Зображення, що містить текст, знімок екрана, Шрифт, лист

Автоматично згенерований опис

1 – 20/20

2 – 15/20 (треба були більше саме описи елементів/компонентів цих ПЗ, але так як я їх назвала, вона все одно зарахувала)

3 – 30/30

4 – 25/30 (варто було відкинути з її коду все, що зайве, так як тут лише один об’єкт мережі, а не кілька)

**ВИКОНАННЯ**

**1. Математичний опис стохастичної мережі Петрі.**

Стохастична мережа Петрі є узагальненням класичної мережі Петрі, де події (перехід) мають імовірнісні характеристики.

Математично мережа Петрі описується наступним чином:



Основні елементи математичного опису наступні:

* **Множини:**

1. Множина позицій:



1. Множина переходів:



1. Множина дуг , що визначає зв'язки між позиціями і переходами:



1. Кратності дуг:



1. Пріоритети та ймовірності запуску переходів:



1. Часові затримки в переходах:



1. Множини вхідних та множина вихідних позицій переходу Т:

 

1. Множини вхідних і множина вихідних переходів позиції Р:

 

* **Стани:**

***Стан стохастичної мережі Петрі в момент часу t:***



1. Стан позицій:



1. Стан переходів:



* 1. Стан переходу *Т*:

,

де q – номер запланованої події виходу маркерів з переходу в момент часу t

або



– якщо найближчим часом не очікується вихід маркерів з переходу

Таким чином ***стан стохастичної мережі Петрі*** записується:



* **Визначення моментів:**

1. Момент запланованої найближчої події для переходу Т на поточний момент часу:



1. Момент запланованої найближчої події для мережі Петрі на поточний момент часу:



1. Визначення моменту найближчої події для мережі Петрі на поточний момент часу:



* **Змінювання стану:**

***Змінювання стану мережі Петрі в момент часу tn:***



* **Вихід маркерів з переходів:**

Зображення, що містить текст, Шрифт, знімок екрана, документ

Автоматично згенерований опис

* **Вхід маркерів в переходи:**

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, почерк, знімок екрана

Автоматично згенерований опис

* Багатократний вхід маркерів в переходи:

Зображення, що містить текст, почерк, Шрифт, каліграфія

Автоматично згенерований опис

Зображення, що містить текст, Шрифт, ряд, почерк

Автоматично згенерований опис

Рівняння станів стохастичної мережі Петрі з часовими затримками, з конфліктними та багатоканальними переходами має наступний вигляд:



**2. Загальні складові сучасного програмного забезпечення з імітаційного моделювання систем.**

***Варто було додати ось це, що на скріншоті, а не так як я надала нижче:***

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Сучасні програмні засоби для імітаційного моделювання мають такі основні компоненти:

* Інтерфейс користувача (інтуїтивно зрозумілий інтерфейс для побудови моделей, інструменти для візуалізації процесів та отриманих результатів).
* Алгоритмічне ядро (реалізовані математичні методи моделювання та генератори випадкових чисел).
* Бібліотеки компонентів (набір готових елементів моделей, як от черги, події, процеси і т.д.).
* Системи збору та обробки даних (механізми збору статистичних даних та інструменти для аналізу результатів).
* Підтримка інтеграції (можливість обміну даними з іншими системами, як от базами даних, CAD-програмами).

При моделюванні систем зазвичай використовують наступні програмні засоби:

* Для ***неперервних моделей***:
  + Matlab

*Використовується для моделювання, аналізу даних, машинного навчання, побудови графіків, а також розробки систем керування і автоматизації.*

* + Simulink

*Моделювання безперервних систем, кіберфізичних систем, систем керування.*

* Для ***дискретно-подійних моделей***:
  + VisSim
  + GPSS

*Дискретно-подійне моделювання простих систем.*

* + CPNTools

*Програмне забезпечення для моделювання, симуляції та аналізу систем за допомогою кольорових мереж Петрі.*

* + AnyLogic

*Гібридне моделювання (дискретно-подійне, агент-орієнтоване, системна динаміка).*

* + ProModel

*Оптимізація та моделювання виробничих, сервісних і логістичних систем.*

* + Simio

*Моделювання виробничих і логістичних процесів із підтримкою Industry 4.0.*

* + Arena Simulation Software

*Моделювання дискретно-подійних систем, таких як виробничі лінії, системи обслуговування клієнтів, логістика.*

Але якщо необхідно реалізувати складні системи, які важко адаптувати до готових програмних засобів або які зручніше реалізувати власноруч, використовують мови програмування, такі як Java, Python і т.д.

