

ЛЕКЦІЯ 11

Програмне забезпечення Петрі-об'єктного моделювання систем

Алгоритм імітації Петрі-об'єктної моделі

- Формувати список Петрі-об'єктів;
- Виконати перетворення $(D^-)^m$ (метод input());
- Доки не досягнутий момент завершення моделювання
 - Просунути час в момент найближчої події;
 - визначити список конфліктних об'єктів та вибрати об'єкт із списку конфліктних об'єктів;
 - для вибраного об'єкта виконати перетворення $(D^-)^m \circ D^+$ (методи output(), input(), doT());
 - для всіх інших об'єктів виконати перетворення $(D^-)^m$ (метод input());
- Вивести результати моделювання.

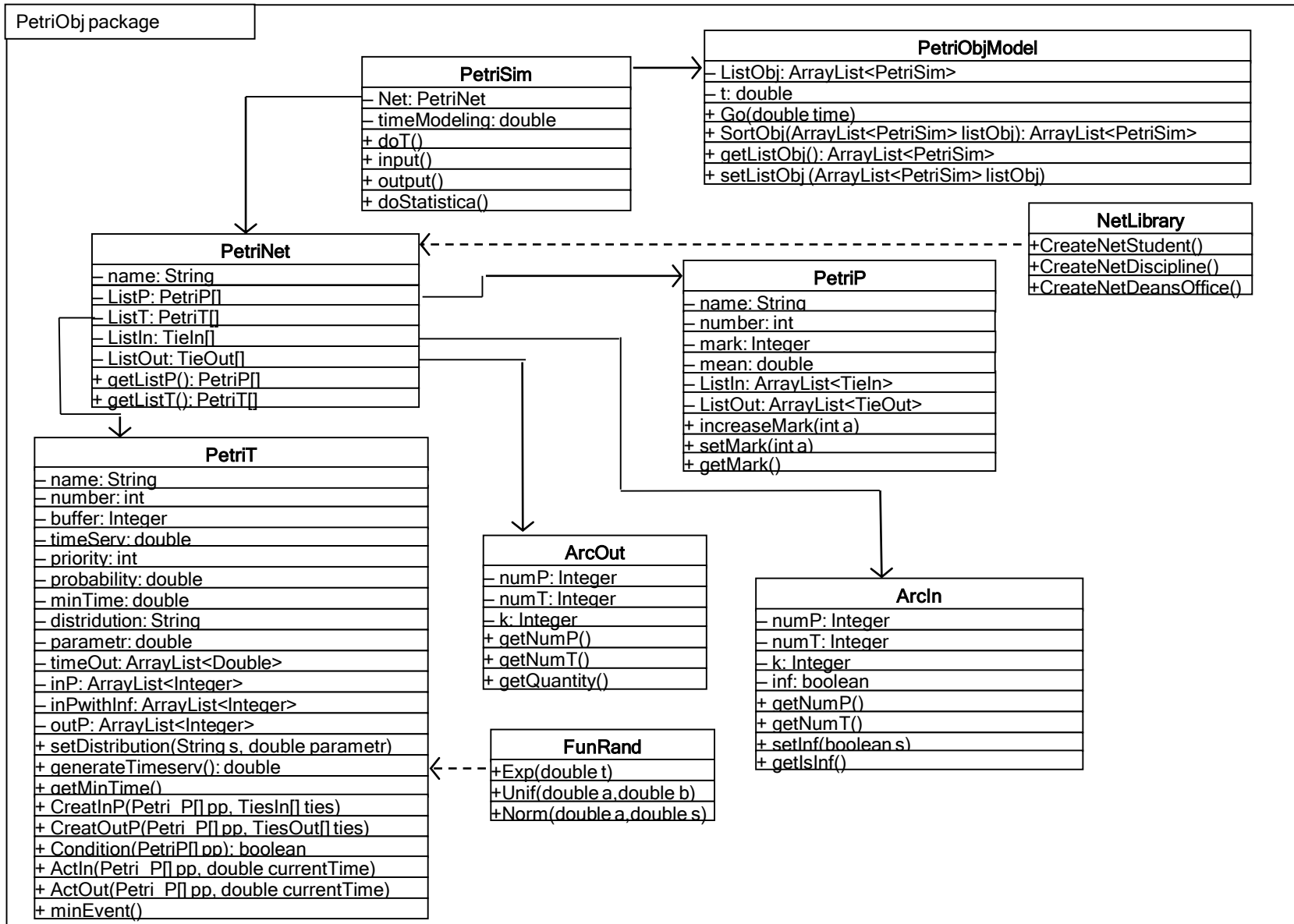
Анализ обчислювальної складності алгоритму:

$$O(|\mathbf{T}|^2 \cdot V \cdot timeMod \cdot \left(\underset{T \in \mathbf{T}}{mean} |T^\bullet| + V + V \cdot |\mathbf{T}| \cdot \underset{T \in \mathbf{T}}{mean} |\bullet T| + V^2 \cdot |\mathbf{T}| + V \cdot K^2 \right))$$

Середня кількість активних каналів переходів

Середня кількість конфліктних переходів

Бібліотека класів PetriObj <https://github.com/StetsenkoInna/PetriObjModelPaint>



Документація

| Here's a Blog About How to W... x Arcin x PetriSim x + | | | |
|---------------------------------------------------------------------------------------------|----------------|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File /Users/innastetsenko/PetriObjModelPaint-master/dist/javadoc/PetriObj/PetriSim.html ☆ | | | |
| All Methods | Static Methods | Instance Methods | Concrete Methods |
| Modifier and Type | | Method and Description | |
| PetriSim | | clone() | |
| PetriT | | doConflikt(java.util.ArrayList<PetriT> transitions) | This method solves conflict between transitions given in parametr transitions |
| void | | doStatistics() | Calculates mean value of quantity of markers in places and quantity of active channels of transitions |
| void | | doStatistics(double dt) | |
| void | | doT() | This method uses for describing other actions associated with transition markers output. Such as the output markers into the other Petri-object. The method is overridden in subclass. |
| void | | eventMin() | Determines the next event and its moment. |
| java.util.ArrayList<PetriT> | | findActiveT() | Finds the set of transitions for which the firing condition is true and sorts it on priority value |
| static java.util.Comparator<PetriSim> | | getComparatorByNum() | |
| static java.util.Comparator<PetriSim> | | getComparatorByPriority() | |
| double | | getCurrentTime() | |
| PetriT | | getEventMin() | |
| java.lang.String | | getId() | |
| java.util.ArrayList<PetriP> | | getListPositionsForStatistica() | |
| java.lang.String | | getName() | |
| PetriNet | | getNet() | |

Метод go() об'єкту PetriObjModel

```
public void go(double timeModeling) {  
    double min;  
    this.setSimulationTime(timeModeling);  
    this.setCurrentTime(0.0);  
  
    getListObj().sort(PetriSim.getComparatorByPriority());  
    for (PetriSim e : getListObj()) {  
        e.input();  
    }  
    ArrayList<PetriSim> conflictObj = new ArrayList<>();  
    Random r = new Random();  
  
    while (this.getCurrentTime() < this.getSimulationTime()) {  
        conflictObj.clear();  
        min = getListObj().get(0).getTimeMin();  
  
        for (PetriSim e : getListObj()) {  
            if (e.getTimeMin() < min) {  
                min = e.getTimeMin();  
            }  
        }  
    }  
}
```

Метод go() об'єкту PetriObjModel

```
if (isStatistics() == true) {  
    for (PetriSim e : getListObj()) {  
        if (min > 0) {  
            if(min<this.getSimulationTime())  
                e.doStatistics((min - this.getCurrentTime()) / min);  
            else  
                e.doStatistics(  
                    (this.getSimulationTime() - this.getCurrentTime()) /  
                    this.getSimulationTime());  
        }  
    }  
}  
this.setCurrentTime(min);
```

Метод go() об'єкту PetriObjModel

```
if (this.getCurrentTime() <= this.getSimulationTime()) {
    for (PetriSim sim : getListObj()) {
        if (this.getCurrentTime() == sim.getTimeMin()) {
            conflictObj.add(sim);
        }
    }
    int num;
    int max;
    if (conflictObj.size() > 1) { //вибір об'єкта, що запускається
        max = conflictObj.size();
        conflictObj.sort(PetriSim.getComparatorByPriority());
        for (int i = 1; i < conflictObj.size(); i++) {
            if (conflictObj.get(i).getPriority() <
                conflictObj.get(i - 1).getPriority()) {
                max = i - 1;
                break;
            }
        }
        if (max == 0) { .    num = 0; }
        else { .    num = r.nextInt(max); }
    } else { .    num = 0; }
```

