基于分层面向对象 Petri 网的软件建模方法研究

冯晓宁[1],王 朔[2],王 卓[1]
FENG Xiao-ning[1],WANG Shuo[2],WANG Zhuo[1]

1.哈尔滨工程大学 计算机科学与技术学院,哈尔滨 150001
2.浙江大学 竺可桢学院,杭州 310058
1.Department of Computer Science & Technology,Harbin Engineering University,Harbin 150001,China
2.College of Chu Kochen Honors,Zhejiang University,Hangzhou 310058,China
E-mail:fengxiaoning@hrbeu.edu.cn

FENG Xiao-ning,WANG Shuo,WANG Zhuo.Research of modeling method based on Hierarchical Object-Oriented Petri Net.Computer Engineering and Applications,2008,44(31):90-93.

Abstract: To enhance and improve the representation ability of the Petri Net,the paper provides a Hierarchical Object-Oriented Petri Net called HOOPN modeling method and formal syntax and semantic of HOOPN in detail.HOOPN supports a wide range features of the concept of object-oriented such as abstraction,encapsulation,inherit and polymorphism.And HOOPN also supports most modeling and analyzing methods at present.At last the paper gives the example of application modeled and analyzed by HOOPN to prove the validity of HOOPN.

Key words: Hierarchical Object-Oriented Petri Net(HOOPN);software modeling;incremental analysis

摘 要:为了增加和扩展 Petri 网的表现能力,提出了一种分层的面向对象 Petri 网 HOOPN,并且对 HOOPN 的正式的语法和语义进行了详细的阐述。HOOPN 建模方法在很大范围上支持面向对象的抽象、封装、继承和多态等典型特征。HOOPN 同样也支持很多的建模和分析机制,最后的应用实例建模与分析证明了 HOOPN 的有效性。

关键词:分层面向对象 Petri 网;软件建模;增量分析
DOI:10.3778/j.issn.1002-8331.2008.31.026 **文章编号**:1002-8331(2008)31-0090-04 **文献标识码**:A **中图分类号**:TP39

1 绪论

尽管以 Petri 网为代表的形式化方法在大型复杂系统建模方面有着清晰准确的优点,但是这种方法在对不同领域的系统进行建模或分析时在表现能力和分析能力上存在着不足。因此许多专家将面向对象概念融入到 Petri 理论中,提出了对象 Petri 网(OPN)[1]、VDM++[2]、Object-Z[3],这些方法虽然在 Petri 网中体现了面向对象的概念,也得到了一定的应用,但是并不能完全的支持面向对象方法的主要概念,没有实现面向对象与 Petri 网的有机融合。提出了一种分层的面向对象 Petri 网——HOOP(Hierarchical Object-Oriented Petri Net),并且对HOOPN 的正式的语法和语义进行了详细的阐述。设计 HOOPN 的目的是帮助面向对象软件系统的建模与分析,也是为了在为复杂软件系统建模、分析和构造原型过程中减少使用 Petri 网和面向对象两个技术之间的沟壑。

2 HOOPN 的定义

HOOPN 在很大范围上支持面向对象的典型特征,比如抽象性、封装性、对象模块化、对象内部的消息传递、继承性和多态性。

2.1 HOOPN 的形式定义

HOOPN 模型是 Petri 网模型的变体,一个 HOOPN 模型的表现形式对应与面向对象中的一个类。HOOPN 由三部分组成,对象识别库所 Object Identification Place,OIP、内部对象网(Internal Object Net,ION)和数据字典(Data Dictionary,DD)。OIP 是一个类的唯一标志;ION 是一个描述类的行为(方法)的网;DD 描述了类的属性。

定义 1 HOOPN 是一个三元组,$HOOPN=(OIP,ION,DD)$,满足下面条件:

(1)OIP 是一个特殊的库所可以定义为一个元组 $OIP=(oip,pid,M_0,status)$,其中:oip 是 HOOPN 唯一的变量名;pid 是区分一个类的多实例的唯一过程标识符,包括了类的返回地址;M_0 是一个针对 OIP 中托肯给出指定值的函数;$status$ 是制定 OIP 状态的标识变量。

(2)ION 是着色 Petri 网(CPN)的变体,用来表现属性值和

基金项目:哈尔滨工程大学校基金(the Foundation of Harbin Engineering University under Grant No.HEUFT05035)。
作者简介:冯晓宁(1976-),男,博士研究生,讲师,主要研究领域为 Petir 网、系统仿真等;王朔,男;王卓(1977-),女,博士,副教授,主要研究领域为数据库与知识库、软件工程、系统仿真等。
收稿日期:2007-12-05 **修回日期**:2008-04-25