***Додала для захисту:***

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

Зображення, що містить текст, знімок екрана, Шрифт, документ

Автоматично згенерований опис

**3. Складіть мережу Петрі, що описує функціонування системи.**

Зображення, що містить текст, почерк, схема, ескіз

Автоматично згенерований опис

**4. Складіть алгоритм імітації системи.**

***Схема мережі Петрі:***

Зображення, що містить схема, велосипед

Автоматично згенерований опис

***Лістинг коду:***

Метод для запуску симуляції протягом заданого часу:

*/\*\*  
 \* Simulating from zero time until the time equal time modeling.<br>  
 \* Simulation protocol is printed on console.  
 \*  
 \** ***@param*** *timeModeling time modeling  
 \*   
 \*/*public void go(double timeModeling) {  
 double min;  
 this.setSimulationTime(timeModeling);   
 this.setCurrentTime(0.0);   
   
 getListObj().sort(PetriSim.*getComparatorByPriority*()); //edited 9.11.2015, 12.10.2017  
 for (PetriSim e : getListObj()) { //edited 9.11.2015, 18.07.2018  
 e.input();  
 }  
 if (isProtocolPrint() == true) {  
 for (PetriSim e : getListObj()) {  
 e.printMark();  
 }  
 }  
 ArrayList<PetriSim> conflictObj = new ArrayList<>();  
 Random r = new Random();  
  
 while (this.getCurrentTime() < this.getSimulationTime()) { // edited 18.07.2018  
  
 conflictObj.clear();  
  
 min = getListObj().get(0).getTimeMin(); //пошук найближчої події  
  
 for (PetriSim e : getListObj()) {  
 if (e.getTimeMin() < min) {  
 min = e.getTimeMin();  
 }  
 }  
 /\* if(min\_t<t){ // added 24.06.2013 !!!!Подумать...при отрицательных задержках висит!!!!  
 JOptionPane.showMessageDialog(null, "Negative time delay was generated! Check parameters, please/");  
 return;  
   
 }\*/  
 if (isStatistics() == true) {  
 for (PetriSim e : getListObj()) {  
 if (min > 0) {  
 if(min<this.getSimulationTime())  
 e.doStatistics((min - this.getCurrentTime()) / min); //статистика за час "дельта т", для спільних позицій потрібно статистику збирати тільки один раз!!!  
 else  
 e.doStatistics((this.getSimulationTime() - this.getCurrentTime()) / this.getSimulationTime());   
 }  
  
 }  
 }  
  
 this.setCurrentTime(min); // просування часу //3.12.2015  
   
 if (isProtocolPrint() == true) {  
 System.*out*.println(" Time progress: time = " + this.getCurrentTime() + "\n");  
 }  
 if (this.getCurrentTime() <= this.getSimulationTime()) {  
  
 for (PetriSim sim : getListObj()) {  
 if (this.getCurrentTime() == sim.getTimeMin()) // розв'язання конфлікту об'єктів рівноймовірнісним способом  
 {  
 conflictObj.add(sim); //список конфліктних обєктів  
 }  
 }  
 int num;  
 int max;  
 if (isProtocolPrint() == true) {  
 System.*out*.println(" List of conflicting objects " + "\n");  
 for (int ii = 0; ii < conflictObj.size(); ii++) {  
 System.*out*.println(" K [ " + ii + " ] = " + conflictObj.get(ii).getName() + "\n");  
 }  
 }  
  
 if (conflictObj.size() > 1) { //вибір об'єкта, що запускається  
 max = conflictObj.size();  
 conflictObj.sort(PetriSim.*getComparatorByPriority*());  
 for (int i = 1; i < conflictObj.size(); i++) { //System.out.println(" "+conflictObj.get(i).getPriority()+" "+conflictObj.get(i-1).getPriority());  
 if (conflictObj.get(i).getPriority() < conflictObj.get(i - 1).getPriority()) {  
 max = i - 1;  
 //System.out.println("max= "+max);  
 break;  
 }  
  
