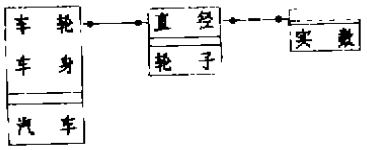

图4 类型图

协议图(Protocal Diagram)，描述使用方法时的参数类型，即方法被激活时传递给它的参数的类型和方法返回时返回值的类型。如图6中，Fill_closed_object 方法将需要两个参数：自身参数(即类 Polygon 的某一实例)和填充的颜色 Color 的实例，返回结果 Result 仍是一多边形。

分配映像(Dispatch Mapping)，表示类继承图中类特征与方法分类图中方法之间的对应关系。它可以在继承图中在相应类的例程名后写上方法集名表示，也可以在方法分类图中方法集名后加上【类名：例程名】表示。

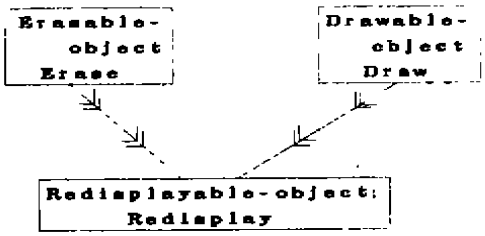

图5 客户图

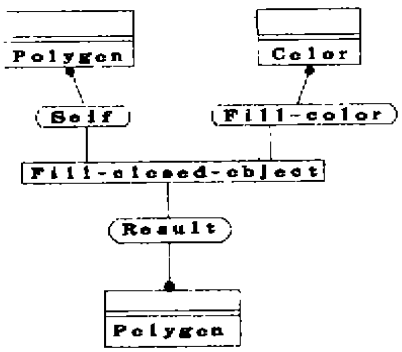

图6 协议图

3. IDEF4设计开发过程

IDEF4将设计开发过程看成是由一系列不断前进的、有序而相关的步骤组成，每一步都可以相对独立地完成一部分任务。在软件设计中大致使用以下过程：

(1)分析不断变化的系统需求；

(2)开发类层次；

(3)进行方法分类；

(4)分析类结构，形成类型图；

(5)开发协议图；

(6)进行算法分解，形成客户图。

此外，在设计过程中还可以借助于已有的 IDEF 模型，从中抽取需要的信息，加快设计开发进度。

三、IDEF4与其它各种 OOD 方法的比较

目前，国外已提出各种 OOD 方法，这些方法看起来五花八门，可究其本质，它们是共同的，即从对象的角度对系统建模；都支持面向对象方法中的类、对象、继承、抽象和封装等概念，并以这些概念为基础，从不同的视角描述系统的以下信息：(1)对象、类本身的组成结构；(2)对象、类之间的关系。

这些信息又可以分为静态信息和动态信息。静态信息包括对象、类的组成结构、类之间的继承关系、对象之间的组成关系等；动态信息包括系统和对象的状态转换、对象类之间的消息联系、事件的执行流程等。

然而，IDEF4方法和其他 OOD 方法又存在许多不同之处，包括建模角度、使用的特殊的概念、不同的图形(见表1)。由该表可以看出，IDEF4方法有其新颖的有用的一面，也有其不完善的一面。

IDEF4方法的一个显著特点是将系统行为抽象出来形成一个单独的子模型，强调行为与对象的结构同等重要，这是其他 OOD 方法所不具备的。其实系统的行为和类层次一样，都可以成为复用的资源(如"排序"这一古老的话题)，不过这种复用和类的复用一样，在实现时有一定的困难，因为没有哪一个设计者愿意花费精力去设计通用的，可复有的方法层次(网格)，他只想尽快完成自己这一具体的系统设计。

另一值得注意的特点是，该方法对类特性进行了详细的划分，并且提供了丰富的图形语法来描述特性的具体含义。如用继承图描述类层次和类的具体结构；用协议图描述特性的输入输出特性；用类型图描述属性的值的类型；用客户图描述例程的组成等。这种描述的详细性易于在 OOD 工具中实现代

码的自动生成。

虽然IDEF4方法有其独到之处，但也并非完美无缺，前面已提到过，系统的信息分为动态信息和静态信息，而IDEF4除了在方法子模型的客户图中简单地提到过方法分解之外，只提供了数据页机制，用文字描述各种限制或契约或事件流程；而没有如状态转换图、事件流程图及消息传递等描述动态结构的图形语法。

此外，IDEF4中也未提及子系统的概念，尽管在描述类层次时可能应用了该思想。在一个大系统中，类及对象的数量是庞大的，应提供一种管理它们的机制，子系统将类(对象)按一定的方法分组，从而形成层次的系统结构，我们认为子系统是一个有用的管理手段。

还可以看到，IDEF4中使用了大量图形语法，这些图有的很小(如协议图和客户图)，使得复杂系统的图形文档相当庞大，而且方法分类图与类继承图有一定程度的重复。

表1 各种OOD方法的比较

OOD方法	主要模型	文档形成	评 价
Grady Booch 综合OOD方法(1986，1987，1991，1993)	从多个视角描述系统(动态、静态、逻辑、物理等)	类结构图、对象结构图、模块结构图、状态转换图、时序图	具有丰富的图形技术，但过于丰富，没有形成一个完整的设计过程，使人感到是一堆技术的堆砌
Coad/Yourdan OOD方法(1991，1992)	设计从四个方面(人机界面、论域、任务管理、数据管理)和五个层次进行	主题层结构图 类与对象层次结构图 属性与服务层次结构图	方法比较成熟且实用，但缺少对功能语义、全局控制的支持，使用时需加以改进
Wirfs-Brock et al. 责任驱动方法(1990)	以卡片形式记录组成系统的类结构、类的职责及类与其它类之间的协作	CRC卡片 契约数据页 子系统组织结构	是一个应用非形式化技术引导开发的方法，不适用于大型系统的开发。但有些概念值得借鉴
Lvar Jacobson 应用驱动设计方法(1992)	系统的分析和设计围绕应用进行，将各种功能分配到不同的对象中去，由它们相互协作完成系统功能	应用模型 方块图 交互图	是一个由经验而来的系统工程方法，深入掌握，不失为一个有效方法。
Armstrong 实验室IDEF4方法(1992)	从系统的组成结构和行为两方面来把握系统。(建立类子模型和方法子模型)	继承图、类型图、客户图、协议图、方法分类图、实例图、契约数据页、类不变数据页	视角比较新，易于码的自动生成。但缺乏动态语义

(上接第37页)

机构无关。

可审查性指测试相关信息应记录在案，完备无缺，以备审查。

STEP一致性测试方法论体现了STEP实现须遵循的标准性和约束性。和其它技术的实现不同，STEP实现应认真考虑如何满足一致性要求，以维护STEP标准的宗旨。

五、结论

STEP的开发、应用和测试方法论是相互联系、有机组成STEP方法论。了解、掌握STEP技术，应用STEP技术进行开发，测试STEP实现，均须在STEP方法论指导下进行。

参考文献

[1]ISO CD10303-1，"Product Data Representation and Exchange—Part1：Overiew and Fundamental Principles"，ISO TC184/SC4，Sept. 1992

[2]ISO CD10303-203，"Product Data Representation and Exchange—Part203—Application Protocols：Configuration Controlled Design"，ISO TC184/SC4，Aug. 1992

[3]ISO CD10303-31，"Product Data Representation and Exchange—Part31：Conformance Testing Methodology and Framework：General Concepts"，ISO TC184/SC4，Jan. 1992