Метод go() об'єкту PetriObjModel

```
    for (PetriSim sim: getListObj()) {
        if (sim.getNumObj() == conflictObj.get(num).getNumObj()) {
            sim.doT();
            sim.output();
        }
    }

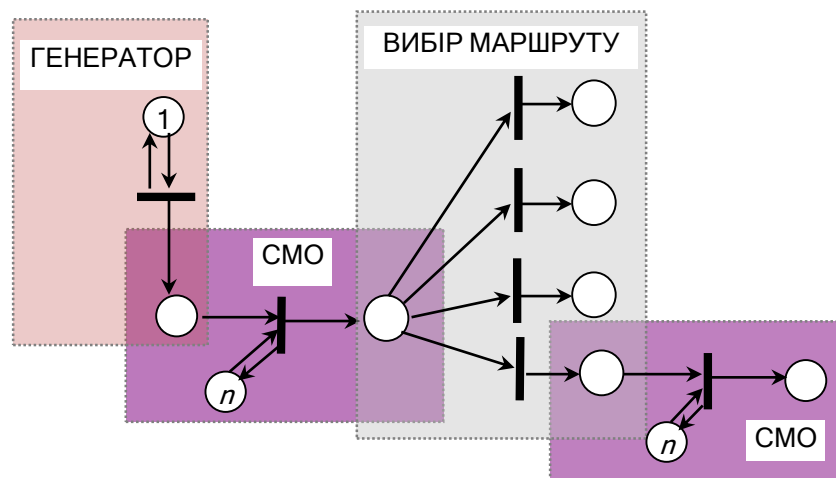
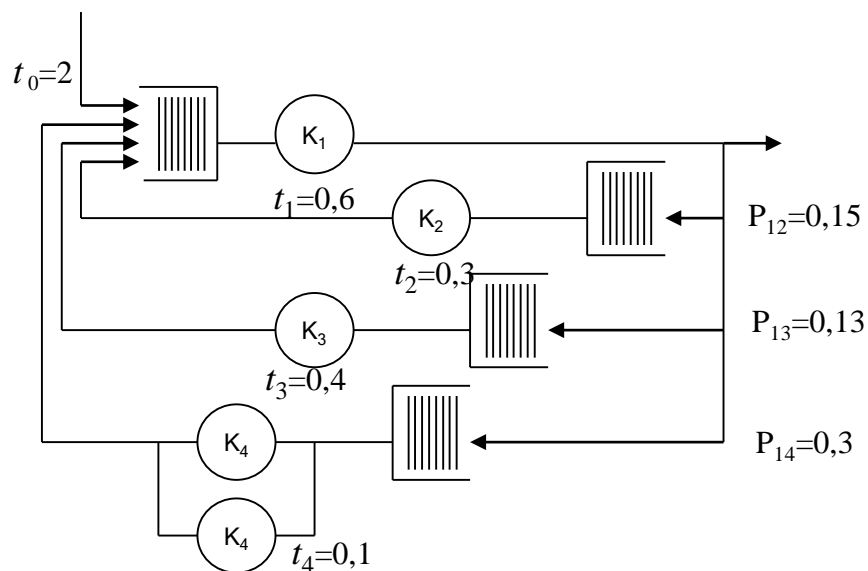
    Collections.shuffle(getListObj());

    getListObj().sort(PetriSim.getComparatorByPriority());

    for (PetriSim e : getListObj()) {
        e.input();
    }
}

getListObj().sort(PetriSim.getComparatorByNum());
}
```


Приклад розробки Петрі-об'єктної моделі з використанням бібліотеки PetriObjLib



Використання графічного редактора для розробки мережі Петрі

The screenshot displays the 'Discrete Event Simulation System' (DESS) Net Designer interface. The main window shows a Petri net diagram with three places (P1, P2, P3) and one transition (T1). Place P1 contains 0 tokens, P2 contains 0 tokens, and P3 contains 0 tokens. Transition T1 has a delay of $t=0.0$ and a bias of $b=0$. The diagram is connected by directed edges from P1 to T1 and from T1 to P2.

The interface includes a menu bar with 'File', 'Edit', 'Save', 'Animate', and 'Run'. Below the menu bar are three tabs: 'Net designer', 'Model designer', and 'Experiment designer'. The 'Net designer' tab is active, showing a toolbar with a circle icon and a diagonal line icon. A list of functions is visible on the left side of the Net designer tab, including 'CreateNetAdmin()', 'CreateNetConsistency(doub', 'CreateNetFriend()', 'CreateNetFriendUsingCores', 'CreateNetGenerator2(doub', 'CreateNetMalware()', 'CreateNetNewPool()', 'CreateNetTask(double a)', 'CreateNetTest3()', 'CreateNetTestInfArc()', 'CreateNetTestStatistics()', 'CreateNetThread3()', and 'CreateNetThreadStartAndEr'.

A 'Transition parameters' dialog box is open, showing the following settings:

- Basic parameters:** Name: T1
- Time delay:** Delay mean value: 0.0, Standard deviation: 0.0, Time delay distribution: null, Distribution (parameter name):
- Transition priority:** Priority value: 0, Priority (parameter name):
- Activation probability:** Probability value: 1.0

The dialog box has 'OK' and 'Cancel' buttons. The bottom status bar shows 'Net name: Untitled', 'Time start: 0', 'Time modeling: 1000', and 'Animation speed'.

Конструювання мережі Петрі

```
public static PetriNet CreateNetSMOwithoutQueue(int numChannel, double timeMean,
String name) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure{
    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",0));
    d_P.add(new PetriP("P2",numChannel));
    d_P.add(new PetriP("P3",0));
    d_T.add(new PetriT("T1",timeMean,Double.MAX_VALUE));
    d_T.get(0).setDistribution("exp", d_T.get(0).getTimeServ());
    d_T.get(0).setParamDeviation(0.0);
    d_In.add(new ArcIn(d_P.get(0),d_T.get(0),1));
    d_In.add(new ArcIn(d_P.get(1),d_T.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(0),d_P.get(2),1));
    PetriNet d_Net = new PetriNet("SMOwithoutQueue"+name,d_P,d_T,d_In,d_Out)
    PetriP.initNext();
    PetriT.initNext();
    ArcIn.initNext();
    ArcOut.initNext();
    return d_Net;
}
```

Конструювання Петрі-об'єктів

```
ArrayList<PetriSim> list = new ArrayList<>();  
    list.add(new PetriSim(NetLibrary.CreateNetGenerator(2.0)));  
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.6, "First")));  
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.3, "Second")));  
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.4, "Third")));  
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(2, 0.1, "Forth")));  
    list.add(new PetriSim(NetLibrary.CreateNetFork(0.15, 0.13, 0.3)));
```

Графічний редактор Петрі- об'єктної моделі

Конструювання Петрі-об'єктної моделі

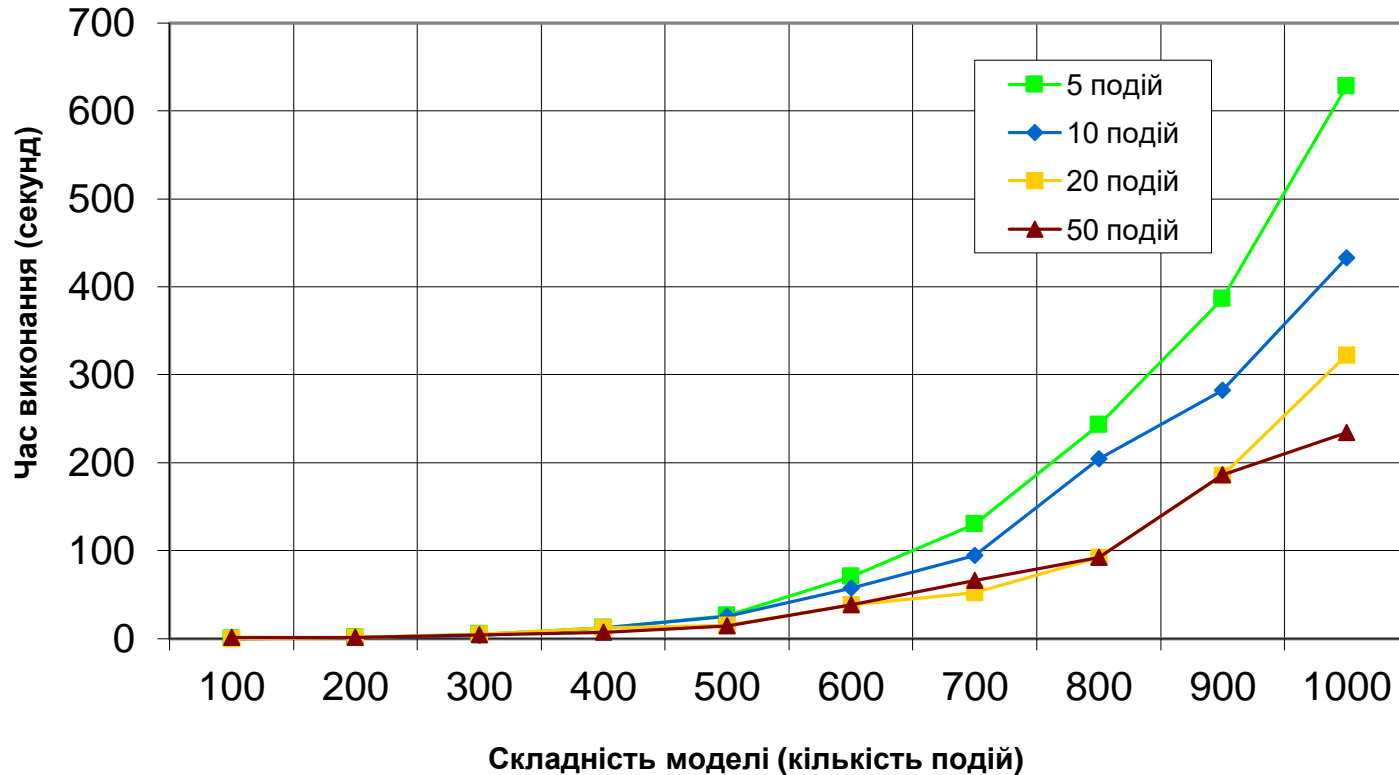
```
public static PetriObjModel getModel() throws ExceptionInvalidTimeDelay,
ExceptionInvalidNetStructure{
    ArrayList<PetriSim> list = new ArrayList<>();
    list.add(new PetriSim(NetLibrary.CreateNetGenerator(2.0)));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.6, "First")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.3, "Second")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(1, 0.4, "Third")));
    list.add(new PetriSim(NetLibrary.CreateNetSMOwithoutQueue(2, 0.1, "Forth")));
    list.add(new PetriSim(NetLibrary.CreateNetFork(0.15, 0.13, 0.3)));

    list.get(0).getNet().getListP()[1] = list.get(1).getNet().getListP()[0];
    list.get(1).getNet().getListP()[2] = list.get(5).getNet().getListP()[0];
    list.get(5).getNet().getListP()[1] = list.get(2).getNet().getListP()[0];
    list.get(5).getNet().getListP()[2] = list.get(3).getNet().getListP()[0];
    list.get(5).getNet().getListP()[3] = list.get(4).getNet().getListP()[0];
    list.get(2).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
    list.get(3).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
    list.get(4).getNet().getListP()[2] = list.get(1).getNet().getListP()[0];
    PetriObjModel model = new PetriObjModel(list);
    return model;
}
```

Точність результатів моделювання

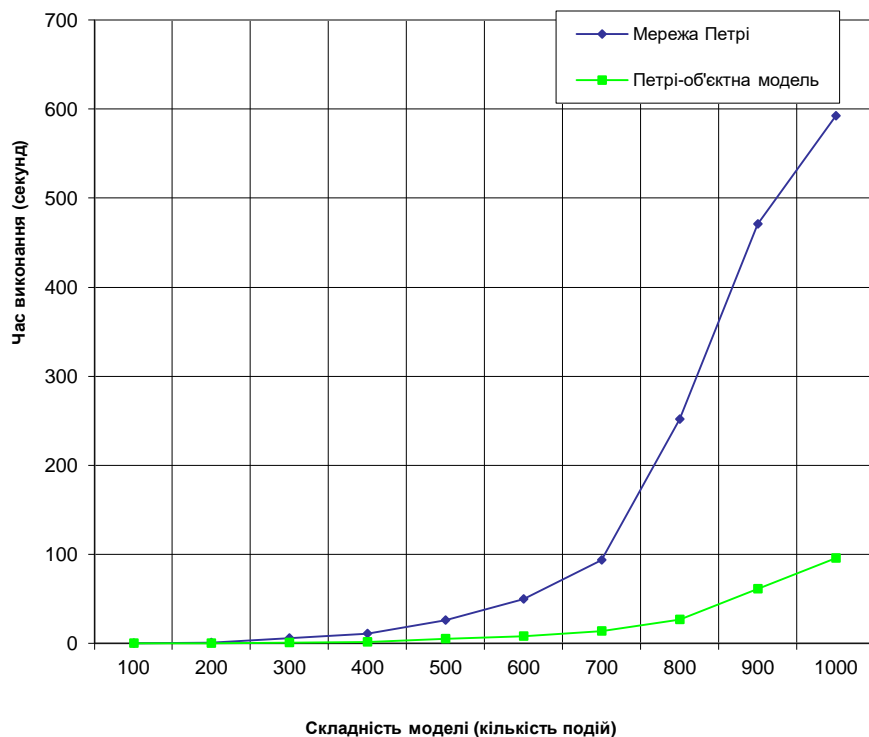
| Результати аналітичного моделювання | Результати Петрі-об'єктного моделювання |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Середня довжина черги СМО1 = 1,786 Середня довжина черги СМО2 = 0,003 Середня довжина черги СМО3 = 0,004 Середня довжина черги СМО4 = 0,00001 Середня зайнятість пристроїв СМО1 = 0,714 Середня зайнятість пристроїв СМО2 = 0,054 Середня зайнятість пристроїв СМО3 = 0,062 Середня зайнятість пристроїв СМО4 = 0,036 | Середня довжина черги СМО1 = 1,766 Середня довжина черги СМО2 = 0,0041 Середня довжина черги СМО3 = 0,0035 Середня довжина черги СМО4 = 0,00001 Середня зайнятість пристроїв СМО1 = 0,714 Середня зайнятість пристроїв СМО2 = 0,054 Середня зайнятість пристроїв СМО3 = 0,065 Середня зайнятість пристроїв СМО4 = 0,035 |

Дослідження ефективності алгоритму імітації Петрі-об'єктної моделі при зростанні складності об'єктів

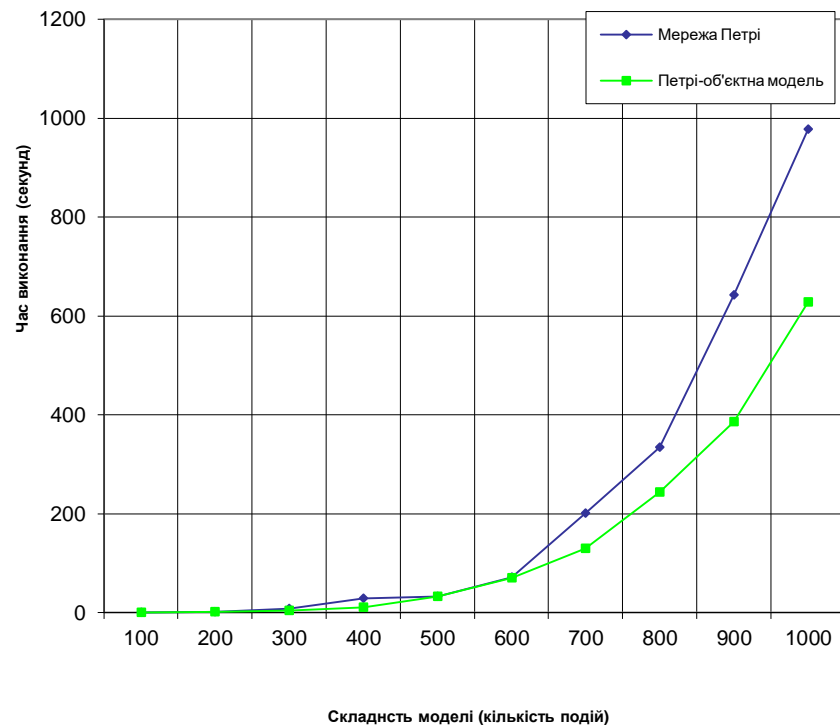


Дослідження ефективності алгоритму імітації Петрі-об'єктної моделі

а) переходи з детермінованими затримками



б) переходи зі стохастичними затримками



Петрі-об'єктна модель

- це засіб формального опису систем, який:

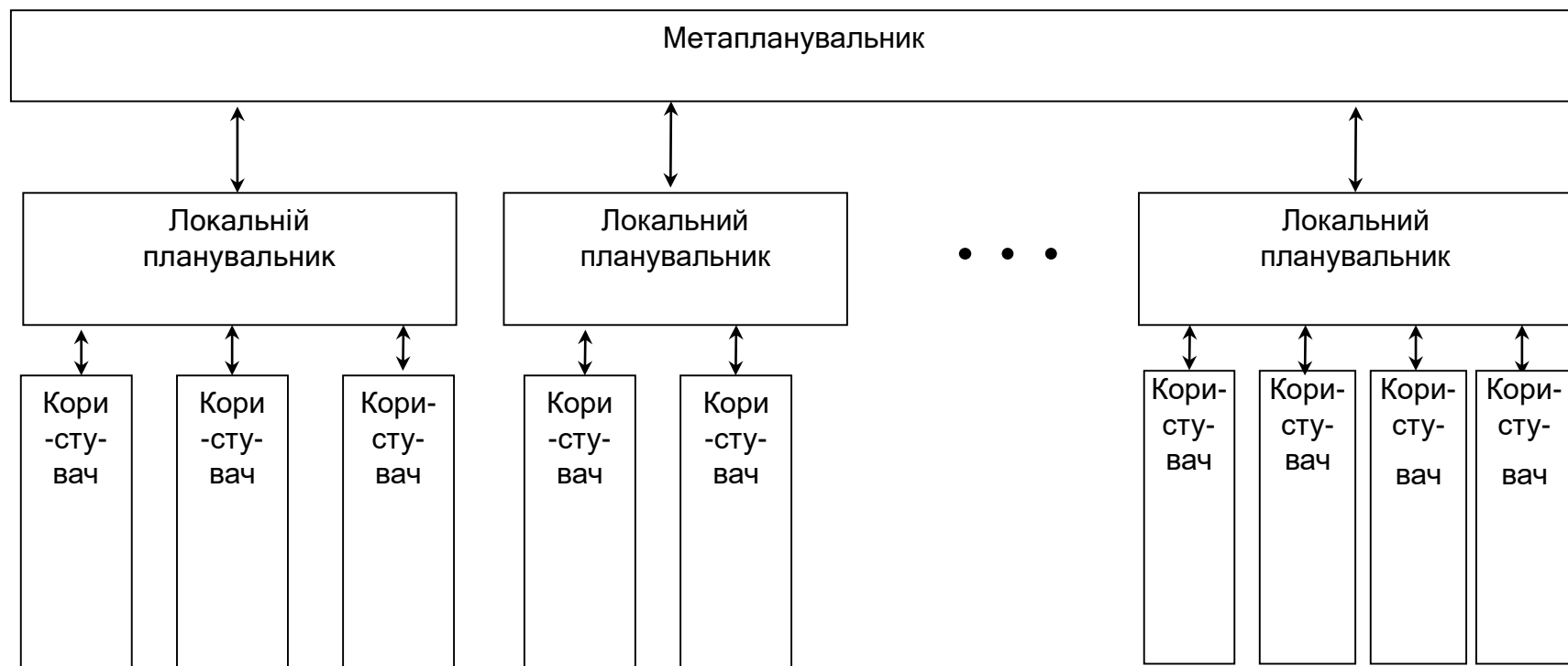
- 1) має математичний опис, а значить, має високу ступінь абстракції та найбільш формалізований опис алгоритму імітації;
- 2) допускає використання не тільки імітаційних методів дослідження, але й аналітичні методи;
- 3) Надає можливість створювати моделі великих і складних систем з найменшими витратами часу та зусиль;
- 4) ґрунтується на ОРеменно-стохастическій сеті Петрі, а значить, допускає найбільш деталізоване описання дискретно-подійних процесів функціонування;
- 5) ґрунтується на об'єктно-орієнтованій технології, а значить, допускає моделювання структури великих систем і сумісність з іншими інформаційними технологіями.

Практичне застосування Петрі-об'єктного моделювання

- Моделювання систем управління (навчальний процес, транспортні системи, грид-системи)
- Моделювання паралельних обчислень (багатопоточність, синхронізація, дослідження: дедлоків, конфліктів доступу до спільної пам'яті)
- Моделювання процесів управління організаціями та підприємствами (процесно-орієнтований підхід до управління)
- Розподілене моделювання (модель великої системи будується за участю колективу розробників). Це ідея і концепція, якої притримувались при розробці ООП
- Петрі-процесор (Петрі-машина) і «нова парадигма обчислень»

Петрі-об'єктна модель системи управління розподіленими обчислювальними ресурсами

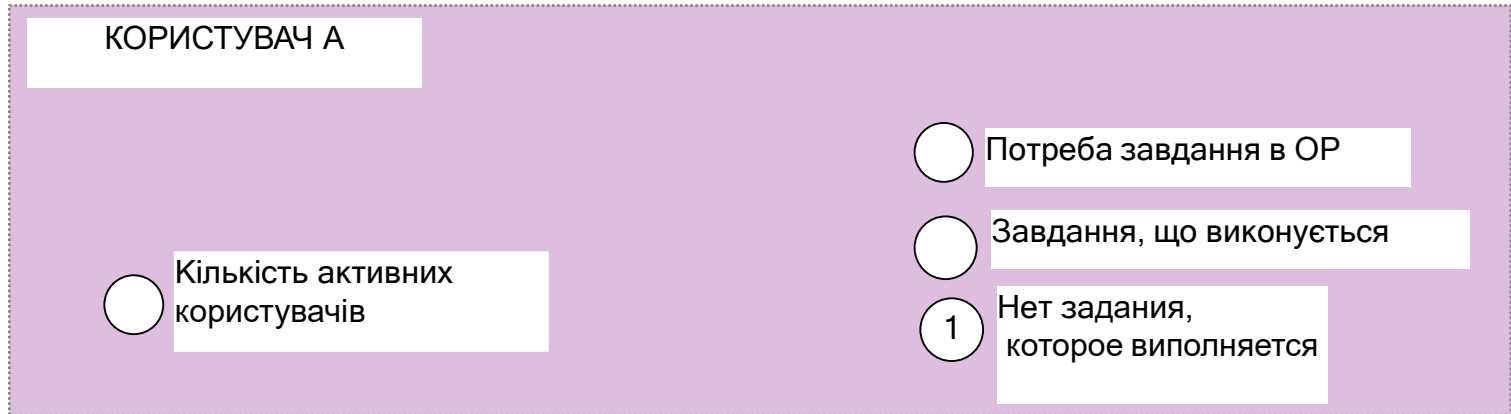
Архітектура двохрівневої грід-системи



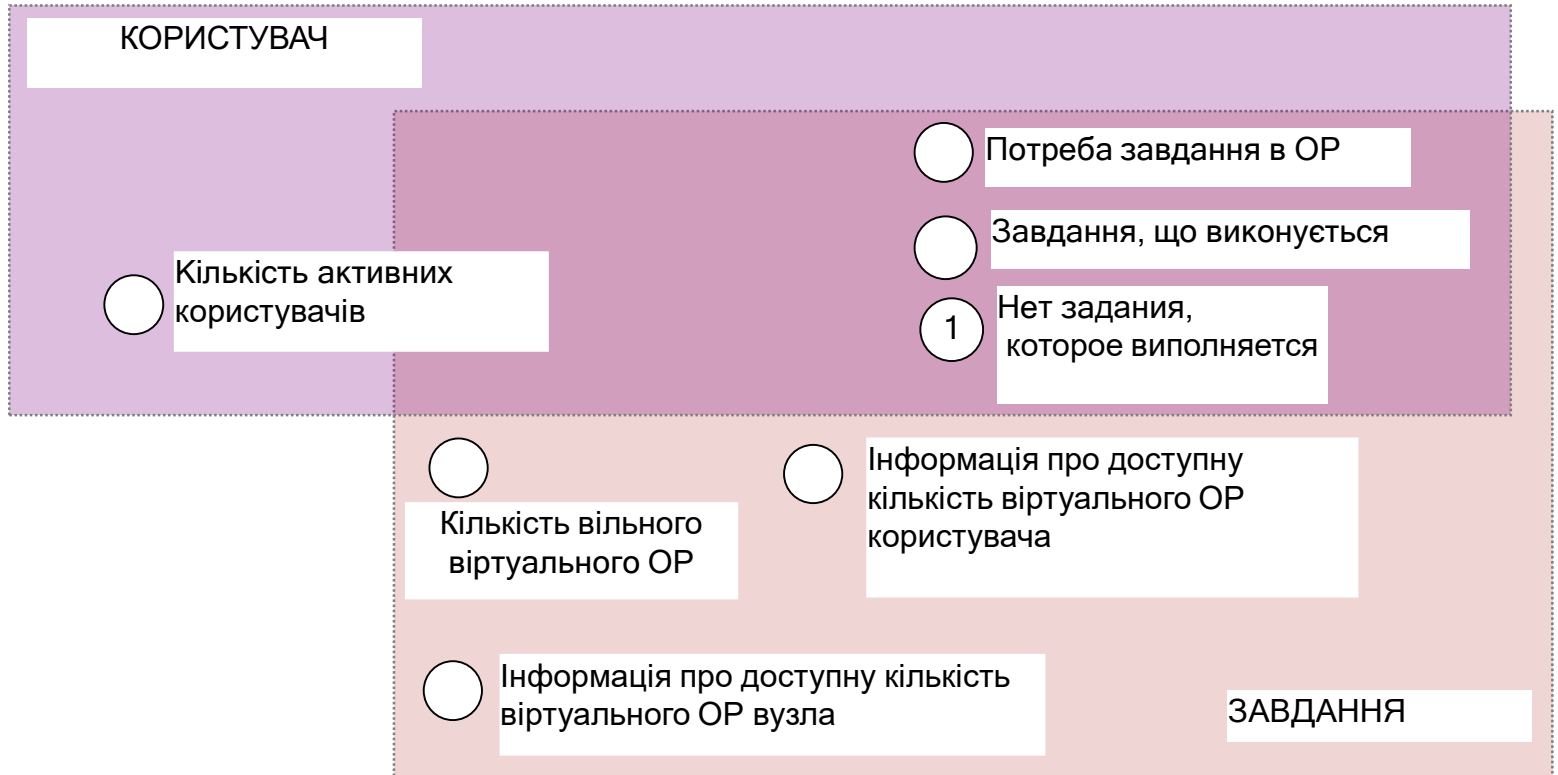
Частина доступного ресурсу:

$$x_i = \frac{p_i}{\sum_{i \in A} p_i} \quad \sum_i x_i = 1$$

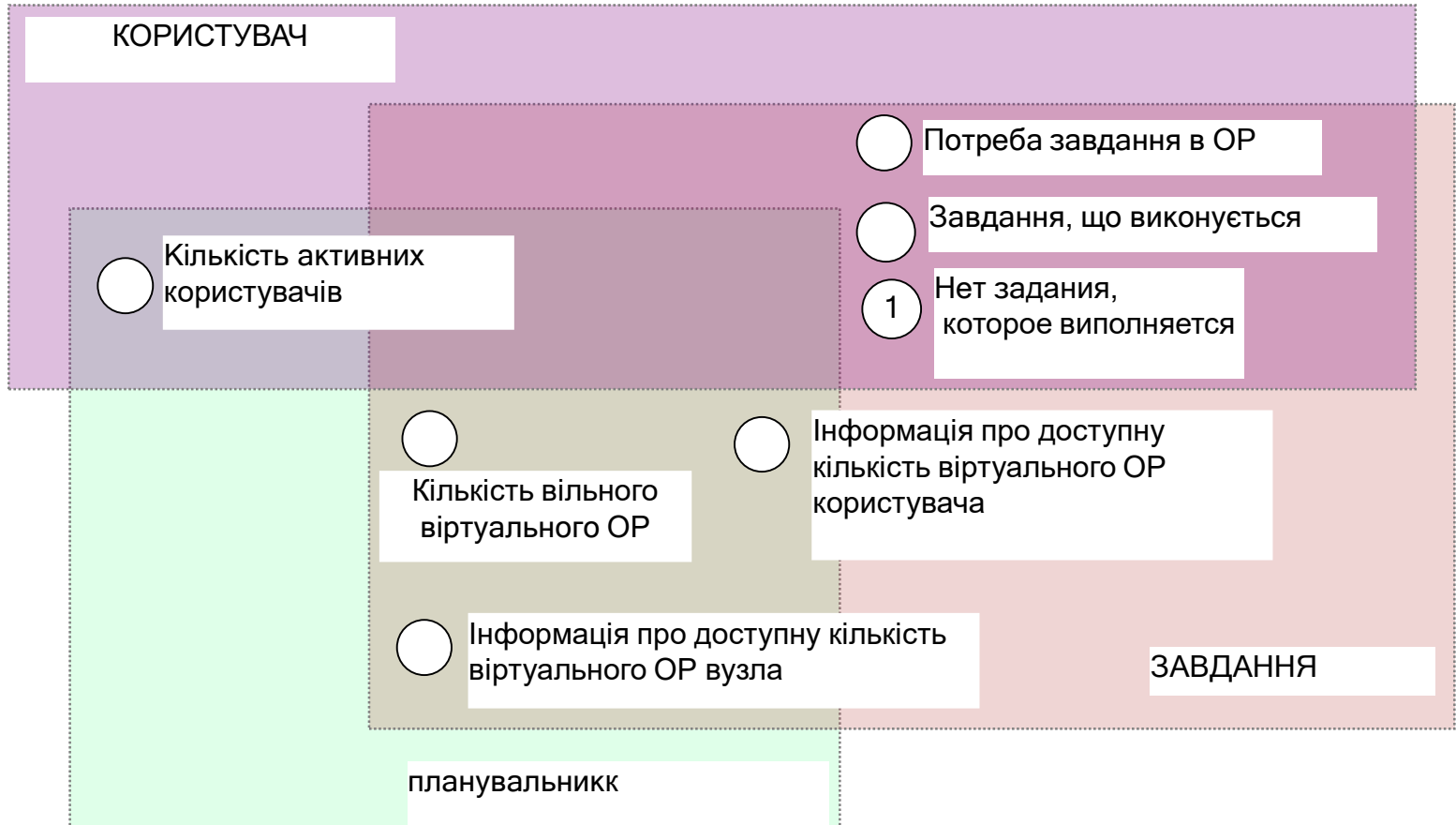
Петрі-об'єктна модель системи управління розподіленими обчислювальними ресурсами



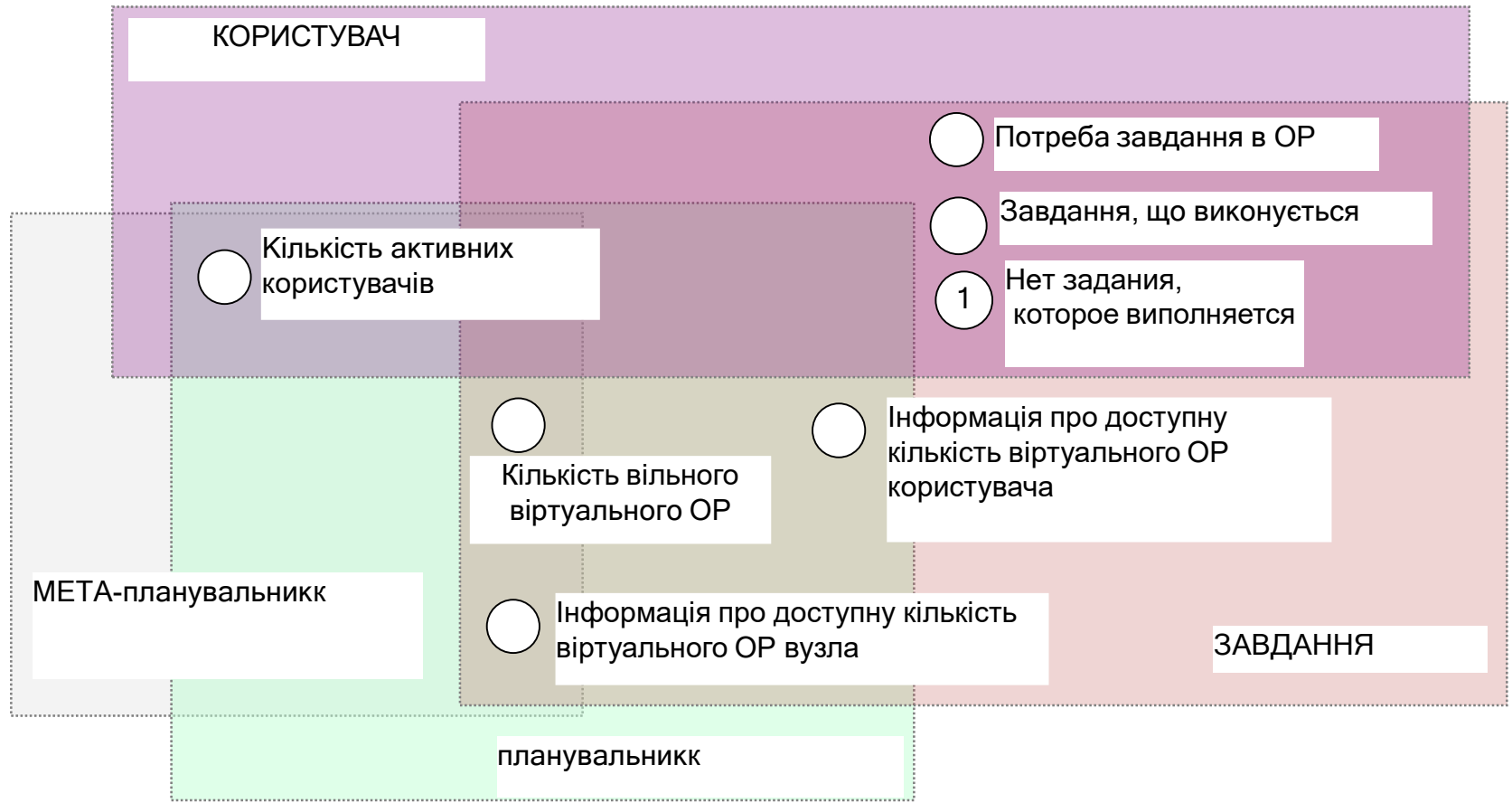
Петрі-об'єктна модель системи управління розподіленими обчислювальними ресурсами



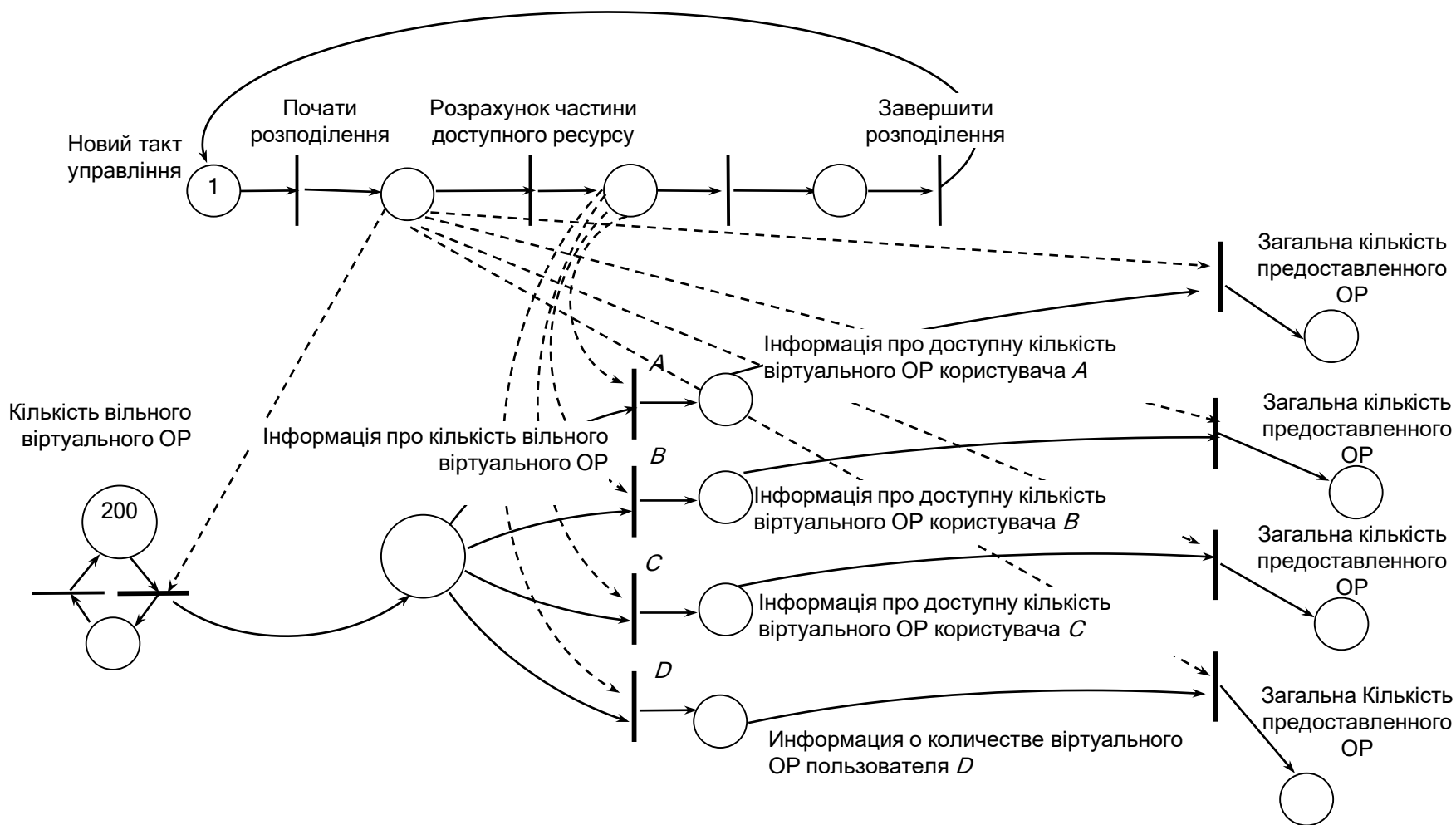
Петрі-об'єктна модель системи управління розподіленими обчислювальними ресурсами