 }  
 if (max == 0) {  
 num = 0;  
 } else {  
 num = r.nextInt(max);  
 }  
 } else {  
 num = 0;  
 }  
  
 if (isProtocolPrint() == true) {  
 System.*out*.println(" Selected object " + conflictObj.get(num).getName() + "\n" + " NextEvent " + "\n");  
 }  
  
 for (PetriSim sim: getListObj()) {  
 if (sim.getNumObj() == conflictObj.get(num).getNumObj()) {  
 if (isProtocolPrint() == true) {  
 System.*out*.println(" time = " + this.getCurrentTime() + " Event '" + sim.getEventMin().getName() + "'\n"  
 + " is occuring for the object " + sim.getName() + "\n");  
 }  
 sim.doT();  
 sim.output(); // added by Inna 11.07.2018  
 }  
 }  
 if (isProtocolPrint() == true) {  
 System.*out*.println("Markers output:");  
 for (PetriSim sim : getListObj()) //ДРУК поточного маркірування  
 {  
 sim.printMark();  
 }  
 }  
   
 Collections.*shuffle*(getListObj()); // added by Inna 11.07.2018, need for correct functioning of Petri object's shared resource   
   
 getListObj().sort(PetriSim.*getComparatorByPriority*());  
   
 for (PetriSim e : getListObj()) {  
 //можливо змінились умови для інших обєктів  
 e.input(); //вхід маркерів в переходи Петрі-об'єкта  
  
 }  
 if (isProtocolPrint() == true) {  
 System.*out*.println("Markers input:");  
 for (PetriSim e : getListObj()){ //ДРУК поточного маркірування  
 e.printMark();  
 }  
 }  
 }  
 }  
 getListObj().sort(PetriSim.*getComparatorByNum*()); // return the initial order in the list for a correct output of the results (in SMO test)  
}

Метод, який визначає вхід маркерів у перехід:

*/\*\*  
 \* It does the transitions input markers  
 \*/*public void input() {//вхід маркерів в переходи Петрі-об'єкта  
  
 ArrayList<PetriT> activeT = this.findActiveT(); //формування списку активних переходів  
  
 if (activeT.isEmpty() && isBufferEmpty() == true) { //зупинка імітації за умови, що  
 //не має переходів, які запускаються,і не має маркерів у переходах  
 timeMin = Double.*MAX\_VALUE*;  
 //eventMin = null; // 19.07.2018 by Sasha animation  
 } else {  
 while (activeT.size() > 0) {//запуск переходів доки можливо  
  
 this.doConflikt(activeT).actIn(listP, this.getCurrentTime()); //розв'язання конфліктів  
 activeT = this.findActiveT(); //оновлення списку активних переходів  
 }  
 this.eventMin();//знайти найближчу подію та ії час  
 }  
}

Метод, який визначає вихід маркерів з переходу:

*/\*\*  
 \* It does the transitions output markers  
 \*/* public void output(){  
 if (this.getCurrentTime() <= this.getSimulationTime()) {  
 eventMin.actOut(listP, this.getCurrentTime());//здійснення події  
 if (eventMin.getBuffer() > 0) {  
 boolean u = true;  
 while (u == true) {  
 eventMin.minEvent();  
 if (eventMin.getMinTime() == this.getCurrentTime()) {  
 eventMin.actOut(listP,this.getCurrentTime());  
 } else {  
 u = false;  
 }  
 }  
  
 }  
 for (PetriT transition : listT) { //ВАЖЛИВО!!Вихід з усіх переходів, що час виходу маркерів == поточний момент час.  
   
 if (transition.getBuffer() > 0 && transition.getMinTime() == this.getCurrentTime()) {  
 transition.actOut(listP, this.getCurrentTime());//Вихід маркерів з переходу, що відповідає найближчому моменту часу  
 if (transition.getBuffer() > 0) {  
 boolean u = true;  
 while (u == true) {  
 transition.minEvent();  
 if (transition.getMinTime() == this.getCurrentTime()) {  
 transition.actOut(listP, this.getCurrentTime());  
 } else {  
 u = false;  
 }  
 }  
 }  
 }  
 }  
  
