УДК 004.94: 519.876.5

И.В. СТЕЦЕНКО

АЛГОРИТМ ИМИТАЦИИ ПЕТРИ-ОБЪЕКТНОЙ МОДЕЛИ

Анотація. *Розглядається імітація Петрі-об'єктної моделі, що ґрунтується на об'єктно-орієнтованій технології та стохастичній мережі Петрі з часовими затримками. Розроблений алгоритм імітації Петрі-об'єктної моделі. Реалізація алгоритму створена засобами Java-класів.*
Ключові слова: *мережа Петрі, моделювання, об'єктно-орієнтована технологія, рівняння станів, Java, алгоритм імітації.*

Аннотация. *Рассматривается имитация Петри-объектной модели, основывающейся на объектно-ориентированной технологии и стохастической временной сети Петри. Разработан алгоритм имитации Петри-объектной модели. Реализация алгоритма создана средствами Java-классов.*
Ключевые слова: *сеть Петри, моделирование, объектно-ориентированная технология, уравнения состояний, Java, алгоритм имитации.*

Abstract. *The simulation of Petri-object model based on object-oriented technology and time delay stochastic Petri nets is regarded. The Petri-object model's simulation algorithm is developed. The implementation of the algorithm is created by means of Java-classes.*
Keywords: *Petri net, modeling, object-oriented technology, state equations, Java, simulation algorithm.*

1. Вступление

Временные сети Петри являются универсальным средством формализации процессов, имеющих событийный характер, и широко используются в различных областях. Ежегодно проводится конференция International Conferences on Application and Theory of Petri Nets (в 2011 году проводилась 32-ая конференция), где обсуждаются новейшие достижения в области теоретических и прикладных исследований с применением сетей Петри.

Формализация процессов функционирования средствами временных сетей Петри позволяет еще на этапе формализации описать сложные взаимосвязи между элементами системы с учетом процессов, происходящих во времени параллельно. Однако использование сетей Петри для целей имитационного моделирования на сегодняшний день ограничено, о чем свидетельствует и тот факт, что известная монография по имитационному моделированию [1] не содержит каких-либо упоминаний о сетях Петри. Причиной является то, что недостаточно разработаны 1) методы формализации сложных процессов временной сети Петри, 2) методы математического моделирования стохастической временной сети Петри, 3) алгоритмы имитации стохастической временной сети Петри.

В [2] разработана теория Петри-объектного моделирования, позволяющего, в отличие от существующих технологий моделирования, создавать модели больших систем средствами объектно-ориентированного моделирования и стохастических временных сетей Петри с конфликтными и многоканальными переходами.

В настоящей работе излагается технология построения алгоритма имитации Петри-объектной модели и описываются особенности реализации этого алгоритма средствами Java.

2. Библиотека Java-классы для реализации Петри-объектов

Для профессионального программиста не представляет труда составить несколько классов для реализации сети Петри без временных задержек и даже оснастить их графическим интерфейсом с элементами анимации. Перечень программ, реализующих различной сложности сети Петри, можно найти на сайте [2]. К сожалению, среди этих продуктов мало действительно грамотно реализованных программ, способных осуществлять имитацию времен-

ных сетей Петри с достаточно большим количеством элементов. Как правило, они имеют характер обучающих программ и служат для ознакомления с правилами функционирования сети Петри. Кроме того, их нельзя достроить и применить для своих целей в другом проекте. В связи с этим разработка библиотеки классов для реализации Петри-объектного моделирования начиналась с разработки библиотеки Java-классов для реализации Петри-объектов.

Библиотека Java-классов, разработанная в данной работе, основывается на реализации стохастической временной сети Петри с многоканальными и конфликтными переходами, с информационными связями. Библиотека разрабатывалась для создания и моделирования Петри-объектов, но может использоваться и как самостоятельная. Основная ее цель – стать основой будущих проектов по Петри-объектному моделированию, создавая условия для усовершенствования и приспособления к конкретным условиям задачи.

Сеть Петри-объекта определяется, как и обычная временная сеть Петри, множеством позиций, множеством переходов, множеством дуг, множеством информационных дуг, множеством натуральных чисел, задающих кратности дуг (количество связей), множеством пар значений, задающих приоритет и вероятность запуска переходов, множеством неотрицательных действительных значений, характеризующих временные задержки в переходах.

Правила функционирования сети Петри сформулированы как очень простые, однако допускают различное толкование различными исследователями в зависимости от контекста задачи [3]. Отсюда большое количество различных модификаций и расширений сетей Петри. Данный проект основывается на классическом понятии сети Петри с конфликтными переходами [3], временной сети Петри с многоканальными переходами [4], стохастической сети Петри как сети Петри, в которой временные задержки заданы случайными числами с заданными законами распределений, и сети Петри с информационными связями [5].

