分布对象计算 继承机制 面向对象 软件工程 (18)

计算机科学2000Vol. 27№. 2

69-72

分布对象计算中继承机制的研究

On Inheritance Mechanism in Distributed Object Computing

艾丽蓉 刘西洋 蔡希尧

(西安电子科技大学软件工程研究所 710071)

Abstract It is the basic tendency of distributed object technique to evolve from centered-computing to Client/Server computing, and then to the net-centric computing. This paper first discusses the semantic characteristics of net-centric computing and the necessity of employing inheritance. Compared with the inheritance mechanism in centered-computing and Client/Server computing, also with the inheritance based on classes and the delegation based on prototype, we propose multiview open inheritance as the inheritance mechnism of net-centric computing, in which the interface inheritance and the implementation inheritance are integrated.

Keywords Net-centric object computing, Inheritance, Delegation, Interface inheritance, Implementation inheritance

1. 引言

以 Web, CORBA, Java, ActiveX 以及对象-关系数据库的集成为先导,以 Oracle 的网络计算体系结构 NCA (Network Computing Architecture, http://www.oracle.com/nca)为实例,分布对象技术在经历了从单一环境、单一地址空间、单一语言的集中计算到客户/服务器(Client/Server)计算之后,正在进化为基于 Internet/Intranet 的以网络为中心的计算。"Internet 最终无可置疑地将对象技术纳入计算技术的主流"[1]。以网络为中心决不简单地等同于以 Internet/Intranet 为中心,其深刻的内涵是:

·以大规模的计算机网络为中心的基础计算设施,计算最终映射为网络上的复杂协调操作;

·以软构件网络为中心的软件构造方法。以离散的、满足统一接口规范的软件构件互联建造大型软件系统,从而软件系统的构造方法由紧耦合的单块式方法进化为软构件互联,以超拓扑信息网络 Web 和分布对象网络 CORBA, Java 及 ActiveX 为中心的软件组织和构造;

·以网络为中心的需求分析和软件设计面向系统、面向整体、面向领域,以设计样本、软件体系结构和特定域软件体系结构(DSSA)为突出代表,软构件网络是其中承上启下的中间环节,而软构件存在的前提是统一的软构件体系规范。

由此充分说明,分布对象技术的应用环境和需求正在拓展和深化,关于分布对象技术的传统理解和假设因此而面临严峻的挑战。

2. 继承机制的引入

继承性被普遍认为是面向对象范型区别于传统结构化范型的基本特征,是区分基于对象与面向对象的重要判据。从面向对象的定义:面向对象=对象+类+继承性[3]不难看出其重要地位。异构分布环境下的继承机制是跨越不同的网络协议、不同体系结构、不同程序设计语言、不同地址空间的动态层次式资源共享机制,它比传统的继承机制更开放、更自然。以网络为中心的对象计算不可能,也不应该回避继承性,其原因在于:

(1)以网络为中心的对象计算的语义完整性。以无环路的层次结构(树或有向无环图)组织给定上下文中的相似概念是人类认识的基本方法,同时树和有向无环图是计算科学中应用最为广泛的非线性数据结构,而这正是继承的认识论基础。与此同时,AI 关于继承推理的研究实践表明,继承语义具有普遍的数学理论基础(以文[4]为代表)。上述认识论基础和数学理论基础并不会因对象技术应用于以网络为中心的计算环境而动摇。

代理(Delegation)[5]是在经典继承机制之后提出的以原型对象(Prototypical Object)而不是以类为中心的对象组织和共享机制。传统观点认为,代理适合于并发、分布计算,而"继承与分布是不一致的[3]",似应在

网络计算环境中排斥继承性。然而，对象技术的研究表明：

·继承与代理的争论在很大程度上归结为类与原型的争论，传统意义上的基于类的继承与基于原型的代理在概念上统一于对象继承机制。

·继承与并发、分布的不一致性很大程度上缘于传统继承语义和实现的封闭和僵化，而不是继承性概念。即继承语义固化于OO语言或环境的编译器或解释器之中，继承性的实现往往并不通过显式的消息传递而是借助类似指针的机制，其隐含的基本前提为：继承语义的作用域约束在同一地址空间中由统一OO语言或环境定义的对象系统，因而在网络计算环境下实现继承机制必然面临种种协同问题（文[3][6]已初步涉及）并未全面暴露出来。

因此，代理机制的提出以及继承与并发分布的不一致性并不能成为在以网络为中心的计算环境中排斥继承性的理由，而恰恰说明继承机制的语义和实现亟待研究和深化。

(2)递增式软件开发和软件规格说明的结构化。软件工程中，基于继承性的软件重用在软件开发过程中起着重要作用，例如，主流的工业化OOPL（如C++和Eiffel、Smalltalk，CLOS，及Java等）均以继承性作为其共享机制；典型的OO开发方法（如Coad-Yourdon方法，Booch方法，OMT，OOSE，VMT方法等）也均以继承性作为其基本概念；当前的软件重用概念和技术如软件体系结构、设计样本都以继承性作为其实现共享的机制之一。总之，继承性在OO软件开发的实质性作用可以概括为：

·继承性作为有序、规范的软件递增式修改机制支持递增式软件开发；

·继承性允许创建不完全实现的概念抽象，支持分析、设计和实现过程中软件规格说明的逐步求精，从而作为软件规格说明的结构化工具。

(3)OO范型的统一性和连续性。继承性是由一般到个别，由共性派生出个性。这种语义在客观世界中是普遍存在的，以网络为中心的对象计算也不应该排斥继承性，否则将会造成面向对象范型在各计算环境中的严重不一致性。可以肯定，OO范型从概念上与以网络为中心的计算环境是一致的，但如果在上述环境中排斥目前工业界主流的基于类或接口的继承机制而代之以基于原型的代理，则必然导致OO范型在传统集中环境与以网络为中心的计算环境之间的严重不一致性，从而损害其统一性和连续性。

但同时应当看到，尽管学术界和工业界很早就开始研究网络计算环境下的继承性，但直至目前仍然没有在继承机制的语义和实现上形成共识，在工业化主流分布对象系统中往往回避继承性[3]，直接表现在：

·分布对象计算的国际标准开放分布处理参考模型RM-ODP在定义实现继承[5]时指出："ODP系统并不要求支持实现继承"，实际上在规范中回避了网络计算环境下的继承问题。

·分布对象计算的工业标准CORBA2.0[3]尽管在OMG核心对象模型和接口定义语言中引入基于接口的多重继承机制，但既不支持多重继承中名字冲突的解析，也不允许子接口重新定义超接口的任何操作，从而回避了继承层次中的过载和重置，并且整个规范自始至终都没有明确涉及网络计算环境下接口继承[6]的确切语义和实现策略。

·Java同时引入类和接口的概念，支持基于类的单重继承和基于接口的多重继承。类与接口之间存在着有向的多对多的"Implements"关系（一个类可以同时实现多个接口，一个接口也可以同时被多个类所实现），类与类、接口与接口之间的继承关系以及类与接口之间的"Implements"关系从整体上建立起Java的立体继承层次，从而以统一的对象概念实现了接口继承与实现继承的集成。尽管如此，Java继承机制的作用域仍然局限于同一Java虚拟机，并没有克服传统OO语言中继承机制的封闭性。但是应当看到，在实现Java Applet中引入的继承层次的动态复制和重建技术正在为网络计算环境下继承机制的实现提供崭新的思路和方法。

3. 传统继承的语义封闭性

以网络为中心的对象计算与集中计算和客户/服务器计算相比在开放性、动态性、复杂性等方面都存在着质的差别。由于前提和背景发生了根本变化，导致对继承机制的要求也随之改变。集中计算环境下的继承机制语义上是封闭的，其实现上是基于指针的，而以网络为中心的对象计算环境中的继承机制语义上是开放的，实现上是基于消息传递的，故不能直接从集中环境中引入继承机制。尽管CORBA[9]和SOM/DSOM[10]已经提出了接口和接口继承的概念，但仍远不能适应以网络为中心的对象计算对继承机制的要求。传统的继承性无法适应以网络为中心的对象计算的特点和要求，因为继承性在语义上是封闭的。

3.1 C++中继承性的语义

在集中环境下，传统的继承性是以紧耦合隐式上下文为特征的，即子类的结构和行为强烈依赖于其父类的结构和行为，并且父类定义的结构和行为隐含传播至子类。

C++语言作为一种典型的强类型的OOPL，在工业界被普遍使用。以下就以C++为例讨论在集中环境

下的继承机制存在的问题：

·易碎基类问题。由于子类方法的实现与接口一致的父类方法的实现出现不一致，导致子类与超类之间的继承语义的失效。在C++中，不论在任何时候往一个类中添加新的方法或实例变量，所有引用它的类都要求重新编译，否则它们将会崩溃。

·多重继承的异常导致的熵增问题[6]。以网络为中心的环境是一个高度开放的大规模复杂环境，在此环境中计算资源的分布（主要是逻辑分布）和各自的属性之间的差异是必然的。这种资源的分布和差异必将导致实现细节的差异。传统的集中环境中基于代码共享和以紧耦合为特征的继承机制将不再适用于这种环境。

3.2 客户/服务器环境中的继承机制

接口是针对概念实现的提供者与其客户之间关于提供者能向客户提供哪些服务的契约的描述。接口实质上是对概念实现的外特性（而不是内部实现细节）的集中描述。CORBA中仅仅定义了接口继承，对接口中方法的实现不作约束，接口中可以既有属性也有方法，因此其开放程度仍不够高。

4. Java中的继承语义

面向Internet的OOP语言Java中也支持接口继承。Java中的接口只提供一系列方法的规格说明，而不含实例变量及方法的实现（见图1中的接口Ia）。也就是说，接口中的方法与其实现是独立的。

```
interface Ia{
    void methoda();
};
interface Ib extends Ia{
    void methodb();
};
interface Ic extends Ia{
    void methodc();
};
interface Id extends Ib, Ic{
    void methodd();
};
public class C1 implements Id{
    public C1(){
        ...
        OverridedMethod();
    };
    void OverridedMethod(){…};
    ...
    public void methoda(){…};
    public void methodb(){…};
    public void methodc(){…};
};
public class inheritance{
static public void main(String args[]){
    C1 m1=new C1();
    C4 m4=new C4();
    m1.methoda();
    m1.methodb();
    m1.methodc();
    m1.methodd();
    m4.test(m1);
}};
```

```
public class C2 implement Ia,Ib{
        ...
    public void methoda(){…};
    public void methodb(){…};
};
public class C3 implement Ia,Ic{
        ...
    public void methoda(){…};
    public void methodc(){…};
};
```

图1 Java语言接口继承示例

其继承层次的拓扑图如图2所示。

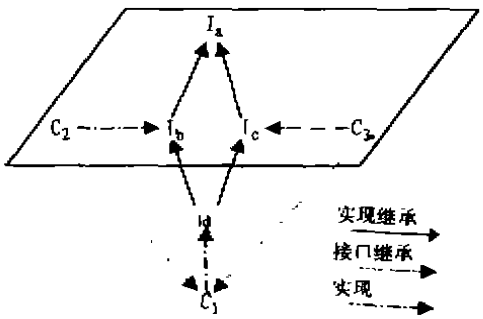

图2 Java语言接口继承实例的继承层次

·在以网络为中心的对象计算环境中的资源随时都有可能在更新，这就要求在这种环境下的软件应具有高度的动态性。否则，易碎基类问题还是得不到解决。Java解释执行的特点使得Java程序能在运行时动态地装入被引用的类。这样一来，如果在父类中添加实例变量或方法，只须将父类重新编译，而引用它的那些类就不需要再重新编译了。这一点对于开发大型的软件系统是很重要的（尤其是在维护阶段）。

·Java中一个类实现一个接口时，必须在这个类或其子类中实现该接口中所有的方法。Java中一个类也可以实现多个接口，如图2中，类C2就同时实现了接口Ia和Ib。同样，类C3同时实现了接口Ia和Ic。Java中一个接口也可以被多个类实现，并且在运行时动态联编。图2中的接口Ia分别被类C2和C3实现，methoda()在运行时根据具体环境判断执行C2中的实现还是C3中的实现。不难看出，接口完成了从对象到方法的映射，起到了统一不同实现细节的作用。由于接口中不含实例变量，就不存在父结构的共享与非共享问题，也不存在多个直接父类的继承顺序问题。不同接口中同名方法的区分问题也没有了，因为这些同名方法在一个类中只对应唯一的实现。

·为了避免集中环境下继承性的问题，Java对实现继承只允许单重继承，而接口则可以被多重继承。与C++中的继承机制相比，子类对父类的依赖性减弱了，

其语义的复杂性也有所降低。

·以网络为中心的对象计算环境的另一特点是流动性。现在的 Internet 网上每时每刻都有无数的信息在传送。Java 将接口中的方法与其实现分离，并且在运行时动态地装载被引用的类、这种措施在一定程度上有利于资源的流动性和安全性。

上述继承机制的特征正是以网络为中心的对象计算环境的特点所要求的。一个接口对应多个实现同时也使得这种继承机制的开放性得以提高。

Java 中接口继承的这种语义是以网络为中心的对象计算所需要的。同时、为了贯彻 OO 的概念、实现继承同样必不可少。总之，以网络为中心的对象计算中的继承机制应该是集成接口继承和实现继承两者的多元化开放继承体系。

5. 多元化开放继承体系

由于以网络为中心的环境的高度动态性，多元化开放继承体系的首要任务是保持网络中资源的一致性。在实现继承和接口继承的类层次中、当类或接口被改变，或者它们的继承关系被改变时，所有引用实现它们的类也应该改变以保持一致性。

·假设在网络协同开放过程中，一个节点上的类实现另一个节点上的接口。接口中方法的改变(如增加或删除方法)应该为 Web 服务器所知到。于是、服务器发出通知给实现接口的类所在的节点，有关的实现被更新。在这个过程中，Web 服务器是软件开放的协调者和管理者。

·在运行时也有类似的情况发生。Java 中的类装载器(classloader)可以动态地装入更新后的类。实际上，类装载器完成此功能有赖于 Web 服务器的帮助。只是当接口被修改时处理要更困难，因为方法的实现并不在接口中，而是在实现它的类中，这就要求类装载器和 Web 服务器有更强的功能。Web 服务器的位置如图3所示。

总之，在 Web 服务器的协调之下，多元化开放继承体系能更好地适应以网络为中心的环境。以网络为中心的对象计算中的多元化开放继承体系仍然存在一些问题，因为其高度的流动性和动态性，管理就显得非常重要。如何管理流动的资源就直接影响到网络的可控性和有效性。

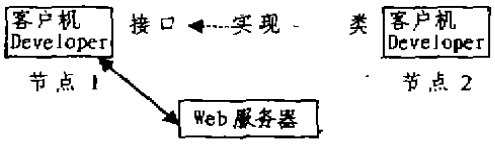

图3 多元化开放继承体系

参 考 文 献

1 Jeffery、et al Components on the Internet OOPSLA'96 Panel, ACM SIGPLAN Notices, 1996,31(10)

2 Taivalsaari A. On the Notion of Inheritance. ACM Computing Surveys, 1996,28(3)

3 Wegner P. Dimensions of Object-Based Language Design. ACM SIGPLAN Notices, 1987,22(12)

4 Touretzky D S. The Mathematics of Inheritance Systems. Morgan Kaufmann Pub. 1986

5 Lieberman H Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems OOPSLA'86 Proc.,ACM SIGPLAN Notices, 1986,21(11)

6 刘西洋. 异构分布环境下继承机制的研究.[西安电子科技大学硕士学位论文]. 1995

7 GTE Lab. ANSI X3117 Object Model Features Matrix, Inheritance and Delegation

8 Dvorak J. Conceptual Entropy and Its Effect on Class Hierarchies. IEEE Computer, 1994,27(6)

9 Object Management Group. CORBA 2.0-Interoperability-Universal Networked Objects. March 20,1995,OMG TC Document 95,3.xx

10 Sessums R、Coskun N. Object-Oriented Programming in OS/2 2.0 Using SOM. The Personal Systems Developer, IBM, Winter 1992

(上接第56页)

3 张云涛，龚玲，周伯鑫. 语义网络在开发工程数据库管理系统 CIMS/EDBMS 中的应用. 计算机研究与发展，1997,34(8):616～620

4 卢昭泉，陈德人，谭孟恩. CIMS 环境下基于对象的信息集成框架. 计算机研究与发展，1998,35(5):436～441

5 陈滢，樊惠娟，王能斌. 分布对象技术综述. 计算机科学，1997,24,(3):11～16

6 Otte R, Patrick P, Roy M. Understanding CORBA、Prentice Hall, 1996

7 Understanding Product Data Management. http://www.pdmic.com

8 Deen S M, Amin R R, Taylor M C. Data integration in distributed database. IEEE Trans Software Eng, 1987, SE-13(7):860～864

9 Bertino E, et al. Integration of heterogeneous database application through an object-oriented interface. Inform Syst, 1989,14(5):407～420

10 Tsao S S. An overview of product information management. http://www.pdmic.com

11 韩鑫，王建民，孙家广. 产品数据管理系统的构造框架和实现方法. 软件学报，1999,9(12):881～883