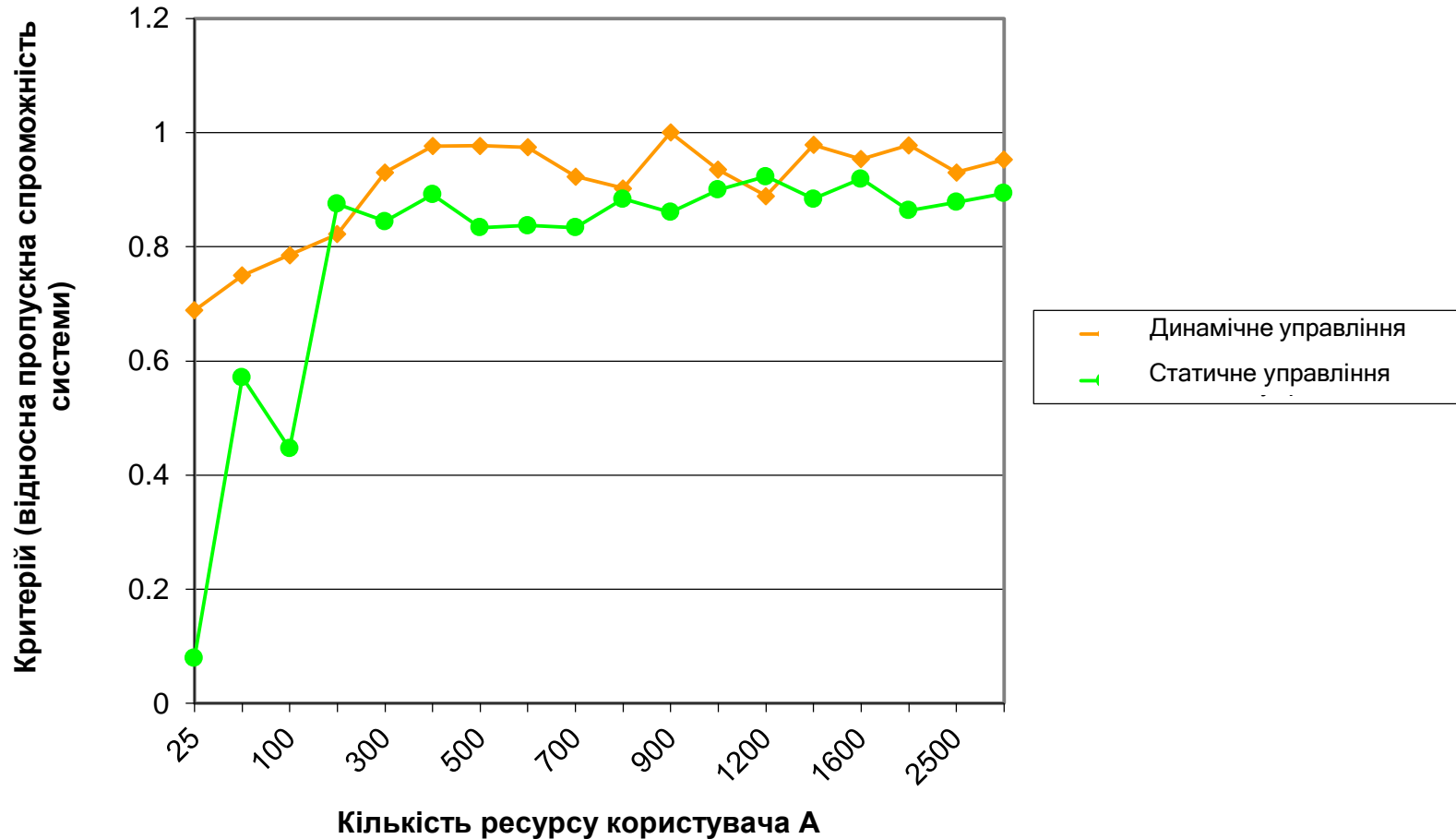
Петрі-об'єктна модель системи управління розподіленими обчислювальними ресурсами



Мережа Петрі-об'єкта «планувальник»



Результати дослідження впливу типу управління на ефективність функціонування системи



Петрі-об'єктна модель системи управління транспортним рухом. Постановка задачі

Входні змінні моделі:

- структура ділянки дорожнього руху,
- Інтенсивності надходження авто у вхідні точки ділянки дорожнього руху
- середня швидкість руху та довжина шляху між сусідніми перехрестями
- засоби регулювання дорожнім рухом на перехрестях
- параметри світлофорних об'єктів управління
- вимоги безпеки дорожнього руху

Вихідні змінні моделі:

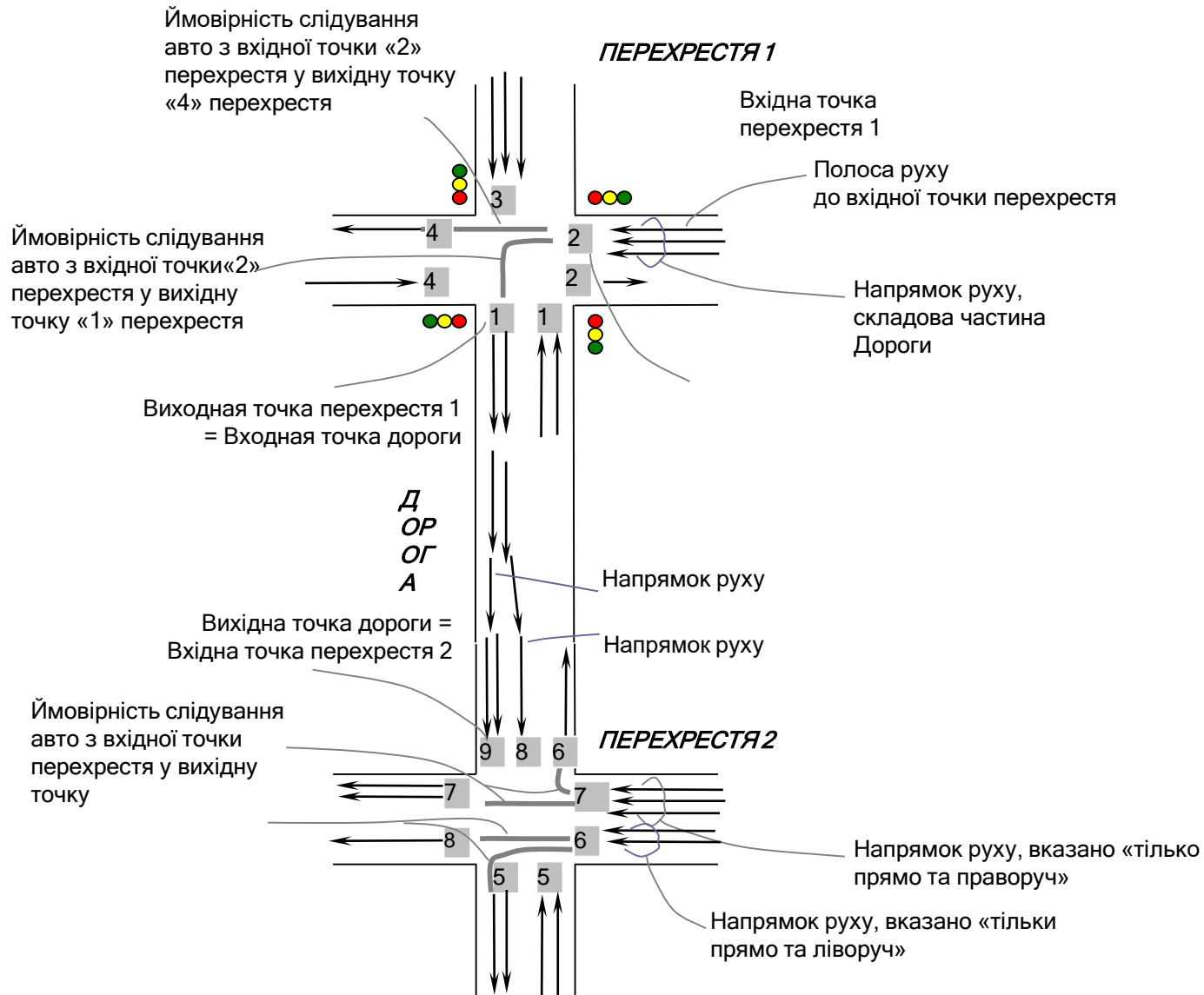
- середня кількість авто, що очікують переїзду, на кожному перехресті в кожному напрямку,
- найбільше зі значень середньої кількості авто, що очікують переїзду, на кожному перехресті в кожному напрямку
- оптимальні значення параметрів управління