Функционирование сети Петри рассматривается во времени, продвигающемся от одного события, связанного с выходом маркеров из перехода, до ближайшего следующего. Такой способ продвижения времени широко используется при построении алгоритмов имитации сложных систем [1]. Обозначим последовательность моментов времени, соответствующих событиям, $t_1, t_2 \ldots t_n, \ldots$, причем в течение времени $t_{n-1} < t < t_{n-1}$ в сети Петри не происходит никаких событий.

Во временной сети Петри маркеры, при выполнении условия запуска, входят в один из свободных (не активных) каналов перехода, и в течение времени, равного временной задержке этого перехода, канал находится в состоянии «активен». По истечении временной задержки происходит выход маркеров из перехода. Изменение маркировки приводит к тому, что становятся, возможно, выполненными условия запуска других переходов сети Петри. В каждый момент времени $t_1, t_2 \ldots t_n, \ldots$, соответствующий событию, происходит выход маркеров из всех переходов, для которых наступил момент выхода, и вход маркеров в переходы, для которых выполнено условие запуска. Поэтому для временной сети Петри традиционное понятие запуска перехода, объединяющего в себе вход и выход маркеров из перехода, не пригодно. Следует различать вход маркеров в переходы сети Петри и выход маркеров из переходов сети Петри.

При выходе маркеров из перехода сети Петри осуществляется выход из каналов перехода и состояние сети Петри изменяется следующим образом: 1) из множества моментов выхода из каналов перехода удаляются моменты времени, равные текущему моменту времени t_n; 2) в выходные позиции перехода добавляются маркеры в количестве, равном кратности соответствующей дуги, помноженному на количество каналов, из которых осуществляется выход в текущий момент времени.

При входе маркеров в переход сети Петри осуществляется вход в каналы перехода и состояние сети Петри изменяется следующим образом: 1) к множеству моментов выхода из каналов перехода добавляется момент времени, равный текущему моменту времени плюс временная задержка; 2) из входных позиций перехода удаляются маркеры в количестве, равном кратности соответствующей дуги.

Обозначим преобразования состояния сети Петри, происходящие при выходе маркеров из переходов и при входе маркеров в переходы, D^+ и D^- соответственно. Математическое описание преобразований состояния сети Петри, происходящие при выходе маркеров из переходов и входе маркеров в переходы, содержится в работе [6].

Поскольку переходы сети Петри многоканальные, то возможен многократный вход маркеров в переходы сети Петри $\left(D^-\right)^m = \underbrace{D^- \circ D^- \circ ... \circ D^-}_{m}$. Причем количество m входов определяется достижением такого состояния сети Петри, при котором ни для одного из ее переходов не выполнено условие входа маркеров в переход. Пример изменения состояния сети Петри приведен на рис. 1. Продвижение времени и выполнение соответствующих преобразований показано на рис. 2.

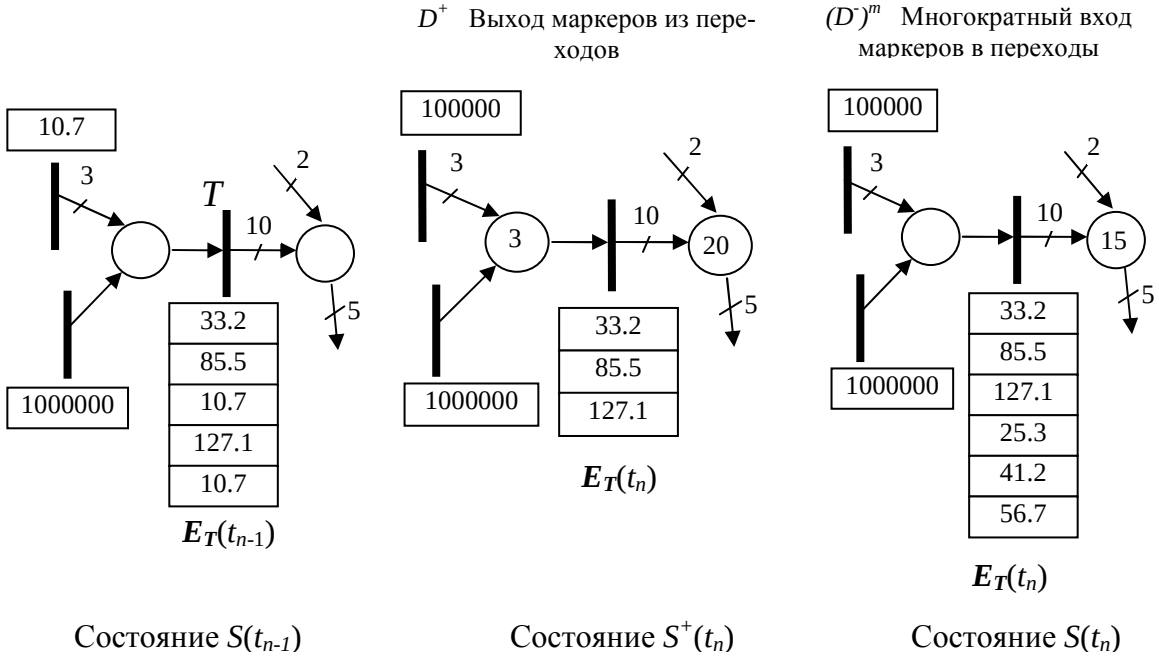

Рис. 1. Преобразование состояния фрагмента сети Петри в момент времени $t_n = 10{,}7$

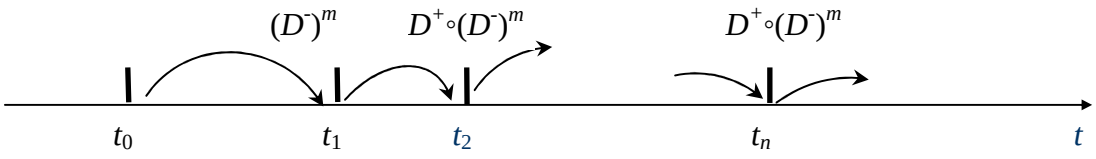

Рис. 2. Продвижение времени и выполнение преобразований сети Петри

Основными классами библиотеки являются класс PetriNet (Петри-сеть), конструирующий сеть Петри, и класс PetriSim (Петри-имитатор), осуществляющий имитацию функционирования временной сети Петри.

Класс PetriNet (Петри-сеть) выполняет агрегирование массивов позиций, переходов, входных и выходных связей (по отношению к переходам). Методы этого класса обеспечивают доступ ко всем массивам данных (рис. 3).

Класс PetriP (Петри-позиция) содержит значение маркировки позиции и методы, обеспечивающие добавление и удаление маркеров из позиции, а также подсчет среднего количества маркеров, находящихся в позиции.

Класс PetriT (Петри-переход) содержит самые важные методы, определяющие функционирование сети Петри. Основные поля и методы этого класса представлены на рис. 4. Значение buffer описывает количество активных каналов перехода. Список timeOut содержит множество значений, равных моментам выходов маркеров из перехода, и содержит значение ∞, если все каналы перехода свободны (не активны). Количество активных каналов buffer равно длине списка timeOut, если есть хоть один активный канал, и равно нулю, если все каналы перехода свободны. Список inP содержит номера позиций, являющихся входными для перехода, список inPwithInf – номера позиций, связанных с переходом информационной связью, список outP – номера позиций, являющихся выходными для перехода.

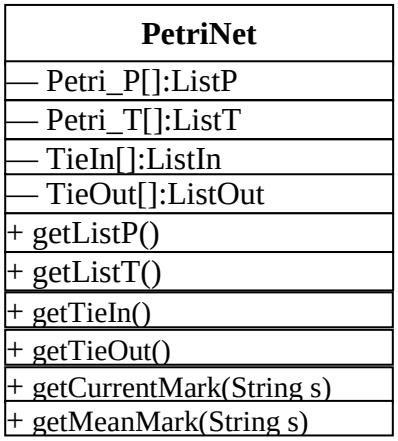

Рис. 3. Основные поля и методы класса Петри-сеть

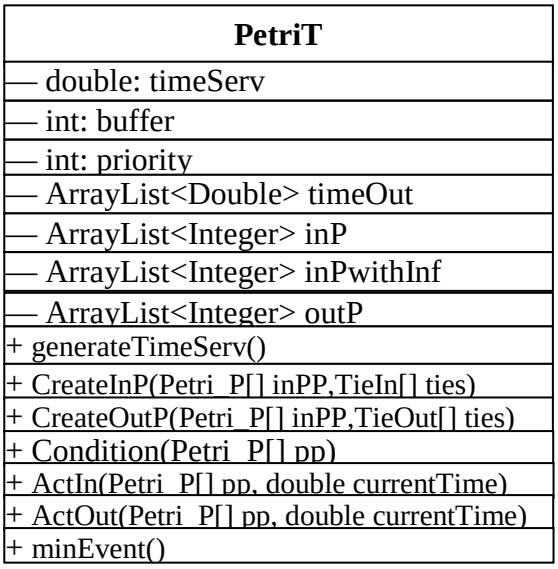

Рис. 4. Основные поля и методы класса Петри-переход

Метод CreateInP(Petri_P[] inPP,TieIn[] ties) формирует списки входных позиций перехода inP и inPwithInf, а метод CreateOutP(Petri_P[] inPP,TieOut[] ties) – список выходных позиций перехода outP. Условие запуска перехода проверяется с помощью метода Condition(Petri_P[] pp), а для запуска перехода предназначены методы ActIn(Petri_P[] pp, double currentTime) и ActOut(Petri_P[] pp, double currentTime), осуществляющие вход и выход маркеров из перехода в соответствии с множествами входных и выходных позиций из перехода и кратностями связей. Метод minEvent() определяет наименьшее из всех значений, содержащихся в списке timeOut, и запоминает номера каналов, соответствующих наименьшему значению timeOut.

Метод ActOut () описывается следующим Java-кодом:

```
public void ActOut(Petri_P[] pp, double currentTime)
{
 if(buffer>0)
 {
  for (int j=0; j<outP.size();j++)
     pp[outP.get(j)].increaseMark(quantOut.get(j));
```

```
        if(num==0&&(timeOut.size()==1)) timeOut.set(0, timeModeling+100);
        else
           timeOut.remove(num);
        buffer--;
       }
       else ;
}
```

Метод Condition() определяет выполнение условия запуска перехода и содержит проверку не только для обычных связей, но и для информационных, что описывается следующим Java-кодом:

```
public boolean Condition(Petri_P[] pp) {
       boolean a = true; boolean b = true;
         for (int i=0; i<inP.size();i++)
           if(pp[inP.get(i)].getMark()<quantIn.get(i))
                {a=false;break;}
         for (int i=0; i<inPwithInf.size();i++)
           if(pp[inPwithInf.get(i)].getMark()<quantInwithInf.get(i))
                {b=false; break; }
        if (a==true && b==true)
           return true;
        else
           return false;
}
```

Метод ActIn(), с учетом того, что отнимание маркеров происходит только вдоль обычных дуг, описывается следующим Java-кодом:

```
public void ActIn(Petri_P[] pp, double currentTime) {
      if (this.Condition(pp)==true) {

          for (int i=0; i<inP.size();i++)
            pp[inP.get(i)].decreaseMark(quantIn.get(i));
          if(buffer==0) timeOut.set(0, currentTime+this.getTimeServ());
          else timeOut.add(currentTime+this.getTimeServ());
          buffer++;
         this.minEvent();
       }
       else ;
}
```

Алгоритм имитации стохастической сети Петри с многоканальными и конфликтными переходами, с информационными связями строится на основе событийного подхода с продвижением времени по принципу до ближайшего события. Этот принцип широко используется в имитационном моделировании [1]. Поскольку элементом сети Петри является переход, представляющий событие, то алгоритм имитации очевиден. Однако при наличии многоканальных и конфликтных переходов, информационных связей алгоритм имитации усложняется следующим образом:

1. В сети Петри с конфликтными переходами сначала проверяется условие запуска всех ее переходов, потом решается конфликт для конфликтных переходов и только потом осуществляется вход маркеров в переходы, «выигравшие» конфликт.

2. В сети Петри с многоканальными переходами условие запуска перехода проверяется и осуществляется запуск перехода столько раз, сколько это позволяет количество маркеров в его входных позициях.

3. В сети Петри с информационными связями при проверке условия запуска перехода наличие маркеров в количестве, равном кратности связей, проверяется, но при входе маркеров в переход маркеры вдоль информационных связей не отнимаются.

Алгоритм имитации стохастической сети Петри с временными задержками, многоканальными переходами, информационными связями состоит из следующих действий:

◦ Формировать данные о структуре и начальном состоянии сети Петри (списки позиций ListP, переходов ListT, входных связей ListIn и выходных связей ListOut).

◦ Осуществить первоначальный запуск переходов (преобразование $\left(D^-\right)^m\left(\left(\mathbf{S}(t_0)\right)\right)$).

◦ Пока не исчерпано время моделирования $(t_n < timeMod)$:

• продвинуть время в момент ближайшего события;

• осуществить выход маркеров из переходов сети Петри (преобразование $D^+(\mathbf{S}(t_{n-1}))$;

• определить конфликтные переходы и решить конфликт переходов;

• осуществить вход маркеров в переходы сети Петри (преобразование $\left(D^-\right)^m\left(\left(\mathbf{S}^+(t_n)\right)\right)$).