方法中行为的变化。它在定义 2 中给出正式的定义。

(3)DD 是变量、托肯类型、函数的声明部分。

HOOPN 的一般结构如图 1 所示。

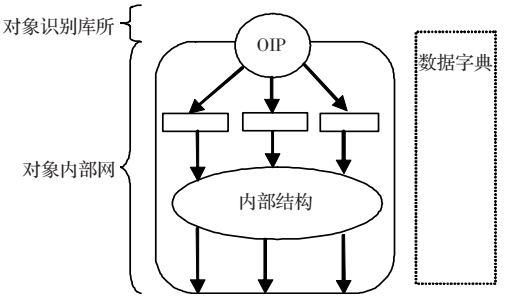

图 1　HOOPN 的一般结构

定义 2 HOOPN 的内部对象结构,可以定义为:$ION=(P,T,A,K,N,G,E,F,M_0)$,其中:(1)$P$ 和 T 分别表示有限库所集和有限变迁集;(2)A 是有限弧集,满足:$P\cap T=P\cap A=T\cap A=\varnothing$;(3)$K$ 是从 P 到声明在数据字典(DD)中的托肯类型集的映射函数;(4)N、G 和 E 分别表示结点函数,保护函数,弧表达式函数,与文献[4]中定义的一样;(5)F 是一种特殊的弧,可以从任何一个变迁到达 OIP,并且可以标志为 ION 的结构框架;(6)M_0 是初始化任何库所初始状态的函数。

HOOPN 中托肯类型在数据字典(DD)中描述,被分成两类:一类是原始类型;一类是抽象类型。原始类型的托肯和着色 Petri 网(CPN)中的定义相同,抽象类型的托肯是原始类型的托肯的复合。托肯的抽象类型用来配合抽象库所来表示抽象的状态,抽象类型声明为"记录"类型。HOOPN 模型中用抽象信息来表示的托肯应当表现为抽象类型托肯,因为托肯类型所表现的详细信息不足以表现模型抽象行为的状态。当设计者在抽象层声明托肯类型是可以接受的。然而,这样的表示对抽象库所来说就不是简练的并可能导致在进一步精化中发生变化。抽象托肯类型的声明要有前缀"complex"。这种类型在模型细化分解中被分解为多种子类型(原始类型)。抽象托肯的分解在第 3 章解释。

3　HOOPN 中面向对象的表现

因为支持面向对象概念,HOOPN 比 OPN 形式可以提供更多的表现能力。为了能够为面向对象系统需求,HOOPN 支持类/对象的表现形式、信息隐藏、分层抽象和继承。

3.1　信息隐藏

HOOPN 模型在面向对象形式表现为一个类,这个类包括对象识别库所(OIP)、数据字典 DD(对应于数据)、内部对象网 ION(对应行为)。OIP 在 HOOPN 中通过对外部唯一的名字表现为公共接口,同时 OIP 也是开始和终止模型中的内部行为的一个库所。DD 是查阅变量、函数和托肯类型定义的字典,用来描述 ION 内部行为。这些 HOOPN 结构上的组件可以完全支持信息隐藏概念因为从外部看来 HOOPN 模型就像一个黑盒子看不见内部行为。

3.2　抽象

在 HOOPN 模型中,抽象的概念表现在库所、变迁和托肯中。当一个抽象库所(ABP)被细化分解时,它的约束条件应该被保留在 ABP 的托肯类型以及输入弧和输出弧表达式中。抽象变迁(ABT)被细化分解时,托肯类型和输入输出库所以及防卫表达式就是约束条件。HOOPN 中可见态射的定义可以给出如下:

定义 3 可见态射(observable morphism),$\forall C\in\{ABP,ABT,ATN\}, C\Rightarrow RM$ 应该满足下面的条件(其中⇒表示"细化分解为"):

(1)ABP 和 ABT 的托肯类型

$\forall p\in ABP, TYPE(p)\subseteq TYPE(TT_{RM})$, and $\forall t\in ABT, \{TYPE(t\cdot)\cup TYPE(t\cdot)\}\subseteq TYPE(TT_{RM})$

其中 TT_{RM} 定义在细化分解模型(RM)中数据字典(DD)里的所有托肯类型的集合。$TYPE(p)$ 表示每一个库所 p 到托肯类型集的映射。

(2)对 ABP 所有的输入输出弧表达式 a_i、a_o,对所有的 ABT 的输入弧 a_i 和 ABT·的输出弧 a_o, $\bigcup_{a_i,a_o} Var(\exp r)\subseteq Var(OIP_{RM})$。

(3)对 ABT 的所有保护表达式,$t\in ABT, G(t)\equiv\cup_{\forall t_i\in(OIP_{RM})\cdot} G(t_i)$,其中 $G(t)$ 是 t 的所有保护函数集。

抽象和细化分解能够使得系统建摩通过递进和分层的方式进行,这是一个行之有效的用来降低系统分析和设计过程的方法。

3.3　继承实现

继承是类之间基于一种分层关系的属性和操作的共享。一个类可以被定义的很广泛(粗框),然后继续的细化分解成合适的子类。每一个子类或者继承或者合并它的父类的所有属性,然后加入自己所特有的属性。而父类中的属性不需要在每个子类中被再次声明[4]。

当被继承的类实例化时,子类的数据字典(DD)包括了父类数据字典的所有内容,并且包括了可以被 $E(OIP,t)\langle b\rangle$ 使能的变迁集。在继承机制中一个要考虑的问题就是通过重载来修改父类中的属性。子类中的重载属性改变替换了父类中的属性。重载父类的属性或方法的原因是为了指名依赖于子类的行为,并同时缩紧了方法的规范。继承时应当决定哪一个方法是需要被重载的,哪一个是重载的,因为重载用同一个名字重新定义个一个方法但是不允许重载签名和特征形式。这是动态绑定的规则所约束的,规则搜索整个继承分层结构来绑定一个特定操作的执行方法在继承的对象中。

定义 4 从父类继承的类满足下面的条件:让 I、S、N 分别表示为继承的子类、父类、新定义的重载的方法。(1)$TT_I=TT_S\cup TT_N$;(2)a set of enabled transitions, $ET_I=\{t_i|\forall t_N, E(OIP_I,t)\langle b\rangle\}\cup\{t_j|\forall t_S, E(OIP_S,t)\langle b\rangle\}$;(3)for $t_i\in T_N\wedge t_j\in T_S$, $\exists t_i,t_j\in ET_I\mapsto((t_i,b)\in Y\wedge(t_j,b)\notin Y)$。

3.4　参变量的多态性

支持多态意味着可以实现根据一个输入托肯的值来动态绑定一个托肯类型。从操作的角度来说,动态绑定可能被认为是一个调度机制,动作起来就像一个 case 语句来动态的选择适应托肯的值的行为。在 HOOPN 方法中,多态可能发生在当

ABP 或者 ABT 的变量 refine_state 的值为真并且 COT 调用一个外部类的 OIP 的情况。在抽象组件进程中的多态的动态绑定为两个步骤:首先 OIP 中的联合托肯类型被分解为单独的详细的托肯类型;然后 OIP 中托肯的值根据联合类型的值的不同而使用特殊的值来定义。当 C 的托肯值是 normal 并且正常状态表示一个顾客的精神状态时,C 的托肯类型被分解为:

```
TT C=record with {
    IntQ=[0…110]|[111…130]|[131…150];
    EmoQ=[0…3]|[4…7]|[8…10];
    Spek=trouble|common|fluent;}
```

然后,每一种类型的托肯值被分配到{(Int$Q$,[111…130]),(Emo$Q$,[4…7]),(Spek,commpn)}在“ap1”细化分解模型中。然而,当身体状况的值为正常(normal)时托肯类型 C 被分解为:

```
TT C=record with {
    Height=short|average|tall;
    Weight=light|middle|heavy;
    Heart=low|center|high;}
```

并且,托肯的值在细化分解模型中被分配到{(Height,average),(Weight,middle),(Heart,centre)}。如此,一个托肯类型根据参数变量的值的不同(这里是精神上的和身体上的)可以被不同的子类型绑定。

另一个多态的类型关于由继承建立的类的属性和方法同样也要被考虑,这种类型的多态用范围规则和动态绑定在 3.3 节与继承一起被解释了。

3.5 对象间的消息传递

对象之间的消息传递发身在当一个对象的方法调用另一个对象的方法为了交互的时候,当一次方法调用时,调用的对象从 COT 传递了一个托肯到被调用的对象的 OIP(HOOPN),根据 OIP 到来的托肯值,决定了在 OIP·(后缀变迁集)中的那个变迁被使能和点火。当特定的方法的执行结束时,执行的结果通过它本身的 OIP 被传送到 COT,两点要被考虑:

