面向对象的 Petri 网方法及其在软件工程中的应用

范玉顺　张　军　清华大学自动化系　北京(100084)

摘　要　本文介绍了面向对象 Petri 网(OO PN)方法,给出了 OO PN 的基本定义和图形表示,讨论了 OO PN 的特性和子网细化方法。结合软件设计开发周期,给出了 OO PN 在软件工程中的应用方法及步骤。给出了利用 OO PN 方法进行柔性制造系统(FM S)控制软件设计开发的应用实例,并给出了部分对象类和应用程序。利用 OO PN 方法可以对软件系统设计开发过程进行有效的控制,保证软件系统的可靠性和规范化水平。

关键词　面向对象, Petri 网, 软件工程, 柔性制造系统

OBJECT-ORIENTED PETRI NET METHOD AND ITS APPLICATION IN SOFTWARE ENGINEERING

Fan Yushun　Zhang Jun

Department of Automation, Qinghua University, Beijing　100084

Abstract　In this paper Object- Oriented Petri Net (OOPN) method is introduced. The basic definitions and graphical denotations are presented. The characteristics and subnet refinement method of the OOPN are given. In consideration of software design and development life cycle, the method and application steps in software engineering using OOPN are formulated. An example of the design and development of control software for flexible manufacturing system using OOPN is given. Some of the codes for the object class and application programs are also presented. The using OOPN in software engineering will improve the efficiency in controlling the software system development process and ensure the reliability and standardization of software system.

Keywords　Object- oriented, Petri Net, Software engineering, Flexible manufacturing system

1　引言

Petri 网方法已经被广泛应用于 CM 系统的设计、分析和实施中。它提供了一种关于复杂制造系统的有效分析和实施方法。近年来, Petri 网方法也被用于软件的开发、验证中。Meta 软件公司开发了一系列软件。这些软件可集成 SAADT(IDEF0)模型和递阶有色 Petri 网,从而产生一个集成的软件开发方法学[1]。文献[2]研究了将 Petri 网方法应用于实时控制软件的开发中,指出这种 Petri 网模型可由数据驱动结构实现并可基于任意的标准高级语言。文中给出了一套基于 Petri 网方法的软件开发流程。文献[3]主要研究了 Petri 网方法在软件工程、系统设计中的应用,给出了基于谓词/变迁网表示动态系统特性的方法。

面向对象技术目前被人们看作是解决复杂问题的最有希望的方法之一,综合考虑了有色 Petri 网的"颜色"这一抽象机制与面向对象中类、封装、继承等概念相融合,出现了面向对象 Petri 网(OO PN)方法[4-6]。从这些研究者对 OO PN 的研究,可以得出这样的基本结论:

面向对象模型和 Petri 网模型具有一致性;

将 Petri 网模型引入面向对象的开发方法,可以克服面向对象方法缺少模型分析、验证手段不足的缺点;

将面向对象方法引入 Petri 网理论中,可以克服 Petri 网在描述复杂系统上的不足。

2　面向对象 Petri 网方法介绍

2 1　面向对象 Petri 网基本定义

关于 OO PN 有许多不同的定义方式,这些不同的定义方式从不同的角度来理解对象及其它们和 Petri 网的结合,因此所产生的 OO PN 也有很大差别。本文给出的仅是 OO PN 的一种定义。OO PN 可通过扩展有色 Petri 网而得到,由一组库所、变迁和弧组成:每个库所代表一类对象,每类对象的不同实例可由不同的托肯颜色来区分;对象之间通过消息库所通过变迁连接。每个变迁代表一个动作;变迁上

收稿日期: 1997-12-22　范玉顺。主要从事 863/CM S 专题及烟草工业 CM S 研究与开发。

可以定义相应的布尔表达式, 表示什么条件下该变迁可以发生, 每条弧上有相应的弧表达式, 用以决定要移动的托肯的实际颜色。下面给出 OOPN 的定义。

【定义 1】一个面向对象 Petri 网的对象子网是一个扩展的有色 Petri 网, 它是一个八元组 $OOPN_s = [P_s, T_s, F_s, MF_s, C_s, G_s, E_s, M_{os}]$, 其中:

(1) $P_s = P_n \cup P_{mi} \cup P_{mo}$, P_n 是对象子网内部的库所集(它可以是普通的 CPN 的库所, 也可以是对象子网库所), P_{mi} 是对象子网的输入消息库所集, P_{mo} 是对象子网的输出消息库所集;

(2) T_s 是对象子网内部的变迁集;

(3) $P_s \cup T_s \neq \phi, P_s \cap T_s = \phi$

(4) F_s 是 P_s 与 T_s 之间的有向弧集(流关系), $F_s = P_s \times T_s \cup T_s \times P_s$;

(5) MF_s 是由对象的类方法组成的有限集合, 每个类方法集合必须有限且非空;

(6) C_s 是由不同颜色组成的有限集合, 每个颜色集必须是有限且非空;

(7) G_s 是决定某变迁触发条件的布尔表达式集合;

(8) E_s 是定义在弧集合 F_s 上的表达式函数;

(9) M_{os} 是定义在 P_s 上的初始标识。

【定义 2】一个系统包含 K 个对象子网 $O_1, \ldots, O_k$ 的面向对象 Petri 网模型为, $OOPN = (\bigcup_{i=1}^{k} O_i) \cup T_m$, , 其中 T_m 为连接对象子网输入消息库所和输出消息库所的变迁集合。

遵循标准 Petri 网图示法的惯例, OOPN 在表示为网时, 有以下规则:

1) 一类对象用一个矩形框库所表示如图 1, 矩形框边界上的库所为消息库所。

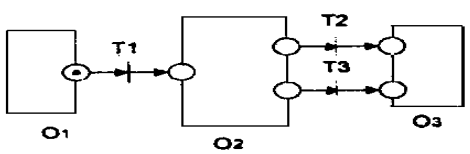

图 1 OOPN 的图形化表示

2) 这一类对象可由对象子网来细化如图 2, 直到没有表示对象的库所为止。对象子网的颜色集代表的是该对象的所有实例。

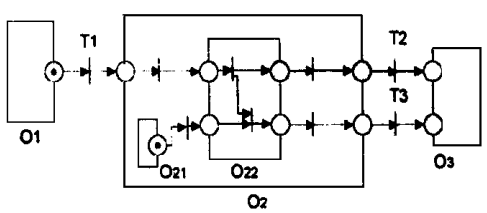

图 2 对象类 O2 的细化

3) 如果两个对象之间有交互作用, 即一方要给另一方发送消息, 则这两个对象之间有一个变迁, 与该变迁相关的托肯代表消息。

OOPN 的运行和有色 Petri 网的运行方式相似, OOPN 运行时, 网中的变迁被触发, 导致托肯的转移。对象之间的消息传递通过连接消息库所的变迁的触发完成。

2.2 OOPN 的面向对象特性与子网结构

面向对象方法的一个重要特性是对象封装, 对象的这一封装机制的目的在于将对象的内部特性和外部操作分开, 使用者不必知道对象内部实现的细节, 只需用设计者提供的操作方式去完成所需要的功能。

在 OOPN 中, 每个对象具有内部结构和外部联系, 外部联系用于对象间的消息传递, 而对象的内部数据和方法都由内部结构来表示, 即对象子网。对象的封装可以用 O_2 细化后的结构反映。如图 2 所示, O_2 有一个非常清楚的边界, 图示法中用矩形框表示, 对象的所有私有数据、方法都被限制在这个边界内, 对外不可见。O_2 可通过与其它对象的接口, T_1, T_2, T_3 进行消息的发送、传递和响应, 从而实现了对象的封装。

OOPN 对象子网的内部结构包含变迁, 库所和有向弧, 但其内部的库所可以是 Petri 网通常意义上的库所, 也可以是对象类。其内部的变迁可以是 Petri 网通常意义上的变迁, 也可以是对象之间消息的发送与接收。对象子网的结构如图 3 所示。

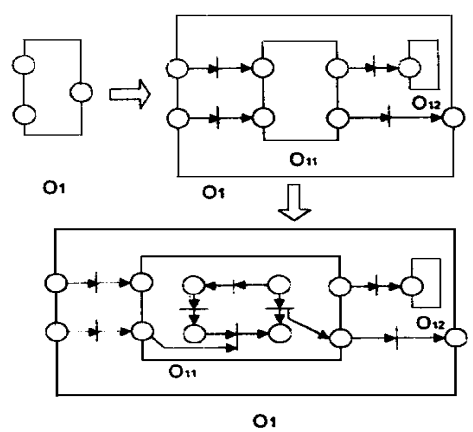

图 3 对象子网的结构

OOPN 的对象继承问题比较复杂, 在此不进行深入讨论。由 OOPN 的图示可以看出, 随着对对象子网的逐步细化, 对象类间的聚合关系, 即包含关系, 将会很清晰地表现出来。如图 3 中, 对象类 O_1 包含对象类 O_{11} 和 O_{12}。但是, 这种子网内部的对象类和其原对象类只是一种包含关系, 而不是继承关系。

