НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОБЛЕМ МОДЕЛЮВАННЯ
В ЕНЕРГЕТИЦІ ІМЕНІ Г. Є. ПУХОВА

О. О. Супруненко, Ю. Є. Гребенович

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ КОМПОНЕНТНО-ОРІЄНТОВАНОГО МОДЕЛЮВАННЯ ПРОГРАМНИХ СИСТЕМ ТА PN-ПАТЕРНИ

Монографія

2022

УДК 004.942 : 519.876.5
С 89

Рекомендовано до видання Вченою радою Інституту проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України (протокол № 2 від 24 березня 2022 р.)

Рецензенти: **Чемерис О. А.**, д.т.н., заступник директора з наукової роботи Інституту проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України.

Онищенко Б. О., к.ф.-м.н., доцент, декан факультету обчислювальної техніки інтелектуальних та управляючих систем Черкаського національного університету ім. Б. Хмельницького.

Супруненко О. О., Гребенович Ю. Є.

С 89 Інструментальні засоби компонентно-орієнтованого моделювання програмних систем та PN-патерни. / О. О. Супруненко, Ю. Є. Гребенович. – Черкаси: Видавець Чабаненко Ю. А., 2022, – 82 с.

ISBN 978-996-920-628-2

У монографії розглянуті проблеми розробки інструментальних засобів для імітаційного моделювання програмних систем, які належать до систем з паралелізмом. При проектуванні програмних систем потрібно аналізувати множину пов'язаних між собою паралельних асинхронних процесів, які мають забезпечувати передбачуване функціонування програмної системи.

Розглянуті та проаналізовані 20 основних й 23 додаткових WF-патернів, які широко застосовуються для моделювання управління робочими процесами. На їх основі сформовані еквівалентні PN-патерни, що дозволяють розширити можливості застосування WF-патернів. Запропоновані PN-патерни сформовані на основі управляючих, оціночних та безпечних мереж Петрі з часовими елементами, що належать до класу формальних мов L_G-типу і дозволяють описувати та аналізувати моделі програмних систем. Запропоновано класифікацію PN-патернів, яка

дозволяє розмежувати патерни за специфікою їх використання у моделях систем з паралелізмом.

Пропоновані інструментальні засоби дозволяють однозначно відобразити недетермінований характер функціонування компонентів моделі програмної системи, створюють підґрунтя для автоматизованої перевірки динамічних властивостей моделі графоаналітичними методами.

Для фахівців з моделювання та проектування програмних систем та інших складних систем з паралелізмом.

The monograph discussed the problems of developing instrumental tools for the simulation of software systems that belong to systems with parallelism. When designing software systems, it is necessary to analyze the set of interconnected parallel asynchronous processes that should ensure the intended operation of the software system.

The 20 main and 23 additional WF-patterns, which are widely used for workflow management modelling, were considered and analyzed. Based on them, equivalent PN-patterns were formed, which allow expanding the possibilities of using WF-patterns. The proposed PN-patterns were formed on the basis of control, evaluation and safe Petri nets with elements of time belonging to the class of formal L_G-type languages and allowed to describe and analyze models of software systems. A classification of PN-patterns was proposed, which allows differentiating the patterns according to the specifics of their use in models of systems with parallelism.

The proposed tools allow reflecting unambiguously the non-deterministic nature of the functioning of the components of the software system model, create a basis for automated checking of the dynamic properties of the model by graph-analytical methods.

For specialists in modeling and software systems design, and other complex systems with parallelism.

УДК 004.942 : 519.876.5

ISBN 978-996-920-628-2

ЗМІСТ

ВСТУП

При моделюванні програмних систем, які характеризуються суттєвим паралелізмом, необхідно описати і проаналізувати аспекти їх структурної будови, варіанти функціонування та адаптації до середовища, в якому буде працювати проектована система [1]. Програмна система є сукупністю пов'язаних компонентів, що взаємодіють для досягнення поставлених цілей [2] та характеризуються структурними особливостями, визначеною функціональною й інформаційною складністю. Найбільш важливим і складним при її моделюванні є опис та аналіз взаємодії паралельних і конкуруючих процесів на внутрішньокомпонентних та міжкомпонентних рівнях для оцінки працездатності та надійності системи. Для моделювання структурних і динамічних аспектів систем з паралелізмом найбільш ефективними виявились технології імітаційного моделювання [3, 4], зокрема дискетно-подієве моделювання, та моделювання динамічних систем [1], інструментальні засоби яких дозволяють поводити синтез та аналіз моделі на її графовому та аналітичному представленні та є основою комбінованого підходу до моделювання систем з паралелізмом [5].

Для моделювання будь-якої системи з паралелізмом потрібно сформувати її архітектуру, при конкретизації якої відобразити паралельні та конкуруючі процеси та особливості їх функціонування. Зокрема моделювання програмної системи переважно проводиться на етапі її проектування і дозволяє уточнити структуру системи, з'ясувати необхідні умови її передбачуваної поведінки, а також здешевити створення програмного продукту за рахунок раннього виявлення неузгодженостей зі специфікацією вимог та помилок у реалізації функціональних і нефункціональних вимог до системи. Моделювання також часто проводиться паралельно з конструюванням про-

грамної системи [6], що дозволяє перевірити отримані рішення та уточнити особливості реалізації поточних задач. Оскільки програмні системи належать до недетермінованих динамічних систем, їх моделювання важливо проводити з використанням засобів для динамічного моделювання, що дозволяють описати структуру системи, відобразити та проаналізувати множину процесів, серед яких є асинхронні паралельні та конкуруючі процеси, оцінити їх керованість та надійність.

При побудові інструментарію моделювання систем з паралелізмом потрібно враховувати розмірність та складність їх моделей [5], оскільки використовуючи лише графові засоби моделювання, при зростанні кількості елементів системи та зв'язків між ними, дуже складно забезпечити швидкий і надійний аналіз моделі, особливо при використанні моделювання разом з конструюванням системи, як при розробці програмних систем.

Інструментальні засоби моделювання мають дозволяти сформувати відповідний математичний опис, за яким можливо у автоматичному чи автоматизованому режимі провести перевірку ключових властивостей системи та її компонента. Цим вимогам відповідає апарат мереж Петрі, який має переваги у порівнянні з іншими засобами імітаційного моделювання [5]. Він сполучує у собі переваги двох методологій імітаційного моделювання [1, 7]: дискретно-подієвого моделювання, оскільки має розвинуті графові засоби побудови моделей паралельних систем, та моделювання динамічних систем, оскільки мережева модель однозначно описується математично, що дозволяє проаналізувати низку важливих характеристик досліджуваної системи. Таким чином, на мереж Петрі надають підґрунтя для формування графоаналітичних інструментальних засобів моделювання систем з паралелізмом.

Спільне використання даних методологій імітаційного моделювання обумовлено тим, що класичні технології дискретно-подієвого моделювання належать до концепцій кількісного моделювання [7, 8] і характеризуються слабким зв'язком процесів та даних у моделі, що не дозволяє з висо-

кою адекватністю відображати досліджувані системи. Так, численні спостереження за функціонуванням переважної більшості систем показують [9], що такі моделі є на порядок більш стабільними, ніж процеси, які в них відбуваються. Тому ці технології потребують розширення засобів адекватного опису моделі. Застосування методології моделювання динамічних систем дозволяє підсилити переваги попередньої методології, автоматизувати процес аналізу моделі.

Оскільки програмні системи є складними об'єктами, перший етап їх проектування пов'язаний з мінімізацією складностей (Minimizing Complexity), для якої були запропоновані об'єктні технології моделювання (об'єктно-орієнтоване моделювання, ООМ). Вони базуються на тісному зв'язку даних і процесів системи, що дозволяє розробляти більш надійні системи, які стійкі до змін та відповідають функціонуванню системи в реальному світі [9]. При створенні сучасних програмних систем широко застосовується і компонентно-орієнтована технологія розробки програмних систем [10, 11], яка підтримує концепцію успадкування (з ООМ) та передбачає використання нових концепцій, таких як інтроспективність (здатність до самоопису), модульність, персистентність, здатність до повторного використання компонентів [12]. Ці технології підтримуються ієрархічними та іншими інтерпретаціями мереж Петрі, які дозволяють адекватно описати системи, що створюються з використанням сучасних технологічних досягнень. Так персистентність фактично означає збереження та відновлення стану компонента, що співвідноситься з такими властивостями мереж Петрі, як збережуваність і повторюваність; повторне використання компонентів у моделях забезпечується правилами побудови та взаємодії підмоделей в ієрархічних мережах Петрі.

Ще одним аспектом, який треба враховувати при проектуванні програмних систем є те, що управління процесом набагато складніше, ніж управління даними [12]. При цьому потрібно враховувати, що програмні системи часто створюються для управління певними процесами [2], хоча процеси управ-

ління даними є теж важливими, оскільки програмні системи створені й для обробки даних, а також на основі даних будуються елементи прийняття рішень в автоматичному режимі чи автоматизовано операторами системи, дії яких впливають на вищерозглянуті процеси. Тому при розробці інструментарію моделювання програмних систем, для яких і управління процесом, і управління даними є важливими, потрібно враховувати можливість адекватного відображення основних властивостей динаміки програмних систем та обмежень, що накладаються робочими процесами й особливостями майбутнього оточення системи. Для моделювання подібних задач найчастіше застосовуються безпечні, оціночні та ієрархічні мережі Петрі [5], з допомогою яких можливо описати процеси управління та ресурси, які впливають на поточне функціонування системи, перевірити ступінь керованості програмної системи, досяжність всіх елементів моделі та її обмеженість.

Ускладнює створення інструментів динамічного моделювання програмних систем велике різноманіття видів діяльності при їх проектуванні [2, 13], що, як наслідок, веде до ускладнення побудови проектних рішень і сприяє зростанню розмірності моделей та їх складових, які потрібно перевірити на коректність та інші суттєві властивості. Вирішити задачу перевірки проектних рішень почасти можливо при застосуванні методів аналізу мереж Петрі, які використовують матричний опис побудованої моделі й дозволяють перевірити цілий ряд важливих властивостей програмної системи [14]. Для ефективного застосування даних методів потрібно також передбачити способи зменшення розмірності моделі.

Апарат мереж Петрі також дозволяє вирішити й іншу задачу – побудову графічного представлення моделі, на якому можливо проводити візуальний аналіз та обговорення проектних рішень, що є важливою складовою у роботі команди розробників програмних систем.

1. ВЛАСТИВОСТІ ФОРМАЛЬНИХ МОВ ДЛЯ ОПИСУ КОМБІНОВАНИХ ГРАФОАНАЛІТИЧНИХ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ МОДЕЛЮВАННЯ СИСТЕМ З ПАРАЛЕЛІЗМОМ

Велика кількість розроблених інтерпретацій та модифікацій мереж Петрі [4] частково дозволяє підібрати інструментальні засоби для моделювання паралельних процесів на їх основі до особливостей розв'язуваної задачі. Разом з тим виникає питання, які саме інтерпретації та модифікації треба обрати для створення спеціалізованих комбінованих інструментальних засобів моделювання систем з паралелізмом, зокрема програмних систем [15]. На це питання дає відповідь аналіз формальних мов опису мереж Петрі [5, 14] та обмеження, які накладаються методами аналізу на моделі, побудовані на основі мереж Петрі.

