基于 Petri 网和 UML 的流水作业调度模型设计及实现*

赵国毅，张广泉
(重庆师范大学 数学与计算机科学学院，重庆 400047)

摘 要：为了对流水作业调度的行为提供支持，需要对其进行有效的建模和模型分析。Petri 网具有坚实的理论基础和易于使用的图形表示，是一种理想的建模和分析工具。UML 是一种易于编程实现的面向对象建模工具。针对 UML 和 Petri 网建模的特点，本文采用 Petri 网与 UML 相结合的建模分析方法，融合了 Petri 网精确、严格的形式化描述和 UML 友善的用户界面的优点。并提出了将 Petri 网转化为一种可以表示对象间的同步、并发的扩展状态图的方法。利用 Petri 网描述系统的动态特性，经过分析、验证或仿真后，通过提供的转化规则将 Petri 网模型转化成相应的 UML 动态图、扩展的状态图，对流水作业调度问题建模。利用动态规划算法解决满足 Johnson 法则的最优作业调度问题，并用 Java 实现。

关键词：Petri 网；UML；流水作业调度；动态规划算法；Johnson 法则
中图分类号：TP311.11 **文献标识码**：A **文章编号**：1672-6693(2008)02-0040-06

Petri 网[1-4]在离散事件动态系统中是一种逻辑层次的建模方法，能够对分布、并发过程进行有效形式的建模，能够对研究对象的系统结构和动态行为进行有效的分析、验证。UML[2,5-6]是一个通用的可视化建模语言，可直观地描述用户需求、系统的静态模型和动态行为，能够从不同的角度考察系统，便于领域专家和用户与系统开发人员之间的交流。本文将 Petri 网和 UML 这两种优秀的建模方法结合起来，优势互补。并提出了将 Petri 网转化为一种可以表示对象间的同步、并发的扩展状态图的方法。对流水作业调度问题建模。利用动态规划算法解决满足 Johnson 法则的最优作业调度问题，通过 Java 编程上机实现。

1 Petri 网与 UML 结合的建模分析方法

该方法主要思想是利用 Petri 网对系统动态、并发部分建模，直接分析验证。利用相应的转化规则将其转化为 UML 行为模型视图。静态部分直接用 UML 描述。二者集成为完整的系统模型，利用 UML 建模工具很容易实现程序代码的转换。其核心步骤是 Petri 网向 UML 的转化。

1.1 Petri 网建模

针对系统中具有动态、并发特性的部分进行 Petri 网建模。对所建模型进行分析，根据其特征，采用合适的验证方法[8]（状态空间搜索、模型检验、结构方法、使用演绎和进程代数的一些高级方法等）也可将不同方法结合起来，对其不同性质（有界性、活性、可逆性等）进行验证。

1.2 将 Petri 网转化为 UML 状态图

首先，遍历 Petri 网中所有的库所，将其所描述的事物归类对应为 UML 中的对象。对 Petri 网中的所有库所按照它们表示的对象分组，称其为该对象的库所组。如果是子 Petri 网[9-10]模型向状态图转化不需要此过程。各对象每一组连通的库所，按照规则转换：库所转化为 UML 状态图中的状态；根据变迁的不同语义，映射成不同类型的事件；与本对象库所具有同步、并发性质的库所转化为相应转换的监护条件；资源位置也对应监护条件；弧对应着转换。

1.2.1 顺序 Petri 网向状态图转化 应用上面的规则，如图 1 所示。Petri 网中库所 A1 和 A2 表示对象 A 的两个状态，B1 是对象 B 某一状态。

* 收稿日期：2007-03-23 修回日期：2007-11-29
资助项目：重庆市自然科学基金(No. CSTC,2006BB2259)
作者简介：赵国毅(1981-)，男，硕士研究生，研究方向为软件工程与形式化方法。通讯作者：张广泉，Email：zgqxyz@vip.sina.com

