ISSN1000-9825 Journal of Software 软件学报 2000,11(1):67~72

基于数据分解的并发面向对象程序开发方法*

杨大军 吕建

(南京大学计算机软件新技术国家重点实验室 南京 210093)
(南京大学计算机软件研究所 南京 210093)

摘要 提出了一种从VDM-SL(Vienna development method-specification language)规约到并发面向对象程序的开发方法,这种方法基于DD VDM(data decomposition-Vienna development method).在此基础上提出了虚拟原子、服务并行和内部并行等概念,继而提出一种嵌套面向对象结构来体现这些功能.分别从共享量并行系统和分布并行系统的角度讨论了嵌套面向对象结构的实现技术.

关键词 形式化方法,数据分解,嵌套面向对象结构,并发面向对象程序设计语言.

中图法分类号 TP311

并发与面向对象的结合是当前程序设计语言的主要特点之一[1].然而,开发高质量、高可靠度的并发面向对象程序需要一套好的形式化开发方法.VDM(Vienna development method)[2]是当前较为成熟的形式化开发方法之一,它在顺序程序的开发中已有很多成功的应用[3].如何将VDM方法应用于并行程序的开发是一个值得探讨的问题.为使VDM具有开发并行程序的能力,文献[4~6]提出了著名的rely/guarantee方法,并进行了改进.近年来,这方面工作的重点开始转移到面向对象思想与VDM的有机结合上来[7,8].但是,基于VDM的并行程序开发方法还存在一些不足,因此,我们提出了比VDM更加一般的形式化方法DD-VDM(data-decomposition-Vienna development method)[9,10].目前,这种方法已得到国际上同行的注意和认可.文献[11]用较大的篇幅讨论了我们所提出的数据分解概念及其与VDM++[11,12]中的用于开发并行面向对象程序的退火步之间的关系,从一个侧面说明了数据分解在并行面向对象程序开发中的潜在作用.

本文采用一种与VDM++完全不同的途径来研究并行面向对象程序的开发方法,提出了一种基于DD-VDM的并行面向对象程序开发方法,并说明在某种意义上可用数据分解与转换相结合的思想来表达VDM++中的退火步,同时避免了后者为保证正确性而增加的附加推理部分.其基本思想是,从用VDM-SL(Vienna development method-specification language)[13]表示的软件规约开始,用DD-VDM中所提供的数据精化、操作分解和数据分解对其加以精化和分解,从而得到反映开发过程的树形结构.然后将此树形结构直接对应于一种嵌套的并行面向对象设计结构.最后,可根据需求分别从共享量并行系统和分布并行系统的角度将此嵌套的设计结构用一定的语言机制加以表示和实现,从而完成从VDM-SL规约到并行面向对象程序的开发过程.在此过程中,利用基于模型的VDM-SL规约与类规约的相似性,自然地将DD-VDM与面向对象思想有机结合起来,然后利用DD-VDM中数据分解获得的多个分解类间正交的特点自然地引入服务并行、内部并行和虚拟原子的概念.

1 基于DD-VDM开发基于对象的并发程序

1.1 DD-VDM简介

DD-VDM作为一种软件开发方法,与VDM一样包括形式规约和可验证的设计步.它的规约语言使用

* 本文研究得到国家自然科学基金(No.69873021)、国家863高科技项目基金(No.863-306-ZT02-02-03)、国家攀登计划基金和国家杰出青年科学基金(No.61525204)资助.作者杨大军,1972年生,博士生,主要研究领域为并发面向对象语言,形式化方法.吕建,1960年生,博士,教授,博士生导师,主要研究领域为软件自动化,并行程序形式化方法,面向对象语言和环境.

本文通讯联系人:杨大军,南京 210093,南京大学计算机软件新技术国家重点实验室

本文1998-11-13收到原稿,1999-09-03收到修改稿

VDM 的规约语言 VDM-SL. DD-VDM 与 VDM 的主要不同是为软件开发引入了数据分解设计步,它可以与 VDM 中的数据精化和操作分解交叉进行. DD-VDM 的主要思想是数据分解. 从概念上讲,数据分解包含模型分解、子规约形成、规约重构和操作分裂这 4 个步骤. 其中最重要的特点是同层子模型的正交性. 详细内容和例子见文献[9]. 在 DD-VDM 的基础上,我们提出了一种适合并发面向对象程序的开发方法.