У роботі [16] зазначено, що формальні мови, які породжуються класичними мережами Петрі, належать до класу регулярних мов, задачі еквівалентності регулярних мов є розв'язуваними. При дослідженнях мов мереж Петрі зазначено [17], що вони належать до контекстно-вільних мов, також можуть породжувати біль широкий клас мов. У роботах [17, 18] доведено, що клас комунікаційно-вільних мереж Петрі (клас автоматних мереж Петрі) еквівалентний комунікативній контекстно-вільній граматиці й проблема еквівалентності досяжності для цього класу мереж Петрі є розв'язуваною, хоча для класичних мереж Петрі, які призначені для моделювання паралельних процесів, еквівалентність досяжності залишається нерозв'язуваною. Але різні класи мереж Петрі, побудовані на класичній теорії, можна перетворювати один в інший [19], що дозволяє звернути увагу на класи мереж Петрі, які є

обмеженими, але мають більшу потужність моделювання ніж автоматні та інші елементарні класи мережі Петрі.

При формуванні інструментарію моделювання на базі мереж Петрі варто уникати класів PN з інгібіторними дугами, в яких реалізована перевірка вершини на нуль, для них задача еквівалентності та деякі інші властивості є нерозв'язуваними, хоча вони за потужністю моделювання еквівалентні машині Тьюрінга [17]. У ряді статей [20, 21, 22] розглядаються моделі на їх основі з кінцевим числом станів, для яких доведення еквівалентності та ряду властивостей є успішним, оскільки досліджувані моделі мають кінцеве число елементів і в них відсутні деякі властивості загальних моделей даного класу мереж. Але доведення основних властивостей для цих класів мереж Петрі з допомогою матричних методів опису є ускладненим (перетвореннями, такими як редукція чи еквівалентна заміна). Тому їх не доцільно застосовувати в якості інструментів моделювання, що повинні бути простими у використанні та мати універсальні засоби аналізу.

Клас мов мереж Петрі L-типу є найбільшим та найбільш дослідженим типом формальних мов мереж Петрі, її називають мовою цілком закінчуваних мереж Петрі. Вони мають певні обмеження, які дозволяють спростити аналіз мереж Петрі, що ними описуються. Так при доведенні досяжності чи покриваємості всіх вершин цілком закінчуваної мережі Петрі можемо стверджувати, що вона є повторюваною. Також для даного класу мереж Петрі розв'язуваною є проблема бездефектності [23], яка характеризує динамічні властивості моделі. Важливо, що даний клас мов мереж Петрі є замкнутим до нескінченної конкатенації, що дозволяє використовувати його для опису та аналізу складних систем з паралелізмом. Найбільш відомим класом мереж Петрі, що описується мовами L-типу та є добре дослідженим [23, 24, 25] – це WorkFlow мережі (WF-мережі).

Натомість обмеження, які накладають на класичну мережу Петрі правила мов L-типу, ускладнюють її використання

при моделюванні таких, характерних для програмних систем, конструкцій, як «контроль доступу до критичної секції» та «передача ресурсу з обмеженим буфером», а також при аналізі проміжних станів системи, починаючи з деякої довільної розмітки. Тому класи мереж Петрі, що описуються мовою мереж Петрі L-типу потрібно доповнити певним набором елементів та правил, які дозволять адаптувати їх для моделювання більш широких класів складних систем, включно з моделюванням програмних систем. Це дозволяють зробити мови мереж Петрі G-типу, які називають класом вільних мов мереж Петрі [5]. Вони дозволяють відобразити поточний стан мережі й використати його в якості початкової розмітки. Також у даному класі мов немає строгого обмеження вигляду кінцевої розмітки, що дозволяє використовувати певні вершини місць для контролю доступу до критичних секцій. Класом мов G-типу описують ієрархічні мережі Петрі [26], які дозволяють отримувати модель шляхом злиття окремих її фрагментів. Це важлива властивість, яка використовується при проектуванні програмних систем. Але якщо побудувати інструментальні засоби моделювання програмних систем на менш обмежених мовах G-типу, значно ускладниться аналіз моделей. Тому розглянемо необхідне розширення мови L-типу елементами правил мов G-типу.

Розширенням мови мереж Петрі L-типу елементами мови G-типу є L_G-мова, що описує мережу Петрі $PN=(P,T,K,Q,R)$ з поміченням вершин переходів $\sigma : T \to \Sigma$, початковим маркуванням m_0, проміжним маркуванням m'та множиною підсумкових маркувань M_F, таких, що

$$L_{L_G} = \{\sigma(t_j) \in \Sigma^* \mid t_j \in T \wedge (\delta(\mu_0, t_j) \vee \delta(\mu', t_j)) \in M_F\}.,$$

де $\Sigma^* = \Sigma^+ \cup \{\lambda\}$ – множина всіх рядків з символів абетки Σ, $\sigma(t_j)$ – функція помічення вершин переходів t_j. Мова мереж Петрі L_G-типу дозволяє проводити моделювання, починаючи з будь-якої розмітки, яка належить до кінцевої множини

підсумкових маркувань мережі M_F. Також мережі Петрі, які описуються мовами L_G-типу, можуть мати скінченну множину контролюючих вершин, розмітка яких має відновлюватись по закінченні сеансу імітації моделі. Подібні моделі можуть бути перетворені до моделей, що описуються формальними мовами L-типу, але такі L-моделі будуть порушувати структурну подібність, що ускладнить їх візуальний аналіз, та більшу кількість елементів, що збільшить складність розрахунку їх динамічних властивостей.

2. ПІДҐРУНТЯ ДЛЯ ПОБУДОВИ КОМБІНОВАНИХ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ МОДЕЛЮВАННЯ СИСТЕМ З ПАРАЛЕЛІЗМОМ

Формальні мови мереж Петрі описують узагальнену модель з нескінченною кількістю станів, тобто утворюють нескінченний ланцюжок виведення слів. У практичних задачах при моделюванні складних систем з паралелізмом використовуються мережі Петрі з кінцевою кількістю вершин місць та вершин переходів, але потенційно вони можуть мати нескінченну кількість станів.

Інструментарій для динамічного моделювання прикладних систем будують на основі мереж Петрі з кінцевим числом станів, але якщо аналізувати побудовану модель системи в цілому, вона буде мати велику кількість елементів і це вимагатиме великої кількості часу та потужних засобів. У даній роботі ця проблема вирішується шляхом розбиття моделі на компоненти, кожен з яких може бути побудований із застосуванням комбінованого інструментарію, що оснований на класичних мережах Петрі, а також проаналізований на основі їх аналітичного опису. Відлагоджені компоненти утворюють модель вищого рівня ієрархії. Базується таке рішення на компоненто-орієнтованому підході [27, 28], який дозволяє формувати модель з відносно невеликих компонентів, які можуть бути проаналізовані на критичні властивості і згорнуті у макроелементи, як показано у роботі [26] на основі ієрархічних мереж Петрі, для подальшого аналізу всієї моделі в цілому. У комбінованих моделях на основі компонентного підходу задачу аналізу вирішують при розгляді компонентів, міжком-

понентних зв'язків та елементів мережі, які розміщені за межами компонентів.

Однією з модифікацій класичних мереж Петрі, яка побудована на базі оціночних та часових інтерпретаціях, є модифікація WF-мереж [4]. Простота вирішення задач доведення властивостей для WF-мереж частково компенсується зниженнями потужності моделювання [23, 25], що вимагає розвитку даного інструменту шляхом залучення елементів інших інтерпретацій та модифікацій мереж Петрі, які мають менші обмеження [4], та аналізу їх критичних властивостей [29].

Одним зі шляхів перевірки критичних властивостей є доведення їх еквівалентності з підкласами мереж Петрі, які таких критичних властивостей не мають, або в яких дослідження певних критичних властивостей є розв'язуваною задачею [17, 19]. Наприклад, автоматна мережа Петрі є строго збережуваною [17], всі питання аналізу для цієї мережі розв'язувані, але вона має таку ж потужність моделювання як і кінцеві автомати [25]. Маркована мережа Петрі є безконфліктною [30], але має знижену потужність моделювання – її засобами неможливо описати моменти прийняття рішень та конфліктні ситуації. Оскільки автоматні та марковані мережі Петрі є простими для аналізу, зокрема, для маркованих мереж Петрі доведені властивості живості, безпечності та досяжності, їх використовують для аналізу інших підкласів мереж Петрі, прикладом цього можуть слугувати WF-мережі з автоматним покриттям [23]. Автоматні та марковані мережі Петрі також можна використовувати для опису міжкомпонентних зв'язків у комбінованій моделі.

Перевагою інтерпретацій та модифікацій (класів) мереж Петрі, побудованих на основі класичних мереж Петрі є можливість перетворення одного класу мереж Петрі в інший [19]. Так, мережу Петрі будь-якого класу можна перетворити у класичну мережу Петрі, і навпаки класичну мережу Петрі можна перетворити до визначеної інтерпретації чи модифікації [4],

які побудовані на її основі. Також WF-мережу (мережу Петрі стандартного виду) можна перетворити у будь-яку іншу інтерпретацію чи модифікацію мереж Петрі, яка побудована на основі класичних мереж Пері [17].

WF-мережі дозволяють описувати розпаралелювання, синхронізацію, однак мають обмеження [31], зокрема, має бути одна вхідна та одна вихідна вершини у моделі, у результаті моделювання з початково розміченої вхідної вершини мітка має, подолавши певний шлях, перейти у вихідну вершину, при цьому у кінцевій розмітці не має залишитись міток у жодній проміжній вершині місця [23]. Також у WF-мережі потрібно забезпечити відсутність тупиків, надлишкових переходів, необґрунтованого використання ресурсів [32, 33]. Формальним критерієм виконання даних вимог є бездефектність, яка складається з двох умов:

1) коректне завершення процесу з будь-якої розмітки мережі, яка може утворитися з початкової розмітки (мітка лише у початковій вершині), і по завершенню процесу у вершинах WF-мережі, крім кінцевої, не має залишитись жодної мітки; у класичних мережах Петрі дана умова асоціюється з умовою обмеженості;

2) забезпечення відсутності у WF-мережі переходів, що ніколи не спрацьовують; у класичних мережах Петрі такі переходи називають неживими [31].

Для WF-мереж бездефектність є вирішуваною за поліноміальний час задачею [23]. Розроблені також WF-мережі, які за побудовою є бездефектними [34]. Для ще однієї модифікації – вкладених WF-мереж доведена перша умова бездефектності, умова живості таких мереж не доведена [34], тому потрібні подальші дослідження по створенню коректного і бездефектного інструментарію моделювання складних систем з паралельними та конкуруючими процесами.