◦ Вывести результаты моделирования.

Результаты моделирования формируются на основе наблюдения состояния модели, характеризующегося количеством маркеров в позициях и количеством активных каналов переходов, в течение времени моделирования с помощью статистических методов.

PetriSim
— Net: PetriNet
— int: priority
— double: timeMod
— Petri_T: eventMin
— double: timeMin
— boolean: STOP
— double: timeCurr
+ setPriority(int a)
+ EventMin()
+ findActiveT()
+ DoConflikt(ArrayList<Petri_T> TT)
+ Start()
+ NextEvent()
+ DoStatistica()
+ DoT()
+ Go(double time)

Рис. 5. Основные поля и методы класса Петри-имитатор

Класс объектов PetriSim (Петри-имитатор) реализует имитацию некоторого реального объекта в соответствии с динамикой функционирования, заданной стохастической временной сетью Петри с конфликтными и многоканальными переходами (рис. 5). Информация о сети Петри содержится в поле PetriNet объекта Петри-имитатор. Поле priority предназначено для хранения информации о приоритете Петри-объекта. Поле timeCurr сохраняет информацию о текущем моменте времени. Поле timeMin хранит информацию о моменте ближайшего события, а поле eventMin – информацию о переходе, соответствующем моменту ближайшего события. Поле STOP сигнализирует о том, что сеть Петри не активна, т.е. все ее переходы находятся в состоянии «не активен» и маркировка позиций не позволяет запуск ни одного из переходов.

Метод Start() выполняет первоначальное преобразование сети Петри $\left(D^-\right)^m\left(\left(\mathbf{S}(t_0)\right)\right)$, заключающееся в многократном входе маркеров в переходы сети Петри. Продвижение времени осуществляется с использованием метода EventMin(). Преобразование сети Петри $\left(D^-\right)^m\left(D^+(\mathbf{S}(t_{n-1}))\right)$, соответствующее текущему моменту времени, осуществляется с помощью метода NextEvent(). Этот метод использует метод findActiveT() и метод DoConflikt(ArrayList<Petri_T> TT) для решения конфликта переходов. Метод findActiveT() формирует список ArrayList<Petri_T> ActiveT активных в текущий момент времени переходов. А метод DoConflikt(ArrayList<Petri_T> TT) осущест-

вляет выбор из списка активных переходов одного перехода. Выбор осуществляется на основании заданных приоритетов и вероятностей запуска переходов: сначала все переходы из списка активных переходов сортируются по значению приоритета, а затем из переходов с наивысшим значением приоритета выбирается один с заданной вероятностью запуска.

Метод Go(double time) выполняет имитацию сети Петри последовательным запуском метода NextEvent(), пока не достигнут момент времени time. Метод DoStatistica() содержит алгоритм сбора информации о среднем количестве маркеров в позициях и переходах сети Петри. Информация о дополнительных действиях, которые выполняются при выходе маркеров из переходов, содержится в методе DoT(). Этот метод задан в классе как пустой, но классы, которые наследуют класс Петри-имитатор, смогут использовать его для выполнения специфических действий, связанных с выполнением события. Например, событие «сдал экзамен» модели учебного процесса сопровождается выставлением оценки в журнале дисциплины. Было бы неверно моделировать оценки студента маркерами в позиции (хотя возможно). Это гораздо удобнее сделать в массиве оценок, что и описывается в методе DoT().

Метод NextEvent() описывается следующим Java-кодом:

```
public void NextEvent() {
    if (timeCurr<=timeMod) {
        eventMin.ActOut(ListP, timeCurr);
        if(eventMin.getBuffer()>0)   {
          boolean u = true;
          while (u==true) {
              eventMin.minEvent();
              if (eventMin.getMinTime()==timeCurr)
                    eventMin.ActOut(ListP,timeCurr);
              else   u=false;
           }
        }
     }
     ArrayList<Petri_T> ActiveT = new ArrayList<Petri_T>();
    ActiveT=this.findActiveT();
    if (ActiveT.size()==0&&IsBufferEmpty()==true) {
        STOP=true;
        timeMin = timeMod+1;
        timeCurr = timeMin;
     }
     else {
        STOP=false;
        while(ActiveT.size()>0)   {
        this.DoConflikt(ActiveT).ActIn(ListP, timeCurr);
            ActiveT=this.findActiveT();
        }
        this.EventMin();
     }
  }
```

Класс Петри-имитатор содержит все необходимые функции для выполнения имитации заданной сети Петри. Существование метода DoT() обеспечивает определенную гибкость моделирования систем – действия, не описывающие или не удобные для моделирования сетью Петри, но связанные с событиями сети Петри, исследователь может задать в этом методе. Таким образом, динамика функционирования модели описывается сетью

Петри. Все, что не связано с динамикой и неудобно для представления сетью Петри, помещается исследователем в метод DoT().