(1)同步或者异步的方法调用:在同步调用中 COT 必须接收到调用方法的执行结果,因此,在 COT·中的库所不能被标记直到结果返回,同时 target 的值是“yes”。在异步调用中,即使托肯已经从 COT 传送到了 OIP,在 COT·集中的库所应当有标记。在用异步类型定义的 COT 中,托肯被同时传到了 OIP 和 COT·中。被调用方法的执行结果被保存在了全局变量中以便以后使用。

(2)嵌套方法调用:在调用方法的执行过程中,有可能还要调用其它对象或者自身的方法。提供了变量 pid 来表示嵌套的方法调用。当 COT 调用一个方法,设置了变量 pid 为传送托肯的值和指示了被方法调用的类的实例的进程指示器。变量 pid 的大小随着后继调用的出现而成比例的增长。

3.6 对象实例化

当一个类初始化系统行为的时候一个实例就被创建了(在过程语言中对应 main()主函数),或者当一个类被其它类(或对象)调用的时候。当建立了一个实例时,这个实例有一个唯一的标识符与变量 pid 的值不同。实例 pid 的值对应着 main()主函数本身,而被调用实例 pid 的值是调用类的返回指针。

4 HOOPN 的建模过程实例

4.1 一个汽车租赁系统(CRS)系统的简单需求

一个汽车租赁系统(CRS)最初的需求如下:

(1)系统支持基本的租赁服务,比如租车和还车;

(2)一个客户要租用一辆汽车,必须要填写保险政策并且租赁代理要检查客户的信誉度;

(3)客户的信息包括姓名(标识符),驾照号码,信誉卡号等等;汽车的信息包括一个标识符,汽车的类型,汽车的状况,保险费用等等;

(4)当租赁成功后发生了意外事故时,客户与租赁代理联系,而不是保险代理;

(5)当发生了意外事故后归还汽车时,租赁代理把车送到保险代理处,当车再从保险代理归还时,车应当是完全修复了的;

(6)即使车的租用期未满,如果顾客发生了意外事故也应当归还车,然后缴纳一定的费用;

(7)其它的需求按照备注解释说明。

4.2 系统建模

(1)高层建模:顾客“CUST”的 HOOPN 模型,建模所采用的过程在前面已经用分解的需求在图 2 中详细说明了。“CUST”模型有一个抽象库所 OpeCar 和两个通讯变迁 $REAG_1$ 和 $REAG_2$。这些组件显示了一个顾客与租车代理进行交互来租车或者还车,在这些组件中,库所 OpeCar 包括了租车和还车的行为,同时还车动作包括了正常还车和事故还车。

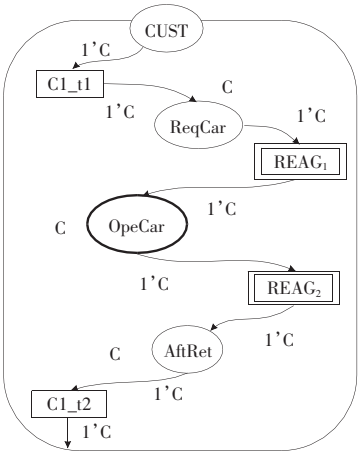

图 2 CUST 的 HOOPN 模型

(2)数据字典(DD):DD 声明了托肯类型、变量和功能函数,同样可以包含一些抽象库所 ABP、抽象变迁 ABT 和 COT 的执行结果表达式。DD 是按照 CPN ML(着色 Petri 网建模语言)的语法和语义的变量。这里有一些声明模型组件的类型,是一个类型集={primitive,complicated,user_definded},原始类型(primitive types)包括布尔型,字符串型,字符型,整型,实数型和数组;复杂类型(complicated types)是记录和联合类型,记录类型是几个原始类型声明合成的类型并且可以用记录类型定义嵌套的托肯结构。联合类型可以由那些没定义的类型组成。所以那些没定义的类型可以通过参数的动态绑定被细化分解成一些其他的类型。HOOPN 模型“CUST”的数据字典(DD)如下:

```
Level 1:Data Dictionary of  "CUST"
Var +CX=Boolean; /*rent or not*/
TT C=complex;
COT(REAG)={
        Fun(CX= =F):CX=T∧M(OpeCar,C);
        Fun(CX= =T):CX=F∧M(AftCar,C);
        };
```

在这个数据字典中,CUST 中的托肯类型 C 用联合类型定义,这个托肯类型将被分解成细化分解的托肯类型在下一层模型的数据字典中。全局变量"CX"表示了顾客是否租用了车,动作术语 COT(REAG)对应了 COT 的执行后条件,当 CX 的值为假(False)时改变 CX 的值为真(True)然后标记库所 OpeCar。如果 HOOPN 模型 REAG 画好了后这个动作将被丢弃并且不会影响 COT 的行为的。