Хоча класичні WF-мережі мають доведену властивість бездефектності, яка може бути перевірена за поліноміаль-

ний час, але правила їх побудови та функціонування вимагають початкової розмітки, яка пов'язана тільки з початковою вершиною місця, та кінцевої розмітки лише у кінцевій вершині місця. Тому WF-мережі не підтримують конструкції, що часто зустрічаються при моделюванні програмних систем, наприклад такі, як конструкції доступу до критичної секції, передачі ресурсу з обмеженим буфером та інші. Вони також не дозволяють використати важливу властивість класичних мереж Петрі, в яких можливо почати моделювання системи з будь-якого її проміжного стану [19], тобто в якості початкової розмітки може використовуватись будь-яка розмітка з множини допустимих розміток $\forall \mu \in M(PN, \mu_0)$.

Оскільки WF-мережі мають досить жорсткі обмеження, які не дозволяють будувати структурно-подібні моделі програмних систем, що значно ускладнює їх моделювання, але вони є добре дослідженими, у даній роботі вони використовуються в якості основи для комбінованого підходу [29] та побудови на його основі інструментарію моделювання систем з паралелізмом, до яких належать програмні системи.

При розробці нових інтерпретацій та модифікацій мереж Петрі дослідники намагаються підвищити їх потужність моделювання, але доповнення, що стосуються елементів мереж та правил її функціонування можуть ускладнювати їх побудову та аналіз. Прикладом такої мережі Петрі є PRO-мережа [35], вершини місць та вершини переходів якої мають кілька доданих характеристик. Складними є правила спрацьовування переходів PRO-мережі, вони потребують розділення процесу спрацьовування вершин переходів на 3 етапи, які містять кілька складових [35]. При розширенні існуючих інтерпретацій та модифікацій мереж Петрі потрібно врахувати подібні приклади, зручність їх використання при моделюванні та аналізі систем, і брати за основу такі варіанти характеристик елементів мереж та правил їх функціонування, які мінімально ускладнять їх побудову та дозволятимуть проводити аналіз моделі з прийнятним рівнем складності.

Розширити інструментарій моделювання програмних систем можливо, використовуючи інтерпретації та модифікації класичних мереж Петрі, які дозволяють описати структурні та динамічні аспекти модельованої системи, а також відобразити її суттєві характеристики [29, 36]. Наприклад, такими інтерпретаціями можуть бути оціночні та управляючі мережі Петрі [37, 38], які дозволяють моделювати ресурсний потенціал цільової системи, хоча їх аналіз проводити дещо складніше, ніж аналіз WF-мереж. Тому моделі, які побудовані на їх основі потребують обмеження у кількості елементів, що обумовлює можливість їх застосування при побудові моделі з невеликих блоків чи модулів, що є характерним для об'єктно-орієнтованого та компонентно-орієнтованого підходів до розробки програмних систем [10, 39]. Для розширення інструментарію WF-мереж також можуть бути застосовані однолічильникові мережі Петрі, для яких доведена розв'язуваність бісимуляції [40], у сполученні з компонентним підходом до проектування та розробки програмних систем [10, 41].

3. ПІДҐРУНТЯ ДЛЯ ФОРМУВАННЯ БАЗОВИХ КОНСТРУКЦІЙ ІННСТУМЕНТАЛЬНИХ ЗАСОБІВ МОДЕЛЮВАННЯ СИСТЕМИ З ПАРАЛЕЛІЗМОМ

При розвитку інструментарію мереж Петрі автори деяких робіт [17, 42] вказують на необхідність перетворення мереж Петрі, що зберігають мову мереж Петрі. Таким чином, можливо отримати інтерпретації та модифікації мереж Петрі, які враховують особливості побудови моделей складних програмних систем, а також дозволяють провести їх аналіз, використовуючи напрацьовані аналітичні методи [25, 26, 43]. Підхід до моделювання взаємодіючих паралельних асинхронних процесів, що представляються потоками робіт з описом робіт як подій (event-based), збігається із запропонованим у Маніфесті аналізу процесів (manifest process mining) від IEEE [44], де також зазначається про необхідність розвитку інструментів динамічного моделювання.

У численних роботах пропонується використовувати певні інтерпретації та модифікації мереж Петрі (PN) для моделювання та верифікації алгоритмів керування потоками робіт [32, 33, 34]. Проблеми, які постають при автоматичному формуванні, корегуванні та аналізі моделей потоків робіт [32, 45, 46], можуть бути вирішені багатьма способами [47]. У останніх роботах [46, 48] моделі потоків робіт називають моделями потоків управління. Найбільш розповсюдженими є пропозиції, з яких перша стосується побудови універсального інструментарію моделювання «для всіх випадків», а друга – розширення існуючого інструментарію, який добре зарекомендував себе

при розв'язанні певного класу задач, для вирішення інших класів задач. Другий підхід є більш прагматичним і рекомендується багатьма авторами, в тому числі авторами [46, 49], які працюють над проблемами моделювання паралельних процесів. У даній роботі використовується другий підхід, що передбачає розвиток інструментарію динамічного моделювання, до якого належить комбінований інструментарій моделювання систем з паралелізмом на основі мереж Петрі. Класичні інтерпретації мереж Петрі дозволяють досліджувати модель як аналітичними так і імітаційними методами, виконувати перетворення моделі у режимі реального часу, що важливо при проектуванні та реалізації програмних проектів [44].

Одна з класичних модифікацій мереж Петрі – WF-мережа описується формальною мовою L-типу. Вона має ряд переваг [23], що стосуються доведення основних властивостей досліджуваної моделі. У термінах формальних мов L-типу доведення таких властивостей, як безпечність, обмежуваність, збережуваність, живість, досяжність та покриття, зводиться до доведення існування (досяжність одного стану досліджуваної системи з іншого заданого) та приналежності (можливості зазначеної послідовності дій у моделі), що значно спрощує аналіз [17].

Натомість WF-мережі мають і певні недоліки. Зокрема, при моделювання програмних систем WF-мережі складно використовувати для опису процесів, при моделюванні яких потрібні певні «контролюючі» вершини, де залишаються мітки у початковій та кінцевій розмітках [29]. Приклад такої конструкції представлений на рис. 1, *a*. На рис. 1, *b* представлена конструкція, яка еквівалентна наведеній на рис. 1, *a*, побудована в термінах WF-мереж, у якій безпечний доступ до критичної секції можна забезпечити такою ж структурою з додатковими вершинами та дугами, які дозволяють не встановлювати початкову розмітку у контролюючій вершині місця p_c, а обмежитись лише початковим маркуванням $\mu(p_0) = 1$. Така конструкція у простих моделях дозволяє отримати з до-

даткової вершини переходу t_0 маркування μ_1, яке відповідає початковому маркуванню мережі з прикладу на рис. 1, *a*. Після відпрацювання моделі з прикладу на рис. 1, *b* до вершини t_{end} переходять всі мітки, які залишились у контролюючих вершинах, що дозволяє уникнути непередбачуваної поведінки моделі у наступних імітаційних експериментах. Але коли подібні конструкції застосовуються як елементи більш складної моделі компонента цільової системи, додаткові вершини та дуги ускладнюють аналіз моделі, крім того порушується і принцип структурної подібності, який значно полегшує перевірку моделі у візуальному режимі.

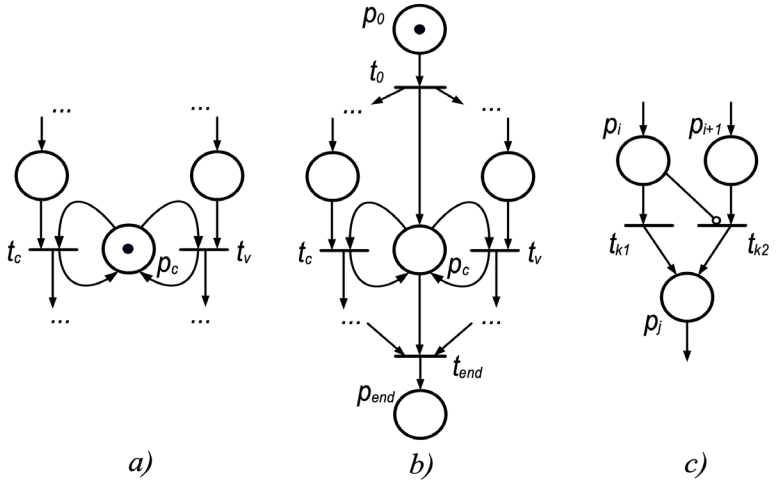


Рис. 1. Опис доступу до критичної секції засобами безпечних мереж Петрі, що описуються *a)* мовою G -типу та *b)* мовою L-типу; *c)* опис безпечного доступу до вершини з одиничним обмеженням.

На відміну від попередньо згаданої інтерпретації, безпечні та оціночні інтерпретації мереж Петрі, які описуються мовами мереж Петрі G-типу, можуть бути застосовані для опису конструкцій з обмеженими «контролюючими» вершинами.

Зокрема безпечні мережі Петрі дозволяють описати доступ до критичної секції кількома способами з використанням інгібіторних дуг (рис. 1, *c*), а також без їх використання (рис. 1, *a*), що дозволяє застосовувати аналітичні методи перевірки критичних властивостей.

Оціночні мережі Петрі дозволяють у більш компактній формі представляти алгоритмічні конструкції, вони можуть бути переформовані до безпечних мереж Петрі шляхом еквівалентних перетворень [19, 26], хоча таке представлення буде більш громіздким. Але воно може бути використане при моделюванні та аналізі проектних рішень на етапі детального проектування компонентів системи з метою глибшої деталізації опису процесів, а також на етапі дослідження властивостей моделей, побудованих на основі інших інтерпретацій PN, оскільки в термінах безпечних мереж Петрі простіше довести їх основні властивості.

Управляючі мережі Петрі також мають потужний потенціал відображення особливостей функціонування паралельних процесів. Зокрема цією модифікацією мереж Петрі можливо представити конструкції, які, як стверджується у роботі [25], не можливо описати у термінах класичних мереж Петрі. Це можливості скасування виконання певного підпроцесу чи області моделі, а також відсутність шаблону злиття синхронізації, що значить злиття активних гілок та ігнорування неактивних гілок, які входять у з'єднання. Автором [25] зазначається, що вони реалізовані у мові YAWL, семантика якої не визначається у термінах мереж Петрі, а використовує терміни системи переходів [50].