3. Создание классов Петри-объектов

В [7] введено понятие Петри-объекта (PetriObj) как объекта, являющегося наследником объекта Петри-имитатор (PetriSim): $\text{PetriObj} \xrightarrow{inherit} \text{PetriSim}$. Применение механизма наследования обеспечивает воссоздание всех полей и методов объекта-родителя в объекте-наследнике.

Конструирование сетей Петри-объектов удобно производить с помощью библиотеки сетей Петри, например, организованной в виде абстрактного класса NetLibrary. Такой подход обеспечивает возможность использования одной и той же функции из класса NetLibrary для создания сетей Петри множества однотипных объектов, что, в свою очередь, гарантирует однотипность обращения к позициям и переходам таких объектов. На данный момент эта библиотека содержит классы Петри-объектов, необходимые для воссоздания учебного процесса вуза [8]: Дисциплина, Студент, Преподаватель, Расписание, Деканат, Контроль задолженностей. Например, класс Петри-объектов Дисциплина создается следующим образом:

public class Discipline_Sim extends Petri_Sim {…}.

С помощью конструктора этого класса создаются дисциплины из учебного плана специальности и добавляются в список дисциплин:

```
ListDisc.add (new Discipline_Sim ("Моделювання систем",
    NetLibrary.CreateNetDiscipline (2, 4, 4, 2, timeMod, 2, 2), timeMod, 0.2, 0.3, 0.5)).
```

Либо в два шага:

```
Petri_net controlNet = NetLibrary.CreateNetControlVisit(periodControlVisit, timeMod);
ControlVisit_Sim ControlVisit = new ControlVisit_Sim(limit,controlNet, timeMod).
```

Тестирование Петри-объектов выполняется имитацией на различных интервалах времени при различных допустимых значениях начальных маркировок и различных допустимых значениях параметров.

4. Структура Петри-объектной модели

Пример диаграммы классов Петри-объектной модели представлен на рис. 2. Классы Петри-объектов $Sim_1, Sim_2, ...Sim_m$ являются классами объектов в терминах объектно-ориентированного моделирования и могут быть как прямыми наследниками класса объектов PetriSim, так и наследниками, или агрегаторами, других классов Петри-объектов.

Связи Петри-объектов между собой осуществляются двумя способами (рис. 3):

1) с помощью общих позиций (общая позиция является позицией сети Петри нескольких различных Петри-объектов):

$$SimOne.getNet().getListP()[k] = SimOther.getNet().getListP()[m]; \quad (1)$$

2) с помощью инициализации событий (из перехода сети Петри-объекта O_N при каждом выходе маркеров из перехода передаются маркеры в позицию сети Петри-объекта O_J в заданном количестве $w_{T,P}$ в момент времени, соответствующий моменту выхода маркеров из перехода):

$$SimOther.getNet().getListP()[m].setMark(n). \quad (2)$$

Алгоритмически существование общей позиции обеспечивается совпадением адресов памяти, где хранятся значения маркировок соответствующих позиций Петри-объектов. Инициализация необходимых событий задается в методе DoT().

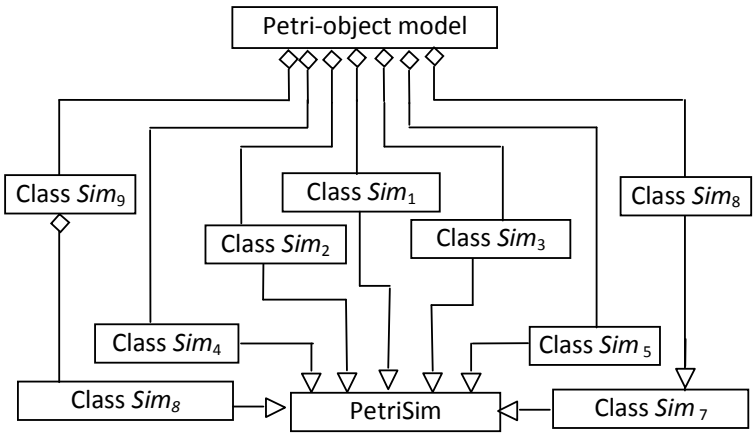

Рис. 6. Пример диаграммы классов Петри-объектной модели системы

5. Алгоритм имитации Петри-объектной модели

Как следует из диаграммы классов (рис. 6), Петри-объектная модель является результатом агрегирования Петри-объектов, динамика каждого из которых описывается сетью Петри и воспроизводится с помощью методов класса PetriSim. Отсюда алгоритм имитации Петри-объектной модели может быть построен на основе имитации ее Петри-объектов. Теоретически доказано, что Петри-объектная модель описывается сетью Петри, составленной из сетей ее Петри-объектов [6].

