НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки

(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління

(назва кафедри)

"На правах рукопису"
УДК 004.94:519.876.5+004.42

«До захисту допущено»
В.о. завідувача кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)
“____” _____ 20*19* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *126 Інформаційні системи та технології*

(код та назва спеціальності)

ОПП *Інформаційні управляючі системи та технології*

(код та назва спеціалізації)

на тему: *Аналіз ефективності використання паралельних обчислень в інформаційній технології*

Виконала: студентка *VI* курсу групи ІС-81мп

(шифр групи)

Дифучина Олександра Юріївна _____

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *проф., д.т.н., доц. Стеценко І.В.* _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *проф., д.т.н, проф. Томашевський В.М.* _____

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент *проф., д.т.н, проф. Клименко І.А.* _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студентка _____

(підпис)

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *126 Інформаційні системи та технології*
(код і назва)

ОПП *Інформаційні управляючі системи та технології*
(код і назва)

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ О.А.Павлов
(підпис) (ініціали, прізвище)
«__» _____2019 р.

ЗАВДАННЯ

на магістерську дисертацію студентці

Дифучиній Олександрі Юріївні
(прізвище, ім'я, по батькові)

1. Тема дисертації *Аналіз ефективності використання паралельних обчислень в інформаційній технології*

науковий керівник дисертації *Стеценко І.В., д.т.н., доц.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "*28*" *10* 20*19* р. № 3770-с

2. Строк подання студентом дисертації "*2*" *12* 20*19* р.

3. Об'єкт дослідження *процес розробки паралельних обчислень в інформаційних технологіях.*

4. Перелік завдань, які потрібно розробити *1.Проаналізувати сучасні засоби і методи тестування та аналізу ефективності паралельних обчислень. 2.Розробити моделі основних інструментів багатопоточності на основі стохастичних мереж Петрі. 3. Розробити бібліотеку базових фрагментів багатопоточної програми. 4. Розробити компонент для анімації функціонування Петрі-об'єктної моделі.*

5.Виявити залежність ефективності використання певного інструменту багатопоточності від його параметрів на основі розроблених моделей.

5. Орієнтовний перелік ілюстративного матеріалу

1. Схема структурна варіантів використання

2. Схема структурна діяльності

3. Схема структурна класів програмного забезпечення

4. Схема структурна послідовності

5. Схема структурна компонентів програмного забезпечення

6. Рішення з математичного забезпечення

6. Орієнтовний перелік публікацій *Сім публікацій: три статті у фахових виданнях та четверо тез доповідей на наукових конференціях*

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання “ *2* ” *вересня* 20 *19* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Систематизація результатів огляду літератури*	*15.09.2019*	
2	*Порівняльний аналіз існуючих методів аналізу ефективності паралельних обчислень*	*1.10.2019*	
3	*Постановка та формалізація задачі*	*3.10.2019*	
4	*Модифікація існуючих методів аналізу ефективності паралельних обчислень*	*8.10.2019*	
5	*Розробка програмного забезпечення*	*1.11.2019*	
7	*Проведення експериментальних досліджень розробленого методу*	*7.11.2019*	
8	*Оформлення документації*	*19.11.2019*	
9	*Подання роботи на попередній захист*	*20.11.2019*	
10	*Подання роботи на основний захист*	*02.12.2019*	

Студентка _____ *О.Ю. Дифучина*

(підпис) (ініціали, прізвище)

Науковий керівник _____ *І.В. Стеценко*

(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація: 89 с., 22 рис., 21табл., 1 додаток, 41 джерело.

Актуальність. Розробка паралельних програм є актуальною сферою діяльності у програмуванні. Кожний програмний продукт піддається тестуванню на багатьох стадіях реалізації (від прототипу до готового продукту) для оцінки ефективності та якості функціонування. Однак багатопоточні програми відрізняються своєю стохастичною поведінкою та сильно залежать від обчислювальних ресурсів, на яких вони виконуються. Тож для випробування паралельної програми необхідно запустити її велику кількість разів на різних ресурсах. На жаль, традиційні методи тестування не здатні врахувати всі особливості функціонування паралельних алгоритмів та випробувати багатопоточну програму належним чином, передбачивши усі можливі сценарії її виконання. Окрім цього, під час розробки інформаційної технології з використанням паралельних обчислень виникають питання ще й стосовно коректності використання того чи іншого інструменту багатопоточності, та доцільності розпаралелювання алгоритму взагалі. У даній роботі пропонується спосіб тестування багатопоточної програми шляхом моделювання її засобами стохастичних мереж Петрі для аналізу ефективності використання паралельних обчислень в програмі, що дозволить врахувати не детермінований порядок інструкцій, виконуваних потоками та об’єм обчислювальних ресурсів.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Методи візуального програмування Петрі-об'єктних моделей» (№ 0117U000918).

Метою дослідження є підвищення ефективності використання паралельних обчислень в інформаційній технології.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати сучасні засоби і методи тестування та аналізу ефективності багатопоточних програм;

- розробити моделі основних низько- та високорівневих інструментів багатопоточності на основі стохастичних мереж Петрі:
  - створення потоку, початок і кінець його роботи;
  - блокування потоку;
  - блок синхронізації дій потоку;
  - доступ потоків до спільних даних;
  - пул потоків.
- визначити правила співставлення фрагмента програмного Java-коду частині мережі Петрі;
- розробити бібліотеку базових фрагментів багатопоточної програми на основі застосування DESS(Discrete Event System Simulation);
- розробити компонент для анімації функціонування Петрі-об’єктної моделі;
- виявити залежність ефективності використання певного інструменту багатопоточності від його параметрів на основі розроблених моделей.

Об'єктом дослідження є процес розробки паралельних обчислень в інформаційних технологіях.

Предмет дослідження – методи аналізу ефективності використання паралельних обчислень в інформаційних технологіях.

Методи дослідження – методи імітаційного моделювання і математичної статистики, метод Петрі-об'єктного моделювання.

Наукова новизна одержаних результатів:

- вперше розроблено метод Петрі-об'єктного моделювання паралельних обчислень, який дозволяє скоротити часові, фінансові та ресурсні затрати при тестування паралельних алгоритмів;
- вперше запропоновано критерій оцінювання ефективності використання паралельних багатоядерних обчислень в інформаційній технології з урахуванням обчислювальних ресурсів, що надає можливість оцінювати доцільність використання в програмі тих чи інших інструментів багатопоточного програмування.

Апробація результатів дисертації. Результати дослідження, що включені до дисертації оприлюднені на міжнародних наукових конференціях ICCSEEA2018 та ICCSEEA2019.

Публікації. Результати роботи опубліковані в періодичному виданні «Advances in Intelligent Systems and Computing» (volume 754[35], volume 938[39]), що індексується в наукометричній базі Scopus, в тезах Дванадцятої[31] та Тринадцятої[33] міжнародної науково-практичної конференції «Математичне та імітаційне моделювання систем. МОДС» (2017, 2018), в статті фахового журналу «Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка» (випуск 66)[25], в тезах п'ятої Міжнародної науково-практичної конференції «Управління розвитком технологій»[32], та у тезах третьої всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019)[41].

ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, СТОХАСТИЧНІ МЕРЕЖІ ПЕТРІ, БАГАТОПОТЧНЕ ПРОГРАМУВАННЯ, ПЕТРІ-ОБ’ЄКТНИЙ ПІДХІД, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ЕФЕКТИВНІСТЬ

ABSTRACT

Master's Thesis: 89 pp., 22 fig., 21 tab., 1 app., 41 sources.

Topicality. The development of parallel programs is a topical area in programming. Each software product should be tested during the many stages of implementation (from prototype to finished product) in order to evaluate the efficiency and quality of its functioning. However, multithreaded programs differ in their stochastic behavior and are highly dependent on the computational resources they are running on. Therefore, to test a parallel program, you need to run it many times on different resources. Unfortunately, traditional testing methods are not able to take into account all the features of running parallel algorithms and to test a multi-threaded program properly, assuming all possible scenarios for its implementation. In addition, during the development of information technology using parallel computing, there are also questions about the correctness of the use of a multithreaded tool, and about the necessity of parallelizing the algorithm at all. This paper proposes a method to test a multithreaded program by modeling it with stochastic Petri nets to analyze the efficiency of using parallel computations in the program. This method takes into account the non-deterministic order of instructions executed by the threads and the amount of computing resources.

Relationship of work with scientific programs, plans, themes. carried out at the Department of Automated Systems for Information Processing and Management of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» by the topic of "Methods of visual programming of Petri-object models" (№ 0117U000918).

The aim of the research is to increase the efficiency of using parallel computing in information technology.

To achieve this goal the following tasks must be accomplished :

- to analyze modern tools and methods of testing and analyzing the effectiveness of multithreaded programs;
- to develop models of basic low- and high-level multithreading tools based on stochastic Petri nets:
    - creation of the thread, beginning and end of its work;

- thread blocking;
- threads synchronization;
- threads access to shared data;
- thread pool.

– to determine the rules for transforming the Java code fragments into the Petri net;

– to develop a library of basic fragments of multithreaded program which is based on the software DESS (Discrete Event System Simulation);

– to develop a component to animate the functioning of the Petri-object model;

– to determine the dependence of the efficiency of a particular multithreading tool on its parameters using the developed models.

The object of study is the process of developing parallel computing in information technologies.

The subject of study is methods of efficiency analysis of parallel computing in information technologies.

Research methods – methods of simulation modeling and mathematical statistics, the Petri-object simulation method.

Scientific novelty of the obtained results:

- first developed a Petri-object simulation method for parallel computing that reduces time, financial and resource costs when testing parallel algorithms;
- for the first time proposed a criterion for evaluating the efficiency of parallel multi-core computing in information technology, which takes into account computational resources, and provides an opportunity to evaluate the necessity of using certain multithreading tools in the program.

Approbation of the thesis results. The research results included in the thesis were published at the international scientific conferences ICCSEEA2018 and ICCSEEA2019.

Publications. The results of the work are published in the book series “Advances in Intelligent Systems and Computing” (volume 754[35], volume 938[39]), which is indexed in the Scopus database, in the theses of the Twelfth[31] and Thirteenth[33] international scientific-practical conference “Mathematical modeling and simulation of systems.

MODS"(2017, 2018), in the article of the professional journal "Visnyk NTUU "KPI". Informatics, Management and Computer Engineering" (Issue 66)[25], in the theses of the Fifth International Scientific and Practical Conference "Management of Technology Development"[32], and in the theses of the Third All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Management Technologies"(ISTU-2019)[41].

PARALLEL COMPUTING, STOCHASTIC PETRI NETS, MULTITHREADED PROGRAMMING, PETRI-OBJECT APPROACH, SIMULATION, EFFICIENCY

ЗМІСТ

ВСТУП

Незважаючи на потужні технічні засоби сучасних інформаційних технологій, проблема швидкодії обробки даних залишається актуальною дотепер. Одним із шляхів підвищення швидкодії є впровадження паралельних обчислень, які використовуються в усіх високорівневих інформаційних технологіях. Однак, незважаючи на широке застосування паралельних обчислень, існує ряд проблем, що перешкоджають їх ефективному використанню.

Кожний програмний продукт піддається тестуванню на багатьох стадіях реалізації (від прототипу до готового продукту) для оцінки ефективності та якості функціонування. Однак багатопоточні програми відрізняються своєю стохастичною поведінкою та сильно залежать від обчислювальних ресурсів, на яких вони виконуються. Тож для випробування паралельної програми необхідно запустити її велику кількість разів на різних ресурсах. На жаль, традиційні методи тестування не здатні врахувати всі особливості функціонування паралельних алгоритмів та випробувати багатопоточну програму належним чином, передбачивши усі можливі сценарії її виконання. Окрім цього, під час розробки інформаційної технології з використанням паралельних обчислень виникають питання ще й стосовно коректності використання того чи іншого інструменту багатопоточності, та доцільності розпаралелювання алгоритму взагалі.

На сьогодні у світі не існує самостійних інструментів для тестування та налагодження багатопоточних програм. Усі існуючі інструменти ефективні лише для зневадження та оптимізації виконання однопотокових програм [1]. Є багато досліджень на тему проблем багатопоточного програмування, однак засобів виявлення або попередження їх немає. Наприклад, опис численних проблем, пов’язаних з використанням паралельних обчислень, можна знайти в роботі [2]. Небезпеки багатопоточності, що ховаються у Java-програмах детально розглянуті в публікації [3]. Дефекти синхронізації, що викликають помилки у багатопоточних програмах, описані в статті [4].

У даній роботі пропонується спосіб тестування багатопоточної програми шляхом моделювання її засобами стохастичних мереж Петрі. Цей спосіб дозволяє

врахувати не детермінований порядок інструкцій, виконуваних потоками, та обсяг обчислювальних ресурсів. Представлений метод Петрі-об’єктного моделювання паралельних обчислень дозволяє виконати аналіз ефективності використання паралельних багатоядерних обчислень. Даний метод сприяє виявленню помилок у паралельних алгоритмах на стадіях проектування та тестування багатопоточних програм. Запропонований у роботі критерій ефективності використання паралельних обчислень дозволяє визначити коректність та доцільність використання інструментів багатопоточності в інформаційній технології.

1 ОГЛЯД СУЧАСНИХ РІШЕНЬ В МОДЕЛЮВАННІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Все частіше паралельні обчислення розглядаються як єдиний економічно ефективний метод швидкого вирішення обчислювально великих та об'ємних даних [5]. З огляду на це, паралельні обчислення на сьогоднішній день використовуються майже в усіх великих проектах з інформаційних технологій та систем для підвищення швидкодії алгоритмів. Навіть елементарні програмні застосування потребують паралелізму в реалізації графіки та анімаційних процесів. Клієнт-серверні додатки використовують багатопоточний підхід при роботі з багатьма клієнтами. У сфері Big Data також мають місце паралельні обчислення під час аналізу великих масивів даних. Широке застосування багатопоточної технології мають комп'ютерні ігри. Основні принципи та інструменти, що використовуються при розробці паралельних програм описані в роботі [6].

Розробка багатопоточних програм є складним завданням з причини не детермінованого порядку інструкцій, виконуваних потоками. Серед дослідників даної проблеми існує думка, що нетривіальні багатопотокові програми є незбагненними для людини і вони повинні бути більш передбачуваними та зрозумілими [7]. Іншою проблемою є залежність результату запуску паралельної програми від обчислювальних ресурсів, на яких вона запускається.

На сьогодні у світі не існує самостійних інструментів для тестування та налагодження багатопоточних програм. Усі існуючі інструменти ефективні лише для зневадження та оптимізації виконання однопотокових програм [1]. Є багато досліджень на тему проблем багатопоточного програмування, однак засобів виявлення або попередження їх немає. Наприклад, опис численних проблем, пов'язаних з використанням паралельних обчислень, можна знайти в [2]. Небезпеки багатопоточності, що ховаються у Java-програмах детально розглянуті в [3]. Дефекти синхронізації, що викликають помилки у багатопоточних програмах, описані в [4]. Часткові проблеми паралельного програмування вивчаються дослідниками, однак їх неможливо об'єднати простим чином. Так, наприклад, у [8] запропоновано метод статичного виявлення проблеми deadlock. Крім того, були розроблені спеціальні

інструменти для виявлення проблеми deadlock [9] або помилок узгодженості пам'яті [10] в паралельній програмі. Однак розробка інструменту, придатного для тестування всіх існуючих проблем, є складним завданням через відсутність єдиного підходу до їх дослідження.

У цій роботі представлено метод Петрі-об’єктного моделювання паралельних обчислень, який застосовується для аналізу ефективності використання тих чи інших механізмів багатопоточності в інформаційній технології. Цей метод сприяє виявленню помилок у паралельних алгоритмах на стадіях проектування та тестування багатопоточних програм. Запропонований у роботі критерій ефективності використання паралельних обчислень дозволяє визначити коректність та доцільність використання інструментів багатопоточності в інформаційній технології.

Паралельні обчислення можуть розглядатися як в розподілених системах, так і в рамках одно процесорних систем. Проте і в тому, і в іншому випадках підходи розпаралелювання є однаковими. Спочатку задача розбивається на підзадачі, кожну з яких виконує окремий потік. В ідеалі кожна підзадача має виконуватися незалежно від інших, що забезпечує коректну роботу потоків та вирішення головної задачі. Однак на практиці розбиття на абсолютно незалежні під задачі зрідка можливе, що і спричиняє основні проблеми: deadlock(мертве блокування) та проблема доступу до спільних даних.

Deadlock відбувається, коли два або більше потоків чекають сигналу, який ніколи не можна отримати. Наприклад, потік А чекає сигналу з потоку В, але останній очікує сигналу з потоку А. Тому вони впадають у стан взаємного очікування один одного. Deadlock має чудовий опис за допомогою базової мережі Петрі. Широко відомий приклад з 5 філософами, описаний Дж. Пітерсоном у його книзі [11]. До того ж, спеціальний клас базової мережі Петрі, що називається мережею Гадара, запропоновано в роботі [12] для дослідження deadlock ситуацій у багатопоточній програмі.

Проблема доступу до спільних даних означає помилку обчислень, коли більше ніж один потік одночасно модифікують спільні дані. Для виявлення помилок доступу

до спільних даних у багатопоточних програмах розроблено метод з використанням спеціальних мереж Петрі з даними (PD-net) , що пропонується в [13].

Детальний опис основних проблем паралельного програмування міститься в роботі [14].

Варто зазначити, що серед досліджень в області паралельних обчислень трапляються спроби моделювання багатопотокових програм засобами мереж Петрі. Наприклад, використання базових мереж Петрі для моделювання багатопоточних додатків наведено в [15]. Розробка інструмента для візуалізації багатопоточної Java-програми з використанням базових мереж Петрі представлена в [16]. Однак майже у всіх вищезгаданих роботах застосовуються базові мережі Петрі. Хоча використання базових мереж Петрі дозволяє представити окремі механізми багатопоточних програм, проте деталі функціонування паралельного алгоритму не можуть бути відтворені у цьому випадку і лише логіка програми може бути змодельована таким чином. Тому для досягнення найбільш точного відтворення складної взаємодії між потоками має бути застосоване імітаційне моделювання. Підхід до імітаційного моделювання, заснований на формальному описі стохастичними мережами Петрі є кращим для таких складних динамічних систем, як паралельні програми [17]. Стохастичні мережі Петрі дозволяють дослідити виконання програми враховуючи час виконання дій виконуваних потоком, а також збільшити точність побудованої моделі.

До того ж, жодна зі згаданих робіт не бере до уваги ресурси, які використовує паралельна програма під час роботи. Очевидно, що різні комп’ютери мають різний обсяг ресурсів і це має бути враховано під час розробки програми. Через таку залежність від ресурсів для тестування та оптимізації паралельного алгоритму програма має бути запущена на декількох комп’ютерах, що є не зручним і займає немало часу. У цьому випадку має бути застосоване моделювання паралельної програми, що дозволяє врахувати ресурси. Саме за допомогою стохастичної мережі Петрі та Петрі-об’єктного підходу це можливо забезпечити. Така модель дозволить провести тестування, зневаження(debugging) та оцінку ефективності програми із вказаним обсягом ресурсів, тим самим значно заощадивши час розробки програми.

У даній роботі розробляється метод конструювання моделі за кодом програми паралельних обчислень. Будуть визначені правила співставлення наборів інструкцій програмного коду фрагментам імітаційної моделі, будуть розроблені шаблони базових конструкцій моделей для паралельних програм, а також сформульований критерій оцінювання ефективності використання механізмів багатопоточності у програмі. За допомогою цих правил та шаблонів програміст матиме змогу швидко побудувати модель програми для її тестування, зневадження та перевірки ефективності використання паралельних обчислень у програмі.

1.1 Опис аналогів існуючих рішень

Наразі існує велика кількість середовищ розробки, які мають вбудовані інструменти типу зневаджувача(debugger), що дозволяє контролювати виконання програми, та статичного аналізу коду, що відслідковує синтаксичні помилки. Проте складним завданням є пошук помилки в логіці виконання паралельного алгоритму через не детермінований порядок інструкцій, що виконуються потоками, та розподіл ресурсів [18].

Мережі Петрі були розроблені саме для моделювання паралельних процесів. Проте найчастіше їх застосовують для моделювання паралельних процесів у бізнесі та виробництві, але не в програмуванні [11].

Найвідомішими засобами побудови мереж Петрі є POSES++ [19] та CPN Tools [20]. POSES++ – програмне забезпечення для імітації дискретних подій. Цей інструмент має великий недолік у вигляді відсутності графічного редактора, що робить його незручним і недоцільним у використанні. CPN Tools – інструмент для редагування, імітації та аналізу кольорових мереж Петрі. Застосування кольорових мереж Петрі у випадку моделювання паралельних програм є не потрібним, це лише перевантажує модель як і безліч інших додаткових параметрів наявних у CPN Tools, в результаті чого побудована модель є не зрозумілою. До того ж, обидва інструменти націлені на універсальний підхід до моделювання мережами Петрі та не містять ніяких спеціалізованих бібліотек для моделювання паралельних програм. Особливістю паралельних програм є велика кількість повторюваних фрагментів,

очевидно що при використанні звичайних засобів побудова моделі займе багато часу. Тому, задля полегшення і пришвидшення роботи програміста по розробці моделі в даній роботі пропонується Петрі-об'єктний підхід до моделювання паралельних обчислень, де фрагменти програми представляються у вигляді об'єктів. Детально даний підхід описаний в роботі [21].

Здійснивши огляд існуючих методів оцінювання ефективності використання паралельних обчислень можна виокремити існування двох форм оцінювання:

1) оцінювання прискорення процесу обчислень;
2) оцінювання максимально можливого прискорення процесу обчислень.

У роботі [22] детально описані показники ефективності паралельного алгоритму. Загалом оцінювання ефективності паралельних обчислень може здійснюється за такими показниками:

- прискорення, що отримується від використання процесорів;
- ефективність використання процесорів;
- вартість обчислень.

Прискорення (Speedup), одержуване при використанні p процесорів для реалізації паралельного алгоритму складності n, оцінюється величиною, наведеною у формулі 1.1:

$$S_p(n) = T_1(n)/T_p(n), \tag{1.1}$$

де T_1 – час виконання послідовного алгоритму на скалярній ЕОМ;

T_p – час виконання паралельного алгоритму;

Ефективність (Efficiency) використання p процесорів при паралельній реалізації алгоритму складності n оцінюється величиною, наведеною у формулі 1.2:

$$E_p(n) = T_1(n)/\big(p \cdot T_p(n)\big) = S_p(n)/p, \tag{1.2}$$

де T_1 – час виконання послідовного алгоритму на скалярній ЕОМ;

T_p – час виконання паралельного алгоритму;

$S_p(n) \leq p$ – прискорення, одержуване при використанні p процесорів для реалізації паралельного алгоритму складності n.

Величина ефективності $E_p(n) \leq 1$ визначає середню частку часу виконання алгоритму, протягом якої процесори реально використовуються для вирішення завдання.