Критерій оптимальності

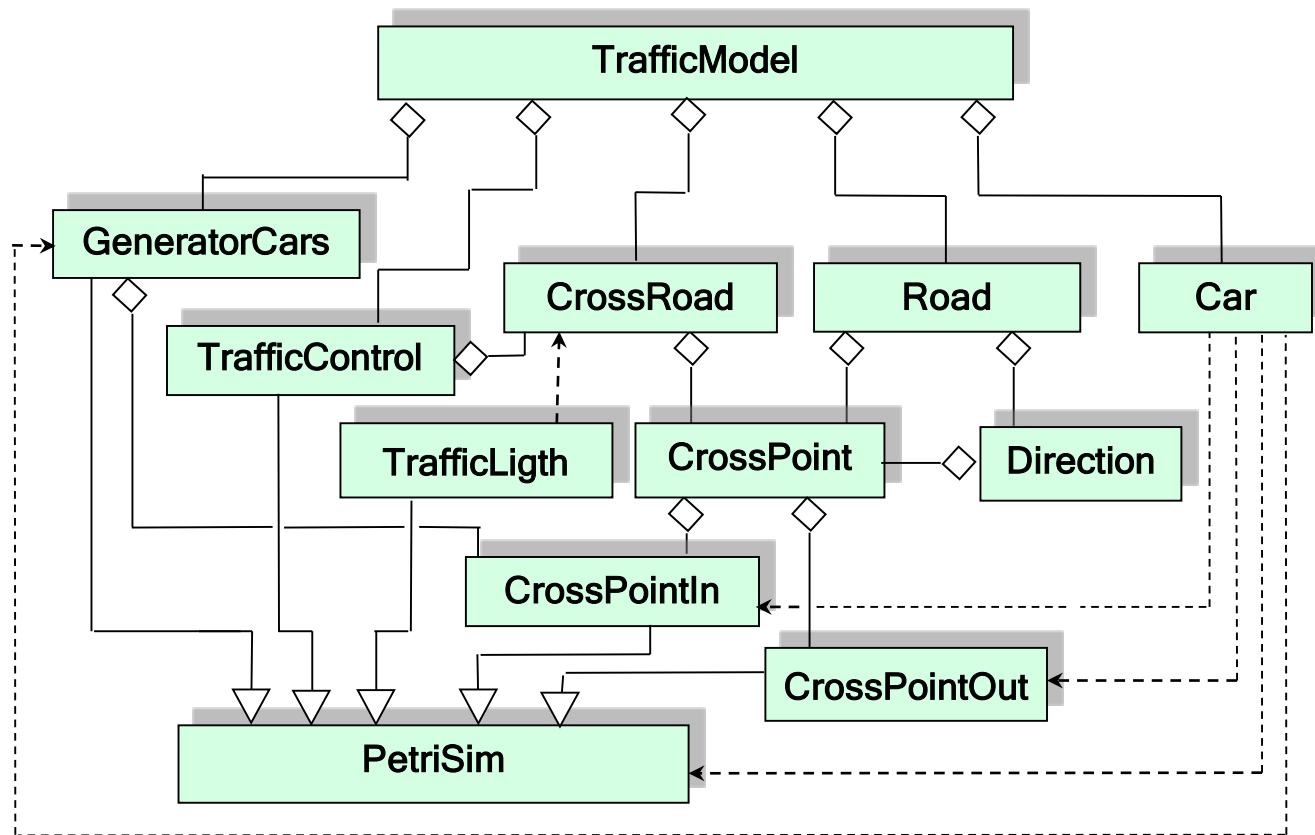
$$z = \max(L_1, L_2, L_3, \dots, L_k) \rightarrow \min$$