У термінах управляючих мереж Петрі [26], які побудовані на базі класичних мереж Петрі, можливість відображення таких конструкцій існує. Так, скасування виконання певного підпроцесу можливо відобразити, побудувавши модель у вигляді окремих компонентів моделі, кожен з яких відлагоджується, як окремий елемент (як і при використанні YAWL), і з'єднується у загальну модель через керований перехід [51], тоді у разі надходження зовнішнього управляючого впливу, що реалізується вектором X_i (спрямованим до відповідної вершини переходу в

середині підпроцесу), скасування виконання вже запущеного підпроцесу може бути здійсненим до моменту закінчення його роботи [52]. У мові YAWL відміна виконання певного екземпляру завдання також виконується зовнішнім об'єктом, який чинить вплив на відповідний внутрішній елемент [25].

Крім того, управляючі мережі Петрі дозволяють передати фокус управління моделлю на інший процес, використовуючи вихідний вектор керування Y_i, в якому відображені особливості запуску наступного процесу в залежності від стадії виконання попереднього процесу [26].

При реалізації шаблону злиття синхронізації, тобто при злитті активних гілок, можливість ігнорування неактивних гілок реалізується конструкцією (табл. 1 (додаток А), п. 7), у якій передбачена перевірка функціонування активних гілок шляхом передачі значень вихідного вектора керування Y_i чи кількох вихідних векторів, як показано на рис. 2 до вхідного вектора синхронізуючого з'єднання (у прикладі: X_3) Таким чином, у моделі відображається, які вхідні вершини місць повинні мати мітки для активізації певної вершини переходу при синхронізації.

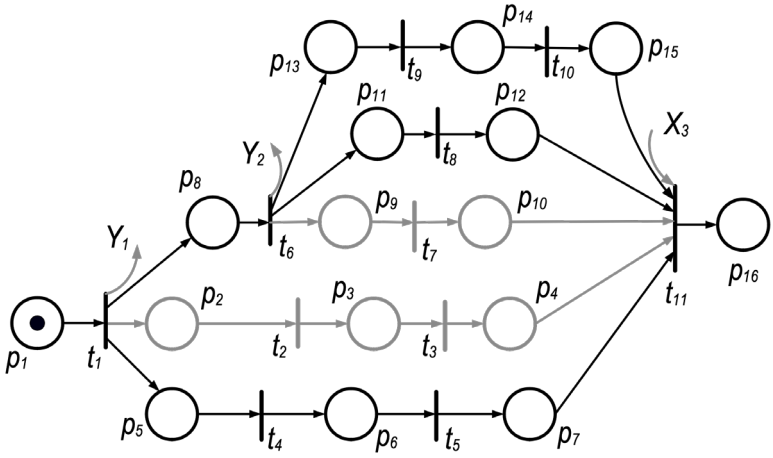


Рис. 2. Варіант реалізації шаблону злиття синхронізації, представлений управляючою мережею Петрі

Мови мереж Петрі замкнуті по відношенню до кінцевої підстановки [17], що дозволяє використовувати конструкційні примітиви для моделювання складних систем, в яких певна вершина місця може бути замінена на визначений примітив чи кінцевий відрізок мережі Петрі. Заміна відбувається шляхом злиття початкової вершини місця вбудовуваної ділянки, наприклад, вершини p'_1 з вершиною місця p_3, в яку здійснюється кінцева підстановка, та кінцевої вершини місця ділянки p'_3 з дублікатом цієї ж вершини p_{31} (рис. 3).

Якщо проаналізувати причини виникнення критичних властивостей у моделях, побудованих з використанням мереж Петрі, то потрібно розділити вершини переходів на вершини з обчислювальними функціями (в них обробляються дані) і вершини прийняття рішень, які пов'язані з управлінням моделлю. Розділимо вершини місць на прості вершини, які передують обчисленням, та вершини, в яких описується прийняття рішень. Вершини, пов'язані з обчисленнями мають одну вхідну та одну вихідну дуги, вони відповідають властивості безконфліктності, яка притаманна автоматним мережам Петрі [26, 53].

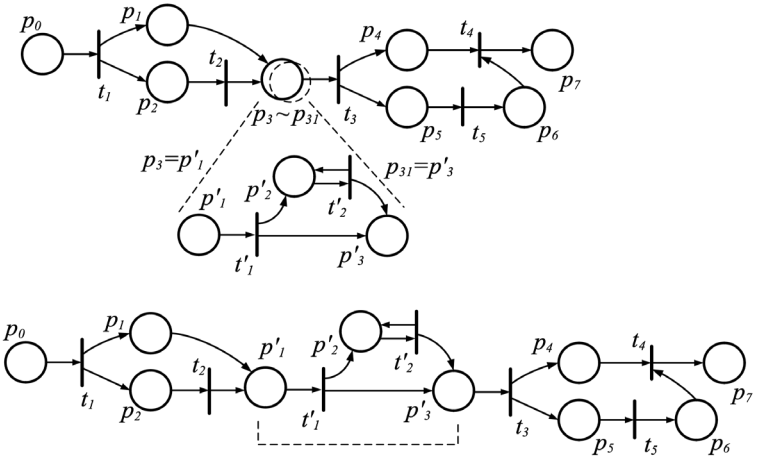


Рис. 3. Кінцева підстановка у вершину місця p_3 ділянки мережі Петрі.

Вершини, що описують прийняття рішень мають кілька вихідних чи вхідних дуг, що і може породжувати критичні властивості. Також до таких вершин можуть застосовуватись управляючі вектори вхідний $X = \{x_1, x_2, ..., x_s\}$ та вихідний $Y = \{y_1, y_2, ..., y_v\}$ [26, 54]. Ці вектори можуть бути застосовані і до вершин місць, які мають кілька вихідних дуг, і в них вибирається наступний шлях по одній з цих дуг.

Треба враховувати, що якщо у моделі є вершини переходів, які мають кілька вихідних дуг, які описують утворення паралельних потоків, то мають бути і вершини переходів, які мають відповідну кількість вхідних дуг для злиття потоків, у такій моделі має забезпечуватись властивість збережуваності мережі Петрі [15]. Використання таких вершин є необхідним при моделюванні систем з паралелізмом, але зумовлює можливості з виникнення критичних ситуацій, тому потрібно дотримуватись правил для коректного функціонування.

4. АНАЛІЗ БАЗОВИХ WORKFLOW-ПАТЕРНІВ У НОТАЦІЇ МЕРЕЖ ПЕТРІ

Для застосування розширеного інструментарію мереж Петрі у практичній діяльності потрібен набір патернів, що дозволять використовувати комбінований інструментарій, отриманий з інтерпретації та модифікації мереж Петрі, дослідниками та практиками, які детально не знайомі з особливостями апарату мереж Петрі, але володіють методикою їх практичного застосування у дослідженні моделей цільових систем.

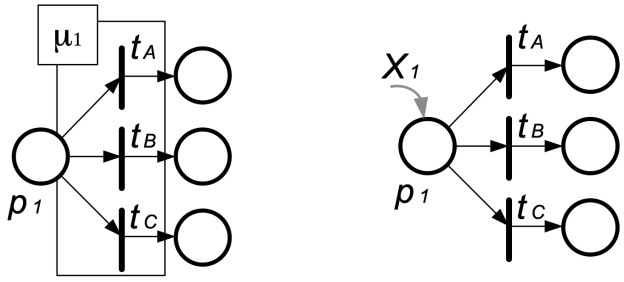
Під патерном розуміють конкретну конструкцію, яка постійно повторюється у певних непередбачуваних контекстах [46]. На початку формувалися патерни для опису моделей потоків робіт [48], для їх представлення використовувались різні нотації [53], більшість з яких були несумісними між собою, що викликало проблеми аналізу рішень, представлених різними нотаціями для опису потоків робіт. Мови опису потоків робіт основувались на теорії мереж Петрі та на р-численні [55], але відрізнялись від строгих теорій тим, що були нестрогими (зокрема мови WPDL, XPDL), тобто дозволяли будувати мовні конструкції, які не були визначені однозначно, що породжувало проблеми при аналізі моделей. Для вирішення даних проблем були запропоновані [25] строгі мови, такі як YAWL, які, як стверджували розробники, не можуть бути описані класичними мережами Петрі. Пізніше були сформовані 20 базових патернів для опису потоків управління [46], що дозволяли будувати моделі, які незалежні від технологій розробки програмних систем. У подальшому вони були розширені ще 23 патернами, отримати які допоміг їх опис кольоровими мережами Петрі [46, 48]. Доповнення новими патернами було викликане тим, що деякі з основних патернів можуть мати більше одного тлумачен-

ня, а доповнені – дозволяють глибше зрозуміти особливості застосування патернів при побудові і дослідженні моделі, а також досягти однозначного опису потоків управління.

Для узгодженого використання патернів сформовані блоки вхідною (вхідними) та вихідною (вихідними) вершинами місць [31], що дозволяє виконувати кінцеві підстановки, подібні до описаних на рис. 3 та на рис. 38.

У наведеній таблиці 1 (додаток А) в якості базових елементів використовуються складові управляючих (а) та безпечних і оціночних (б) мереж Петрі. У деяких патернах використовуються дуги, що передають сигнали управління ззовні (елементи управляючих мереж Петрі), вони не починаються з вершин місць, таким чином їх можна легко відрізнити від внутрішніх елементів моделі (вони позначені сірим кольором). До деяких елементів застосовуються часові характеристики.

Патерни з 1-го по 3-й відображаються мережами Петрі подібно до стандартних зображень WF-патернів. Наступні патерни, починаючи з 4-го, потребують деяких додаткових елементів. Зокрема, у патерні 4 «Виключний вибір» реалізований вибір однієї гілки у вершині місця p_1 для наступного виконання з допомогою умов. У відображенні патерну мережею Петрі такий вибір однієї гілки відокремлений прямокутником з позначенням μ_1 (рис. 4, а) чи позначений вхідним управляючим вектором X_1 (рис. 4, б). Деяки патерни дуже схожі за

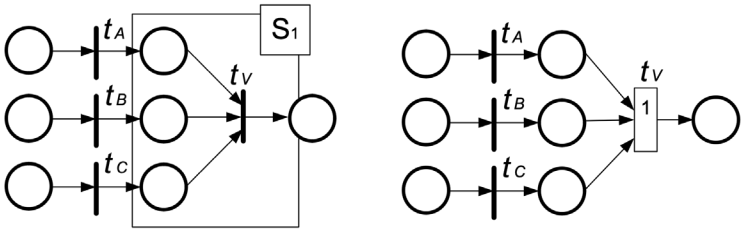


а) *б)*

Рис. 4. Позначення патерну 4 «Виключний вибір»

зображенням та функціями, вони у наступному пункті роботи будуть представлені узагальненими патернами *i**.

У патерні 5 «Просте з'єднання» потрібно відобразити, що вихідний перехід t_v активізується міткою з будь-якої вхідної вершини місця, для цього у відображенні патерну 5 використані два варіанти (рис. 5): а) ділянка мережі, відокремлена прямокутним регіоном S_1, який у компактній формі передає умову спрацьовування переходу від однієї будь-якої вхідної вершини місця, б) відображення самої вершини переходу t_v у вигляді вузького прямокутника із зображенням «1», що і є позначенням умови спрацьовування переходу.



а) *б)*

Рис. 5. Позначення патерну 5 «Просте з'єднання»:
***а)* з допомогою прямокутного регіону S_1,**
***б)* спеціальним зображенням вершини переходу t_v.**

У патерні 6 «Множинний вибір» вибір кількох гілок з наявної множини відбувається (рис. 6) за умовами для відповідних гілок у вершині t_v (з вектором $\overline{Y}_v$ (а), який містить інформацію для подальшого злиття активізованих потоків, чи компактне представлення цього вибору у спеціальному зображенні вершини переходу з символом m (рис. 6, б)), у патерні 7 «Синхронізуюче з'єднання» злиття (рис. 7) відповідних попередньому вибору гілок – у вершині переходу t_v (з вектором $\overline{X}_v$ (рис. 7, а), який прийняв інформацію від вектора $\overline{Y}_v$, чи теж позначеній символом μ (рис. 7, б), що означає умову спрацьовування переходу, визначену певною попередньою розміткою вибраних

гілок). У варіантах (а) даних патернів більш наочно відображений процес передачі вибору відповідних гілок з допомогою векторів $\overline{Y}_v$ та $\overline{X}_v$ управляючих мереж Петрі, натомість варіанти (б) є більш компактними.

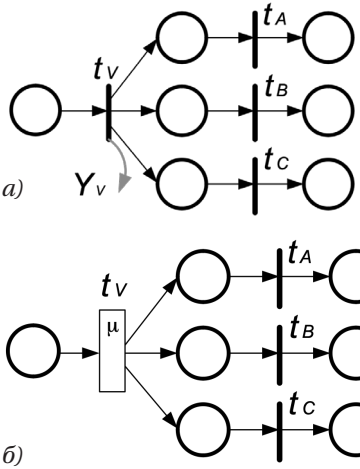


Рис. 6. Патерн 6 «Множинний вибір»

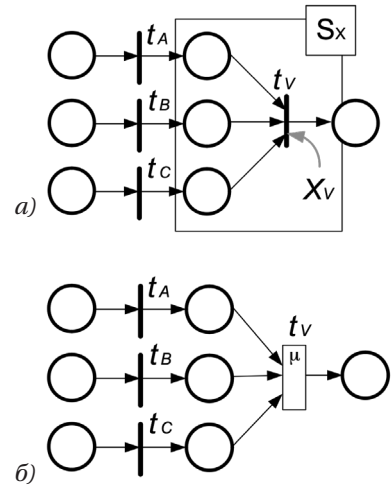
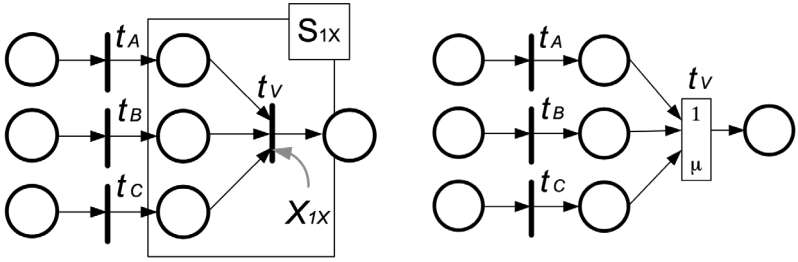


Рис. 7 Патерн 7 «Синхронізуюче з'єднання»

У патерні 8 «Множинне з'єднання» (рис. 8) відображена можливість спрацьовування вихідної вершини переходу t_v, яка активується міткою з будь-якої однієї чи кількох вхідних вершин місць, відображається прямокутником з S_{1X} та управляючим вектором $\overline{X}_{1X}$ (а), або поміткою вершини переходу символами «1» та m, що відповідає обом варіантам умов спрацьовування.

У патерні 9 «Структурований дискримінатор» (рис. 9) факт ігнорування наступних за першим із виконаних потоків робіт відображається цифрою «1» над цифрою «0» на тлі вершини переходу t_v. Вони позначають, що вершина переходу t_v спрацьовує при надходженні першої з можливих міток з вхідних вершин місць, інші – ігноруються.



а) *б)*

Рис. 8. Патерн 8 «Множинне з'єднання»

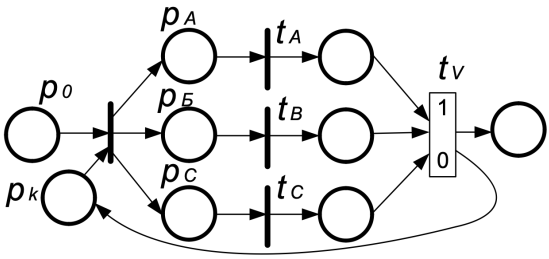


Рис. 9. Патерн 9 «Структурований дискримінатор»

Патерн 10 «Довільний цикл» (рис. 10) відображає варіант реалізації циклу з кількома точками входу або виходу [46], відображаючи переходи між відповідними елементами безпечних мереж Петрі. Використовувати патерн 10 у конструюванні моделі приходиться обережно, оскільки потрібно перевіряти коректність точок входу та виходу з циклу, що описують конкретну неструктуровану частинку моделі. Можливо також перетворення неструктурованого циклу у структуровані стандартні цикли [56], що спростить перевірку моделі.

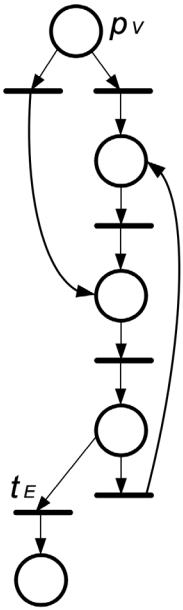


Рис. 10. Патерн 10
«Довільний цикл»

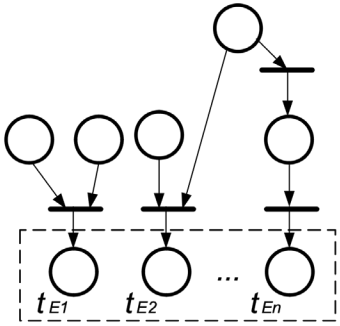


Рис. 11. Патерн 11
«Неявне припинення»

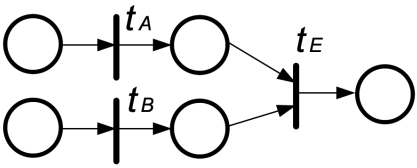
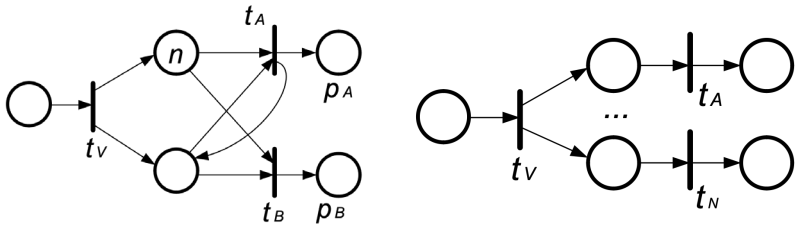


Рис. 12. Патерн 43
«Явне припинення»

Патерни 11 «Неявне припинення» (рис. 11) та 43 «Явне припинення» (рис. 12) представляють властивість WF-мереж однозначно відображати закінчення відпрацювання моделі. У патерні 43 закінчення моделювання потоків робіт (потоків управління) відтворений класичний варіант WF-мереж зі зведення результатів роботи мережі до єдиної вершини t_E. Патерн 11 «Неявне припинення» не відповідає класичному визначенню WF-мереж, у цьому патерні закінчення роботи мережі може відображатись переміщенням міток до кількох кінцевих вершин, але такий варіант закінчення можливий завдяки контролю вмісту всіх кінцевих вершин, розміщення міток у яких свідчить не тільки про саме закінчення відпрацювання мережі, а і про відсутність міток у інших вершинах

місць досліджуваної мережі. Даний варіант має еквівалентне відображення у класичних WF-мережах, але реалізація такого варіанту безпосередньо у моделі ускладнює її розуміння розробниками моделі, що показано авторами роботи [56]. Патерн 11 «Неявне припинення» може застосовуватись, наприклад, при моделюванні інтерфейсів програмного забезпечення, за умов дотримання обмежень, що визначені для WF-мереж, а також правил визначення неявного припинення, які дозволяють відрізнити його від тупика (deadlock), наприклад, коли вимагається для кожного процесу у моделі визначити досягнення конкретного кінцевого вузла [46].

Патерн 12 «Кілька екземплярів без синхронізації» (рис. 13) дозволяє створювати незалежні екземпляри завдань, які можуть оброблятися одночасно і не потребують синхронізації при завершенні.



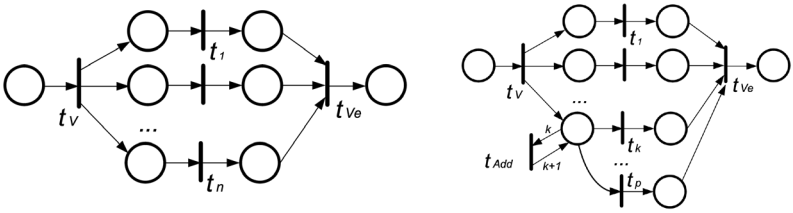
а) *б)*

Рис. 13. Патерн 12 «Кілька екземплярів без синхронізації»

Два варіанти відображення цього патерну передбачають, що можна створити певну кількість екземплярів однієї задачі $A_1 \ldots A_n$ (кількість екземплярів визначена заздалегідь) у циклі (t_A) та потім створювати екземпляр наступного завдання B у вершині t_B (а); або можна створювати незалежні ланцюжки з n екземплярів певного завдання (б).

Відображення 13 та 14 патернів схожі (рис. 14, а) і передбачають створення заздалегідь відомої кількості екземплярів завдання, які виконуються паралельно і незалежно.

Різняться ці патерни параметрами завдань. Так, патерн 13 «Кілька екземплярів із апріорними знаннями під час проектування» вимагає знань про кількість екземплярів завдання, які будуть створюватись, патерн 14 «Кілька екземплярів із апріорними знаннями про час виконання» вимагає додатково знань про час обробки екземплярів завдання, включаючи доступність ресурсів та комунікації з іншими процесами при виконанні екземплярів завдання [46]. У обох цих патернах після закінчення обробки екземплярів завдання потрібно їх синхронізувати перед тим, як розпочати обробку наступного завдання (у вершині місця t_{Ve}).

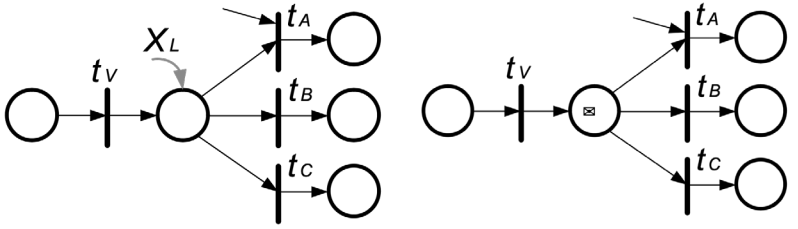


а) *б)*

Рис. 14. Патерни «Кілька екземплярів …»:
***а)* патерн 13 «…із апріорними знаннями під час проектування» та патерн 14 «…із апріорними знаннями про час виконання», *б)* патерн 15 «… без апріорних знань про час виконання».**

У патерні 15 «Кілька екземплярів без апріорних знань про час виконання» (рис. 14, б) передбачено, що не всі параметри паралельно виконуваних екземплярів завдання, які створюються, будуть відомі заздалегідь. У патерні 15 передбачена можливість створення за потреби додаткових екземплярів завдань (з використанням вершини переходу t_{Add}). Патерн 15, як і два попередні патерни, потребує синхронізації наприкінці виконання всіх створених екземплярів завдання, тобто перед початком обробки нового завдання.

Патерн 16 «Відкладений вибір» (рис. 15) є розширенням патерну 4 «Виключний вибір», який дозволяє вийти з тупикової ситуації, коли вибране завдання визначається іншим процесом (а) чи не виконується певну кількість часу (очікування), яка передбачена заздалегідь (б). Коли час очікування спливає, «очікуюче» завдання блокується та ініціюється виконання альтернативного завдання. Для опису процесів керування у патерні можуть бути використані управляючий вектор $\overline{X}_L$ у вершині місця (а) або позначення «конверт» у вершині місця (б), які реалізують подальший вибір гілки для спрацювання в залежності від встановлених параметрів.



а) *б)*

Рис. 15. Патерн 16 «Відкладений вибір»

Патерн 17 «Interleaved-паралельна маршрутизація» (рис. 16) дозволяє побудувати певний псевдопослідовно-паралельний маршрут виконання набору завдань, порядок виконання завдань може змінюватись у допустимих межах, але жодне завдання не може виконуватись одночасно з іншим, що забезпечує «контролююча» вершина місця p_k. Також жодне з завдань не може бути призупиненим для виконання іншого завдання. Відображення даного патерну засобами UML не передбачене [57], оскільки немає примітивів, які дозволяють реалізувати «контролюючу» вершину.

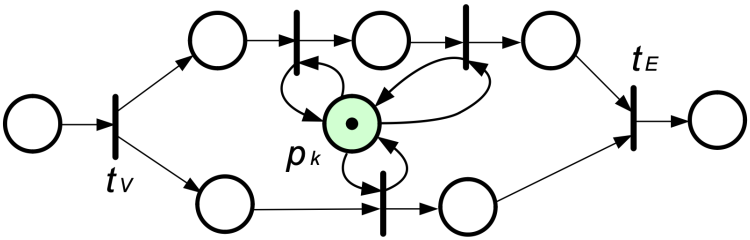


Рис. 16. Патерн 17 «Interleaved-паралельна маршрутизація»

Патерн 18 «Етап» (рис. 17) відображає залежність виконання певного завдання t_k (яке можна назвати тестовим) у поточній гілці від досягнутого стану p_s у паралельній гілці моделі, причому виконання чи невиконання даного завдання не впливає на паралельну гілку. Послідовність виконання завдань t_B та t_k чи t_k та t_B не має значення. Також патерн 18 асоціюється з такими поняттями, як «крайній термін» (deadline) чи «відізвати повідомлення», тобто якщо завдання А не буде виконане, то і перехід t_k не спрацює і завдання, пов'язане з ним не буде виконане.

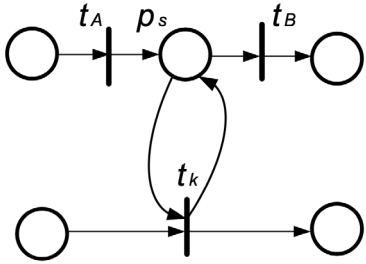
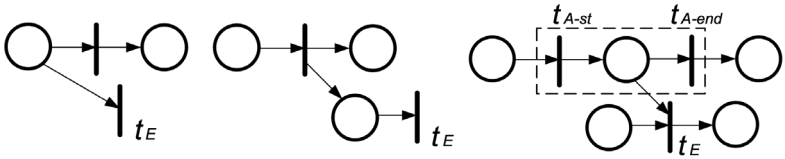


Рис. 17. Патерн 18 «Етап»

Патерн 19 «Скасувати завдання» (рис. 18) дозволяє припинити виконання завдання, якому передана мітка (а) (до початку його виконання), або запущене на виконання завдання (б-в) (під час його виконання) і видалити мітку з поточної гілки.

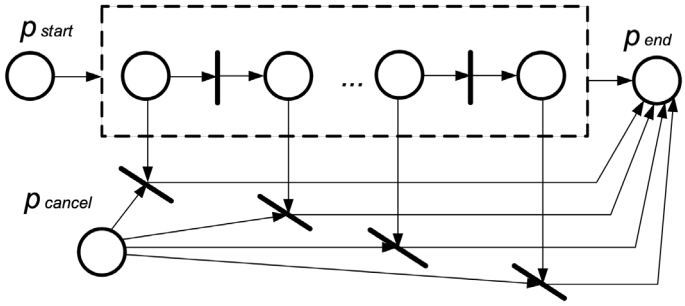
Патерн 19 (рис. 18, в) відображає особливості припинення запущеного на виконання завдання у UML, яке полягає у розміщенні такого завдання у перервній області, яка може бути активована чи деактивована у будь-який момент часу іншим завданням або сигналом управління через вершину t_E. У багатьох інших спеціалізованих мовах та програмних середовищах [57] припинення запущеного завдання здійснюється через механізми обробки помилок та несправностей.



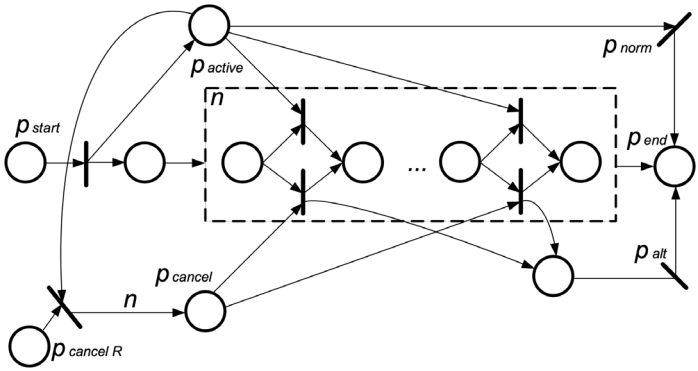
а) *б)* *в)*

Рис. 18. Патерн 19 «Скасувати завдання»

Патерн 20 «Скасувати процес» (рис. 19) забезпечує припинення виконання певного процесу і всіх завдань, що з ним пов'язані. У варіанті (а) цього патерну реалізовано механізм вилучення міток з усіх проміжних вершин місць процесу, що скасовується.



а)



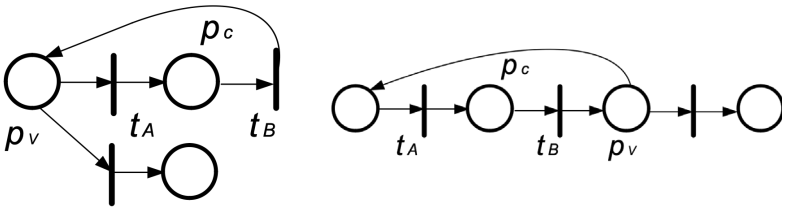
б)

Рис. 19. Патерн 20 «Скасувати процес»:
***а)* скасування виконання всіх активних завдань, *б)* нормальна робота та альтернативний варіант – скасування процесу**

Але для коректного скасування всіх задач, потрібно, щоб серед них не було таких задач, які пов'язані із задачами інших процесів, оскільки після скасування з інших процесів у скасоване завдання можуть надходити мітки. Тому варіант (б), в якому представлені нормальна робота процесу та альтернатива скасування процесу, дозволяє контролювати активності у процесі та у разі його скасування блокувати будь-які події з обробки завдань. При моделюванні процесів у програмних системах важливо, щоб після скасування окремого процесу з усіх вершин місць цього процесу не тільки вилучалися мітки, але і у «контролюючих» вершинах місць (якщо такі застосовуються у моделі) встановлювалась кількість міток, яка відповідає початковій розмітці цього процесу (для забезпечення коректного запуску цього процесу у наступному циклі моделювання).

Патерн 21 «Структурований цикл» може бути представлений циклом з передумовою (рис. 20, а) та циклом з постумовою (рис. 20, б). У циклі з передумовою тіло циклу може не виконуватись ні разу або бути виконаним певну кількість разів, у циклі з постумовою тіло циклу має виконатись хоча б

один раз. У даних патернах у вершині місця p_V визначається, чи буде продовжуватись виконання завдань у циклі, чи управління буде передане на наступне завдання за циклом. Важливо, що у будь-який момент моделювання може виконуватись лише один екземпляр циклу, тому дані патерни побудовані на основі безпечних мереж Петрі, за правилами функціонування яких забезпечується дана умова коректної роботи з циклом.



а) *б)*

Рис. 20. Патерн 21 «Структурований цикл»:
***а)* цикл з передумовою, *б)* цикл з постумовою**

Патерн 22 «Рекурсія» (рис. 21) дозволяє відобразити, що запущене на виконання завдання (t_A) може викликати самого себе чи дочірнє завдання, до закінчення виконання якого основне завдання буде очікувати (p_A), а після закінчення зможе завершити своє виконання (t_{Aend}). Багаторазова рекурсія у даному патерні може бути реалізована через відтворення даного патерну вкладеним у вершину переходу t_2 (при цьому p_0 збігається з p_2, p_e збігається з p_3).

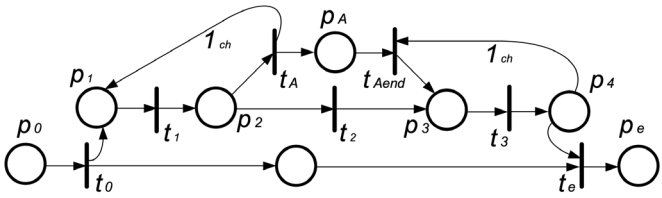


Рис. 4.21. Патерн 22 «Рекурсія».

Патерн 23 «Перехідний тригер» дозволяє відобразити управляючі сигнали, які можуть надходити з інших частин даної моделі чи від інших (зовнішніх) систем. У перехідному тригері реалізується тип тригера [45, 46], сигнал від якого може бути прийнятий негайно, інакше він втрачається (t_{remove}) (рис. 22, а). Мітка у вершині місця p_{tr} затримуватись не може, і якщо її не готовий реалізувати основний процес через вершину переходу t_V, то у наступний момент часу мітка тригера буде видалена з моделі через вершину t_{remove}. На відміну від перехідного тригера у патерні 24 «Стійкий тригер» реалізується можливість передати сигнал управління, який відобразиться міткою у вершині p_{tr}(рис. 22, б), через певний проміжок часу. Такий тригер може очікувати на передачу управління вершині переходу t_V, поки він надійде (у вершину p_V). Стійкий тригер також може бути реалізований управляючими мережами Петрі через механізм управляючих векторів X_V та Y_V [26].

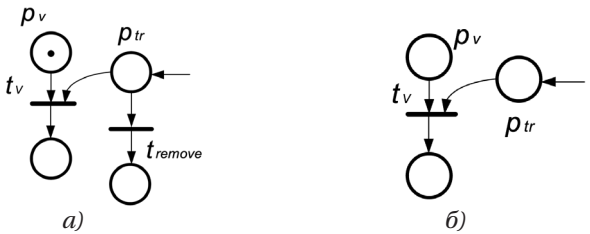
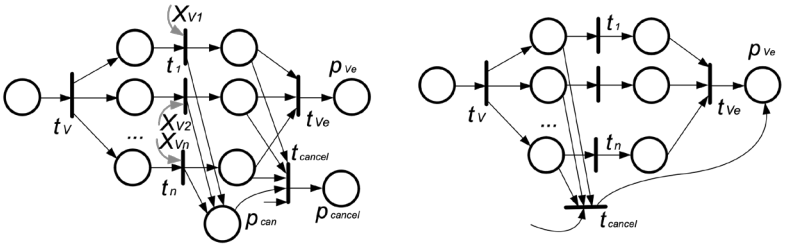


Рис. 22. Патерни з тригерами: *а* – патерн 23 «Перехідний тригер» та *б* – патерн 24 «Стійкий тригер»

Патерн 25 «Скасувати набір завдань» реалізується аналогічно варіанту (б) патерну 20 (рис. 19, б), оскільки дозволяє скасувати виконання не тільки взаємопов'язаних у процесі наборів завдань, але і незалежних між собою наборів завдань за рахунок представленого у патерні механізму точкового скасування завдання з вершини p_{cancel}, основаного на альтернативних процесах для виведення міток або управляючих векторах X_V та Y_V. Повноцінне застосування патерну 25 також

забезпечене у діаграмах діяльності UML, інші спеціалізовані мови та засоби моделювання забезпечують лише часткову підтримку цього патерну [57].

Патерн 26 «Скасувати багаторазову активність» (рис. 23,а) дозволяє скасувати подальшу обробку завдань при паралельному їх виконанні, які ще незавершені, завершені екземпляри завдань не скасовуються.

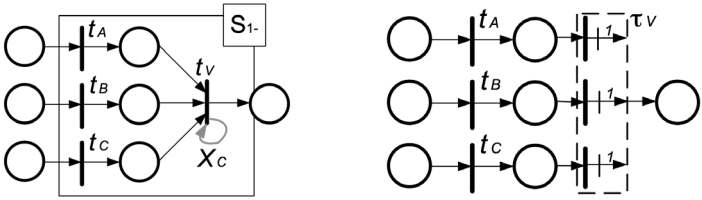


а) *б)*

Рис. 23. Патерни припинення багаторазової активності:
***а* – патерн 26 «Скасува-ти багаторазову активність»,**
***б* – патерн 27«Завершити багаторазову активність»**

При реалізації даного патерну з усіх вершин місць, які відображають виконані завдання, але не оброблені у синхронізуючій вершині переходу t_{Ve}, вилучаються всі мітки у вершину t_{cancel}, також і з вершин переходів мітки переходять у вершину місця p_{can}, а далі до вершини переходу t_{cancel}, результати скасування – у вершині місця p_{cancel}. Патерн 27 «Завершити багаторазову активність» (рис. 23, б) має відмінність від попереднього патерну у тому, що виконання завдань, які не завершились, завершується, але інші екземпляри не створюються – мітки з вхідних місць до вершин переходів $t_1 \ldots t_n$ виконання завдань вилучаються у вершину переходу t_{cancel}, і як тільки це відбувається, на наступному кроці роботи моделі потік управління може бути переданий наступному процесу через кінцеву вершину місця p_{Ve}.

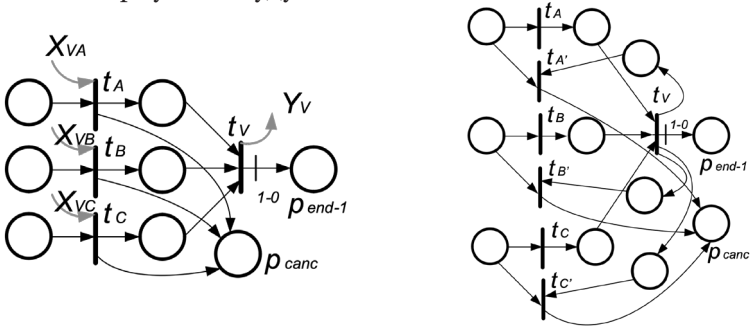
Патерн 28 «Блокуючий дискримінатор» (рис. 24) дозволяє реалізувати запуск процесу у вершині переходу t_V при надходженні мітки з будь-якої вхідної вершини, після чого передача міток від цієї гілки блокується до того часу, поки не відбудеться передача міток від інших гілок.



а) *б)*

Рис. 24. Патерн 28 «Блокуючий дискримінатор»

Патерн 29 «Скасовуючий дискримінатор» дозволяє описати випадок, коли для подальших процесів потрібен результат завдання, яке швидше за інші виконалось, виконання всіх інші завдань скасовується. Даний шаблон реалізований управляючою мережею Петрі (рис. 25, а) та класичною мережею Петрі (рис. 25, б). Дуга від вершини переходу t_V результатів завдання, яке перше виконалось, до вихідної вершини місця p_{end-1} позначена «1-0», що відображає, що лише перша мітка може бути передана по даній дузі у поточному сеансі роботи патерну, інші будуть скасовані.



а) *б)*

Рис. 25. Патерн 29 «Скасовуючий дискримінатор»

При застосуванні патерну 29 виникає проблема [58] у випадку, коли процес слабко структурований або існує зв'язок процесів у дискримінаторі з іншими процесами, які не потребують скасування. У першому випадку шаблон можна реалізувати шляхом вказування процесів для скасування через вищенаведені механізми (умовно структуруючи такий процес), у другому випадку цей патерн не може бути застосований.

Патерн 30 «Структуроване часткове з'єднання» (рис. 26) передбачає, що його застосуванню передувало «Паралельне розщеплення» (патерн 2) чи «Множинний вибір» (патерн 6), кількість активних паралельних гілок у яких більша ($n > m$), ніж потрібно для виконання наступних завдань. Для відбору m перших завдань з n активізованих застосовується патерн, який накопичує перші m міток у вершині місця p_V і запускає на виконання завдання у вершині переходу t_V, яке передає результати наступному процесу через вершину місця p_{v2}. Інші мітки, що надходять пізніше, спрямовуються до вершини переходу t_e і результати екземплярів завдань, що їм відповідають, вилучаються у вершині переходу t_e та у подальшому не впливають на наступні процеси.

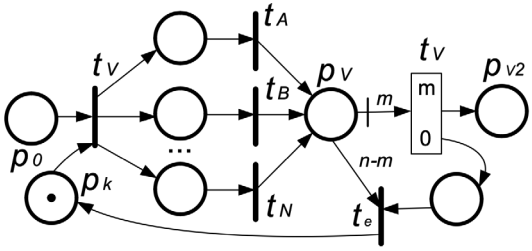


Рис. 26. Патерн 30 «Структуроване часткове з'єднання»

Патерн 31 «Блокування часткового з'єднання» (рис. 27) відображає можливість паралельного виконання у n гілках лише по одному екземпляру певного завдання (в початковому

стані у вершині p_0 є N міток, що відповідає кількості виконуваних завдань), результати виконаних m перших завдань будуть впливати на наступні завдання (мітки передаються через вершину місця p_{v2}), а результати *(n-m)* завдань будуть блокуватися (у вершині місця t_e).

Наступні m завдань (по одному екземпляру кожного завдання) можуть виконатися тільки після передачі наступному процесу результатів поточного виконання патерну та відновленню розмітки у вершини місця p_0 після блокування *(n-m)* завдань.

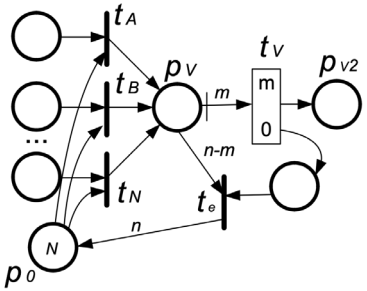


Рис. 27. Патерн 31 «Блокування часткового з'єднання»

Патерн 32 «Скасування часткового з'єднання» (рис. 28) відображає можливість паралельного виконання у n гілках по одному екземпляру певного завдання, результати виконаних m перших завдань будуть впливати на наступні завдання (мітки передаються через вершину місця p_{v1}), а виконання *(n-m)* завдань будуть скасовані до їх закінчення.

Патерн 33 «Узагальнене and-з'єднання» (рис. 29) відображає злиття певної кількості гілок у одну, коли завдання у кожній гілці виконане і у вершинах місць, вхідних до вершини переходу t_V є достатня кількість міток його активізації. У залежності від вирішуваної задачі вага дуг m може змінюватись ($m = 1 \ldots k$).

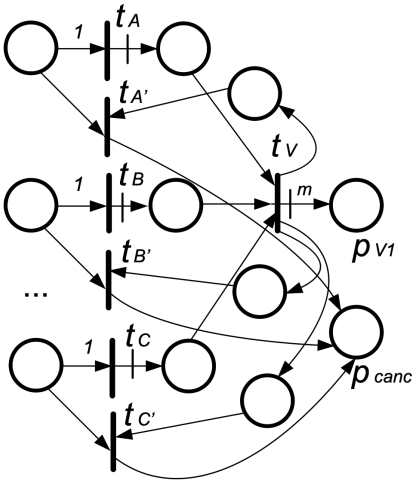


Рис. 28. Патерн 32 «Скасування часткового з'єднання»

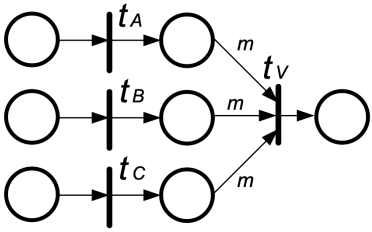


Рис. 29. Патерн 33 «Узагальнене and-з'єднання»

Патерн 34 «Часткове статичне об'єднання для кількох екземплярів» (рис. 30) описує утворення та одночасну обробку кількох екземплярів певного завдання, з яких результати m перших оброблених екземплярів мають впливати на подальші завдання, а інші $(n - m)$ – мають бути вилучені з патерну перед його наступним сеансом роботи. Відображення патерну 34 схоже з патерном 30, але має відмінність – у паралельних гілках обробляються екземпляри одного завдання, а не різні завдання, як у патерні 30.