3　OO PN 在软件工程中的应用

用 OO PN 对所设计的软件系统进行建模及系统分析应遵循面向对象方法处理复杂问题时的基本原则:

1) 对所研究的系统在本研究层次上抽象为一些"对象",建立对象子网模型,用消息库所连接不同对象;

2) 对外部提供的是对象子网的消息库,而隐藏其内部的所有库所及变迁;

3) 同一对象子网的所有颜色,即对象的实例,均继承改子网的结构特征,靠颜色标志及其携带的信息区别不同的对象;

4) 在处理复杂问题时,应特别注意研究问题域的对象的分类。对象分法的统一有利于软件系统的分析和综合。

使用 OO PN 实施软件开发的基本步骤如下:

1) 概念建立。确定要开发的软件的总目标,给出它的功能、性能、可靠性以及接口等方面的设想;确定是否使用 OO 的方法,并讨论可行性,进行评估。

2) 确定现实系统中的实体对象,以及它们与周围环境的关系。从现实系统中明确类、对象(他们的角色、职责、协作关系),进行系统建模。通过场景说明描述系统的主要功能点(不必是彻底完备的,但不能遗漏重要功能点)。

3) 在第二步的基础上抽象出类,明确各类的对象间消息的传递关系。

4) 建立 OO PN 模型。建立消息库所,根据对象消息关系画出 OO PN 网图;明确各类对象的内部数据和方法,生成每类对象的对象子网,根据所研究的层次进行细化;将各类对象的实例用不同的颜色集表示,确定初始托肯。

5) 模型验证。验证各对象间消息发送是否正确;验证对象子网是否封装完好,网运行过程中有无死锁等。

6) 软件编程实现,将 OO PN 模型转化为高级程序语言,如 C+ + 。

7) 软件测试。测试软件是否能完成所要求的功能;测试软件的可靠性。

8) 运行维护。管理系统,局部修改;更新产品版本。

根据以上基本步骤可以完成一个软件项目的整个生命周期的工作。

4　应用示例

下面举一个 FM S 控制软件开发的例子。考虑如图 4 所示的 FM S,其中有两种加工设备,车床(2 台)和磨床(3 台),每个工件需在某一台车床和某一台磨床上进行加工。机器人 1 将工件由传送带 1 装载到车床 1 或车床 2,加工完毕后,将其放到传送带 2。机器人 2 将工件由传送带 2 装载到磨床 1、磨床 2 或磨床 3 上,加工完毕后,将工件放至仓库。车床、磨床或机器人一次只能对一个工件进行操作,假定传送带和仓库容量无限。

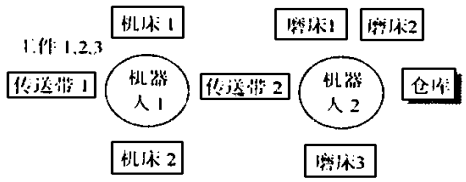

图 4　FM S 示意图

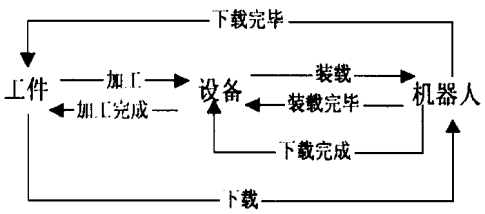

图 5　加工顺序(对象间消息顺序)

该系统中的实体对象有工件,传送带,车床,机器人,磨床。假定传送带在实际系统中一旦开启便不需控制,因此不将它们规为对象类,这样可以得到系统的对象类为: 工件类,设备类,机器人类。其中工件类有 3 个实例(p1, p2, p3),设备类有 5 个实例(车床 c1, c2,磨床 m 1, m 2, m 3),机器人类有 2 个实例(r1, r2)。各工件的加工顺序为(各对象间的消息传递方式)如图 5 所示。图 6 给出了该系统的 OO PN 图(模型)。

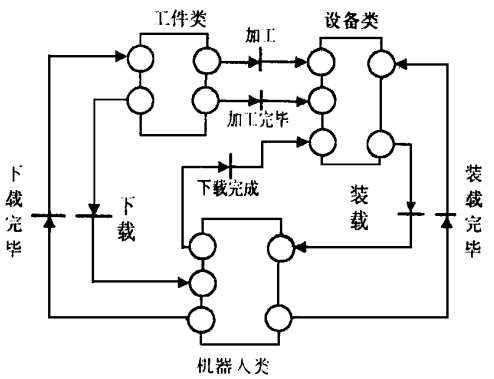

图 6　FM S 的 OO PN 图

下面进行的工作是对象子网的生成和细化工作,细化的详细程度可以根据研究问题层次的需要来确定。限于篇幅,这里仅给出设备类的对象子网。

设备类对象的具体描述为:

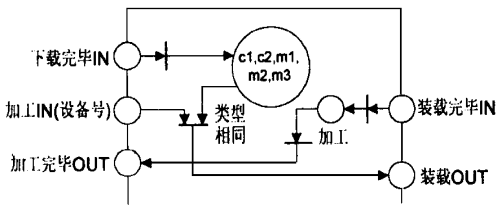

图 7 设备类的对象子网

实例: c1, c2, m1, m2, m3;

发送消息: 装载, 加工完毕;

接收消息: 加工, 装载完毕, 加工完成;

消息库所: 装载 OUT, 加工完毕 OUT, 加工 IN, 装载完毕 IN, 加工完成 IN;

方法: 机器忙/闲判断, 加工, 下载完成后机器由忙变闲。

其面向对象的程序描述为:

```
Class Machine
{    Private:
      int     Machine_ID;
      char    Machine_Name;
      PlaceList    Places;
      TransitionList    Transitions;
   Public:
      Idle_Or_Busy(machine);
      Process();
      Busy_To_Idle(machine); } c1, c2, m1, m2, m3;
Machine::Idle_Or_Busy(machine)
{    while (machine! = Machine_ID or Machine state is busy)
      Wait();       //If machine isn't available, then wait
      return machine;    // Send Message to robot to load part
```

对象内部操作(方法)的程序实现可以用面向对象语言直接实现, 也可以根据对象子网中各个变迁的触发和托肯的移动来表示。下面给出的是库所和变迁的面向对象定义, 它们在程序设计中是非常有用的, 可以减少冗余, 大大增加程序的可重用性。

```
Class Place
{   Private:
      int                  Place_ID;
      char                 Place_Name;
      TransitionList       In_Places;
      TransitionList       Out_Transitions;
      Colorset             Allowed_Colors;
      TokenList            Initial_Token;
      Tokenlist            Current_Token;
   Public:
      AddToken(Transition_ID);
      MoveToken(Transition_ID); }
Class Transiton:
{   Private:
      int                  Transition_ID;
      char                 Transition_Name;
      PlaceList            In_Places;
      PlaceList            Out_Places;
   Public:
      bool           All_Marked();
      bool           Can_Fire();
      float          Firing_Probability();
      fire(); }
bool Transition::Can_Fire()
{   float p;
      if (! All_Marked())
            return False;
      p= Firing_Probability();
      if (p= = 0) return False;
      if (Condition of this transition is not satisfied)
            return False;
      return True; }
Transition::fire()
{   if (Can_Fire() is True)
       In_Places.Move_Token(Transition_ID);
       Out_Places.Add_Token(Transition_ID);
      else
      Display Message( Can't Fire!); }
```

有了库所和变迁的类定义以后, 可以由它们派生出普通库所、消息库所、普通变迁、消息消息传递变迁等类的定义。

由于对象内部的方法在 OOPN 的对象子网中是由一系列库所和变迁组成的, 因此, 对象内部方法的实现可以看成是变迁被逐个触发的结果。这样, 对象内部的方法的程序顺序也可以由变迁的触发操作完成。

按照这样的方法, 逐步给每一个对象的方法写出程序。最后, 再定义一个管理各个对象之间消息传递的主程序, 整个软件系统的程序设计实现就完成了。可以看出, 面向对象设计概念和面向对象程序设计语言的引入, 很多已经设计好的程序段或定义好的类都可供引用, 大大提高了程序开发速度。

5 结 论

本文介绍了面向对象的 Petri 网方法, 并将它应用于应用系统的软件设计与开发过程中。给出了 FMS 控制软件设计示例。利用 OOPN 方法的最大好处是可以对软件系统设计开发过程进行有效的控制, 保证软件系统的可靠性和规范化水平。首先在软件未开发之前就可以利用形式化的方法对它进行检验, 从而减少软件设计中逻辑上的错误。另外在软件测试、调试和运行维护过程中, 利用 Petri 网的形式化和规范化的方法可以很快发现问题, 节省开发维护时间。但是目前 OOPN 在软件工程中的应用还需要大量的人工参与, 如程序编制、调试, 因此下一步研究目标是形成以 OOPN 方法为基础的一整套软件设计、代码自动生成与调试工具。

(下转 22 页)

表二 类的服务的功能测试用例

服务名称	person::loadIfWaiting(...)		
输 入	输入参数		引用数据
	P={elevNow In}		Du={waitingForElev, floorNowOn}
输 出	输出参数		修改数据
	P={loadIfWaiting}		Dd={waitingForElev}
输入数据名称	典型值	边界值	其它值
elevNow In	{0, 1, -1, MAXELEVS}	{MAX int, MIN int}	{-1, -2, 2, MAXELEVS, MAXELEVS+1}
waitingForElev	{0, 1, -1}	{MAX int, MIN int}	{-2, 2}
floorNowOn	{0, 1, -1, MAXELEVS}	{MAX int, MIN int}	{-1, -2, 2, MAXELEVS, MAXELEVS+1}
测试用例序号	输入数据		期望输出
1	waitingForElev=0 elevNow In=1 floorNowOn=0		loadIfWaiting=0 waitingForElev=0
2	waitingForElev=1 elevNow In=MAXELEVS floorNowOn=MAXELEVS		loadIfWaiting=1 waitingForElev=0

参考文献

[1] 郑人杰, 殷人昆, 陶永雷. 实用软件工程. 北京: 清华大学出版社 (第二版), 1997.4.

[2] 郑人杰. 计算机软件测试技术. 北京: 清华大学出版社, 1992.12.

[3] (美) Coad, P., Yourdon, E. 著. 邵维忠, 廖钢城, 李力译, 杨芙清校. 面向对象的分析. 北京: 北京大学出版社, 1992.2.

[4] (美) Coad, P., Yourdon, E. 著. 邵维忠, 廖钢城, 苏渭珍译. 杨芙清校. 面向对象的设计. 北京: 北京大学出版社, 1994.11.

[5] Jorgensen, P. and Erickson, C. Object-Oriented Integration Testing. Commun. ACM 37, 9 (Sept. 1994), 30-38.

[6] Murphy, G.C., Townsend, G. and Wong, S.W. Experience with Cluster and ClassTesting. Commun. ACM 37, 9 (Setp. 1994), 39-47.

[7] Kung, D., Gao, J., Hsia, P., Toyoshima, Y., Chen, C., Kim, Y.S., and Song, Y.K. Developpingan Object-Oriented Software Testing and Maintenance Environment. Commun. ACM 38, 10 (Oct. 1995), 75-87.

[8] 邵维忠, 王立福, 梅宏, 杨芙清. 面向对象的软件测试—方法研究及系统设计. 软件工程进展. 北京: 清华大学出版社, 1996, 115-122.

[9] 赵元聪, 朱三元. 面向对象软件测试的认识. 计算机应用与软件, 1996(3)13.

[10] Kung, C.H., Gao, J., Hsia, P., Lin, J., and Toyoshima, Y. Design Recovery for SoftwareTesting of Object-Oriented Porgrams. In Proceedings of the Working Conference on ReverseEngineering (Baltimore, MD, May21-23, 1993), 202-211.

(上接 18 页)

参考文献

[1] Pinci V.O., Shaprio R.M. An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets. Advances in Petri Nets 1990 (ed. G. Rozenberg). Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1990.

[2] Cofrancesco P., Cristoforetti A., Scattolini R. Petri Nets Based Approach to Software Development for Real-time Control. IEE Proceedings-D. Vol 138, No. 5, 1991, 138(5)pp. 474-478.

[3] Reisig W. Petri Nets in Software Engineering. In: Proceeding of Advances in Retri Nets, 1986, LNCS 254, Springer-Verlag, pp. 63-96.

[4] Lakos C.A., Keen C.C. Simulation with Object-Oriented Petri Nets. Technical Report. Department of Computer Science. University of Tasmania, Australia, April, 1994.

[5] Camurri A., Franchi P., Vitale M. Object-Oriented Approach to High-Level Petri Nets. Microprocessing and Microprogramming. 1992, 35. 213-220.

[6] Keen C.D., Lakos C.A. A Methodology for the Construction of Simulation Models Using Object-Oriented Petri Nets. Technical Report, Depart of Computer Science, University of Tasmania, Australia, April 1994.