Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній ступінь)

на тему: **Методи та засоби моделювання комп'ютерних систем на основі мереж Петрі**

Виконав: студент (ка) 6 курсу, групи СІм-61
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

	(підпис)	Ващук М.В. (прізвище та ініціали)
Керівник	(підпис)	Луцків А.М. (прізвище та ініціали)
Нормоконтроль	(підпис)	Луцик Н.С. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Осухівська Г.М. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра комп'ютерних систем та мереж

ЗАТВЕРДЖУЮ
Завідувач кафедри *Осухівська Г.М.*
«_____» _____ 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУРОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Ващуку Максиму Васильовичу*
(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) *Методи та засоби моделювання комп'ютерних систем на основі мереж Петрі*
Керівник проекту (роботи) *Луцків Андрій Мирославович, к.т.н., доц.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*28*» *вересня* 2020 року №*4/7-687*
2. Термін подання студентом завершеної роботи
3. Вихідні дані до роботи *Моделі комп'ютерних систем, види мереж Петрі, принципи трансляції програмного коду, застосування мови програмування С++*
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз сучасного стану досліджень у сфері моделювання комп'ютерних систем. 2. Розробка методу трансляції мережі Петрі у програмний код мови С++ 3. Розробка програмного засобу для моделювання комп'ютерних систем на основі мереж Петрі. 4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета дослідження. 2. Задачі дослідження, , об'єкт і предмет, наукова новизна і практична цінність дослідження. 3. Класифікація комп'ютерних систем. 4. Типи задач моделювання. 5(6). Класифікація та математичний опис мереж Петрі. 7. Типи переходів у мережі Петрі. 8. Вимоги, алгоритм роботи та ER-діаграма бази даних програмного засобу моделювання комп'ютерних систем на основі мереж Петрі. 9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	*Осухівська Г.М.*		
безпека в надзвичайних	*Стадник І.Я.*		
ситуаціях			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів проекту (роботи)	Примітка
1.	*Аналіз сучасного стану досліджень у сфері моделювання*	*29.09.2020-*	*виконано*
	комп'ютерних систем	*11.10.2020*	
2.	*Розробка методу трансляції мережі Петрі у програмний*	*15.10.2020 –*	*виконано*
	код мови С++	*25.10.2020*	
3.	*Розробка програмного засобу для моделювання*	*26.10.2020 –*	*виконано*
	комп'ютерних систем на основі мереж Петрі	*19.11.2020*	
4.	*Охорона праці та безпека в надзвичайних ситуаціях*	*21.11.2020 – 04.12.2020*	*виконано*
5.	*Оформлення пояснювальної записки*	*04.12.2020- 05.12.2020*	*виконано*
6.	*Оформлення графічного матеріалу*	*06.12.2020- 10.12.2020*	*виконано*
7.	*Попередній захист дипломної роботи магістра*	*13.12.2020*	*виконано*
8.	*Захист дипломної роботи магістра*	*22.12.2020*	

Студент _____ *Ващук М.В.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Луцків А.М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи та засоби моделювання комп’ютерних систем на основі мереж Петрі // Дипломна робота // Ващук Максим Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем та програмної інженерії, група СІм-61 // Тернопіль, 2020 // с. – 79, рис. – 19, табл. – 9, аркушів А1 – 9, додат. – 2 , бібліогр. – 22.

Ключові слова: метод, засіб, моделювання, комп’ютерна система, мережа Петрі.

У дипломній роботі магістра досліджено методи і засоби моделювання комп’ютерних систем із застосуванням апарату мереж Петрі з можливістю трансляції параметрів моделі, виражених природною мовою, у програмний код та візуалізацією архітектури системи.

Для того, щоб врахувати особливості структури комп’ютерної системи, потенційні маршрути передачі даних, затримки маршрутизації і забезпечити можливість швидкого встановлення «вузьких» місць системи перед початком її впровадження обгрунтовано та побудовано модель на основі EN-мереж Петрі.

У роботі запропоновано метод та процедури аналізу вхідного тексту параметрів мережевих моделей EN, спосіб перетворення одержаних лексем у код програми на мові С++.

Розроблено програмний засіб підтримки запропонованих моделі і методу трансляції мережі Петрі у програмний код та візуалізації її структури.

ABSTRACT

Methods and tools of computer systems modelling based on Petri networks / Master thesis/ Vaschuk Maksym Vasylyovych/ Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and software engineering, group CIm -61 // Ternopil, 2020// p. - 79, fig. – 19, table. – 9, Sheets A1 – 9, Add – 2, Ref. – 22.

Keywords: method, tool, modelling, computer system, Petri network.

In the master's thesis the methods and tools of modeling computer systems with the use of Petri nets with the possibility of translating the parameters of the model, expressed in natural language, into the program code and visualization of the system architecture are investigated.

In order to take into account the structure of the computer system, potential routes of data transmission, routing delays and to ensure the possibility of rapid establishment of "bottlenecks" of the system before its implementation, a model based on EN-Petri nets is justified.

The paper proposes a method and procedures for analyzing the input text of the parameters of network models EN, a method of converting the obtained tokens into program code in C ++. A software tool for supporting the proposed models and method of translating the Petri net into program code and visualization of its structure has been developed.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

БД	База Даних
КС	Комп’ютерні Системи
МП	Мережі Петрі
ПЗ	Програмне Забезпечення
EN	Enumerable Network
UML	Unified Modeling Language

ВСТУП

Актуальність теми. Теперішній розвиток методів, методологій та інструментальних засобів проектування комп’ютерних систем (КС) вимагає залучення значних фінансових та людських ресурсів, які покликані забезпечити реалізацію функціонально-повних та зручних у використанні програмно-апаратних комплексів у відповідності до потреб замовників чи «стейкхолдерів».

Одним з процесів, що дає змогу знизити рівень витрат та ризиків у загальному процесі розробки та впровадження КС чи їх компонентів, є процес моделювання архітектури майбутньої системи. Модель архітектури КС забезпечує можливість більш чітко зрозуміти основні сутності предметної області та зв’язки між ними, а графічне представлення архітектури – визначити потенційно «вузькі» місця та можливості щодо гнучкості, масштабування та внесення змін у систему.

Динамічні комп’ютерні системи характеризуються здатністю перетворення та обробки інформації і, на відміну від інших систем, основою їхнього функціонування є цифрові канали та засоби передачі даних, а також здатність до зміни своєї структури шляхом масштабування та/або переналаштування параметрів взаємодії між структурними компонентами.

Найбільш важливу роль при побудові та функціонуванні комп’ютерних систем відіграє надійність передачі даних та стійкість роботи в умовах зміни концептуальної архітектури, або зміни параметрів функціонування окремих компонентів.

Тому задача моделювання комп’ютерних систем стає все більш актуальною, оскільки часові та фінансові затрати на моделювання є значно меншими, ніж проектування, впровадження і подальша адаптація під потреби конкретних замовників.

На сьогодні ученими та інженерами досліджено та розроблено багато методів математичного та концептуального моделювання, які дають змогу відображати сутності предметної області та процеси, які у ній протікають. Дослідженню проблем моделювання у різних предметних областях, зокрема при побудові

технічних систем, присвячено праці як українських, так і закордонних учених, зокрема, А.Г. Івахненко, В.М. Томашевського, І.В. Стеценко, Ю.М. Теслі, А.А. Тимченко, Р. Шеннона, В. Кельтона, А. Лоу, В.Е. Котова та ін. Виходячи з результатів досліджень науковців, найбільш ефективним та універсальним методом при моделюванні комп'ютерних систем є мережі Петрі. Однак, враховуючи стрімкий розвиток комп'ютерних мереж, технологій IoT, використання «хмарних» сервісів, масивів великих даних (Big Data) необхідно провести додаткові дослідження щодо застосування апарату мереж Петрі при моделюванні комп'ютерних систем з метою адаптації його до сучасних вимог та потреб в галузі комп'ютерної інженерії. Актуальними задачами при цьому є розробка методу і засобу моделювання комп'ютерних систем на основі мереж Петрі з можливістю трансляції її параметрів, виражених природною мовою, у програмний код та візуалізацією архітектури системи.

Мета і задачі дослідження. Мета дипломної роботи магістра полягає у дослідження методів і засобів моделювання комп'ютерних систем із застосуванням апарату мереж Петрі з можливістю трансляції параметрів моделі, виражених природною мовою, у програмний код та візуалізацією архітектури системи.

Для цього в роботі розв'язуються наступні задачі:

– аналіз наукових та прикладних досліджень у галузі моделювання комп'ютерних систем;

– дослідження особливостей та способів використання апарату мереж Петрі;

– обґрунтування та формалізація концептуальної моделі комп'ютерних систем з врахуванням неоднорідності її компонентів;

– розробка методу трансляції параметрів моделі мережі Петрі, виражених природною мовою, у програмний код та візуалізацією архітектури системи;

– розробка програмного засобу автоматизованого перетворення параметрів мережі Петрі з візуалізацією архітектури комп'ютерної системи.

Об'єкт дослідження*:* процес моделювання комп'ютерних систем на основі апарату мереж Петрі.

Предмет дослідження: методи і засоби трансляції параметрів моделі мережі Петрі у програмний код, засоби візуалізації мережі Петрі.

Методи дослідження: Для того, щоб розв'язати поставлені задачі застосовано такі методи: аналіз та порівняння – при аналітичному огляді методів і засобів моделювання комп'ютерних систем; моделювання систем, теорія мереж Петрі – при створенні лексичних граматик та розробці методу трансляції параметрів моделі мережі Петрі в програмний код; програмування – при розробці програмного засобу трансляції параметрів, виражених природною мовою у код мови С++; експеримент і тестування – при апробації запропонованого методу і засобу.

Наукова новизна отриманих результатів. Наукова новизна полягає у вирішенні науково-практичної задачі класифікації атрибутів якості комп'ютерних систем, при цьому одержано наступні результати:

– уперше запропоновано метод та процедури, на основі правила лексичного аналізу вхідного тексту мови мережевих моделей EN і правила перетворення одержаних лексем в код програми на мові С++, що дає змогу формувати транслятор специфікації протоколів та одержувати модель динамічної комп'ютерної системи, аналізувати можливі стани системи та керувати параметрами задля оптимізації маршрутизації обміну даними між компонентами системи.

– набули подальшого розвитку моделі мереж Петрі і програмні засоби моделювання, які дали змогу забезпечити ефективність трансляції її параметрів, які задані вербально у програмний код на мові С++ .

Практичне значення одержаних результатів. Створено програмний засіб, що дає змогу транслювати параметри моделі мережі Петрі та налаштування переходів у програмний код мовою С++ та візуалізувати архітектуру комп'ютерної системи.

Публікації. Результати дослідження апробовано на IX міжнародній науково - технічній конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя та на VIII науково-технічній конференції

Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року) у вигляді тез конференцій.

1. Луцків А.М., Ващук М.В. Мережі Петрі як метод моделювання динамічних комп'ютерних систем. Матеріали IX міжнародної науково - технічної конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (26-27 листопада 2020 р.) Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: ТНТУ. 2020. С. 41.

2. Луцків А.М., Ващук М.В**.** Граматика перетворення параметрів моделі мережі петрі у програмний код мови С++. Матеріали VIII науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (9-10 грудня 2020 року). Тернопіль: ТНТУ. 2020. С. 98.

Структура роботи. До складу дипломної роботи магістра входить розрахунково-пояснювальна записка та графічний матеріал. Розрахунково-пояснювальна записка містить вступ, 4 розділи, загальні висновки, список використаної літератури і додатки. Обсяг роботи: розрахунково-пояснювальна записка – 81 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ ДОСІЛДЖЕНЬ У СФЕРІ ЗАДАЧ МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

1.1. Аналіз характеристик та особливостей функціонування комп'ютерних систем

На сьогодні спроектовано та експлуатується доволі багато різних за функціональним призначенням комп'ютерних систем, які покликані виконувати найрізноманітніші задачі. Згідно [1], комп'ютерні системи класифікують за наступними критеріями:

– спосіб проектування;
– функціональне призначення;
– місце і спосіб зберігання інформації;
– кількість центральних обчислювальних вузлів;
– тип апаратного забезпечення
– протоколи обміну даних
– топології з'єднань комп'ютерних систем.

За способом проектування чи побудови комп'ютерні системи можуть бути централізованими або розподіленими.

Щодо функціонального призначення комп'ютерні системи поділяють на системи автоматизованого опрацювання даних та автоматизовані системи контролю та управління, наприклад, технологічними процесами чи об'єктами.

В свою чергу автоматизовані системи опрацювання інформації поділяють на:

– інформаційні – основне функціональне призначення яких полягає в інформаційному обслуговуванні користувачів;

– обчислювальні – основне завдання яких полягає у виконанні задач обчислення, забезпечення обміном даними між вузлами комп'ютерної мережі;

– гібридні інформаційно-обчислювальні комп’ютерні системи – можуть одночасно виконувати функції інформаційних та обчислювальних комп’ютерних систем.

За принципом зберігання інформації у комп’ютерних системах їх поділяють на централізовані банки даних, які формується на одному з вузлів системи та розподілені, які зберігають дані в окремих локальних банках даних, розташованих в межах однієї комп’ютерної системи. Під банком даних вданому випадку розуміють, бази даних, системи керування базами даних і програмне забезпечення, що їх використовує.

Приклад комп’ютерних систем за способом проектування наведено на рис. 1.1.

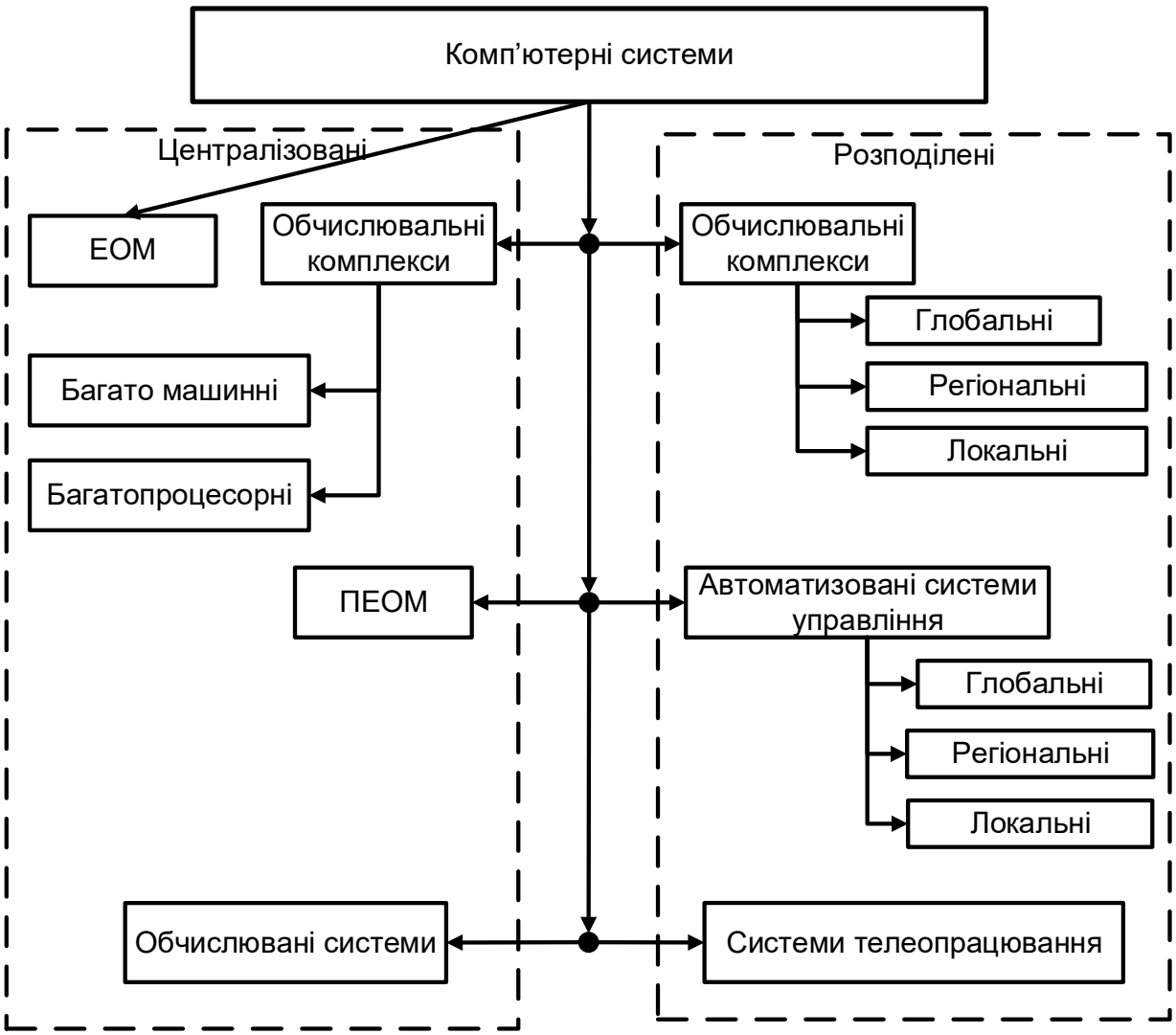


Рис. 1.1. Приклад класифікації комп’ютерних систем

За рівнем територіальної розподіленості можна виділити крупномасштабні, або глобальні обчислювальні системи (охоплюють територію однієї або кількох країн з відстанню між вузлами системи у тисячі кілометрів; регіональні системи, що охоплюють міста, райони, області і т.д; локальні обчислювальні системи з максимальним розподілом між вузлами на відстань кількох кілометрів.

За кількістю центральних обчислювальних вузлів виділяють мережі з кількома або одним таким вузлом. Останні відносяться до обчислювальних систем телеопрацювання, що представляють собою комплекси, до складу яких входять самі обчислювальні вузли та віддалені абонентські пункти (АП), які комунікують за допомогою різних каналів передачі даних.

За типом апаратного забезпечення виділяють однорідні системи – системи, до складу яких, входять сумісні програмно-апаратні платформи та неоднорідні – різне програмно-апаратне забезпечення. На практиці комп'ютерні системи та мережі часто є неоднорідними.

За методом передачі даних розрізняють комп'ютерні системи, що використовують обчислювальні мережі з:

– комутацією каналів;
– комутацією повідомлень;
– комутацією пакетів;
– змішаною комутацією.