(3)细节说明与建模。在第 1 级模型中,REAG 模型同样可以画出来表达租用代理的行为,REAG 的主要动作就是租出一辆车和收回一辆车。REAG 的 HOOPN 模型如图 3 所示。

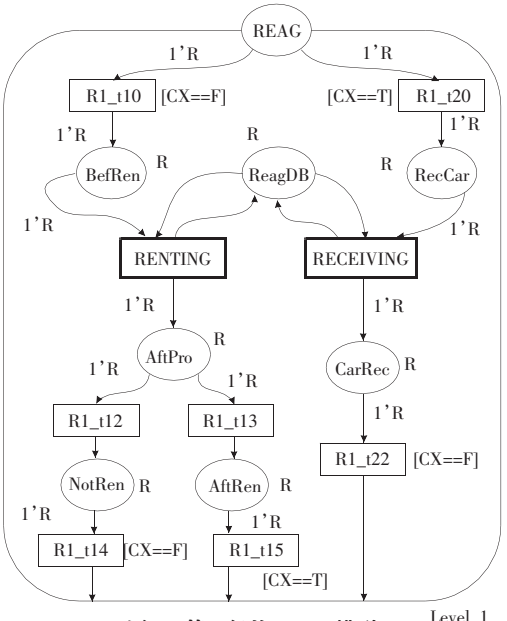

图 3　第 1 级的 REAG 模型

REAG 的数据字典(DD)和 CUST 的数据字典很相近,全局变量 CX 同样在这个模型中使用, 在图 4 的 REAG 模型中,动作起始于变迁 R1_t10 和 R1_t20,R1_t10 表示向顾客租出一辆车,R1_t20 表示从顾客处回收一辆归还的车,回收汽车行为同样可以分解为两个主要的行为:正常回收和事故回收,这个细化分解省略了。这里库所"NotRen"表示顾客由于某种原因无法租用一辆车。

当使用 HOOPN 为一个系统建模时,不需要考虑所有的复杂的和扩展的系统信息,因为建模的步骤是迭代递增的。模型的可达性分析也同样能递增的获得,这种方法可以帮助解决大型复杂的软件系统的建模问题并且易于修改模型。

5　结束语

面向对象概念的应用变得越来越成熟,并且被越来越多的人应用。一种需求就是对大型复杂软件系统的建模与分析中的应用,这种应用要求设计模型必须是非常明确的、精确的和可验证的。为了满足这些要求,很多好的建议就是将面向对象概念和一些形式化方法的有机结合。尽管很多具有对象概念的高级 Petri 网在特殊的领域里被提出, 但是这些方法并不完全的支持在用面向对象概念为系统建模时所需的足够的特性。

为了解决这个问题,提出了一个高级面向对象 Petri 网——分层的面向对象 Petri 网 HOOPN, 分层面向对象 Petri 网对面向对象概念有着很好的支持,HOOPN 建模方法支持接口的消息隐藏;对库所、变迁和托肯的抽象;抽象托肯动态绑定中的多态性;类中属性共享的继承性;消息传递的交互过程等一系列面向对象对象的深层概念,它使用明确的语义支持面向对象概念的大多数特性。此外,描述了这种建模分析方法为系统建模的过程,并且可以对复杂系统进行反复迭代的建模。这通过封装、对象模块化、抽象信息建模、分解及精炼方法和增量可大型分析来实现。

参考文献:

[1] Bastide R.Approaches in unifying Petri nets and the object-oriented approach[C]//Proceeding of the International Workshop on Object-Oriented Programming and Models of Concurrency,Turin,Italy,2005:172-189.

[2] Brauer W,Gold R,Vogler W.A survey of behavior and equivalence preserving refinement of Petri nets[C]//LNCS 483:APN'90,1991:1-46.

[3] Cardoso J,Valette R,Dubios D.Petri nets with uncertain markings[C]//LNCS483:APN 90,1991:64-78.

[4] Eliens A.Principles of object-oriented software development[M].UK,Wokhingham:Addison-Wesley,1995.

[5] Fehling R.A concept of hierarchical Petri nets with building blocks[C]//LNCS 674:APN'03,2003:148-168.

[6] Harel D,Gery E.Executable object modeling with statechart[C]//Proceedings of the 18th International Conference on Software Engineering,Germany,1996:246-257.