 }  
}

Метод для створення мережі Петрі:

public static PetriNet CreateNet() throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay {  
 ArrayList<PetriP> d\_P = new ArrayList<>();  
 ArrayList<PetriT> d\_T = new ArrayList<>();  
 ArrayList<ArcIn> d\_In = new ArrayList<>();  
 ArrayList<ArcOut> d\_Out = new ArrayList<>();  
 d\_P.add(new PetriP("P1",1));  
 d\_P.add(new PetriP("P2",0));  
 d\_P.add(new PetriP("P3",0));  
 d\_P.add(new PetriP("P4",0));  
 d\_P.add(new PetriP("P5",0));  
 d\_P.add(new PetriP("P7",0));  
 d\_P.add(new PetriP("P9",1));  
 d\_P.add(new PetriP("P10",1));  
 d\_P.add(new PetriP("P11",1));  
 d\_P.add(new PetriP("P12",1));  
 d\_P.add(new PetriP("P3",0));  
 d\_P.add(new PetriP("P4",0));  
 d\_T.add(new PetriT("Надходження запитів",2.0));  
 d\_T.get(0).setDistribution("unif", d\_T.get(0).getTimeServ());  
 d\_T.get(0).setParamDeviation(1.0);  
 d\_T.add(new PetriT("Попередня обробка запиту",2.0));  
 d\_T.add(new PetriT("Пошук інформації на комп1",3.0));  
 d\_T.get(2).setDistribution("unif", d\_T.get(2).getTimeServ());  
 d\_T.get(2).setParamDeviation(1.0);  
 d\_T.get(2).setProbability(0.75);  
 d\_T.add(new PetriT("Обмін даними між компами",1.0));  
 d\_T.get(3).setProbability(0.25);  
 d\_T.add(new PetriT("Пошук інформації на комп2",3.0));  
 d\_T.get(4).setDistribution("unif", d\_T.get(4).getTimeServ());  
 d\_T.get(4).setParamDeviation(1.0);  
 d\_T.add(new PetriT("Інф-ія не на місці",0.0));  
 d\_T.get(5).setProbability(0.25);  
 d\_T.add(new PetriT("Інф-ія на місці",0.0));  
 d\_T.get(6).setProbability(0.75);  
 d\_In.add(new ArcIn(d\_P.get(4),d\_T.get(6),1));  
 d\_In.add(new ArcIn(d\_P.get(0),d\_T.get(0),1));  
 d\_In.add(new ArcIn(d\_P.get(7),d\_T.get(2),1));  
 d\_In.add(new ArcIn(d\_P.get(11),d\_T.get(2),1));  
 d\_In.add(new ArcIn(d\_P.get(1),d\_T.get(1),1));  
 d\_In.add(new ArcIn(d\_P.get(6),d\_T.get(1),1));  
 d\_In.add(new ArcIn(d\_P.get(9),d\_T.get(3),1));  
 d\_In.add(new ArcIn(d\_P.get(10),d\_T.get(3),1));  
 d\_In.add(new ArcIn(d\_P.get(3),d\_T.get(4),1));  
 d\_In.add(new ArcIn(d\_P.get(8),d\_T.get(4),1));  
 d\_In.add(new ArcIn(d\_P.get(4),d\_T.get(5),1));  
 d\_Out.add(new ArcOut(d\_T.get(6),d\_P.get(11),1));  
 d\_Out.add(new ArcOut(d\_T.get(0),d\_P.get(0),1));  
 d\_Out.add(new ArcOut(d\_T.get(0),d\_P.get(1),1));  
 d\_Out.add(new ArcOut(d\_T.get(2),d\_P.get(2),1));  
 d\_Out.add(new ArcOut(d\_T.get(2),d\_P.get(7),1));  
 d\_Out.add(new ArcOut(d\_T.get(1),d\_P.get(4),1));  
 d\_Out.add(new ArcOut(d\_T.get(1),d\_P.get(6),1));  
 d\_Out.add(new ArcOut(d\_T.get(3),d\_P.get(3),1));  
 d\_Out.add(new ArcOut(d\_T.get(3),d\_P.get(9),1));  
 d\_Out.add(new ArcOut(d\_T.get(4),d\_P.get(5),1));  
 d\_Out.add(new ArcOut(d\_T.get(4),d\_P.get(8),1));  
 d\_Out.add(new ArcOut(d\_T.get(5),d\_P.get(10),1));  
 PetriNet d\_Net = new PetriNet("4",d\_P,d\_T,d\_In,d\_Out);  
 PetriP.*initNext*();  
 PetriT.*initNext*();  
 ArcIn.*initNext*();  
 ArcOut.*initNext*();  
  
 return d\_Net;  
}

Результат роботи: