

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

«До захисту допущено»
Завідувач кафедри

О.А.Павлов
(підпис) (ініціали, прізвище)
“ ” 20 18 р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ
на здобуття ступеня магістра

за спеціальністю 122 Комп'ютерні науки та інформаційні технології
(код та назва спеціальності)

спеціалізацією Інформаційні управляючі системи та технології
(код та назва спеціалізації)

на тему: Веб-сервіс моделювання дискретно-подійних систем

Виконав: студент VI курсу групи ІС-61м
(шифр групи)

Дифучин Антон Юрійович
(прізвище, ім'я, по батькові)

(підпис)

Науковий керівник проф., д.т.н., проф. Томашевський В.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант к.т.н., доц. Жданова О.Г.
(науковий ступінь, вчене звання, прізвище, ініціали)

(підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

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

Студент

(підпис)

РЕФЕРАТ

Магістерська дисертація: 93 с., 45 рис., 14 табл., 1 додаток, 40 джерел.

Актуальність. Потужності обчислювальної техніки з кожним роком зростають, разом з тим розширюються можливості створення веб-додатків. Веб-додатки все більше витісняють додатки рівня операційної системи, оскільки вимагають від користувача тільки встановленого браузера. З цього випливає, що веб-додатки автоматично являються кросплатформними, оскільки браузер є невід'ємним атрибутом будь-якої сучасної ОС. На сьогодні існує багато Петрі-імітаторів, проте майже всі вони є додатками рівня операційної системи, тобто вимагають локальної інсталяції на комп'ютер, що викликає певні незручності порівняно з використанням веб-додатків. Ідея створення онлайн Петрі-імітатора має значні переваги перед офлайн Петрі-імітаторами:

- відсутність необхідності встановлення додаткового програмного забезпечення;
- всі створені моделі зберігаються на віддаленому сервері в обліковому записі користувача;
- легкість обміну моделями між користувачами;
- значно спрощується колективна робота;

Петрі-об'єктне моделювання є технологією імітаційного моделювання систем, що ґрунтується на стохастичних мережах Петрі та надає можливість створювати моделі складних систем з конструктивних елементів. На відміну від інших відомих технологій імітаційного моделювання, Петрі-об'єктна технологія ґрунтується на формалізованому описі динаміки системи мережею Петрі, що дозволяє досягти найбільш абстрактного і водночас найбільш детального опису процесів функціонування.

У зв'язку з цим актуальною науковою задачею є розробка ефективного веб-сервісу для створення Петрі-об'єктних моделей.

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 «Створення засобів імітаційного моделювання дискретно-подійних систем» (№ 0117U000923).

Мета дослідження – створення веб-сервісу з компонентами візуальної розробки динаміки Петрі-об'єктної моделі, які збільшують швидкість та зручність конструювання моделі.

Для досягнення цієї мети необхідно виконати наступні **завдання**:

- виконати огляд відомих засобів автоматизованого імітаційного моделювання дискретно-подійних систем;
- детально ознайомитися з технологією Петрі-об'єктного моделювання, виконати її порівняльний аналіз з іншими технологіями моделювання дискретно-подійних систем;
- виконати огляд існуючих засобів моделювання, що базуються на мережах Петрі, в тому числі здійснити пошук таких засобів, які дозволяють створювати Петрі-об'єктні моделі в онлайн режимі для подальшого використання в імітаційному моделюванні;
- прийняти рішення щодо візуального представлення елементів мереж Петрі та Петрі-об'єктів у майбутній системі візуального програмування мереж Петрі та Петрі-об'єктних моделей (створення Петрі-об'єктів і конструювання зв'язків між ними);
- спроектувати дану систему з використанням алгоритму імітації Петрі-об'єктних моделей та звичайних стохастичних мереж Петрі;
- виконати програмну реалізацію спроектованої системи;
- провести тестування розробленої системи на конкретних моделях;
- виконати аналіз коректності роботи та швидкодії розробленого програмного продукту.

Об'єкт дослідження – процес побудови веб-сервісу для імітаційного моделювання складних дискретно-подійних систем з використанням Петрі-об'єктної технології.

Предмет дослідження – засоби та методи побудови веб-сервісу для імітаційного моделювання складних дискретно-подійних систем на основі Петрі-об'єктного формалізму.

Методи дослідження – фундаментальні положення математичного (імітаційного) моделювання, загальнонаукові принципи та методи проведення досліджень, зокрема: методи аналізу та синтезу, методи прийняття рішень при проектуванні архітектури системи, методи систематизації, абстрагування, структурування для виконання оцінки існуючих засобів імітаційного моделювання.

Наукова новизна отриманих результатів. Вперше запропонована архітектура веб-сервісу, що забезпечує ефективну розробку імітаційних моделей на основі інтеграції java-бібліотеки Петрі-об'єктного моделювання та веб-орієнтованого графічного редактора. Удосконалено графічне представлення Петрі-об'єктних моделей, що надає можливість візуальної розробки Петрі-об'єктних моделей на двох рівнях:

- розробка стохастичної мережі Петрі та її використання для створення Петрі-об'єктів;
- розробка динаміки моделі з множини Петрі-об'єктів;

Публікації. Матеріали роботи опубліковані на 8-ій міжнародній конференції «Intelligent Data Acquisition and Advanced Computing Sysytems: Technology and Applications (IDAACS)» [29] та опубліковані в електронній бібліотеці Scopus; представлені на 7-ій міжнародній конференції «Internet Technologies and Applications, ITA 2017» [40] та опубліковані в електронній бібліотеці Scopus; представлені в рамках 18-ї Міжнародної науково-технічної конференції SAIT 2016 «Системний аналіз та інформаційні технології» [28].

ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ДИСКРЕТНО-ПОДІЙНА СИСТЕМА, СТОХАСТИЧНА МЕРЕЖА ПЕТРІ, ПЕТРІ-ОБ'ЄКТНА МОДЕЛЬ.

ABSTRACT

Master dissertation: 93 pp., 45 fig., 14 tab., 1 app., 40 sources.

Topicality. The power of computer technology grows every year, however, the possibilities of creating web applications are expanding. Web applications are increasingly pushing the application level of the operating system, since they require the user only the installed browser. It follows that web applications are automatically cross-platform because the browser is an integral part of any modern operating system. To date, there are many Petri simulators, but almost all of them are application-level operating systems, that is, they require a local installation on the computer, which causes some inconvenience compared to the use of web applications. The idea of creating an online Petri-simulator has significant advantages over offline Petri-simulators:

- no need to install additional software;
- all created models are stored on the remote server in the user account;
- ease of exchange of models between users;
- collective work is greatly simplified;

Petri-object simulation is a simulation technology based on Petri's stochastic networks and provides the ability to create models of complex systems from constructive elements. Unlike other well-known simulation techniques, Petri-object technology is based on a formalized description of the dynamics of the Petri network system, which allows for the most abstract and, at the same time, most detailed description of the functioning processes.

In this regard, the actual scientific task is to develop an effective Web service for the creation of Petri-object models.

Relationship of work with scientific programs, plans, themes. The work was carried out at the Department of Automated Systems for Information Processing and Management of the National Technical University of Ukraine "Kyiv Polytechnic Institute. Igor Sikorsky "within the framework of the topic «Development of tools for discrete-event system simulation » (No. 0117U000923).

The aim of the research is the creation of a web service with components for visual development of the dynamics of the Petri-object model, which increase the speed and convenience of designing the model.

To achieve this goal, the following tasks must be performed:

- carry out an overview of the known means of automated simulation of discrete-event systems;
- to familiarize with the Petri-object modeling technology in detail, to perform its comparative analysis with other technologies of the modeling of discrete-event systems;
- perform an overview of the existing modeling tools based on Petri's networks, including searching for such tools that allow the creation of Petri-object models in online mode for further use in simulation;
- Decide on the visual representation of elements of Petri and Petri-objects networks in the future system of visual programming of Petri Networks and Petri-object models (creation of Petri-objects and construction of links between them);
- To design this system using the simulation algorithm of Petri-object models and Petri stochastic networks;
- carry out the program realization of the designed system;
- carry out testing of the developed system on specific models;
- perform analysis of the correctness of the work and the speed of the developed software product.

The object of research is the process of building a web service for simulation of complex discrete-event systems using Petri-object technology.

The subject of research is the means and methods of constructing a web service for simulation of complex discrete-event systems based on the Petri-object formalism.

Research methods are the fundamental provisions of mathematical modeling, general scientific principles and methods of conducting research, in particular: methods of analysis and synthesis, methods of decision making in the design of the architecture of the system, methods of systematization, abstraction, structuring for the evaluation of existing means of simulation.

Scientific novelty of the obtained results. For the first time, we propose a web service architecture that provides an effective development of simulation models based on the integration of the Petri-object model java library and Web-based graphic editor. The

graphical representation of Petri-object models is improved, which provides the possibility of visual development of Petri-object models on two levels:

- development of stochastic Petri net and its use for creating Petri-objects
- development of dynamics of the model from the set of Petri-objects;

Publications

The materials of research are published in theses of the 8th International Conference «Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)» and published in Scopus Digital Library [29]; presented at the 7th International Conference «Internet Technologies and Applications (ITA 2017)» and published in Scopus Digital Library [40]; presented at the 18th International Conference on «System analysis and information technologies (SAIT 2016)».

SYSTEM SIMULATION, DISCRETE-EVENT SYSTEM, STOCHASTIC PETRI NET, PETRI-OBJECT MODEL.

ЗМІСТ

ВСТУП.....		11
1	ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ МОДЕЛЮВАННЯ ДИСКРЕТНО-ПОДІЙНИХ СИСТЕМ	15
1.1	Універсальні пакети імітаційного моделювання дискретно-подійних систем	17
1.2	Мережі Петрі як універсальний засіб моделювання дискретно-подійних систем	21
1.3	Висновки до розділу	28
2	МАТЕМАТИЧНИЙ АПАРАТ ПЕТРІ-ОБ'ЄКТНОГО МОДЕЛЮВАННЯ ...	31
2.1	Мережі Петрі як засіб формалізації дискретно-подійних систем	31
2.2	Математичний опис Петрі-об'єктної моделі	38
2.3	Алгоритм імітації Петрі-об'єктної моделі	40
2.4	Процес побудови Петрі-об'єктної моделі	41
2.5	Висновки до розділу	45
3	ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ МОДЕЛЮВАННЯ ДИСКРЕТНО-ПОДІЙНИХ СИСТЕМ	46
3.1	Архітектура	46
3.2	Засоби розробки	48
3.2.1	Вибір мови розробки клієнтської частини	48
3.2.2	Вибір мови розробки серверної частини	49
3.2.3	Вибір середовища розробки	50
3.3	Проектування клієнтського застосунку	50
3.3.1	Вхідні дані	51
3.4	Діаграма класів	60
3.5	Специфікація функцій	62
3.6	Керівництво користувача	65
3.7	Висновки до розділу	71
4	РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ	72
4.1	Порядок проведення досліджень	72

4.1.1	Результати випробувань.....	80
4.2	Висновки до розділу	82
	ЗАГАЛЬНІ ВИСНОВКИ.....	83
	ПЕРЕЛІК ПОСИЛАНЬ.....	84

ВСТУП

Розробка та удосконалення сучасних систем у різних галузях людської діяльності нерозривно пов'язані з розвитком технологій моделювання процесів функціонування складних систем. Від якості використовуваних на етапі проектування моделей насамперед залежить якість створюваної системи. Оптимізація параметрів управління, визначення ефективності функціонування системи при заданих параметрах та інші задачі розв'язуються методами аналітичного та імітаційного моделювання. Імітаційні моделі, на відміну від аналітичних, дозволяють відтворити процес функціонування системи та управління нею найбільш детально.

На сьогодні існує багато візуальних інструментів для моделювання дискретно-подійних систем. Проте пошуки нових ефективних підходів до моделювання складних систем з великою кількістю взаємопов'язаних елементів та різноманітністю процесів функціонування елементів і підсистем постійно продовжуються. В даній роботі запропоноване веб-орієнтоване візуальне середовище моделювання дискретно-подійних систем, що збільшує зручність та швидкість розробки моделей.

Основні випадки, в яких дослідники вдаються до імітаційного моделювання, наступні:

- необхідно провести експерименти із реальною системою, але це неможливо або занадто дорого;
- немає адекватної аналітичної моделі системи або методів розв'язання такої моделі, зокрема через стохастичність системи або її функціональну чи структурну складність;
- є необхідність в імітації поведінки системи в часі.

Імітаційне моделювання знаходить застосування у безлічі сфер людської діяльності. Можна виділити наступні задачі, при розв'язанні яких моделювання виявляється особливо ефективним:

- проектування і аналіз виробничих систем;
- оцінка різноманітних систем озброєнь та вимог до їх матеріально-технічного забезпечення;

- визначення вимог до обладнання та протоколів мереж зв'язку;
- визначення вимог до обладнання та програмного забезпечення різноманітних комп'ютерних систем;
- проектування та аналіз роботи транспортних систем, наприклад аеропортів, автомагістралей, портів і метрополітену;
- оцінка проектів створення різноманітних організацій масового обслуговування, наприклад центрів обробки замовлень, закладів швидкого харчування, лікарень, відділів зв'язку;
- модернізація різноманітних процесів у діловій сфері;
- визначення політики в системах управління запасами;
- аналіз фінансових і економічних систем [2].

Імітаційні моделі розробляються для вивчення поведінки окремих підприємств і корпорацій, дослідження ринкової рівноваги, оцінки наслідків державного регулювання в економіці, оптимального управління логістичними системами, вивчення епідеміології (прогнозування динаміки розповсюдження захворювань) і розв'язання інших практичних задач [3].

За можливістю змінювати в часі свої властивості моделі поділяються на статичні та динамічні. Статичні моделі, на відміну від динамічних, не змінюють своїх властивостей у часі [1]. Динамічні моделі використовуються для опису поведінки динамічних систем, тобто таких систем, що змінюються з часом і "для яких однозначно визначено поняття стану як сукупності деяких величин у даний момент часу" [4]. Надалі розглядатимемо лише динамічні моделі, оскільки, незважаючи на те, що будь-яку математичну модель можна назвати імітаційною, імітаційне моделювання переважно полягає в дослідженні саме динамічних моделей.

Залежно від того, яким чином відтворюються в часі стани моделі, розрізняють дискретні, неперервні й дискретно-неперервні (комбіновані) моделі [1]. Очевидно, що мова йде про динамічні моделі, оскільки оперуємо поняттям часу. Дискретна система – це така система, в якій змінні стану змінюються лише в дискретні моменти часу [5]. Неперервна система – це система, в якій змінні стану змінюються безперервно з плином часу [5]. Прикладами дискретної системи можуть слугувати релейно-

контактні схеми, цифрові системи комутації; неперервної – механічні та термодинамічні фізичні системи. В даній роботі розглядатимемо тільки дискретні системи. По-перше, вони принципово простіші, ніж неперервні. По-друге, неперервні системи можна з певною точністю представити дискретними моделями. Нарешті, найскладніші кібернетичні системи зазвичай виявляються дискретними.

Дискретні динамічні системи, в свою чергу, можна поділити на подійні (дискретно-подійні), в яких зміна станів повністю залежить від настання дискретних подій, та часові, в яких зміна станів відбувається через фіксовані інтервали часу (такти). Надалі будемо розглядати лише дискретно-подійні системи.

Стрімкий розвиток інформаційних технологій спричинив зростання потреби в імітаційному моделюванні складних систем для відшукування оптимальних параметрів управління ними та прийняття рішень. Підвищення складності систем, поведінку яких необхідно моделювати, пов'язане з тенденцією до зростання кількості їх структурних елементів, а також кількості взаємозв'язків між ними. В зв'язку з цим відбувається зростання вимог до засобів моделювання, зокрема постає задача забезпечення високої швидкості побудови імітаційних моделей дискретно-подійних систем, а також представлення моделей у вигляді, що максимально полегшив би їхнє сприйняття користувачем. Крім того, набуває важливості швидкість модифікації моделей в зв'язку зі структурними та функціональними змінами [6].

Петрі-об'єктне моделювання є технологією імітаційного моделювання систем, що ґрунтується на стохастичних мережах Петрі та надає можливість створювати моделі складних систем з конструктивних елементів. На відміну від інших відомих технологій імітаційного моделювання, Петрі-об'єктна технологія ґрунтується на формалізованому описі динаміки системи мережею Петрі, що дозволяє досягти найбільш абстрактного і водночас найбільш детального опису процесів функціонування системи.

Все це призводить до актуальності розробки такого програмного забезпечення, яке дозволило б імітувати поведінку складних дискретно-подійних систем за допомогою технології Петрі-об'єктного моделювання, швидко будувати та модифікувати імітаційні моделі та яке б забезпечувало зручність представлення

імітаційних моделей користувачу. Застосовуючи візуальне програмування, можна забезпечити високу швидкість та гнучкість процесу конструювання імітаційних моделей.

Мета даного дослідження полягає в створенні веб-сервісу з компонентами візуальної розробки динаміки Петрі-об'єктної моделі, які збільшують швидкість та зручність конструювання моделі. Для досягнення цієї мети необхідно виконати наступні завдання: виконати огляд відомих засобів автоматизованого імітаційного моделювання дискретно-подійних систем; детально ознайомитися з технологією Петрі-об'єктного моделювання, виконати її порівняльний аналіз з іншими технологіями моделювання дискретно-подійних систем; виконати огляд існуючих засобів моделювання, що базуються на мережах Петрі, в тому числі здійснити пошук таких засобів, які дозволяють візуально програмувати Петрі-об'єктні моделі для подальшого використання в імітаційному моделюванні; прийняти рішення щодо візуального представлення елементів мереж Петрі та Петрі-об'єктів у майбутній системі візуального програмування мереж Петрі та Петрі-об'єктних моделей (створення Петрі-об'єктів і конструювання зв'язків між ними); спроектувати дану систему з використанням алгоритму імітації Петрі-об'єктних моделей та звичайних стохастичних мереж Петрі; виконати програмну реалізацію спроектованої системи; провести тестування розробленої системи на конкретних моделях; виконати аналіз коректності роботи та швидкодії розробленого програмного продукту.

Об'єктом дослідження є процес побудови веб-сервісу для імітаційного моделювання складних дискретно-подійних систем з використанням Петрі-об'єктної технології. Предмет дослідження – засоби та методи побудови веб-сервісу для імітаційного моделювання складних дискретно-подійних систем на основі Петрі-об'єктного формалізму. В якості методів дослідження вирішено використовувати фундаментальні положення математичного (імітаційного) моделювання, загальнонаукові принципи та методи проведення досліджень, зокрема: методи аналізу та синтезу, методи прийняття рішень при проектуванні архітектури системи, методи систематизації, абстрагування, структурування для виконання оцінки існуючих засобів імітаційного моделювання.

1 ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ МОДЕЛЮВАННЯ ДИСКРЕТНО-ПОДІЙНИХ СИСТЕМ

Програмне забезпечення з імітаційного моделювання систем охоплює широке різноманіття підходів: імітація з використанням колекцій блоків, налаштованих на виконання тих чи інших функцій – очікування, обробка, комплектація, транспортування та інша (Arena, ExtendSim, PlantSim) [2]; математичне моделювання з використанням динамічних блоків (VisSim, Simulink) [2]; гібридне моделювання з використанням блоків обох попередніх підходів (AnyLogic); відкриті бібліотеки програм для моделювання дискретно-подійних систем (DESMO-J, SIM.JS); імітація на основі опису процесу функціонування системи у вигляді стохастичної мережі Петрі (CPN Tools).

Всі вищеперераховані засоби імітаційного моделювання можна поділити на 2 групи: пакети імітаційного моделювання та мови імітаційного моделювання.

Мови імітаційного моделювання дискретно-подійних систем

Мови імітаційного моделювання дискретних систем поділяють на 3 категорії:

- мови, що орієнтовані на транзакти;
- мови, що орієнтовані на процеси;
- мови, що орієнтовані на події;

Розглянемо найбільш яскравих представників кожної категорії.

Транзактно-орієнтовані мови побудовані на концепції руху об'єктів у просторі та часі. Такі об'єкти називають транзактами, які частіше за все являються елементами потоку заявок. Основне призначення транзактів це рух по елементах системи.

Однією з найпопулярніших мов імітаційного моделювання орієнтованою на транзакти є мова GPSS. Вона з'явилася на початку 60-х років, та вдосконалювалася протягом декількох років, щоб задовольнити новим вимогам, пов'язаним із моделюванням складних систем. Щоб ефективно використовувати цю мову, користувачу необхідно налаштувати близько вісімдесяти різноманітних блоків. Незважаючи на велику кількість років використання GPSS, мова все ще має деякі

особливості моделювання, які важко пояснити. Прикладом може слугувати необхідність апроксимувати неперервні розподіли ймовірностей їх кусочно-лінійними аналогами [12]. GPSS є високо структурованою транзактно-орієнтованою імітаційною мовою, що була найвідомішою та найпоширенішою мовою дискретного моделювання. Мова використовує блокові діаграми для представлення опису системи. Тимчасові сутності (транзакти) створюються та протікають по діаграмі. Оскільки GPSS не є процедурною мовою, вона не може виконувати деякі складні розрахунки, які можуть виконувати інші мови [11]. Приклад простої програми мовою GPSS наведено на рисунку 1.4.

```

; GPSS World Sample File - BARBER.GPS.
*****
*                               *
*      Моделирование парикмахерской      *
*                               *
*****
Waittime  QTABLE   Barber,0,2,15   ;Объявление таблицы для гистограммы
                                           ;распределения времен ожидания
                                           ;клиентов в очереди
                                           ;Создание новых посетителей
GENERATE  3,1
TEST LE  Q$Barber,3,Finis ;Если в очереди больше 3 человек,
                                           ;то клиент покидает парикмахерскую
SAVEVALUE Custnum+,1 ;Общее число оставшихся клиентов
ASSIGN   Custnum,X$Custnum ;Присвоение клиенту номера
QUEUE   Barber ;Начало ожидания в очереди
SEIZE   Barber ;Семидание парикмахера или
                                           ; начало стрижки
DEPART  Barber ;Конец ожидания в очереди
ADVANCE 6,2 ;Стрижка занимает несколько минут
RELEASE Barber ;Стрижка завершена
Finis   TERMINATE 1 ;Посетитель уходит

```

Рисунок 1.1 – Приклад програми мовою GPSS

Яскравим прикладом мови, що орієнтована на процеси є мова SIMULA. SIMULA започаткувала багато наступних розробок, які використовуються і до сьогодні. Процес – це послідовність подій, що зв’язані між собою логікою певних відносин. Процес поділяється на класи, кожен з яких представляється у вигляді процедури. Процедура виконується одночасно для всіх представників класу, які існують в системі на даний час. Програми імітаційного моделювання систем, написані на процесно-орієнтованих мовах, відрізняються тим, що мають таку ж саму структуру, що і модельований об’єкт.

SIMULA є розширенням мови ALGOL. Дана мова, що орієнтована на процеси, здатна виконувати дуже складні обчислення, які не може виконувати GPSS. SIMULA є більш потужною та гнучкою ніж GPSS, а також має ширшу область застосування. У SIMULA було вперше введено “концепцію процесу”. Розробники мови ввели концептуальні зміни, які тепер часто називають “об’єктно-орієнтованими”. Важливими концепціями даної імітаційної мови є абстрактні типи даних, класи та квазіпаралельне виконання процесів. SIMULA традиційно вважається першою об’єктно-орієнтованою мовою у світі. Це також перша імітаційна мова, в якій дійсно реалізовано процесно-орієнтований підхід [11].