Варто звернути увагу, що спроби підвищення якості паралельних обчислень по одному з показників (прискоренню або ефективності) може привести до погіршення ситуації за іншим показником, тому що показники якості паралельних обчислень є суперечливими. Наприклад, підвищення прискорення зазвичай може бути забезпечено за рахунок збільшення числа процесорів, що призводить, як правило, до падіння ефективності. І навпаки, підвищення ефективності досягається в багатьох випадках при зменшенні числа процесорів. Тож можна зробити висновок, що розробка методів паралельних обчислень часто потребує вибору деякого компромісного варіанту з урахуванням бажаних показників прискорення та ефективності [22].

Оцінка вартості (cost) обчислень, наведена у формулі 1.3, визначається як добуток часу паралельного виконання алгоритму і числа процесорів, що використовуються:

$$\mathrm{C}_p = pT_p, \tag{1.3}$$

де p – кількість процесорів;

T_p – час паралельного виконання алгоритму.

Очевидним мінусом перелічених показників ефективності паралельних обчислень є те, що вони розраховуються за результатами виконання алгоритму на конкретній машині. Тобто, попередньо оцінити ефективність паралельного алгоритму таким чином неможливо, необхідний запуск програми на реальних обчислювальних ресурсах. Окрім результатів запуску паралельної програми, необхідні ще результати запуску переробленого послідовного алгоритму, що потребує додаткових часових витрат. До того ж, отримана оцінка ефективності буде дійсна лише для конкретних використаних ресурсів. Це означає, що оцінювання

ефективності паралельних обчислень за цими показниками є не вигідним, адже потребує великих часових та ресурсних затрат.

Іншою формою оцінювання ефективності паралельних обчислень є оцінювання максимально досяжного паралелізму. Як правило, алгоритм не може бути повністю розпаралелений, завжди існують частини, які мусять виконуватися послідовно. Такі послідовні частини часто перешкоджають досягненню максимального прискорення. Закон Амдала (формула 1.4) [23] визначає верхню межу прискорення процесу обчислень, враховуючи частку послідовних обчислень в алгоритмі.

$$S_p \leq \frac{1}{f+(1-f)/p} \leq \frac{1}{f}, \tag{1.4}$$

де f – частка послідовних обчислень в алгоритмі;

$(1-f)$ – частка паралельних обчислень в алгоритмі;

p – кількість процесорів, що використовуються для реалізації паралельного алгоритму.

Однак закон Амдала ніяким чином не враховує часові затрати, які йдуть на розпаралелювання. Досить часто трапляються випадки, що ці затрати є більшим, ніж отриманий виграш у часі від розпаралелювання.

Існує також закон Густавсона-Барсіса [24] (формула 1.5), що оцінює максимально досяжне прискорення виконання програми при використанні паралельних обчислень з урахуванням частки послідовних обчислень.

$$S_p = g + (1-g)p = p + (1-p)g, \tag{1.5}$$

де $g = \frac{\tau(n)}{\tau(n)+\pi(n)/p}$ – частка послідовних обчислень, $\tau(n)$ – час послідовної частини виконуваних обчислень, $\pi(n)$ – час паралельної частини виконуваних обчислень;

p – кількість процесорів.

Обидва наведені закони дозволять оцінити лише верхню межу прискорення, тобто ідеальний випадок використання паралелізму. На практиці ж ефективність використання паралельних обчислень залежить від багатьох факторів починаючи від

обчислювальних ресурсів і закінчуючи грамотним підбором параметрів інструментів багатопотоковості. До того ж, основною перевагою даних методів оцінювання прискорення є врахування частки послідовних обчислень, однак точно оцінити її досить важко.

Існуючі способи аналітичної оцінки прискорення є надто не точними через те, що не враховують деталі паралельного алгоритму та характеристики обчислюваних ресурсів, на яких реалізовуються паралельні обчислення. Через необхідність використання синхронних обчислень в паралельних алгоритмах важко оцінити реальну частку обчислень, що виконуються одночасно. Тому, якщо в обчисленнях задіяно *n* процесорів, з цього не слідує, що обчислення будуть виконуватися в *n* разів швидше. Виходячи з цього виникає потреба у методі оцінювання ефективності використання паралельних обчислень, який враховує обсяг обчислювальних ресурсів та параметри паралельного алгоритму. Саме тому в даній роботі пропонується критерій ефективності використання паралельних обчислень в інформаційній технології, який базується на методі Петрі-об’єктного моделювання паралельних обчислень, що дозволяє побудувати точну модель алгоритму, врахувавши всі часові та ресурсні затрати, та оцінити доцільність використання тих чи інших інструментів багатопотокового програмування.

1.2 Опис процесу діяльності

Для аналізу ефективності використання паралельних обчислень в інформаційній технології використовується метод Петрі-об’єктного моделювання паралельних програм, який являє собою бібліотеку програмного забезпечення і призначений для швидкої розробки моделей паралельних програм та аналізу їх ефективності. Імітаційна модель складається з готових шаблонів мереж Петрі базових фрагментів паралельних програм та окремих елементів мережі Петрі. Збереження моделі здійснюється у файл.

Дії, які повинен виконати користувач для побудови та виконання моделі наведені в графічному матеріалі («Схема структурна діяльності»).

Користувач запускає програму DESS.jar, після чого за допомогою панелі шаблонів (зверху) та вбудованих елементів мережі Петрі користувач будує модель на панелі малювання (по середині). Налаштування параметрів елементів відбувається у вікнах, що відкриваються на вимогу користувача. Потім користувач має змогу зберегти модель у файл та виконати імітацію (з анімацією або без) на вказаному інтервалі часу. Результати імітації виводяться на панель справа.

Опис розробленого програмного забезпечення для моделювання багатопотокових програм викладений у статті [25].

1.3 Опис функціональної моделі

Схема структурна варіантів використання представлена у графічному матеріалі («Схема структурна варіантів використання»).

Далі розглянемо потоки варіантів використання.

Потік варіанту використання «Користувач»:

Передумова: користувач повинен запустити застосування DESS.

Основний потік:

- початок відбувається при відкритті застосування;
- надається панель для конструювання моделі;
- використання шаблонів базових фрагментів;
- редагування елементів моделі;
- створення нових елементів;
- налаштування параметрів елементів;
- налаштування параметрів імітації;
- збереження моделі;
- запуск імітації;
- перегляд результатів.

Альтернативний потік А1:

- відкриття вже існуючої моделі паралельної програми.

1.4 Опис постановки задачі

1.4.1 Призначення розробки

Призначенням розробки є надання математично обґрунтованого способу побудови моделей паралельних обчислень та аналізу ефективності використання таких обчислень для розробників багатопоточних програм та застосувань.

1.4.2 Мета розробки

Метою роботи є підвищення ефективності використання паралельних обчислень в інформаційній технології.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- розробка моделей базових фрагментів багатопоточної програми на основі стохастичних мереж Петрі:
    - створення потоку, початок і кінець його роботи;
    - блокування потоку;
    - блок синхронізації дій потоку;
    - доступ потоків до спільних даних;
    - пул потоків.
- визначення правил співставлення фрагмента програмного Java-коду частині мережі Петрі;
- розробка бібліотеки базових фрагментів багатопоточної програми на основі застосування DESS(Discrete Event System Simulation) [26];
- впровадження змін в інтерфейсі DESS для коректного використання бібліотеки;
- розробка компоненту для анімації функціонування Петрі-об’єктної моделі.

Висновки до розділу

У даному розділі проаналізоване предметне середовище та здійснений огляд існуючих аналогів в області моделювання паралельних обчислень.

При аналізі існуючих рішень було виявлено та проаналізовано декілька розробок. Жодна з них повністю не задовольнила вимоги до моделювання паралельних програм та оцінювання ефективності використання паралельних обчислень, що і спонукало до створення даної розробки. Істотною відмінністю від попередніх досліджень є те, що в Петрі-об’єктній моделі паралельні обчислення відтворюються не тільки покроково, але й у часі з урахуванням затримок програмних інструкцій. Це важливий аспект, який необхідно враховувати для правильної імітації взаємодії потоків та подальшого аналізу їх ефективності. Крім того, в моделі враховується вплив наявних обчислювальних ресурсів.

2 ТЕОРЕТИЧНІ ЗАСАДИ ПЕТРІ-ОБ'ЄКТНОГО МОДЕЛЮВАННЯ

2.1 Змістовна постановка задачі

Нехай задана багатопоточна програма, кожний потік якої виконує задану послідовність інструкцій та взаємодіє з іншими потоками, використовуючи призупинення та синхронізацію його дій з діями інших потоків. Потрібно поставити у відповідність паралельному алгоритму обчислень, який виконує багатопоточна програма, формалізовану модель обчислень, представлену стохастичною мережею Петрі.

2.2 Стохастичні мережі Петрі

Математичний опис стохастичної мережі Петрі наведений у [27].

Під стохастичною мережею Петрі розумітимемо багатоканальну мережу Петрі з часовими затримками $\text{PetriNet} = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{W}, \mathbf{K}, \mathbf{R})$, що задається множиною позицій $\mathbf{P} = \{P\}$; множиною переходів $\mathbf{T} = \{T\}$, $\mathbf{P} \cap \mathbf{T} = \emptyset$; множиною дуг $\mathbf{A} \subseteq \mathbf{P} \times \mathbf{T} \cup \mathbf{T} \times \mathbf{P}$; множиною натуральних чисел $\mathbf{W}: \mathrm{A} \to \mathrm{N}$, що задають кратності дуг; множиною пар значень $\mathbf{K} = \{(c_T, b_T) | T \in \mathbf{T}, c_T \in N, b_T \in [0; 1]\}$, які задають пріоритет і ймовірність запуску переходів; множиною невід'ємних дійсних значень $\mathbf{R}: \mathrm{T} \to \Re_+$, що характеризують часові затримки в переходах (N – множина натуральних чисел, $\Re_+$ – множина дійсних невід'ємних чисел). Часова затримка переходу може бути визначена випадковою величиною із заданим законом розподілу або детермінованим значенням.

Варто зазначити, що будь-яка мережа Петрі повинна містити хоча б один перехід. При цьому кожний перехід мусить мати хоч одну вхідну і хоч одну вихідну позицію. Необхідною також є наявність у стохастичній мережі Петрі хоч одного переходу із ненульовою часовою затримкою.

Опис стану мережі Петрі в поточний момент часу t здійснюється станом її позицій $\mathbf{M}(t)$ та станом її переходів $\mathbf{E}(t)$:

$$\big(\mathbf{M}(t), \mathbf{E}(t)\big) = \mathbf{S}(t).$$

Стан позицій визначається маркіруванням мережі Петрі в момент часу t: $M(t) = \{M_P(t) | M_P(t) \in N_0, P \in \mathbf{P}\}$, де N_0 - множина цілих невід'ємних чисел. Стан переходів визначається множиною $\mathbf{E}(t) = \{E_T(t) | T \in \mathbf{T}\}$. Стан кожного переходу $E_T(t)$ описується множиною моментів виходу маркерів із каналів переходу, які на момент часу t активні :

$$E_T(t) = \{ [E_T(t)]_q | [E_T(t)]_q \in \Re_+, q \in N \},$$

де q - номер каналу переходу, $q = 1,2...|E_T(t)|$, $|E_T(t)|$ - кількість активних каналів в момент часу *t*.

У випадку коли всі канали переходу вільні, стан переходу набуває значення $E_T(t) = \{\infty\}$, що тлумачиться як «в найближчий час не очікується вихід маркерів з переходу».

Змінна $\tau_T(t)$ визначає найменший момент виходу маркерів із переходу T:

$$\tau_T(t) = \min_q [E_T(t)]_q.$$

Тоді момент найближчої події визначається мінімальним значенням з усіх $\tau_T(t)$:

$$t_n = \min_T \tau_T(t_{n-1}),\ t_n \geq t_{n-1}.$$

Умовою запуску переходу є наявність маркерів у всіх вихідних позиціях переходу у кількості, рівній кратності дуги.

У багатоканальній стохастичній мережі Петрі у разі виконання умови запуску маркери входять в один із каналів переходу і займають його на період часу, рівний часовій затримці R_T. Одразу після спливу часової затримки відбувається вихід маркерів з переходу, що призводить до зміни маркірування. Внаслідок такої зміни можуть виконатися умови запуску для інших переходів мережі. Кожного разу, коли момент часу відповідає найближчій події, відбувається вихід маркерів, і в той же момент часу виконується вхід маркерів. Очевидно, що на відміну від базової мережі Петрі, яка виконує вхід і вихід маркерів у вигляді однієї дії, стохастична мережа Петрі потребує розподілу входу маркерів в переходи і виходу маркерів з переходів на дві різні дії [28]:

$$S(t_{n-1}) \rightarrow S^+(t_n) \rightarrow S(t_n),$$

де S^+ - перехідний стан мережі Петрі, який є результатом виходу маркерів з її переходів у момент часу t_n.

Вихід маркерів з каналу багатоканальної мережі Петрі відбувається тоді, коли поточний момент часу співпадає з найменшим моментом виходу маркерів із переходу. При виході маркерів з переходу в кожну його вихідну позицію маркери видаляються в кількості, рівній кратності дуги, яка з'єднує цей перехід з цією позицією.

Введемо предикат $\mathrm{Y}(\mathrm{T}, t_n)$, який визначає виконання умову виходу маркерів з переходу T в момент часу t_n:

$$\left(\min_q [E_T(t_{n-1})]_q = t_n\right) \Rightarrow \mathrm{Y}(\mathrm{T}, t_n) = 1,$$

$$\left(\min_q [E_T(t_{n-1})]_q \neq t_n\right) \Rightarrow \mathrm{Y}(\mathrm{T}, t_n) = 0.$$

Перетворення стану мережі Петрі $D^+: S(t_{n-1}) \to S^+(t_n)$, що відтворює вхід маркерів у її переходи має такий математичний опис:

$$\forall P \in \mathbf{P} \quad M_P^+(t_n) = M_P(t) + \sum_{T \in {}^\bullet P} Y(T, t_n) \cdot W_{T,P} \mid s_T(t) \mid,$$

$$\forall T \in \mathbf{T} \mid Y(T, t') = 1 \quad E_T^+(t') = \begin{cases} \{\infty\} \leftarrow |s_T(t)| = |E_T(t)|, \\ E_T'(t') = E_T(t) \setminus \left\{ [E_T(t)]_q \mid q \in s_T(t) \right\} \leftarrow |s_T(t)| \neq |E_T(t)|, \end{cases}$$

де $s_T(t) = \left\{ q \in \mathrm{N} \mid [E_T(t)]_q = \tau_T(t) \right\}$ - множина каналів переходу, для яких момент виходу співпадає з найменшим значенням з усіх моментів виходу переходу T, ${}^\bullet P$ – множина переходів, для яких існує вихідна дуга, що зв'язує їх з позицією P.

Вхід маркерів у перехід мережі Петрі відбувається лише у випадку, коли у всіх вхідних позиціях цього переходу наявна кількість маркерів рівна кількості зв'язків (вхідних дуг). При вході маркерів в перехід з кожної його вхідної позиції маркери видаляються в кількості, рівній кратності дуги, яка з'єднує цю позицію з цим переходом.

Конфліктними називають переходи, для яких одночасно виконана умова запуску [29]. Конфлікт переходів вирішується таким чином: визначаються всі переходи з найвищим пріоритетом, потім з них вибирається один за допомогою

випадкового числа, рівномірно розподіленого в інтервалі [0;1] та значень ймовірностей запуску конфліктних переходів, розрахованих за заданими значеннями параметрів. За результатом вирішення конфлікту переходів визначається множина переходів $\Psi \subset \mathbf{T}$, для яких в даний момент часу виконується вхід маркерів. Введемо предикат $\mathrm{X}(\mathrm{T}, t_n)$, який визначає умову входу маркерів у перехід T в момент часу t_n:

$$T \in \Psi \Rightarrow \mathrm{X}(\mathrm{T}, t_n) = 1$$

$$T \notin \Psi \Rightarrow \mathrm{X}(\mathrm{T}, t_n) = 0$$

Перетворення стану мережі Петрі $D^-: S^+(t_n) \to S(t_n)$, що відтворює вхід маркерів у її переходи має такий математичний опис:

$$\forall P \in \mathbf{P} \quad M_P(t_n) = M_P^+(t_n) - \sum_{T \in P^\bullet} W_{P,T} \cdot X(T, t_n),$$

$$\forall T \in \mathbf{T} \mid X(T, t_n) = 1 \quad E_T(t_n) = \begin{cases} \{t_n + R_T\} \leftarrow \tau_T = \infty \\ E_T^+(t_n) \cup \{t_n + R_T\} \leftarrow \tau_T < \infty \end{cases},$$

де $P^\bullet$ – множина переходів, для яких існує вхідна дуга, що зв'язує їх з позицією P.

У мережі Петрі з багатоканальними переходами вхід маркерів відбувається багатократно до досягнення стану мережі Петрі, в якому для жодного з її переходів не виконана умова запуску:

$$(D^-)^m: S^+(t_n) \to S(t_n),$$

$$m: (D^-)^m\left(S^+(t_n)\right): \overset{\vee}{\mathrm{T}} Z(T, t_n) = 0,$$

де $Z(T, t_n)$ – предикат, що визначає виконання умови запуску переходу T в момент часу t_n:

$$\left(\forall P \in {}^\bullet T \quad M_P^+(t_n) \geq W_{P,T}\right) \Rightarrow Z(T, t_n) = 1$$

$$\left(\exists P \in {}^\bullet T \quad M_P^+(t_n) < W_{P,T}\right) \Rightarrow Z(T, t_n) = 0,$$

${}^\bullet T$ – множина вхідних позицій переходу T.

Повне функціонування стохастичної багатоканальної мережі Петрі в часі описується системою рівнянь:

$$\begin{cases} t_n = \min\limits_{T}\left(\min\limits_{q}\left[E_T(t_{n-1})\right]_q \right), t_n \geq t_{n-1} \\ \mathbf{S}(t_0) = \left(D^-\right)^m (\mathbf{S}(t_0)) \\ \mathbf{S}(t_n) = \left(D^-\right)^m \left(D^+(\mathbf{S}(t_{n-1})\right) \\ n = 1,2,\ldots \end{cases}$$

Алгоритм імітації складається у відповідності до математичного опису [30].

2.3 Петрі-об’єктне моделювання

Петрі-об’єктом є об’єкт в термінах об’єктно-орієнтованого програмування (ООП) універсального класу PetriSim, який в одному зі своїх полів містить опис стохастичної мережі Петрі, а в одному зі своїх методів - алгоритм імітаційного моделювання стохастичної мережі Петрі. Таким чином Петрі-об’єкт поєднує в собі переваги ООП та стохастичних мереж Петрі:

- тиражування об’єктів зі схожою динамікою та заданими параметрами;
- розробка динаміки складних об’єктів на основі спадкування універсальних об’єктів;
- використання уніфікованого опису динаміки об’єктів у вигляді мереж Петрі;
- можливість використання візуальних програмних засобів для розробки динаміки об’єктів.

До того ж, у тезах [31] доведено, що Петрі-об’єктна технологія надає переваги в швидкості не тільки на етапі виконання алгоритму імітації, але й на етапі розробки моделі в програмному забезпеченні моделювання дискретно-подійних систем. Процес розробки моделі у згаданому програмному забезпеченні описаний в [32].

Зв’язки між Петрі-об’єктами передбачені двох типів:

1) з використанням спільної позиції;
2) з використанням ініціалізації події.

Доведено, що ці зв’язки забезпечують таке з’єднання Петрі-об’єктів в модель, що динаміка моделі представляється стохастичною мережею Петрі [21]. Це важливо для обґрунтування обчислюваності моделі та оцінки складності обчислень.

Математичний опис Петрі-об’єкта співпадає з описом стохастичної мережі Петрі. Перетворення D^- і D^+ Петрі-об’єктної моделі розкладаються в перетворення D^- і D^+ Петрі-обєктів. Тому рівняння станів Петрі-об’єктної моделі має вигляд:

$$\begin{cases} t_n = \min\limits_{T}\left(\min\limits_{q}\left[E_T(t_{n-1})\right]_q \right), t_n \geq t_{n-1} \\ \mathbf{S}(t_1) = \begin{pmatrix} \left(D^-\right)^m\left(\tilde{\mathbf{S}}_1(t_0)\right) \\ \dots \\ \left(D^-\right)^m\left(\tilde{\mathbf{S}}_N(t_0)\right) \\ \dots \\ \left(D^-\right)^m\left(\tilde{\mathbf{S}}_L(t_0)\right) \end{pmatrix}, \ \mathbf{S}(t_n) = \begin{pmatrix} \left(D^-\right)^m\left(D^+\left(\tilde{\mathbf{S}}_1(t_{n-1})\right)\right) \\ \dots \\ \left(D^-\right)^m\left(D^+\left(\tilde{\mathbf{S}}_N(t_{n-1})\right)\right) \\ \dots \\ \left(D^-\right)^m\left(D^+\left(\tilde{\mathbf{S}}_L(t_{n-1})\right)\right) \end{pmatrix}, \forall \tilde{\mathbf{S}}_N(t_n): \bigvee\limits_{T\in T_n} \mathbf{Z}(T,t_n) = 0 \end{cases},$$

де $\tilde{S}_N$ – вектор стану Петрі-об’єкта, модифікований з урахуванням зв’язків з іншими об’єктами.

Алгоритм імітації Петрі-об’єктної моделі складається у відповідності до математичного опису [30].

2.4 Метод Петрі-об’єктного моделювання паралельних обчислень

У тезах доповіді [33] запропоновано метод тестування паралельних програм на моделях, створених засобами стохастичних мереж Петрі, з урахуванням зайнятості обчислювальних ресурсів. Розглядаючи основні методи багатопоточного алгоритму, описані в підручниках Java [34], їм у відповідність були поставлені фрагменти мережі Петрі. Зауважимо, що всі переходи, які відтворюють інструкції програми, використовують основні ресурси (рисунок 2.1). Однак для спрощеного представлення ці ресурси не відображатимуться для кожного переходу, їхня присутність передбачається за замовченням.

Створення, початок і завершення роботи потоку ініціюються основним процесом наступного програмного коду:

```
public static void main(String[] args) {
        Thread thread = new Thread(new Runnable());
        thread.start();
}
```

Цей код має еквівалентне представлення у вигляді мережі Петрі, яка містить розгалуження після події "start", коли новий потік починає роботу (рисунок 2.1). Коли створення потоку виконано, відбувається подія "end" (завершення роботи) потоку, яка створює інший потік.

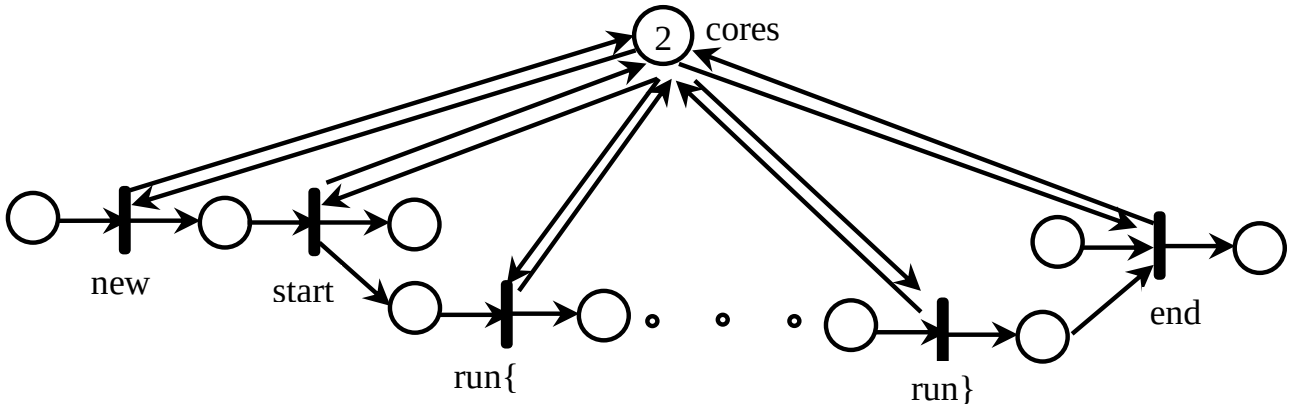


Рисунок 2.1 – Реалізація створення, початку і завершення роботи потоку засобами стохастичних мереж Петрі

Коли два або більше потоків мають доступ до одного розділеного ресурсу, вони потребують забезпечення того, що ресурс буде використаний лише одним потоком одночасно. Процес, за допомогою якого це досягається, називається синхронізацією. Цей процес у багатопоточному програмуванні реалізується за допомогою таких механізмів, як блокування потоку та блок синхронізації дій потоків.

Блокування потоку (locking) широко використовується для синхронізації потоків та описується таким програмним кодом:

```
private final Lock lock = new ReentrantLock();
try{
   lock = lock.tryLock();
   …// synchronized actions
} finally {
    lock.unlock();
}
```

Поставимо у відповідність об’єкту lock стани “lock” і “unlock” . Захват об’єкта lock відбувається, коли він знаходиться у стані “unlock”, а звільнення об’єкта lock відбувається коли він у стані “lock” (рисунок 2.2).

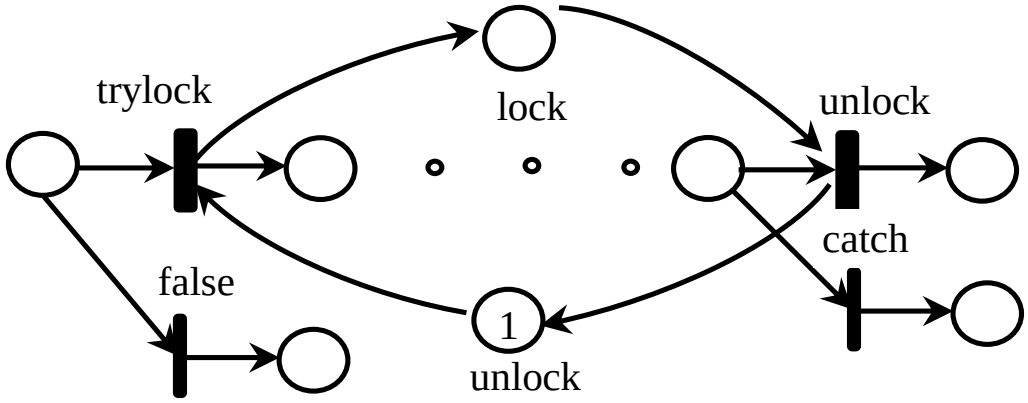


Рисунок 2.2 – Реалізація механізму блокування потоку засобами стохастичних мереж Петрі

Блок синхронізації дій потоку – інший механізм синхронізації, який дозволяє ставити один з потоків у стан очікування доки деяка умова не буде виконана. Інший потік мусить виконати метод-сповіщення “notify” та надіслати сигнал першому потоку, щоб знову перевірити умову очікування. Блок синхронізації дій потоку завжди виконується в тілі блокованого методу. Програмні інструкції виглядають так:

```
    public synchronized void method() throws
InterruptedException{
       while (!condition()) {
         wait();
       }
         …// some actions
       notifyAll();
    }
```

Поставимо у відповідність станам сигналу позиції “signal from another thread” (сигнал від іншого потоку) та “signal to another thread”(сигнал іншому потоку) (рисунок 2.3). Варто відмітити, що альтернативна подія “wait” має вищий пріоритет. Коли кількість потоків у програмі більше двох, маркер надходить до позиції “signal from another thread” від будь-якого потоку, і маркер в позиції “signal to another thread” може бути використаний будь-яким потоком.

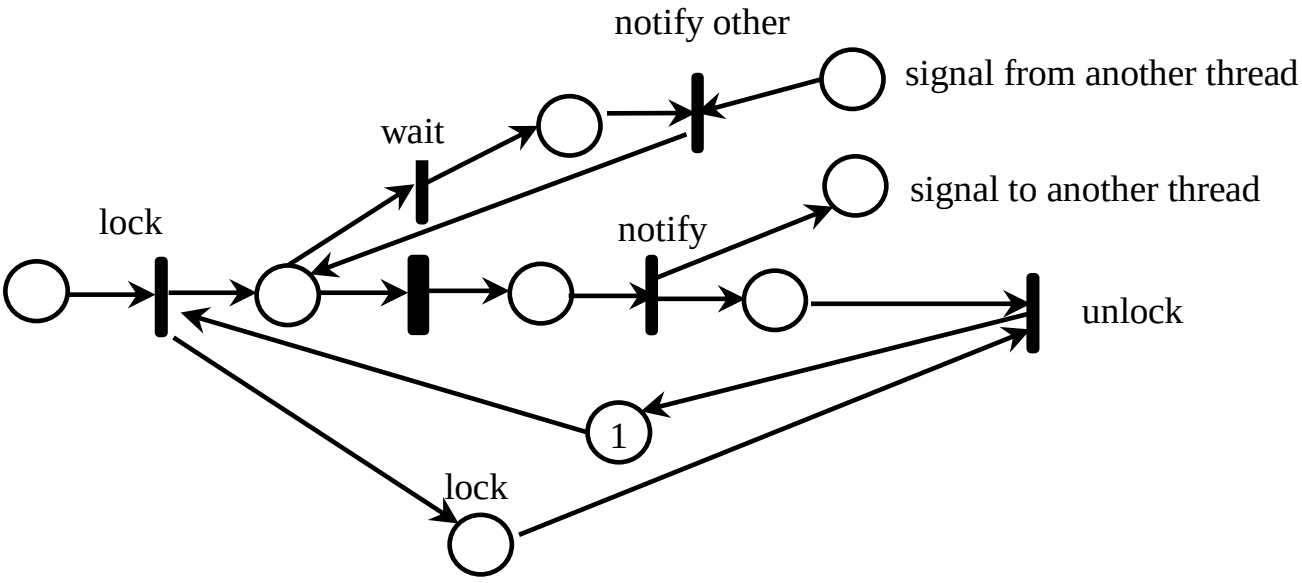


Рисунок 2.3 – Реалізація блоку синхронізації дій потоку засобами стохастичних мереж Петрі

Якщо дані використовується більше ніж одним потоком, це може спричинити проблему доступу потоків до спільних даних. Наприклад, навіть простий метод збільшення значення для спільних даних може виконуватись не правильно. Для коректного функціонування має бути використаний механізм блокування:

```
public synchronized void  incMethod(){
       local++;
}
```

Відповідний фрагмент мережі Петрі містить послідовно події “read” (зчитування значення), “modify” (модифікація значення) та “write” (запис значення). У випадку, коли блокування не використовується, інший потік може у той самий час модифікувати значення спільних даних (рисунок 2.4).

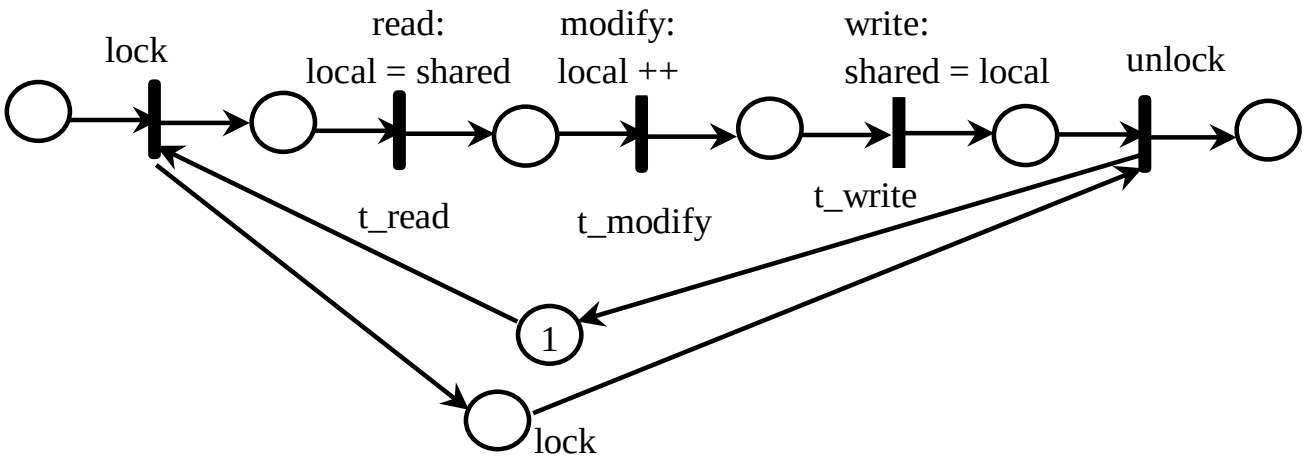


Рисунок 2.4 – Реалізація рішення проблеми доступу до спільних даних потоку засобами стохастичних мереж Петрі

Детальний опис розроблених вище моделей базових фрагментів паралельної програми засобами стохастичних мереж Петрі викладений у статті [35].

2.5 Петрі-об’єктна модель пулу потоків

Пул потоків є найбільш ефективним інструментом для реалізації багатопоточних алгоритмів. Він використовується багатьма популярними мовами програмування, такими як Java, C++, C#, Python і т.д. Будучи високорівневим інструментом багатопоточності, він забезпечує зручність, швидкість і простоту використання. Це дозволяє зменшити накладні витрати на створення потоку і, як наслідок, збільшити ефективність використання ресурсів. Однак досить часто розробники не отримують бажаного результату від використання пулу потоків. Це пов’язано із вибором параметрів для даного інструменту таких, як кількість потоків та кількість завдань. У роботі [36] висвітлюються загальні проблеми використання пулу потоків. У статті [37] автор вказує на важливість налаштування потрібних параметрів пулу потоків.

Пул потоків повторно використовує потоки для виконання поточних завдань. Оскільки потік вже існує, коли надходить запит на виконання, затримка, викликана створенням потоку, пропускається. Це головна перевага використання пулу, що

замість створення нового потоку, будь-яке нове завдання обслуговується вільним потоком з пулу.

Пул потоків складається з робочих потоків, які здатні виконувати обчислювальну роботу. Робота для обчислення в потоці може бути представлена тільки об'єктом класу, який реалізує інтерфейс Runnable або Callable. Такі об'єкти називаються "завданнями" (tasks) відповідно до документації Java [38]. Для того, щоб робочі потоки почали процес обчислення, завдання повинні бути завантажені в пул. Після цього за допомогою ExecutorService завдання можуть бути виконані пулом потоків. Таким чином, для роботи з пулом потоків програміст повинен створити пул, після цього завантажити завдання до пулу, які були створені раніше, після чого припинити завантаження і чекати закінчення обчислення всіх завдань.

Наступний код демонструє створення пулу потоків та завантаження завдань у цей пул:

```
ExecutorService executor =
Executors.newFixedThreadPool(2);
for (int i = 0; i < 5; i++) {
   executor.execute(new TaskCounter(new Counter()));
}
executor.shutdown();
executor.awaitTermination(50, TimeUnit.MINUTES);
```

де TaskCounter – клас, що імплементує інтерфейс Runnable.

Експериментальні результати показують, що прискорення (speedup), отримане при паралелізмі, сильно залежить від обчислювальної складності алгоритму та його параметрів, таких як кількість використовуваних потоків. На рисунку 2.5 зображено зміну прискорення для різної складності обчислень (від 10_3 операцій до 10_9 операцій), коли число потоків і кількість завдань у пулі зростають. Якщо кількість операцій менше 10_6 накладні витрати на створення ресурсів для обчислень більше, ніж скорочення, отримане при паралельній реалізації алгоритму. Тому в цьому випадку паралелізм не є ефективним (на рисунку видно прискорення менше 1.0). Прискорення до майже 2.0 можна спостерігати, якщо тільки кількість операцій сильно перевищує 10_9, а кількість потоків - не менше двох. Однак значення прискорення більше 2.0 не може бути досягнуто через обмеження обчислювальних ресурсів. Таким чином, для

кожного паралельного алгоритму є великим питанням те, які параметри забезпечують його ефективне виконання. Для того, щоб відповісти на це питання, була створена модель пулу потоків. Ця модель дозволяє змінювати кількість потоків і завдань у пулі потоків з урахуванням ресурсів комп'ютера. Замість того, щоб запускати багатопоточні програми на різних комп'ютерах багато разів, можна запускати імітацію моделі пулу потоків. У цьому випадку спрощується пошук правильних параметрів ефективної роботи програми, скорочуються витрати часу та ресурсів.

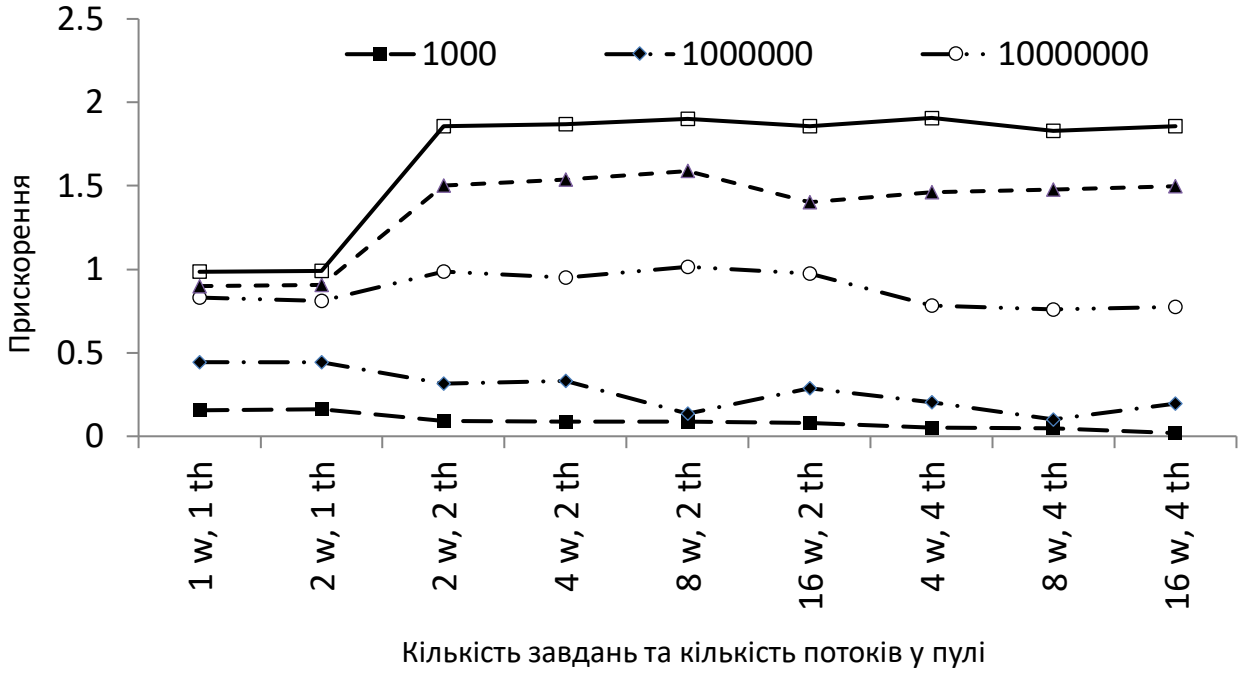


Рисунок 2.5 – Залежність прискорення від кількості потоків (th) та кількості завдань (w) у пулі для різної складності обчислень за результатами роботи багатопоточної програми на двоядерному процесорі.

Розроблені вище моделі базових фрагментів були використані для побудови моделі пулу потоків, одного з високорівневих інструментів багатопоточності. Модель описує пул потоків, з завантаженими воркерами (завданнями), які виконують певний метод певного класу. Для простоти було обрано метод, що збільшує значення, яке міститься в класі з ім'ям Counter. Відповідно до об'єктів програми модель складається з Петрі-об'єктів «Main», «ThreadPool», «Task», «Counter», з’єднаних між собою через спільні позиції мереж.

Елементарна операція програми (наприклад, додавання або множення чисел) представлена переходом з часовою затримкою. Повторювані інструкції можуть бути

представлені одним переходом. Однак слід додати умову повторення. Тому цикл з n простих інструкцій представлений двома переходами, один з яких має більший пріоритет (рисунок 2.6). Таким чином, другий перехід може бути запущеним, коли перший не може. Слід зазначити, що перехід з вищим пріоритетом зображується з більшою шириною.

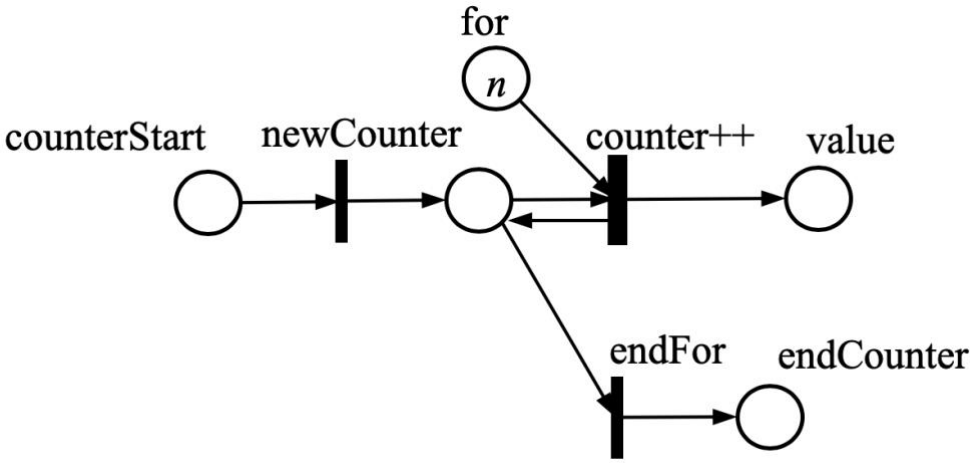


Рисунок 2.6 – Петрі-об’єкт «Counter» без синхронізації

Можна розглядати два випадки: завдання виконуються асинхронно або синхронно. Петрі-об'єкт «Counter» з механізмом синхронізації зображений на рисунку 2.7. Обидві мережі включають одну й ту ж конструкцію з альтернативою подій. Подія інкрементування "counter++", оскільки вона має вищий пріоритет, буде виконана n разів (маркірування в позиції ‘for’), при цьому кожного разу результат операції буде збережено як маркірування позиції "value". Після цього відбувається подія "endFor".

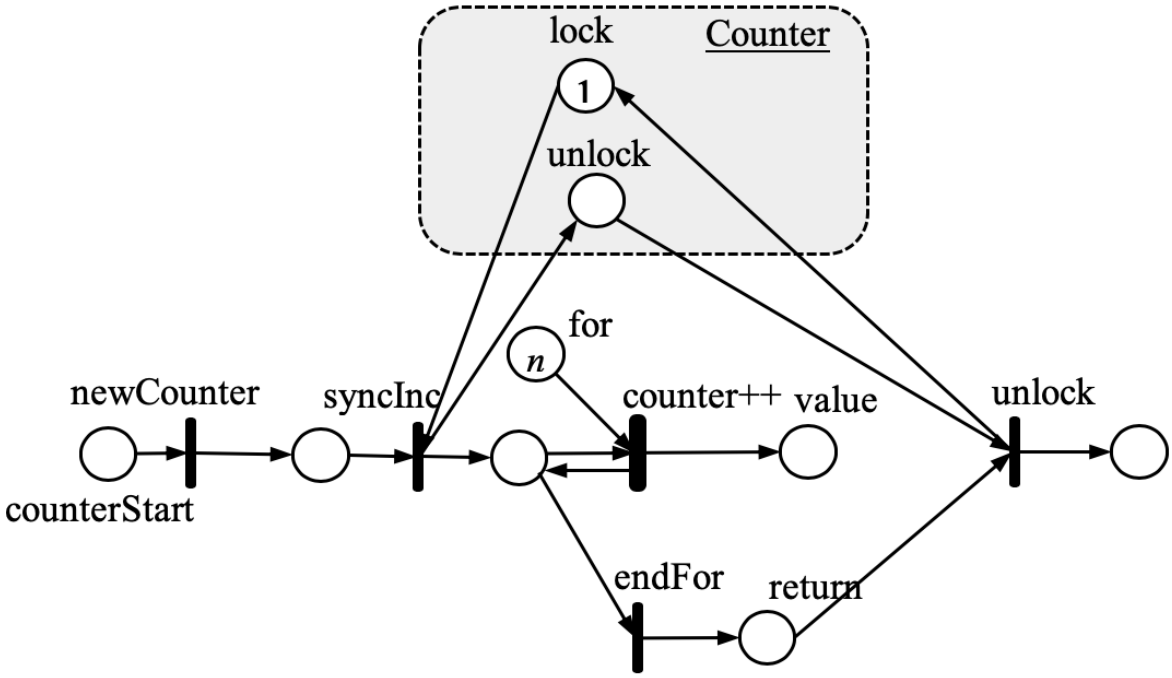


Рисунок 2.7. – Петрі-об’єкт «Counter» із синхронізацією

Мережа Петрі для синхронного лічильника містить механізм блокування – позиція на рисунку. Це означає, що перед тим, як метод інкременту може бути виконаний, об'єкт "lock" блокування повинен бути захоплений. Про це свідчать переходи «syncInc» і «unlock», а також позиції «lock» і «unlock».

Все починається з Петрі-об'єкта «ThreadPool», приєднаного до мережі Петрі «Main» через позиції «poolStart» і «endPool» (рисунок. 2.8). Позиція "cores" імітує ядра процесора як обчислювальний ресурс і є спільною для всіх об'єктів моделі. Більше того, всі події в моделі використовують цю позицію так само, як і об'єкт «Main», але для підвищення читабельності мережі Петрі відповідні елементи будуть упущені в інших Петрі-об’єктах.

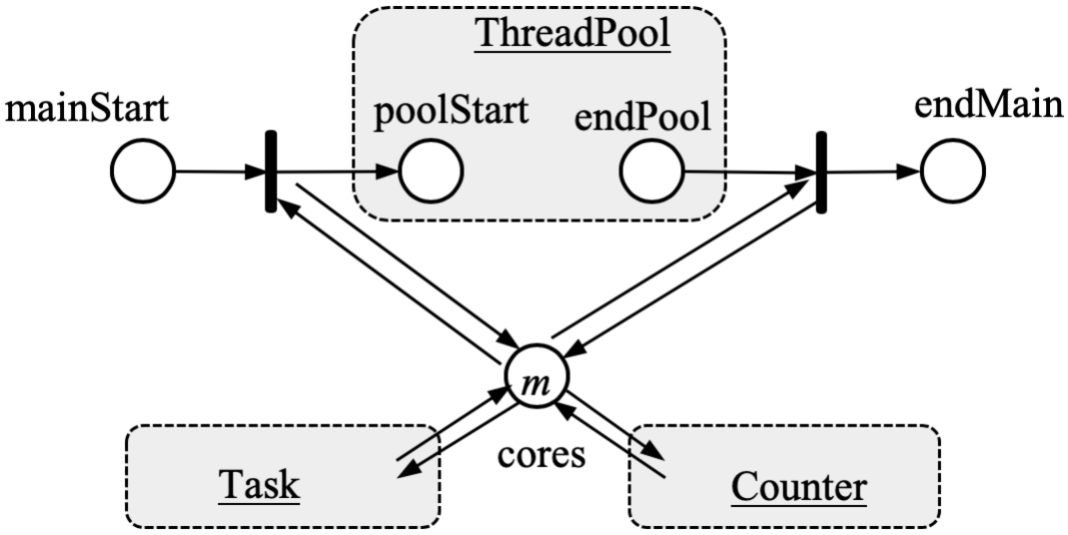


Рисунок 2.8. – Петрі-об’єкт «Main»

Мережа Петрі-об'єкта «ThreadPool», у свою чергу, має зв'язок з Петрі-об'єктом «Task». Вона містить спільні позиції "taskStart" і "endTask", які належать до "Task " (рисунок 2.9). Перехід "newThreadPool" відтворює подію ініціалізації пулу потоків. Вихідна дуга з кратністю k задає кількість потоків у пулі. Рухаючись вперед, виникає альтернатива подій. Конфлікт переходів «execute» і «shutdown» вирішується за допомогою параметра пріоритету. Перехід 'execute' має вищий пріоритет (він зображений з більшою шириною), ніж 'shutdown', тому подія 'execute', яка відповідає завантаженню воркерів (завдань) в пул, буде виконуватися першою, поки буде виконуватися умова входу. Маркірування позиції "numTasks" відповідає кількості воркерів(завдань) у пулі потоків. Коли всі воркери завантажені в пул потоків, виконується подія завершення "shutdown".

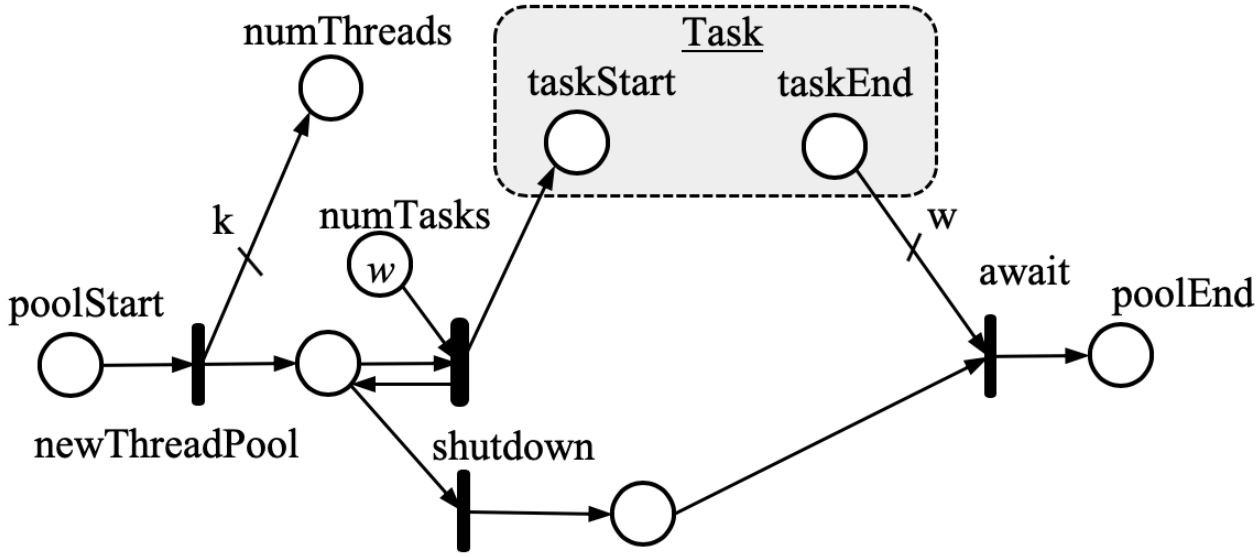


Рисунок 2.9 – Петрі-об’єкт «ThreadPool»

Як тільки воркер завантажується в пул, він починає виконувати роботу. Цей процес описується Петрі-об'єктом «Task» (рисунок 2.10). Загалом, будь-який Runnable-об'єкт з обчислювальним алгоритмом у його методі run() може виконуватися воркером. У нашому випадку воркер виконує метод класу Counter, який з’єднаний з об’єктом "Task" через спільні позиції "counterStart" і "endCounter". Задачу лічильника (інкременту) було обрано через простоту обчислювального алгоритму, який добре паралелізується, оскільки він не має послідовних інструкцій.

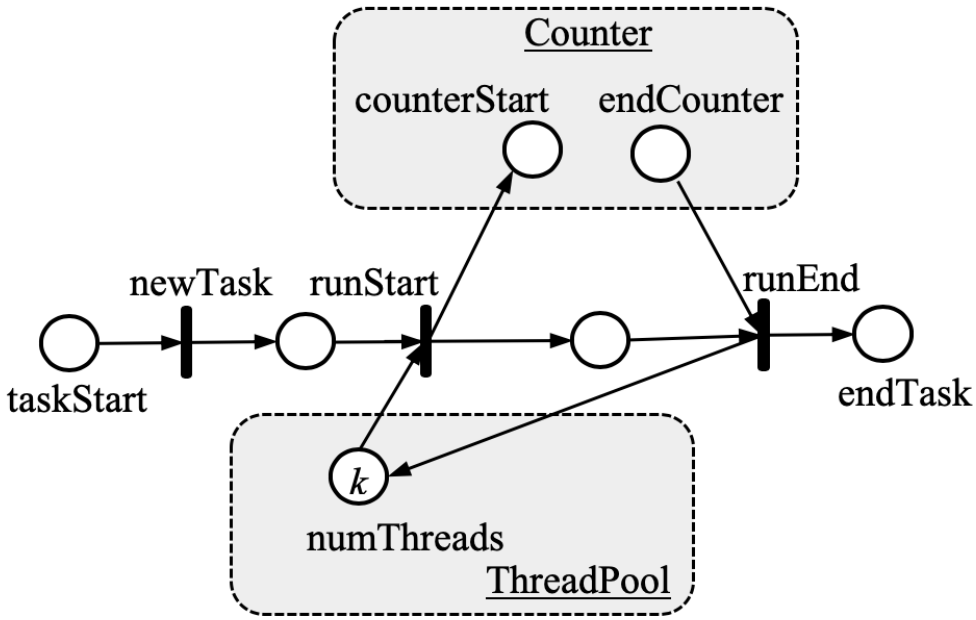


Рисунок 2.10 – Петрі-об’єкт «Task»

Закінчення моделювання відбувається, коли подія "endMain" Петрі-об'єкта "Main" виконана і в позиції "endMain" з’явився один маркер. Момент цієї події реєструється як час завершення багатопоточної програми. Цей час розглядається як результат моделювання в наступних експериментах.

Висновки до розділу

У даному розділі була сформульована змістовна постановка задачі та наведений математичний опис стохастичної мережі Петрі з багатоканальними конфліктними переходами. Представлені перетворення станів мережі Петрі в процесі імітації за допомогою системи логіко-алгебраїчних рівнянь. Описані теоретичні основи Петрі-об’єктного моделювання. Наведені фрагменти стохастичних мереж Петрі, які моделюють основні механізми багатопоточності і які побудовані за відповідними цим механізмам програними кодами. Представлена Петрі-об’єктна модель високорівневого інструменту багатопоточності – пулу потоків.

3 АНАЛІЗ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

3.1 Критерій ефективності використання паралельних обчислень в інформаційній технології

Оцінювання ефективності використання паралельних обчислень на моделі відбувається за критерієм, наведеним у формулі 3.1, що включає в себе час виконання алгоритму та ймовірність успішного виконання алгоритму:

$$K(X) = T(X) \cdot P(X), \tag{3.1}$$

де X – множина параметрів виконання паралельного алгоритму;

$T(X)$ – час виконання алгоритму;

$P(X), = \begin{cases} 1, & \text{якщо алгоритм виконаний успішно} \\ 0, & \text{якщо алгоритм виконаний не успішно} \end{cases}$ – предикат успішності виконання паралельного алгоритму.

3.2 Процес оцінювання ефективності використання паралельних обчислень в інформаційній технології

Оцінювання ефективності використання паралельних обчислень відбувається у 3 кроки:

1) Побудова моделі паралельного алгоритму.

На даному кроці відбувається побудова моделі відповідно до програмного коду алгоритму, а також верифікація моделі.

2) Проведення експериментів з моделлю.

На цьому етапі проводиться ряд експериментів з моделлю шляхом варіювання параметрів моделі. Ґрунтуючись на технічних характеристиках апаратного забезпечення, що використовується для реалізації інформаційної технології, програміст формує множину можливих наборів параметрів Ψ. Для кожного набору параметрів $X \in \Psi$ проводиться n експериментів, в кожному з яких фіксуються час виконання імітації моделі та успішність її виконання, отримуються значення критерію ефективності $K(X_i)$ для кожного експерименту .

3) Оцінювання ефективності.

На останньому кроці здійснюється оцінювання ефективності використання паралельних обчислень в програмі в середньому для кожного набору параметрів на основі експериментальних даних за такою формулою:

$$K_{\text{сер}}(X) = \left(\prod_{i=1}^{n} K\left(X_i\right) > 0\right) \cdot \frac{\sum_{i=1}^{n} K(X_i)}{n}, \quad (3.2)$$

де $\left(\prod_{i=1}^{n} K\left(X_i\right) > 0\right)$ – умова успішності виконання алгоритму в n експериментах;

$\frac{\sum_{i=1}^{n} K(X_i)}{n}$ – середній час виконання імітації моделі;

n – кількість експериментів проведених для кожного набору.

Алгоритм вважається успішно завершеним якщо час його повного виконання є скінченною величиною і відповідно неуспішним якщо час виконання є нескінченністю.

За результатами оцінювання ефективності здійснюється вибір параметрів паралельного алгоритму. Набір параметрів, який забезпечує найменше значення критерію ефективності, є рекомендованим до використання в інформаційній технологій.

3.3 Оцінювання точності моделей паралельних обчислень

З метою оцінки точності моделей було побудовано модель дедлоку та модель доступу до спільних даних та проведено ряд експериментів з ними.

Для побудови моделей та проведення експериментів було використано програмне забезпечення DESS. Для нього було попередньо розроблено бібліотеку для моделювання паралельних програм.

3.3.1 Модель Deadlock

Проблема дедлоку ілюструється задачею про друзів, що кланяються один одному, внаслідок чого можливе виникнення конфлікту потоків. Рішення конфлікту можливе завдяки використанню механізму блокування потоків (локерів). Модель побудована відповідно до програмного коду, який наведений у [34]. Коли хтось із

друзів (об’єкти класу Friend) знаходить іншого зайнятого поклоном (виконує в потоці метод bow, що блокує інший об’єкт класу Friend), він рахує невдалі спроби поклону (невдалі спроби захопити об’єкт “lock”, тобто коли потік переходить у стан очікування). Мережа Петрі, яка відтворює динаміку одного об’єкта класу Friend, зображена на рисунку 3.1. Зареєстровані значення кількості успішних та невдалих спроб наведені в таблиці 3.1. Результати показують сильну залежність ймовірності виникнення конфлікту потоків від часових затримок.

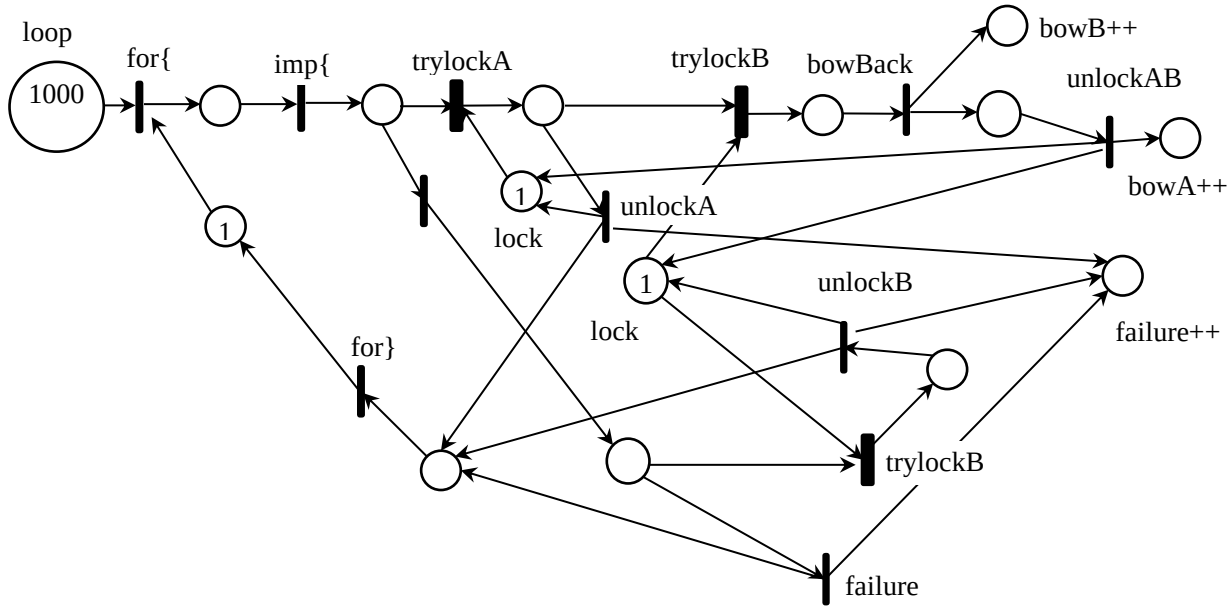


Рисунок 3.1 – мережа Петрі-об'єкта “Friend”

Проведено експеримент, спрямований на вивчення впливу часових затримок переходів. Часовій затримці переходу "for {" було встановлено значення d, а іншим переходам було встановлено значення d · r, де r - співвідношення часових затримок переходів. Коли один об’єкт класу Friend не може захопити об’єкт “lock” іншого об’єкта класу Friend, трапляється невдала спроба, викликана паралельним виконанням потоків. Отже, значення відносної частоти появи невдалої спроби f характеризує частоту виникнення конфліктів потоків. Результати експерименту, зображені у вигляді графіку на рисунку 3.2, вказують на сильну залежність значень f і r. Дійсно, коли часова затримка в переході " for {" значно більша, ніж затримки в інших переходах, це означає, що час, витрачений на дії, значно менший, ніж інтервал

між ними. Отже, ймовірність одночасних дій потоків зменшується. Порівняння результатів моделі та результатів запущеної програми наведено в таблиці 3.1.

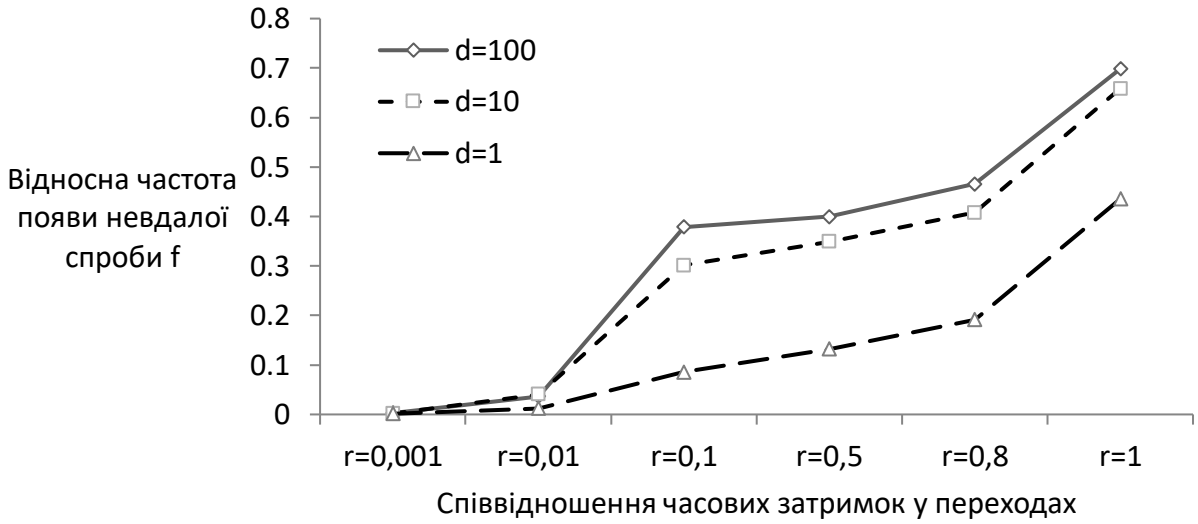


Рисунок 3.2 – Експериментальне дослідження на моделі залежності виникнення конфлікту потоків від значення часових затримок

З графіку видно, що в усіх трьох випадках (d=100, d =10, d=1), ймовірність виникнення конфлікту потоків була мінімальною, коли значення співвідношення часових затримок переходів було мінімальним (тобто часова затримка у переході “for{“ була набагато більше за часові затримки в інших переходах).

Таблиця 3.1. Порівняння результатів моделювання та роботи програми

Кількість потоків	Багатопоточна програма	Імітаційна модель (d=100, r=1.00)	Похибка
2	0.981450	0.978650	0.29%
4	0.959042	0.963417	0.46%

Оскільки значення похибки моделювання дуже мале, менше 0.5%, можна зробити висновок, що точність моделі є досить високою.

3.3.2 Модель доступу до спільних даних

Основна проблема доступу до спільних даних - це невідповідність даних, коли більше, ніж один потік, намагаються одночасно змінювати спільні дані. Дослідження цієї проблеми проводиться на простому прикладі асинхронного запуску лічильника [34]. Якщо в мережі Петрі, представленій на рис.2.4, позиції «lock» та «unlock» будуть

видалені, мережа Петрі буде досліджена для асинхронного запуску потоків лічильника. Арифметичні дії лічильника здійснюються методом класу PetriSim, який виконується до перетворення $D+$ мережі та виконує операції з полями даних Петрі-об'єкта. Значення точності обчислення характеризує конфлікт потоків. Наприклад, якщо k потоків роблять n кроків, то очікуване значення лічильника дорівнює kn. Однак через одночасну асинхронну роботу потоків значення лічильника становить c. Точність обчислення визначається формулою 3.3:

$$\frac{(kn - c)}{(kn)} \cdot 100\%. \qquad (3.3)$$

Результати моделювання та запуску програми представлені в таблиці 3.2.

Таблиця 3.2. Порівняння результатів моделювання та роботи програми

Кількість потоків	Багатопоточна програма	Імітаційна модель (d=100, r=1.00)	Похибка
2	49.173	48.460	1.45%
10	45.336	48.460	6.89%

З таблиці видно, що побудована модель є досить точною.

3.4 Приклад оцінювання ефективності використання пулу потоків

Для прикладу оцінювання ефективності паралельних обчислень в інформаційній технології розглянуто програму, що містить один з механізмів багатопоточності – пул потоків, побудовано відповідну модель та проведені експерименти з моделлю та реальною програмою.

Значення часових затримок для подій в моделі були оцінені статистичним дослідженням відповідних значень під час обчислень на двоядерному процесорі. Існують програмні інструкції, яким потрібно набагато більше часу для їх завершення, ніж інші. Наприклад, час створення пулу потоків приблизно в 20 разів більший за час створення нового об'єкта, який, у свою чергу, у 30000 разів більший за час найпростішої арифметичної операції.

В експерименті досліджується прискорення (speedup), яке досягається в багатопоточній програмі з різними параметрами пулу потоків. Змінюючи обчислювальну складність алгоритму, який розпаралелюється, від 10^3 до 10^9, ми можемо побачити різке збільшення швидкості до приблизно 2.0 лише для складності більше 10^8. Якщо складність менше 10^6, спостерігається прискорення менше 1.0, що означає, що використання пулу в цьому випадку є неефективним (рисунок 3.3). Порівняння результату моделювання та результату, отриманого під час роботи програми на двоядерному процесорі, підтверджує правильність моделі. Модель правильним чином виявила обмеженість обчислювальних ресурсів через те, що швидкість досягає значення 2.0. Слід відмітити, що це значення дорівнює кількості ядер. Пошук параметрів пулу потоків за умови максимального прискорення представлений в таблиці 3.3. Якщо складність алгоритму менше 10^6, використання пулу потоків не рекомендується через низький рівень прискорення, що досягається. У випадку складності алгоритму, що перевищує 10^6, максимальне прискорення досягається, коли в пулі потоків 2 задачі та 2 потоки.

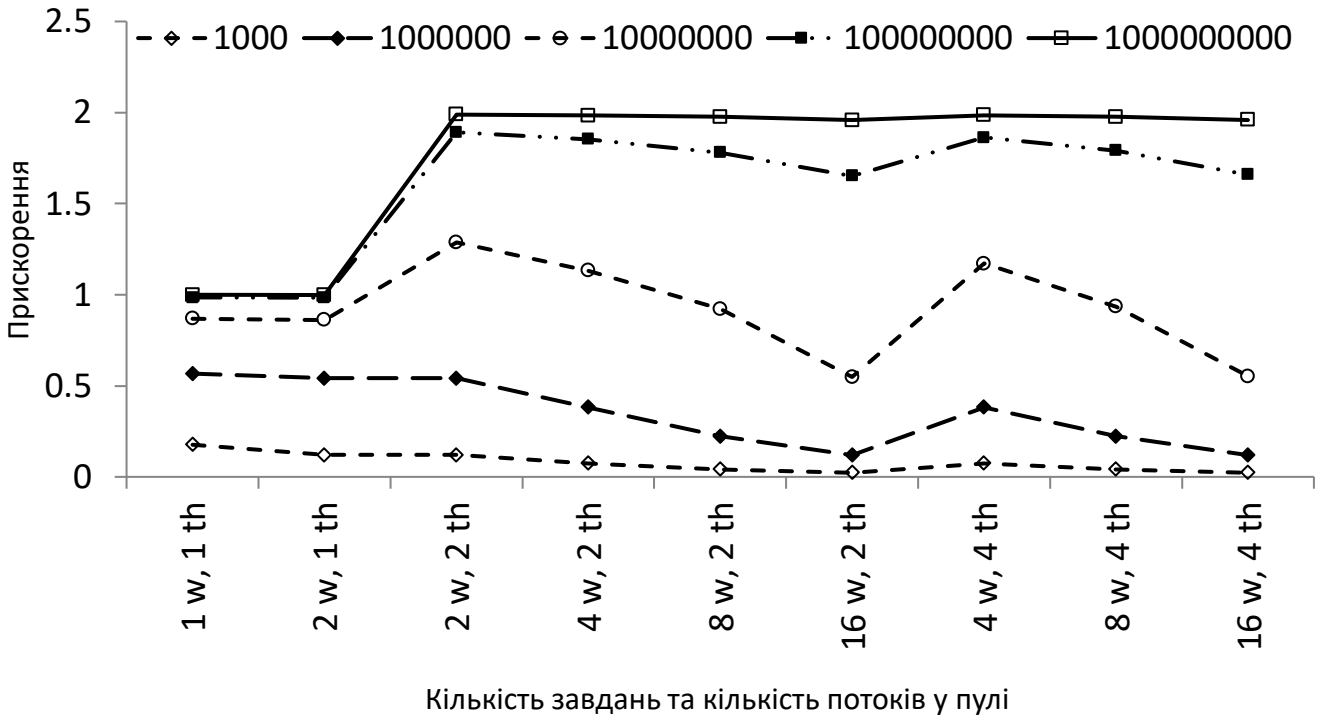


Рисунок 3.3 – Залежність прискорення від параметрів пулу потоків (th – потоки, w - завдання)

Таблиця 3.3. Параметри пулу потоків (th – потоки, w – завдання) для досягнення максимального прискорення

Параметри / Складність	1 w, 1 th	2 w, 1 th	2 w, 2 th	4 w, 2 th	8 w, 2 th	16 w, 2 th	4 w, 4 th	8 w, 4 th	16 w, 4 th
1000	**0.18**	0.12	0.12	0.07	0.04	0.02	0.07	0.04	0.02
1000000	**0.57**	0.54	0.54	0.38	0.22	0.12	0.38	0.22	0.12
10000000	0.87	0.86	**1.29**	1.13	0.92	0.55	1.17	0.93	0.55
100000000	0.98	0.98	**1.89**	1.85	1.78	1.65	1.86	1.79	1.66
1000000000	1.00	1.00	**1.99**	1.98	1.98	1.96	1.99	1.98	1.96

Прискорення, що спостерігається в багатопоточній програмі та в її моделі, за умови наявності 2 потоків та 2 завдань у пулі потоків, зображені на рисунку 3.4. У випадку, коли складність обчислень величезна (10_9), модель точно відтворює прискорення. В іншому випадку модель показує, як правило, більше значення прискорення, ніж багатопоточна програма. Це можна пояснити тим, що в модель не включається використання системних ресурсів іншими процесами. У майбутній версії моделі це може розглядатися як відповідна подія, яка буде додана до моделі.

Точність моделювання прискорення представлена в таблиці 3.4. Значення точності обчислюється як різниця між прискоренням, отриманим в моделі, і реальною багатопоточною програмою, вираженою у відсотках від останнього значення. Кількість завдань w та кількість потоків th варіюються. Зростання абсолютного значення продуктивності в часі, отриманого в моделі та в реальній багатопоточній програмі для 2-х завдань та 2-х потоків у пулі потоків, зображено на рисунку 3.5.

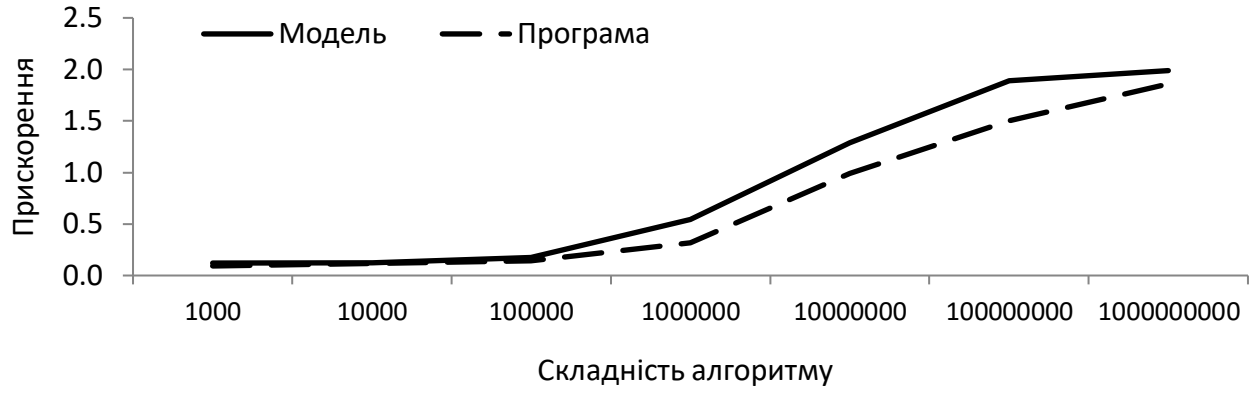


Рисунок 3.4 – Порівняння прискорення в багатопоточній програмі та її моделі

Таблиця 3.4. Точність результатів імітації (складність обчислень – 10^9)

Кількість завдань (w) та потоків (th)	1 w, 1 th	2 w, 1 th	2 w, 2 th	4 w, 2 th	8 w, 2 th	16 w, 2 th	4 w, 4 th	8 w, 4 th	16 w, 4 th
Програма	0.985	0.990	1.859	1.869	1.901	1.856	1.907	1.828	1.855
Модель	0.998	0.998	1.989	1.984	1.976	1.977	1.985	1.977	1.960
Похибка	1%	1%	7%	6%	4%	7%	4%	8%	6%

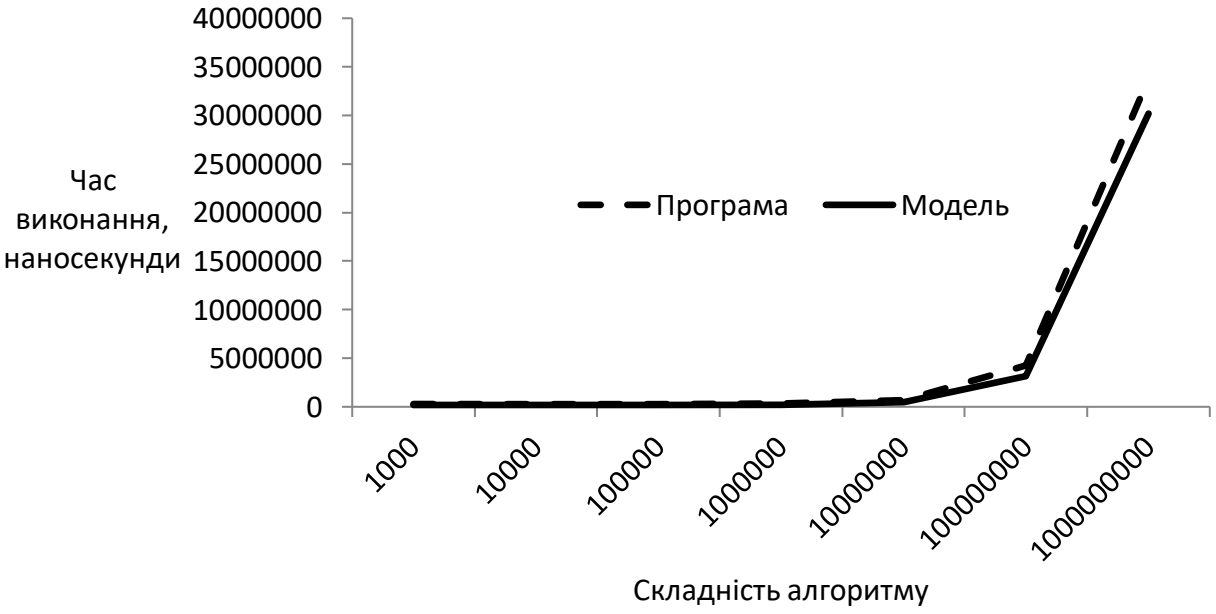


Рисунок 3.5 – Порівняння часу виконання багатопоточної програми та її моделі, за умови наявності 2 завдань та 2 потоків у пулі

Результати дослідження залежності ефективності використання пулу потоків від його параметрів шляхом моделювання викладені у статті [39].

Оцінювання ефективності використання пулу потоків здійснено у 3 кроки згідно з пунктом 3.2. Після побудови моделі та її верифікації, сформовано множину можливих наборів параметрів моделі. До набору входить складність обчислювального алгоритму, що варіюється від 10^3 до 10^9, кількість потоків – від 1 до 4, та завдань – від 1 до 16, а також кількість ядер процесору (обсяг обчислювальних ресурсів), на якому виконувались обчислення, – 2. Для кожного набору параметрів проведено 200 експериментів, в яких зафіксовано час виконання імітації моделі та успішність виконання алгоритму. На основі отриманих результатів експериментів для кожного набору параметрів за формулою 3.2 розраховано середнє значення критерію ефективності. Отримані значення наведені у таблиці 3.5. Видно, що для складності алгоритму менше за 10^6 найменше значення критерію спостерігається при 1 потоці та 1 завданні у пулі. Це свідчить про те, що використання пулу потоків у даному випадку є не ефективним і доцільніше виконати даний алгоритм послідовно, а не паралельно. У випадку ж, коли складність алгоритму більша за 10^6, мінімальне значення критерію ефективності досягається при 2 потоках та 2 завдання у пулі потоків.

Таблиця 3.5. Середнє значення критерію ефективності для кожного набору параметрів (th – потоки, w – завдання).

Параметри / Складність	1 w, 1 th	2 w, 1 th	2 w, 2 th	4 w, 2 th	8 w, 2 th	16 w, 2 th	4 w, 4 th	8 w, 4 th	16 w, 4 th
1000	**0.13**	0.19	0.19	0.31	0.55	1.03	0.31	0.55	1.03
1000000	**0.22**	0.23	0.23	0.32	0.55	1.03	0.32	0.55	1.03
10000000	0.72	0.72	**0.48**	0.55	0.68	1.14	0.53	0.67	1.13
100000000	6.12	6.12	**3.18**	3.25	3.38	3.65	3.23	3.36	3.63
1000000000	60.12	60.12	**30.18**	30.25	30.38	30.65	30.23	30.36	30.63

На основі отриманих значень критерію ефективності можна зробити висновок, що рекомендованим набором параметрів для ефективного використання пулу потоків, за умови проведення обчислень на двоядерному процесорі, є набір: складність алгоритму більше 10_6, 2 потоки та 2 завдання.

Суттєвою перевагою даного способу оцінювання ефективності паралельних обчислень є те, що розрахунок прискорення, на відміну від представленого критерію, потребує замірів часу виконання послідовного алгоритму. Адаптація паралельного алгоритму під послідовне виконання займає багато часу. Окрім цього, іноді паралельний алгоритм не може бути у точності відтворений послідовно. Тому оцінювання ефективності паралельних обчислень через прискорення є не досконалим способом. Запропонований критерій, у свою чергу, розраховується за часом виконання тільки паралельного алгоритму та успішністю виконання алгоритму, і тому є більш зручним способом оцінювання ефективності використання інструментів паралелізму в інформаційній технологій.

Висновки до розділу

У даному розділі наведено критерій оцінювання ефективності використання паралельних обчислень в інформаційній технології, описано процес оцінювання ефективності використання паралельних обчислень в інформаційній технології, який складається з 3 кроків. Експериментально встановлена точність моделей фрагментів паралельної програми. Наведений приклад оцінювання ефективності використання пулу потоків в багатопоточній програмі, в результаті якого виявлено, що максимальне прискорення досягається, коли в пулі потоків 2 завдання та 2 потоки.

4 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

При розробці бібліотеки для моделювання паралельних програм використовувалась мова програмування Java 8 та інтегроване середовище програмування NetBeans. Головною причиною, що зумовила такий вибір став застосунок DESS, реалізований мовою Java, для якого створюється дана бібліотека. Важливими факторами, що вплинули на вибір засобу розробки, також стали відкритість (open source), швидкість та висока продуктивність даної мови програмування.

Java – об’єктно-орієнтована мова програмування, створена Джеймсом Ґослінґом і випущена 1995 року компанією «Sun Microsystems», яка широко використовується для розробки як окремих програм, так і повноцінних застосувань та веб-сервісів [40]. Ця характеристика (об’єктно-орієнтованість) забезпечує модульність розробки програм. Java також є мовою високого рівня, а отже має велику кількість різноманітних бібліотек та зручний синтаксис, що значно полегшує процес розробки програми. Однією з сильних сторін віртуальної машини Java завжди була її здатність з легкістю управляти декількома потоками. JVM оптимізована для великих багатоядерних машин, і вона без проблем може керувати сотнями потоків. Саме якісно реалізований механізм багатопоточності дозволив в даному проекті візуально, шляхом анімації, представити процес імітації моделі в часі. До того ж, у майбутньому планується розвинути розроблену бібліотеку у компонент для автоматизованого тестування багатопоточних Java-програм.

Найбільшою перевагою мови програмування Java є швидкодія виконання обчислювальних задач. За цим критерієм Java є лідером разом із C++, інші мови на кшталт Python, C#, Ruby значно поступаються місцем.

Корисним плюсом є навчальний ресурс, наданий компанією Oracle, що займається мовою Java з 2009 року, - Java Oracle Tutorials, який постійно оновлюється та удосконалюється [34].

4.2 Вимоги до технічного забезпечення

Для правильної роботи даної програми до складу технічних засобів повинні входити:

- комп’ютер із наступною конфігурацією:
  1) процесор з тактовою частотою не нижче 2 ГГц;
  2) достатній об’єм оперативної пам’яті не менше 1 Гб;
  3) простір на диску не менше 130 Мб;
  4) інші складові можуть мати будь-які параметри, тому що вони не значимим чином впливають на роботу програми.
- додатково має бути встановлене таке програмне забезпечення:
  1) операційна система Windows (Windows 10, Windows 8.x (Desktop), Windows 7, Windows Vista);
  2) Java Runtime Environment (не менше 8).
- комп’ютерна периферія, до складу якої входить:
  1) монітор;
  2) мишка;
  3) клавіатура;
  4) системний блок із вище переліченими параметрами.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Діаграми основних класів програмного забезпечення наведені у графічному матеріалі. Опис класів:

- клас PetriP відповідає позиції мережі Петрі кількість маркерів та середнє значення маркірування та містить такі поля і методи:
  1) поле name – назва позиції;
  2) поле mark – кількість маркерів;
  3) поле mean – середнє значення маркірування;
  4) поле listIn – список вхідних дуг;

5) поле listOut – список вихідних дуг;
6) метод increaseMark – збільшує кількість маркерів у позиції на вказане значення;
7) метод setMark – задає кількість маркерів у позиції;
8) метод getMark – повертає кількість маркерів у позиції;

- клас PetriT відповідає переходу мережі Петрі та містить такі поля і методи:
  1) поле name – назва переходу;
  2) поле buffer – кількість значень моментів виходу у переході;
  3) поле timeServ – середнє значення часової затримки;
  4) поле minTime – мінімальне значення часової затримки у буфері;
  5) поле distribution – тип розподілу випадкової величини, якою задається часова затримка;
  6) поле parameter – параметр розподілу;
  7) поле timeOut – список значень моментів виходу у переході;
  8) поле inP – список вхідних позицій;
  9) поле outP – список вихідних позицій;
  10) метод setDistribution – задає тип розподілу, та параметр цього розподілу;
  11) метод generateTimeServ – генерує випадкове число(середнє значення часової затримки);
  12) метод getMinTime – повертає мінімальне значення часової затримки у буфері;
  13) метод actIn – здійснює вхід маркерів у перехід;
  14) метод actOut - здійснює вихід маркерів з переходу;
- клас ArcOut відповідає вихідній дузі, що з’єднує перехід та позицію, містить такі поля та методи:
  1) поле numT – ідентифікатор переходу, з якого виходить дуга;
  2) поле numP – ідентифікатор позиції, в яку входить дуга;
  3) поле k – кратність дуги;

4) метод getNumP – повертає ідентифікатор позиції, в яку входить дуга;
5) метод getNumT – повертає ідентифікатор переходу, з якого виходить дуга;
6) метод getQuantity – повертає кратність дуги.

- клас ArcIn відповідає вхідній дузі, що з'єднує позицію та перехід, містить такі поля та методи:
  1) поле numP – ідентифікатор позиції, з якої виходить дуга;
  2) поле numT – ідентифікатор переходу, в який входить дуга;
  3) поле k – кратність дуги;
  4) метод getNumP – повертає ідентифікатор позиції, з якої виходить дуга;
  5) метод getNumT – повертає ідентифікатор переходу, в який входить дуга;
  6) метод getQuantity – повертає кратність дуги.

Вище зазначені класи є основою для моделювання мережі Петрі. Графічне зображення елементів мережі Петрі задається за допомогою класів GraphPosition, GraphTransition, GraphArc бібліотеки GraphPresentation. Класи, що поєднують в собі графічне зображення елементів і їх функціональність містяться у бібліотеці GraphNet. Для анімаційного представлення функціонування мережі Петрі удосконалені класи бібліотеки GraphPresentation шляхом розробки методів animateP, animateT, animateIn, animateOut класу PetriNetsPanel та впровадження додаткових властивостей lineWidth, color графічних елементів мережі Петрі. Лістинг програми містить у додатку Б.

4.3.2 Діаграма послідовності

Діаграма послідовності наведена у графічному матеріалі для основного потоку подій – побудови моделі паралельної програми. Анотація до діаграми:

- користувач за допомогою панелі шаблонів(зверху) та вбудованих елементів мережі Петрі будує модель на панелі малювання(по середині);
- користувач налаштовує параметри елементів;

- користувач зберігає модель у файл;
- користувач запускає імітацію моделі з анімацією (або без неї) на вказаному інтервалі часу;
- внаслідок імітації виводяться результати на панель справа.

4.3.3 Діаграма компонентів

Бібліотека шаблонів для моделювання паралельних програм засобами стохастичних мереж Петрі функціонує як компонент програмного застосування DESS. Застосування містить такі компоненти:

- бібліотека PetriObj – відповідає за опис структури та функціонування мережі Петрі, Петрі-об’єкта та Петрі-об’єктної моделі;
- бібліотека graphnet – відповідає за графічне представлення елементів мережі Петрі;
- бібліотека graphpresentation – відповідає за графічний інтерфейс застосування;
- бібліотека graphreuse – відповідає за збереження мережі Петрі у файл чи метод та відкриття мережі Петрі з файлу чи методу;
- бібліотека LibNet – містить збережені у методи мережі Петрі.

Діаграма компонентів застосування DESS наведена у графічному матеріалі.

4.3.4 Специфікація функцій

В таблиці 4.1 описаний метод класу PetriNetsFrame, що відповідає за відкриття шаблонів з файлу.

Таблиця 4.1 – Опис методу класу PetriNetsFrame

Метод	**Опис методу**	**Параметри**	**Опис параметрів**
ptrnButtonActionPerformed	Подія, що відбувається при натисканні на іконку шаблону.	evt, fileName	Об’єкт події, ім’я файлу, що містить шаблон.

В таблиці 4.2 описаний метод класу PetriObjModel, що відповідає за запуск імітації з анімацією.

Таблиця 4.2 – Опис методу класу PetriObjModel

Метод	**Опис методу**	**Параметри**	**Опис параметрів**
go	Викликається при запуску імітації з анімацією.	timeModelin, area, panel	Час моделювання, текстова область для виведення результатів, панель на якій відображається мережа Петрі та анімація її функціонування.

В таблиці 4.3 описані методи класу PetriNetsPanel, що відповідають за анімацію графічних елементів мережі Петрі під час імітації.

Таблиця 4.3 – Опис методів класу PetriObjModel

Метод	**Опис методу**	**Параметри**	**Опис параметрів**
animateIn	Запускає для вхідних дуг в перехід, заданий в параметрі, процеси зміни розміру від найменшого до найбільшого і знову до найменшого, та зміни кольору переходу з чорного на червоний і знову на чорний, із часовою затримкою.	tr	Перехід, який виконує вхід маркерів в поточний момент часу.

Продовження таблиці 4.3

animateOut	Запускає для вихідних дуг із переходу, заданого в параметрі, процеси зміни розміру від найменшого до найбільшого і знову до найменшого, та зміни кольору з чорного на червоний і знову на чорний, із часовою затримкою.	eventMin	Перехід, який виконує вихід маркерів в поточний момент часу.
animateP	Запускає для списку позицій, заданого в параметрі, процеси зміни розміру від найменшого до найбільшого і знову до найменшого, та зміни кольору з чорного на червоний і знову на чорний, із часовою затримкою.	inP	Список позицій, в які входять або виходять маркери в поточний момент часу.
animateT	запускає для переходу, заданого в параметрі, процеси зміни розміру від найменшого до найбільшого і знову до найменшого, та зміни кольору з чорного на червоний і знову на чорний, із часовою затримкою.	tr	Перехід, який виконує вхід або вихід маркерів в поточний момент часу.

4.4 Настанова користувача

Для запуску програмного застосування можна скористатися двома шляхами:

- з графічного інтерфейсу подвійним кліком лівою кнопкою миші по файлу DESS.jar;
- з консолі за допомогою команди java -jar DESS.jar.

Головне вікно застосування містить меню користувача, панель інструментів та шаблонів. Головне вікно застосування показано на рисунку 4.1.

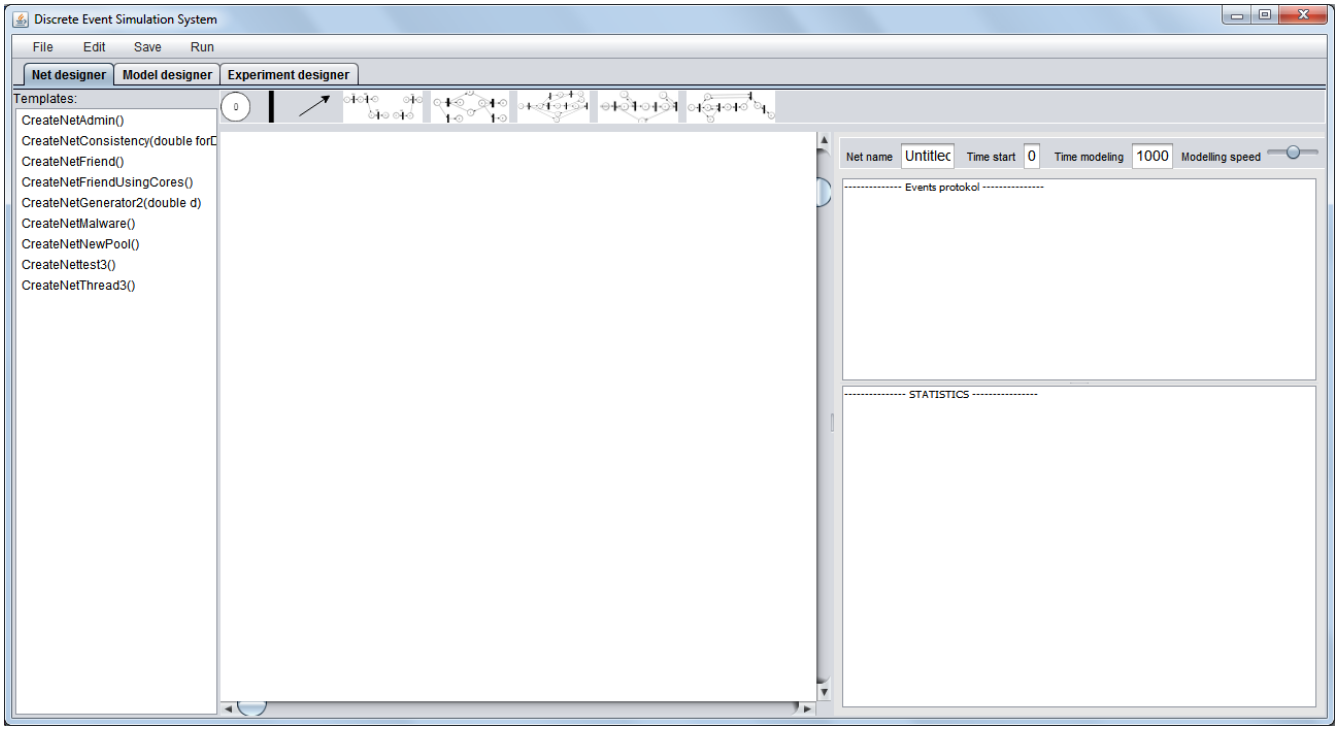


Рисунок 4.1 – Головне вікно застосування DESS

У головному вікні застосування можна виокремити найважливіші елементи інтерфейсу - панель інструментів та шаблонів (рисунок 4.2) та панель параметрів імітації моделі та результатів її виконання (рисунок 4.3).

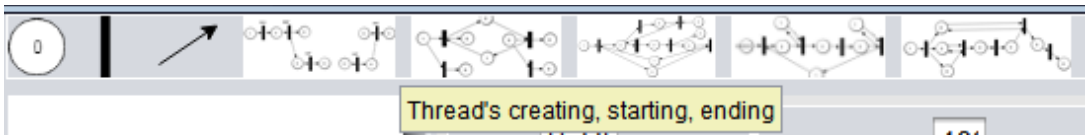


Рисунок 4.2 – Панель інструментів та шаблонів

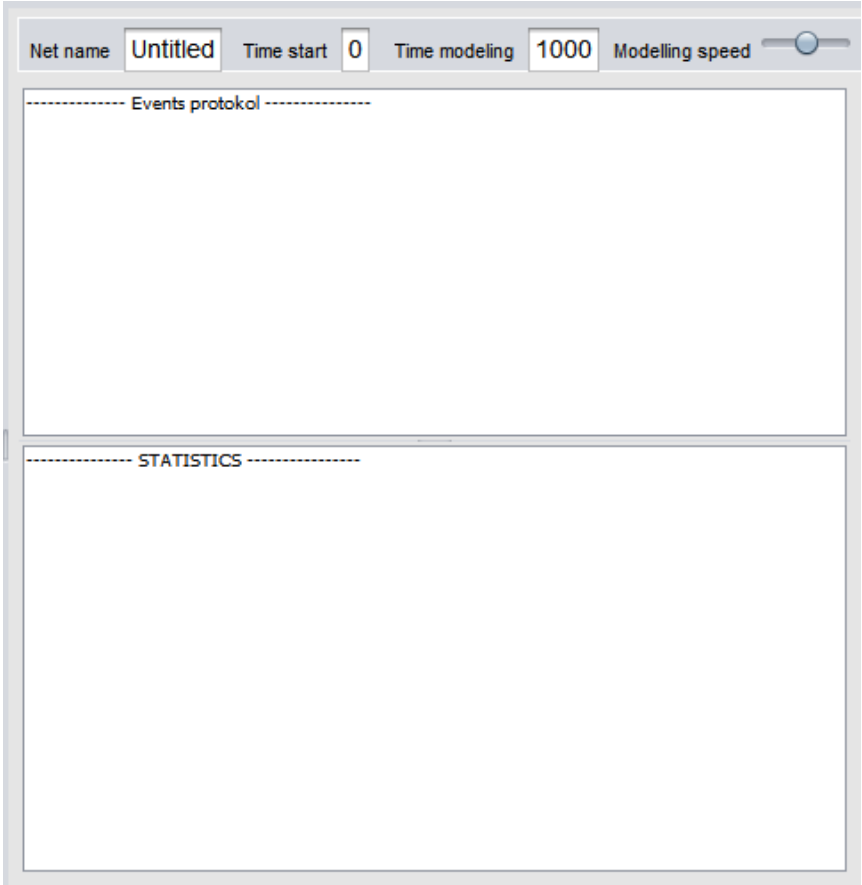


Рисунок 4.3 – Панель параметрів імітації моделі та результатів її виконання

На панелі інструментів та шаблонів містяться основі компоненти для моделювання паралельних обчислень:

- створення позиції;
- створення переходу;
- створення дуги;
- відкриття шаблону «створення, початок і завершення роботи потоку»;
- відкриття шаблону «блокування потоку»;
- відкриття шаблону «блок синхронізації дій потоку»;
- відкриття шаблону «спільний доступ до даних»;
- відкриття шаблону «пул потоків».

Панель параметрів імітації моделі та результатів її виконання дає змогу користувачеві встановити час моделювання, швидкість моделювання, а також переглянути протокол подій, що відбувалися в процесі імітації та статистику по кожному елементу мережі Петрі.

Побудова моделі здійснюється у робочій панелі , що знаходиться в центрі вікна. Використовуючи елементи з панелі інструментів та шаблонів користувач має змогу побудувати модель паралельної програми засобами стохастичної мережі Петрі. Відкриття шаблонів базових конструкцій моделей для паралельних програм та вікно зміни їх параметрів, що викликається шляхом подвійного кліку по елементу мережі, зображені на рисунках 4.4, 4.5.

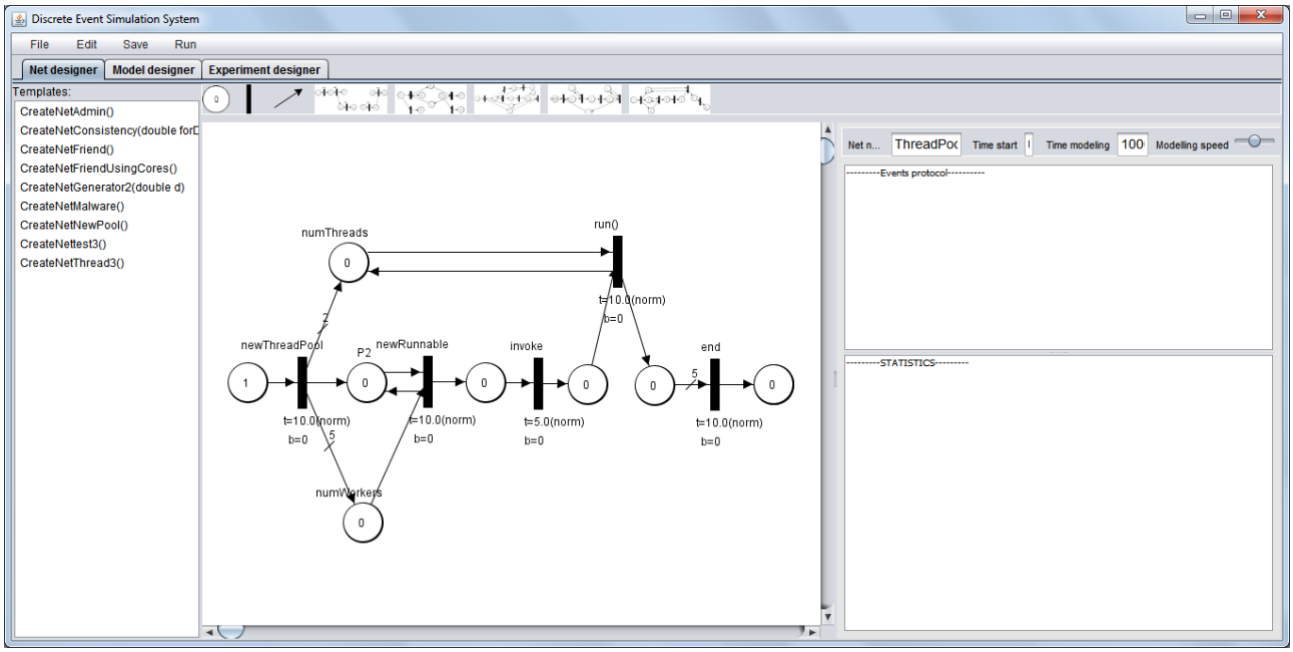


Рисунок 4.4 – Робоча панель з відкритим шаблоном «пул потоків»

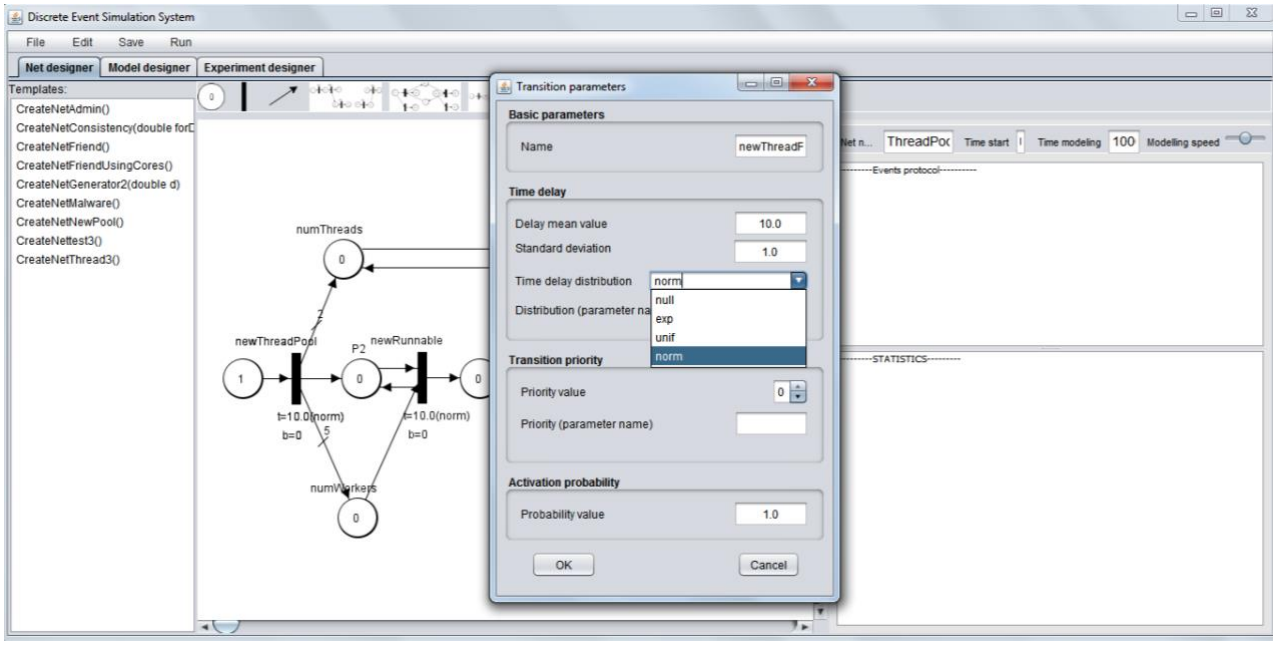


Рисунок 4.5 – Вікно зміни параметрів переходу newThreadPool шаблону «пул потоків»

Запуск імітації моделі відбувається після натискання на кнопку Run у рядку меню зверху. Візуальне представлення зміни стану елементів мережі Петрі у часі в процесі імітації зображено на рисунку 4.6.

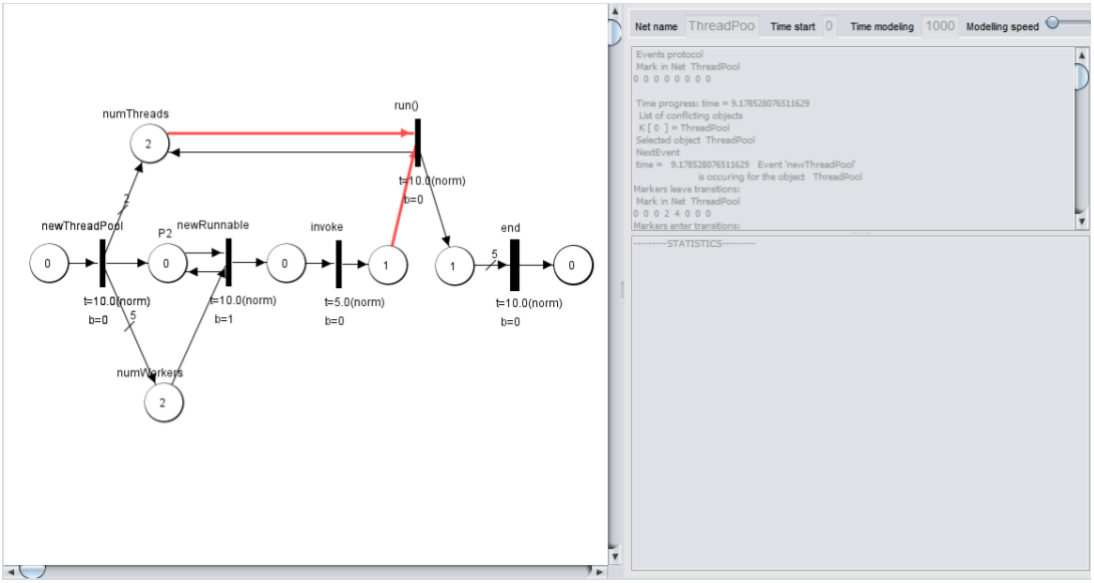


Рисунок 4.6 – Анімація процесу імітації

По завершенню імітації результати виводяться на панель справа у вигляді протоколу подій та статистичних даних по елементам мережі Петрі (рисунок 4.7).

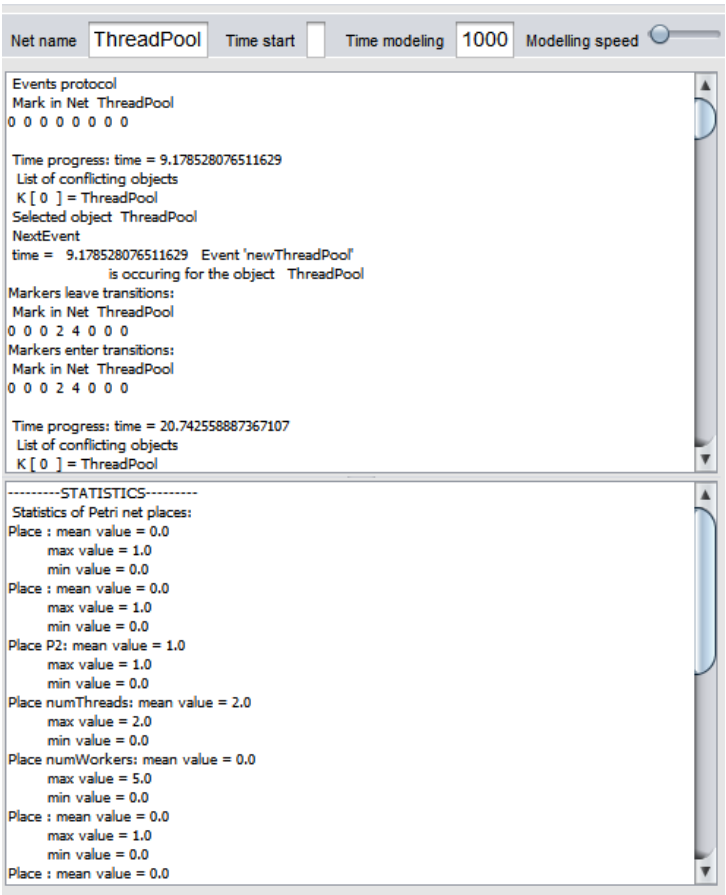


Рисунок 4.7 – Панель результатів імітації

Висновки до розділу

В даному розділі були розглянуті засоби розробки, які використовувались під час створення програмного продукту, а також описані вимоги до технічного забезпечення. Дано опис діаграми класів, діаграми послідовності та описана специфікація функцій класів програмного забезпечення.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

5.1.1 Зміст ідеї

У нижченаведеній таблиці 5.1 описано зміст ідеї стартап-проекту, напрямки його застосування, а також переваги, які отримає користувач від проекту.

Таблиця 5.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення інструменту (фреймворку) для моделювання паралельних обчислень (багатопоточних програм) на основі Петрі-об’єктного підходу моделювання стохастичними мережами Петрі для аналізу ефективності використання паралельних обчислень в інформаційній технології.	1. Проектування багатопоточних програм	Розробник багатопоточних програм матиме змогу на моделі перевірити правильність побудови логіки програми на ранніх етапах її проектування задля уникнення помилок на наступних стадіях розробки.
	2. Тестування багатопоточних програм	Можливість тестування багатопоточної програми з урахуванням обсягу обчислюваних ресурсів, на яких вона має запускатися та часу її функціонування.
	3. Зневадження (відладка) багатопоточних програм	Відстеження помилок у роботі потоків, їх взаємодії та у роботі програми в цілому.

Ідея проекту полягає у розробці методу для моделювання паралельних обчислень, в основі якого лежить Петрі-об’єктний підхід моделювання. Точна модель багатопоточної програми, побудована засобами стохастичних мереж Петрі, даватиме змогу користувачу пришвидшити, полегшити та підвищити якість проектування, тестування та зневадження програми.

5.1.2 Аналіз переваг ідеї

Система має наступних конкурентів:

- CPN tools — інструмент для редагування, моделювання та аналізу кольорових мереж Петрі;
- POSES ++ — система для моделювання та розробки мереж Петрі;
- SoftwareVerify — інструмент для пошуку, виправлення та попередження помилок у програмах. Зневаджувач(debugger) для C, C++, Delphi, Fortran, Visual Basic, та VB.Net. Відстежує тупикові ситуації (deadlock) у взаємодії потоків.

Таблиця 5.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічна характеристика	(потенційні) товари/концепції конкурентів				W (слаб-ка)	N (ней-траль-на)	S (силь на)
		Мій проект	CPN tools	POSES ++	SoftwareVerify			
1	Вартість експлуата-ції	$0 – версія з обмеженим функціоналом $30 – повний функціонала $50 – замовити побудову моделі	–	–	–		+	
2	Безвідмов-ність	Висока	Достатня	Низька	Низька			+
3	Ремонто-придат-ність	Висока, за рахунок використання об’єктно-орієнтованої технології програмуван-ня	–	–	–			+
4	Зручність	+	-	-	-			+
5	Графічний інтерфейс	+	+	-	-		+	
6	Дизайн	Середній	Середній	Низький	–		+	
7	Безпека даних користува-ча	Висока	Висока	Висока	Висока		+	

Продовження таблиці 5.2

8	Наявність моделей-шаблонів	+	-	-	–			+
9	Урахування обсягу обчислювальних ресурсів	+	-	-	-			+
10	Орієнтованість на багатопоточне програмування	+	-	-	-			+

Із наведеної таблиці можна зробити висновок, що система має значні переваги над існуючими конкурентами.

5.2 Технологічний аудит ідеї проекту

Таблиця 5.3. Технологічна здійсненність ідеї проекту

Ідея проекту	**№ п/п**	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
Аналіз ефективності використання паралельних обчислень в інформаційній технології	1.	Java – об’єктно-орієнтована мова програмування	Так	Так
	2.	Netbeans – інтегроване середовище розробки з підтримкою мови Java та можливістю розробки графічного інтерфейсу	Так	Так
	3.	DESS – інструмент для побудови та моделювання стохастичних мереж Петрі	Так	Так
	4.	Петрі-об’єктний підхід – технологія моделювання великих систем, що поєднує в собі об’єктно-орієнтовану технологію та стохастичну мережу Петрі	Так	Так

Вищенаведена таблиця надає можливість зробити висновок про те, що даний стартап-проект може бути реалізований з допомогою наступних технологій: Java, Netbeans, DESS та Петрі-об’єктний підхід, адже вони доступні та наявні на ринку технологій.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Розглянемо деякі особливості ринкового середовища, в якому планується подальше впровадження фреймворку для моделювання паралельних обчислень. На сьогоднішній день існує три головних конкурента через яких може бути ускладнено входження проекту на ринок, а саме: FindBugs, SoftwareVerify, Visual studio built-in tool for debugging threads. Та варто зазначити , що динаміка ринку наразі зростає, а це сприятиме попиту на продукт.

На даний момент обмеження для входу на ринок тестування та зневадження паралельних програм відсутні. Жодних специфічних вимог до стандартизації та сертифікації немає.

Потрібно визначити перелік можливостей, що можуть бути використані для планування ринкового розвитку стартап-проекту. Почнемо з формування потреб потенційних клієнтів (таблиця 5.4) та пропозицій проектів-конкурентів.

Таблиця 5.4. Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільові сегменти ринку**	**Відмінності у поведінці різних груп клієнтів**	**Вимоги споживачів до товару**
1.	Розробка ефективних паралельних програм	1. Компанії, які займаються розробкою великих проектів з інформаційних технологій 2. Компанії, які займаються розробкою графіки та анімації 3. Компанії, які використовують	У різних компаній потреби у моделюванні роботи багатопоточних програм виникають в залежності від специфіки їхньої діяльності. Тобто компанії, що безпосередньо розробляють паралельні програми можуть бути зацікавлені у використанні даного фреймворку на стадіях	1. Простота використання системи.
2.	Проектування багатопоточних програм			2. Візуалізація процесу функціонування паралельної програми.
3.	Мало затратне тестування багатопоточних програм у часі з урахування обсягу обчислювальних ресурсів			3. Наявність моделей-прикладів.

Продовження таблиці 5.4

4.	Зневадження багатопоточних програм	Паралельні обчислення у свої розробках (BigData).	проектування та тестування програм. Проте компанії, сферою діяльності яких є розробка веб-сервісів, зацікавлені лише у тестуванні їх з допомогою фреймворку.	4. Наявність інструкції для допомоги у користуванні системою
5.	Попередження виникнення проблем багатопоточності у програмі			5. Автоматизація процесу створення моделі багатопоточної програми та ведення її параметрів.

Необхідно провести аналіз факторів можливостей стартап-проекту (таблиця 5.5). Слід передбачити також і фактори загроз, що можуть стати на шляху реалізації фреймворку (таблиця 5.6). Фактори в обох таблицях подані в порядку зменшення їхньої значущості.

Таблиця 5.5. Фактори можливостей

№ п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1.	Незадоволені користувачі	Незадоволення потреб користувачів аналогічними системами	Аналіз потреб користувачів та ринкового середовища для виокремлення основних необхідних критеріїв для розробки проекту
2.	Обмеженість продуктів-аналогів	Відсутність в існуючих системах необхідного функціоналу	Реалізація функціоналу, який відсутній в аналогічних системах, та покращення роботи вже існуючого

Продовження таблиці 5.5

3.	Перевантаженість існуючих продуктів	Складність використання продуктів-аналогів та їх незрозумілий інтерфейс	Використання у системі сучасних тенденцій візуалізації та автоматизованість процесу моделювання

Таблиця 5.6. Фактори загроз

№ п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1.	Наявність сильних конкурентів	Незацікавленість користувачів у використанні нової невідомої їм системи	Популяризація проекту серед потенційних користувачів та пошук зацікавлених компаній
2.	Відсутність попиту	Обмеженість сфери можливого використання системи	Реалізація у продукті функціональності для ширшої сфери використання
3.	Невідповідність вимогам ринку	Зміна ситуації на ринку в процесі розробки фреймворку	Перегляд існуючого функціоналу та підлаштування його під поточні потреби користувачів

Надалі необхідно провести аналіз пропозиції: визначити загальні риси конкуренції на ринку (таблиця 5.7).

Таблиця 5.7. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність компанії**
1. Тип конкуренції – олігополія	Домінування на ринку трьох основних компаній, що розробляють системи-аналоги	Вивчення того, що пропонують на ринку конкуренти, та реалізація у

Продовження таблиці 5.7.

		системі можливих переваг над конкурентами
2. Міжнародний рівень конкурентної боротьби	Проект призначено для використання на міжнародному ринку	Орієнтування на задоволення потреб міжнародного ринку
3. Міжгалузева ознака конкуренції	Конкуренція в галузях моделювання та розробки програмного забезпечення	Досконалий аналіз усіх галузей конкуренції та врахування особливостей кожної з них
4. Товарно-видова конкуренція	Конкуренція між аналогічними продуктами	Реалізація більшого переліку можливостей використання продукту порівняно з його аналогами
5. Нецінові конкурентні переваги	Конкуренція за рахунок більш високої якості, чіткішої орієнтованості та надійності продукту	Покращення показників роботи продукту, а також його функціональних переваг
6. Не марочна конкуренція	Марка проекту не відіграє важливу роль, бо це новий та невідомий на ринку стартап-проект	Під час виходу проекту на ринок орієнтуватися не на його торгівельну марку, а на нецінові конкурентні переваги

Фреймворк для моделювання паралельних обчислень у вигляді стартап-проекту має непогані можливості для роботи на ринку з огляду на присутню на цьому ринку конкуренцію. Проект повинен мати ряд сильних сторін, щоб бути конкурентоспроможним на ринку розробки багатопоточних програм. Розглянемо його фактори конкурентоспроможності у таблиці 5.8.

Таблиця 5.8. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування чинників, що роблять фактор для порівняння конкурентних проектів значущим
1.	Моделювання та візуалізація роботи багатопоточної програми	Варіювання обсягу та зайнятості обчислювальних ресурсів, відтворення стохастичної поведінки потоків.
2.	Статистика	Збір статистичної інформації за весь період моделювання
3.	Налаштування параметрів механізмів багатопоточності	Знаходження оптимальних довжин фаз блокування потоків
4.	Зручність використання	Зрозумілий інтерфейс користувача завдяки використанню сучасних засобів розробки програмного забезпечення

На основі визначених факторів конкурентоспроможності проведемо аналіз сильних та слабких сторін фреймворку для моделювання паралельних обчислень. Його результати представлено у таблиці 5.9.

Таблиця 5.9. Порівняльний аналіз сильних та слабких сторін фреймворку для моделювання паралельних обчислень

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні зі стартап-проектом						
			–3	–2	–1	0	+1	+2	+3
1.	Моделювання та візуалізація роботи багатопоточної програми	20							
2.	Статистика	15							
3.	Налаштування параметрів механізмів багатопоточності	18							
5.	Зручність використання	19							

Завершальним етапом є складання SWOT-аналізу на основі проаналізованих вище даних (таблиця 5.10). Перелік загроз та можливостей випливає із відповідних факторів, що наведені у таблицях 5-6.

Таблиця 5.10. SWOT- аналіз стартап-проекту

Сильні сторони:	**Слабкі сторони:**
1. Моделювання та візуалізація роботи багатопоточної програми. 2. Можливість варіювання параметрів механізмів багатопоточності. 3. Врахування зайнятості та обсягу обчислювальних ресурсів. 4. Статистика.	1. Зручність використання. 2. Потреба додаткових знань в області стохастичних мереж Петрі. 3. Відсутність повної автоматизації процесу побудови моделі.
Можливості:	**Загрози:**
1. Заохочення користувачів інших продуктів-аналогів переходити на даний проект завдяки низці його переваг. 2. Покриття більшої сфери застосування даного проекту порівняно з його аналогами. 3. Підвищення попиту на ринку розробки багатопоточних програм.	1. Стартап-проект залишиться непоміченим для користувачів після виходу на ринок. 2. Відсутність потреби у використанні подібного продукту через необхідність наявності додаткових знань. 3. Незадоволення потреб користувачів через невідповідність продукту їх вимогам.

5.4 Розрахунок економічної ефективності інноваційного проекту за методикою ЮНІДО

Таблиця 5.11. Вихідні дані для визначення доходності проекту

N п/п	Показники	Од.вим.	Значення
1.	Витрати на обладнання	$	2000
2.	Термін роботи обладнання після вводу	квартали	8
3.	Гарантований обсяг продажів продукції на квартал	$	30000
4.	Квартальний обсяг замовлень	шт.	60
5.	Валютний депозит	%	17,0
6.	Фактор ризику	%	3,2
7.	Інфляція на валютному ринку	%	9,1

Поточні витрати по роках наведені в наступній таблиці 5.12.

Таблиця 5.12. Поточні витрати на здійснення проекту

Витрати	1 кв	2 кв	3 кв	4 кв	1 кв	2 кв	3 кв	4 кв	Всього
1. Зарплата	4000	4000	4000	4000	4000	4000	4000	4000	32000
2.Соц. нарахування	880	880	880	880	880	880	880	880	7040
3.Амортизація	125	125	125	125	125	125	125	125	1000
4. Інші	2100	2100	2100	2100	2100	2100	2100	2100	16800
Всього	7105	7105	7105	7105	7105	7105	7105	7105	56840

Визначення обсягу грошових потоків (чистого доходу-собівартість) - (чистий дохід-собівартість)*0.18 + амортизації (враховуючи оподаткування прибутку), дол.:

1-й кв = (30000-7105)*0.18 + 125 = 19585.75

2-й кв = (30000-7105)*0.18 + 125 = 19585.75

3-й кв = (30000-7105)*0.18 + 125 = 19585.75

4-й кв= (30000-7105)*0.18 + 125 = 19585.75

1-й кв = (30000-7105)*0.18 + 125 = 19585.75

2-й кв = (30000-7105)*0.18 + 125 = 19585.75

3-й кв = (30000-7105)*0.18 + 125 = 19585.75

4-й кв= (30000-7105)*0.18 + 125 = 19585.75

$26836 – витрати на розробку ПЗ, враховуються протягом 0-го кварталу

Визначення норми дисконтування проекту (d):

d = 17%,

Визначення чистого дисконтованого доходу(ЧДД) та чистої поточної вартості (ЧПВ)

$$ЧДД = \sum_{t=0}^{n} \frac{(D_t - K_t)}{(1+d)^t},$$

де D_t - чисті доходи *t*-го періоду;

K_t - витрати *t*-го періоду.

Таблиця 5.13. Розрахунок чистого дисконтованого доходу проекту

Квартал	*D*	*K*	$\frac{1}{(1+d)_t}$	$\frac{D}{(1+d)_t}$	$\frac{K}{(1+d)_t}$	*ЧДД*	*ЧПВ*
0	0	26836	1,00	0	26836	-26836	-26836
1	19585.75	0	0.85470085	16739.96	0	16739.96	-10096.04
2	19585.75	0	0.73051355	14307.66	0	14307.66	4211.61
3	19585.75	0	0.62437056	12228.77	0	12228.77	16440.38
4	19585.75	0	0.53365005	10451.94	0	10451.94	26892.32
1	19585.75	0	0.45611115	8933.28	0	8933.28	35825.59
2	19585.75	0	0.38983859	7635.28	0	7635.28	43460.88
3	19585.75	0	0.33319538	6525.88	0	6525.88	49986.76
4	19585.75	0	0.28478237	5577.68	0	5577.68	55564.43
Всього:	156686	26836	x	82400.43	26836	55564.43	x

Визначення терміну окупності проекту

Термін окупності проекту ($Т_{ок}$) визначається на підставі попередніх розрахунків ЧДД та ЧПВ (табл. 5.13):

$$Т_{ок} = р + ЧПВ_р / ЧДД_{р+1},$$

де р - останній квартал, коли ЧПВ < 0;

$ЧПВ_р$ - значення ЧПВ в р-му кварталі (без мінусу);

$ЧДД_{р+1}$ - значення ЧДД в (р+1)-му кварталі.

$$Т_{ок} = 1 + 10096.04/14307.66 = 1,71 \text{ (квартала)}.$$

Термін окупності проекту дорівнює двом кварталам першого року.

Індекс доходності та середньорічна рентабельність проекту:

Індекс доходності (ІД) - це відношення сумарного дисконтованого доходу до сумарних дисконтованих витрат.

$$ІД = \sum_{t=0}^{n} \frac{D_t}{(1+d)^t} / \sum_{t=0}^{n} \frac{K_i}{(1+d)^t} = \frac{82400.43}{26836} = 3,07.$$

Повинно витримуватись співвідношення ІД>1. Оскільки 3,07 >1, то по цьому показнику проект можна рекомендувати до впровадження.

Тоді середньорічна рентабельність проекту (R) буде:

$$R = \frac{ІД}{n} \times 100\% = \frac{307}{8} = 38,4\%.$$

Таким чином, даний проект є високорентабельним. За два квартали компанія зможе розрахуватись з інвестором, а наступні шість - буде працювати на власний прибуток.

Висновки до розділу

У даному розділі було описано зміст ідеї стартап-проекту, проведено аналіз ринкових можливостей запуску стартап-проекту. Результати розрахунку економічної ефективності інноваційного проекту показали, що проект є високорентабельним.

ЗАГАЛЬНІ ВИСНОВКИ

При виконанні магістерської дисертації проаналізовано предметне середовище паралельних обчислень. Розв’язане важливе науково-прикладне завдання – підвищення ефективності використання паралельних обчислень в інформаційній технології. Розроблений метод Петрі-об’єктного моделювання паралельних обчислень, за допомогою якого побудовано моделі основних механізмів багатопотоковості. Дані моделі утворюють бібліотеку шаблонів базових фрагментів багатопотокових програм, яка полегшує і пришвидшує побудову моделі паралельної програми. Запропонований критерій ефективності використання паралельних обчислень, який розраховується за результатами моделювання паралельної програми. Критерій дозволяє оцінити ефективність використання того чи іншого інструменту багатопотоковості та визначити набір параметрів алгоритму, який забезпечує мінімальний час виконання алгоритму та ефективне використання обчислювальних ресурсів.

З огляду на актуальність використання паралельних обчислень в інформаційних технологіях варто зазначити, що нерідко багатопотоковість застосовується без обґрунтування та досконалого дослідження ресурсів, на яких виконується розпаралелювання. З цієї причини часто спостерігається не ефективне використання інструментів паралельного програмування, коли не досягається ефект у вигляді прискорення, що очікувався. Або ж спостерігається не ефективне використання ресурсів, тобто ресурсів використовується більше, ніж необхідно, і прискорення можна отримати при менших ресурсних витратах. Аналіз ефективності використання паралельних обчислень, який проводиться у даній роботі, корисний також з точки зору вибору способу розпаралелювання на етапі проектування інформаційної технології.

Використовуючи розроблений метод побудовано модель високорівневого механізму багатопотоковості – пулу потоків – з метою оцінки ефективності його використання. Результати експериментів проведених на моделі та на реальній програмі підтверджують коректність методу. Похибка результатів, що складає менше 9%, свідчить про точність побудованої моделі. За результатами моделювання

розраховано критерій ефективності, в наслідок чого виявлено залежність ефективності роботи пулу потоків від його параметрів (складність обчислювального алгоритму, кількість потоків і кількість завдань) та обсягу обчислювальних ресурсів. На основі значення критерію ефективності визначено набір параметрів пулу потоків, який гарантує максимальне прискорення на двоядерному процесорі: складність обчислювального алгоритму більша за 10_6, 2 потоки і 2 завдання в пулі. При інших параметрах використання пулу потоків є не ефективним. Результати запуску реальної програми підтверджують виявлену залежність.

Представлена настанова користувача, в якій наведено основні етапи моделювання багатопоточної програми за допомогою розробленого методу Петрі-об’єктного моделювання паралельних обчислень.

У майбутньому планується удосконалювати проект таким чином, щоб досягти повної автоматизації процесу побудови моделі за програмним кодом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Trümper J., Bohnet J., Döllner J. Understanding complex multithreaded software systems by using trace visualization // SOFTVIS '10 Proceedings of the 5th international symposium on Software visualization. – ACM, Salt Lake City, Utah, USA, 2010. – P.133-142.
2. Singh R. An Optimized Task Duplication Based Scheduling in Parallel System // International Journal of Intelligent Systems and Applications. – 2016. – №8 (8). – P.26-37.
3. Holub A. Taming Java Threads. – Apress, Berkeley, CA, USA, 2000. – 300 p.
4. Praun C. Detecting Synchronization Defects in Multi-Threaded Object-Oriented Programs, PhD thesis / Swiss Federal Institute of Technology. – Zurich, 2004.
5. Grama A., Gupta A., Karypis G., Kumar V. Introduction to parallel computing: design and analysis of algorithms (2nd Edition). – Pearson, 2003.– 656 p.
6. Lea D. Concurrent Programming in Java: Design Principles and Patterns, Second Edition. – Addison Wesley, Boston, 1999. – 422 p.
7. Edward A. Lee. The Problem with Threads. EECS at UC Berkeley [Електронний ресурс] // Режим доступу: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-1.html
8. Owe O., Yu I.C. Deadlock detection of active objects with synchronous and asynchronous method calls // Norsk Informatikkonferanse (NIK) OPJ/PKP. – Halden, Norway, 2014.
9. Software Verify LTD. Smarter tools for better software [Електронний ресурс] // Режим доступу: https://www.softwareverify.com/thread-analysis-deadlock-detection.php
10. Chen Z., Dinan J., Tang Z., Balaji P., Zhong H., Wei J., Huang T., Qin F. MC-Checker: Detecting Memory Consistency Errors in MPI One-Sided Applications // SC14, Institute of Electrical and Electronics Engineers (IEEE). – New Orleans, Loui-siana, USA, 2014.
11. Peterson J. Petri Nets Theory and the Modelling of Systems. – Prentice-Hall, New Jersey, 1981. – 241 p.

12. Liao H., Wang Y., Cho H., Stanley J., Kelly T., Lafortune S., Mahlke S., Reveliotis S. Concurrency Bugs in Multithreaded Software: Modeling and Analyzing Using Petri Nets // Discrete Event Dynamic Systems. – 2013. – №23(2). – P.157-195.
13. Xiang D., Liu G., Yan C., Jiang C. Detecting Data Inconsistency Based on the Unfolding Technique of Petri Nets // IEEE Transactions on Industrial Informatics. – 2017. – №13(6). – P.2995-3005.
14. Friesen J. Java Threads and the Concurrency Utilities. – Apress, New York, 2015. – 200 p.
15. Kavi K., Moshtaghi A., Chen, D. Modeling Multithreaded Applications Using Petri Nets // International Journal of Parallel Programming. – 2002 . – №30(5). – P.353-371.
16. Katayama T., Kitano S., Kita Y., Yamaba H., Okazaki N. Proposal of a Supporting Method for Debugging to Reproduce Java Multi-threaded Programs by Petri-net // Journal of Robotics, Networking and Artificial Life. –2014. – №1(3). – P.207-211.
17. Haas P. Stochastic Petri Nets: Modelling, Stability, Simulation. – Springer-Verlag New York, New York, 2002. – 510 p.
18. Goetz B., Peierls T., Bloch J., Bowbeer J., Holmes D., Lea D. Java Concurrency in Practice. – Addison-Wesley, Boston , 2006. – 384 p.
19. GPC mbH - Poses++ [Електронний ресурс] // Режим доступу: http://www.gpc.de/e_poses.html
20. CPN Tools – A tool for editing, simulating, and analyzing Colored Petri nets [Електронний ресурс] // Режим доступу: http://cpntools.org/
21. Стеценко І.В. Дослідження дискретно-подійних систем з використанням технології Петрі-об'єктного моделювання // Управляючі системи та машини. – Київ, 2014. – №5 (253). – С.77-85.
22. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учебное пособие – Н. Новгород, ННГУ, 2001.
23. Amdahl G. The validity of the single processor approach to achieving large-scale computing capabilities // Proceedings of AFIPS Spring Joint Computer Conference. – 1967. – P.483-485.

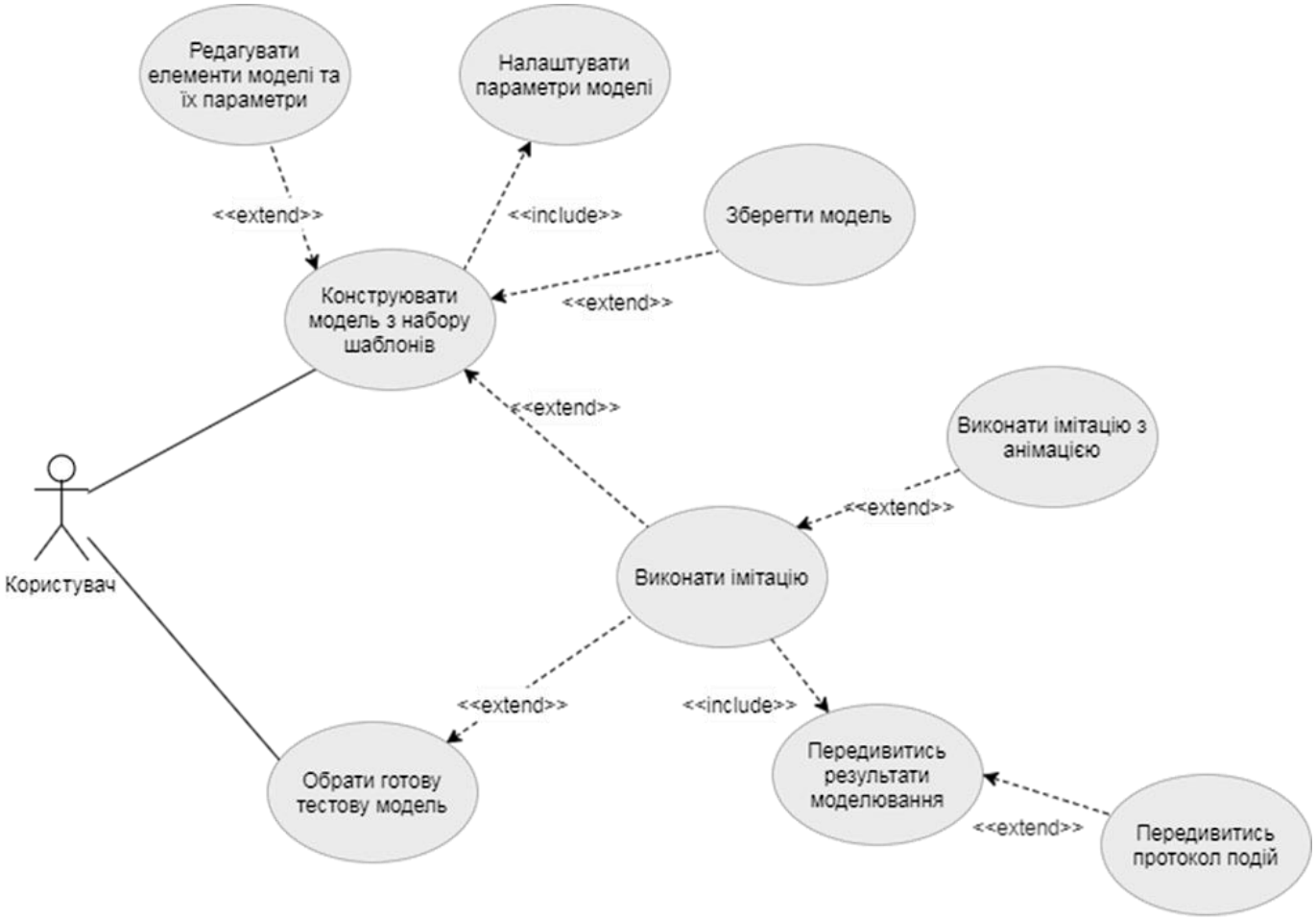
24. Gustafson J.L. Reevaluating Amdahl's law // Communications of the ACM. – 1988. – №31. – P.532-533.
25. Стеценко І.В., Дифучина О.Ю. Моделювання паралельних обчислень стохастичними мережами Петрі // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2017. – № 66. – С.27-31
26. Стеценко І.В., Лещенко К.С. Інтелектуальний компонент візуального програмування стохастичних мереж Петрі // Технічні науки та технології: науковий журнал. – 2016. – № 4 (6). – С.139-147.
27. Stetsenko I.V. State equations of stochastic timed petri nets with informational relations // Cybernetics and Systems Analysis. – 2012. – №5 (48). – P.784-797.
28. Zaitsev, D., Sleptsov, A.: State Equations and Equivalent Transformations of Timed Petri Nets // Cybernetics and Systems Analysis. – 1997. – №33(5). – P.659-672.
29. Стеценко І.В. Моделювання систем: навч. посіб. [Електронний ресурс, текст]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2010. – 399 с.
30. Стеценко И.В. Алгоритм имитации Петри-объектной модели // Математичні машини і системи. – 2012. – № 1. – С.154-165.
31. Стеценко І.В., Дифучина О.Ю. Складність алгоритму розробки моделі дискретно-подійної системи в середовищі візуального програмування. // Математичне та імітаційне моделювання систем. МОДС 2017: тези доповідей Дванадцятої міжнародної науково-практичної конференції (м. Київ – с. Жукін, 26 червня – 29 червня 2017 р.) / М-во осв. і наук. України, Нац. Акад. наук України та ін. – Чернігів: ЧНТУ, 2017 – С.312-316.
32. Стеценко І.В., Дифучина О.Ю. Програмне забезпечення моделювання дискретно-подійних систем // Інформаційні технології розвитку освіти: тези п'ятої Міжнародної науково-практичної конференції «Управління розвитком технологій», КНУБА(30-31 березня 2018, м. Київ) – С.97-98.
33. Дифучина О.Ю. Тестування паралельних програм на моделях. // Математичне та імітаційне моделювання систем. МОДС 2018: тези доповідей Тринадцятої міжнародної науково-практичної конференції (м. Київ – с. Жукін, 25 червня – 29

червня 2018 р.) / М-во осв. і наук. України, Нац. Акад. наук України, Академія технологічних наук України, Інженерна академія України та ін. – Чернігів: ЧНТУ, 2018 – Р.231-234.

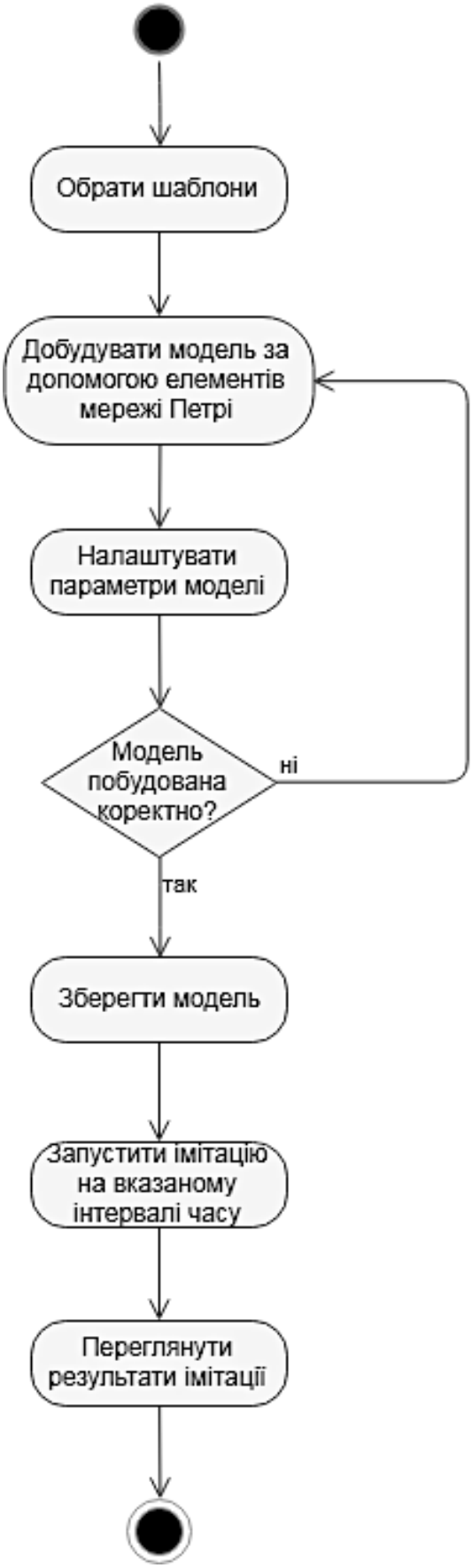
34. The Java™ Tutorials - Oracle Docs [Електронний ресурс] // Режим доступу: https://docs.oracle.com/javase/tutorial/.
35. Stetsenko I.V., Dyfuchyna O. Simulation of multithreaded algorithms using Petri net// Advances in Computer Science for Engineering and Education. ICCSEEA 2018. Advances in Intelligent Systems and Computing. – 2019. – vol 754. – P.391-401.
36. Goetz B. Java theory and practice: thread pools and work queues. [Електронний ресурс] // Режим доступу: https://www.ibm.com/developerworks/library/j-jtp0730/j-jtp0730-pdf.pdf
37. Pepperdine K. Tuning the size of your thread pool. [Електронний ресурс] // Режим доступу: https://www.infoq.com/articles/Java-Thread-Pool-Performance-Tuning
38. Java™ Platform, Standard Edition 8 API Specification. Class ThreadPoolExecutor[Електронний ресурс] // Режим доступу: http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html
39. Stetsenko I.V., Dyfuchyna O. Thread Pool Parameters Tuning Using Simulation// Advances in Computer Science for Engineering and Education II ICCSEEA 2019. Advances in Intelligent Systems and Computing. – 2020. – vol 938. – P.78-89.
40. Horstmann C., Cornell G. Core Java 2, Volume I: Fundamentals (6th Edition). – Prentice Hall, New Jersey, 2002. – 752 p.
41. Дифучина О.Ю. Критерій ефективності використання паралельних обчислень / О.Ю. Дифучина, І.В. Стеценко // Матеріали III всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С.6-8.

ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ

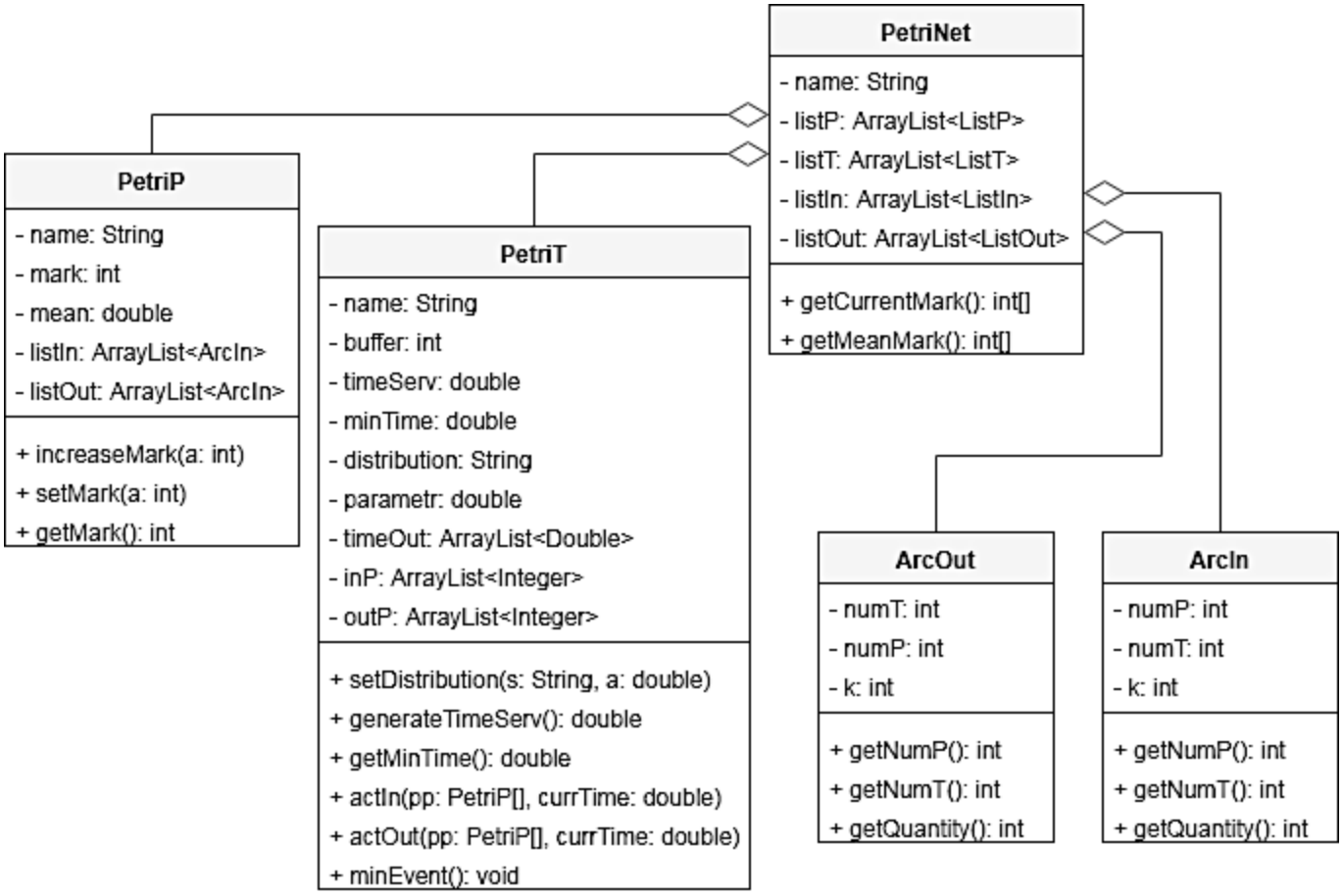
ПЛАКАТ 1 Схема структурна варіантів використання



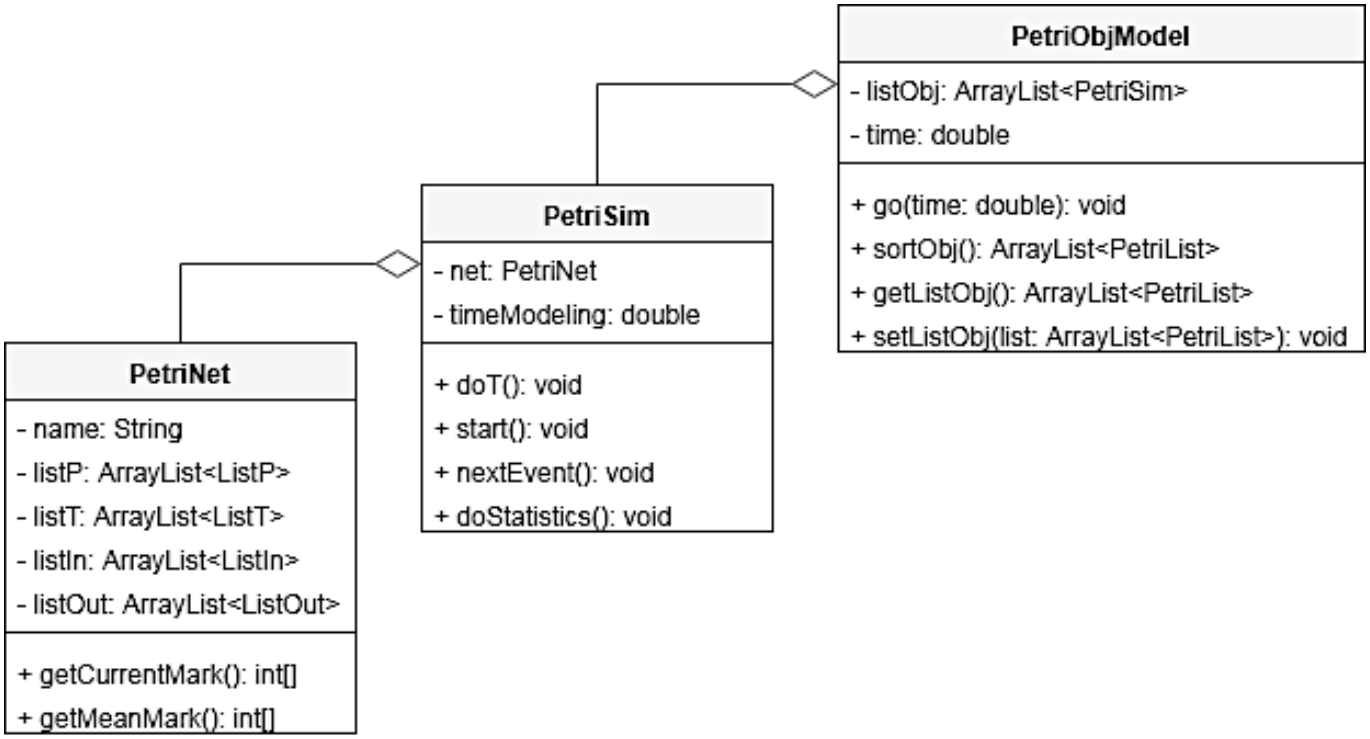
ПЛАКАТ 2 Схема структурна діяльності



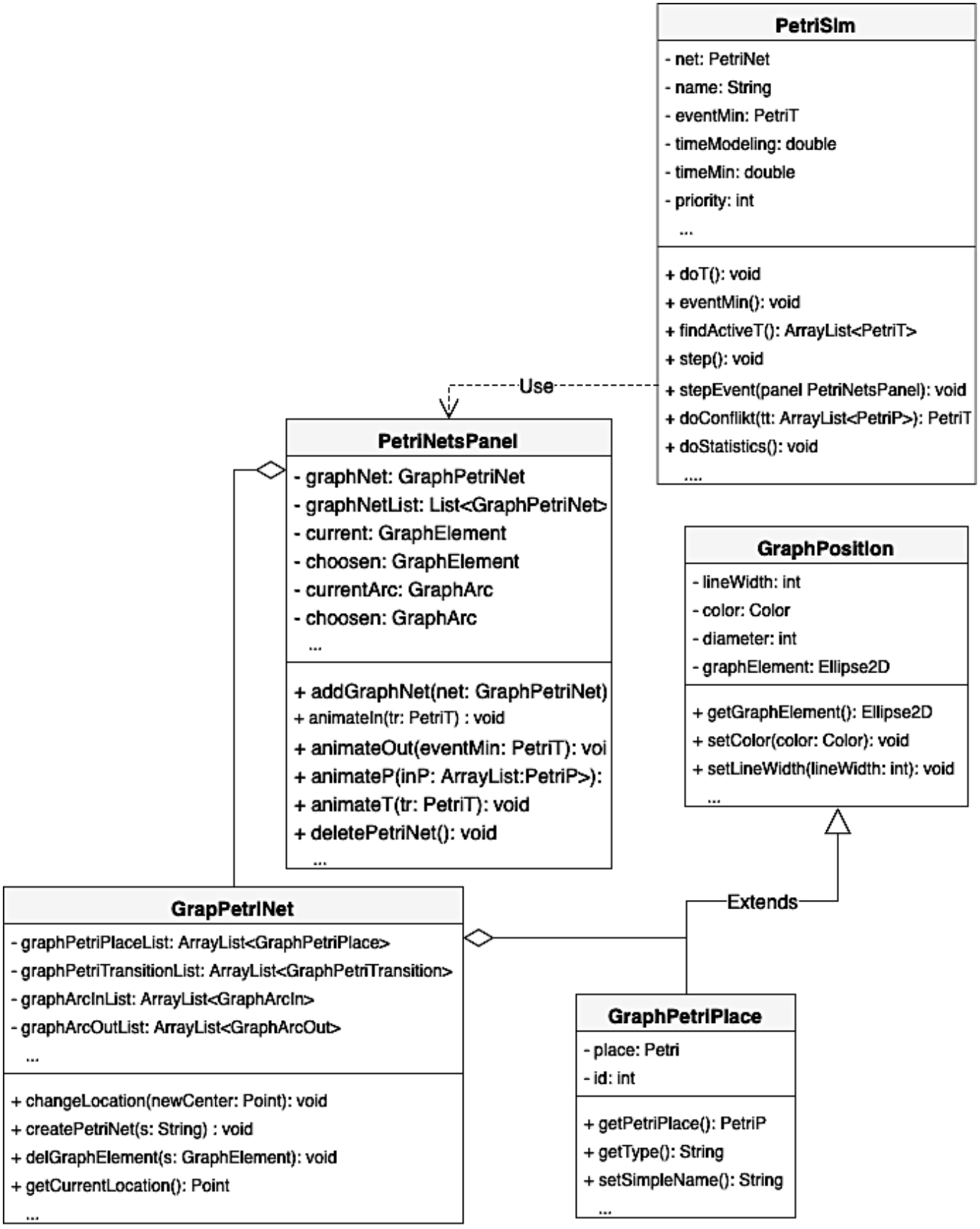
ПЛАКАТ 3 Схема структурна класів програмного забезпечення



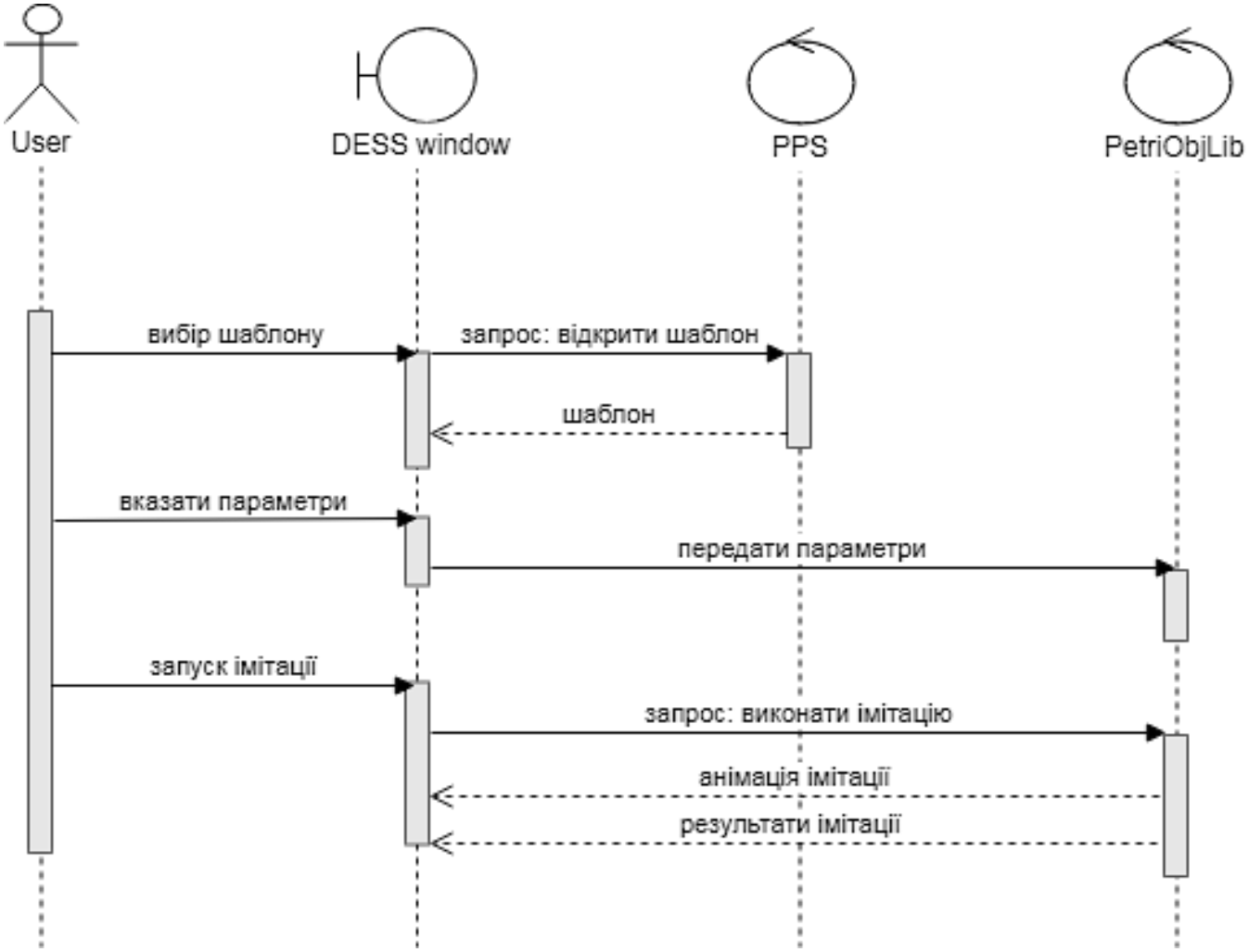
ПЛАКАТ 4 Схема структурна класів програмного забезпечення



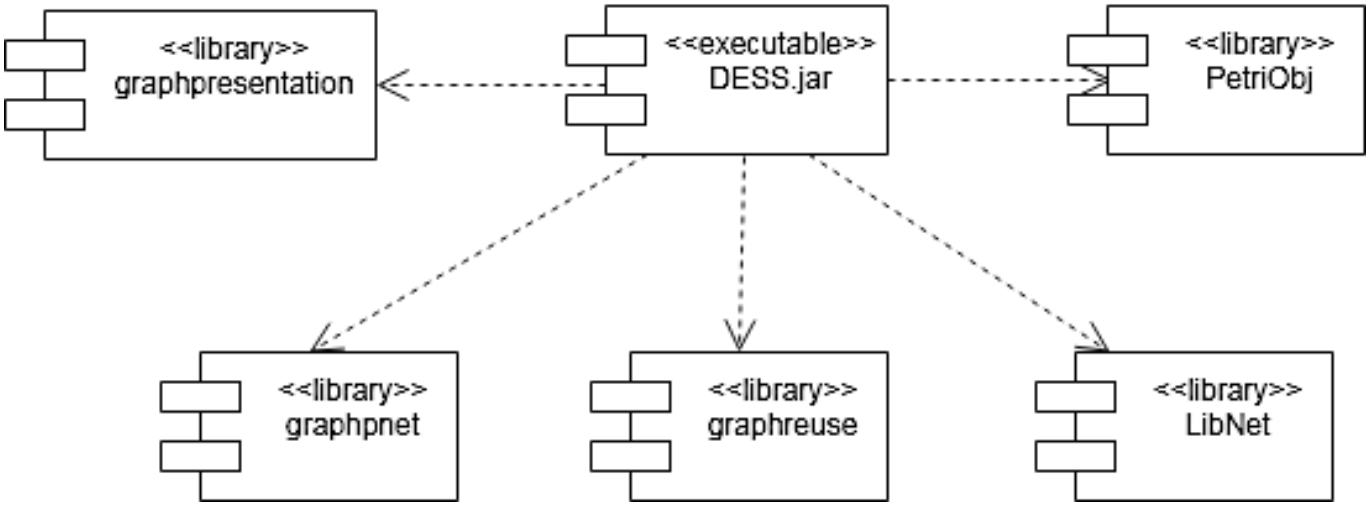
ПЛАКАТ 5 Схема структурна класів програмного забезпечення



ПЛАКАТ 6 Схема структурна послідовності



ПЛАКАТ 7 Схема структурна компонентів програмного забезпечення



ПЛАКАТ 8 Рішення з математичного забезпечення

Стохастична мережа Петрі

$$\text{PetriNet} = (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{W}, \mathbf{K}, \mathbf{R})$$

$\mathbf{P} = \{P\}$ - множина позицій

$\mathbf{T} = \{T\}, \mathbf{P} \cap \mathbf{T} = \emptyset$ - множина переходів

$\mathbf{A} \subseteq \mathbf{P} \times \mathbf{T} \cup \mathbf{T} \times \mathbf{P}$ - множина дуг

$\mathbf{W}: \mathrm{A} \to \mathrm{N}$ - множина натуральних чисел,що задають кратності дуг

$\mathbf{K} = \{(c_T, b_T) | T \in \mathbf{T}, c_T \in \mathrm{N}, b_T \in [0;1]\}$ - множина пар значень, які задають пріоритет і ймовірність запуску переходів

$\mathbf{R}: \mathrm{T} \to \Re_+$ - множина невід'ємних дійсних значень, що характеризують часові затримки в переходах

Рівняння станів Петрі-об'єктної моделі

$$\begin{cases} t_n = \min\limits_{T}\left(\min\limits_{q}\left[E_T(t_{n-1})\right]_q\right), t_n \geq t_{n-1} \\ \mathbf{S}(t_1) = \begin{pmatrix} (D^-)^m(\tilde{\mathbf{S}}_1(t_0)) \\ \dots \\ (D^-)^m(\tilde{\mathbf{S}}_N(t_0)) \\ \dots \\ (D^-)^m(\tilde{\mathbf{S}}_L(t_0)) \end{pmatrix}, \mathbf{S}(t_n) = \begin{pmatrix} (D^-)^m(D^+(\tilde{\mathbf{S}}_1(t_{n-1}))) \\ \dots \\ (D^-)^m(D^+(\tilde{\mathbf{S}}_N(t_{n-1}))) \\ \dots \\ (D^-)^m(D^+(\tilde{\mathbf{S}}_L(t_{n-1}))) \end{pmatrix}, \forall \mathbf{S}_N(t_n): \bigvee\limits_{T \in T_n} \mathbf{Z}(T, t_n) = 0 \end{cases}$$

$\tilde{\mathbf{S}}_N$ - вектор стану Петрі-об'єкта, модифікований з урахуванням зв'язків з іншими об'єктами.

Рівняння станів стохастичної багатоканальної мережі Петрі

$$\begin{cases} t_n = \min\limits_{T}\left(\min\limits_{q}\left[E_T(t_{n-1})\right]_q\right), t_n \geq t_{n-1} \\ \mathbf{S}(t_0) = (D^-)^m(\mathbf{S}(t_0)) \\ \mathbf{S}(t_n) = (D^-)^m(D^+(\mathbf{S}(t_{n-1}))) \\ n = 1,2,\dots \end{cases}$$

$E_T(t_{n-1})$ - множина моментів виходу маркерів із каналів переходу, які на момент часу t_{n-1} активні

$D^-: S^+(t_n) \to S(t_n)$ - перетворення стану мережі Петрі, що відтворює вхід маркерів у її переходи

$D^+: S(t_{n-1}) \to S^+(t_n)$ - перетворення стану мережі Петрі, що відтворює вхід маркерів у її переходи