图1 顺序 Petri 网模型转化为状态图

1.2.2 具有并发性质的 Petri 网向状态图转化 如图2[11]所示。

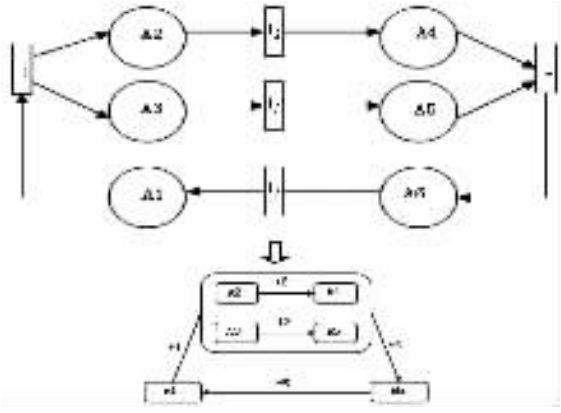

图2 具有并发特性的 Petri 网模型转化为状态图

1.3 将 Petri 网转化为扩展的状态图

由于 UML 行为模型视图中,状态图用于对单个对象的生命期建模,序列图和协作图用于对共同完成某些动作的对象群体进行建模,而不能体现对象状态变化,故本文提出一种扩展的状态图,以描述对象以及对象间状态的控制流。将 UML 状态图中表示并发变迁的同步栅[12]的地位提升,使其可以连接对象间的状态,其它意义不变,用以表示不同对象状态间的同步、并发。

其转化规则为:

1) Petri 网中的所有库所转化为各对象相应的状态。

2) 同步栅的运用。并发变迁的表示涉及到同步栅,它可以有一个或多个来自源状态且指向它的箭头,也可以有一个或多个出自于该同步栅且指向目标状态的箭头。当所有源状态都完成其活动后,并发变迁就会激活。当并发变迁激活后,其所有的目标状态都将被激活。

①并发的转化方式。如图3所示。

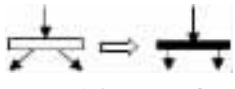

图3 并发的转化方式

②同步的转化方式。如图4所示。

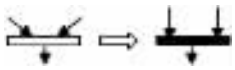

图4 同步的转化方式

③多个源状态和目标状态同步、并发的转化方式。如图5所示。

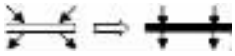

图5 多个源状态和目标状态同步、并发的转化方式

3) Petri 网中同对象库所间的变迁转化为相应对象状态转换的事件,资源位置转化为监护条件。弧对应转换。

1.4 将 Petri 网转换为 UML 顺序图和协作图

UML 中的顺序图将交互关系表示为一个二维图。按照对象分类的库所组表示对象在其生命周期中的各个状态。将消息分为对象之间通讯和对象自身调用,具体方法是:按弧指向的顺序逐个考察库所组中的相邻库所之间的变迁,如果变迁的前置集包含不属于该库所组的库所,那么将该库所到此变迁的弧转换为 UML 消息。如果是库所组内的则转化为本对象自身操作调用。

根据状态机图、顺序图和原 Petri 网可以容易得到协作图。Rational Rose 也提供了直接转化功能,这里不再详述。

1.5 UML 建模

对系统静态部分用 UML 建模,与 Petri 网模型转化出的 UML 行为模型视图一起形成完整的 UML 模型,利用 Rational Rose 等相应的软件工具直接生成 JAVA、C++等语言的代码框架。

2 Petri 网与 UML 结合的流水作业调度问题建模

所谓流水作业调度[12]是指,有 n 个作业,每个作业 i 均被分解为 m 项任务:$T_{i1}, T_{i2}, \cdots, T_{im}$ $(1 \leqslant i \leqslant n)$。要把这些任务安排到 m 台机器上进行加工,每个作业 i 的第 j 项任务 T_{ij} $(1 \leqslant j \leqslant m)$ 只能安排在机器 P_j 上,并且作业 i 的第 j 项任务的开始加工时间均安排在第 $i-1$ 项任务加工完毕之后。若任务 T_{ij} 在机器 P_j 上进行加工需要的时间为 t,最优流水作业调度就要求确定一种安排任务的方法,使得完成这 n 个作业的加工时间为最少。当机器数 $m \geqslant 3$ 时,流水作业调度问题是一个 NP-hard 问题,即求解时间形成指数型困难,该问题至少在目前基本上没有可能找到多项式时间的算法。只有当 $m = 2$ 时,

该问题可能有多项式时间的算法。本文主要讨论利用动态规划算法解决机器数为 2 的情况下流水作业的最优调度问题。

2.1 流水作业调度问题的 Petri 网建模

流水作业调度问题 Petri 网模型如图 6 所示。

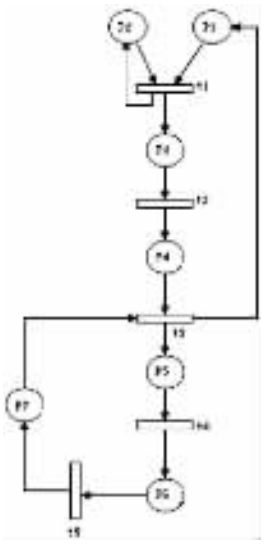

图 6　流水作业调度问题的 Petri 网模型

p1 机器一处于就绪状态;p2 作业处于就绪状态;p3 机器一处于加工状态;p4 机器一处于等待状态、中间件生成;p5 机器二处于加工状态;p6 机器二处于等待状态、本次作业加工完成;p7 机器二处于就绪状态。t1 机器一开始加工;t2 机器一加工完毕 t3 机器二开始加工、机器一就绪 t4 机器二加工完毕 t5 机器二就绪。