1.2 主要思想

模型规约与面向对象的类规约间极大的相似性为我们开发面向对象程序提供了先决条件;数据分解带来的子规约间的不相交性为我们开发并发程序提供了基础,我们的主要思想正是基于此. 具体描述如下.

(1) 将 DD-VDM 应用于面向对象的框架. DD-VDM 是建立在基于模型规约的基础上的. 基于模型的规约主要由一状态集的定义和建立在此模型基础上的操作组成. 因此正如文献[7,8]描述的那样,可以将基于模型的规约看作是类规约,将模型的分解看作类的分解,其中模型中的状态集可以看作是对类的属性,而操作可以看作是方法. 于是,DD-VDM 包含了面向对象设计方法学.

(2) 开发内部并行和服务并行. 数据分解的关键特点是同层子规约间的独立性. 这样,如果原规约中的两个方法在新规约中被划分到两个不具有直接或间接分解关系的分解类中,那么这两个方法就可以被用户并行调用,于是提供了服务并行. 另一方面,在操作分裂过程中,一个方法可以被实现为对相邻分解类中方法的顺序调用. 因为同层分解类是彼此独立的,它们的方法间不存在干扰,因此,顺序地调用可以转化为并行的调用. 于是引入了内部并行性.

(3) 虚拟原子性. 根据默认的规则,一个对象在一个时刻仅能有一个方法被调用,方法的执行是原子性的. 但是在操作分裂中,一个原子动作 A 可能被分解为多个更深层次分解类中的更小原子动作的并行执行. 在此阶段,动作 A 已不是原子动作,但它能够达到原子动作的效果. 于是提供了一种虚拟的原子性. 虚拟原子性能够兼顾程序的正确性证明及其执行效率.

(4) 数据分解引入嵌套面向对象结构. 概念上,普通的面向对象结构本身不支持服务并行和虚拟原子性,于是,我们提出将嵌套的面向对象结构直接对应于使用 DD-VDM 方法得到的开发树,从而支持这种特殊的需求.

(5) 集成增加并行度的转换规则. 因为外层对象的方法可以限制内层对象,我们希望尽可能早地释放外层对象的方法以增加并行度. 在文献[7,8]中的转换规则用以对这一需求提供有效的支持.

1.3 例 子

我们使用文献[9]中 World 的例子说明我们的开发方法. 图 1 所示的开发树是经过数据分解得到的. 我们可进行下面的开发.

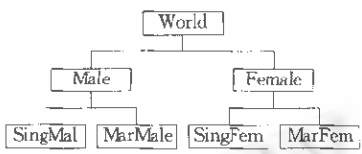


Fig. 1 The development structure of World
图1 World的开发结构

(1) 在对 Marriage 进行操作分裂后. Marriage 变为对分别处于两个相邻内层对象中的 MarriageMale 和 MarriageFemale 的调用;根据数据分解的特点,Marriage 可以直接分解为并行调用这两个方法,即

```
Marriage(m:Name,f:Name)
    male.MarriageMale(m)‖female.MarriageMale(f);
end.
```

这种并行就是我们所说的内部并行. 如果方法之间存在干扰,需要使用 rely/guarantee 方法.

(2) 因为{MarMal,SingMal,NewMal}和{MarFem,SingFem,NewFem}分别属于不相交的两个分解类 Female和 Male,于是,一个集合中的任一方法与另一个集合中的任一方法可以同时被调用. 这种并行就是所谓的服务并行.

(3) 因为分解类 Female 和 Male 能够被进一步分解为更小的分解类,于是,原规约中的原子动作 MarriageMale 和 MarriageFemale 可能被分解为更深层次的更小的原子动作 CHGSM,CHGMM,CHGSF,CHGMF 的并行执行. 这种原子性就是所谓的虚拟原子性.

(4) 作为上面开发的结果,可以用如图2所示的嵌套面向对象结构表示,其中 World-o,Male-o,Female o,SingMale-o,MarMale-o,SingFem-o,MarFem-o 被认为是不同的对象.

非形式地说,嵌套的面向对象结构具有如下含义:

· 嵌套的面向对象结构是树结构开发过程的面向对象直接表示.