Сучасні комп'ютерні системи та мережі використовують комутацію пакетів. Комутація пакетів є результатом розвитку методу комутації повідомлень. Вона дозволяє отримати подальше розширення пропускної здатності мереж, швидкості та надійності передачі даних. Повідомлення, яке надсилає абонент розбивається на пакети, які мають фіксовану довжину, наприклад 1 Кбайт. Пакети помічаються службовою інформацією-заголовком, вказується адреса пункту відправлення, адреса пункту призначення та унікальний номер пакету у повідомленні. У системі передачі даних між абонентами з комутацією пакетів використовуються два способи передачі: дейтаграмний та віртуальний. Дейтаграмний спосіб передбачає передачу даних у вигляді окремих, не пов'язаних між собою пакетів. Важливою

перевагою дейтаграмного способу комутації пакетів є можливість одночасної їх передачі різними маршрутами, що дає змогу зменшити час і збільшити надійність передачі повідомлень. При передачі короткими пакетами зменшуються ймовірність появи повідомлень про помилку та час зайнятості каналів повторним обміном. Однак при цьому спостерігаються випадки «обгону» повідомлень. Прив’язка повідомлень до часу їх видачі та нумерації дозволяють виявляти цю проблему. При використанні дейтаграмного способу передачі даних не гарантується порядок та надійність доставки пакетів.

При використанні віртуального способу, передача даних виконується у вигляді послідовності пов’язаних між собою пакетів. Організація віртуального каналу між двома процесами прирівнюється до виділення їм дуплексного каналу зв'язку, за яким дані передаються в тій послідовності, у якій були відправлені. Віртуальний канал зберігає всі вищеописані переваги комутації пакетів щодо швидкості передачі та мультиплексування, але не вимагає попередніх процедур встановлення з'єднань. Після завершення сеансу зв'язку канал зникає і повертає ресурси для встановлення нових віртуальних з’єднань.

Важливою ознакою класифікації комп'ютерних систем є спосіб організації топологій з’єднань між вузлами мережі. Топологічна структура мережі має значний вплив на її пропускну здатність, стійкість компонентів комп’ютерних систем до відмов, функціональні можливості та вартість мереж. У даний час спостерігається велике різноманіття топологій комп’ютерних мереж. На рис. 1.2 наведено приклад топологій комп’ютерних мереж, які використовуються при проектуванні складних, динамічних систем. Топологія великомасштабних комп'ютерних систем може представляти собою комбінацію декількох топологій. В обчислювальних мережах (системах) абоненти містять спеціальне програмне забезпечення для мережного опрацювання даних. До програмних засобів висувають вимоги щодо збереження працездатності мережі при зміні її структури, при відмовах окремих вузлів в каналів передачі даних, а також забезпечення здатності роботи компонентів комп’ютерної системи з термінальними станціями і взаємодії неоднорідного обладнання.

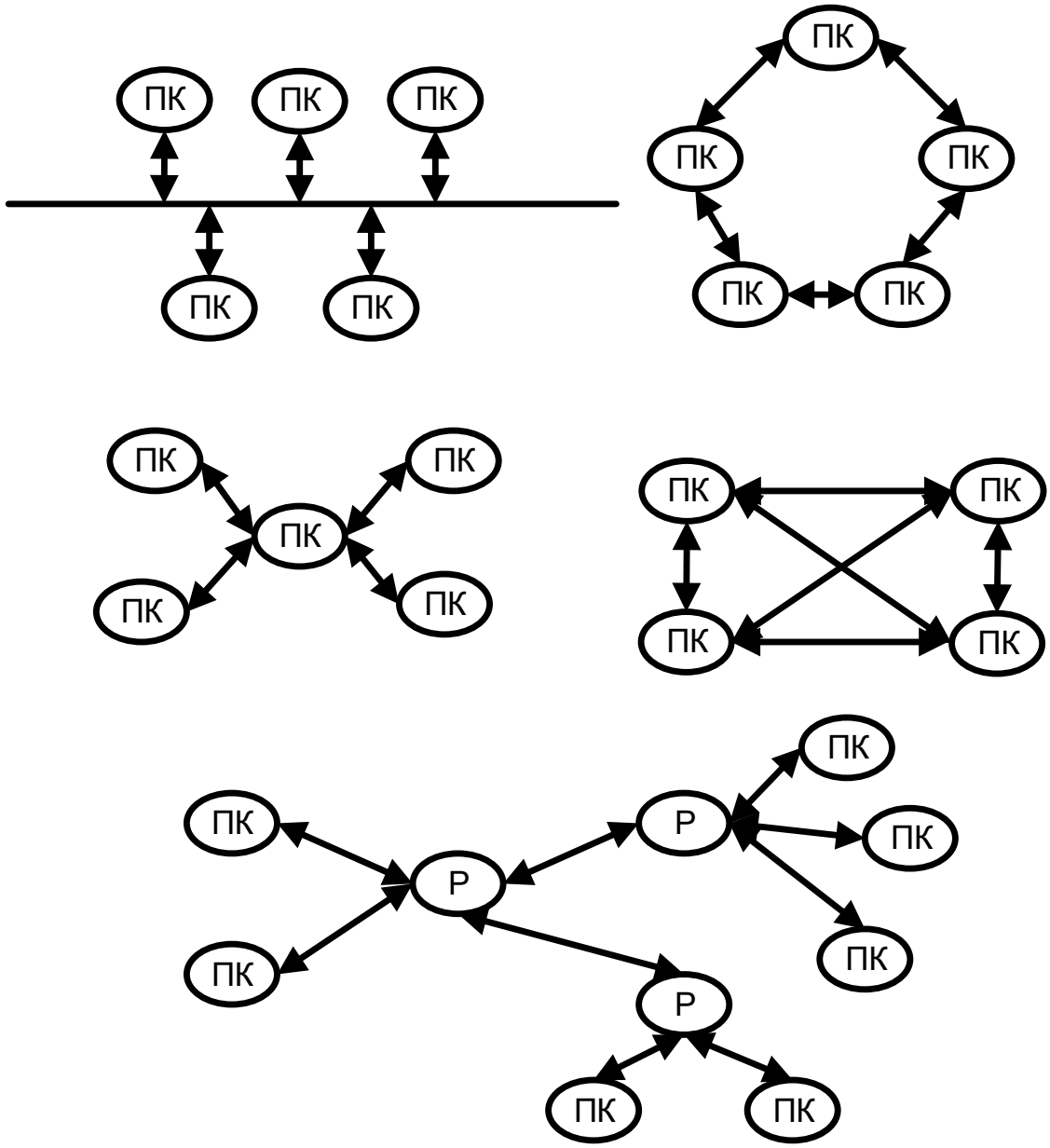


Рис. 1.2. Приклади топологій організації мереж комп’ютерних систем

Враховуючи різноманітність і сферу застосування комп’ютерних систем, а також сучасні тенденції до їх масштабованості, змінюваності конфігурації, віртуалізації вартість проектування і створення прототипів стає дороговартісною процедурою. У зв’язку з наведеними вище аргументами, актуальним є дослідження та обґрунтування універсальних методів і засобів моделювання комп’ютерних систем, що дозволить знизити витрати на прототипування і в подальшому проектування та імплементацію комп’ютерної системи. Проведемо аналіз особливостей моделювання комп’ютерних систем.

1.2. Специфіка моделювання комп’ютерних систем

Оскільки мета дипломної роботи тісно пов’язана з процесом моделювання, а саме із застосуванням апарату мереж Петрі для представлення функціонування комп’ютерних систем і трансляції параметрів моделі у програмний код, то необхідно проаналізувати особливості цього процесу та об’єктів, які моделюються.

У загальному випадку, під моделлю розуміють представлення деякого об’єкта чи системи у вигляді абстракції, що дає змогу застосувати інструментарій того рівня і типу, який забезпечує зручність дослідження як самого об’єкта, так і його характеристик.

Вхідними параметрами моделей є опис реального об’єкту чи процесу у вигляді множини змінних X. Модель, зазвичай, презентується у вигляді сукупності параметрів P, наприклад кількість вузлів комп’ютерної мережі, їх тип, наявність черг при опрацюванні повідомлень та інші, а також алгоритмами, аналітичними функціями чи іншими засобами, які дають змогу формально побудувати залежність між вихідними та вхідними змінними. Вихід моделі представляється як сукупність вихідних змінних, тобто відображається реакція системи на вхідні змінні. На рис. 1.3 схематично наведено структуру моделі у загальному випадку.

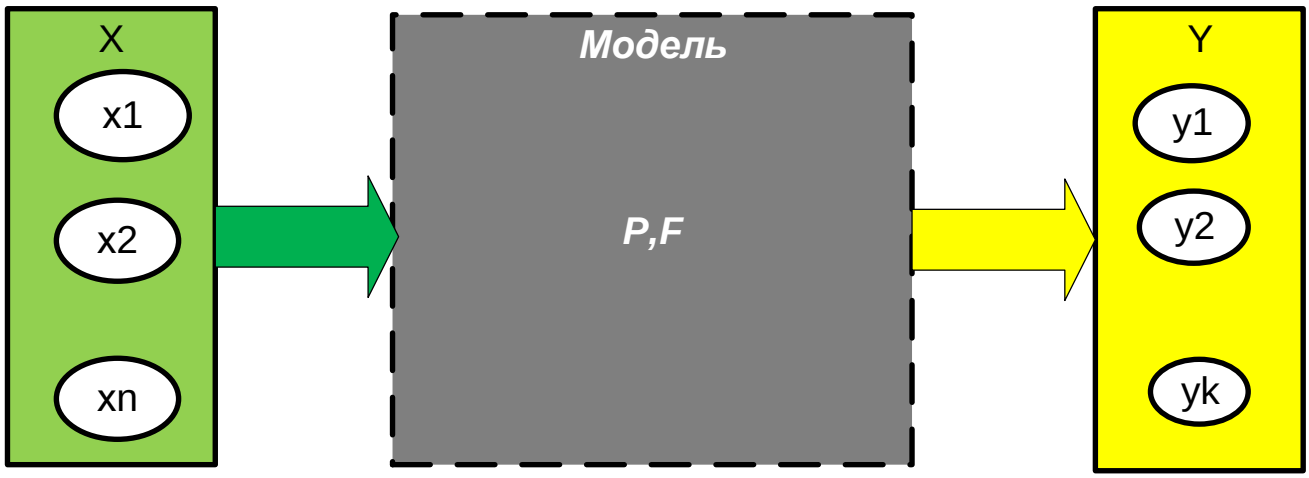


Рис. 1.3. Візуалізація поняття моделі для загального випадку

Для прикладу, модель комп’ютерної системи, в основі якої лежить передача даних через комп’ютерні мережі, можна представити у вигляді системи масового обслуговування.

Вхідними змінними, в такому випадку, можуть виступати інтенсивність потоку вхідних повідомлень при обміні даними, час їхнього опрацювання відповідним вузлом комп’ютерної системи, розподіл імовірностей щодо оптимального маршруту обміну даними та ін. Відповідно, набір вихідних змінних $y_1, y_2, \dots, y_k$ може формувати інтенсивність потоку вихідних пакетів даних, усереднене значення затримок опрацювання повідомлень, довжину черги на певному вузлі комп’ютерної системи та ряд інших, в залежності від типу системи.

Параметрами такої моделі комп’ютерної системи можна вважати, наприклад, кількість обчислювальних вузлів, опрацювання даних на основі черг або без них, обмеження кількості позицій у черзі і т.п.

Алгоритм F може здійснювати розрахунок $y_1, y_2, \dots, y_k$ на основі значень вхідних змінних $x, x_2, \dots, x_n$. В залежності від типу та функціональності комп’ютерної системи в основі алгоритму може лежати проста функція обчислення вихідних змінних або сукупність складних вкладених алгоритмів на основі деяких математичних методів.

На практиці використовують два шляхи для побудови моделей. Суть першого підходу полягає у дослідженні об’єкту, у даному випадку комп’ютерної системи, щодо визначення особливостей і законів функціонування системи. Після цього будується модель на основі якої імітується поведінка об’єкта. Особливістю такої фізичної моделі є те, що параметри P інтерпретують реальні процеси, які відбуваються у системі.

Застосування іншого способу щодо побудови моделі передбачає відсутність фізичного обґрунтування процесів, тобто алгоритм чи функція F обирається на основі знань дослідника, а параметри P є невідомими. Їх визначають у результаті проведення експериментів на основі спостереження за вхідними змінними X та вихідними значеннями Y. Фізика процесів, які протікають у системі, у параметрах P не відображається, а моделі такого типу є не фізичними.

Часто у літературних джерелах [2-5] і при тестуванні систем різної природи [6-7] вживають термінологію «прозорої (біла) скринька», «сіра скринька» та «чорна скринька». Це пов’язано з тим, що у випадку «прозорої скриньки» та «сірої скриньки» відомі параметри моделі P, тобто модель дає змогу інтерпретувати фізичний зміст процесів. У випадку «чорної скриньки», модель не містить фізичного змісту параметрів.

У дипломній роботі для досягнення мети роботи необхідно використовувати «сіру скриньку», тобто фізичну модель, оскільки природа процесів, які протікають у комп’ютерній системі є відомою повністю або частково.

В залежності від типу та виду вихідних змінних, моделі можуть бути статичними або динамічними. У випадку, коли змінна Y не зазнає змін у часі, то модель є статичною, в іншому випадку – динамічною. Оскільки, комп’ютерні системи в сучасних умовах є гнучкими і масштабованими, то вихідні змінні можуть змінюватися у часі. У зв’язку з цим, комп’ютерну систему потрібно описувати із застосування динамічних моделей. Враховуючи той аспект, що зміна Y для таких систем відбувається лише у певні моменти часу, а решту часу вона перебуває у стані без змін, то крім того, що модель комп’ютерної системи є динамічною, вона ще є і дискретною. У випадку динамічної дискретної моделі, коли зміна Y є передбачуваною, то модель є детермінованою, в іншому випадку – стохастичною.

Існує багато видів та інструментів побудови і представлення моделей, в залежності від природи об’єкту та розділу математики, який використовується для дослідження об’єкта, наприклад, імітаційні моделі, моделі, представлені у вигляді диференційних рівнянь, алгебраїчні, графові та векторні моделі.

При моделюванні систем, процесів чи явищ можна використовувати різні моделі, що дають змогу описати їх з різних точок зору та одержати більше інформації про поведінку, структуру, стійкість та здатність до змін досліджуваних об’єктів. Пряма задача моделювання призначена для знаходження розв’язку Y (вихід системи) за відомими вхідними значеннями змінних X, відомою моделлю F та її параметрами P [5].

Задача, суть якої полягає у знаходженні значень вхідних змінних за відомою моделлю, її параметрами та вихідними значеннями є задачею управління при проектуванні комп'ютерних систем. У випадку, якщо необхідно знайти значення параметрів моделі за відомими вхідними та вихідними змінними, а також відомою множиною моделей, то розв'язується задача ідентифікації. Результатом розв'язку такої задачі є деяка єдина модель з наявної сукупності моделей із встановленими параметрами, які задовольняють вхідним і вихідним змінним.

При розв'язанні задачі оптимізації, відомою є модель, критерій оптимізації та можливі вхідні значення змінних. При цьому необхідно визначити реальні значення вхідних змінних, параметри моделі та відповідно вихідні значення змінних, які б відповідали критерію оптимізації. Ще одним видом задач моделювання є задача прогнозування. Її суть можна сформулювати наступним чином: за відомими значеннями як вхідних, так і вихідних змінних, які були притаманні моделі до деякого моменту часу і за відомим часовим інтервалом прогнозування необхідно побудувати деяку модель з відповідними параметрами, які б найкраще відтворювали поведінку системи. У табл 1.1 схематично наведено типи задач моделювання та їх ознаки.

Таблиця 1.1

Типи задач моделювання

Назва задачі	Вхідні змінні, X	Модель, F	Параметри моделі, P	Вихідні змінні, Y	Цільова змінна
Задача прямого моделювання	+	+	+	-	Y => ?
Задача управління	-	+	+	+	X => ?
Задача оптимізації	-	+	-	-	-
Задача ідентифікації	+	+{F}	-	+	f, P => ?
Задача прогнозування	+	-	-	+ (Y_t)	F, P, Y_{t+T} => ?

Виходячи з проведеного аналізу основних концепцій моделювання, можна зробити висновок про те, що моделі сучасних комп’ютерних систем представляють собою динамічні дискретні і, в більшості випадків, детерміновані моделі.

Для моделювання функціональності, структури компонентів та зв’язків між ними у комп’ютерних системах можна використати один з методів, які наведені на рис. 1.4. При цьому, аналітично виразити функціональність та архітектуру комп’ютерних систем доволі складно, а іноді взагалі неможливо, тому на практиці використовують підхід імітаційного моделювання.

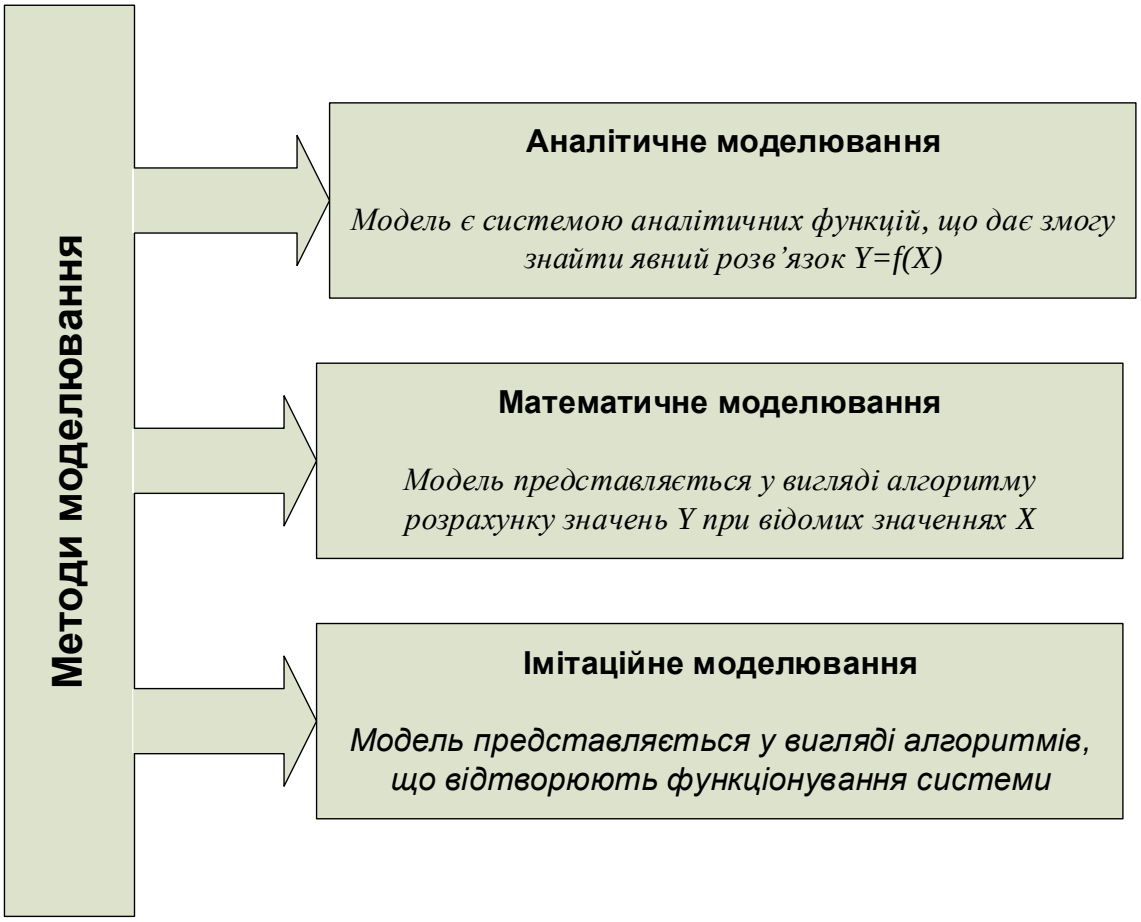


Рис. 1.4. Методи моделювання комп’ютерних систем

Для моделювання комп’ютерних систем у дипломній роботі пропонується використати апарат мереж Петрі, що відноситься до імітаційних методів моделювання. Проведемо аналіз основних концепцій та сфер застосування мереж Петрі.