对所建 Petri 网进行分析,选用前面所述的合适的验证方法,或将不同方法结合使用,验证其各种性质。

2.2 将 Petri 网转化为 UML 状态图

遍历 Petri 网从 p1 到 p7 的 7 个库所中,找到该系统共 3 个对象:作业、机器一、机器二。

将库所按其所描述的对象分组:作业:p2;机器一:p1,p3,p4 机器二:p5,p6,p7。

按照前面阐述的转化方法,得到各对象的状态图。作业、机器一、机器二的状态图如图 7、8、9 所示。

图 7　作业的状态图

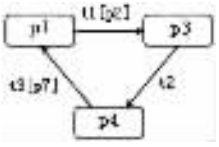

图 8　机器一的状态图

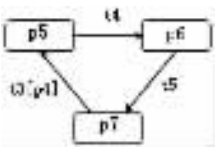

图 9　机器二的状态图

2.3 将 Petri 网转化为扩展的状态图

按照前面阐述的方法得到扩展的状态图。如图 10 所示。

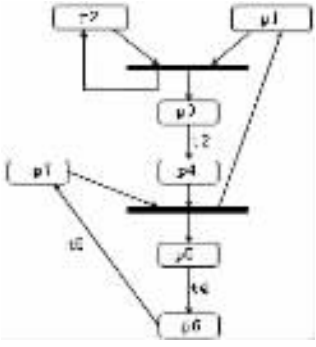

图 10　流水作业调度的扩展状态图

2.4 将 Petri 网转化为 UML 顺序图

如图 11 所示。

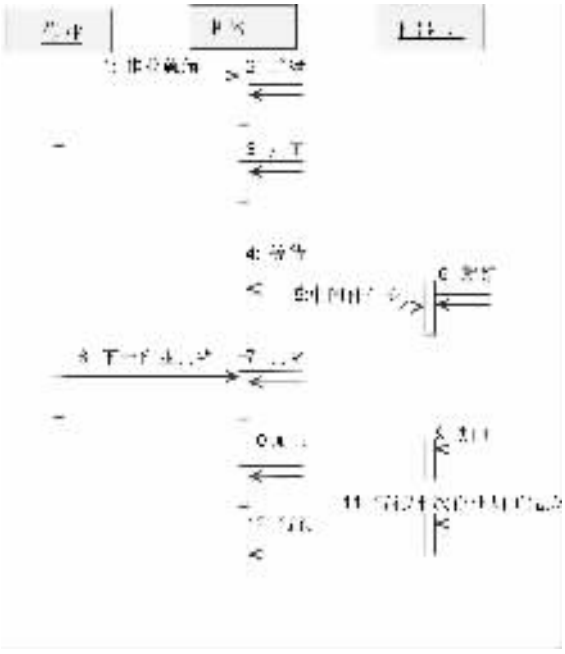

图 11　流水作业的顺序图

通过以上从 Petri 网转化为 UML 状态图、扩展的状态图、顺序图的建模过程可以对流水作业调度问题进行有效的建模和模型分析,直观地展现流水作业调度的行为过程,易于理解。

3 流水作业调度的 Johnson 法则及其 JAVA 实现

设$\prod$是作业集 S 在机器 M_2 的等待时间为 t 时的任一最优调度。M_1、M_2 加工作业 i、j 所需的时间分别被表示为 a_i、b_i 和 a_j、b_j。若在这个调度中,假设先做第 i 个作业,再做第 j 个作业,则由动态规划递归式可得

$$T(S,t)=a_i+T(S-\{i\},b_i+\max\{t-a_i,0\})=$$
$$a_i+a_j+T(S-\{i,j\},b_j+$$
$$\max\{b_i+\max\{t-a_i,0\}-a_j,0\}$$

假设先做第 j 个作业,再做第 i 个作业,同理可得

$$T'(S,t)=a_j+T(S-\{j\},b_j+\max\{t-a_j,0\})=$$
$$a_j+a_i+T(S-\{j,i\},b_i+$$
$$\max\{b_j+\max\{t-a_j,0\}-a_i,0\})$$