· 为了确保类规约中不变式的有效性,只有原规约中的方法能够被用户调用.这部分方法由 Pub 标识;对象中的方法能且仅能调用相邻分解类中的私有方法.

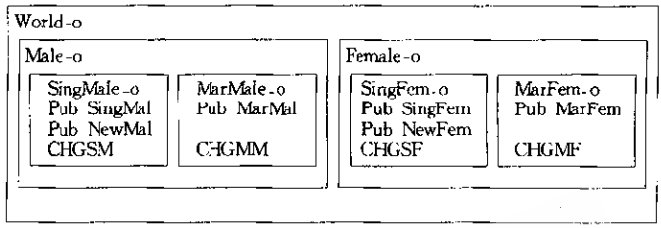


Fig. 2 The nested object-oriented structure of World
图2 World的嵌套的面向对象结构

· 嵌套面向对象结构执行的语义可以非形式地描述如下.一个对象可能处于活动、被锁住、被限制和空闲这4种状态.当一个对象正在执行一个方法时,它处于活动状态.处于活动状态的所有其他方法均不能被调用.当一个对象的内部对象处于活动状态时,它被锁住,被锁住对象的所有方法均不能被调用.当一个对象的外层对象处于活动状态时,它被限制,被限制对象的所有公开方法均不能被用户调用,而且私有方法仅能被相邻外层对象调用.当一个对象不处于上面任何一种状态时,它处于空闲状态.很明显,这个语义支持内部并行、服务并行和虚拟原子性.

· 关于服务并行,一个重要的问题是如何保证不变式成立,因为几个方法在相同的状态空间同时操作.嵌套面向对象结构的语义保证能够服务并行的方法分别属于不同的独立对象,也就是说,它们是无干扰的,因此,我们可以将服务并行看作是顺序执行,其最终的结果是一样的[11].于是,可以用方法顺序执行的方式处理不变式,这就保证了不变式的正确性.

(5) 增加并行度的转换.如果我们仔细考察嵌套面向对象结构的语义,就会发现限制并行的某些因素.当一个外层对象中的方法被调用时,内层对象就会受到限制.在某些情况下,这种限制是不必要的.这个问题可以由文献[7,8]中的转换规则解决.例如,Marriage 可以进行如下转换:

```
Marriage(m:Name,f:Name)                Marriage(m:Name,f:Name)
    MarriageFem(f);                        return;
    MarriageMale(m);          ⟹           MarriageFem(f);
    return;                                MarriageMale(m);
end.                                   end.
```

2 嵌套结构的实现与转换

利用数据分解,我们可以得到一个嵌套的面向对象结构.这个嵌套结构是概念级和设计级的,在开发中通常可以根据不同的需求和条件对其进行不同的实现.

2.1 嵌套面向对象结构的直接实现

由于嵌套结构是以对象为单元的,而且这个结构本身具有一定的语义,因此,直接实现通常具有下面两重含义.一方面,直接实现程序的组成单元应该是对象而不是类.现有语言中基于原型(或对象)的面向对象程序语言可以满足这一要求.另一方面,直接实现是指对象执行的语义应体现在对象结构本身,不需要额外的同步机制进行限制.具有上述性质的语言可以很好地匹配嵌套面向对象结构.我们用伪代码将其框架描述如下.

```
object World
    objects:
        object Male
            objects:
                object SingMale
                    instance variables:...
                    methods:...
```

```
                    end SingMale
                    object MarMale
                    ...
                    end MarMale
                method:...
            end Male
            object Female
                ...
            end Female
        methods:
            Pub Marriage(...)...
end World
```

接口可有两种考虑. 一种是方法的集合,即嵌套对象中所有标识为 Pub 的方法. 另一种是对象和方法集组成的二元组,其中对象是这个嵌套对象中所有标识为 Pub 的对象,而方法集是在此对象中定义的所有标识为 Pub 的方法,这样,嵌套对象的用户就可以通过内部对象直接调用相应的方法. 至于选择哪种接口方式取决于所用的实现语言和开发环境. 由于原型式面向对象语言在实际开发中很少使用,因此提出了下面的间接实现.

2.2 间接实现嵌套结构

间接实现可能出于下面两种考虑:一种是对象间嵌套通过类的嵌套进行描述,但嵌套结构本身并不反映上面嵌套面向对象结构的执行语义;另一种是对象的嵌套结构由普通的对象结构实现,嵌套结构本身的语义由其他机制间接实现.