1.3. Аналіз сфери та особливостей застосування мереж Петрі

Теорію мереж Петрі запропоновано одноіменним німецьким ученим Карлом-Адамом у час науково-технічного прогресу, зокрема при появі багатопроцесорних комп’ютерних систем [7]. Моделі паралельно-працюючих процесорів вимагали створення нових теоретичних засад для опису їх роботи і прогнозування поведінки.

Мережі Петрі (МП) набули широкого практичного застосування, оскільки дозволяли змоделювати функціонування декількох пристроїв, які описуються окремими взаємодіючими автоматами. Тому теорія МП є розвитком теорії цифрових автоматів і дає змогу в єдиній термінологічній і методологічній базі використовувати програмне забезпечення, апаратну платформу та інформаційну підтримку при проектуванні комп’ютерних систем.

Мережі Петрі є наочним, математично формалізованим апаратом для побудови моделей поведінки паралельних систем з асинхронною взаємодією. Перевагою використання підходу МП є те, що він дозволяє у зручному вигляді презентувати структуру комп’ютерних систем, зв’язки між її компонентами за встановлених початкових умов. При цьому забезпечується високий рівень абстракції на основі використання лише двох простих об’єктів – подій та умов.

Моделювання комп’ютерних систем на основі мереж Петрі є досить ефективним імітаційним інструментом, оскільки дозволяє інтегрувати граф, як статичну структуру у дискретні динамічні системи, а це дозволяє більш точно побудувати модель реальної системи.

Характерною особливістю МП є те, що у них відсутній строго фіксований аналітичний порядок взаємодії між вхідними та вихідними вузлами системи і система стає алгоритмічно не визначено. Тому використання мереж Петрі є ефективним засобом моделювання комп’ютерних систем, що поєднує переваги аналітичного та імітаційного моделювання.

Сферами застосування мереж Петрі є не тільки моделювання процесів і динамічних комп’ютерних систем, але й галузь теоретичного програмування, зокрема при розв’язані задач формальної специфікації функціональних вимог, у

процесах верифікації програмного забезпечення (Model Checking), організації та управління обчислювальних процесів.

Застосування МП є ефективним при розв’язанні наступних задач:

– інтерпретація – адекватне представлення програмної структури, що змодельована у вигляді деякої мережі Петрі;

– програмування моделі під конкретне операційне середовище;

– дослідження моделі об’єкту моделювання;

– крос-трансляція з мови мереж Петрі на мови програмування, що є однією із задач дипломної роботи.

Враховуючи потужність інструментального набору мереж Петрі для вирішення наведених вище задач, сьогодні спостерігається стрімкий їхній розвиток та модифікації. Вербально узагальнений процес побудови мереж Петрі можна описати наступним чином.

Найбільш простими, з обмеженим функціоналом, мережами Петрі є автоматні мережі і марковані графи. Такі структури передбачають один вхід і один вихід для переходів (дуг) і позицій (вершин) відповідно [7].

Якщо ввести мультиплікативність для дуг і кількості маркерів у вершинах (позиціях), то це дозволить змоделювати не тільки процес функціонування паралельних систем, але й забезпечити здатність прогнозування динаміки використання ресурсів. Таке представлення мереж забезпечує зростання можливостей щодо опису фізичних моделей, однак вимагає впровадження штучних елементів для опису взаємодіючих процесів, що ініційовані в одних і тих же функціонально неподільних підсистемах. Такими штучними елементами є забарвлені вершини і дуги. Окрім цього, для комп’ютерних систем важливим є час виконання операцій, або затримки на кожному з вузлів системи, тому окрім кольору, вводять також і часові параметри.

Однією з найважливіших областей застосувань мереж Петрі є управління дискретними об’єктами, якими представляються комп’ютерні системи. У даному випадку апарат МП може використовуватись на стадіях проектування і побудови алгоритмічного базису для управління.

До складу сучасних комп’ютерних систем входять набори різних компонентів, які можуть відрізнятися своїми властивостями, функціональним призначенням, складністю внутрішньої структури. Для того, щоб сконструювати адекватну математичну модель на основі мереж Петрі потрібно:

– визначити тип розв’язуваних задач (якісні, кількісні), наприклад: чи відповідають виконувані функції системи її призначенню; чи задовольняє система критеріям ефективності; чи можливе настання помилкових чи аварійних ситуацій при функціонуванні системи; чи існують «вузькі» місця; чи підлягає система спрощенню без порушення функціонування; чи можна з існуючих систем побудувати складнішу, яка задовольняє висунутим заданим вимогам і т.п.;

– забезпечити відображення структурних та функціональних особливостей складових комп’ютерних систем у певні абстрактні форми за допомогою введення концептуальних понять з подальшим переходом до математичного представлення.

Компоненти системи з відповідними операціями описуються абстрактними подіями, для прикладу: виконання оператора циклу у коді програми, ініціалізація переривання операційної системи, початок деякої стадії проекту і т.д. [7]. Абстрактна подія може відбутися одноразово, багаторазово та не відбутися жодного разу. Сукупність таких подій у системі формують процес її функціонування. У загальному випадку, за одних і тих же умов одна система може функціонувати по-різному, тобто бути не детермінованою.

Взаємодія подій у великих асинхронних системах має, як правило, складну динамічну структуру Для представлення таких складних подій (глобальних ситуацій) використовуються так звані локальні умови реалізації подій. Ці умови визначають три стани:

– 0 – умова не виконана;

– 1 – умова виконується;

– n – умова виконана з n-кратним запасом.

При виконанні певної послідовності умов можливе настання деякої події, а те, що подія відбулася може змінити певні умови (постумови). Виходячи з цього, події взаємодіють з умовами, а умови – з подіями. Таким чином, передбачається,

що для вирішення задач досить уявити систему як структуру, утворену з елементів двох типів – подій та умов. У мережах Петрі події і умови представлені абстрактними символами з двох алфавітів, які не перетинаються – множина переходів та множина позицій.

Тому актуальною задачею, є розробка методу і засобу, які б давали змогу за вказаними параметрами будувати мережу Петрі і забезпечувати трансляцію з природної мови в програмний код.

1.4. Висновки до розділу

1. Проаналізовано особливості та класифікацію комп’ютерних систем, підходи до їхнього проектування, що дало змогу обґрунтувати актуальність та необхідність моделювання як їхньої структури, так і функціональної поведінки, заважаючи на сучасні вимоги до масштабованості і гнучкості комп’ютерних систем, а також витрати на прототипування.

2. Проведено аналіз специфіки процесу моделювання комп’ютерних систем у результаті якого встановлено, що більшість сучасних систем представляються у вигляді імітаційних моделей, що є фізичними динамічними структурами з дискретними або стохастичними станами.

3. Проаналізовано сферу застосування та особливості представлення мереж Петрі, що дало змогу обґрунтувати їх застосування при моделюванні структури та поведінки динамічних комп’ютерних систем.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ТРАНСЛЯЦІЇ МЕРЕЖІ ПЕТРІ У ПРОГРАМНИЙ КОД МОВИ С++

2.1. Обґрунтування вибору класу мереж Петрі для представлення комп'ютерних систем

Мережі Петрі можуть застосовуватись для того, щоб забезпечити ефективність моделювання паралельних асинхронних процесів, які протікають у динамічних комп'ютерних системах та здійснити алгоритмічний опис об'єкту. МП, з точки зору їх імплементації, поділяють на три основні класи, що відрізняються між собою принципами організації позицій та переходів у мережі. До цих трьох класів входять:

– безпечні МП – мережі, у вершинах (позиціях) яких при виконанні переходу(ів) може бути відсутня або наявна лише одна мітка;

– обмежені МП – мережі, в яких у вершинах міститься цілочисельне значення міток, а дуги, у вигляді цілих чисел, визначають кількісний їх розподіл у мережі після того, як вони пройшли через переходи (перехід спрацював);

– Е-мережі Петрі – тип мереж, де переходи можуть належати до кількох наперед визначених видів, і спрацювання яких відбувається лише за наявності визначеної кількості міток у мережі, а складні макропереходи здатні до зміни своєї структури.

У випадку, коли немає обмежень на мережі Петрі, то вони належать до класу загальних або узагальнених мереж. Перевагою таких мереж є те, що вони дають змогу компактно зображати взаємодію компонентів у складних комп'ютерних системах. Даний клас мереж Петрі є особливо важливим, зважаючи на стрімкий розвиток і застосування у різних прикладних сферах і, з однієї сторони, забезпечує строгу класифікацію, а з іншої – здатність до інтерпретації, розширення та модифікації. Проведемо аналіз підкласів загальних мереж .

2.1.1. Забарвлені мережі Петрі. Забарвлені мережі Петрі або по іншому мережі високого рівня дають змогу якісно представляти паралельні взаємодіючі процеси. Компактність представлення мережі забезпечуються тим, що маркери володіють властивістю (змінною) кольору, а кратність дуг визначається у вигляді функцій від властивості кольору. При цьому компоненти матриці інцидентності представляються не дискретною величиною, а функцією від властивості кольору. Застосування забарвлених мереж показали свою ефективність при моделюванні протоколів передачі даних, комп'ютерних систем на основі черг та ін.

2.1.2. Мережі на основі предикатів. Мережі на основі предикатів є розвитком та модифікацією забарвлених мереж Петрі. Характерною особливістю цього класу є те, що маркером може виступати як колір, так і змінні, що належать області визначення кольорів мережі. Окрім цього, для кожного переходу задається деякий предикат і в залежності від його істинності відбувається або не відбувається його спрацювання.

На даний час можуть використовуватись декілька модифікацій мереж на основі предикатів, зокрема з мережі з унарними предикатами, Е-мережі, забарвлені мережі з інтеграцією предикатів. Такий клас МП показав ефективність застосування при верифікації мережевих протоколів, паралельного та розподіленого програмного забезпечення.

2.1.3. Часові мережі. У загальному випадку, розрізняють два типи часових мереж:

– часові мережі, що характеризуються часом протягом якого маркер перебуває у позиції;

– часові мережі на основі аналізу часу спрацювання переходу.

Перший тип часових мереж на практиці застосовують у випадку, коли необхідно визначити умови функціонування за найкоротший проміжок часу. Другий тип мереж дає змогу встановити та визначити умови щодо здатності до імплементації наперед заданих періодичних режимів, які задаються характеристичними векторами за найкоротший час («максимальна швидкість»).

2.1.4. Стохастичні мережі. Стохастичні мережі є трансформацією часових мереж. Сфера застосування цих мереж – оцінювання надійності та продуктивності комп'ютерних систем. У них для кожного переходу задається імовірність його спрацьовування за деякий інтервал часу. Поведінка мережі, а також її властивості, відповідають критеріям загальних МП без інтерпретації. Для різних мереж характерні деякі власні вершини і зв'язки, однак за однакової структури можливі різні множини станів системи та діаграми. На рис. 2.1 наведено узагальнену інтерпретацію мереж Петрі.

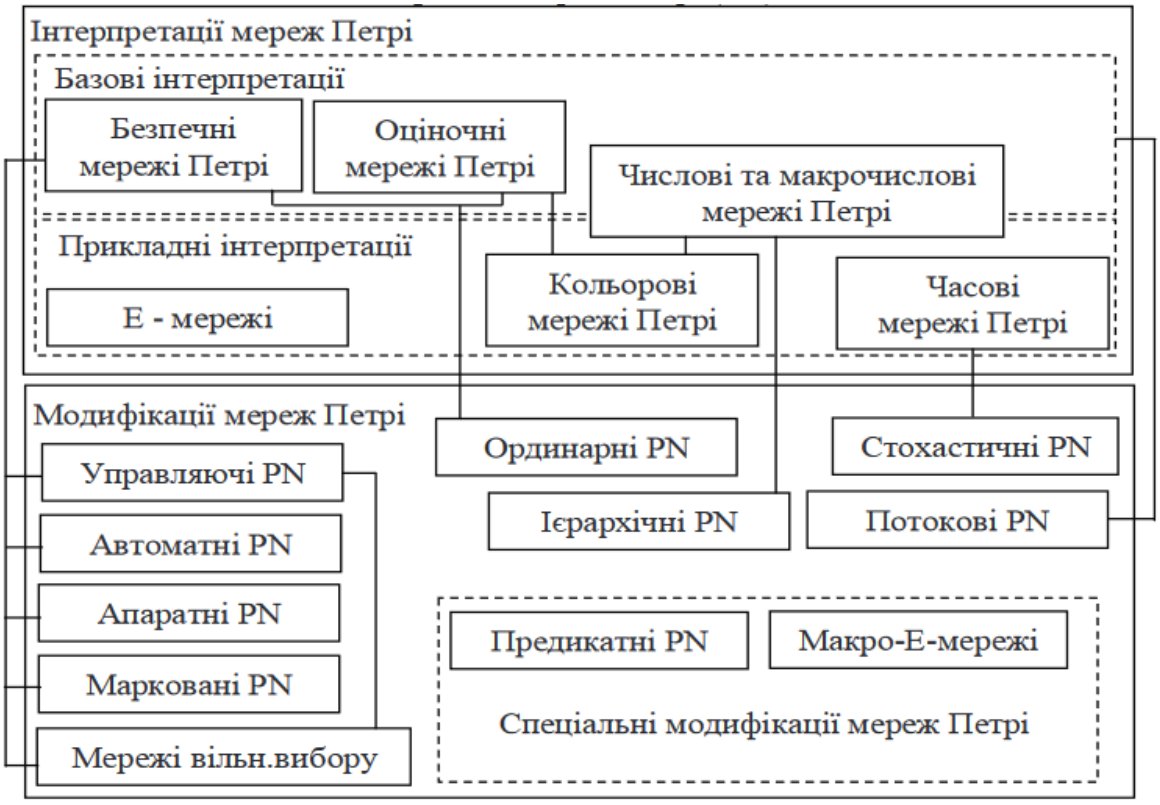


Рис. 2.1. Класифікація МП

Отже, важливим аспектом дослідження комп'ютерних систем на основі мереж Петрі є аналіз їх статичних і динамічних характеристик, тобто статичні конфігурації описують архітектуру комп'ютерних систем, а динамічні – процес взаємодії між компонентами системи.

2.2. Модель представлення комп'ютерної системи за допомогою мереж Петрі

Для представлення структури динамічної комп'ютерної системи можна скористатися описом моделі скінченої мережі Петрі у вигляді

$$EN =< P, T, F, M, G > \tag{2.1}$$

де $P = \{p_1, p_2, \dots, p_n, r_1, r_2, \dots, r_k\}$ – скінченна множина позицій, що включає підмножини простих позицій p і дозволяються позиції r;

$T = \{t_1, t_2, \dots, t_l\}$ – скінчена множина переходів;

$F \subseteq \{I: P \times T \to (0,1)\} \cup \{0: T \times P \to (0,1)\}$ – множина потокових відношень, що містить вхідні і вихідні позиції для кожного переходу;

$M: P \to (0,1,2 \dots)$ – функція розмітки, що визначає маркування позицій у вигляді додатних цілих чисел;

$G: P \to Y$ – функція стану пам'яті мережі, що описує стани векторів комірок пам'яті мережі для кожного варіанту розмітки через вектори пам'яті позицій, помічених мітками з визначеними атрибутами.

Формально функціонування мережевої моделі EN описується виразами зміни маркування (2.2) і змінами стану пам'яті мережі (2.3)

$$\forall p \in P, M'(p) = M(p) - I(p,t) + Q(t,p) \tag{2.2}$$

$$\forall p \in P, G'(p) = G(p) - I(p,t)A(p) + Q(t,p)\Pi(A) \tag{2.3}$$

де $\Pi = \{\rho_1, \rho_2, \dots, \rho_i\}$ – процедури перетворення атрибутів міток для кожного з i переходів мережі;

$A(p)$ – вектор атрибутів мітки, що знаходиться у позиції p.

Мережна модель EN довільної складності складається з фіксованого набору конструкцій, які називають примітивами або елементарними мережами *en(t)*, які визначаються як сукупність, що складається з переходу *t* і множини усіх структурних відношень, у яких бере участь перехід *t*.

en(t), що описує перехід *t*, можна описати формулою:

$$en(t) = \langle X, Y, \Psi, \tau \rangle \tag{2.4}$$

де X, Y – скінченні множини вхідних і вихідних позицій переходу *t* відповідно;

Ψ – процедура функціонування переходу *t*;

τ – функція часової затримки;

Процедура Ψ визначається сукупністю функцій C, r_1, r_2 і ρ або

$$\Psi = \{C, r_1, r_2, \rho\} \tag{2.5}$$

де C – необхідна умова спрацювання переходу *t*;

r_1, r_2 – функції вхідного і вихідного вибору відповідно;

ρ – функція перетворення атрибутів міток елементарної мережі.

Умова *C* задається у вигляді логічного виразу, аргументами якого є $\mathrm{B} = \{b, \bar{b}\}$, що позначають доступність чи недоступність вхідних і вихідних позицій для міток.

Функції r_1 і r_2 записують у вигляді виразу умови

$$r_i(t) = \{\Pi_{i1} \to X_{i1}, \Pi_{i2} \to X_{i2}, \dots, \Pi_{il_i} \to X_{il_i}\}, i = 1,2, \dots \tag{2.6}$$

де $П_{ij}$ – деякий предикат;

X_{ij} – визначена підмножина вхідних ($i = 1$) або вихідних ($i = 2$) позицій; $j = 1,2, \dots, l_i$.

Значенням функції $r_i(t)$ є підмножина $X_{i,j}$, якому відповідає перший істинний предикат $П_{ij}$, при перегляді умовного виразу зліва направо.

Значення предиката $П_{ij}$ визначається в залежності від розмітки заданої елементарної мережі, а також від стану пам'яті мережі і атрибутів міток.

Функція ρ визначає перетворення пам'яті елементарної мережі, що виконується в кінці фази активності переходу, який спрацював

$$\rho = [\rho_1 \to (l_{11}, l_{12}, \dots, l_{1k}); \dots; \rho_s \to (l_{s1}, l_{s2}, \dots, l_{sm})] \quad (2.7)$$

де $l_{11}, \dots, l_{sm}$ – операції над описом міток.

Задача трансляції в загальному випадку передбачає перетворення тексту, описаного на деякій вхідній мові L_{inp}, в об'єктно-орієнтовану програму на вихідній мові L_{out} [5]. Формально її можна представити у вигляді

$$Z = F(W) \quad (2.8)$$

де W – сукупність символів з алфавіту A вхідної мови L_{inp};

Z – сукупність, що складається із символів алфавіту B вихідної мови L_{out};

F – функція (набір правил) перетворення.

Специфікація протоколу, описаного на мові мережевих моделей EN, представляється у вигляді графічного опису логічної структури (рис. 4.1) і таблиці, що представляє множину вхідних даних (табл. 4.1), що також включає визначення початкового стану мережі.

Таблиця 2.1

Множина вхідних даних

	r_1	r_2	τ	A	ρ
t_1	$M(p_1) - 1$	$M(p_2) + 1$	40	5	$f(a_1) = a_1 + 1$
t_2	$M(p_2) - 1$	$(a_4 = 1): M(p_3) + 1$ $(a_4 \neq 1): M(p_4) + 1$	3	5	$f(a_3) = 0$
$M_0 = \{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$					
$G_0 = \{A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$					

Для розробки правил трансляції необхідно, в першу чергу, визначити синтаксичну і семантичну структуру мови мережевих моделей EN. Для цього скористаємось визначеннями з теорії формальних мов.

Мовою L називається множина ланцюгів скінченної довжини в алфавіті Σ.

Граматикою G_L називається математична система, що визначає мову L.

Формально граматика описується наступним чином

$$G = (N, T, P, S) \tag{2.9}$$

де N – кінцева множина нетермінальних символів (нетерміналів);

T – скінченна множина термінальних символів (терміналів) ($N \cap T = \emptyset$. множини T і N – не перетинаються)

P – скінченна не порожня множина правил (продукцій), кожне з яких має вигляд $\alpha \to \beta$, де α і β – ланцюги над об'єднаним алфавітом $N \cup T$.

S – виділений елемент нетермінального словника, $S \in N$.

На рис. 2.2 схематично показано модель логічної структури протоколу EN_{PR}.

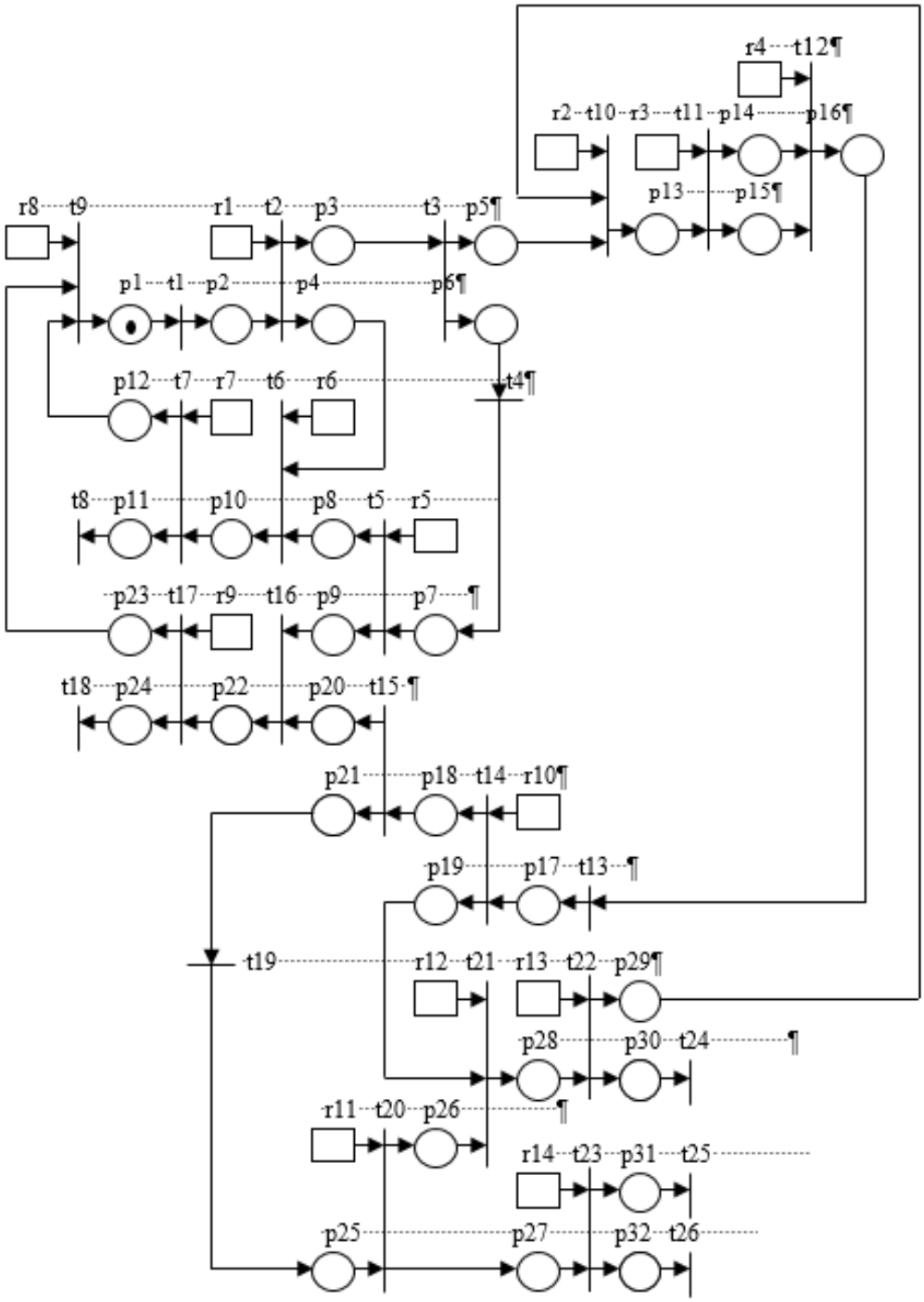


Рис. 2.2. Модель логічної структури протоколу EN_{PR}

2.3. Визначення термінального словника мови EN

Визначимо елементарні символи (термінали), що складають мову $EN \in T$:

– p_i – елементи множини P, що вказують на деяку визначену позицію;

– t_i – елементи множини T, що вказують на визначений перехід;

– a_i - елементи множини A, що вказують на визначений атрибут мітки;

– $M(p_i)$ – елементи множини M, що визначають наявність або відсутність мітки у позиції p_i;

– $A(p_i)$ – елементи множини A, що визначають вектор комірок пам'яті позиції p_i;

– $f(a_i)$ – правила перетворення атрибутів міток;

– +,- – операції додавання і видалення міток з позицій, а також перетворення атрибутів комірок пам'яті;

– $>, <, =, \neq, \geq, \leq$ – операції порівняння.

Для визначення терміналів (лексем) при вводі тексту, що описує модель на деякій вхідній мові, необхідно визначити правила формування лексем з множини вхідних елементів. Ці правила будуть застосовуватись на етапі лексичного аналізу вхідної мови.

2.4. Розробка лексичних граматик

Визначимо набір елементів, які формують термінальний словник і правила лексичного виведення терміналів мови EN:

1. $< N >$ – число;

$S \rightarrow DS$

$D \rightarrow$ '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'

$D \rightarrow \varepsilon$

де D – нетермінал, що позначає цифри, з яких складено число;

' ' – позначення (виділення) елементарного символу;

| – знак «або» (альтернатива);

ε - «порожній» символ.

$< N >$ має атрибут $N.atr$, який визначає значення числа.

2. $< U >$ - операції порівняння;

$S \rightarrow$ '>'|'<'|'='|'><'|'>='|'<='

3. <*Z*> - операції додавання/видалення;

S→ ’+’|’-’

Термінали <*U*> і <*Z*> атрибутів не мають

4. <*P*> – позиція;

S→ ’ p’ <N>

<*P*> має атрибут *P.atr,* що визначає номер позиції.

5. <*T*> - перехід;

S→ ’ t’ <N>

<*T*> володіє атрибутом *T.atr,* який визначає номер переходу.

6. <*AI*> - перехід;

S→ ’a’ <N>

<*AI*> містить атрибут *AI.atr,* що визначає номер комірки пам’яті (атрибуту мітки).

7. <*M*> - елемент розмітки;

S→ ’M(’ <N> ’)’ S

<*M*> володіє атрибутом *M.atr,* який містить номер позиції на множині розмітки мережі.

8. <*A*> - вектор комірок пам’яті мережі;

S→ ’A(’ <N> ’)’ S

<*A*> містить атрибут *A.atr,* що характеризує значення вектора комірок пам’яті на множині векторів комірок пам’яті мережі.

9. <*F*> - правило перетворення;

S→ ’ f ’ <N>

<*F*> включає атрибут *F.atr,* що призначений для зберігання номера правила перетворення.

Розроблені граматики синтаксичної структури представляються регулярними виразами. Звідси випливає, що на основі регулярних виразів можна побудувати скінченний автомат, який може транслювати мову, що задається цими

виразами. Побудуємо діаграму переходів такого автомата, що представлений на рис. 2.3.

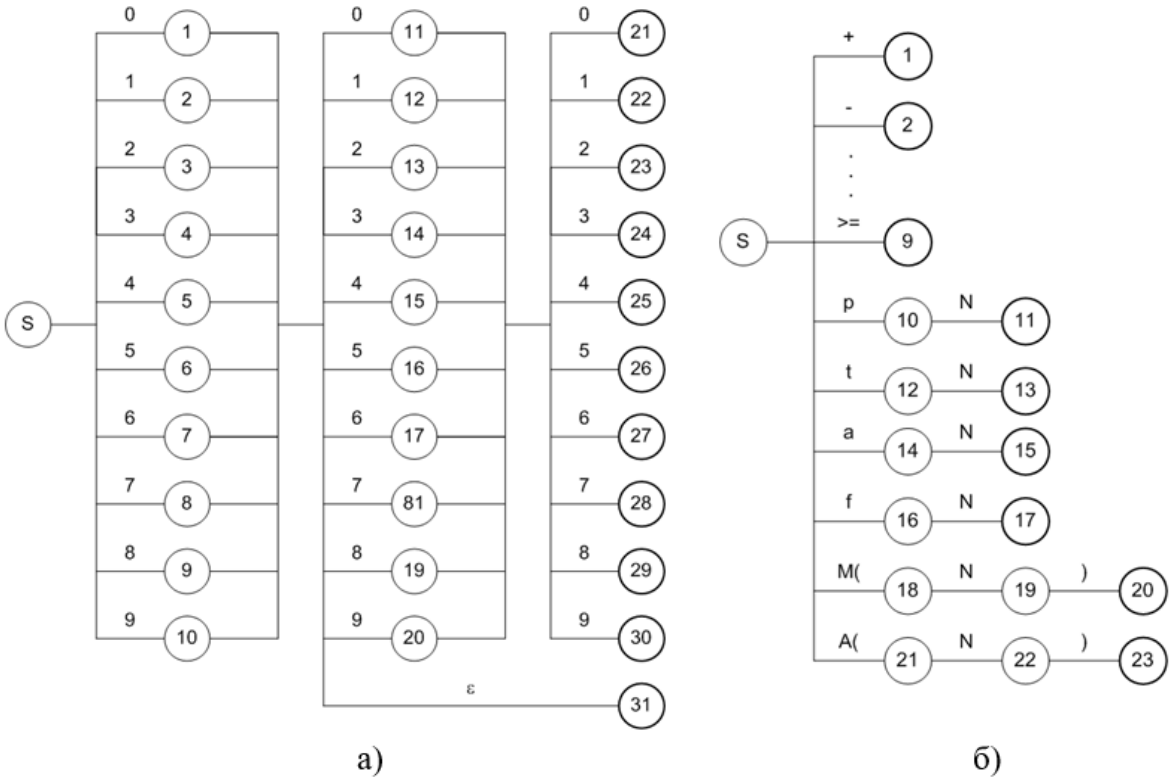
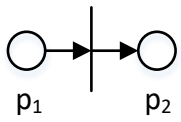


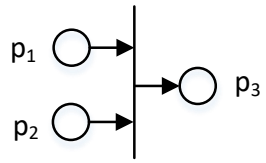
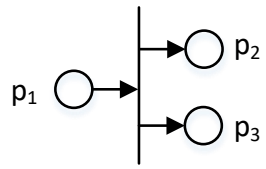
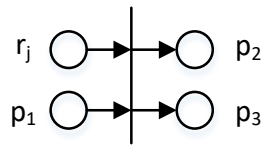
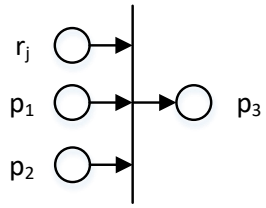
Рис. 2.3. Діаграма переходів скінченного автомата лексичного аналізу: а) для розбору граматики $G_{<N>}$, б) для повного лексичного аналізу мови EN

Таблиця 2.2

Типи переходів

Графічне позначення	Символьне позначення	Суть переходу
	$T(p_1, p_2)$	(1,0)’(0,1) Перехід активний, якщо позиція p_1 має мітку, а p_2 – вільна. Після спрацювання переходу з’являється мітка в позиції p_2 і видаляється з p_1

Продовження табл. 2.2

Графічне позначення	Символьне позначення	Суть переходу
	$I(p_1, p_2, p_3)$	(1,1,0)’(0,0,1) Перехід активний, якщо p_1 і p_2 мають мітки, а p_3 – вільна. Після спрацювання переходу з’являється мітка в p_3, мітки з p_1 і p_2 видаляються
	$F(p_1, p_2, p_3)$	(1,0,0)’(0,1,1) Перехід активний, якщо p_1 мають мітку, а p_2 і p_3 вільні. Після спрацювання переходу з’являються мітки в p_2 і p_3, з p_1 мітка видаляється
	$X(r_j, p_1, p_2, p_3)$	(0,1,0,0) ’(*,0,1,0); (1,1,0,0) ’(*,0,0,1) (0,1,1,0) ’(*,0,1,1); (0,1,0,1) ’(*,0,1,1) (1,1,1,0) ’(*,0,1,1); (1,1,0,1) ’(*,0,1,1) Якщо p_2 і p_3 вільні, а в p_1 є мітка, то на переході виникає конфліктна ситуація. Конфлікт вирішується за значенням позиції $M(r_j)$: 1. Якщо $M(r_j) = 0$, то мітка переходить з p_1 в p_2; 2. Якщо $M(r_j) = 1$, то мітка переходить з p_1 в p_3.
	$Y(r_j, p_1, p_2, p_3)$	(0,1,1,0)’(*,0,1,1); (1,1,1,0)’(*,1,0,1) (0,1,0,0)’(*,0,0,1); (0,0,1,0)’(*,0,0,1) (1,1,0,0)’(*,0,0,1); (1,0,1,0)’(*,0,0,1) Аналогічно до попереднього випадку

Значення $M(r_j)$ можна знайти у результаті застосування процедури $q(r_j) \in Q$. Суть цієї процедури полягає у виконанні перевірки значень предикатів $pr_1^{r(j)}$ і $pr_2^{r(j)}$.

При істинному значенні першого предиката – $M(r_j) = 1$, при істинності другого – $M(r_j) = 0$.

Якщо обидва предикати є хибними, то $M(r_j) = \infty$. У цьому випадку, позиція r_j блокує перехід до моменту результативного виконання процедури $q(r_j)$, що вимагає необхідності повторного визначення $M(r_j)$.

Якщо на переході не виникає конфліктної ситуації (вільна будь-яка вихідна позиція) і наявна мітка в позиції p_1, то перехід буде виконуватися по Т-схемі і по єдино можливому шляху, незалежно від того, яке значення при цьому має позиція r_j.

2.5. Розробка граматики транслятора

Для реалізації граматики трансляції мереж Петрі обрано одну з найбільш використовуваних мов програмування – С++, яка є лідером у застосуванні у сфері системного програмування.

Для представлення структури будь-якої програми за допомогою мови високого рівня програмування можна представити у вигляді сукупності блоків:

```
<заголовок програми>
<ініціалізація вхідних даних>
<оголошення необхідних об’єктів>
<змінні>
<масиви>
<структури>
<функції>
<ініціалізація початковими значеннями>
<блок операторів (тіло програми)>
< оператори завершення роботи програми>
```

Правила, які визначені граматикою трансляції, повинні передбачати операції заповнення відповідних блоків коду програми.

Початковий символ граматики визначає запис заголовку програми і оператора завершення:

$S \rightarrow$ *void main () { <лексема> }*

<лексема> $\rightarrow$ *<P>|<T>|<A>|<*M_0*>|<*G_0*>|<*t_i*>*

<P> $\rightarrow$ *int Np = P.atr*

Ініціалізація змінної *Np* (кількість позицій) та ініціалізація значення атрибуту *P.atr*.

<P> $\rightarrow$ *int masM [P.atr]*

Ініціалізація масиву *masM* для зберігання даних розмітки розмірністю *P.atr*.

<T> $\rightarrow$ *int Nt = T.atr*

Ініціалізація змінної *Nt* (кількість переходів) і присвоєння їй значення атрибута *T.atr.*

<A> $\rightarrow$ *int Na = A.atr*

Ініціалізація змінної *Na* (розмірність вектора комірок пам'яті) і присвоєння їй значення атрибута *A.atr.*

<A> $\rightarrow$ *int masA [A.atr] [P.atr]*

Ініціалізація двомірного масиву *masA* для зберігання значень векторів комірок пам'яті мережі розмірністю *P.atr*.

<M_0*>* $\rightarrow$ $\{ masM\ [N.atr]=1 \mid <M_0> \}_{N \in P}$

Заповнення початковими значеннями масиву поточної розмітки, значення *masM [i]=1* характеризує наявність мітки в позиції p_i.

<G_0*>* $\rightarrow$ $\{ masA\ [A.atr]\ [P.atr] = N.atr \mid <G_0> \}_{N \in P,\ ai \in P}$

Заповнення початковими значеннями масиву стану векторів комірок пам'яті у відповідності до значень *N.atr*.

<t_i*>* $\rightarrow$ `func_t T.atr () {`

`<затримка спрацювання переходу>`

`<перевірка умови>`

`<переміщення міток>`

```
<перетворення вектора пам'яті> }
```

Ініціалізація функції *func_t<i>*, що визначає повний набір правил функціонування переходу t_i.

```
<tᵢ>→ struct_t T.atr () {
int <X>        // початкові позиції переходу
int <Y>        // вихідні позиції переходу
int <TR>        // затримка спрацювання переходу
struct_R<rᵢ> () {
int <RI>            // умова спрацювання переходу
int <PM>   // видалити мітку з
int <AM> }     // додати мітку в
struct_RO<ρᵢ> () {
int <ROI>  // правило перетворення атрибутів мітки
int <C1>  // ліва частина правила
int <C2>  }   }         // права частина правила
```

Ініціалізація структури *struct_t<i>*, значення полів якої містять дані повного опису переходу t_i.

<t_i>→ if <умова спрацювання переходу> then func_t T.atr

Заповнення <тіла програми>. Стрічка коду описує виклик на виконання функції *func_t<i>* при істинності умови спрацювання переходу t_i.

Реалізація моделі формального опису в конкретному програмному чи програмно-апаратному середовищі є невід'ємною частиною процесу створення динамічних комп'ютерних систем.

2.6. Висновки до розділу

1. Обґрунтовано вибір класу мереж Петрі для моделювання статичної і динамічної компонент комп'ютерних систем, що дало змогу використати гібридні моделі на основі предикатів та часових мереж, які окрім візуалізації архітектури системи і її поведінки, дають змогу генерувати програмний код мовою С++.

2. Побудовано модель представлення комп’ютерної системи на основі EN-мереж Петрі, що враховує структуру системи, потенційні маршрути передачі даних, затримки маршрутизації і дає можливість швидко визначити «вузькі» місця системи перед початком її впровадження.

3. Визначено термінальний словник та розроблено лексичні граматики, які можна задавати регулярними виразами, що дає змогу побудувати скінченний автомат для трансляції елементів з однієї мови на іншу.

4. Запропоновано метод та процедури на основі правила лексичного аналізу вхідного тексту мови мережевих моделей EN і правила перетворення одержаних лексем в код програми на мові C++, що дає змогу формувати транслятор специфікації протоколів та одержувати модель динамічної комп’ютерної системи, аналізувати можливі стани системи та керувати параметрами задля оптимізації маршрутизації обміну даними між компонентами системи.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ НА ОСНОВІ МЕРЕЖ ПЕТРІ

3.1. Визначення функціональних вимог та побудова алгоритмів роботи програмного засобу побудови моделі мережі Петрі

Важливим аспектом розробки програмного засобу для моделювання комп'ютерних систем є побудова алгоритмів реалізації мережі Петрі на основі моделі і методу, які запропоновані у другому розділі. На рис. 3.1 наведено use case діаграму, яка дає змогу в подальшому врахувати особливості моделювання при реалізації мовою C++.

Рис. 3.1. Вимоги до програмного забезпечення трансляції мережі Петрі у програмний код на мові C++

Основними функціональними вимогами до програмного засобу є:

– забезпечення можливості вводу вхідних параметрів мережі Петрі на основі структури динамічної комп’ютерної системи;

– забезпечення можливості генерації програмного коду (трансляції) мовою С++ на основі параметрів, які введені природньої мовою;

– заповнення значень комірок пам’яті мережі Петрі;

– заповнення та виведення значень переходів у проектованій мережі;

– перевірка цілісності та повноти параметрів проектованої мережі;

– графічне відображення спроектованої мережі Петрі;

– генерація коду на С++ в окремому вікні.

Деталізована use case діаграма щодо побудови та налаштування параметрів моделі мережі Петрі спроектована та наведена на рис. 3.2.

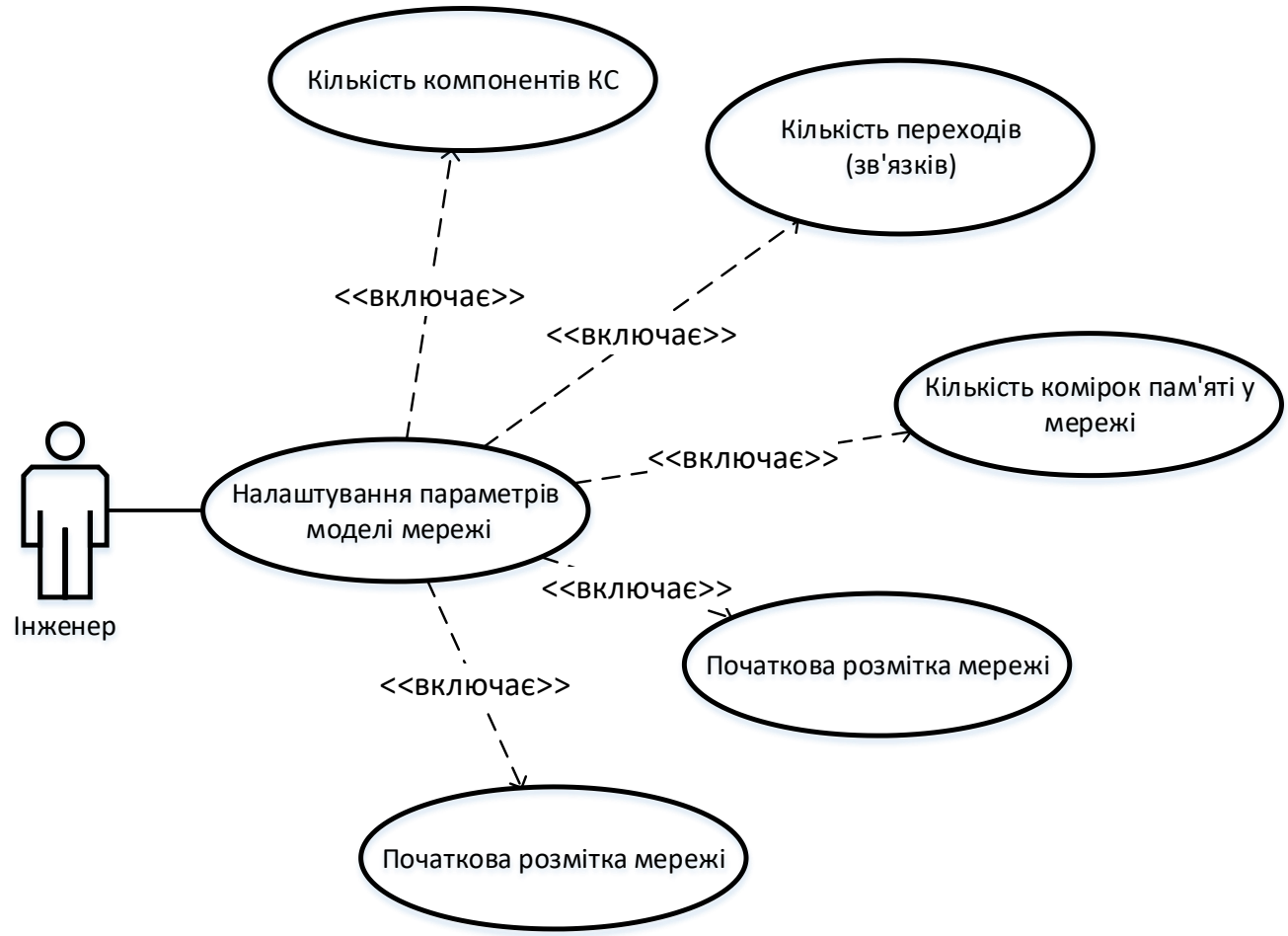


Рис. 3.2. Детальні вимоги до налаштування параметрів моделі мережі Петрі

Окрім загальних налаштувань моделі мережі Петрі необхідно деталізувати вимоги до переходів (зв’язків) між позиціями, які відображають компоненти комп’ютерних систем. Для цього побудовано use case діаграму, яка наведена на рис. 3.3.

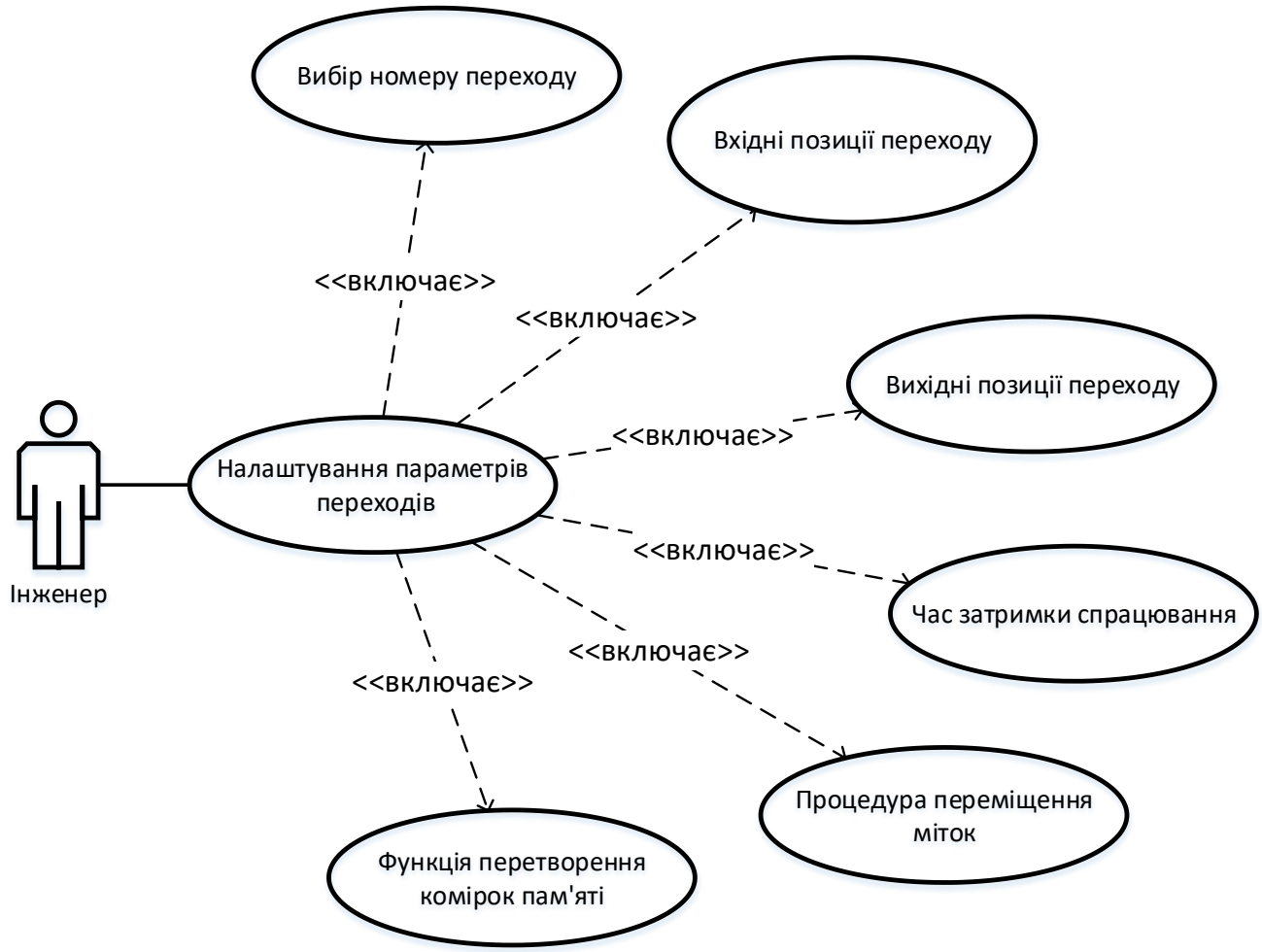


Рис. 3.3. Вимоги до налаштування переходів у мережі Петрі

Враховуючи особливості функціональних вимог до програмного засобу, розроблено алгоритм його роботи, який зображено на рис. 3.4. Оскільки, програмний засіб повинен забезпечувати трансляцію параметрів мережі Петрі, виражених природною мовою, у програмний код мови С++, то доцільним є застосування цієї мови програмування для забезпечення можливості побудови інтерактивних користувацьких інтерфейсів. При цьому пропонується використати середовище MS Visual Studio.

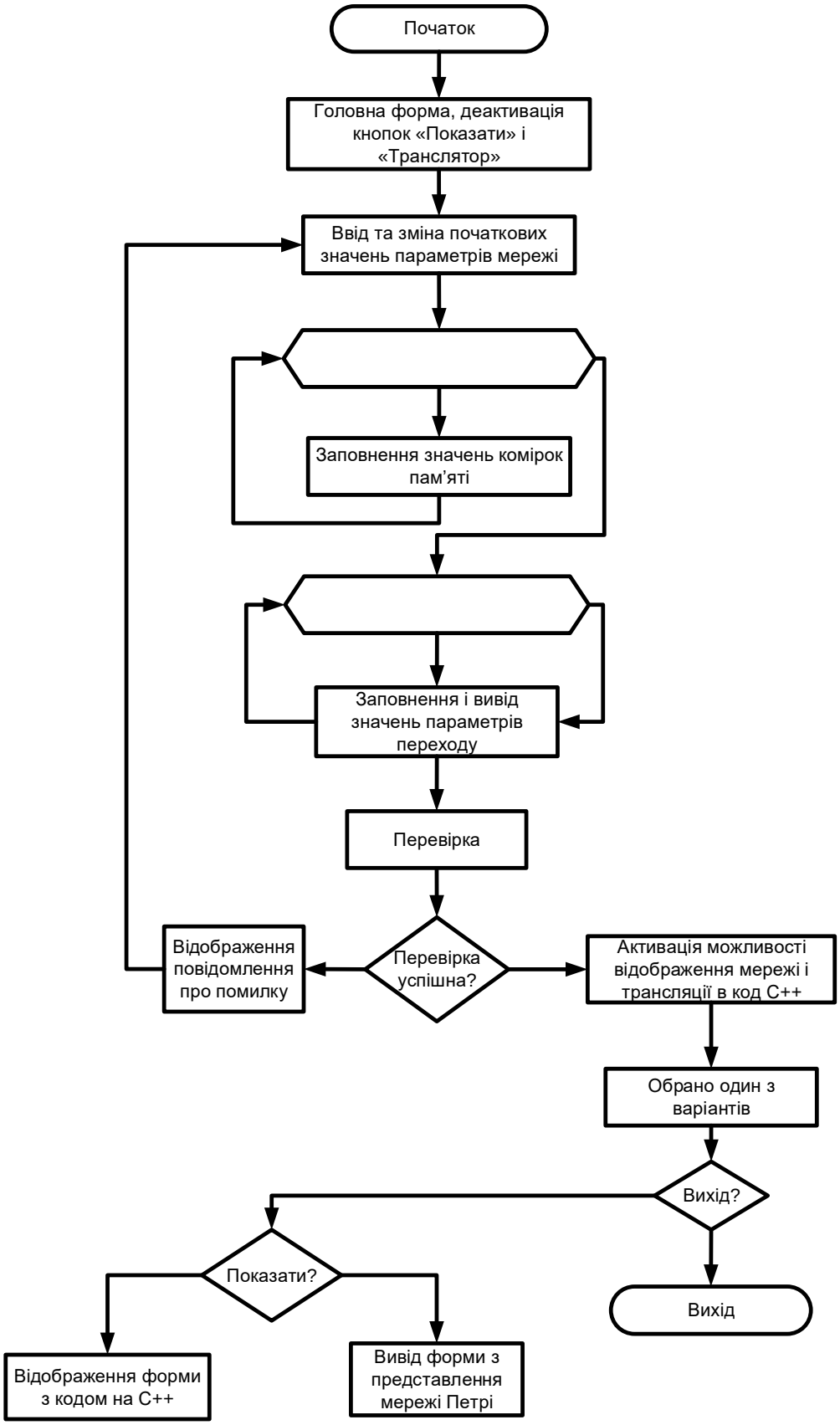


Рис. 3.4. Алгоритм моделювання комп'ютерних систем на основі мереж Петрі

Однак, одним з найскладніших етапів у розробці програмного засобу є динамічне промальовування (графічне представлення) мережі Петрі на основі вхідних параметрів. Для розв’язку цієї задачі запропоновано алгоритм, який наведено на рис.3.5.

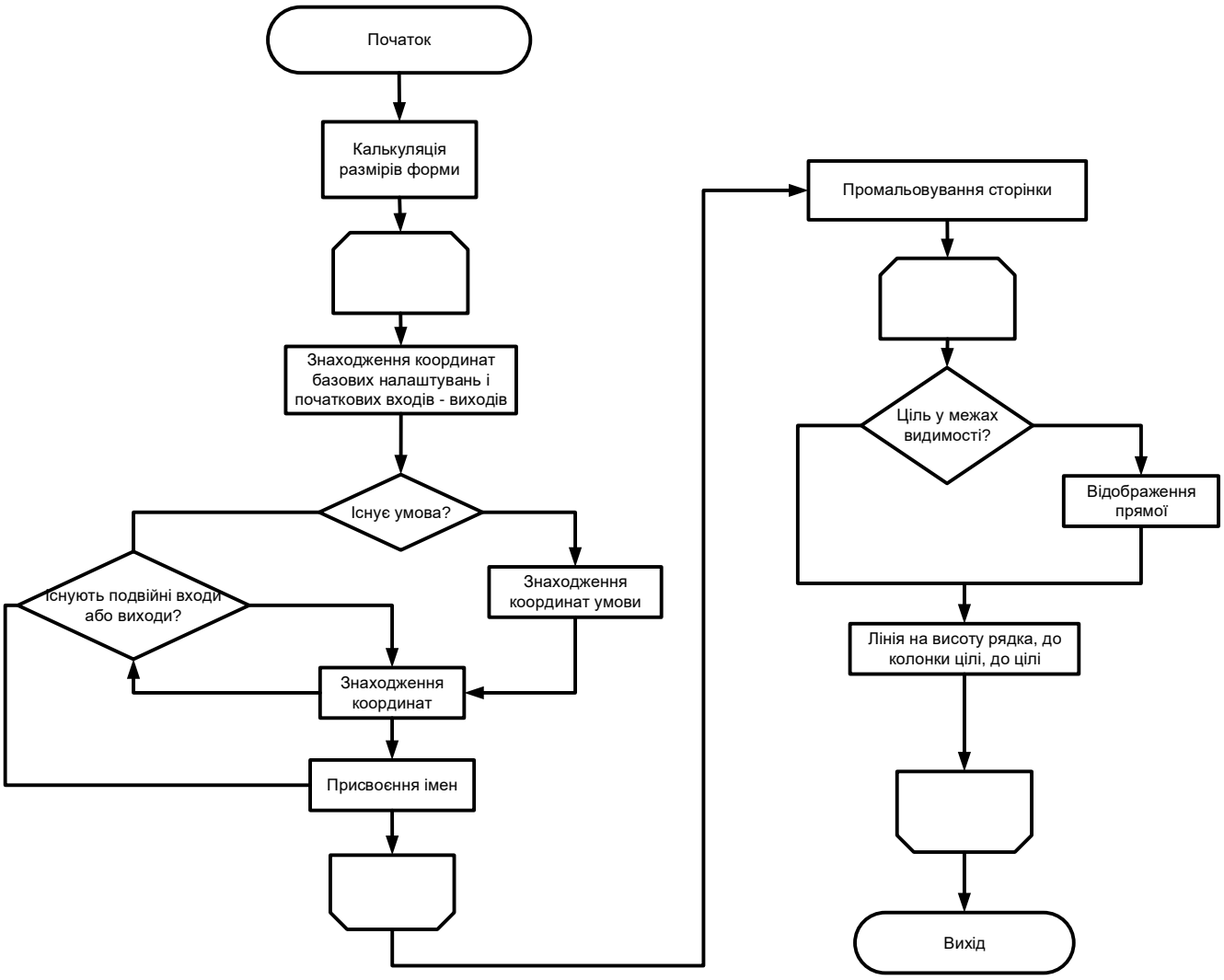


Рис. 3.5. Алгоритм графічного представлення комп’ютерних систем за допомогою мереж Петрі

Розробивши основну концепцію програмного засобу моделювання динамічних комп’ютерних систем на основі мереж Петрі, перейдемо до його безпосередньої реалізації, зокрема проектування схеми бази даних та архітектури програмного засобу.

3.2. Проектування схеми бази даних для збереження параметрів та версій мережі Петрі

Для того, щоб забезпечити гнучкість моделювання комп'ютерних систем, необхідно реалізувати базу даних, яка б зберігали об'єкти, що необхідні для побудови мереж Петрі та спрощення процесу трансляції цієї мережі у програмний код мовою С++.

Для зберігання об'єктів мережі Петрі пропонується скористатися реляційним представленням даних у середовищі MS SQL Server. У табл. 3.1 – 3.9 наведено структури таблиць бази даних та описано їхні атрибути і типи.

В загальному випадку, для опису параметрів мережі Петрі на основі яких буде відбуватися трансляція в програмний код мови С++, спроектовано табл. 3.1.

Таблиця 3.1

Схема «Petri_Model»

Атрибут	Тип	Примітка
ID_Model	int	Первинний ключ
Position_Count	int	
Cross_Count	int	
Description	varchar(150)	
ID_Version	int	Зовнішній ключ

Як видно з табл. 3.1, атрибут ID_Version є зовнішнім ключем на реляцію «Network_Version», що містить інформацію про версії моделювання мереж Петрі. У табл. 3.2 наведено схему «Network_Version».

Таблиця 3.2

Схема «Network_Version»

Атрибут	Тип	Примітка
ID_Version	int	Первинний ключ
Ver_Number	varchar(15)	
Description	varchar(150)	

Для представлення маркерів, їх типів та номерів спроектовано в середовищі MS SQL Server структуру «Marker», що наведена у табл. 3.3.

Таблиця 3.3

Схема «Marker»

Атрибут	Тип	Примітка
ID_Marker	int	Первинний ключ
Marker_Type	int	
Description	varchar(150)	

Для опису переходу, як окремого елементу, створено таблицю «Cross», атрибути якої наведено у табл. 3.4.

Таблиця 3.4

Схема «Cross»

Атрибут	Тип	Примітка
ID_Cross	int	Первинний ключ
Cross_Number	int	
Description	varchar(150)	

За зберігання позицій у мережі Петрі відповідає таблиця «Position», схема якої наведена у табл. 3.5.

Таблиця 3.5

Схема «Position»

Атрибут	Тип	Примітка
ID_Position	int	Первинний ключ
ID_Marker	int	Зовнішній ключ
Position_Number	int	
Description	varchar(150)	

Переходи між вершинами EN-мережі виконуються на основі істинності предикатів. Для зберігання та оперування предикатами створена таблиця «Predicate», структуру якої наведено у табл. 3.6.

Таблиця 3.6

Схема «Predicate»

Атрибут	Тип	Примітка
ID_Predicate	int	Первинний ключ
Condition	varchar(MAX)	
Description	varchar(150)	

Ефективність та оптимальність переходів забезпечуються за рахунок функцій переходів, які будуються на основі виконання операцій над предикатами. Тому сукупність таких функцій записують у таблицю «Cross_Func», атрибути якої приведені у табл. 3.7.

Таблиця 3.7

Схема «Cross_Func»

Атрибут	Тип	Примітка
ID_Func	int	Первинний ключ
Func_Contest	varchar(100)	
Description	varchar(150)	

У табл. 3.8 наведено схему реляції «Cross_Position» , яка є базовою при спрацюванні або не спрацюванні переходу. Атрибутами даної схеми є переважно зовнішні ключі з, практично усіх, раніше побудованих таблиць.

Таблиця 3.8

Схема «Cross_Position»

Атрибут	Тип	Примітка
ID_Cross_Position	int	Первинний ключ
Input_position	int	Зовнішній ключ
Output_position	int	Зовнішній ключ
ID_Predicate	int	Зовнішній ключ
ID_Func	int	Зовнішній ключ
ID_Version	int	Зовнішній ключ
Description	varchar(150)	
ID_Cross	int	Зовнішній ключ

Ще одна таблиця, яка зберігається у базі даних, відповідає за стани та версії мереж Петрі, її схему представлено у вигляді табл. 3.9.

Таблиця 3.8

Схема «Cross_Position»

Атрибут	Тип	Примітка
ID_Model_Cross	int	Первинний ключ
ID_Model	int	Зовнішній ключ
ID_Cross_Position	int	Зовнішній ключ

У результаті визначення схеми таблиць бази даних для оперування параметрами моделі та переходів EN-мережі Петрі, одержана ER-діаграма, яка наведена на рис. 3.6.

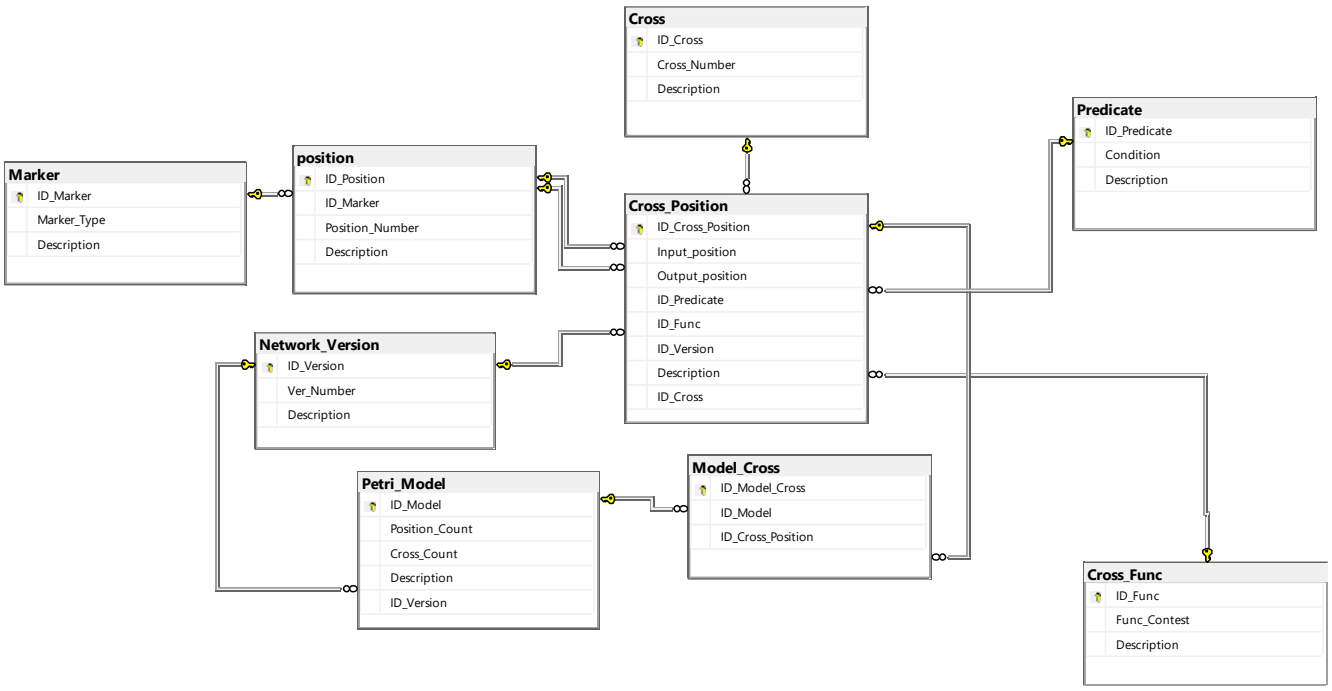


Рис. 3.6. ER-діаграма бази даних EN-мережі Петрі

ER-діаграма показує зв’язки між сутностями предметної області і є нормалізованою, що дає змогу ефективно будувати та контролювати версії комп’ютерних систем.

3.3. Реалізація програмного засобу реалізації, трансляції та відображення моделі мережі Петрі

Головною формою, яка з’являється при запуску програмного засобу, є форма, яка показана на рис. 3.7. Дана форма представляє собою інструмент для введення вхідних даних щодо моделі мережі Петрі, які відображають:

– кількість позицій, тобто кількість компонентів динамічної комп’ютерної системи;

– кількість переходів – відображається зв’язки між компонентами системи або шляхи комунікації між компонентами системи;

– кількість комірок пам’яті мережі Петрі – відображає усі маршрути обміну даними у динамічній комп’ютерній системі;

– початкова розмітка мережі – інтерпретує маркери у вершинах мережі Петрі

– початковий стан пам’яті мережі – відомі маршрути обміну даними у комп’ютерній системі.

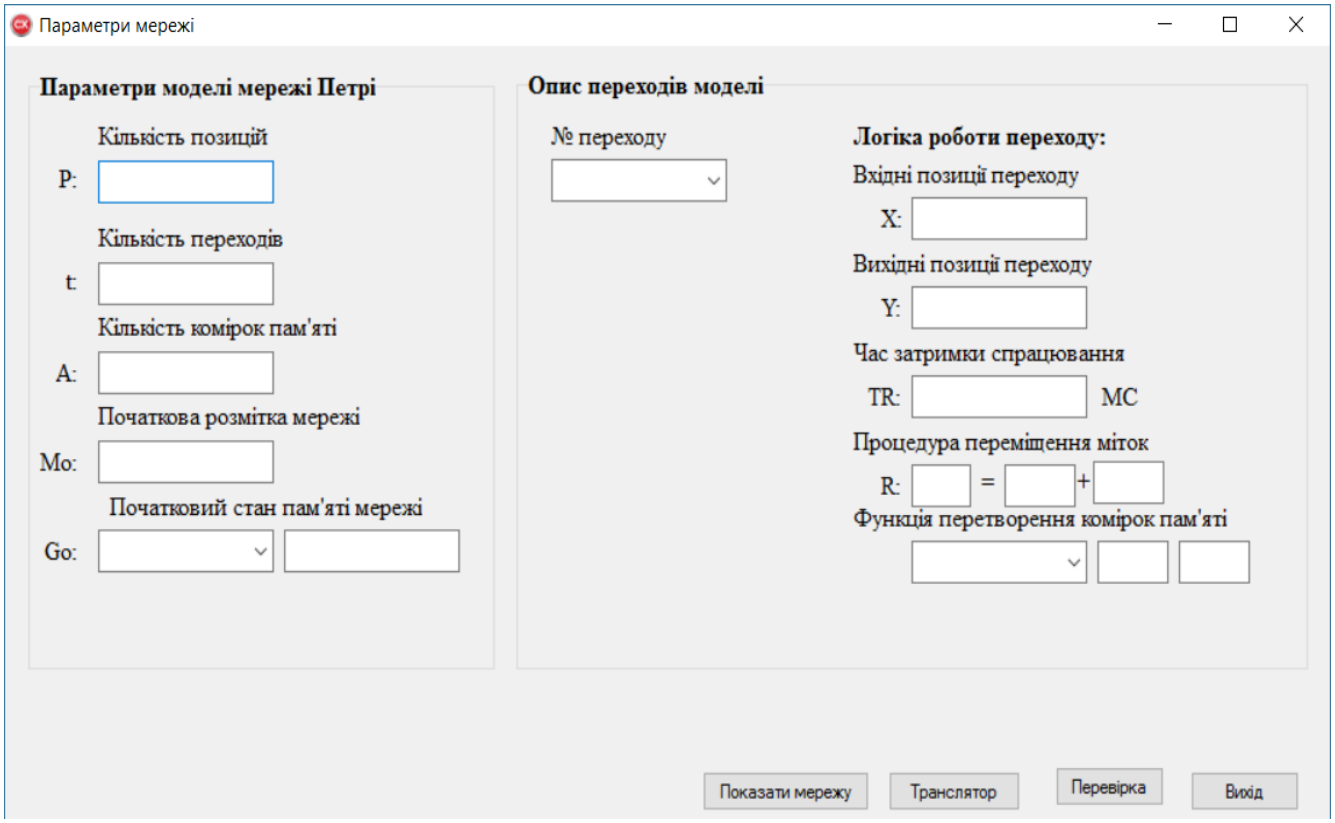


Рис. 3.7. Головна форма програмного засобу

Окрім цього, на формі наявні елементи налаштування переходів мережі:

– номер поточного переходу;

– вхідна позиція (вершина) переходу;

– вихідні позиції переходу;

– час затримки спрацювання переходів (мс);

– процедура переміщення маркерів по мережі;

– функція перетворення комірок пам’яті – сума, різниця, множення та ділення.

На рис. 3.8 наведено форму із заповненими параметрами моделі та переходів.

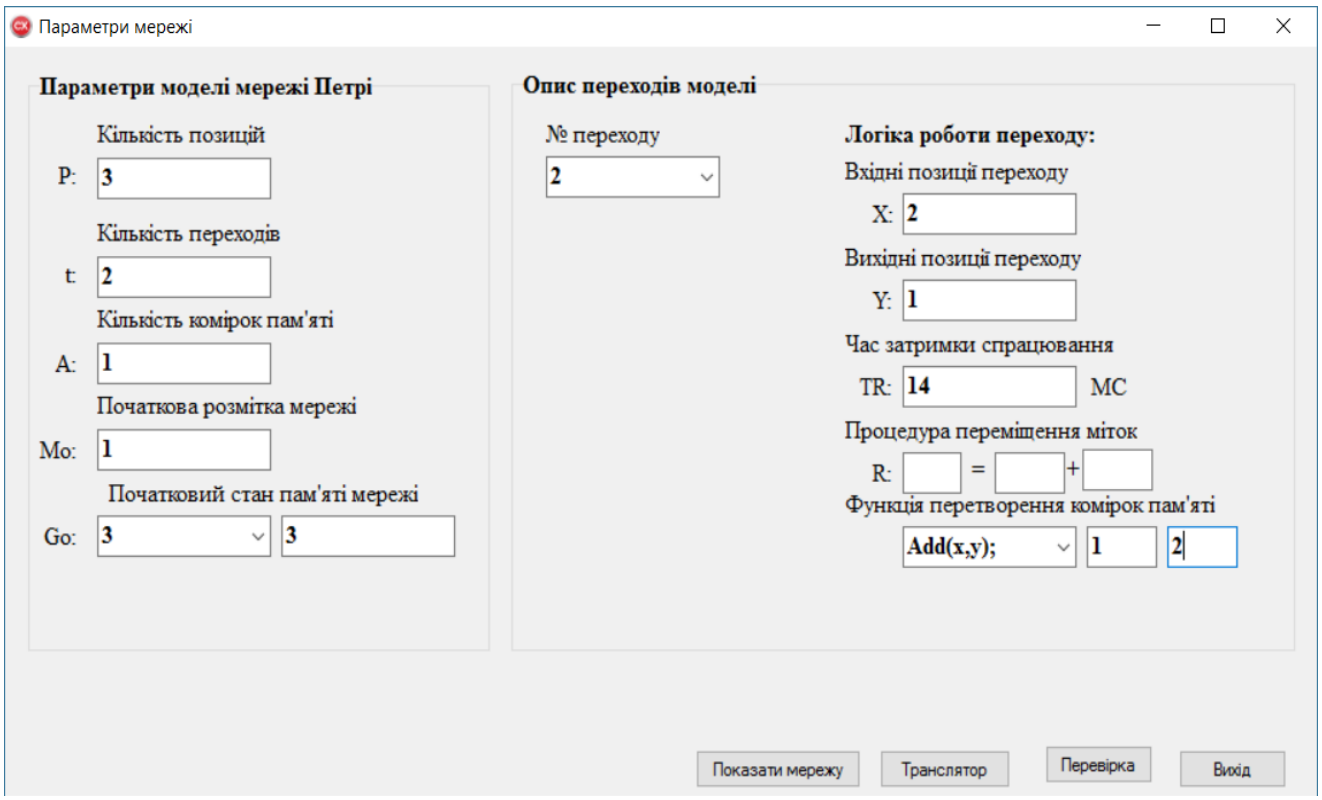


Рис. 3.8. Тестовий набір даних налаштувань моделі та переходів мережі Петрі

На формі налаштувань наявні чотири кнопки керування:

– показати мережу;

– транслятор;

– перевірка;

– вихід.

Натиснувши кнопку «Транслятор» з’являється вікно, яке показано на рис. 3.9. Дана форма містить код мовою C++, що фактично є генерацією параметрів введених природньою мовою.

Характерною особливістю генерації мереж Петрі з точки зору програмування є використання шаблону, який є постійним для усіх типів мереж незалежно від параметрів мережі.

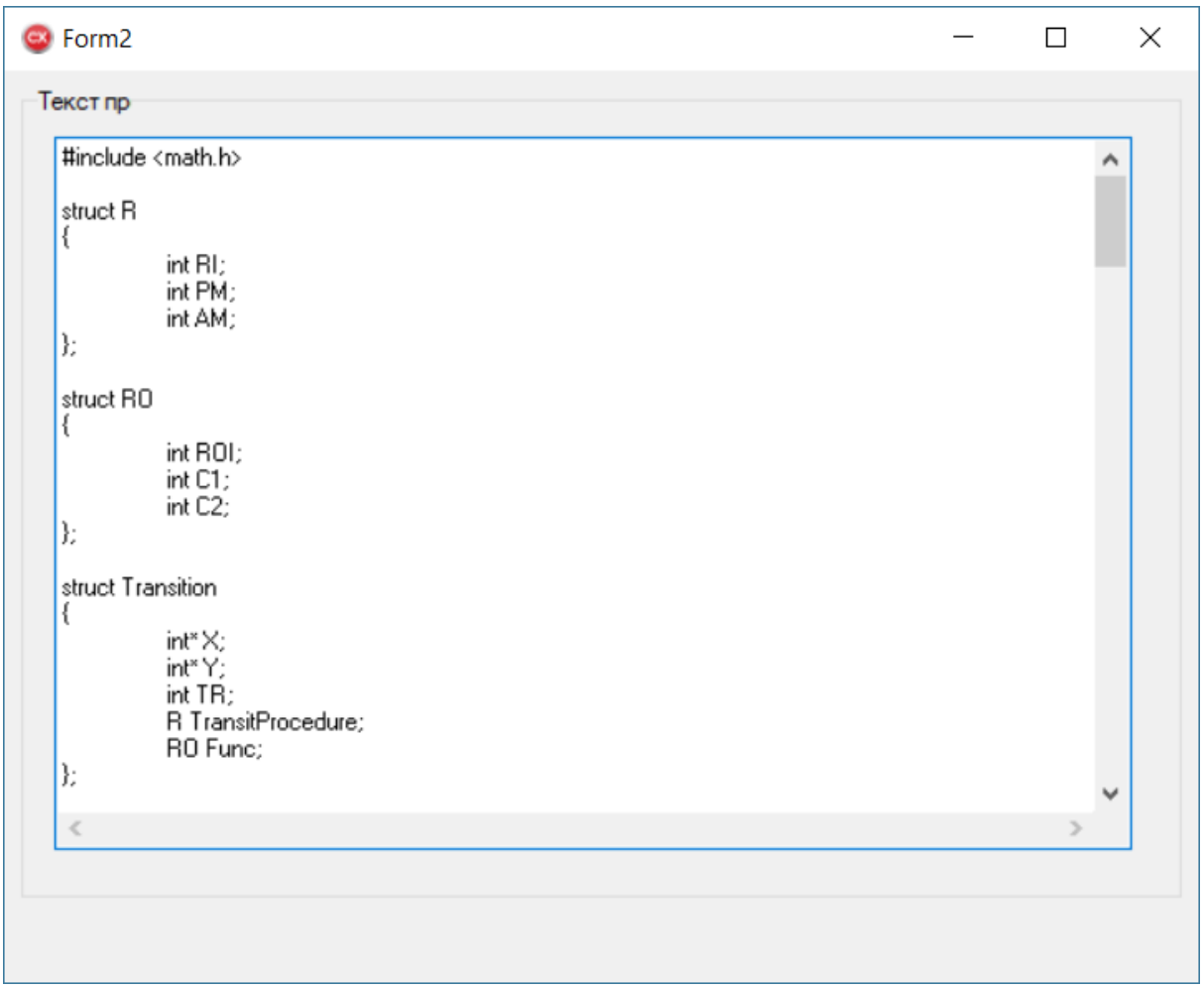


Рис. 3.9. Згенерований код формування мережі Петрі мовою C++

Стандартний фрагмент генерації мережі Петрі показано у лістингу 3.1.

Лістинг 3.1. Стандартна частина формування мережі Петрі

```
#include <math.h>

struct R
{
 int RI;
 int PM;
 int AM;
};

struct RO
{
 int ROI;
 int C1;
 int C2;
};
```

```
struct Transition
{
	int* X;
	int* Y;
	int TR;
	R TransitProcedure;
	RO Func;
};

void func_t(int* arrM,int* arrA, Transition* ti,int P, int A)
{
Sleep(ti->TR);
bool right=false;
for (int i=0;i<ti->X.Length();i++)
{
	if (ti->TransitProcedure.RI!=0 && *(arrM+ti->X[i])==1 &&
*(arrM+ti->X[i])==ti->TransitProcedure.RI)
	right=true;
}
if (right)
{
	*(arrM+ti->TransitProcedure.PM)=0;
	*(arrM+ti->TransitProcedure.AM)=1;
}
switch (ti->Func.ROI)
{
	case 1:
	{
	for (int m=0;m<(ti->X.Length());m++)
		for (int k=0;k<A;k++)
		{
			*(arrA+m*(ti->X[m])+k)+=ti->Func.C1;
		}
	for (int m=0;m<(ti->Y.Length());m++)
		for (int k=0;k<A;k++)
		{
			*(arrA+m*(ti->Y[m])+k)+=ti->Func.C2;
		}
break;
	}
	case 2:
	{
	for (int m=0;m<(ti->X.Length());m++)
		for (int k=0;k<A;k++)
		{
			*(arrA+m*(ti->X[m])+k)*=ti->Func.C1;
		}
	for (int m=0;m<(ti->Y.Length());m++)
		for (int k=0;k<A;k++)
		{
			*(arrA+m*(ti->Y[m])+k)*=ti->Func.C2;
		}
break;
```

```
      }
      case 3:
      {
      for (int m=0;m<(ti->X.Length());m++)
            for (int k=0;k<A;k++)
            {
                  *(arrA+m*(ti->X[m])+k)=(int)(*(arrA+m*(ti-
>X[m])+k)/ti->Func.C1);
            }
      for (int m=0;m<(ti->Y.Length());m++)
            for (int k=0;k<A;k++)
            {
                  *(arrA+m*(ti->Y[m])+k)=(int)(*(arrA+m*(ti-
>X[m])+k)/ti->Func.C2);
            }
     break;
      }
      case 4:
      {
      for (int m=0;m<(ti->X.Length());m++)
            for (int k=0;k<A;k++)
            {
                  *(arrA+m*(ti->X[m])+k)-=ti->Func.C1;
            }
      for (int m=0;m<(ti->Y.Length());m++)
            for (int k=0;k<A;k++)
            {
                  *(arrA+m*(ti->Y[m])+k)-=ti->Func.C2;
            }
     break;
      }
     }
     }
```

Код транслятора, що враховує особливості параметрів моделі та переходів мережі Петрі наведено у лістингу 3.2.

Лістинг 3.2. Фрагмент коду, що враховує параметри моделі і переходів мережі Петрі

```
int main(void)
{
int Np=4;// кількість позицій
int masM[4]={1,0,1,0}; // запис початкового стану мережі
int Nt=2;// кількість переходів
int Na=3;// кількість комірок пам'яті
int masA[4][3]= {
{2,4,6},
{4,6,8},
```

```
{6,12,24},
{8,16,32}
};// ініціалізація масиву комірок пам'яті
Transition Transits[2];// Створюється 2 переходи
Transits[0].X=new int[1];// Виділення місця для вхідних позицій
Transits[0].X[0]=1;// запис вхідних позицій
Transits[0].Y=new int[2];// Виділення місця для вихідних позицій
Transits[0].Y[0]=2;// Запис першої вихідної позиції
Transits[0].Y[1]=3;// Запис другої вихідної позиції
Transits[0].TR=14;// Запис затримки
R locR0;// Структура для умови переходу
locR0.RI=1;// Умова переходу
locR0.PM=1;// Откуда изъять
locR0.AM=3;// Куди перемістити
Transits[0].TransitProcedure=locR0;
RO locRO0;// Створення структури функції перетворення пам'яті
locRO0.ROI=3;// Правило перетворення
locRO0.C1=2;// Ліва частина
locRO0.C2=1;// Права частина
Transits[0].Func=locRO0;
func_t((int*)masM,(int*)masA,&Transits[0],Np,Na);
// Виклик функції переходу
Transits[1].X=new int[1];
Transits[1].X[0]=3;
Transits[1].Y=new int[2];
Transits[1].Y[0]=4;
Transits[1].Y[1]=1;
Transits[1].TR=15;

R locR1;
locR1.RI=0;
locR1.PM=;
locR1.AM=;

Transits[1].TransitProcedure=locR;

RO locRO1;
locRO1.ROI=2;
locRO1.C1=2;
locRO1.C2=6;

Transits[1].Func=locRO;

func_t((int*)masM,(int*)masA,&Transits[1],Np,Na);
}
```

Візуально мережу Петрі можна представити у формі, як показано на рис. 4.8.

Рис. 3.10. Графічне представлення мережі Петрі згідно параметрів

Ще один приклад налаштування параметрів моделі і переходів мережі Петрі, що представляє динамічну комп’ютерну систему, наведено на рис. 3.11, а графічне представлення мережі – на рис. 3.12.

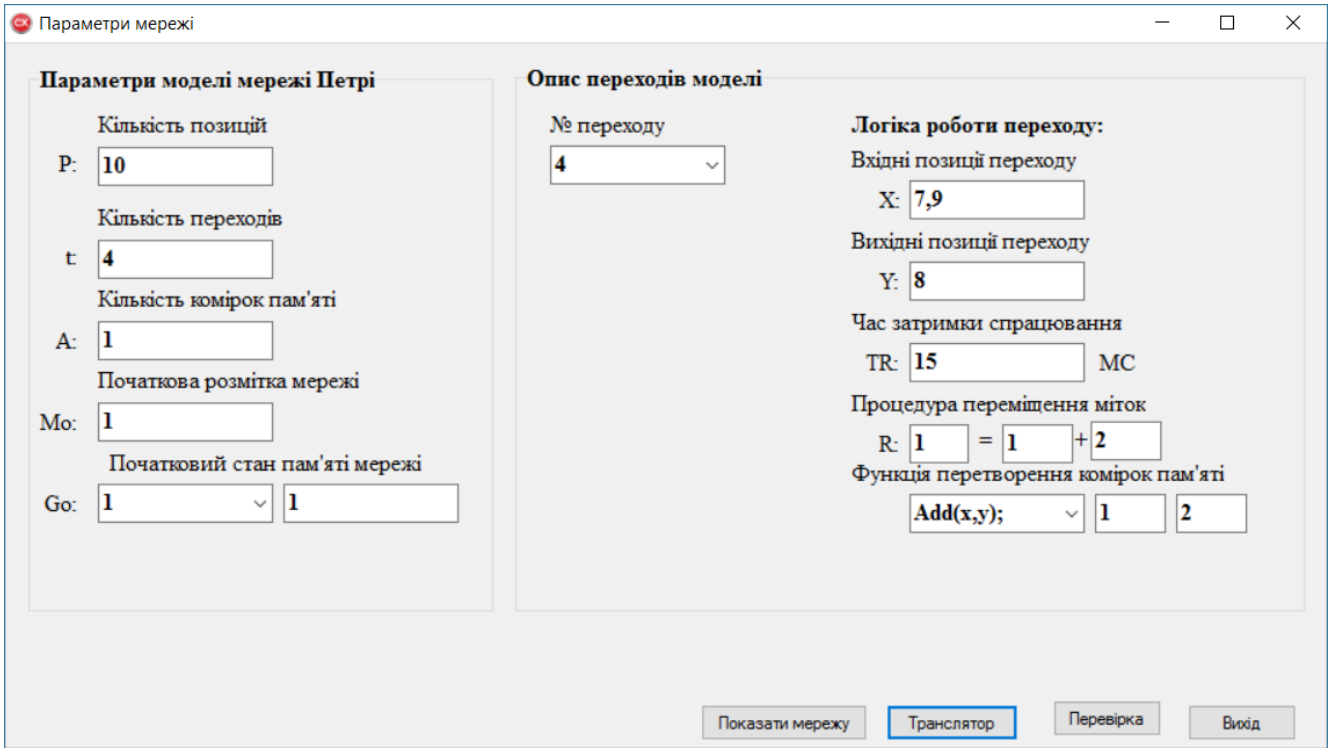


Рис. 3.11. Приклад налаштування параметрів мережі Петрі

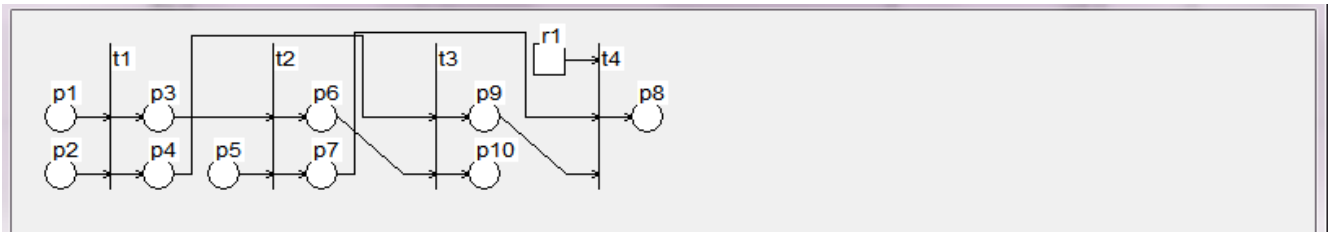


Рис. 3.12. Графічне представлення мережі Петрі

Таким чином, розроблено програмний засіб, що дозволяє моделювати динамічні комп’ютерні системи на основі мереж Петрі, транслювати налаштування моделі і переходів у програмний код на мові C++ та візуалізувати структуру системи.

3.4. Висновки до розділу

1. Із застосуванням UML діаграм визначено вимоги до функціональності програмного засобу моделювання комп’ютерних систем на основі EN мереж Петрі, що дало змогу в подальшому розробити алгоритми та користувацькі інтерфейси для представлення архітектури комп’ютерної системи і трансляції параметрів моделі у код на мові C++.

2. На основі реляційного підходу спроектовано та реалізовано у MS SQL Server 2019 базу даних для встановлення початкових умов моделі і переходів мережі Петрі.

3. За допомогою мови C++, середовища Visual Studio реалізовано програмний засіб моделювання комп’ютерних систем і трансляції структури мережі Петрі в програмний код мовою C++, що дало змогу одержати ефективний інструмент для аналізу зміни компонентів чи масштабованості комп’ютерних систем.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

У дипломній роботі магістра розроблено метод і програмний засіб моделювання комп’ютерних систем на основі мереж Петрі. Оскільки, моделювання систем проводиться на ПК, то важливим аспектом роботи користувача є його безпека. У зв’язку з цим, необхідно проаналізувати і врахувати вимоги і норми охорони праці, а також правила техніки безпеки при використанні електронно-обчислювальних засобів і периферійних пристроїв.

На сьогодні основним нормативним документом, який визначає і регламентує норми і правила експлуатації електронно-обчислювальної техніки є НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [26]. Правила встановлюють вимоги безпеки до обладнання робочих місць операторів ЕОМ.

НПАОП 0.00-7.15-18 є обов'язковим для виконання роботодавцями, операторами електронно-обчислювальних машин, операторами комп'ютерного набору, операторами комп'ютерної верстки та працівників інших професій, які у своїй роботі застосовують ЕОМ з ВДТ і ПП [26].

Згідно НПАОП 0.00-7.15-18 електронно-обчислювальні засоби повинні відповідати вимогам чинних в Україні стандартів і пройти державну санітарно-епідеміологічну експертизу у Порядку проведення державної санітарно-епідеміологічної експертизи.

З метою забезпечення електробезпеки користувачів ПК при використанні програмного засобу моделювання комп’ютерних систем на основі мереж Петрі необхідно, щоб комп’ютери і периферійні пристрої відповідали І-му класу захисту, або були заземленими відповідно до вимог НПАОП 40.1-1.32-01. Неприпустимим є використання клем функціонального заземлення для підключення захисного заземлення [27].

При організації робочих місць користувачів розробленої системи моделювання, необхідно забезпечити дотримання вимог щодо їх розташування, зокрема відстань робочого місця від стіни повинна складати 1м, а відстань між робочими місцями повинна становити 1,7 м. Площа, яка видіялється на одне робоче місце, обладнане ПК становить >=6.0 $м^2$, а об'єм – >=20 $м^3$ [26].

При виборі кімнат для розміщення робочих місць ПК враховано ступінь відбиття світла на екранах дисплеїв, яке проходить через вікна і яке може викликати значне осліплення в тих, хто сидить перед ними, особливо влітку та в сонячні дні. Тому, ПК і оргтехніка розміщені біля стін, які не знаходяться біля вікон або навпроти них [26].

Оскільки, при незадовільному освітленні знижується продуктивність праці користувачів ПК, і можливі негативні впливи на здоров'я такі, як короткозорість, швидка втомленість, тому всі приміщення, які облаштовані робочими місцями з ПК, мають природне і штучне освітлення. Не допускається розташування робочих місць з ПК в підвальних приміщеннях [26].

Робочі місця з ПК при виконанні творчої роботи, яка потребує значної розумової концентрації, зокрема при моделюванні комп'ютерних систем, ізольовано одне від одного перегородкою висотою 1,6 м [26]. Поверхня підлоги у приміщеннях повинна бути оздоблена керамічною плиткою і бути рівною та зручною для очищення та вологого прибирання.

Штучне освітлення у приміщеннях повинно бути виконано у вигляді комбінованої системи освітлення з використанням люмінесцентних джерел світла у світильниках загального освітлення, які розташовувати над робочими поверхнями у рівномірно-прямокутному порядку. Штучне освітлення забезпечує на робочих місцях з ПК освітленість 300 – 500 Лк [26].

Для запобігання засвітленню екранів ПК прямими світловими потоками лінії світильників розташовані з достатнім бічним зміщенням відносно рядів робочих місць, а також паралельно до світлових отворів. При цьому кожне вікно повинно мати світлорозсіюючі штори з коефіцієнтом відбивання 0,7 [26].

У приміщенні також необхідно забезпечити і природнє освітлення, при цьому

на кожному вікні закріплені жалюзі з вертикальними ламелями, що регулюються для зменшення прямого попадання сонячного світла на екран комп’ютерів.

З метою запобігання нещасним випадкам та організації охорони праці на виробництві розробляються інструкції з охорони праці і техніки безпеки при використанні комп’ютерної техніки. Дія інструкції поширюється на всі структурні підрозділи установи [27].

До роботи на ПК допускаються особи, які пройшли спеціальне навчання, медичне обстеження, вступний інструктаж з охорони праці, інструктаж на робочому місці та інструктаж з пожежної безпеки [27].

З ергономічної точки зору, при розташуванні елементів робочого місця враховано наступні фактори [26]:

– простір для розміщення користувача;

– можливість огляду елементів робочого місця;

– можливість огляду простору за межами робочого місця;

– можливість робити записи, розміщення документації і матеріалів, які використовує користувач.

При дослідженні та експлуатації програмного засобу моделювання комп’ютерних систем на основі мереж Петрі проаналізовано та враховано необхідні вимоги охорони праці і правила техніки безпеки, що дозволило забезпечити зручні умови для ефективної роботи користувачів комп’ютерів.

4.2. Забезпечення безпеки життєдіяльності при роботі з ПК

Безпека життєдіяльності при роботі з ПК передбачає виконання ряду вимог щодо захисту людини від негативного впливу компонентів комп’ютерної техніки і містить вимоги з електробезпеки, ергономічних вимог, пожежної безпеки та інших.

Заходи щодо усунення небезпеки ураження електричним струмом зводяться до правильного розміщення устаткування та електричних кабелів. Інші заходи щодо забезпечення електробезпеки, збігаються з загальними заходами пожежо- та електробезпеки [28].

В якості профілактичних заходів для забезпечення пожежної безпеки слід використовувати скриту електромережу, надійні розетки з пожежобезпечних матеріалів, силові мережі живлення устаткування виконувати кабелями, розрахованими на підключення в 3-5 разів більшого навантаження, включати й виключати живлення обладнання за допомогою штатних вимикачів [28]. Треба регулярно робити очистку внутрішніх частин комп'ютерів, іншого устаткування від пилу, розташовувати комп'ютери на окремих неспалюваних столах. Для запобігання іскріння необхідно рідше встромляти і виймати штепсельні вилки з розеток [28].

Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду. Якщо він розташований під кутом, то стає причиною сутулості. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Перед екраном монітора, особливо старих типів, повинен бути спеціальний захисний екран. При його відсутності треба сидіти на відстані витягнутої руки від монітора. Ще одним моментом, який стосується зору, є необхідність створення неоднорідного поля зору [28].

Важливою є форма спинки крісла, яка повинна повторювати форму спини. Висота крісла повинна бути такою, щоб користувач не почував тиску на куприк або стегна. Крісло бажано обладнати бильцями. Його потрібно встановити так, щоб не треба було тягтися до клавіатури. Періодично користувачу необхідно рухатися, вчасно змінювати положення тіла і робити перерви у роботі [28].

При напруженій роботі за комп'ютером щогодини необхідно робити перерву на 15 хвилин через кожну годину і треба займатися іншою справою. Декілька разів на годину бажано виконувати серію легких вправ для розслаблення.

Наслідками регулярної роботи з комп'ютером без застосування захисних засобів можуть бути: захворювання органів зору (60% користувачів); хвороби серцево-судинної системи (20%); захворювання шлунково-кишкового тракту (10%); шкірні захворювання (5%); різноманітні пухлини.

Режим праці та відпочинку при роботі з персональною електронно-обчислювальною машиною (ПЕОМ) залежить від категорії трудової діяльності. Всі

роботи з ПЕОМ ділять на три категорії. Перша - епізодичне зчитування і робота з інформацією не більше 2-х годин за 8-годинну робочу зміну. Друга - зчитування інформації або творча робота не більше 4-х годин за восьми годинну зміну. Третя - зчитування інформації або творча робота тривалістю більше 4-х годин за зміну [28].

Якщо у приміщенні експлуатується більше одного комп'ютера, то треба врахувати, що на одного користувача можуть впливати випромінювання від інших, в першу чергу бокових, а також і задньої стінки сусіднього дисплея. Тому необхідний захист спеціальними фільтрами і щоб користувач розміщався від бічних і задніх стінок інших дисплеїв на відстані не ближче одного метра [28].

Всесвітня організація охорони здоров'я (ВООЗ) роботу з персональним комп'ютером віднесла до небезпечних, бо їй притаманний фактор постійно діючого стресу. Через це небезпеці піддаються всі життєво важливі органи людини, з'являється ризик виникнення серйозних хвороб [28].

Електромагнітні поля комп'ютерної техніки, особливо низькочастотні, негативно впливають на людину і в першу чергу на її центральну нервову систему, викликаючи головний біль, запаморочення, нудоту, депресію, безсоння, відсутність апетиту, виникнення синдрому стресу. Причому нервова система реагує навіть на короткі за тривалістю впливи слабких полів: змінюється гормональний стан організму, порушуються біоструми мозку. Це призводить до погіршення зору, ускладненню серцево-судинних захворювань і зниження імунітету.

Характерною рисою професії оператора ПК є статичний режим роботи: великий обсяг праці треба виконувати в сидячому положенні. При цьому більшість груп м'язів постійно напружені, що призводить до швидкої стомлюваності, сприяє розвитку фахових патологічних вигинів хребта: грудному гіперкифозу, сплощенню шийного лордозу і формуванню сколіозів [28].

Неправильне розташування дисплеїв по висоті - занадто низьке або високе, під неправильним кутом - є головною причиною появи сутулості. Занадто високе розташування дисплея призводить до тривалої напруги шийного відділу хребта,

що, зрештою, може призвести до розвитку остеохондрозу. Ненормальний стан хребта може стати причиною захворювання всього організму [28].

Висновки.

Отже, щоб запобігти негативним впливам необхідно знати небезпечні сторони комп'ютерної техніки і правила безпечної роботи з ними, вміти використовувати засоби запобігання небезпекам. Негативні фактори перед усім пов’язані із загально відомими небезпечними чинниками – ураження електричним струмом, пожежонебезпечністю, шумом та вібрацією.

ВИСНОВКИ

У дипломній роботі досягнуто поставленої мети та розв’язано наступні задачі, зокрема:

1. Проаналізовано особливості та класифікацію комп’ютерних систем, підходи до їхнього проектування, що дало змогу обґрунтувати актуальність та необхідність моделювання як їхньої структури, так і функціональної поведінки, заважаючи на сучасні вимоги до масштабованості і гнучкості комп’ютерних систем, а також витрати на прототипування.

2. Проведено аналіз специфіки процесу моделювання комп’ютерних систем у результаті якого встановлено, що більшість сучасних систем представляються у вигляді імітаційних моделей, що є фізичними динамічними структурами з дискретними або стохастичними станами.

3. Побудовано модель представлення комп’ютерної системи на основі EN-мереж Петрі, що враховує структуру системи, потенційні маршрути передачі даних, затримки маршрутизації і дає можливість швидко визначити «вузькі» місця системи перед початком її проектування та впровадження.

4. Визначено термінальний словник та розроблено лексичні граматики, які можна задавати регулярними виразами, що дає змогу побудувати скінченний автомат для трансляції елементів з однієї мови на іншу.

5. Запропоновано метод та процедури на основі правила лексичного аналізу вхідного тексту мови мережевих моделей EN і правила перетворення одержаних лексем в код програми на мові C++, що дає змогу формувати транслятор специфікації протоколів та одержувати модель динамічної комп’ютерної системи, аналізувати можливі стани системи та керувати параметрами задля оптимізації маршрутизації обміну даними між компонентами системи.

6. Із застосуванням UML діаграм визначено вимоги до функціональності програмного засобу моделювання комп’ютерних систем на основі EN мереж Петрі, що дало змогу в подальшому розробити алгоритми та користувацькі інтерфейси

для представлення архітектури комп’ютерної системи і трансляції параметрів моделі у код на мові C++.

7. За допомогою мови C++, середовища Visual Studio реалізовано програмний засіб моделювання комп’ютерних систем і трансляції структури мережі Петрі в програмний код мовою C++, що дало змогу одержати ефективний інструмент для аналізу зміни компонентів чи масштабованості комп’ютерних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проститенко, О.В., Халимон В.И., Рогов А.Ю. Моделирование дискретных систем на основе сетей Петри: учебное пособие – СПб.: СПбГТИ(ТУ), 2017. 69 с.

2. Васильев В.В., Кузьмук В.В. Сети Петри, параллельные алгоритмы и модели, мультипроцессорных систем. Киев: Наук. думка, 1990. 216 с.

3. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. М.: Энергоатомиздат, 1989. 328 с.

4. Томашевський В. М. Моделювання систем / В. М. Томашевський. – Київ : Видавнича група BHV, 2005. – 352 с.

5. Лоу М., Кельтон В. Д. Имитационное моделирование. Классика CS : Пер. с англ. / Аверилл М. Лоу, В. Дэвид Кельтон. – 3-е изд. – Киев : Издательская группа BHV, 2004. 847 с.

6. 3. Акопов А. С. Имитационное моделирование: учебник и практикум для академического бакалавриата. Москва : Издательство Юрайт, 2014. 389 с.

7. Анищенко В. С. Знакомство с нелинейной динамикой: Лекции Соросовского профессора. Москва : Издательская группа URSS, 2008. 224 с.

8. Banks J., Carson J. S., Nelson B. L., Nicol D. M. Discrete-Event System Simulation. New Jersey : Prentice-Hall, 2005. 528 p.

9. 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.

10. Dehouck R. Craft AI. The maturity of visual programming [Електронний ресурс]. Режим доступу до ресурсу – http://www.craft.ai/blog/thematurity-of-visual-programming/ (дата звернення 20.11.2020 р)

11. 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.

12. Beucher O. Introduction to MATLAB and SIMULINK: A Project Approach 3rd ed. Hingham : Infinity Science Press LLC, 2007. 386 p.

13. Colvin R., Hayes Ian J., Colvin R. A semantics for Behavior Trees. ACCS Technical Report ACCS-TR-07-01, ARC Centre for Complex Systems. 2007. 20 p.

14. Garrido J. M. Object-Oriented Discrete-Event Simulation with Java: A Practical Introduction. New York : Kluwer/Plenum, 2001. 255 p.

15. Строгалев В. П. Имитационное моделирование. Москва : Издательство МГТУ им. Н. Э. Баумана, 2015. 295 с.

16. Kelton W. D., Sadowski R. P., Sadowski D. A. Simulation with Arena. Boston : McGraw-Hill, 1998. 547 p.

17. Altiok T., Melamed B. Simulation Modeling and Analysis with ARENA. Burlington : Elsevier, 2007. 458 p.

18. Технології проектування програмного забезпечення URL: http://www.iasa.com.ua/studentam/study-materials-ua/testing/d456dkovska-m-v-lekc456ya-1-zhitt454v456-cikli-rozrobki-programnogo-zabezpechennya (дата звернення 10.10.2020 р.).

19. Ларман К. Применение UML и шаблонов проектирования. М.: Вильямс. 2001. 496 с.

20. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Київ. 2018.

21. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп’ютинґу. Львів: Магнолія-2006. 2012. 544 с.

22. Желібо Е.Н. Безпека життєдіяльності: Навчальний посібникКиїв: «Каравела», Львів: «Новий світ - 2000», 2001. 320с.

Додаток А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Геднмінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том II

IX Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів.
Актуальні задачі сучасних технологій – Тернопіль 25-26 листопада 2020.

УДК 004.94

А.М. Луцків, канд. техн. наук, доцент, М.В. Ващук
Тернопільський національний технічний університет імені Івана Пулюя, Україна

МЕРЕЖІ ПЕТРІ ЯК МЕТОД МОДЕЛЮВАННЯ ДИНАМІЧНИХ КОМП'ЮТЕРНИХ СИСТЕМ

A.M. Lutskiv PhD., Assoc. Prof., M.V. Vashschuk

PETRI NETWORKS AS A METHOD OF DYNAMIC COMPUTER SYSTEMS SIMULATION

Теперішній розвиток методів, методологій та інструментальних засобів проектування комп'ютерних систем (КС) вимагає залучення значних фінансових та людських ресурсів, які покликані забезпечити реалізацію функціонально-повних та зручних у використанні програмно-апаратних комплексів у відповідності до потреб замовників чи «стейкхолдерів».

Одним з процесів, що дає змогу знизити рівень витрат та ризиків у загальному процесі розробки та впровадження КС чи їх компонентів, є процес моделювання архітектури майбутньої системи. Модель архітектури КС забезпечує можливість більш чітко зрозуміти основні сутності предметної області та зв'язки між ними, а графічне представлення архітектури – визначити потенційно «вузькі» місця та можливості щодо гнучкості, масштабування та внесення змін у систему.

Динамічні комп'ютерні системи характеризуються здатністю перетворення та обробки інформації і, на відміну від інших систем, основою їхнього функціонування є цифрові канали та засоби передачі даних, а також здатність до зміни своєї структури шляхом масштабування та/або переналаштування параметрів взаємодії між структурними компонентами.

Для представлення структури комп'ютерної системи можна скористатися описом моделі скінченої мережі Петрі у вигляді:

$$EN = < P, T, F, M, G > \quad (1)$$

де $P = \{p_1, p_2, ..., p_n, r_1, r_2, ..., r_k\}$ – скінченна множина позицій, що включає підмножини простих позицій p і дозволяються позиції r;

$T = \{t_1, t_2, ..., t_l\}$ – скінчена множина переходів;

$F \subseteq \{I:P \times T \to (0,1)\} \cup \{0:T \times P \to (0,1)\}$ – множина потокових відношень, що містить вхідні і вихідні позиції для кожного переходу;

$M:P \to (0, 1, 2 ...)$ – функція розмітки, що визначає маркування позицій у вигляді додатних цілих чисел;

$G:P \to Y$ – функція стану пам'яті мережі, що описує стани векторів комірок пам'яті мережі для кожного варіанту розмітки через вектори пам'яті позицій, помічених мітками з визначеними атрибутами.

Формально функціонування мережевої моделі EN описується виразами зміни маркування (2) і змінами стану пам'яті мережі (3):

$$\forall p \in P, M'(p) = M(p) - I(p,t) + Q(t,p) \quad (2)$$

$$\forall p \in P, G'(p) = G(p) - I(p,t)A(p) + Q(t,p)\Pi(A) \quad (3)$$

де $\Pi(A) = \{\mu_0, \mu_1, ..., \mu_i\}$ – процедури перетворення атрибутів міток для кожного з i переходів мережі;

$A(p)$ – вектор атрибутів мітки, що знаходиться у позиції p.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004.94

А.М. Луцків, канд. техн. наук, доцент, М.В. Ващук
(Тернопільський національний технічний університет імені Івана Пулюя)

ГРАМАТИКА ПЕРЕТВОРЕННЯ ПАРАМЕТРІВ МОДЕЛІ МЕРЕЖІ ПЕТРІ У ПРОГРАМНИЙ КОД МОВИ С++

UDC 004.94

A.M. Lutskiv PhD., Assoc. Prof., M.V. Vashschuk

GRAMMAR OF PETRI NETWORK MODEL PARAMETERS TRANSFORMATION IN C ++

Мережі Петрі можуть застосовуватись для того, щоб забезпечити ефективність моделювання паралельних асинхронних процесів, які протікають у динамічних комп'ютерних системах та здійснити алгоритмічний опис об'єкту. Мережі Петрі, з точки зору їх імплементації, поділяють на три основні класи, що відрізняються між собою принципами організації позицій та переходів у мережі.

До цих трьох класів входять:

– безпечні МП – мережі, у вершинах (позиціях) яких при виконанні переходу(ів) може бути відсутня або наявна лише одна мітка;

– обмежені МП – мережі, в яких у вершинах міститься цілочисельне значення міток, а дуги, у вигляді цілих чисел, визначають кількісний їх розподіл у мережі після того, як вони пройшли через переходи (перехід спрацював);

– Е-мережі Петрі – тип мереж, де переходи можуть належати до кількох наперед визначених видів, і спрацювання яких відбувається лише за наявності визначеної кількості міток у мережі, а складні макропереходи здатні до зміни своєї структури.

Для реалізації граматики трансляції мереж Петрі обрано одну з найбільш використовуваних мов програмування – С++, яка є лідером у застосуванні у сфері системного програмування.

Для представлення структури будь-якої програми за допомогою мови високого рівня програмування можна представити у вигляді сукупності блоків:

<заголовок програми>
<ініціалізація вхідних даних>
<оголошення необхідних об'єктів>
<змінні>
<масиви>
<структури>
<функції>
<ініціалізація початковими значеннями>
<блок операторів (тіло програми)>
< оператори завершення роботи програми>

Правила, які визначені граматикою трансляції, повинні передбачати операції заповнення відповідних блоків коду програми.

Початковий символ граматики визначає запис заголовку програми і оператора завершення:

$S \rightarrow$ *void main () { <лексема> }*

<лексема> $\rightarrow$ *<P>|<T>|<A>|<*M_0*>|<*G_0*>|<*t_i*>*

<P> $\rightarrow$ *int Np = P.atr*

Ініціалізація змінної *Np* (кількість позицій) та ініціалізація значення атрибуту *P.atr*.

<P> $\rightarrow$ *int masM [P.atr]*

Ініціалізація масиву *masM* для зберігання даних розмітки розмірністю *P.atr*.

<T> $\rightarrow$ *int Nt = T.atr*

Додаток Б

Код створення бази даних

```
CREATE DATABASE PETRI_NETWORK
Create table Network_Version
(
ID_Version int primary key identity (1,1),
Ver_Number varchar (15),
[Description] varchar(150)
)

Create table Marker
(
ID_Marker int primary key identity (1,1),
Marker_Type int,
[Description] varchar(150)
)

Create table [Predicate]
(
ID_Predicate int primary key identity (1,1),
Condition varchar (MAX),
[Description] varchar(150)
)

Create table position
(
ID_Position int primary key identity (1,1),
ID_Marker int foreign key references Marker(ID_Marker),
Position_Number int,
[Description] varchar(150)
)

Create table [Cross]
(
```

```
ID_Cross int primary key identity (1,1),
Cross_Number int,
[Description] varchar(150)
)

Create table Cross_Func
(
ID_Func int primary key identity (1,1),
Func_Contest varchar (100),
[Description] varchar(150)
)

Create table Petri_Model
(
ID_Model int primary key identity (1,1),
Position_Count int,
Cross_Count int,
[Version] varchar (15),
[Description] varchar(150)
)

Create table Croos_Position
(
ID_Cross_Position int primary key identity (1,1),
Input_position int foreign key references Position(ID_Position),
Output_position int foreign key references Position(ID_Position),
ID_Predicate int foreign key references
[Predicate](ID_Predicate),
ID_Func int foreign key references Cross_Func(ID_Func),
ID_Version int foreign key references
Network_Version(ID_Version),
[Description] varchar(150)
)
alter table Petri_Model add foreign key (ID_Version) references
Network_Version (ID_Version)
```

```
Create table Model_Cross
(
ID_Model_Cross int primary key identity (1,1),
ID_Model int foreign key references Petri_Model (ID_Model),
ID_Cross_Position int foreign key references Cross_Position (ID_Cross_Position)
)
alter table Cross_Position add foreign key (ID_Cross) references [Cross] (ID_Cross)
```