令 $T\leqslant T'$,经过比较计算得出结论 $\min\{a_j,b_i\}\geqslant\min\{a_i,b_j\}$。当此式成立时,则称作业 i 和 j 满足 Johnson 不等式。下面是流水作业调度问题的一种实现方法。

例如 作业数 $n=5$,每个作业在机器 P_1、P_2 上的运行时间表示为

$$\{\{a_1,b_1\}\ \{a_2,b_2\}\ \{a_3,b_3\}\ \{a_4,b_4\}\ \{a_5,b_5\}\}=$$
$$\{\{3,6\},\{4,2\},\{10,15\},\{8,9\},\{7,3\}\}$$

根据 Johnson 法则,设计一种算法求解流水作业的最优调度问题。步骤如下。

1)对作业集合进行分组。将在机器 P_1 上执行时间小于在 P_2 上执行时间的作业分为一组,即 $N_1=\{i\mid a_i<b_i\}$,其余作业分为另一组,即 $N_2=\{i\mid a_i>=b_i\}$;

2)将 N_1 组中的作业依 a_i 的非递减序排序,将 N_2 组中的作业依 b_i 的非递增序排序;

3)N_1 中的作业接 N_2 中的作业构成满足 Johnson 法则的最优调度。

该算法 Java 实现如下。

```
class Job {        //定义存储工作流的类
  int first;//完成任一作业在机器一上需要的时间
  int second;//完成任一作业在机器二上需要的时间
  int index;//作业的顺序号
  boolean type;//true 代表 first < second
Job(int f, int s, int i, boolean t) {//构造函数
    this.first = f;
    this.second = s;
    this.index = i;
    this.type = t;
    }
Job() {        //默认的构造函数
      this(0,0,0,false);
    }}
public class Schedule {              //定义调度类
public static void main(String args[]) {
int[][]
jobArray = {{3,6},{4,2},{10,15},{8,9},{7,3}};
int jobNum = jobArray.length;
Job[] jobs = new Job[jobNum];
for(int i=0;i<jobNum;i++) {
jobs[i] = new Job(jobArray[i][0],
jobArray[i][1],i+1,
jobArray[i][0]<jobArray[i][1]?true:false);
      }
    sortByType(jobs, jobNum);
    sortTypeTrue(jobs, jobNum);
    sortTypeFalse(jobs, jobNum);
    printJobs(jobs,jobNum);
    }
/*将对象数组按 type 属性分组,true 为先 false 为后*/
private static void sortByType(Job[] jobs, int jobNum) {
    int begin = 0;
    int end = jobNum - 1;
    while(begin<end) {
        while(jobs[begin].type != false)
            begin++;
        while(jobs[end].type != true)
            end--;
        if(begin<end) {
      swap(jobs, begin, end);
        begin++;
        end--;
        }}}
/*将对象数组中属性为 true 的对象按 first 属性大小递增排序*/
private static void sortTypeTrue(Job[] jobs, int jobNum) {
```

```
Num) {
for(int i = 0 ;i < jobNum - 1&&jobs[i]. type = = true ;
i + +) {
for(int j = i + 1 ;j < jobNum&&jobs[j]. type = = true ;
j + +)
    if(jobs[i]. first > jobs[j]. first)
        swap(jobs, i, j) ;
        }}
/ * 将对象数组中属性为 false 的对象按 second 属性
大小递减排序 * /
private static void sortTypeFalse(Job[] jobs,int job-
Num) {
for(int i = jobNum - 1 ;i > - 1&&jobs[i]. type = =
false ;i - -) {
for(int j = i - 1 ;j > 0&&jobs[j]. type = = false ;j -
-)
    if(jobs[i]. second > jobs[j]. second)
            swap(jobs, i, j) ;
        }}
/ * 交换数组中的两个对象 * /
private static void swap(Job[] jobs, int a, int b) {
        Job tempJob = new Job() ;
        tempJob. first = jobs[a]. first ;
        tempJob. second = jobs[a]. second ;
        tempJob. index = jobs[a]. index ;
        tempJob. type = jobs[a]. type ;
        jobs[a]. first = jobs[b]. first ;
        jobs[a]. second = jobs[b]. second ;
        jobs[a]. index = jobs[b]. index ;
        jobs[a]. type = jobs[b]. type ;
        jobs[b]. first = tempJob. first ;
        jobs[b]. second = tempJob. second ;
        jobs[b]. index = tempJob. index ;
        jobs[b]. type = tempJob. type ;
    }
/ * 打印结果 * /
private static void printJobs(Job[] jobs, int jobNum)
 {
    System. out. print("The sequence is :") ;
    for(int i = 0 ;i < jobNum ;i + +)
        System. out. print(jobs[i]. index + " ") ;
            System. out. println() ;
        }}
```