Виконання імітації мовами, що орієнтовані на події відбувається шляхом складання списку подій, по яких відбувається просування часу. Організація програми моделі представлена у вигляді сукупності процедур обслуговування подій. Виконання цих процедур синхронізується списковим механізмом планування (розкладу) подій. Класичним представником даної підгрупи є мова SIMSCRIPT.

1.1 Універсальні пакети імітаційного моделювання дискретно-подійних систем

Серед найвідоміших універсальних засобів моделювання слід виділити Arena, SIMUL8, ExtendSim і AnyLogic. Всі вони можуть бути використані для моделювання дискретно-подійних систем, хоча деякі дозволяють також виконувати імітаційне моделювання неперервних систем.

Arena застосовується для моделювання як дискретних, так і неперервних систем. У першу чергу дане застосування розраховане на моделювання бізнес-процесів. У базовій версії програми динаміка того чи іншого процесу представляється у вигляді ієрархічної діаграми, а інформація про систему зберігається у таблицях з даними. В стандартній версії імітаційні моделі будуються з графічних об’єктів, які називаються модулями та визначають логіку функціонування системи і такі фізичні компоненти, як машини та операторів. Модулі можна об’єднувати в колекції, що називаються шаблонами. Дана версія дозволяє моделювати не лише дискретні та

неперервні системи, а й системи змішаного типу - дискретно-неперервні [5]. Arena дозволяє виконувати анімацію поведінки побудованих моделей. Написано низку книжок, призначених для навчання роботі з даним програмним забезпеченням [14-16]. На рисунку 1.2 представлено вікно програми Arena з простою моделлю системи.

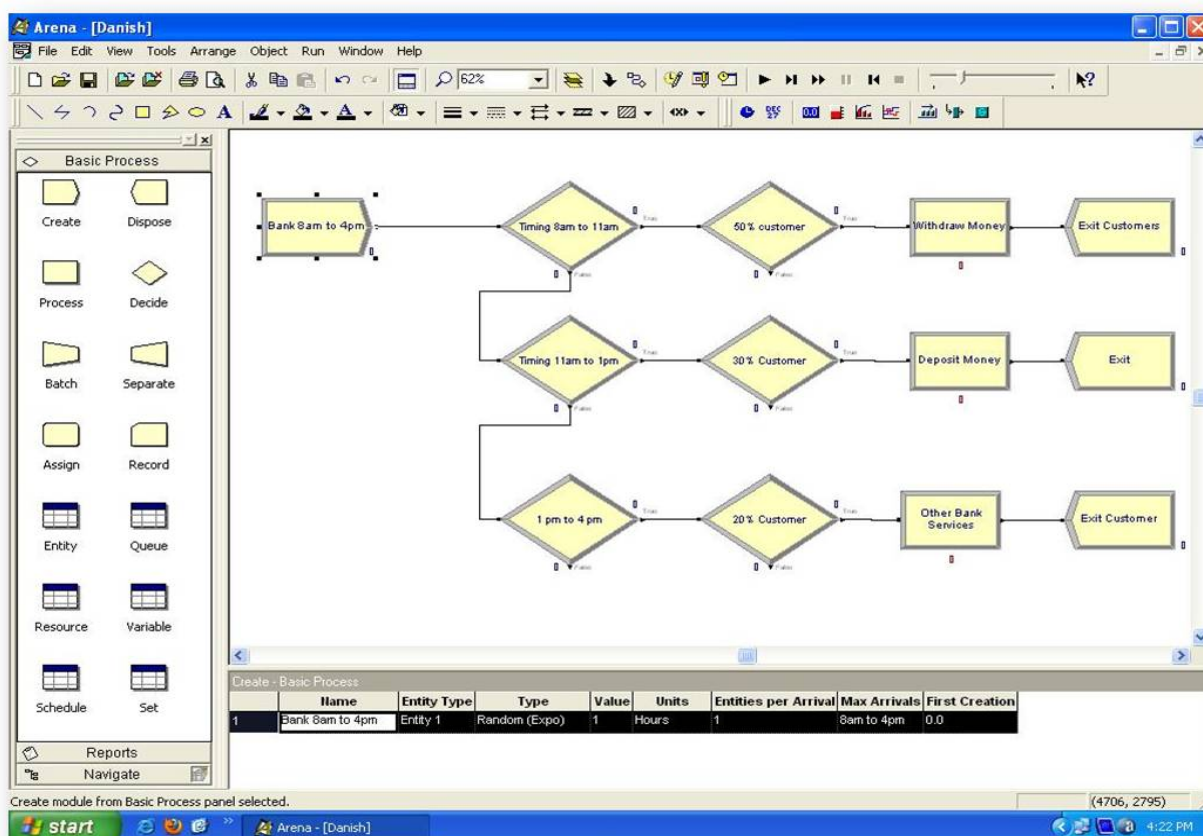


Рис. 1.2 – Проста модель системи в Arena [14]

Arena містить засоби аналізу вхідних та вихідних даних, їх верифікації та валідації, дозволяє виконувати кореляційний аналіз [15]. Застосування також надає можливість легко порівнювати між собою різноманітні конфігурації тієї чи іншої системи, зокрема порівнювати продуктивність отриманих версій системи за допомогою статистичних методів [16].

У 1995 році корпорація SIMUL8 розробила однойменний програмний засіб імітаційного моделювання систем. Імітаційна модель у SIMUL8 створюється шляхом ілюстрації потоку роботи за допомогою іконок та стрілок, які представляють собою ресурси та черги в системі. Для всіх властивостей іконок передбачаються

значення за замовчуванням, що дозволяє виконувати анімацію навіть на ранніх етапах побудови моделі.

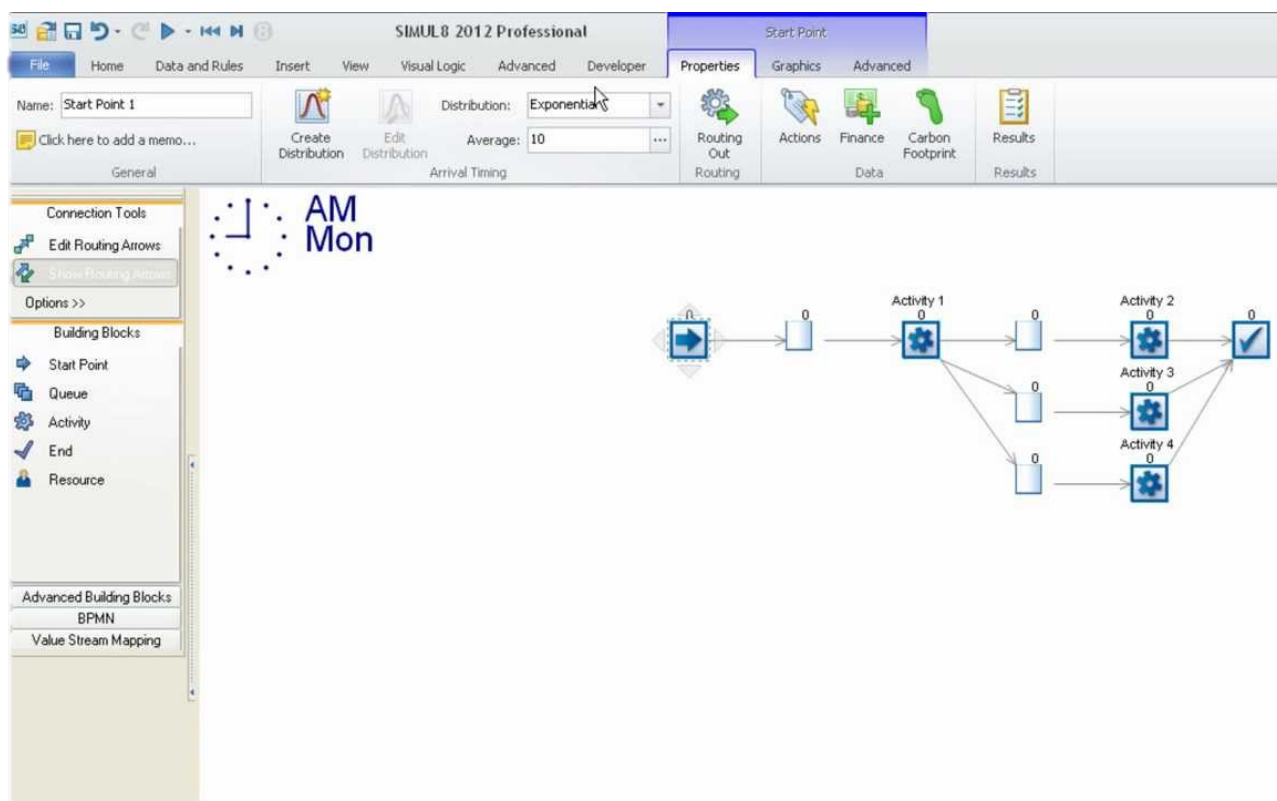


Рис. 1.3 – Проста модель системи в SIMUL8

Як і деякі інші пакети, SIMUL8 також надає шаблони імітаційних моделей, що спростити та прискорити процес побудови моделі. Крім того, користувач має можливість розробляти власні іконки, що можуть бути використані багаторазово та передані іншим користувачам. Це теж дозволяє прискорити процес моделювання та стандартизувати обробку певних ситуацій в межах однієї установи. SIMUL8 зберігає імітаційні моделі та дані у вигляді XML файлів для полегшення інтеграції з іншим програмним забезпеченням [5].

ExtendSim – сімейство програмних засобів, розроблених компанією Imagine That. За допомогою цих засобів можна проводити імітаційне моделювання будь-яких динамічних систем: дискретних, неперервних, неперервно-дискретних. Побудова моделі в ExtendSim відбувається за принципом блок-схеми, причому користувач може як користуватись існуючими блоками, так і створювати власні. Кожен такий блок може містити в собі програмний код, параметри, інтерфейс користувача,

анімацію. Імітаційні моделі будуються шляхом розміщення та поєднання блоків, а також введення параметрів у їхні діалогові вікна, як показано на рисунку 1.3.

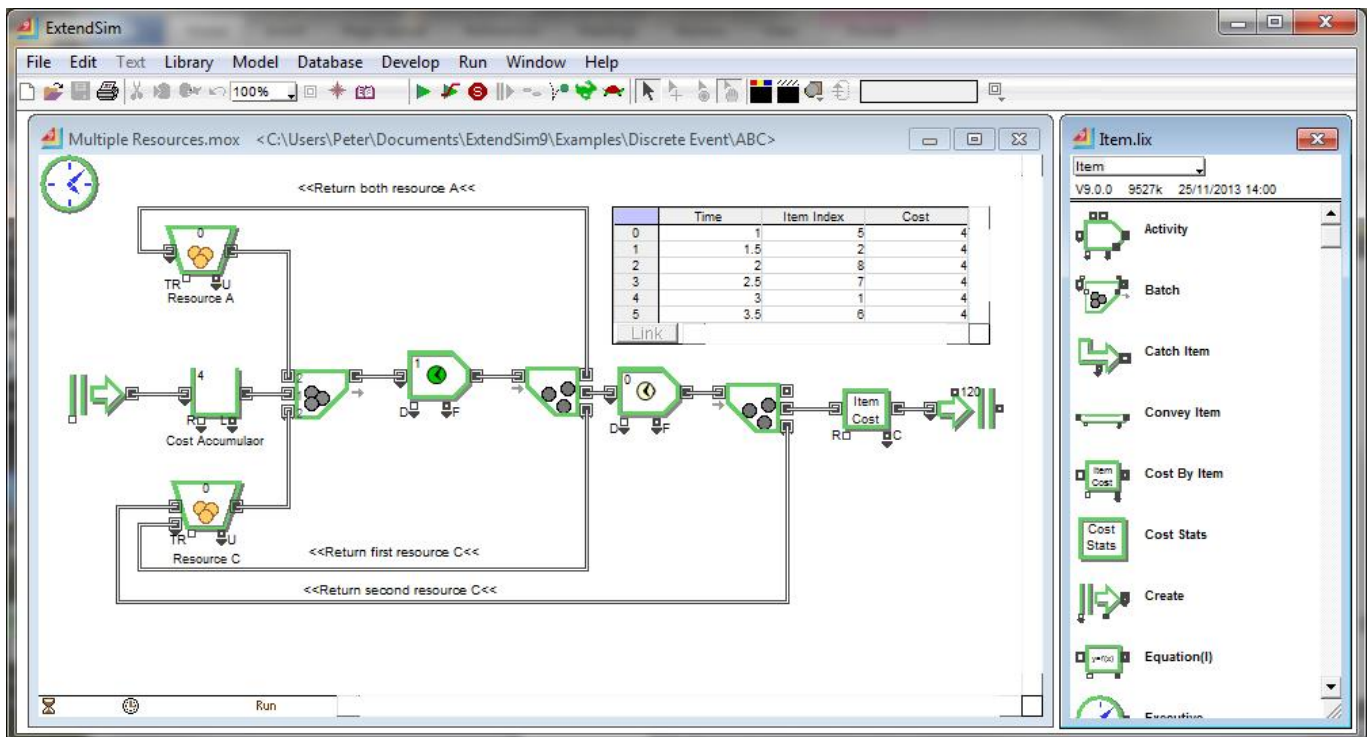


Рисунок 1.4 – Процес побудови моделі в пакеті ExtendSim

ExtendSim дозволяє користуватися, зокрема, наступними елементарними блоками: черга, генератор, діяльність, пул ресурсів, вихід. Сутності переміщуються по блокам через переходи між ними, починаючи з блоків-генераторів. ExtendSim, як і Agena, містить засоби анімації поведінки системи та статистичного аналізу вихідних даних. Крім того, вихідні коди більшості основних блоків доступні для ознайомлення та модифікації, а також розроблені блоки можна зберігати для повторного використання [5].

AnyLogic – інструмент імітаційного моделювання, створений однойменною російською компанією, що підтримує такі підходи до розробки моделей, як процесно-орієнтований (дискретно-подійний), системно-динамічний та агентний, а також їхні комбінації. Серед основних сфер застосування AnyLogic - логістика, виробництво, ринок та конкуренція, охорона здоров'я, склади та перевезення, фінанси та управління активами, бізнес-процеси та системи обслуговування, аеропорти, вокзали, торговельні центри, залізниця, оборона, IT-інфраструктура тощо.

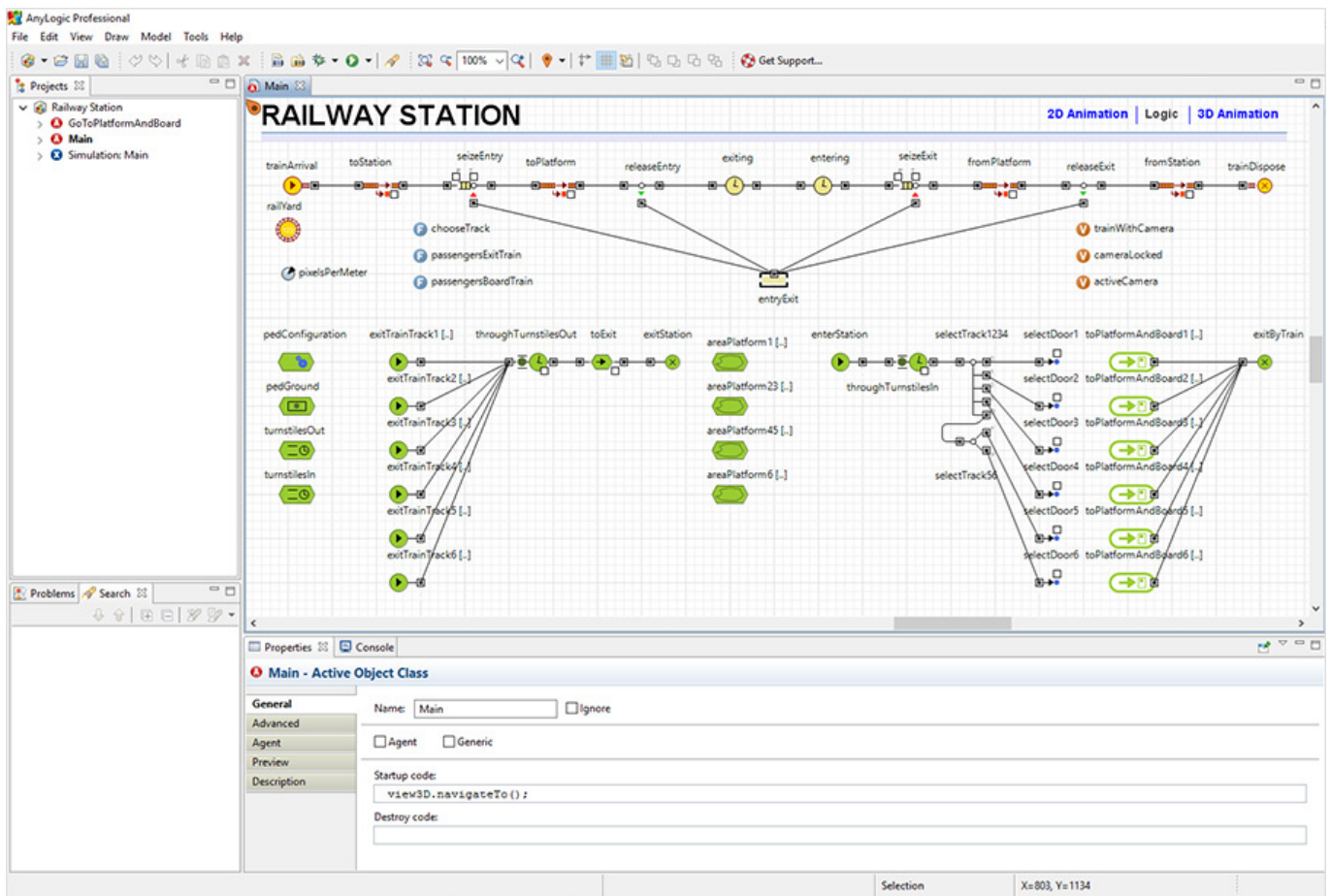


Рисунок 1.5 – Проста модель системи в AnyLogic

AnyLogic містить компоненти для роботи з текстовими файлами, файлами Excel, базами даних, базової відладки моделей, допомоги в написанні коду. Програма дозволяє користувачу створювати власні бібліотеки класів агентів, які він часто використовує, причому вони можуть бути передані іншим користувачам. Також даний засіб надає можливість виконання 2D- і 3D-анімації.

1.2 Мережі Петрі як універсальний засіб моделювання дискретно-подійних систем

Для представлення дискретних процесів управління, що характеризуються наявністю паралельних процесів, а також асинхронних процесів, використовують моделі, що запропоновані К.А. Петрі для моделювання цифрових автоматів в його дисертації [1].

Мережа Петрі являє собою дводольний орієнтований мультиграф, що складається з вершин двох типів - позицій та переходів, з'єднаних між собою дугами. У позиціях можуть бути розміщені мітки (маркери), здатні переміщатися по мережі. Події в системі являють собою спрацьовування переходів, при яких маркери із вхідних позицій переходу переміщуються у його вихідні позиції.

Мережі Петрі на сьогоднішній день широко використовуються для проектування цифрових автоматів, для аналізу властивостей алгоритмів і програм, для моделювання паралельної обробки інформації. Розвитку теорії мережі Петрі сприяли наукові розробки вчених Т.Мурати, Дж. Пітерсона, В.Е.Котова, Д.А. Зайцева. Доведено, що будь-який цифровий автомат може бути представлений мережею Петрі [21]. Тому мережі Петрі представляють більш широкий клас систем, ніж цифрові автомати. Перевагою використання мереж Петрі є їх універсальність для представлення як процесів управління, так і процесів функціонування об'єктів управління. Одним з розширенням мереж Петрі є Е-мережі, які надають додаткові можливості для введення логічних умов функціонування моделі [31].

Отже, Мережі Петрі – це потужний універсальний засіб формального опису поведінки дискретно-подійних систем. Однак через те, що навіть для опису простих систем мережами Петрі зазвичай необхідно застосовувати велику кількість елементів, імітаційне моделювання з використанням апарату мереж Петрі не набуває серед науковців великої популярності. Ще одна проблема пов'язана з тим, що відсутність математичної теорії стохастичних часових мереж Петрі призводить до різноманіття підходів до побудови алгоритмів імітації. Хоча існує багато програмних засобів, що так чи інакше використовують мережі Петрі для моделювання, широко відомих розробок технологій моделювання, що базуються на мережах Петрі, не існує. Таким чином, постає задача зменшення кількості елементів, необхідних для представлення процесів за допомогою мереж Петрі, при збереженні універсальності алгоритму імітації [17]. Крім того, постає задача розробки такого програмного засобу, який дозволив би імітувати поведінку складних дискретно-подійних систем за допомогою технології моделювання, що базується на мережах Петрі, швидко будувати та

модифікувати імітаційні моделі та який би забезпечував зручність представлення моделей користувачу.

У [17] пропонується використання блочної структури побудови моделей мереж Петрі, що дозволяє створювати подібні фрагменти мереж Петрі простим копіюванням чи вставкою відповідного блоку. Це певною мірою полегшує процес побудови імітаційних моделей, однак не вирішує проблему у випадку, коли система складається із сотень подібних елементів, що пов'язані між собою [18]. У [19] пропонується розбивати велику мережу Петрі на функціональні підмережі, що дозволяє замість властивостей основної мережі Петрі досліджувати властивості її підмереж.

Останнім часом з'являються наукові роботи, в яких розглядається об'єднання об'єктно-орієнтованого підходу і моделювання за допомогою мереж Петрі тим чи іншим способом в залежності від вирішуваної задачі ([20, 21]). Увійшов у використання термін “об'єктно-орієнтованих мереж Петрі” (object oriented Petri nets, OPN) [18]. Він закріпився за розширеним поняттям мережі Петрі, в якому існують специфічні елементи мережі Петрі (позиції, переходи, маркери), що виконують функції об'єднання складових частин (які і є об'єктами) мережі Петрі [20]. Однією з реалізацій цього підходу є мова LOOPN Чарльза Лакоса, в якій термін “об'єктно-орієнтована мережа Петрі” означає, що маркери мережі є об'єктами в термінах об'єктно-орієнтованого програмування, а також окремі фрагменти мережі Петрі можуть слугувати об'єктами [21]. В усіх розробках мережа Петрі модифікується у той чи інший спосіб, щоб пристосувати її до вимог об'єктно-орієнтованої парадигми, що значно ускладнює інструмент мереж Петрі. Крім того, вказані підходи дозволяють реалізовувати великі мережі Петрі лише теоретично [18].

Петрі-об'єктне моделювання - технологія імітаційного моделювання систем, що ґрунтується на стохастичних мережах Петрі та надає можливість створювати моделі складних систем з конструктивних елементів. На відміну від інших відомих технологій імітаційного моделювання, Петрі-об'єктна технологія ґрунтується на формалізованому описі динаміки системи мережею Петрі [23], що дозволяє досягти найбільш абстрактного і водночас найбільш детального опису процесів функціонування системи.

Технологія Петрі-об'єктного моделювання ґрунтується на концепції, що модель системи конструюється з елементів, динаміка яких задана стохастичною мережею Петрі. Об'єкти із заданою динамікою тиражуються у заданій кількості і з заданими параметрами, можуть успадковувати динаміку від інших об'єктів та створювати нові об'єкти, динаміка яких пов'язана з іншими об'єктами. Доведено, що динаміка Петрі-об'єктної моделі в цілому теж описується стохастичною мережею. Цей факт є важливим, оскільки гарантує коректність та обчислюваність моделі [24]. Ефективність та поліноміальна обчислювальна складність алгоритму імітації Петрі-об'єктної моделі доведені в [25].

У [22] викладено основні характеристики існуючих програмних засобів імітаційного моделювання, що використовують апарат мереж Петрі. Для кожного засобу наведено перелік типів мереж Петрі, які підтримуються даним програмним забезпеченням, а також перелік наявних компонентів (функціональних підсистем). Витяг із даного матеріалу представлено в таблиці 1.1.

Таблиця 1.1 - Огляд засобів моделювання, що використовують мережі Петрі

Назва засобу	Тип програмного забезпечення	Підтримувані типи мереж Петрі	Наявні компоненти (підсистеми)
ALPHA/Sim	Комерційне	високорівневі мережі Петрі (наприклад, кольорові мережі); інші часові мережі Петрі	графічний редактор; швидка імітація (без графіки); аналіз продуктивності; анімація потоку маркерів
Cosmos	Безкоштовне	традиційні мережі Петрі; стохастичні мережі; стохастичні симетричні мережі	експорт/імпорт моделей (у форматі, що забезпечує інтеграцію з іншими засобами);

			швидка імітація
CPN-AMI	Безкоштовне	високорівневі мережі Петрі; традиційні мережі	графічний редактор; швидка імітація; простори станів; інваріанти позицій; інваріанти переходів; структурний аналіз; експорт/імпорт моделей
CPN Tools	Безкоштовне	високорівневі мережі; інші часові мережі Петрі	графічний редактор; анімація потоку маркерів; швидка імітація; простори станів; аналіз продуктивності; експорт/імпорт моделей
FLOWer	Комерційне (академічні знижки)	традиційні мережі Петрі	графічний редактор; система управління робочими процесами; швидке прототипування
Fluid Survival Tool	Безкоштовне	стохастичні мережі Петрі; інші часові мережі	простий аналіз продуктивності; вдосконалений аналіз продуктивності

GDLoolkit	Комерційне (100% академічні знижки)	високорівневі мережі Петрі; традиційні мережі	автоматичне розміщення елементів
Helena	Безкоштовне	високорівневі мережі Петрі	простори станів; скорочення/стиснення мереж
INA	Безкоштовне	високорівневі мережі Петрі; традиційні мережі; інші часові мережі	простори станів; інваріанти позицій; інваріанти переходів; скорочення/стиснення мереж; структурний аналіз; простий аналіз продуктивності; вдосконалений аналіз продуктивності; експорт/імпорт моделей
JPetriNet	Безкоштовне	традиційні мережі; інші часові мережі	графічний редактор; структурний аналіз
JSARP	Безкоштовне	об'єктно-орієнтовані мережі Петрі	графічний редактор; анімація потоку маркерів; швидка імітація; структурний аналіз
Mercury	Безкоштовне	стохастичні мережі Петрі	інваріанти переходів;

			структурний аналіз; простий аналіз продуктивності; вдосконалений аналіз продуктивності; експорт/імпорт моделей
PACE	Комерційне (академічні знижки)	високорівневі мережі Петрі; традиційні мережі; стохастичні мережі; інші часові мережі Петрі; мережі з атрибутами	графічний редактор; анімація потоку маркерів; швидка імітація; скорочення/стиснен ня мереж
PNetLab	Безкоштовне	високорівневі мережі Петрі; традиційні мережі; інші часові мережі	графічний редактор; анімація потоку маркерів; інваріанти позицій; інваріанти переходів; структурний аналіз; простий аналіз продуктивності; експорт/імпорт моделей

Більшість із вказаних програмних засобів містить графічні редактори, деякі орієнтовані на використання специфічної мови опису мережі Петрі. В основному вони забезпечують можливість роботи з певним класом мереж - традиційними

(базовими) мережами, часовими мережами Петрі, кольоровими мережами. Розглянуті рішення вимагають встановлення на конкретну операційну систему, що значно звужує коло користувачів. Навіть крос-платформні рішення вимагають встановлення віртуальної машини Java під конкретну операційну систему, що може викликати певні незручності. Також, проаналізувавши дані засоби, можна дійти висновку про те, що серед них, на жаль, небагато по-справжньому прикладних реалізованих застосувань, що здатні здійснювати імітацію часових мереж Петрі з достатньо великою кількістю елементів. Як правило, ці програмні засоби більше придатні для ознайомлення з правилами функціонування мереж Петрі, тобто являють собою скоріше навчальні програми [23]. Крім того, їх неможливо доробити і застосувати для своїх цілей у власному проєкті, а можна лише використати на практиці в межах реалізованого функціоналу, якого здебільшого недостатньо для імітаційного моделювання реальних систем. Можна також помітити, що далеко не всі наявні програмні засоби дозволяють працювати зі стохастичними мережами Петрі. Таким чином, в умовах зростаючої популярності веб-ресурсів та відсутності подібних аналогів програмного забезпечення, прийняте рішення про створення веб-орієнтованого середовища моделювання дискретно-подійних систем.

Зазначимо, що спроби знайти хоча б один існуючий засіб, який дозволяє за допомогою графічного інтерфейсу створювати Петрі-об'єктні моделі для подальшого використання в імітаційному моделюванні, не дали жодних результатів, тобто таких засобів наразі не існує.

1.3 Висновки до розділу

З розвитком інформаційних технологій зростає потреба в імітаційному моделюванні складних систем. У зв'язку з цим відбувається зростання вимог до засобів моделювання, зокрема постає задача забезпечення високої швидкості та зручності побудови імітаційних моделей. Переваги веб-застосувань над застосуваннями рівня операційної системи очевидні. На даному етапі веб-технології дозволяють будувати складні веб-застосування, що в більшості не поступаються по

якості і продуктивності застосуванням рівня операційної системи. Нажаль в процесі огляду існуючих рішень з імітаційного моделювання дискретно-подійних систем за допомогою мереж Петрі не виявлено жодного завершеного робочого засобу, що працював би в онлайн режимі.

Мережі Петрі являють собою універсальний засіб формального опису поведінки дискретно-подійних систем. У багатьох наукових роботах відзначається ефективність використання формалізму мереж Петрі для опису та дослідження паралельних процесів та процесів управління. Однак широко відомих розробок технологій імітаційного моделювання, що базуються на мережах Петрі, не існує. Одним із суттєвих недоліків використання даного апарату для моделювання є те, що навіть для опису простих систем мережами Петрі зазвичай необхідно застосовувати велику кількість елементів. Серед існуючих програмних засобів, що використовують мережі Петрі для моделювання систем, небагато дійсно грамотно реалізованих застосувань, які здатні здійснювати імітацію часових мереж Петрі з достатньо великою кількістю елементів. Це призводить до актуальності розробки такого програмного забезпечення, яке дозволило б імітувати поведінку складних дискретно-подійних систем за допомогою технології Петрі-об'єктного моделювання - технології імітаційного моделювання систем, що ґрунтується на стохастичних мережах Петрі та надає можливість створювати моделі складних систем з конструктивних елементів. Доведені ефективність та поліноміальна обчислювальна складність алгоритму імітації Петрі-об'єктної моделі. Визначено, що наразі не існує інших засобів, що дозволяли б за допомогою графічного інтерфейсу створювати Петрі-об'єктні імітаційні моделі.

Застосовуючи візуальне програмування імітаційних моделей, можна забезпечити високу швидкість та гнучкість процесу конструювання імітаційних моделей.

Таким чином, мета даного дослідження полягає в підвищенні швидкості побудови моделей складних дискретно-подійних систем і зручності їх візуального представлення шляхом розробки такого програмного забезпечення, яке дозволило б імітувати поведінку складних систем за допомогою технології Петрі-об'єктного

моделювання та візуального програмування, швидко будувати і модифікувати імітаційні моделі та яке б забезпечувало зручність представлення моделей користувачу.

Для досягнення поставленої мети необхідно виконати наступні завдання: виконати огляд відомих засобів автоматизованого імітаційного моделювання дискретно-подійних систем; детально ознайомитися з технологією Петрі-об'єктного моделювання, виконати її порівняльний аналіз з іншими технологіями моделювання дискретно-подійних систем; виконати огляд існуючих засобів моделювання, що базуються на мережах Петрі, в тому числі здійснити пошук таких засобів, які дозволяють візуально програмувати Петрі-об'єктні моделі для подальшого використання в імітаційному моделюванні; прийняти рішення щодо візуального представлення елементів мереж Петрі та Петрі-об'єктів у майбутній системі візуального програмування мереж Петрі та Петрі-об'єктних моделей (створення Петрі-об'єктів і конструювання зв'язків між ними); спроектувати дану систему з використанням алгоритму імітації Петрі-об'єктних моделей та звичайних стохастичних мереж Петрі; виконати програмну реалізацію спроектованої системи; провести тестування розробленої системи на конкретних моделях; виконати аналіз коректності роботи та швидкодії розробленого програмного продукту.

2 МАТЕМАТИЧНИЙ АПАРАТ ПЕТРІ-ОБ'ЄКТНОГО МОДЕЛЮВАННЯ

2.1 Мережі Петрі як засіб формалізації дискретно-подійних систем

Мережі Петрі винайдені Карлом Адамом Петрі у 1939 році і описані ним у його докторській дисертації у 1962 [Petri C. Kommunikation mit Automaten, 1962]. Мережа Петрі є орієнтованим дводольним графом, що має чотири основних типи елементів.

Перехід		позначає подію
Позиція	○	позначає умову
Дуга	○ →	позначає зв'язки між подіями та умовами
Маркер	○ •	позначає виконання (або не виконання) умови

Рисунок 2.1 – Елементи мережі Петрі

Для позначення великої кількості маркерів в позиції, будемо використовувати замість традиційного позначення (у вигляді крапки) число, яке означає кількість маркерів. На рисунку 2.2 представлений приклад позначення великої кількості маркерів у позиції.

Багато маркерів ○56○

Рисунок 2.2 – Позначення багатократного виконання умови

Якщо передумовою запуску переходу є наявність не однієї, а певної кількості маркерів у позиції, то між позицією та переходом існує не один, а декілька зв'язків. Для великої кількості зв'язків будемо використовувати позначення групи зв'язків, як показано на рисунку 2.3.

Багато дуг $\xrightarrow{16}$

Рисунок 2.3 – Позначення групи зв'язків

Класичну мережу Петрі також доповнюють такі поняття як:

- часові затримки;
- багатоканальні переходи;
- конфліктні переходи;
- інформаційні зв'язки.

Ці поняття введені для більш точного відтворення реальної складної системи, що моделюється. Розглянемо їх детальніше.

Часові затримки є додатковими властивостями переходів. Запуск переходу з часовою затримкою складається з двох дій, виконуваних у різні моменти часу. По-перше, при виконанні умови запуску переходу маркери з всіх вхідних позицій переходу віднімаються та запам'ятовується момент виходу маркерів як поточний час моделювання плюс час затримки переходу. По-друге, у запам'ятований момент виходу маркерів здійснюється їх додавання в усі вихідні позиції переходу [17]. Якщо перехід моделює деякий пристрій у системі, то часова затримка може відповідати часу, необхідному пристрою для виконання одиниці своєї роботи.

Наявність багатоканальних переходів передбачає наявність у кожного переходу такого параметра як кількість каналів, що може мати значення 1 або більше (додатне ціле число або нескінченність). При збудженні переходу з двома чи більше каналами запуск переходу може бути здійснено декілька разів підряд - доки виконується умова запуску (скільки цього дозволяють маркери у вхідних позиціях). При цьому при кожному запуску переходу (активізації одного з його каналів) запам'ятовується момент виходу маркерів з нього [17].

Наявність конфліктних переходів зумовлює необхідність розробки правил розв'язання конфліктів, що виникають при одночасному збудженні декількох переходів. За правилами класичних мереж Петрі при виникненні конфлікту необхідно обрати перехід, який запускається, за допомогою випадкового числа. Потім слід його

запустити, і тільки після цього можуть запуститися інші переходи. Існують наступні способи розв'язання конфліктів: рівноймовірнісний, пріоритетний та ймовірнісний. При першому способі переходи, що конфліктують, запускаються з рівною ймовірністю. При пріоритетному способі запускається той перехід, для якого вказано вищий пріоритет. При ймовірнісному способі для конфліктних переходів вказується ймовірність запуску переходу, і першим запускається той перехід, на який вказало випадкове число [17].

Інформаційний зв'язок - це особливий тип вхідної дуги. Якщо одна з вхідних дуг переходу являє собою інформаційний зв'язок, то, як і завжди, для запуску переходу необхідною умовою є наявність певної кількості маркерів у позиції, яку пов'язує ця дуга з даним переходом. Особливість полягає в тому, що після запуску такого переходу маркери з позиції, яку пов'язує з переходом інформаційна дуга, не віднімаються. На практиці цьому може відповідати, наприклад, дозвіл на виконання певних операцій. Інформаційні зв'язки позначаються пунктирними відрізками. Фрагмент мережі Петрі з інформаційним зв'язком наведено на рисунку 2.4.

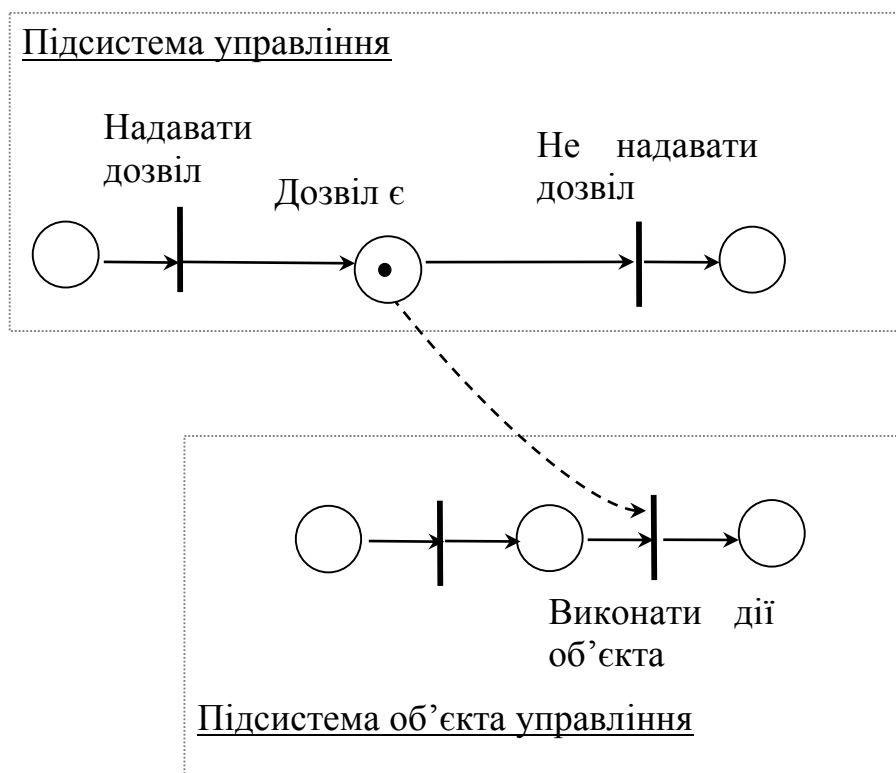


Рисунок 2.4 – Фрагмент мережі Петрі, що моделює надання дозволу на виконання дії об'єкта, який надається підсистемою управління [33]

Результатом функціонування об'єкта управління є змінювання стану об'єкта, але не підсистеми управління. Якщо підсистема об'єкта управління «знає» про підсистему об'єкта управління «щось», то це призводить до корегування правил функціонування об'єкта управління, але не змінює стан підсистеми об'єкта управління. Саме для відтворення зв'язків виду «знає, але не змінює» призначені інформаційні зв'язки [33].

Мережею Петрі з часовими затримками, з конфліктними та багатоканальними переходами, з інформаційними зв'язками називатимемо $\text{PetriNet} = (P, T, A, W, K, I, R)$, де $P = \{P\}$ - множина позицій, $T = \{T\}$ - множина переходів, $P \cap T = \emptyset$, $A \subseteq P \times T \cup T \times P$ - множина дуг, $W: A \rightarrow N$ - множина натуральних значень, що задає кількість зв'язків, $K: T \rightarrow Z_+ \times [0; 1]$ - статус конфліктних переходів, що складається зі значення пріоритету запуску переходу та числового значення (ваги), що характеризує ймовірність запуску переходу, $I \subseteq P \times T$ - множина інформаційних зв'язків, $R: T \rightarrow \mathfrak{R}_+$ - невід'ємні часові затримки в переходах, що представляються випадковою величиною із заданим законом розподілу або детермінованою величиною із заданим значенням, N - множина натуральних чисел, Z_+ - множина цілих невід'ємних чисел, \mathfrak{R}_+ - множина дійсних невід'ємних чисел [13].

Зауважимо, що будь-який перехід мережі Петрі повинен мати непорожню множину вхідних позицій та непорожню множину вихідних позицій: $\forall T \in T \ T \neq \emptyset \wedge T \neq \emptyset$. В протилежному випадку мережа Петрі не має змісту. Дійсно, якщо перехід має порожню множину вхідних позицій, то не має змісту вхід маркерів в перехід, а, якщо перехід має порожню множину вихідних позицій, то не має змісту вихід маркерів з переходу.

Стан мережі Петрі в кожний момент часу t описується станом її позицій $M(t)$ та станом її переходів $E(t)$:

$$S(t) = (M(t), E(t)), \quad (2.1)$$

де $\forall P \in P \ M_P(t) \in Z_+$ - вектор маркірування позицій, Z_+ - множина цілих невід'ємних чисел, $E(t) = \{E_T(t) | T \in T\}$ - вектор станів переходів.

Стан переходу $E_T(t)$ визначається множиною моментів виходу з переходу маркерів, які на момент часу t знаходяться в переході:

$$\forall q \in N [E_T(t)]_q | [E_T(t)]_q \in \mathfrak{R}_+ \quad (2.2)$$

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

Якщо всі канали переходу вільні (не активні), то стану переходу присвоюється значення $E_T(t) = \{\infty\}$, тобто «не очікується вихід маркерів з переходу». Тоді найближчий момент виходу маркерів з переходу визначається найменшим з усіх значень масиву $E_T(t)$. Введемо змінну $\tau_T(t)$, що визначає найближчий момент виходу маркерів з переходу T :

$$\tau_T(t) = \min_q [E_T(t)]_q. \quad (2.3)$$

Момент найближчої події визначається найменшим з усіх значень $\tau_T(t_{n-1})$:

$$t_n = \min_T \tau_T(t_{n-1}). \quad (2.4)$$

Перехід мережі Петрі називається активним, якщо в його каналах є маркери, що очікують виходу. Очевидно, що перехід мережі Петрі активний в момент часу t тоді і тільки тоді, коли $\tau_T(t) < \infty$.

В мережі Петрі з часовими затримками маркери, при виконанні умови запуску, входять в один із вільних (не активних) каналів переходу, і протягом часу, рівного часовій затримці цього переходу R_T , канал знаходиться у стані «активний». Як тільки сплине часова затримка R_T , здійснюється вихід маркерів з переходу. Змінювання маркування в результаті виходу маркерів з переходу призводить до того, що стають, можливо, виконаними умови запуску для деяких переходів мережі Петрі. У кожний момент часу, який відповідає події, здійснюється вихід маркерів з переходів, для яких настав момент виходу, і вхід маркерів в переходи, для яких виконана умова запуску. Таким чином, для мережі Петрі з часовими затримками традиційне поняття запуску переходу, що поєднує в собі вхід та вихід маркерів з переходу, не придатне. Слід розрізняти вхід маркерів в переходи мережі Петрі та вихід маркерів з переходів мережі Петрі.

Вхід маркерів у перехід мережі Петрі здійснюється, якщо і тільки якщо в його вхідних позиціях є кількість маркерів кількості, рівній кількості зв'язків. При вході

маркерів в перехід мережі Петрі з усіх вхідних позицій переходу, зв'язок яких з ним не є інформаційним, маркери віднімаються у кількості, рівній кількості зв'язків.

Вихід маркерів з каналу багатоканального переходу мережі Петрі здійснюється, якщо поточний момент часу співпадає з моментом виходу з каналу переходу. При виході маркерів з каналу багатоканального переходу в усі вихідні позиції переходу маркери додаються у кількості, рівній кількості зв'язків.

Поставимо у відповідність виходу маркерів з переходів мережі Петрі перетворення D^+ стану мережі $S(t)$, а входу маркерів в переходи мережі Петрі перетворення D^- стану мережі $S(t)$. Оскільки переходи мережі Петрі багатоканальні, то потрібно виконувати m -кратний вхід маркерів в переходи мережі Петрі до досягнення такого стану мережі Петрі, при якому для жодного з її переходів не виконана умова запуску переходу. В протилежному випадку порушується логіка функціонування мережі Петрі.

Таким чином, перетворення стану мережі Петрі в кожний момент часу t_n здійснюється послідовним виконанням перетворення D^+ і $(D^-)^m$. Позначимо стан мережі Петрі, який є результатом виходу маркерів з переходу, $S^+(t_n)$. Тоді в кожний момент часу t_n стан мережі Петрі перетворюється у два етапи:

$$S(t_{n-1}) \xrightarrow{D^+} S^+(t_n) \xrightarrow{(D^-)^m} S(t_n). \quad (2.5)$$

Перетворення мережі Петрі продовжуються, доки не буде вичерпаний заданий час моделювання або не буде досягнутий стан мережі Петрі, при якому жоден з її переходів не запускається і моменти виходу маркерів в усіх її переходах більші за час моделювання.

Петрі-об'єктний підхід до моделювання дискретно-подійних систем

Поняття об'єктно-орієнтованого підходу являє собою парадигму програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою. Першою мовою, що реалізує концепцію об'єктно-орієнтованого програмування традиційно вважають мову SIMULA, яка описана у підрозділі 1.1. Об'єктно-орієнтований підхід є потужним та ефективним засобом конструювання великих систем, що складаються з однотипних елементів. Застосування цієї

парадигми при складанні алгоритмів імітації має певні особливості, пов'язані з тим, що програмісту необхідно описувати велику кількість можливих подій для кожного класу. З цього випливає, що процес розробки може викликати помилки, які складно відшукати та займати багато часу.

В технології Петрі-об'єктного моделювання, яка запропонована в роботі [24], пропонується використовувати комбінований підхід: представлення структури моделі у вигляді об'єктів і представлення динаміки функціонування об'єктів моделі засобами стохастичних мереж Петрі з часовими затримками, інформаційними зв'язками, з конфліктними та багатоканальними переходами. Даний підхід до побудови імітаційної моделі складної системи дозволить швидко конструювати алгоритм імітації великих та складних систем з дискретно-подійним функціонуванням.

Кожна Петрі-об'єктна модель складається з об'єктів, які мають динамічні властивості, описані засобами стохастичної мережі Петрі. Такі об'єкти отримали назву *Петрі-об'єкт*. Усі Петрі-об'єкти створюються як нащадки одного класу *PetriSim*, який містить алгоритм імітації стохастичної мережі Петрі. Таким чином для всіх Петрі-об'єктів задані однакові правила функціонування, а застосування об'єктно-орієнтованого підходу забезпечує тиражування об'єктів з заданою функціональністю у будь-якій кількості.

Зв'язки Петрі-об'єктів між собою реалізуються двома способами:

- за допомогою спільних позицій (одна і та сама позиція належить мережам Петрі декількох різних Петрі-об'єктів; алгоритмічно існування спільної позиції можна забезпечити, наприклад, співпадінням адрес пам'яті, де зберігаються значення маркування відповідних позицій Петрі-об'єктів);
- за допомогою ініціалізації подій (із переходу мережі одного Петрі-об'єкта при кожному виході маркерів передаються маркери в позицію мережі іншого Петрі-об'єкта в заданій кількості та в момент часу, що відповідає моменту виходу маркерів із переходу) [23].

Формування зв'язків між об'єктами першим способом продемонстровано на рисунку 2.5, другим - на рисунку 2.6

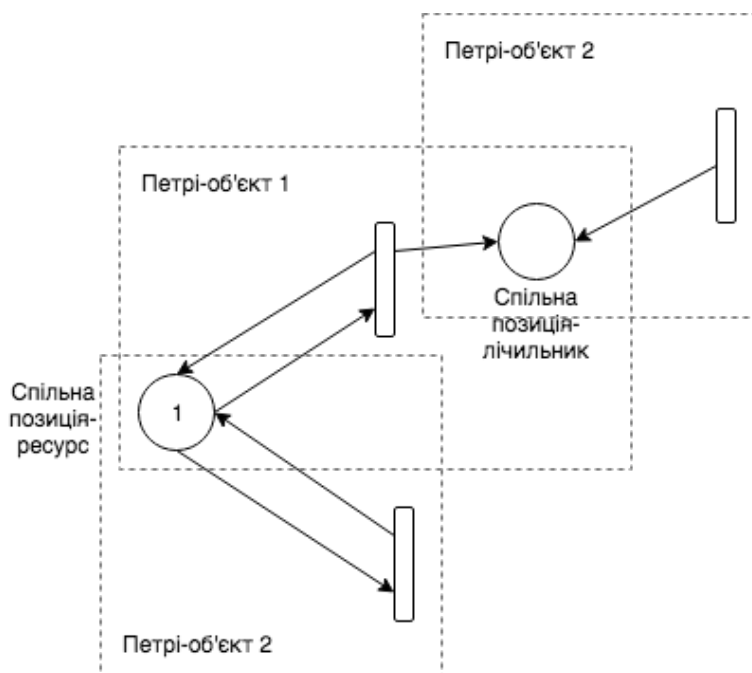


Рисунок 2.5 – Приклад зв'язку за допомогою спільних позицій [24]

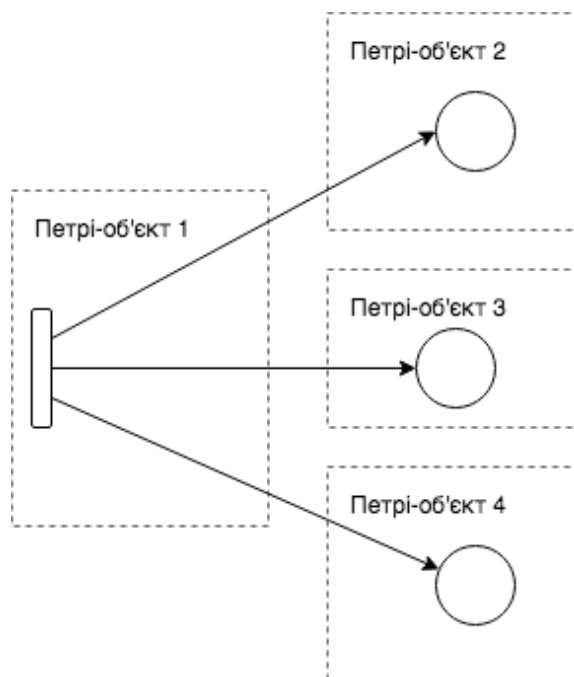


Рисунок 2.6 – Приклад зв'язку за допомогою ініціалізації подій [24]

2.2 Математичний опис Петрі-об'єктної моделі

Математичний опис мережі Петрі-об'єкта аналогічний опису стохастичної мережі Петрі наведеному у підрозділі 2.1. Кожний Петрі-об'єкт характеризується своєю множиною переходів. Перехід, що належить до множини переходів одного Петрі-об'єкта, не може належати множині переходів іншого Петрі-об'єкта. Іншими

словами, множини переходів Петрі-об'єктів не перетинаються. Тому існує однозначна відповідність між переходом та Петрі-об'єктом.

Математичний опис функціонування Петрі об'єкта має вигляд:

$$\begin{cases} t_n = \min_{T \in \mathbf{T}_N} \tau_T(t_{n-1}), t_n \geq t_{n-1}, \\ \mathbf{S}_N(t_1) = (D^-)^m(\mathbf{S}_N(t_0)), \\ \mathbf{S}_N(t_n) = (D^-)^m(D^+(\mathbf{S}_N(t_{n-1}))), \\ n = 2, 3, \dots \end{cases} \quad (2.6)$$

де N – мережа Петрі відповідного Петрі-об'єкта, $\mathbf{S}_N(t_n)$ - стан Петрі-об'єкта.

Петрі-об'єктна модель описується стохастичною мережею Петрі з часовими затримками, що є об'єднанням мереж Петрі-об'єктів, з яких вона складається. Стан Петрі-об'єктної моделі повністю описується станом мереж Петрі-об'єктів, з яких вона складається, оскільки і позиції, і переходи належать тільки Петрі-об'єктам моделі.

Об'єктно-орієнтована модель системи представляється множиною взаємопов'язаних об'єктів $Model = \{O_1, O_2, \dots, O_n\}$. Кожний об'єкт O_i представляє певну алгоритмічну сутність і має задану функціональну здатність, опис якої міститься в описі класу, до якого належить об'єкт: $O_i \subset C_j$.

Петрі-об'єктною моделлю називається модель, що є результатом агрегування Петрі-об'єктів:

$$Model = \bigcup_N O_N, \quad (2.7)$$

де $O_N \xrightarrow{inherit} PetriSim$.

Рівняння станів Петрі-об'єктної моделі мають вигляд:

$$\begin{aligned} t_n &= \min_N \tau_N, t_n \geq t_{n-1}, & (2.34) \\ \mathbf{S}(t_n) &= \begin{pmatrix} (D^-)^m(D^+(\tilde{\mathbf{S}}_1(t_{n-1}))) \\ \dots \\ (D^-)^m(D^+(\tilde{\mathbf{S}}_N(t_{n-1}))) \\ \dots \\ (D^-)^m(D^+(\tilde{\mathbf{S}}_L(t_{n-1}))) \end{pmatrix}, \quad \forall \tilde{\mathbf{S}}_N(t_n): \bigvee_{T \in \mathbf{T}_N} Z(T, t_n) = 0, & (2.35) \end{aligned}$$

де m - кількість перетворень D^- , що здійснюються до досягнення стану Петрі-об'єкта, при якому жоден із його переходів не запускається [24].

Розбиття на Петрі-об'єкти дозволяє значно скоротити кількість елементарних операцій, необхідних для здійснення перетворення мережі Петрі. Перетворення $D^+(\tilde{\mathbf{S}}_N(t_{n-1}))$ необхідно фактично здійснювати лише в тому випадку, якщо $\tau_N = t_n$, оскільки в іншому випадку значення предикатів $X(T, t_n) = 0 \quad \forall T \in \mathbf{T}_N$ і здійснення перетворення $D^+(\tilde{\mathbf{S}}_N(t_{n-1}))$ не призведе до зміни стану Петрі-об'єкта. Перетворення $(D^+)^m$, що здійснюються для кожного Петрі-об'єкта, потребують фактичного виконання m_j -кратного ($m_j \leq m$) входу маркерів у переходи до досягнення стану, при якому жоден із переходів Петрі-об'єкта не запускається. Оскільки $m_j < m$ для більшості Петрі-об'єктів і ще для більшої кількості Петрі-об'єктів $m_j = 0$ (скільки для них час виходу маркерів не співпадає з поточним моментом часу і не змінилось їх маркування в поточний момент часу), то фактична кількість входів маркерів у переходи окремих Петрі-об'єктів значно менше кількості m входів маркерів у переходи Петрі-моделі [24].

2.3 Алгоритм імітації Петрі-об'єктної моделі

Алгоритм імітації Петрі-об'єктної моделі можна сформулювати наступним чином [23]:

Крок 1. Сформувати список Петрі-об'єктів.

Крок 2. Здійснити початковий вхід маркерів у переходи для всіх Петрі-об'єктів.

Крок 3. Поки не вичерпано час моделювання:

Крок 3.1. Здійснити статистичні розрахунки.

Крок 3.2. Для всіх Петрі-об'єктів:

Крок 3.2.1. Знайти найближчий момент виходу маркерів із переходів.

Крок 3.3. Просунути час у момент найближчої події.

Крок 3.4 Для всіх Петрі-об'єктів, для яких момент виходу маркерів співпадає з поточним моментом часу:

Крок 3.4.1. Виконати вихід маркерів із переходів.

Крок 3.4.2. Здійснити перетворення стану.

Крок 3.5. Для всіх Петрі-об'єктів:

Крок 3.5.1. Виконати вхід маркерів у переходи.

Крок 4. Вивести результати моделювання.

З алгоритмічної точки зору розбиття на Петрі-об'єкти дозволяє значно скоротити кількість елементарних операцій, необхідних для реалізації перетворення мережі Петрі. Перетворення D^+ мереж Петрі-об'єктів слід фактично виконувати лише для тих об'єктів, для яких момент виходу маркерів із переходів співпадає з поточним моментом часу. Перетворення $(D^-)^m$, що виконується для кожного Петрі-об'єкта, вимагає фактичного виконання m_j -кратного (m_j не більше m) входу маркерів у переходи до досягнення стану, при якому жоден із переходів Петрі-об'єкта не запускається. Оскільки для великої кількості Петрі-об'єктів $m_j < m$, а для ще більшої кількості Петрі-об'єктів $m_j = 0$ (тому що для них час виходу маркерів не співпадає з поточним моментом часу), то фактична кількість входів маркерів у переходи окремих Петрі-об'єктів значно менше кількості m входів маркерів у переходи Петрі-моделі [23].

2.4 Процес побудови Петрі-об'єктної моделі

Основне призначення технології Петрі-об'єктного моделювання – створити умови для розробки та удосконалення моделей систем, що характеризуються великою кількістю елементів, наявністю процесів управління та процесів прийняття рішень, наявністю паралельних процесів, недетермінованістю та дискретністю функціонування.

Структурні та динамічні елементи системи виділяються з використанням методів системного аналізу. Визначення класів виконується з використанням методів об'єктно-орієнтованого моделювання. Петрі-об'єкти та взаємозв'язки між ними описуються з використанням теорії Петрі-об'єктного моделювання, викладеної у [25].

Петрі-об'єктне моделювання деякої системи можна представити такою послідовністю етапів:

- побудова концептуальної моделі системи (розбиття на підсистеми і елементи, а також визначення чисельних параметрів системи);
- об'єктно-орієнтований аналіз та моделювання (визначення класів);
- формалізований опис Петрі-об'єктів;
- формалізований опис Петрі-об'єктної моделі;
- реалізація моделі;
- дослідження моделі [30].

Дані етапи наведені в загальній схемі процесу Петрі-об'єктного моделювання, що представлена на рисунку 2.7.

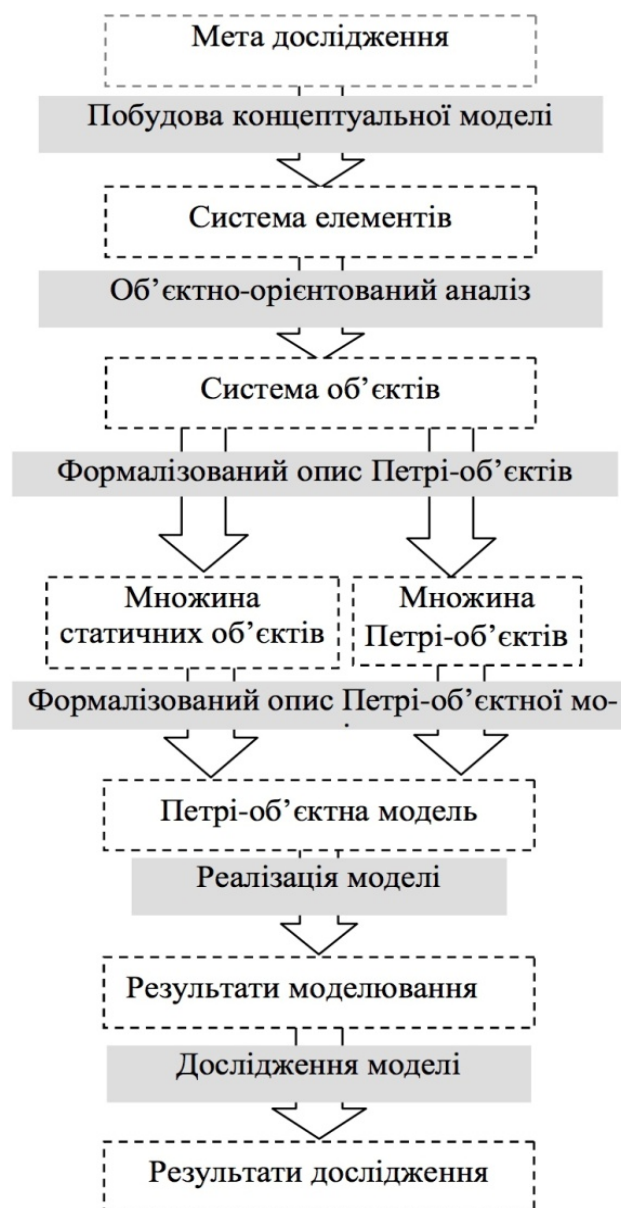


Рис. 2.7 – Процес Петрі-об'єктного моделювання [30]

Об'єктно-орієнтований аналіз системи має на меті поставити у відповідність елементам системи, які визначені на етапі побудови концептуальної моделі системи, множину об'єктів, що відтворюють певну функціональність кожного елементу системи. Якщо на етапі побудови концептуальної моделі системи дослідником виконується розбиття системи на елементи та встановлення взаємозв'язків між елементами системи, то на етапі об'єктно-орієнтованого аналізу кожному елементу системи ставиться у відповідність один чи кілька об'єктів програмування. Об'єктно-орієнтовану модель системи представляють у вигляді діаграми класів, в якій указуються основні зв'язки між класами об'єктів – агрегування, наслідування, залежність [45]. Приклад діаграми класів Петрі-об'єктної моделі наведений на рис. 2.8.

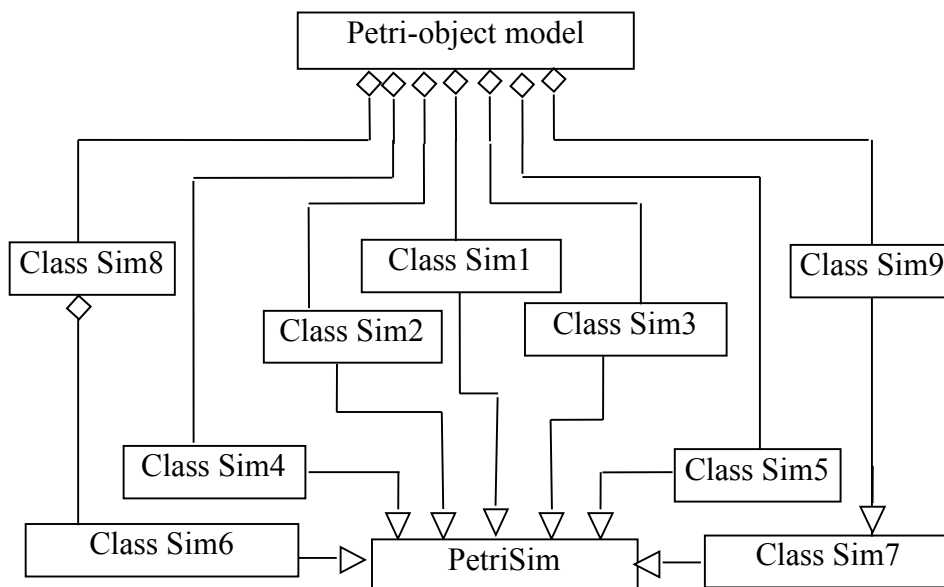


Рис. 2.8. Приклад діаграми класів Петрі-об'єктної моделі системи

Множина об'єктів моделі поділяється на статичні та динамічні об'єкти в залежності від того, чи володіє об'єкт певною функціональністю. Наприклад, об'єкт Студент в моделі навчального процесу призначений для відтворення подій, пов'язаних з навчанням студента, тому цей об'єкт динамічний. Об'єкт Журнал тієї ж моделі – статичний, оскільки він призначений тільки для відтворення інформації про перебіг навчання одного студента з однієї дисципліни.

Статичні об'єкти моделі описуються звичайними засобами об'єктно-орієнтованого програмування і містять набір власних полів та методів для відображення інформації про свій поточний стан. Динамічні об'єкти описуються з використанням поняття Петрі-об'єкта.

Петрі-об'єкт створюється як нащадок об'єкту Петрі-імітатор (PetriSim) з використанням механізму наслідування об'єктно-орієнтованого програмування. Об'єкт Петрі-імітатор (PetriSim) реалізує імітацію деякого реального об'єкта у відповідності до динаміки функціонування, що задається стохастичною мережею Петрі з часовими затримками, з конфліктними та багатоканальними переходами, з інформаційними зв'язками. Технологія розробки стохастичної мережі Петрі з часовими затримками викладена у п. 3.1.

Для того, щоб створити Петрі-об'єкт потрібно:

- успадкувати Петрі-імітатор
- передати в поле Net Петрі-об'єкта мережу Петрі, що описує динаміку функціонування
- описати власні поля та методи Петрі-об'єкту, специфічні для його функціонування. Наприклад, Петрі-об'єкт Студент містить поля, що характеризують індивідуальні характеристики студента – його ім'я, успішність, відмітки про складання заліків та екзаменів та інше.

Петрі-об'єктна модель створюється з Петрі-об'єктів та звичайних об'єктів за допомогою зв'язків, що передбачені об'єктно-орієнтованим програмуванням, а також, зв'язків між Петрі-об'єктами, що відтворюють вплив динаміки функціонування одного Петрі-об'єкта на інший.

Зв'язки Петрі-об'єктів між собою здійснюються за допомогою спільних позицій (спільна позиція є позицією мережі Петрі різних Петрі-об'єктів) та за допомогою ініціалізації подій (з переходу мережі Петрі-об'єкта при кожному виході маркерів з переходу передаються маркери в позицію мережі Петрі іншого Петрі-об'єкта в заданій кількості в момент часу, який співпадає з моментом виходу маркерів з переходу). Зауважимо, що у переважній більшості моделей достатньо використання тільки спільних позицій. Ініціалізація подій необхідна у тих випадках, коли потрібно

передати маркери у множину Петрі-об'єктів, яка залежить від поточного стану моделі чи поточного моменту часу.

Дослідження моделі виконують з використанням класичних методів факторного аналізу, а також сучасних методів дослідження багато-параметричних систем таких, як методи групового урахування аргументів, еволюційні методи відшукування оптимальних значень та інші.

2.5 Висновки до розділу

Формалізація процесів функціонування системи засобами мереж Петрі з інформаційними зв'язками дозволяє ще на етапі формалізації описати складні взаємозв'язки між елементами управляючої системи з урахуванням процесів, що відбуваються в часі паралельно. Високий рівень деталізації формального опису значно зменшує кількість помилок при побудові імітаційної моделі, а значить підвищує якість побудованої моделі.

Основне призначення Петрі-об'єктного моделювання – розробка та дослідження імітаційних моделей великих систем, таких, як система управління транспортним рухом, система управління навчальним процесом ВНЗ, та інших. Моделювання системи за допомогою Петрі-об'єктів дозволяє досліднику зосередитись на складанні мереж Петрі елементів системи. При цьому мережа Петрі-об'єкта відображає поведінкові властивості елемента системи. Налаштування Петрі-об'єкта може бути виконане до об'єднання в систему и не складне, якщо Петрі-об'єкт достатньо простий. Тільки після налаштування Петрі-об'єктів, що представляють елементи системи, виконується конструювання моделі системи. Також однією з переваг використання Петрі-об'єктної моделі є те, що функціонування Петрі-об'єктної моделі може бути розбите на функціонування її Петрі-об'єктів, що зменшує витрати на аналітичне дослідження моделі системи.

Таким чином, визначені процес, алгоритми імітації та методи дослідження Петрі-об'єктної моделі, що створюють Петрі-об'єктну технологію моделювання систем.

3 ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ МОДЕЛЮВАННЯ ДИСКРЕТНО-ПОДІЙНИХ СИСТЕМ

3.1 Архітектура

Для розробки веб-сервісу моделювання дискретно-подійних систем вирішено використовувати клієнт-серверну архітектуру. Архітектура клієнт-сервер є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Розглянемо основні поняття клієнт-серверної архітектури.

Сервер – це об'єкт, що надає сервіс іншим об'єктам мережі за їх запитамі. Сервер працює за завданнями клієнтів і управляє виконанням їх завдань. Після виконання кожного завдання сервер посилає отримані результати клієнту, який послав це завдання.

Сервіс – це процес обслуговування клієнтів. Його функція описується комплексом прикладних програм, відповідно до якого виконуються різноманітні прикладні процеси.

Процес, який викликає сервісну функцію за допомогою певних операцій, називається клієнтом. Ним може бути програма або користувач. Клієнти – це робочі станції, які використовують ресурси сервера і надають зручні інтерфейси користувача. Інтерфейси користувача це процедури взаємодії користувача з системою або мережею. Клієнт є ініціатором і використовує електронну пошту або інші сервіси сервера. У цьому процесі клієнт запитує вид обслуговування, встановлює сеанс, отримує потрібні йому результати і повідомляє про закінчення роботи.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися як до одного сервера, так і до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів. Схема клієнт-серверної архітектури наведена на рисунку 3.1

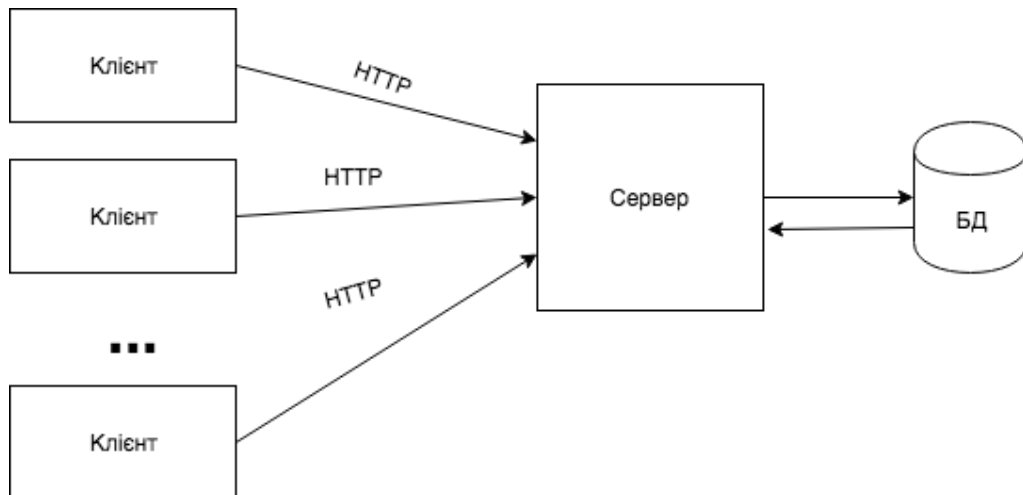


Рисунок. 3.1 – Клієнт-серверна архітектура

Використання клієнт-серверної архітектури у даному проекті має наступні переваги:

- виконання алгоритму імітації відбувається на єдиному сервері, що забезпечує стабільність результатів та незалежність від потужності клієнтської машини;
- моделі користувачів, а також результати виконання зберігаються в єдиному місці в обліковому записі користувача;
- програма може виконуватися на будь-якому комп'ютері, оскільки єдина вимога до технічного забезпечення полягає в тому, щоб на комп'ютері було встановлено сучасний веб-браузер;
- можливість реалізації колективної співпраці над моделями.

3.2 Засоби розробки

3.2.1 Вибір мови розробки клієнтської частини

Для розробки клієнтської частини Петрі-об'єктного моделювання було обрано мову JavaScript як єдине рішення, що підтримується будь-яким сучасним браузером. Для швидкої побудови html сторінок та маніпулюванням деревом елементів вирішено використовувати бібліотеку Javascript React.js, що розроблена компанією Facebook.

React.js – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, які трапляються при розробці односторінкових застосувань. Розробляється компаніями Facebook та Instagram, а також спільнотою індивідуальних розробників. Його мета полягає в тому, щоб бути швидким, простим та масштабованим.

ReactJS дозволяє розробникам створювати великі веб-застосунки, які використовують дані, що змінюються з часом, без перезавантаження сторінки. Це реалізовано через віртуальну об'єктну модель документа. React.js не покладається виключно на DOM браузера. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки потребують оновлення. React.js обробляє тільки інтерфейс користувача. Це відповідає view у шаблоні model-view-controller (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або великими фреймворками MVC, таких як AngularJS. Він також може бути використаний разом з React.js на основі надбудов.

Наразі ReactJS використовують Facebook, Instagram, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші.

Бібліотека ReactJS набула високої популярності та має широке ком'юніті, що забезпечує регулярні оновлення та вдосконалення.

3.2.2 Вибір мови розробки серверної частини

Для розробки серверної частини вирішено відмовитись від мови Ruby на користь мови Java. Виконано експериментальне дослідження часових показників імітації Петрі-об'єкта, що складався з n Петрі-об'єктів з $k = 5$ послідовних подій для кожного. Загальні позиції використовуються для з'єднання об'єктів послідовно. Ця схема моделі дозволяє гнучко контролювати кількість подій nk . Беручи один Petri-об'єкт nk з послідовними явищами отримаємо симуляцію стохастичної мережі Петрі. Це дозволяє порівнювати реалізації моделі Петрі-об'єкта і стохастичної мережі Петрі. Результати виконання алгоритму імітації на мовах Ruby та Java, а також співвідношення часу виконання Ruby до часу виконання Java показані у табл. 3.1, 3.2, 3.3 відповідно. Ці результати підтверджують теоретичну поліноміальну оцінку моделі складності якої показана в [33]. Алгоритм моделювання Petri-об'єктної моделі забезпечує скорочення часу в два рази в порівнянні з моделюванням стохастичних мереж Петрі.

Табл. 3.1 – Час виконання алгоритму імітації на мові Ruby

Параметри		tRuby		
Кількість об'єктів	Кількість переходів, n	Час викон. Петрі-об'єктної моделі, t_{PO}	Час викон. Мережі Петрі, t	t/t_{PO}
20	100	343.73	548.95	1.6
40	200	1238.15	2094.15	1.7
60	300	2862.23	6080.37	2.1
80	400	4920.18	11206.46	2.3
100	500	6917.14	14131.52	2.0

Табл. 3.2 – Час виконання алгоритму імітації на мові Java

Параметри		tJava		
Кількість об'єктів	Кількість переходів, n	Час викон. Петрі-об'єктної моделі, tPO	Час викон. Мережі Петрі, t	t/tPO
20	100	4.54	8.37	1.8
40	200	17.11	32.22	1.9
60	300	38.88	69.61	1.8
80	400	70.90	118.73	1.7
100	500	122.64	177.73	1.4

Табл. 3.3 – Співвідношення часу виконання на мовах Ruby і Java

tRuby/tJava	
Час викон. Петрі-об'єктної моделі, tPO	Час викон. Мережі Петрі, t
76	66
72	65
74	87
69	94
56	80

3.2.3 Вибір середовища розробки

При розробці програмного продукту було використано інтегроване середовище розробки Microsoft Visual Studio Code для клієнтської частини та NetBeans для серверної сторони. Для керування версіями проекту було обрано Git, одну з найефективніших, надійних і високопродуктивних систем керування версіями.

3.3 Проектування клієнтського застосунку

Спроектвана структура директорій проекту наведена на рисунку 3.2.

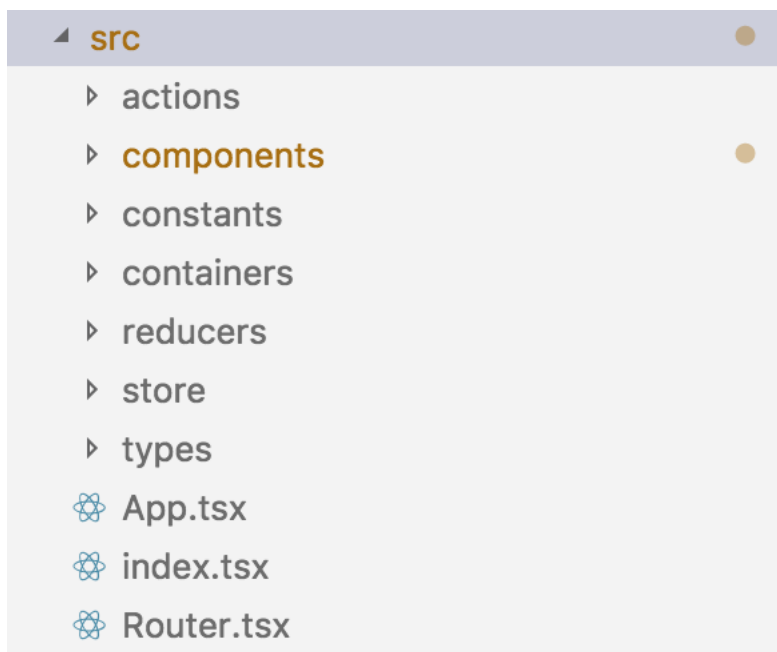


Рисунок 3.1 – Структура директорій проекту

Директорія *src* є кореневою та містить вихідний код застосування. Папка *components* містить React.js компоненти. Папки *actions*, *containers*, *reducers* та *store* забезпечують реалізацію патерну *Redux*. Директорія *types* містить файли, що описують типи даних, які використовуються в застосування. Папка *constants* містить файли, в яких описані глобальні константи проекту. Файл *index.tsx* є точкою входу. *App.tsx* – основний компонент. *Router.tsx* – містить інформацію про маршрутизацію між сторінками.

3.3.1 Вхідні дані

Вхідні дані візуального середовища моделювання дискретно-подійних систем надходять від користувачів. Інтерфейс користувача передбачує створення необхідної структури вхідних даних для подальшої обробки на сервері.

У якості вхідних даних виступає Петрі-об'єктна модель, що побудована за допомогою візуальних компонентів технології Петрі-об'єктного моделювання. На рисунку 3.4 представлений інтерфейс *IModel*, що описує поля об'єкту Петрі-об'єктної моделі та їх тип.

```
export interface IModel {
  id: string;
  params: IModelParams;
  petriObjects: Array<IPetriObject>;
}
```

Рисунок 3.2 – Інтерфейс Петрі-об’єктної моделі

Поле *id* містить унікальний ідентифікатор моделі типу *string*. Інтерфейс, що описує параметри моделювання представлений на рисунку 3.3 та включає в себе час, протягом якого виконуватиметься імітація. Час моделювання задається цілим невід’ємним числом.

```
interface IModelParams {
  simulationTime: number;
}
```

Рисунок 3.3 – Інтерфейс параметрів моделі

Також Петрі-об’єктна модель містить в собі список Петрі-об’єктів, з яких вона складається. Інтерфейс, що описує структуру Петрі-об’єкта представлений на рисунку 3.4.

```
export interface IPetriObject {
  id: string;
  petriNetId: number;
  params: Array<IPetriNetElement>;
}
```

Рисунок 3.4 – Інтерфейс Петрі-об’єкту

Поле *id* є унікальним ідентифікатором моделі та має тип *string*. Оскільки кожен Петрі-об’єкт задається відповідною мережею Петрі із заданими параметрами, виникає необхідність в існуванні ще двох полів: *petriNetId* та *params*. Поле *petriNetId* є ідентифікатором мережі Петрі. Поле *params* – список елементів відповідної мережі Петрі, параметри яких задаються користувачем.

Об’єкт, що описує мережу Петрі, зберігає в собі масив елементів мережі Петрі, масив дуг, а також допоміжні поля для забезпечення необхідного функціоналу

візуального середовища. На рисунку 3.5 представлений інтерфейс, що описує структуру мережі Петрі.

```
export interface IPetriNet {  
  id: string;  
  elements: Array<IPetriNetElement>;  
  arcs: Array<IArc>;  
  ui: {  
    arcDrawer?: IArcDrawer;  
    selectedElementIdx: number;  
    selectedArcIdx: number;  
  };  
}
```

Рисунок 3.5 – Інтерфейс мережі Петрі

Поле *id* є унікальним ідентифікатором. Елементи мережі Петрі, такі як позиція і перехід зберігаються у полі *elements*. Поле *arcs* відповідає за зберігання масиву дуг. Під ключем *ui* відповідають параметри візуального відображення.

Елементом мережі Петрі в контексті веб-сервісу моделювання дискретно-подійних систем є позиція або перехід. Такий вибір обумовлений тим, що структура даних позицій і переходів відрізняється тільки параметрами, в той час як структура даних дуги має зовсім інакший вигляд. На рисунку 3.6 наведена структура об'єкту елемента мережі Петрі.

```
enum PetriNetElementType { Place, Transition }

export interface IPetriNetElement {
  id: string;
  type: PetriNetElementType;
  ui: {
    label: string;
    x: number;
    y: number;
  };
  params: ITransitionParams | IPlaceParams;
}
```

Рисунок 3.6 – Інтерфейс елемента мережі Петрі

Поле *id* є унікальним ідентифікатором. Тип елемента, що визначає чи є елемент позицією або переходом, задається цілим числом – 0 або 1 відповідно. Під ключем *ui* відповідають параметри візуального відображення. Поле *params* задає параметри елемента. У випадку якщо елемент є позицією, то це поле матиме тип *IPlaceParams*, що описує структуру параметрів позиції. У іншому випадку, якщо елемент є переходом, поле *params* матиме тип *ITransitionParams*, який описує структуру параметрів переходу.

На рисунку 3.8 представлений інтерфейс, що описує структуру об'єкту параметрів позиції.

```
interface IPlaceParams {
  mark: number;
  isPublic: boolean;
  isImportant: boolean;
}
```

Рисунок 3.8 – Інтерфейс параметрів позиції

Поле *mark* зберігає поточне значення маркерів у позиції. Поле *isPublic* приймає булеве значення та позначає позицію доступною для поєднання з іншими Петрі-об'єктами. Поле *isImportant* вказує на важливість позиції при виводі звіту користувачем.

На рисунку 3.9 представлений інтерфейс, що описує структуру об'єкту параметрів переходу

```
interface ITransitionParams {
  priority: number;
  param: number;
  paramDeviation?: number;
  distribution: string;
  isPublic: boolean;
  isImportant: boolean;
}
```

Рисунок 3.9 – Інтерфейс параметрів переходу

Поле *priority* визначає пріоритет спрацьовування переходу. Поля *param*, *paramDeviation*, *distribution* задають затримку переходу, як стохастичну величину з експоненційним або нормальним розподілом. Поля *isPublic* та *isImportant* виконують аналогічні функції що й для позиції.

Дуга описується окремим інтерфейсом *IArc*, що представлений на рисунку 3.10.

```
export interface IArc {
  source: string;
  target: string;
  params: IArcParams;
}
```

Рисунок 3.10 – Інтерфейс елементу дуги

Дуга містить *id* позиції та переходу через поля *source* і *target*. Також дуга містить параметри, які описує інтерфейс *IArcParams*, що представлений на рисунку 3.11.

```
interface IArcParams {
  quantity: number;
  isInfo: boolean;
}
```

Рисунок 3.11 – Інтерфейс параметрів дуги

В якості параметрів дуги задаються кількість зв'язків та наявність інформаційного зв'язку. За ці параметри відповідають поля *quantity* та *isInfo* відповідно.

Для роботи з потоком даних у застосуванні вирішено використати патерн Redux, що є патерном поширення даних у застосуванні та розроблений компанією Facebook. Згідно з Redux дані завжди рухаються в одному напрямку, як показано на рисунку 3.2.

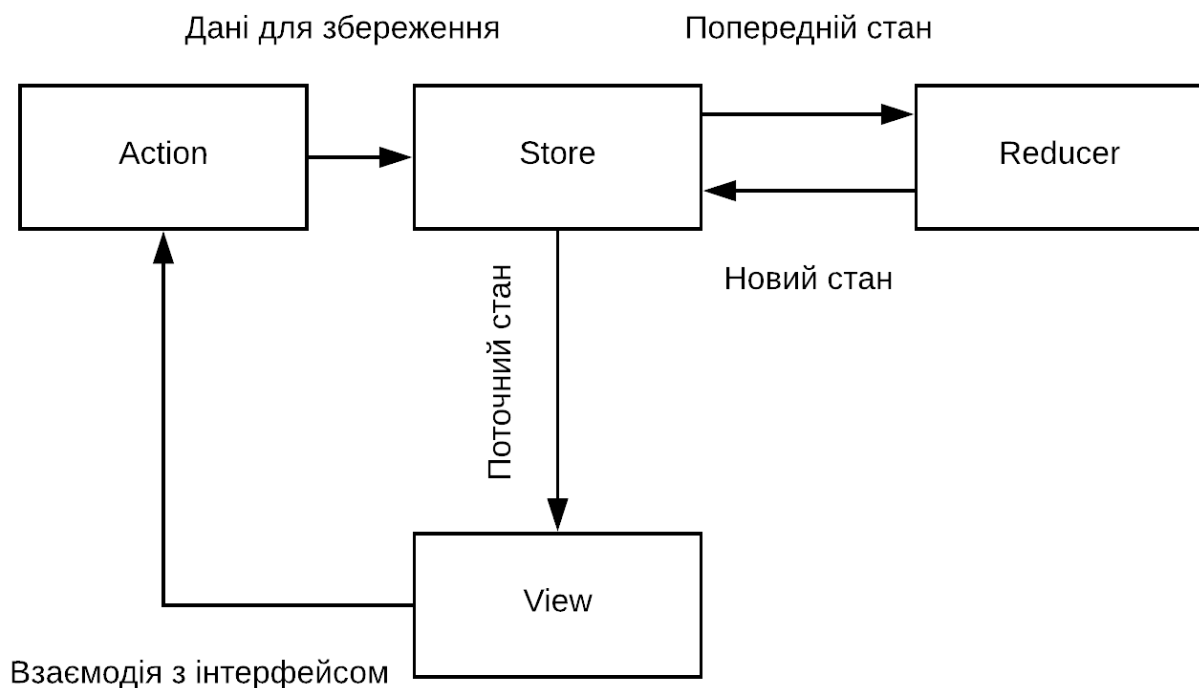


Рисунок 3.12 – Потік даних згідно патерну Redux

Оскільки вимоги до сучасних додатків JavaScript стають все більш складними, доводиться керувати великими кількостями різних станів застосування. Ці стани можуть включати в себе відповіді серверу, кешовані дані, а також дані, створені локально, але ще не збережені на сервері. Це також відноситься до стану інтерфейсу користувача, такими як активна сторінка, виділена вкладка, екран завантаження або номерування сторінок тощо.

Керувати станами, що постійно змінюються викликає складності. Якщо модель може оновити іншу модель, то представлення може оновити модель, яка оновлює іншу модель, а це, в свою чергу, може викликати оновлення іншого представлення. В певний момент програміст втрачає контроль та не знає, що відбувається у

застосуванні. Коли система стає непрозорою і недетермінованою, важко виявити помилку або додати нову функціональність.

Вимоги до розробки інтерфейсу користувача постійно зростають та стають загальними у розробці інтерфейсу продукту. Розробники повинні працювати з автоматичними оновленнями, відображенням на стороні сервера, отримання даних перед виконанням переходів на інші сторінки тощо.

Ця складність виникає через те, що тут змішуються дві концепції, які є складними для розуміння: зміни (мутація) та асинхронність. Обидві концепції можуть бути прекрасними окремо, але використання їх разом може створювати хаос. Такі бібліотеки як React, намагаються вирішити цю проблему на рівні представлення, видаляючи асинхронність і пряме маніпулювання DOM. Однак, React залишає керування станом даних на роботу. Тому розробниками React створений патерн Redux.

Разом з Flux, CQRS та Event Sourcing, Redux намагається зробити прогнозування мутацій стану шляхом накладання певних обмежень на те, як і коли можуть статися оновлення. Ці обмеження відображені у трьох принципах Redux:

- єдине сховище даних;
- поточний стан доступний тільки для читання;
- мутації стану представлені простими функціями, що повертають новий стан;

Перший принцип говорить про те, що стан усього додатка повинен бути збережений в дереві об'єктів в одному сховищі. Це облегшує створення універсальних застосувань. Стан на сервері може бути серіалізовано і відправлено на клієнт без додаткових зусиль. Це спрощує відладку застосувань, коли маємо справу з єдиним деревом стану. Також є можливість зберегти стан застосування для прискорення процесу розробки.

У другому принципі говориться про те, що єдиний спосіб змінити стан – це застосувати дію – об'єкт, який описує, що відбувається. Це гарантує, що представлення або функції, що реагують на події, ніколи не змінюють стан безпосередньо. Поки всі зміни централізовані і застосовуються послідовно в строгому порядку, немає необхідності стежити за "прегонами станів". Дії - це прості об'єкти,

тому вони можуть бути серіалізовані, збережені і потім відтворені для відладки або тестування. На рисунку 3.4 представлені дії, які відправляються користувачем під час роботи з елементами мережі Петрі, а саме: обрати елемент, додати елемент, оновити елемент.

```

export interface ISelectElement {
    type: constants.SELECT_ELEMENT;
    index: number;
}

export interface IReleaseElement {
    type: constants.RELEASE_ELEMENT;
}

export interface IAddElement {
    type: constants.ADD_ELEMENT;
    element: IPetriNetElement;
}

export interface IUpdateElement {
    type: constants.UPDATE_ELEMENT;
    element: IPetriNetElement;
}

```

Рисунок 3.13 – Приклад опису дій

Для визначення того, як дерево стану буде трансформовано діями, введено поняття ред'юсера. Ред'юсери – це прості функції, які беруть попередній стан та дію і повертають новий стан. Завдяки тому, що ред'юсери є функціями, з'являється можливість контролювати порядок, в якому вони викликаються, відправляти додаткові дані. На рисунку 3.5 наведений фрагмент функції-ред'юсера, що обробляє дії користувача.

```

function pNetReducer(state: IPetriNet = INITIAL_STATE,
  switch (action.type) {
    case SELECT_ELEMENT:
      return {
        ...state,
        ui: {
          ...state.ui,
          selectedElementIdx: (<actions.ISelectElement
          selectedArcIdx: -1,
        },
      };
    case RELEASE_ELEMENT:
      return {
        ...state,
        ui: {
          ...state.ui,
          selectedElementIdx: -1,
          selectedArcIdx: -1,
        },
      };
  }
}

```

Рисунок 3.14 – Ред’юсер, що обробляє дії користувача

Ред’юсер *pNetReducer* приймає дію користувача та в залежності від типу дії вирішує яким чином модифікувати поточний стан застосування. Наприклад, якщо користувач зробив клік по елементу, створюється дія *SELECT_ELEMENT*, яка передається аргументом в ред’юсер *pNetReducer*. Оскільки дія містить індекс елементу, по якому зроблено клік, ред’юсер повертає новий стан та зберігає цей індекс в полі *selectedElementIdx*.

Таким чином, за допомогою цих трьох принципів досягнута максимальна стабільність та прозорість програмного коду розробленого клієнтського застосування. Такий підхід дає великі можливості для подальшого розширення функціоналу застосування без витрачання зусиль на синхронізацію дій користувача.

3.4 Діаграма класів

Діаграма класів наведена у графічному матеріалі, де показані основні класи візуального середовища та їх методи. Опис класів та їх методів:

Клас `petriUILabel` відповідає за створення та відображення текстових полів.

- поле `text` – текст;
- метод `onKeyDown` – подія, що відбувається при натисканні клавіши на клавіатурі;
- метод `_numberKey` – перевіряє чи є заданий символ цифрою;

Клас `petriUIPosition` відповідає за графічне представлення елемента «Позиція» мережі Петрі та має наступні поля та методи:

- поле `name` – назва позиції;
- поле `mark` – кількість маркерів;
- поле `num` – ідентифікатор;
- поле `arcsIn` – список вхідних дуг;
- поле `arcsOut` – список вихідних дуг;
- метод `focus` – викликається при подвійному натисканні лівою кнопкою миші на об'єкт позиції, активує текстове поле для введення кількості маркерів;
- метод `updateArcs` – викликається при зміні положення об'єкту на графічній канві, перераховує напрямки дуг;
- метод `getPoint` – розраховує точку, до якої прив'язана дуга;
- метод `isPointNear` – за заданими координатами визначає чи знаходиться точка поблизу об'єкту;
- метод `drawBorders` – малює рамки об'єкту;
- метод `to_json` – повертає хеш, що містить основні параметри об'єкту;

Клас `petriUITransition` відповідає за графічне представлення елемента «Перехід» мережі Петрі та має наступні поля і методи:

- поле `name` – назва переходу;
- поле `num` – ідентифікатор;

- поле `distribution` – тип розподілу;
- поле `param` – параметр розподілу;
- поле `deviation` – відхилення;
- поле `arcsIn` – список вхідних дуг;
- поле `arcsOut` – список вихідних дуг;
- метод `focus` – викликається при подвійному натисканні лівою кнопкою миші на об'єкт переходу, активує текстове поле для введення параметру розподілу;
- метод `updateArcs` – викликається при зміні положення об'єкту на графічній канві, перераховує напрямки дуг;
- метод `getPoint` – розраховує точку, до якої прив'язана дуга;
- метод `to_json` – повертає хеш, що містить основні параметри об'єкту;

Клас `petriUIArc` відповідає за графічне відображення елемента «Дуга» та має наступні поля і методи:

- поле `num` – ідентифікатор;
- поле `text` – кількість зв'язків;
- поле `startObj` – об'єкт, що є початком дуги;
- поле `endObj` – об'єкт, що являє собою кінець дуги;
- поле `isFirst` – визначає чи є дуга першою;
- поле `isSecond` – визначає чи є дуга останньою;
- поле `isInfo` – визначає чи є дуга інформаційним зв'язком;
- поле `arrow` – представляє графічний об'єкт – трикутник, що є кінцем дуги;
- метод `setStartObj` – задає вихідний об'єкт;
- метод `setEndObj` – задає вхідний об'єкт;
- метод `_render` – відповідає за відображення дуги на графічній канві;
- метод `onArrowMoving` – подія, яка відбувається при переміщенні стрілки дуги;
- метод `to_json` – повертає хеш, що містить основні параметри об'єкту;

Клас `petriUI` відповідає за управління параметрами візуального середовища та містить такі методи:

- `setArcMode` – встановлює режим «Дуга», блокує візуальне середовище від вибору та створення нових позицій та переходів;
- `showGrid` – відображає/приховує сітку;

3.5 Специфікація функцій

В таблиці 3.1 описані методи класу, що відповідає за виведення текстових полів.

Таблиця 3.1 – Опис методів класу `petriUILabel`

Метод	Опис методу	Параметри	Опис параметрів
<code>onKeyDown</code>	Подія, що відбувається при натисканні клавіші на клавіатурі	<code>e</code>	Об'єкт події
<code>_numberKey</code>	Перевіряє чи є заданий символ цифрою	<code>_char</code>	Код символу

В таблиці 3.2 описані методи класу, що відповідає за графічне відображення елемента «Позиція» мережі Петрі.

Таблиця 3.2 – опис методів класу `petriUIPosition`

Метод	Опис методу	Параметри	Опис параметрів
<code>focus</code>	Викликається при подвійному натисканні лівою кнопкою миші на об'єкт позиції, активує текстове поле для введення кількості маркерів	-	-

updateArcs	Викликається при зміні положення об'єкту на графічній канві, перераховує напрямки дуг	endPt, isFirst, isSecond	Кінцева точка, дуга є першою, дуга є другою
getPoint	Розраховує точку, до якої прив'язана дуга	-	-
isPointNear	За заданими координатами визначає чи знаходиться точка поблизу об'єкту	point	Точка, що перевіряється
drawBorders	Малює рамки об'єкту	-	-
to_json	Повертає основні параметри об'єкту	-	-

В таблиці 3.3 описані методи класу, що відповідає за графічне відображення елемента «Перехід» мережі Петрі.

Таблиця 3.3 – опис методів класу petriUITransition

Метод	Опис методу	Параметри	Опис параметрів
focus	Викликається при подвійному натисканні лівою кнопкою миші на об'єкт позиції, активує текстове поле для введення параметру розподілу	-	-
updateArcs	Викликається при зміні положення об'єкту на графічній канві,	endPt, isFirst, isSecond	Кінцева точка, дуга є першою, дуга є другою

	перераховує напрямок дуг		
getPoint	Розраховує точку, до якої прив'язана дуга	-	-
isPointNear	За заданими координатами визначає чи знаходиться точка поблизу об'єкту	point	Точка, що перевіряється
to_json	Повертає хеш, що містить основні параметри об'єкту	-	-

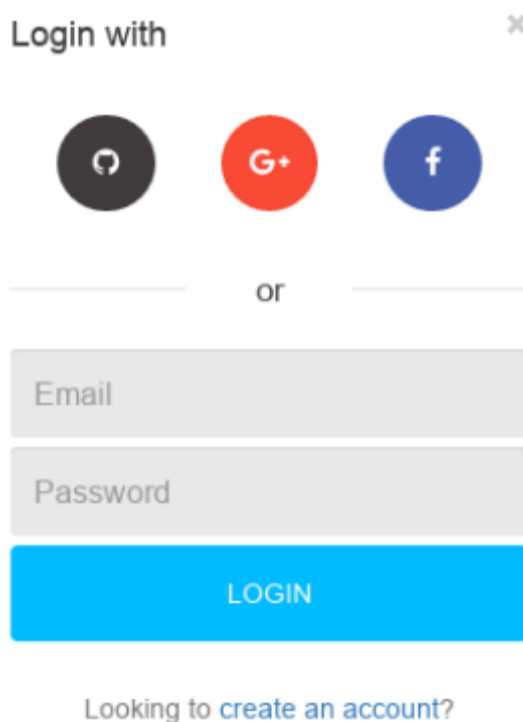
В таблиці 3.4 описані методи класу, що відповідає за графічне відображення елемента «Дуга» мережі Петрі

Таблиця 3.4 – опис методів класу petriUIArc

Метод	Опис методу	Параметри	Опис параметрів
setStartObj	Задає вихідний об'єкт	obj	Об'єкт «позиія» або «перехід»
setEndObj	Задає вхідний об'єкт	obj	Об'єкт «позиія» або «перехід»
onArrowMoving	Подія, яка відбувається при переміщенні стрілки дуги	-	-
_render	відповідає за відображення дуги на графічній канві	ctx	Контекст HTML 5 Canvas
to_json	Повертає хеш, що містить основні параметри об'єкту	-	-

3.6 Керівництво користувача

В першу чергу потрібно перейти на сайт веб-ресурсу і авторизоватися. На рисунку 3.15 представлена форма для входу користувача:



The image shows a login interface. At the top, it says "Login with" followed by a close icon (x). Below this are three circular icons: a black circle with a white 'G', a red circle with a white 'G+', and a blue circle with a white 'f'. Below these icons is the word "or" centered between two horizontal lines. Underneath are two input fields: "Email" and "Password". Below the input fields is a large blue button labeled "LOGIN". At the bottom, there is a link that says "Looking to create an account?"

Рисунок 3.15 – Форма авторизації користувачів

Після авторизації користувач потрапляє на сторінку створення нової мережі Петрі. На рисунку 3.16 зображений початковий стан для нового користувача з вікном створення нової мережі Петрі.

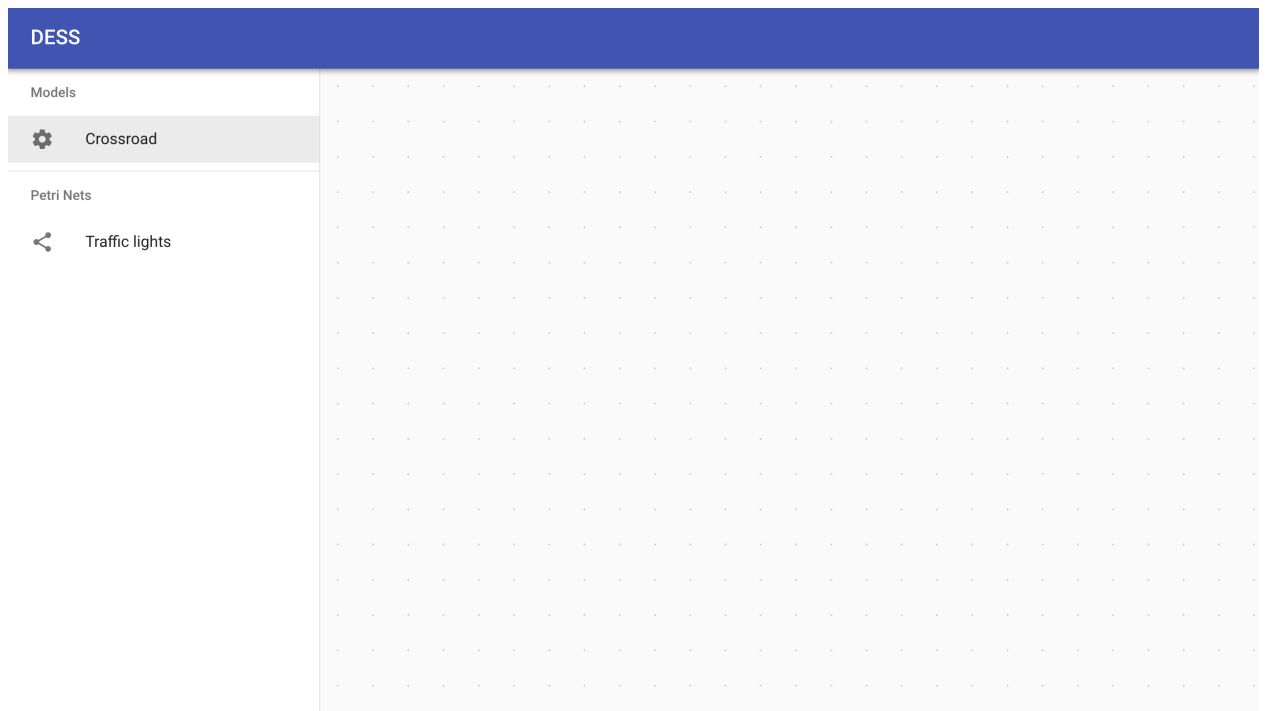


Рисунок 3.16 – Сторінка нового документа

На сторінці документа можна виділити ключові елементи інтерфейсу такі як панель документів користувача (Рисунок 3.17), панель інструментів (Рисунок 3.18), панель звітів процесу імітації (Рисунок 3.19).

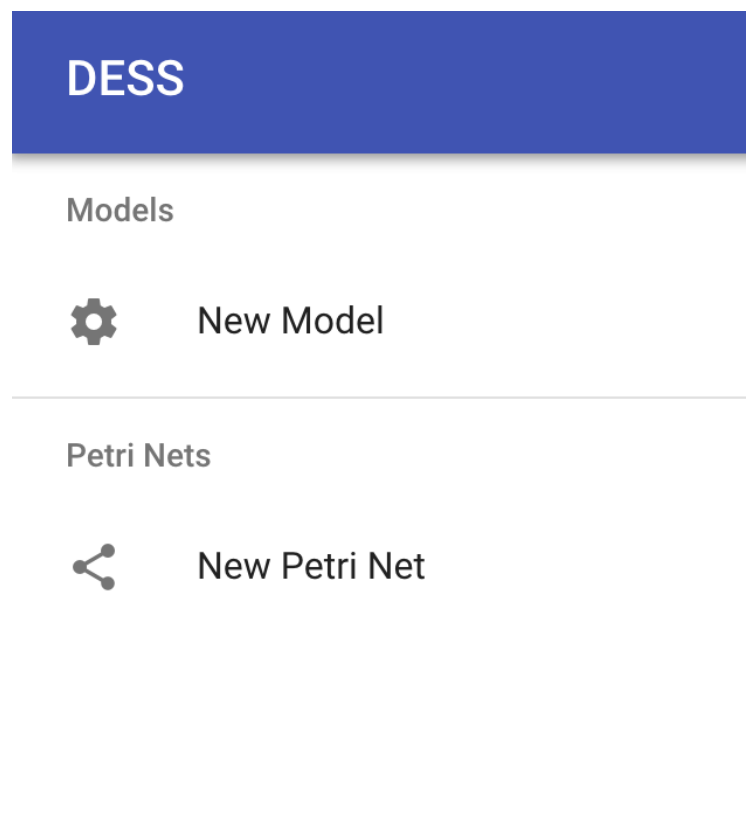


Рисунок 3.17 – Панель документів користувача



Current Time: 0

Рисунок 3.18 – Панель інструментів

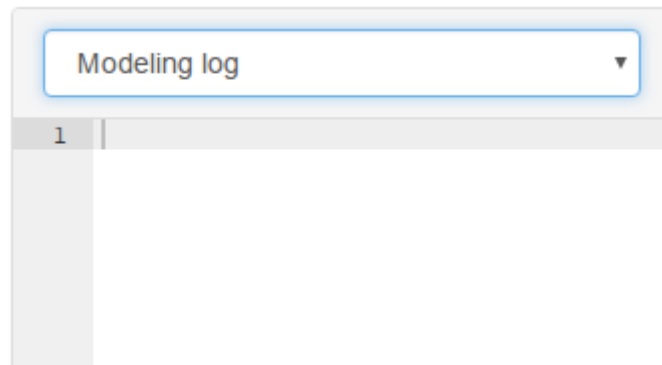


Рисунок 3.19 – Панель звітів процесу імітації

На панелі документів розміщуються створені Петрі-об'єктні моделі та мережі Петрі користувача. У випадку, коли користувач вперше авторизувався в системі, створюється порожній документ для створення нової мережі Петрі, що готовий для редагування.

Панель інструментів містить в собі основні компоненти роботи з візуальним середовищем такі як:

- створення нової мережі Петрі;
- збереження мережі Петрі;
- створення елементів мережі Петрі;
- створення нової Петрі-об'єктної моделі;
- збереження Петрі-об'єктної моделі;
- створення нового Петрі-об'єкту;
- запуск процесу імітації;
- поле для вводу часу імітації;
- поле для виведення поточного часу імітації;

Панель звітів процесу імітації дає змогу користувачеві переглянути протокол подій, що відбувалися в процесі імітації та статистику по кожному елементу мережі Петрі.

Для побудови моделі використовується робоча панель, що розміщується в центрі екрану. Використовуючи панель інструментів, користувач будує мережу Петрі шляхом вибору одного з режимів створення елементів мережі Петрі та кліком лівої кнопки миші у відповідному місці робочій панелі. Створення елементів мережі Петрі таких як: позиція, перехід, дуга показані на рисунках 3.20 – 3.22 відповідно.

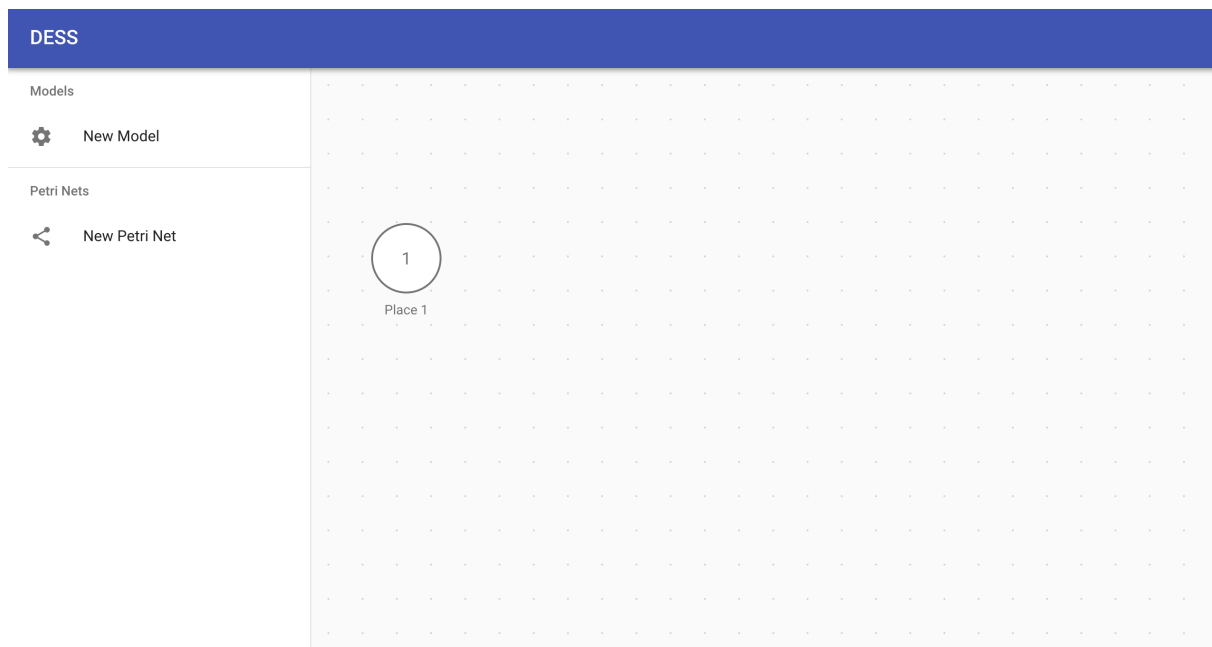


Рисунок 3.20 – Створення елемента «Позиція»

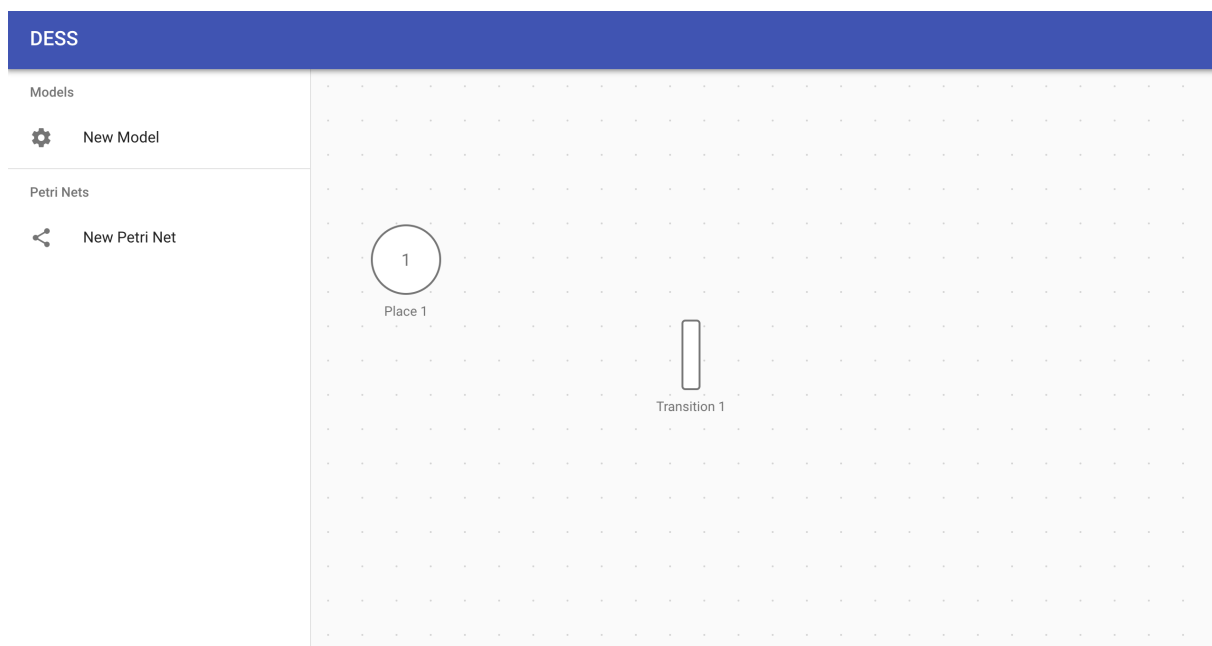


Рисунок 3.21 – Створення елемента «Перехід»

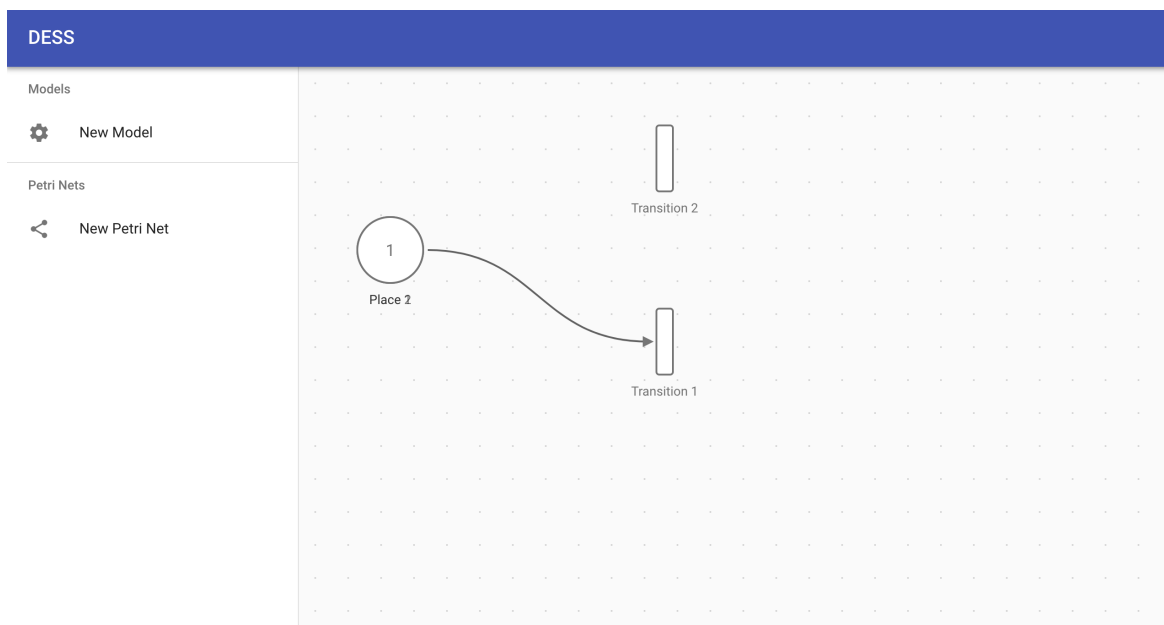


Рисунок 3.22 – Створення елементу «Дуга»

В процесі створення елементів мережі Петрі користувач має можливість налаштувати параметри відповідного елементу використовуючи спеціально призначену панель параметрів або ж подвійним кліком безпосередньо на сам графічний об'єкт шляхом введення в текстове поле відповідних значень.

Налаштування параметрів елементів мережі Петрі відбувається у спеціальному вікні, що з'являється під час вибору того чи іншого елементу шляхом кліку на його візуальне відображення.

Для елементу мережі Петрі «позиція» доступні такі налаштування параметрів:

- назва позиції;
- кількість маркерів;

Наступні параметри можуть бути вказані для переходів мережі Петрі:

- назва переходу;
- тип розподілу;
- параметр розподілу;
- відхилення;

Для дуг доступні такі налаштування параметрів:

- кількість зв'язків;
- є інформаційною;

Ілюстрація даного процесу наведена на рисунках 3.23 – 3.24:

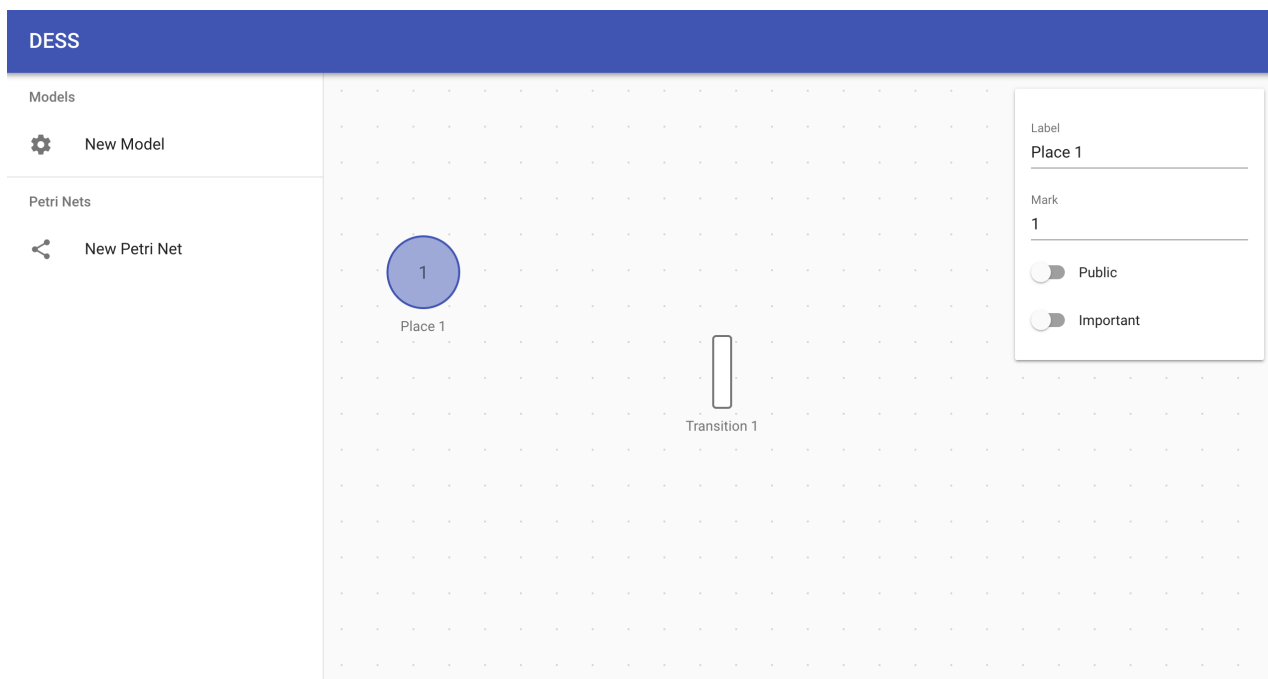


Рисунок 3.23 – Налаштування параметрів елементу «позиція» використовуючи візуальний компонент «панель параметрів»

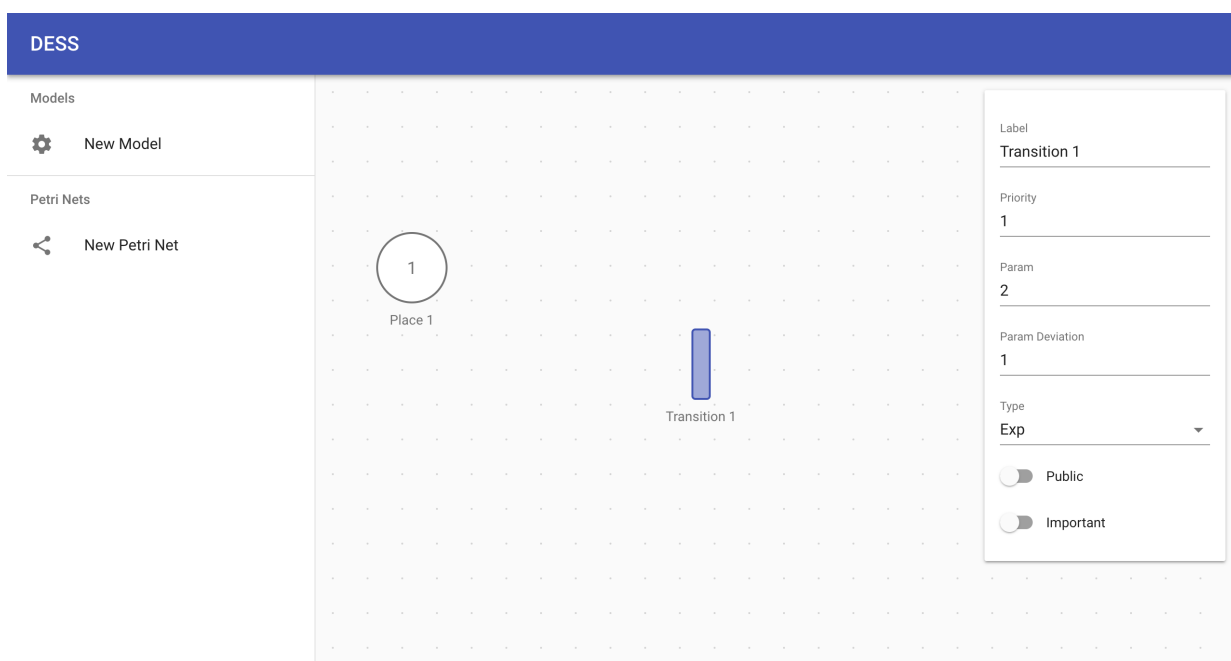


Рисунок 3.24 – Налаштування параметрів елементу «перехід» використовуючи візуальний компонент «панель параметрів»

Імітаційна модель може мати багато параметрів одночасно. Чим більше переходів, позицій, дуг та/або Петрі-об'єктів у моделі, тим більше може бути створено параметрів. При цьому застосування не дозволить користувачу запустити імітацію для мережі Петрі або Петрі-об'єктної моделі, які містять хоча б один параметр, для

якого відсутнє значення. Користувач у такому випадку отримає відповідне повідомлення, приклад якого наведено на рисунку 3.24.

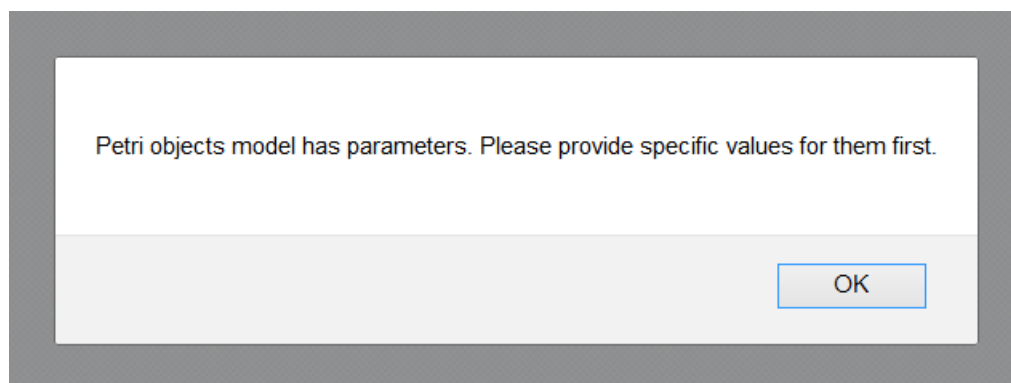


Рисунок 3.24 – Приклад повідомлення про відсутність значень параметрів

3.7 Висновки до розділу

Архітектура та програмне забезпечення веб-застосунку реалізує Петрі-об'єктний підхід до моделювання дискретно-подійних систем. Розроблений веб-застосунок призначений для забезпечення крос-платформної, швидкої та безпечної розробки імітаційних моделей. Технологія Петрі-об'єктного моделювання пропонує зручне конструювання моделей та швидкий алгоритм моделювання, тому його легко реалізувати для систем з великою кількістю елементів. Багатоканальні переходи, дозволяють представлення майже будь-якої елементарної події з будь-яким детермінованим або стохастичним часом затримки, з різними способами вирішення конфлікту з іншими подіями. Налаштування дуги включає параметр, який дозволяє враховувати, але не змінювати стан вхідних позицій. Це забезпечує потужні та універсальні інструменти для представлення моделі.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Порядок проведення досліджень

Досліджувана система представляє собою дорожній рух на перехресті, керований світлофорами, що зображений на рисунку 4.1. Перехрестя є простим перетином двох доріг.

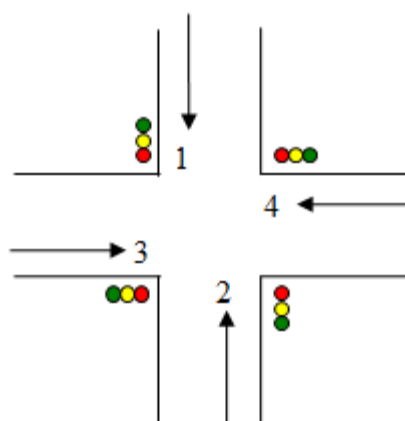


Рисунок 4.1 – Схема дорожнього руху на перехресті

Управління транспортним рухом здійснюється світлофорами так, що протягом певного часу горить зелене світло в першому та другому напрямках руху, а в третьому та четвертому напрямках горить червоне світло. Потім горить жовте світло в усіх напрямках протягом часу, що дозволяє автомобілям, які виїхали на перехрестях, залишити його до початку руху автомобілів з іншого напрямку. Далі вмикається зелене світло в третьому та четвертому напрямках руху, а в першому та другому напрямках горить червоне світло. Потім знову вмикається жовте світло в усіх напрямках і так далі. Тривалості горіння зеленого та червоного світла задаються так, як у таблиці 4.1.

Таблиця 4.1 – Параметри управління світлофорного об'єкта

Напрямок руху	Фаза світлофора			
	I	II	III	IV

	світло	час	світло	час	світло	час	світло	час
1	зелений	20	жовтий	10	червоний	30	жовтий	10
2	зелений	20	жовтий	10	червоний	30	жовтий	10
3	червоний	20	жовтий	10	зелений	30	жовтий	10
4	червоний	20	жовтий	10	зелений	30	жовтий	10

Інтенсивність руху автомобілів в усіх напрямках відома і наведена у таблиці 4.2.

Таблиця 4.2 – Інтенсивність руху автомобілів у напрямках

Напрямок руху	Інтенсивність руху, $\lambda=1/a$, 1/с
1	1/15
2	1/9
3	1/20
4	1/35

Метою моделювання є визначення параметрів управління, при яких максимум середньої кількості автомобілів, що очікують переїзду в різних напрямках, досягає свого найменше значення:

$$L = \min\{L_1, L_2, L_3, L_4\} \rightarrow \min.$$

Виділимо події, які відбуваються при переїзді автомобілів перехрестя в i -ому напрямку:

- надходження автомобіля в i -ому напрямку;
- переїзд автомобілем перехрестя в i -ому напрямку.

Подія «переїзд автомобілем перехрестя в i -ому напрямку» відбувається за умови, що надійшов автомобіль в i -ому напрямку та є зелене світло в i -ому напрямку.

Введемо позиції «є зелене світло в 1-ому та 2-ому напрямках» та «є зелене світло в 3-ому та 4-ому напрямках», за наявності маркеру в яких буде прийматись рішення про переїзд автомобілем перехрестя. В результаті функціонування підсистеми управління маркер в позиції «є зелене світло в 1-ому та 2-ому напрямках» з'являється або зникає.

Маркером у позиції «є зелене світло» повинні користуватись одночасно декілька автомобілів (в декількох рядах і в декількох напрямках), але вона не являється ресурсом, який використовують одночасно, в ній тільки міститься інформація типу «дозвіл».

Автомобілі не конфліктують при захопленні зеленого світла, тому моделювання світлофорного об'єкта звичайною мережею Петрі призводить до зайвих ускладнень, яких не має в реальній системі.

Введемо інформаційний зв'язок між позицією і переходом як такий, що наявність маркеру в позиції перевіряється при здійсненні перевірки умови запуску переходу, але при здійсненні запуску переходу маркер з позиції не віднімається. Тобто звичайний матеріальний зв'язок між позицією та переходом означає пересування маркерів з позиції в перехід при виконанні умови запуску переходу. А інформаційний зв'язок між позицією та переходом означає, що пересування маркерів при запуску переходу не відбувається. Домовимось позначати інформаційний зв'язок пунктирною лінією.

На рисунку 4.2 представлено підсистему руху автомобілів через перехрестя мережею Петрі з інформаційними зв'язками. В такому представленні моделі конфлікт при сумісному використанні маркеру в позиції «є зелене світло» не виникає. По-друге, не виникає необхідності вводити додаткові переходи чи додаткові позиції, що ускладнюють модель. Функціонування такої моделі якнайбільш наближено до функціонування реальної системи.

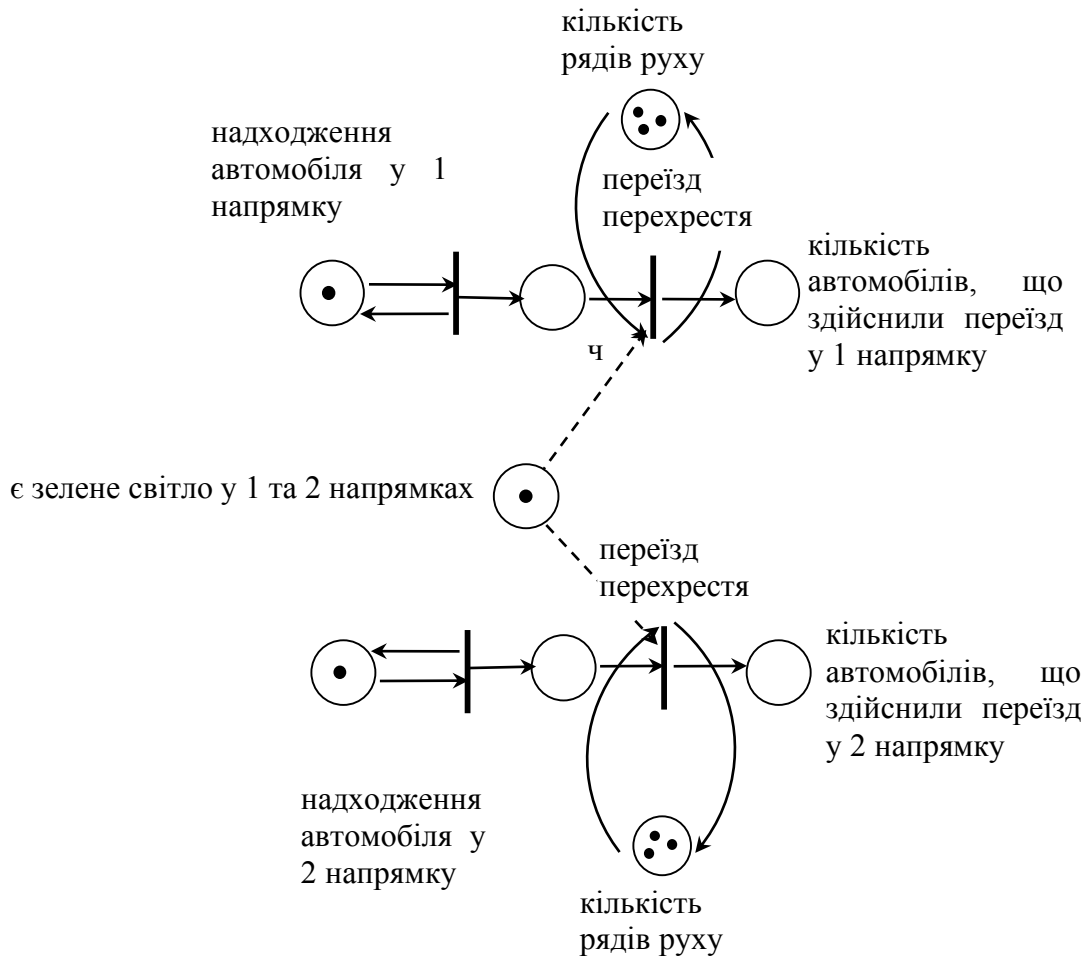


Рисунок 4.2 - Підсистема руху автомобілів в i -ому напрямку, представлена мережею Петрі з інформаційними зв'язками

Часові затримки, які передбачаються для переходів, указані на рисунку ($t_1 = a \cdot \ln \zeta$, де a – середнє значення інтервалу надходження автомобілів, ζ - рівномірно розподілена в інтервалі $(0;1)$ випадкова величина).

Виділимо події, які відбуваються в підсистемі управління:

- горить зелене світло в 1-ому та 2-ому напрямках;
- горить жовте світло в усіх напрямках;
- горить зелене світло в 3-ому та 4-ому напрямках;
- горить жовте світло в усіх напрямках.

Подія «горить зелене світло в 1-ому та 2-ому напрямках» відбувається за умови, що вичерпаний час горіння червоного світла в усіх напрямках і настав час горіння зеленого світла в 1-ому та 2-ому напрямках.

Події «горить зелене світло в першому напрямку», «горить жовте світло в обох напрямках», «горить зелене світло в другому напрямку», «горить жовте світло в обох напрямках» змінюють одна одну утворюючи коло подій, що відбуваються в підсистемі управління.

Умова «є зелене світло в першому напрямку» виникає разом з умовою «вмикається зелене світло на першому напрямку» і зникає разом з початком горіння червоного світла в обох напрямках. Так само умова «є зелене світло в другому напрямку». Таким чином, підсистема управління дорожнім рухом на перехресті представляється мережею Петрі як на рисунку 4.3.

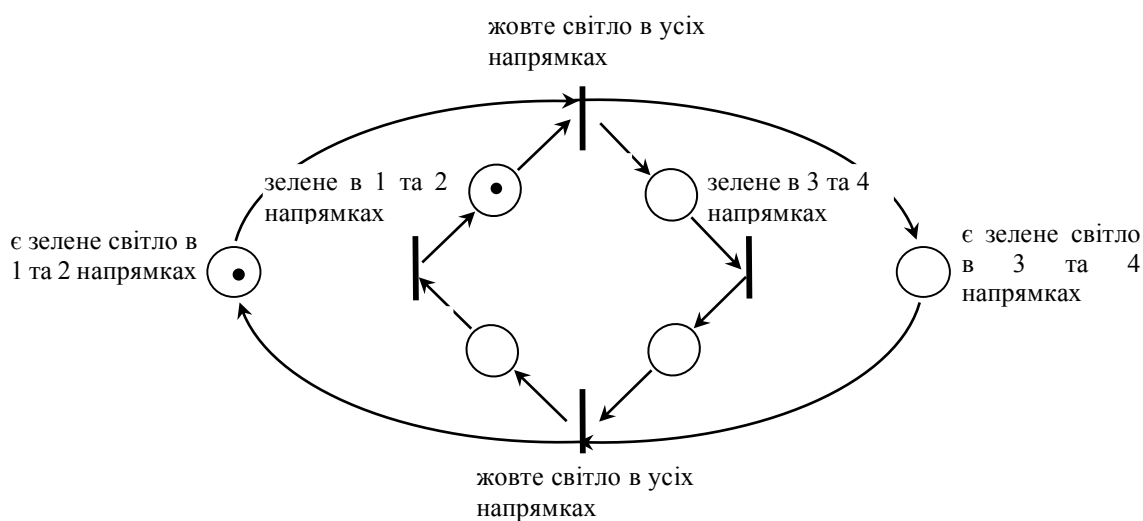


Рисунок 4.3 – Підсистема управління руху автомобілів на перехресті, представлена мережею Петрі

З'єднуючи підсистеми руху в кожному напрямку та підсистему управління рухом, отримуємо модель, що представлена на рисунку 4.4.

В результаті функціонування моделі спостерігається кількість автомобілів, що очікують переїзду, у позиціях «черга», розташованих між переходами «надходження» та «переїзд перехрестя» кожного напрямку руху. Середня кількість машин, що

очікують переїзду в i -ому напрямку, розраховується як середнє динамічної випадкової величини за формулою:

$$\text{середня кількість автомобілів у черзі} = \frac{\sum_{k=0}^n M(\text{черга})_k \cdot \Delta t_k}{T_{\text{mod}}}$$

Середній час очікування автомобілів у черзі розраховується як сумарне очікування автомобілів у черзі, розділене на кількість автомобілів, що здійснили переїзд:

$$\text{середній час очікування автомобілів у черзі} = \frac{\sum_{k=0}^n M(\text{черга})_k \cdot \Delta t_k}{M(\text{кільк_авт_зд_переїзд})}$$

З'єднуючи підсистеми руху в кожному напрямку та підсистему управління рухом, отримуємо модель, що представлена на рисунку 4.13.

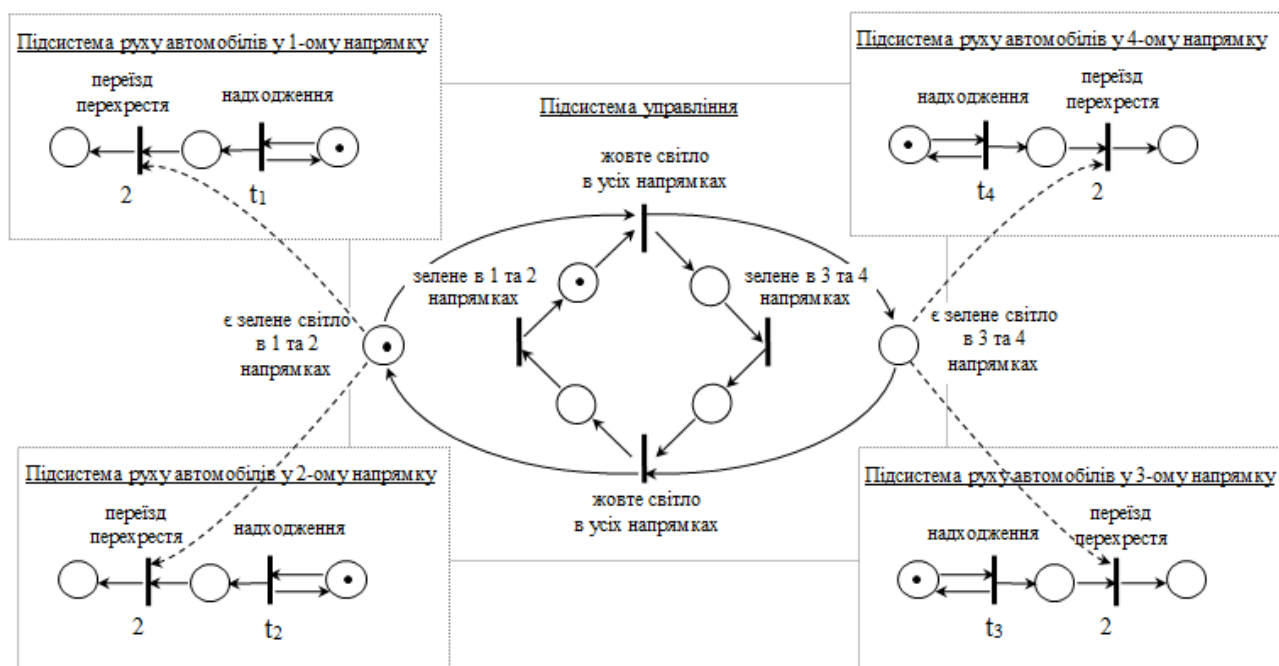


Рисунок 4.4 – Модель дорожнього руху через перехрестя, кероване світлофорами.

Отже, використаємо розроблене візуальне середовище моделювання дискретно-подійних систем для побудови моделі дорожнього руху через перехрестя, кероване світлофорами. На рисунку 4.5 представлений процес побудови моделі.

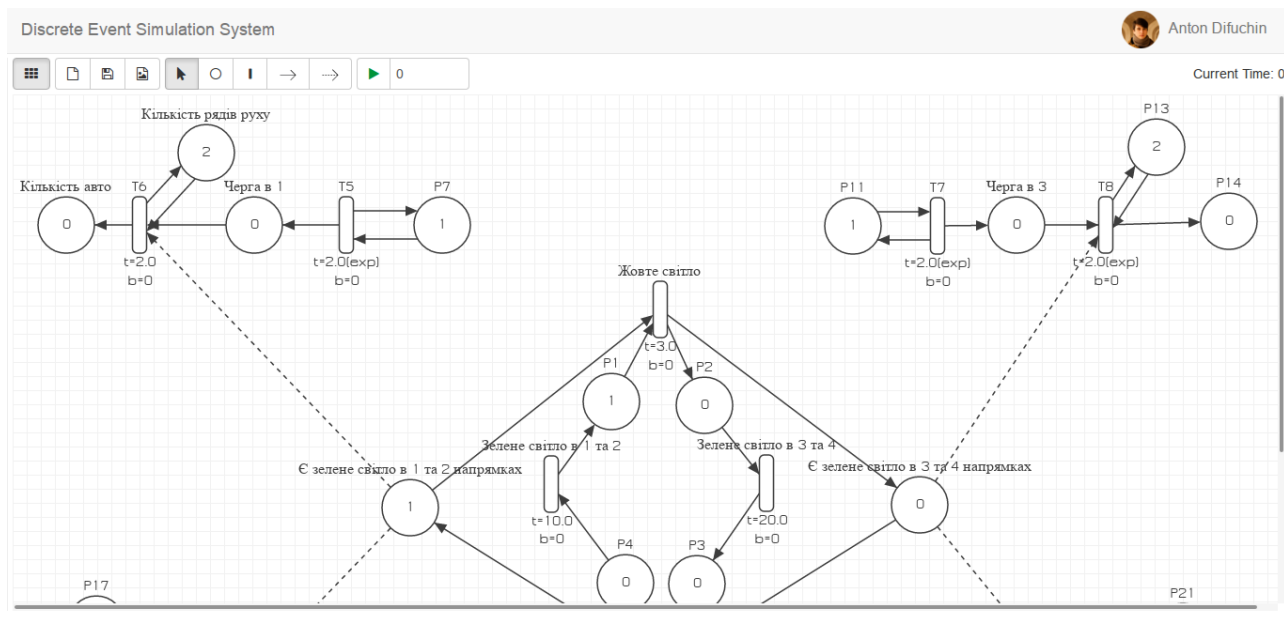


Рисунок 4.5 – Процес побудови мережі Петрі у візуальному середовищі

Оскільки розроблена модель досить громіздка для відображення на екрані, тому на рисунку 4.6 представлена повна розроблена модель у вигляді зображення, що генерується кнопкою на панелі інструментів «Convert to image».

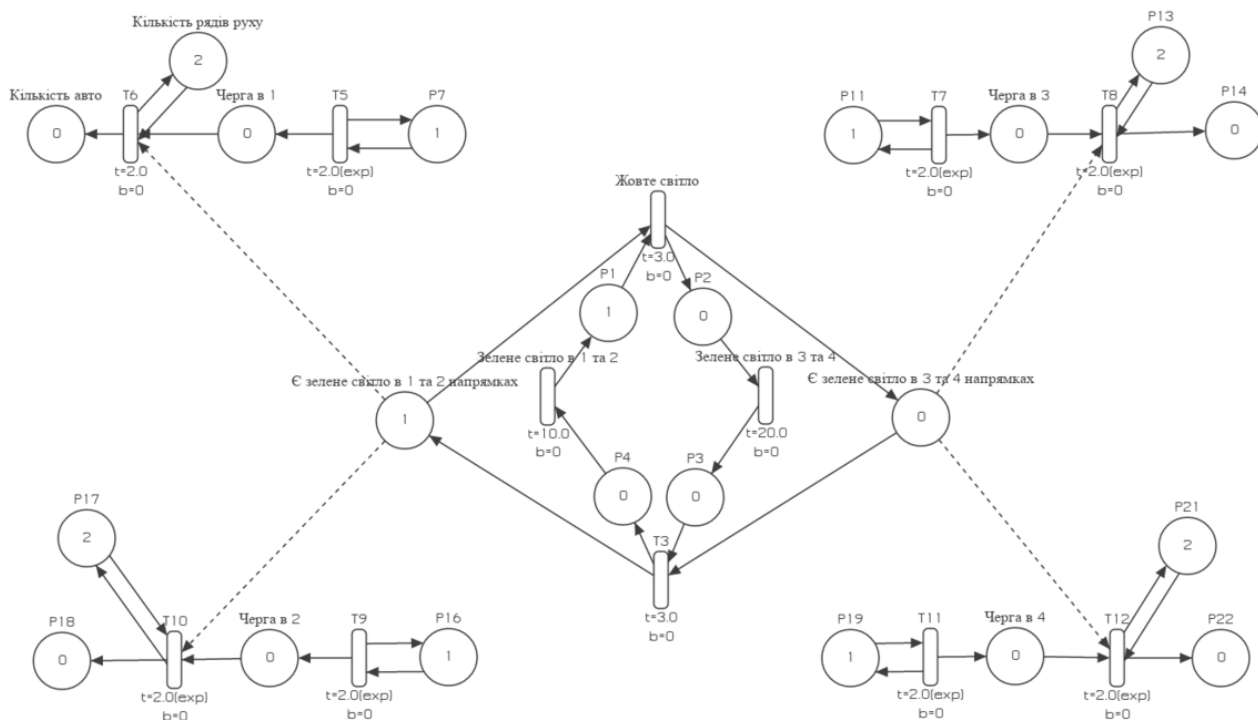


Рисунок 4.6 – Зображення побудованої моделі.

Після побудови моделі можна розпочати процес імітації. Для цього необхідно ввести у відповідному полі бажаний час моделювання. Після натискання кнопки «Run» та введення часу моделювання побудована модель відправляється на сервер, де відбувається процес імітації. Після завершення процесу імітації, сервер відправляє клієнту змінене маркування мережі, досягнений час, протокол подій та статистику, як показано на рисунках 4.7 – 4.8

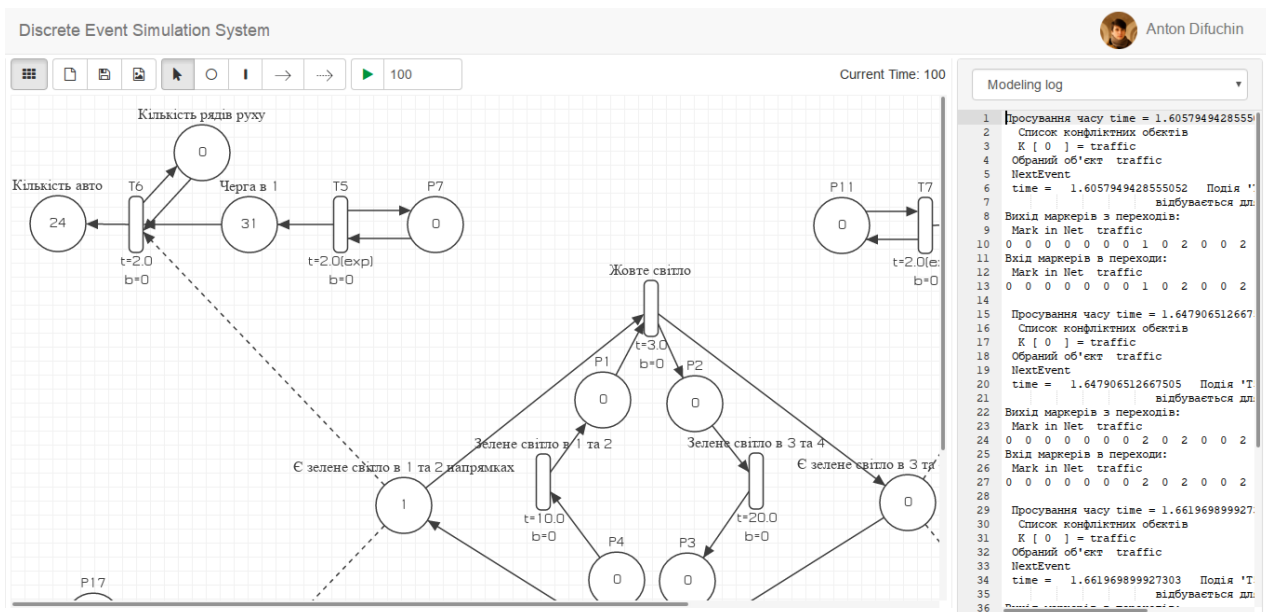


Рисунок 4.7 – Стан моделі після імітації з виведенням протоколу подій.

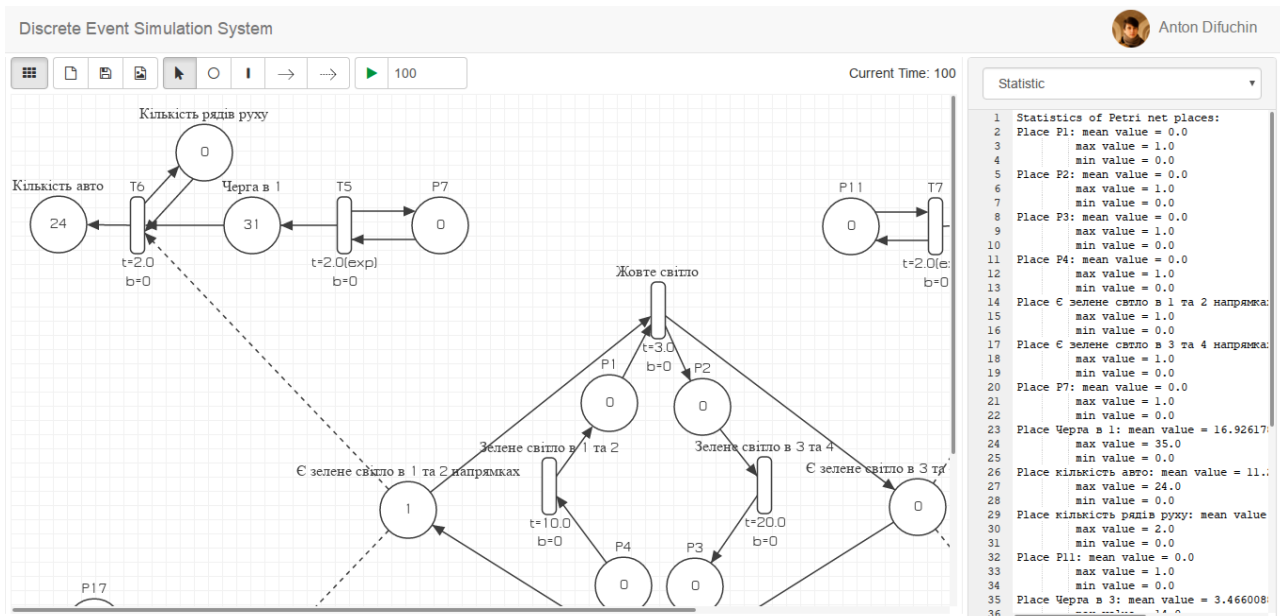


Рисунок 4.8 – Стан побудованої моделі після імітації з виведенням статистики.

4.1.1 Результати випробувань

У процесі проведення випробувань була перевірена функціональність веб-сервісу моделювання дискретно-подійних систем. У наступних таблицях наведено перелік випробувань основних функціональних можливостей (таблиці 4.3 – 4.7).

Таблиця 4.3 – Тестування реєстрації

Мета тесту :	Тестування вибору паролю під час реєстрації користувача
Початковий стан системи :	Відкрите меню реєстрації
Вхідні дані :	Логін Пароль Підтвердження паролю
Схема проведення тесту	Введення імені, паролю, та невірною підтвердження паролю
Очікуваний результат	Реакція додатку у вигляді спливаючих вікон
Стан системи після проведення тесту :	Відкритий новий порожній документ, з'явилося фото та ім'я користувача у верхньому меню навігації.

Таблиця 4.4 - Тестування авторизації

Мета тесту :	Перевірка системи авторизації
Початковий стан системи :	Відкрите меню авторизації
Вхідні дані :	Логін Пароль
Схема проведення тесту	Введення існуючого імені та паролю
Очікуваний результат	Авторизація у системі

Стан системи після проведення тесту :	З'явився список з побудованими моделями користувача, а також фото та ім'я користувача у верхньому меню навігації.
---------------------------------------	---

Таблиця 4.6 – Тестування створення елементів мережі Петрі

Мета тесту :	Перевірити створення елементів мережі Петрі
Початковий стан системи :	Відкритий документ готовий до редагування
Вхідні дані :	Координати елемента
Схема проведення тесту	Вибір мишкою на панелі інструментів відповідного елемента та клік по робочій панелі
Очікуваний результат	З'явлення на робочій панелі елемента мережі Петрі з параметрами за замовчуванням
Стан системи після проведення тесту :	На робочій панелі з'явився елемент мережі Петрі з параметрами за замовчуванням

Таблиця 4.5 – Тестування зміни параметрів елементів мережі Петрі

Мета тесту :	Перевірити зміну параметрів елементів мережі Петрі
Початковий стан системи :	На робочій поверхні є 1 або більше елементів
Вхідні дані :	Параметри елемента
Схема проведення тесту	Клік по елементу, введення відповідних параметрів на панелі параметрів елементів або подвійний клік по елементу та введення параметрів у текстових полях елемента
Очікуваний результат	Змінення параметрів мережі Петрі відповідно введеним даним
Стан системи після проведення тесту :	Параметри мережі змінулись

Таблиця 4.7 – Тестування процесу імітації

Мета тесту :	Перевірка процесу імітації
Початковий стан системи :	Побудована мережа Петрі
Вхідні дані :	Мережа Петрі
Схема проведення тесту	Запуск процесу імітації
Очікуваний результат	Змінене маркування мережі Петрі, змінення досягнутого часу, виведення протоколу подій та статистики
Стан системи після проведення тесту :	Маркування мережі змінилося, також змінився досягнений час, виведений протокол подій та статистика

4.2 Висновки до розділу

В даному розділі описано тестування веб-сервісу моделювання дискретно-подійних систем, перевірений функціонал на прикладі моделюванні дорожньо-транспортного руху, а також перевірена відповідність заявленим вимогам. В результаті встановлено, що система успішно пройшла всі тести і відповідає заявленим вимогам.

ЗАГАЛЬНІ ВИСНОВКИ

При виконанні магістерської дисертації було проаналізовано предметне середовище систем імітаційного моделювання. Проведено огляд існуючих рішень з моделювання дискретно-подійних систем. Розроблено веб-сервіс моделювання дискретно-подійних систем.

Сформульовано математичну задачу відповідності графічного об'єкту формалізованому опису стохатичної мережі Петрі.

Архітектура та програмне забезпечення веб-застосунку реалізує Петрі-об'єктний підхід до моделювання дискретно-подійних систем. Розроблене веб-застосування призначене для забезпечення крос-платформної та швидкої розробки імітаційних моделей.

Розроблена модель бази даних, яка дозволяє ефективно та надійно здійснювати доступ до даних про моделі, які створюють користувачі, зберігати профілі користувачів.

Наведене детальне керівництво користувача, в якому закладені основні принципи роботи з розробленим візуальним середовищем моделювання дискретно-подійних систем.

Тестування веб-сервісу відбувалось за допомогою прикладу про моделювання дорожньо-транспортного руху. За результатами тестування можна запевнитись, що візуальне середовище працює коректно з виведенням необхідних звітів та відображення зміни стану мережі Петрі в графічному представленні на робочій панелі. Також описана методика проведення випробувань, яка показує готовність введення візуального середовища моделювання дискретно-подійних систем в експлуатацію.

ПЕРЕЛІК ПОСИЛАНЬ

1. Томашевський В. М. Моделювання систем / В. М. Томашевський. – Київ : Видавнича група ВНУ, 2005. – 352 с.
2. Лоу А. Имитационное моделирование. Классика CS : Пер. с англ. / Аверилл М. Лоу, В. Дэвид Кельтон. – 3-е изд. – Киев : Издательская группа ВНУ, 2004. – 847 с.
3. Акопов А. С. Имитационное моделирование: учебник и практикум для академического бакалавриата / А. С. Акопов. – Москва : Издательство Юрайт, 2014. – 389 с.
4. Анищенко В. С. Знакомство с нелинейной динамикой: Лекции Соросовского профессора / В. С. Анищенко. – 3-е изд. – Москва : Издательская группа URSS, 2008. – 224 с.
5. Discrete-Event System Simulation / J. Banks, J. S. Carson, B. L. Nelson, D. M. Nicol. – 4th ed. – New Jersey : Prentice-Hall, 2005. – 528 p.
6. Стеценко І. В. Проектування графічного модуля програмного забезпечення Петрі-об'єктного моделювання систем / І. В. Стеценко, О. В. Василевська // Вісник Черкаського державного технологічного університету. – Черкаси : ЧДТУ, 2013. – № 2. – С. 13-18.
7. Dehouck R. Craft AI. The maturity of visual programming [Електронний ресурс] / Rémi Dehouck. – 2015. – Режим доступу до ресурсу: <http://www.craft.ai/blog/the-maturity-of-visual-programming/>.
8. Maloney J. The Scratch Programming Language and Environment / J. Maloney, M. Resnick, B. Silverman, E. Eastmond // ACM Transactions on Computing Education. – 2010. – Vol. 10, No. 4. – 15 p.
9. Beucher O. Introduction to MATLAB and SIMULINK: A Project Approach / Ottmar Beucher, Michael Weeks. – 3rd ed. – Hingham : Infinity Science Press LLC, 2007. – 386 p.

10. Colvin R. A semantics for Behavior Trees / Robert Colvin, Ian J. Hayes // ACCS Technical Report ACCS-TR-07-01, ARC Centre for Complex Systems. – 2007. – 20 p.
11. Garrido J. M. Object-Oriented Discrete-Event Simulation with Java: A Practical Introduction / José M. Garrido. – New York : Kluwer/Plenum, 2001. – 255 p.
12. Таха Х. А. Введение в исследование операций : Пер. с англ. / Хэмди А. Таха. – 6-е изд. – Москва : Издательский дом "Вильямс", 2001. – 912 с.
13. Строгалеv В. П. Имитационное моделирование / В. П. Строгалеv, И. О. Толкачева. – 2-е изд. – Москва : Издательство МГТУ им. Н. Э. Баумана, 2015. – 295 с.
14. Kelton W. D. Simulation with Arena / W. D. Kelton, R. P. Sadowski, D. A. Sadowski. – 2nd ed. – Boston : McGraw-Hill, 1998. – 547 p.
15. Altiock T. Simulation Modeling and Analysis with ARENA / T. Altiock, B. Melamed. – Burlington : Elsevier, 2007. – 458 p.
16. Rossetti M. D. Simulation Modeling and Arena / Manuel D. Rossetti. – 2nd ed. – Hoboken : John Wiley & Sons, Inc., 2016. – 744 p.
17. Стеценко І. В. Система імітаційного моделювання засобами сіток Петрі / І. В. Стеценко, О. В. Бойко // Математичні машини і системи. – Київ, 2009. – № 1. – С. 117-124.
18. Стеценко І. В. Формальное описание систем средствами Петри-объектных моделей / І. В. Стеценко // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка : Зб. наук. пр. – Київ : ВЕК+, 2011. – № 53. – С. 74-81.
19. Zaitsev D. Functional Petri Nets / D. A. Zaitsev // Universite Paris-Dauphine. – Cahier du Lamsade 224. – 2005. – P. 1-62.
20. Lakos C. A. Object Oriented Modeling with Object Petri Nets / Charles A. Lakos // Concurrent Object-Oriented Programming and Petri Nets. – 2001. – P. 1-37.
21. Lakos C. A. LOOPN++: A New Language for Object-Oriented Petri Nets / Charles A. Lakos, Chris Keen // Proceedings of Modelling and Simulation (European Simulation Multiconference). – Barcelona : Society for Computer Simulation, 1994. – P. 369-374.

22. University of Hamburg. Petri Nets Tools Database Quick Overview [Електронний ресурс] – Режим доступу до ресурсу: <http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html>.
23. И. В. Стеценко. Алгоритм имитации Петри-объектной модели / И. В. Стеценко // Математичні машини і системи. – Киев, 2012. – № 1. – С. 154-165.
24. Стеценко И. В. Теоретические основы Петри-объектного моделирования систем / И. В. Стеценко // Математичні машини і системи. – Киев, 2011. – № 4. – С. 136-148.
25. Stetsenko, I. V. Petri-Object Simulation: Software Package and Complexity / I. Stetsenko, V. Dorosh, A. Dyfuchyn // Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015). – Warsaw, 2015. – P. 381-385.
26. Питерсон Дж. Теория сетей Петри и моделирование систем : Пер. с англ. / Джеймс Питерсон. – Москва : Мир, 1984. – 264 с.
27. Стеценко І. В. Імітаційне моделювання систем управління засобами сіток Петрі / І. В. Стеценко, А. А. Данилюк // Вісник Черкаського державного технологічного університету. – Черкаси, 2005. – № 3. – С. 293-295.
28. Томашевський В.М. Візуальне середовище моделювання дискретно- подійних систем / Томашевський В.М., Дифучин А.Ю. // Системний аналіз та інформаційні технології: матеріали 18-ї Міжнародної науково-технічної конференції SAIT 2016, Київ, 30 травня – 2 червня 2016 р. / ННК “ІПСА” НТУУ “КПІ”. – К.: ННК “ІПСА” НТУУ “КПІ”, 2016. – С. 424-425 с.
29. Stetsenko Inna V., Dorosh Vitaliy I., Dyfuchyn Anton Petri-object simulation: sofyware package and complexity // Intelligent Data Acquisition and Advanced Computing Sysytems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference. – IEEE, 2015. – Vol.1. – P.381-385.
30. Стеценко І. В. Об'єктно-орієнтоване моделювання систем з використанням мереж Петрі / І. В. Стеценко // Вісник Черкаського державного технологічного університету. – Черкаси : ЧДТУ, 2011. – № 2. – С. 3-9.

31. Стеценко І. В. Технологія Петрі-об'єктного моделювання систем / І. В. Стеценко // Вісник Черкаського державного технологічного університету. – Черкаси : ЧДТУ, 2011. – № 4. – С. 25-30.
32. Стеценко І. В. Моделювання систем: навч. посіб. / І. В. Стеценко. – М-во освіти і науки України, Черк. держ. технол. ун-т. – Черкаси : ЧДТУ, 2010. – 399 с.
33. Стеценко И. В. Петри-объектное моделирование систем. Диссертация на соискание научной степени доктора технических наук [Электронный ресурс] / И. В. Стеценко. – Киев, 2012.
34. Тимченко А. А. Основи системного проектування та системного аналізу об'єктів. Основи системного підходу та системного аналізу об'єктів нової техніки: навч. посіб. / А. А. Тимченко. – Київ : Либідь, 2004. – 288 с.
35. Murata T. Petri Nets: Properties, Analysis and Applications / Tadao Murata // Proceedings of the IEEE. – 1989. – Vol. 77, No. 4. – P. 541-580.
36. Зайцев Д. А. Инварианты временных сетей Петри / Д. А. Зайцев // Кибернетика и системный анализ. – 2004. – № 2. – С. 92-106.
37. Стеценко І. В. Імітаційне моделювання системи управління навчальним процесом ВНЗ з використанням об'єктно-орієнтованого підходу / І. В. Стеценко // Математичні машини і системи. – Київ, 2011. – № 2. – С. 162-170.
38. Стеценко І. В. Петрі-об'єктна модель системи управління транспортним рухом / І. В. Стеценко // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка : Зб. наук. пр. – Київ : БЕК+, 2011. – № 54. – С. 116-125.
39. Fishman G. Discrete-Event Simulation: Modeling, Programming, and Analysis / George S. Fishman. – Berlin : Springer-Verlag, 2001. – 537 p. Loy A
40. Stetsenko Inna V., Dyfuchyn, A., Leshchenko, K., John, D., Web application for visual modeling of discrete event systems // Internet Technologies and Applications, ITA 2017 - Proceedings of the 7th International Conference 8101916, с. 86-91

ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ

ПЛАКАТ 1 Схема структурна архітектури веб-сервісу

ПЛАКАТ 2 Схема структурна таблиць бази даних

ПЛАКАТ 3 Структура вхідних даних

ПЛАКАТ 4 Схема структурна класів клієнтського застосування

ПЛАКАТ 5 Схема структурна класів серверного застосування

ПЛАКАТ 6 Копії екранних форм

ПЛАКАТ 7 Результати досліджень