Среднее Кількість авто, ожидающих переїзда в i-ой точке

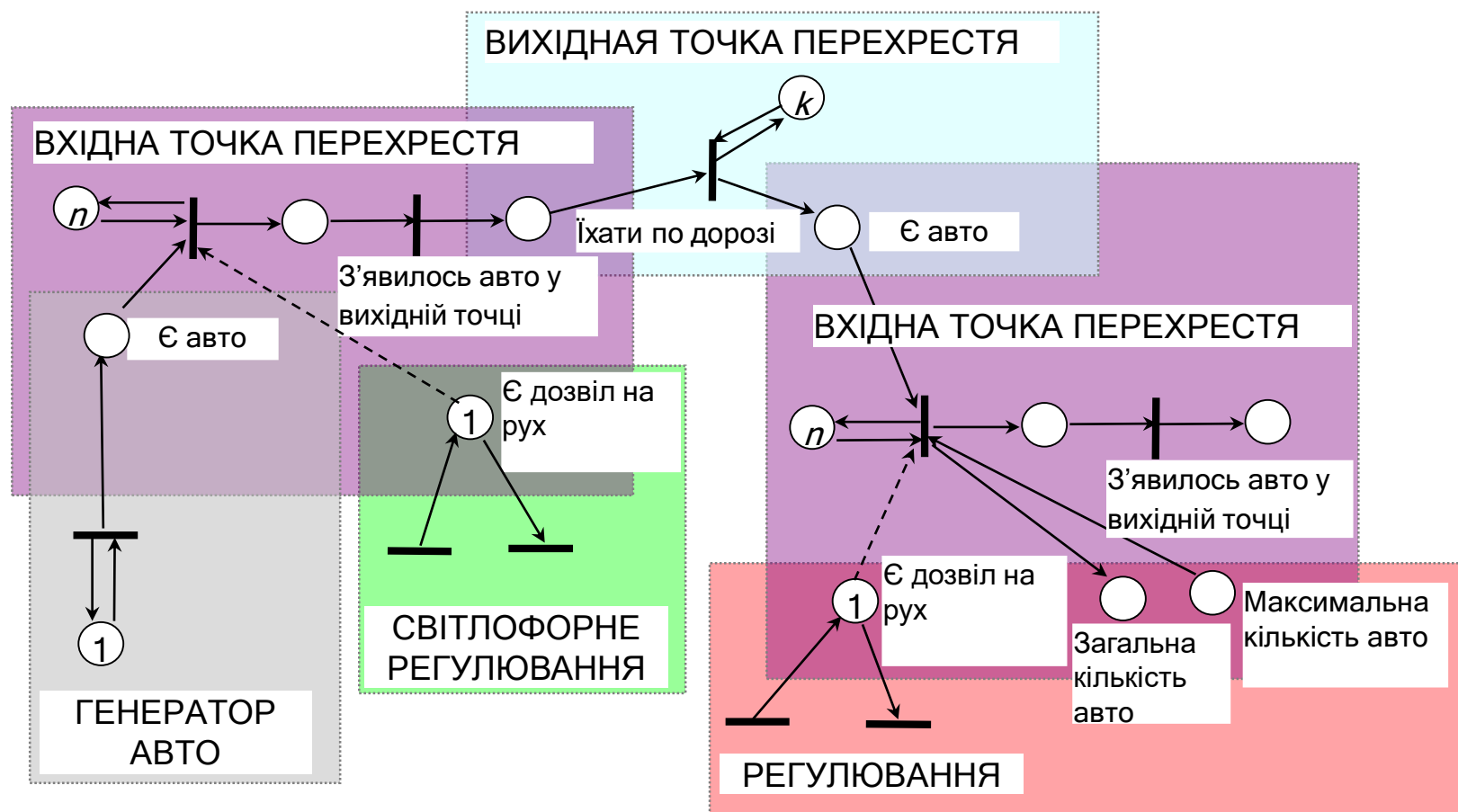
Схема ділянки дорожнього руху



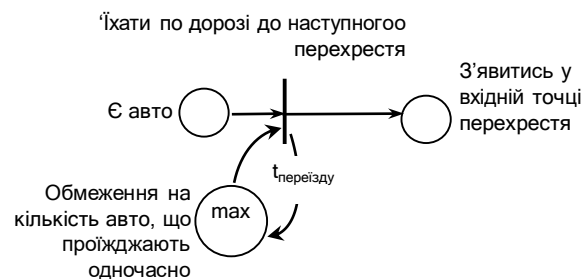
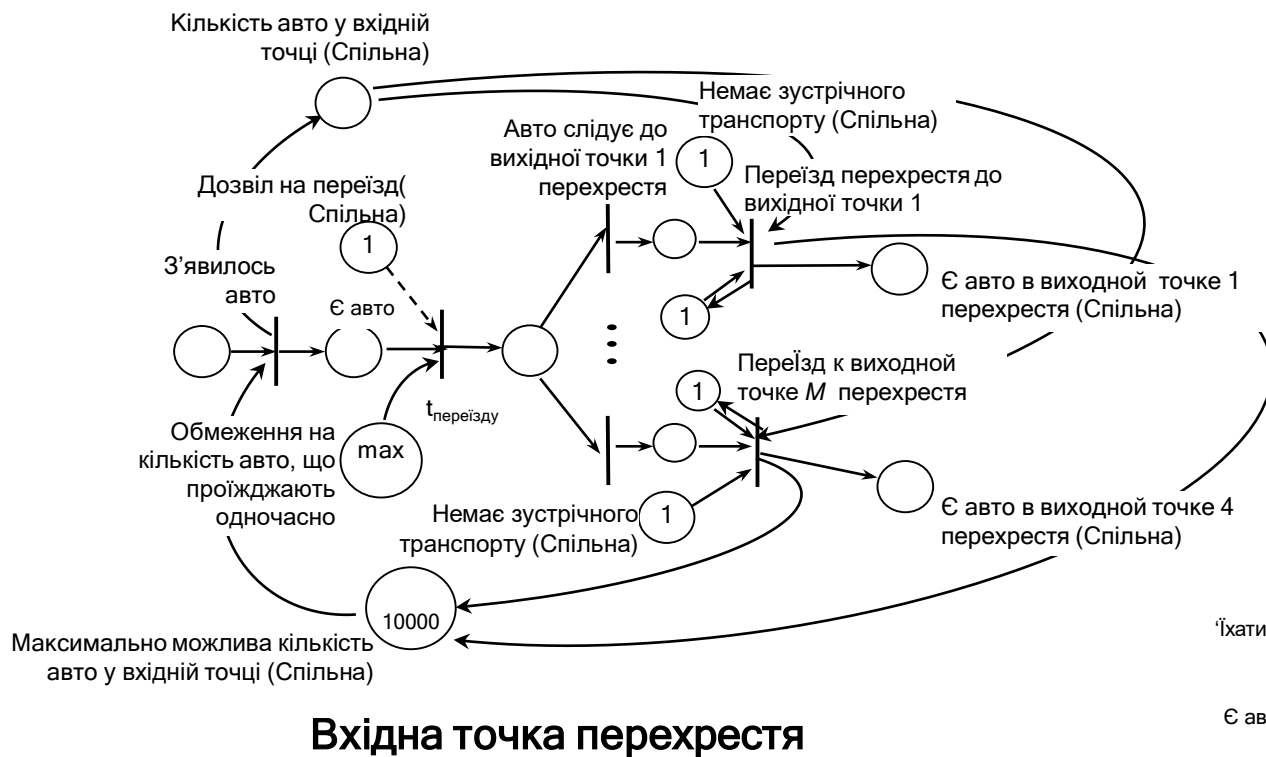
Діаграма класів Петрі-об'єктної моделі системи управління дорожнім рухом



Діаграма зв'язків Петрі-об'єктів моделі системи управління транспортним рухом

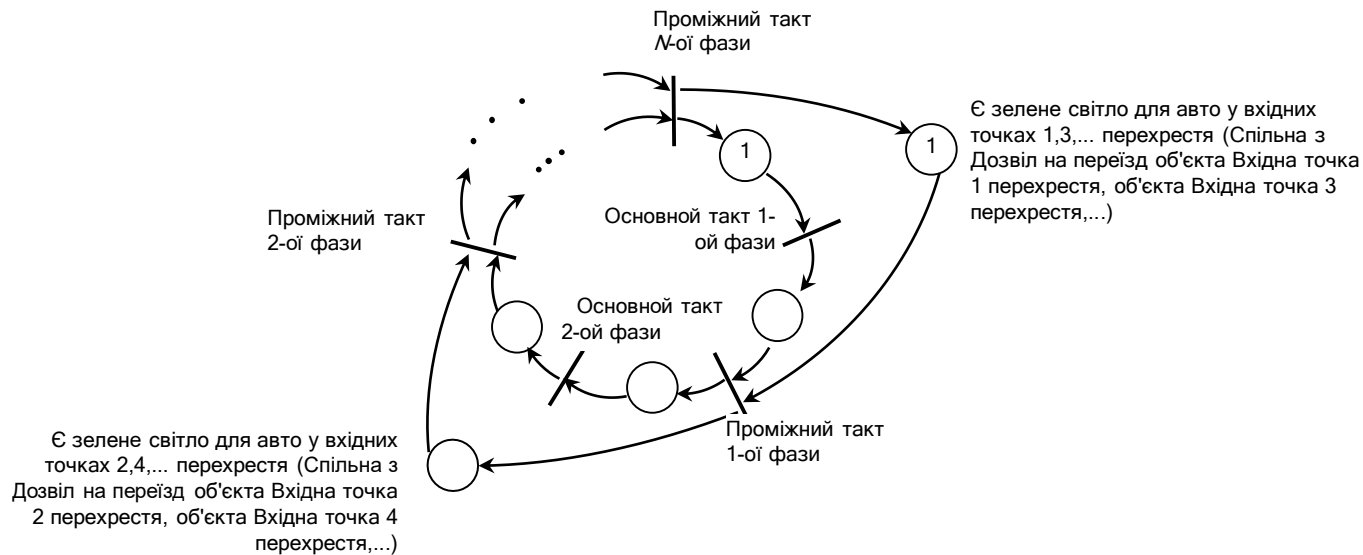
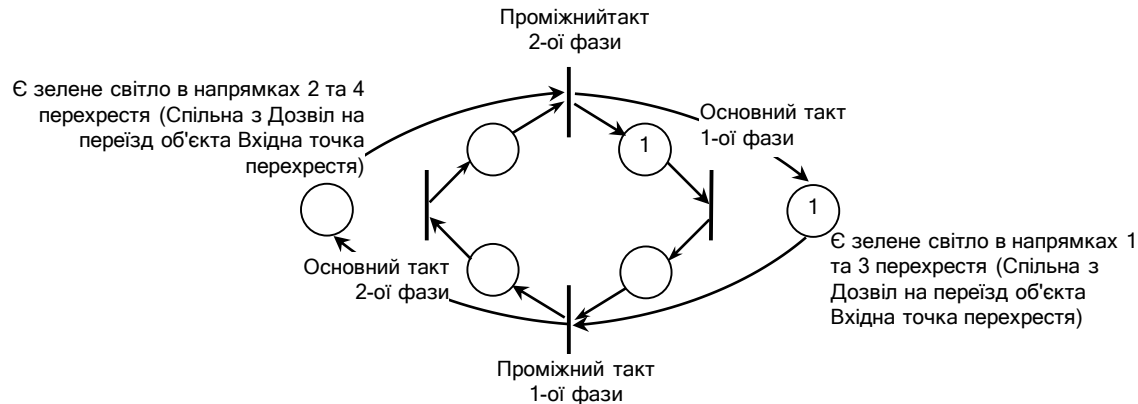


Мережа Петрі-об'єктів Вхідна та Вихідна точки перехрестя

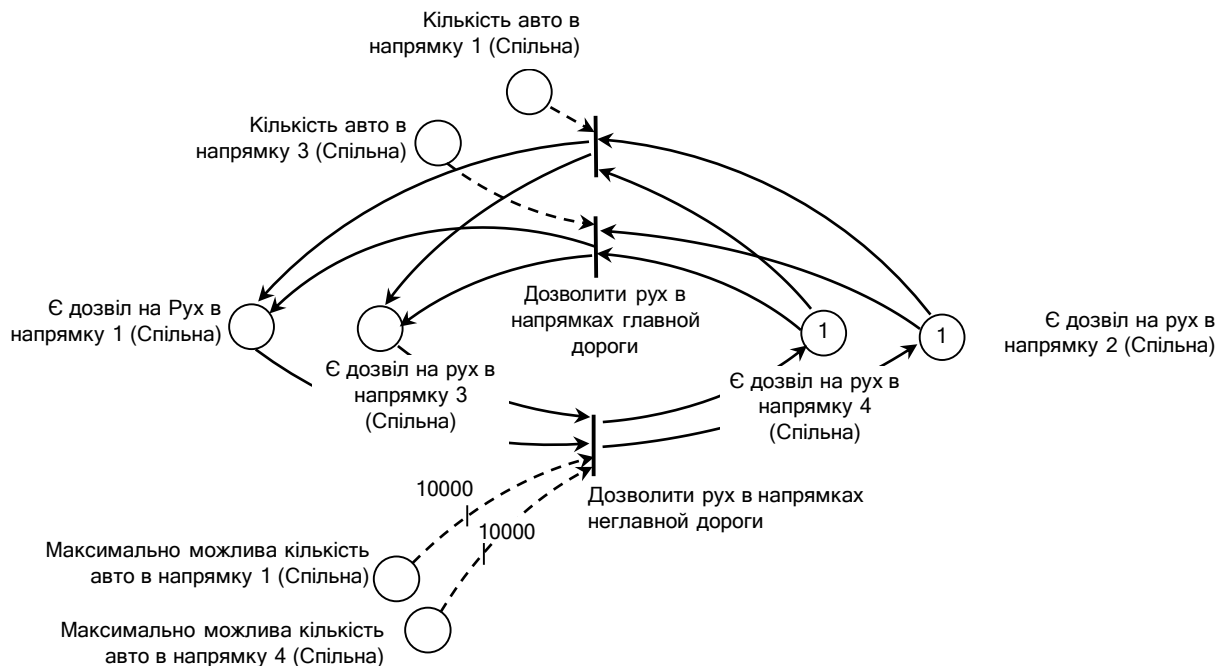


Вихідна точка перехрестя

Мережа Петрі-об'єкта світлофорне регулювання



Сеть Петри-об'єкта Регулювання знаками дорожнього руху



Мережа Петрі-об'єкта Регулювання (правило «Пропустити авто справа»)