程序的执行结果如图 12 所示。

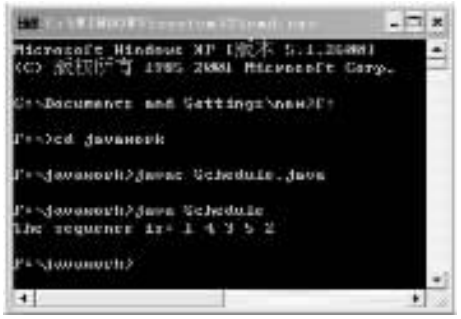

图 12　流水调度算法的 Java 实现结果

4 结论和展望

本文采用一种利用 UML 和 Petri 网相结合的建模分析方法,特别提出了 Petri 网转化为一种可以表示对象间的同步、并发的扩展状态图,应用于解决流水作业调度问题建模。首先可以利用 Petri 网形式化特征及良好的并发描述能力,对系统中并发的部分建模、分析和验证,然后使用本文提出的方法,给出 Petri 网到 UML 的映射,并进一步编程实现。通过 Petri 网模型向 UML 模型的转化,改变了传统方法的不足,使 UML 模型能够充分描述系统的并发、同步和冲突性质。笔者将进一步深入研究 Petri 网向 UML 转化的严密性,如:顺序图中如何描述资源共享、并发;如何更好地描述状态图之间的关系以及该方法在 Web 程序开发中的应用等方向。

参考文献:

[1] 袁崇义. Petri 网原理与应用[M]. 北京:电子工业出版社,2005.

[2] 杨文龙,古天龙. 软件工程[M]. 北京:电子工业出版社,2004.

[3] 古天龙. 软件开发的形式化方法[M]. 北京:高等教育出版社,2005.

[4] 蒋昌俊. Petri 网的行为理论及其应用[M]. 北京:高等教育出版社,2003.

[5] RUMBAUGH J,JACOBSON I,BOOCH G. The UML Reference Manual[M]. 2rd ed. 北京:机械工业出版社,2005.

[6] BOOCH G, RUMBAUGH J, JACOBSON I. The UML User Guide[M]. 北京:机械工业出版社,2001.

[7] 张姝,张广泉. UML 顺序图的一种形式化描述方法[J]. 重庆师范大学学报(自然科学版),2007,24(3):42-45.

[8] 张广泉. 关于软件形式化方法[J]. 重庆师范学院学报

(自然科学版),2002,19(2):1-4.
[9] GIRAULT C,VALK R. Petri Nets for Systems Engineering [M]. 北京:电子工业出版社,2005.
[10] 江金龙,周献中,孙勇成,等. 基于 UML 和 Petri 网的层次建模分析方法[J]. 系统仿真学报,2006,18(2):290-293.
[11] 方丁,郝东,林琳. 一种利用 UML 的 Petri 网软件实现方法[J]. 计算机应用,2004,24(9):132-135.
[12] 徐宝文,周毓明,卢红敏. UML 与软件建模[M]. 北京:清华大学出版社,2006.
[13] 王晓东. 计算机算法设计与分析[M]. 北京:电子工业出版社,2001.

Design of Flow Shop Scheduling Model Based on Petri Net and UML and Implemented with Java

ZHAO Guo-yi, ZHANG Guang-quan

(College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047, China)

Abstract: In order to support the behavior of the Flow Shop Scheduling, an effective modeling and an analysis method are proposed to solve this problem. Petri Nets is a graphical modeling tool and also a formal mathematic tool. UML is an object-oriented modeling tool which is easy to be implemented with program. Owing to the characteristics of UML and Petri net, the modeling analysis method combining UML with Petri net is adopted, this method puts their advantages together, and a method of Petri Net transformed to an extended State Diagram which could express synchronization and concurrence between the object is proposed. Using Petri Net Shows systemic dynamic characteristic, After analysis, confirmation or simulation, we transform them into the corresponding dynamic Diagram, extended State Diagram through provided rule of transform and modeling Flow Shop Scheduling problem. By dynamic programming algorithm, the problem of superior flow jobs scheduling to meet Johnson's Rule is solved and implemented with Java.

Key words: Petri net; UML; flow shop scheduling; dynamic programming algorithm; Johnson's rule

(责任编辑 游中胜)