(1) 类嵌套机制. 这种语言设计的思想是:首先,对象间的嵌套表现为类间的嵌套,对象作为类的实例;其次,作为这个嵌套类的实例对象,其内部对象对用户提供的服务均在最外层对象存在与之对应的接口,从而在内层对象与外层对象之间建立了客户关系;最后,嵌套类的结构本身不具有前述嵌套结构的语义,它是间接地通过同步机制来实现的,其中内部类的提出主要是考虑内部对象对外部对象的私有性和对外不可见性. 这种模型将在另文阐述.

(2) 复原. 数据分解为我们提供了一种可并行执行的面向对象嵌套结构. 但是,在对此嵌套结构进行直接实现时需要对现存语言做许多扩充工作. 因此可以将数据分解作为一个引入并行的过程,一种思想的记录,而将此记录以某种方式反映在非嵌套结构的面向对象语言中. 将其称为复原是因为多模型恢复到原来的单模型. 值得注意的是,保留数据分解所得的私有方法,并且为了保持原多模型的语义,需要具备其他辅助机制(如同步机制)和多个方法可同时调用的可能性(服务并行).

上面提到的各种实现方法各有优缺点,在实际的开发中,我们应该根据应用领域的不同支撑工具的不同、以及开发的时间限制和系统的性质作合理的选择.

2.3 嵌套结构的转换与实现

在一个嵌套结构中,内部对象是作为外部对象的一部分存在的. 这样,嵌套结构就有以下两点局限性:首先,它限制了对内部对象的复用;其次,这种嵌套的结构从概念上只适用于存在共享量的并行系统中,不适合于分布式环境. 基于上面的考虑,在某些环境下,将嵌套的结构转换为平行结构是必要的. 这种转换如图3所示.

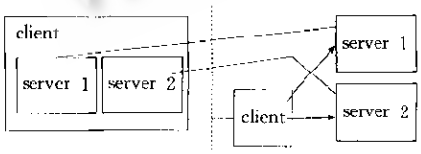


Fig. 3 Transformation from nested structure to flattened structure

图3 嵌套结构向平行结构的转换

这种转换实际上就是将对象间的包含关系转换为对象间的指引关系. 这正是 VDM++中退火步所要达到的目标,为了避免引入 VDM--中的附加推理部分,这种转换应发生在软件开发的实现阶段.

在将嵌套面向对象结构转换为平行面向对象结构时,为了保证嵌套结构的性质,这个转换必须满足以下3个条件. (1) 在平行结构中,服务对象对客户对象是私有的,即客户对象拥有服务对象的私有指引(这是嵌套结构隐含的). 私有指引可以通

过在语法级对参数传递进行限制实现.(2) 用户可见的是客户对象,不能对原内部对象直接进行存取,这是为了与原规约保持一致.因而要求必须在客户对象中提供所有用户可见的接口.(3) 嵌套结构隐含的并行性应该在平行结构中使用同步机制来实现.

嵌套结构记录了每一步开发,有些可能是不必要的、繁琐的.因此,在将嵌套结构转换成平行结构时,可以不必一一对应.以 World 为例,可以有如图4所示的3个可供选择的平行结构.这种平行结构很容易用现存的面向对象语言结构进行实现.值得注意的是,应根据所选语言的机制实现原嵌套结构的语义.

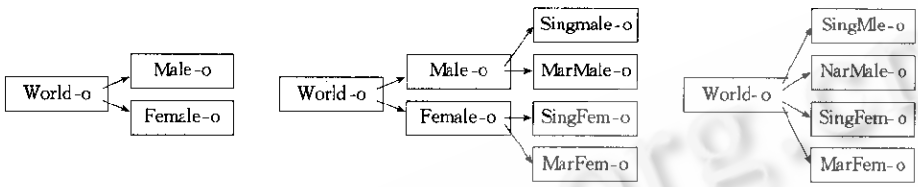
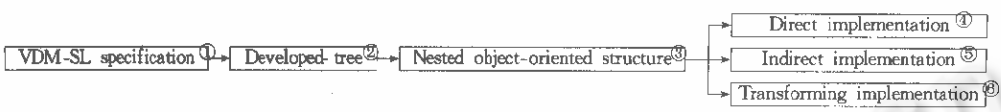


Fig. 4 Three kinds of transformations for the nested structure World
图4 嵌套对象World的3种转换方案

3 结 论

本文在 DD-VDM 的基础上提出了一种形式化的并发面向对象程序的开发方法.开发过程如图5所示.它是 VDM 方法的自然扩充,易于理解,便于使用.值得指出的是,使用 DD-VDM 的数据分解得到的并行是正交并行,结构是树状的,但由于 DD-VDM 还包含 VDM 中的设计步,使用它们能够开发更复杂的并行和网状的结构.数据分解简化了一般设计步使用的 rely/guarantee 方法所带来的复杂规则及其证明过程,这为软件开发者提供了可选择性.我们进一步的工作主要包括基于 VDM-SL 的并行面向对象规约语言设计和将上述开发方法有机地融合进新的开发方法之中.目前这方面的研究已取得阶段性成果,相关内容将另文介绍.



①VDM-SL规约,②开发树,③嵌套结构,④直接实现,⑤间接实现,⑥转换实现.

Fig. 5 The concurrent object-oriented software process using DD-VDM
图5 基于DD-VDM的并发面向对象软件的开发过程

参考文献

1 Yang Da-jun, Zhang Ming, Lü Jian. The study of concurrent object-oriented programming languages. Computer Research and Development, 1998,35(9):769~775
(杨大军,张鸣,吕建.并发面向对象程序设计语言研究.计算机研究与发展,1998,36(9):769~775)
2 Jones C B. Systematic Software Development Using VDM. 2nd ed., Englewood Cliffs: Prentice-Hall, Inc., 1990
3 Jones C B, Shaw R C F. Case Studies in Systematic Software Development. Englewood Cliffs: Prentice-Hall, Inc., 1990
4 Jones C B. Development methods for computer programs including a notion of interface [Ph. D. Thesis]. Oxford University, 1981
5 Jones C B. Specification and design of (parallel) programs. In: Proceedings of IFIP'83. North-Holland, 1983. 321~332
6 Stolen K. Development of parallel programs on shared data-structures [Ph. D. Thesis]. Manchester University, 1990
7 Jones C B. Constraining interference in an object-oriented design method. In: Gaudel M C, Jouannaud J P eds. TAPSOFT'93: Theory and Practice of Software Development. Lecture Notes in Computer Science, London: Springer-Verlag, 1993. 136~150
8 Jones C B. A pi-calculus semantics for an object-based design notation. In: Best E ed. CONCUR'93: the 4th International Conference on Concurrent Theory. Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1993. 158~172
9 Lü Jian, Zhang Jian-ying. A formal software development method DD-VDM. Journal of Software, 1996,7(supplement):

385～393

(吕建,张建莹.形式化软件开发方法DD-VDM.软件学报,1996,7(增刊):385～393)

10 Lü Jian. Introducing data decomposition into VDM for tractable development of programs. ACM SIGPLAN Notices, 1995,30(9):41～50

11 Goldsack S, Lano K, Dürr E H. Annealing and data decomposition in VDM++. ACM SIGPLAN Notices, 1996,31(4): 32～38

12 Goldsack S J, Kent S J H. Formal Methods and Object Technology. London: Springer-Verlag, 1996. 86～112

13 Dawes J. The VDM-SL Reference Guide. London: Pitman, 1991

14 Apt K R, Olderog E R. Verification of Sequential and Concurrent Programs. 2nd ed., New York: Springer-Verlag, 1997

A Data-Decomposition-Based Development Method of Concurrent Object-Oriented Programs

YANG Da-jun　LÜ Jian

(*State Key Laboratory for Novel Software Technology　Nanjing University　Nanjing*　210093)
(*Institute of Computer Software　Nanjing University　Nanjing*　210093)

Abstract　In this paper, a method is proposed to develop a concurrent program from a VDM-SL (Vienna development method-specification language) specification. On the basis of DD VDM (data decomposition-Vienna development method), service parallelism, internal parallelism and virtual atomicity can be observed in the development process. Then a nested object-oriented structure is presented to specify these parallelisms. This nested structure can be implemented with several language structures for two different kinds of applications including share-variables parallel system and distributed parallel system.

Key words　Formal method, data decomposition, nested object-oriented structure, concurrent object-oriented programming language.