С алгоритмической точки зрения разбиение на Петри-объекты позволяет значительно сократить количество элементарных операций, необходимых для осуществления преобразования сети Петри. Преобразование D^+ сетей Петри-объектов следует фактически производить только для тех объектов, для которых момент выхода маркеров из переходов совпадает с текущим моментом времени. Преобразование $\left(D^-\right)^m$, осуществляемое для каждого Петри-объекта, потребует фактического выполнения m_J-кратного $\left(m_J \le m\right)$ входа маркеров в переходы до достижения состояния, при котором ни один из переходов Петри-объекта не запускается. Так как $m_J < m$ для большинства Петри-объектов и еще для большего количества Петри-объектов $m_J = 0$ (так как для них время выхода маркеров не совпадает с текущим моментом времени, и не изменилась их маркировка в текущий момент времени), то фактическое количество входов маркеров в переходы отдельных Петри-объектов значительно меньше количества m входов маркеров в переходы Петри-модели.

Алгоритм имитации Петри-объектной модели состоит из следующих действий:

◦ Формировать список Петри-объектов (список ListObj).

◦ Осуществить первоначальный вход маркеров в переходы для всех Петри-объектов (метод Start()).

◦ Пока не исчерпано время моделирования $(t < timeMod)$:

- продвинуть время в момент ближайшего события;
- осуществить статистические расчеты (метод DoStatistica());
- для всех Петри-объектов:
  - запомнить текущий момент времени (метод changeTimeCurr());
- для всех Петри-объектов, момент выхода из переходов которых совпадает с текущим моментом времени:
  - осуществить выполнение действий, записанных в методе Do_T();
  - выполнить преобразование состояния (метод NextEvent());
- для всех Петри-объектов:
  - выполнить вход маркеров в переходы (метод Start()).

◦ Вывести результаты моделирования.

Формирование списка Петри-объектов для модели учебного процесса, например, описывается следующим фрагментом Java-кода:

```
for (Student_Sim e:ListStudent)  ListObj.add(e);
```

```
for (Teacher_Sim e:Teachers)     ListObj.add(e);
for (Discipline_Sim e:ListDisc)  ListObj.add(e);
ListObj.add(ControlDebts);
ListObj.add(FITIS);
ListObj.add(Shedule);
```

Выполнение имитации Петри-объектной модели описывается следующим Java-кодом:

```
for (Petri_Sim e:ListObj) e.Start();
while(t<timeMod) {
    min_t =  ListObj.get(0).getTimeMin();
    for(Petri_Sim e:ListObj)
       if( e.getTimeMin()<min_t)    min_t = e.getTimeMin();
    for(Petri_Sim e:ListObj)
            e.DoStatistica(min_t -t);
    t = min_t;
    for(Petri_Sim e:ListObj)
            e.changeTimeCurr(t);
    for(Petri_Sim e: ListObj)
          if (t==e.getTimeMin ())   {
             e.Do_T();
             e.NextEvent();
          }
    for(Petri_Sim e:ListObj)
            e. Start();
}
```

Заметим, что в случае, если Петри-объекты имеют общие позиции, являющиеся для их переходов входными, то возможно возникновение конфликта между переходами, имеющими общую входную позицию Петри-объектов. В этом случае следует сначала выбрать объекты с наибольшим приоритетом, а затем выбрать из них случайным образом один.

Алгоритм имитации Петри-объектной модели с разрешением конфликта объектов состоит из следующих действий:

◦ Формировать список Петри-объектов (список ListObj).

◦ Осуществить первоначальный вход маркеров в переходы для всех Петри-объектов (метод Start()).

◦ Создать список конфликтных Петри-объектов (список K).

◦ Пока не исчерпано время моделирования $(t < timeMod)$:

- очистить список конфликтных Петри-объектов;
- продвинуть время в момент ближайшего события;
- определить список конфликтных Петри-объектов и решить конфликт;
- осуществить статистические расчеты (метод DoStatistica());
- для всех Петри-объектов:
  - запомнить текущий момент времени (метод changeTimeCurr());

  для Петри-объекта, выигравшего конфликт:
  - осуществить выполнение действий, записанных в методе Do_T();
  - выполнить преобразование состояния (метод NextEvent());

  для всех Петри-объектов:
  - выполнить вход маркеров в переходы (метод Start()).

◦ Вывести результаты моделирования.

Приведем фрагмент Java-кода, описывающего выбор одного Петри-объекта из множества конфликтных объектов:

```
ArrayList<Petri_Sim> K = new ArrayList<Petri_Sim>();
Random r = new Random();
…
while(t<timeMod) {
K.clear();
…
for(Petri_Sim e: ListObj)
    if (t==e.getTimeMin())   K.add(e);
int num;
int max;
if(K.size()>1)
 {max=K.size();
  SortObj(K);
for(int i=1; i<K.size(); i++) {
     if(K.get(i).getPriority()<K.get(i-1).getPriority()) {
        max=i-1;
        break;
     }
  }
if (max==0)  num=0;
  else    num=r.nextInt(max);
 } else
    num=0;
for(int i=0; i<ListObj.size(); i++) {
    if (ListObj.get(i).getName().equalsIgnoreCase(K.get(num).getName())) {
       ListObj.get(i).Do_T();
       ListObj.get(i).NextEvent();
    }
 }
….
}
```

Построенный алгоритм имитации Петри-объектной модели состоит из максимально формализованных действий, что способствует минимальному количеству ошибок при его реализации. Поскольку элементом Петри-объектной модели является переход сети Петри, то сложные взаимосвязи событий модели продумываются исследователем еще на этапе формализации Петри-объектов сетью Петри, оставляя алгоритму имитации только воссоздание правильно упорядоченной во времени последовательности событий.

6. Выводы

В результате научного исследования:

– создана библиотека Java-классов для реализации стохастической временной сети Петри с конфликтными и многоканальными переходами, с информационными связями;

– разработан алгоритм имитации Петри-объектной модели системы;

– создана библиотека Java-классов для реализации Петри-объектного моделирования систем.

Моделирование системы с помощью Петри-объектов позволяет исследователю сосредоточиться на составлении сетей Петри элементов системы. При этом сеть Петри-объекта отображает поведенческие свойства элемента системы. Отладка Петри-объекта

может быть выполнена до объединения в систему и не сложна, если Петри-объект достаточно простой. Только после отладки Петри-объектов, представляющих элементы системы, выполняется конструирование модели системы.

В отличие от известных алгоритмов имитации, алгоритм имитации Петри-объектной модели системы основывается на формализованном описании динамики системы сетью Петри и позволяет выполнять имитацию систем с очень большим количеством элементов. Разбиение модели на Петри-объекты значительно уменьшает затраты труда на алгоритмическую реализацию модели системы.

СПИСОК ЛИТЕРАТУРЫ

1. Кельтон В. Имитационное моделирование. Классика CS / В. Кельтон, А. Лоу. – [3-е изд.]. – СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.
2. University of Hamburg. Petri Nets Tools Database Quick Overview [Електронний ресурс]. – Режим доступу: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html.
3. Murata T. Petri Nets: Properties, Analysis and Applications / T. Murata // Proc. of IEEE. – 1989. – Vol. 77, N 4. – P. 541 – 580.
4. Зайцев Д.А. Инварианты временных сетей Петри / Д.А. Зайцев // Кибернетика и системный анализ. – 2004. – № 2. – С. 92 – 106.
5. Стеценко І.В. Моделювання систем: навч. посіб. / Стеценко І.В. – Черкаси: ЧДТУ, 2010. – 399 с.
6. Стеценко И.В. Теоретические основы Петри-объектного моделирования систем / И.В. Стеценко // Математичні машини і системи. – 2011. – № 4. – С. 136 – 148.
7. Стеценко І.В. Об’єктно-орієнтоване моделювання систем з використанням мереж Петрі / І.В. Стеценко // Вісник Черкаського державного технологічного університету. – 2011. – № 2. – С. 3 – 9.
8. Стеценко І.В. Імітаційне моделювання системи управління навчальним процесом ВНЗ з використанням об’єктно-орієнтованого підходу / І.В. Стеценко // Математичні машини і системи. – 2011. – № 2. – С. 162 – 170.
9. Хорстман К. Java 2. Библиотека профессионала / К. Хорстман, Г. Корнелл. – М.: Изд. дом «Вильямс», 2007. – Т. 1: Основы. – 896 с.

Стаття надійшла до редакції 14.09.2011