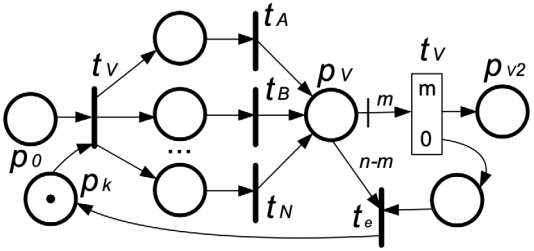


Рис. 30. Патерн 34 «Часткове статичне об'єднання для кількох екземплярів»

Патерн 35 «Скасування часткового об'єднання для кількох екземплярів» (рис. 31) надає можливість не чекати завершення виконання всіх екземплярів завдання, як у патерні 34, а припинити їх як тільки завершать виконання m перших екземплярів завдання. Він реалізований управляючою мережею Петрі (а) та класичною мережею Петрі (б) з конкретизацією конструкції переходів $t_{An} - t_{An'}$.

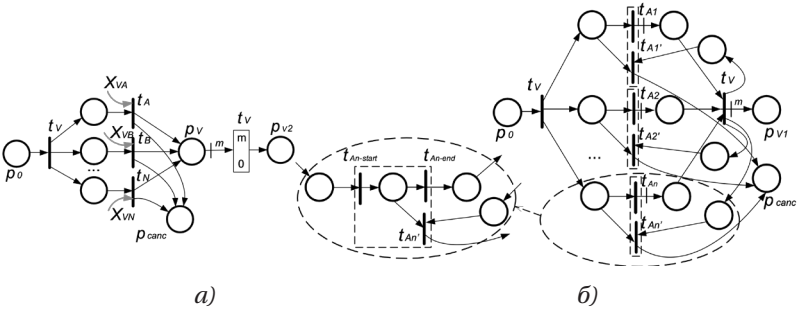


Рис. 31. Патерн 35 «Скасування часткового об'єднання для кількох екземплярів»

Патерн 36 «Динамічне часткове об'єднання для кількох екземплярів» (рис. 32) передбачає, що перед його виконанням не всі параметри паралельно виконуваних екземплярів завдан-

ня, які створюються, будуть відомі заздалегідь, наприклад, кількість екземплярів завдання, яка буде створена. У патерні 36, подібно до патерну 15, реалізована можливість створення додаткових екземплярів завдання, якщо така виникне. Але на відміну від патерну 15 у даному патерні реалізоване часткове об'єднання m перших оброблених екземплярів завдання, результати обробки інших екземплярів завдання вилучаються і не впливають на наступні завдання.

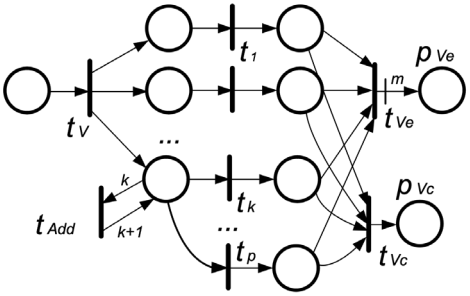
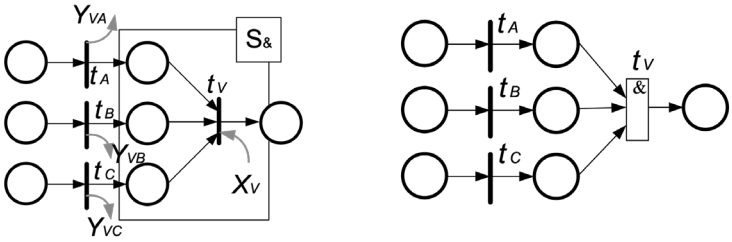


Рис. 32. Патерн 36 «Динамічне часткове об'єднання для кількох екземплярів»

Патерн 37 «Локальна синхронізація злиття» (рис. 33) проводиться за локальною інформацією про те, скільки гілок потребують синхронізації.



а) *б)*

Рис. 33. Патерн 37 «Локальна синхронізація злиття»

Патерн 37 може реалізовуватись засобами управляючих мереж Петрі (а), в якому інформація передається у вектор X_V з кожного локального вектора Y_{Vi}, або відображатись умовно (б). Також передаватись у вектор X_V (а) або у вершину t_V (б) інформація про потребу у синхронізації може з вектора Y_V патерну 6 «Множинний вибір».

Патерн 38 «Загальна синхронізація злиття» (рис. 34) відображає правила злиття паралельних гілок процесів, які використовуються у патернах 7 «Синхронізуюче з'єднання » та 37 «Локальна синхронізація злиття», а також враховують неможливість спрацювання будь-якої гілки у злитті, зберігаючи працездатною модель. У роботі [59] вказується на проблему при реалізації даного патерну у випадку нелокальної семантики, натомість можлива його реалізація засобами локальної семантики.

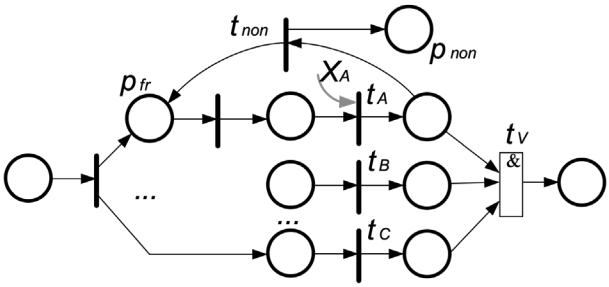


Рис. 34. Патерн 38 «Загальна синхронізація злиття»

Саме такий варіант реалізації патерна 38 наведений на рис. 34, в якому у випадку визначення неможливості активізації певної гілки, вершина переходу t_{non} передає дві мітки, одна з яких переходить у вершині місця p_{non} і означає існування гілки, яка не може спрацювати, а друга – у вершину p_{fr}, цю мітку називають «несправжньою» міткою, вона дозволяє коректно завершити виконання процесу злиття. Даний патерн складний у реалізації і не підтримується більшістю спеціалізованих

мов (включаючи UML) та засобів моделювання. Патерн 38 використовується у моделях з неструктурованими та паралельними процесами, а також з довільними циклами.

Патерн 39 «Критичний розділ» (рис. 35) реалізує алгоритм обробки завдань певним процесом, який може обробляти лише одне завдання в поточний час, інші завдання мають очікувати. Для доступу у критичний розділ використовується «контролююча» вершина місця p_k, яка забезпечує послідовність виконання завдань: завдання, яке першим досягло критичного розділу, обробляється, а завдання, яке надійшло пізніше – чекає, поки попереднє завдання обробиться повністю та залишить критичний розділ, після чого мітка знов переміститься у вершину p_k і наступне завдання може її використати для початку обробки у критичному розділі.

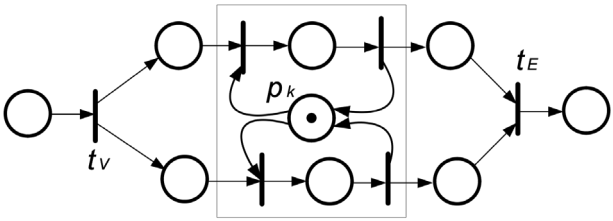


Рис. 35. Патерн 39 «Критичний розділ»

Патерн 39 «Критичний розділ» реалізується лише у частині спеціалізованих мов та засобів моделювання [57], які мають мютексні конструкції, оскільки потребує засобів явного опису станів – основних та контролюючих.

Патерн 40 «Interleaved-маршрутизація» (рис. 36) дозволяє описати, що кожне завдання визначеного набору завдань має виконатись один раз. Не має значення, в якому порядку ці завдання можуть виконуватись, єдине обмеження – вони мають виконуватись послідовно, що при реалізації патерну забезпечується з допомогою «контролюючої» вершини місця p_k.

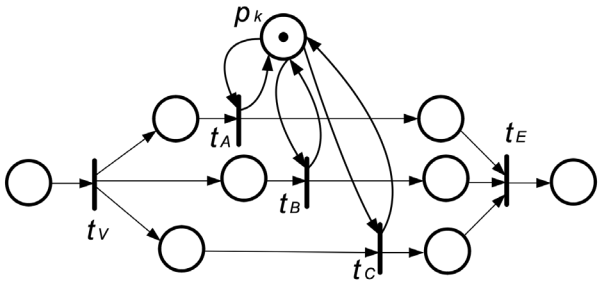
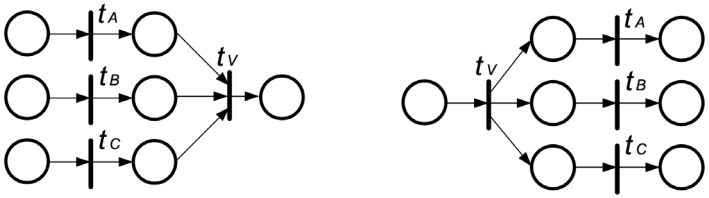


Рис. 36. Патерн 40 «Interleaved-маршрутизація»

Патерн 41 «Об'єднання ниток» (рис. 37, а) передбачає об'єднання кількох екземплярів певного завдання, які виконались паралельно, в один потік, патерн 42 «Розділення ниток» (рис. 37, б) описує утворення кількох екземплярів завдання для їх паралельної обробки. В обох патернах кількість екземплярів завдання повинна бути відома на етапі проектування, в іншому їх відображення побідне до патернів 3 «Синхронізація» та 2 «Паралельне розщеплення».



а) *б)*

Рис. 37. Патерн 41 «Об'єднання ниток» (*а*), патерн 42 «Розділення ниток» (*б*)

При розгляді 43 патернів для моделювання робочих процесів можемо помітити використання елементів безпечних, управляючих та оціночних мереж Петрі. Аналіз патернів у їх відображенні мережами Петрі проведемо у наступному пункті.

5. КЛАСИФІКАЦІЯ ПАТЕРНІВ ДЛЯ МОДЕЛЮВАННЯ СИСТЕМ З ПАРАЛЕЛІЗМОМ

Сформувати повний набір патернів, необхідних для моделювання систем з паралелізмом не є можливим, це неможливо зробити і, звузивши обєкт моделювання, для програмних систем [46], оскільки виділення повторюваних елементів у моделях програмних систем є процесом творчим, який дозволяє відокремлювати певні конструкції для шаблонізації з урахуванням бачення проекту системи конкретним фахівцем. Але розгляд базового набору патернів, що знайшли широке застосування при моделюванні управління потоками робіт [12, 57], дозволяє в результаті провести більш глибокий аналіз особливостей використання таких конструкцій з метою спрощення побудови та забезпечення функціоналу для дослідження динамічних властивостей моделей програмних систем. Для формування базового набору патернів у даній роботі аналізуються WF-патерни, які призначені для опису управління потоками робіт [46, 57]. В результаті їх аналізу та представлення мережами Петрі (табл. 1, додаток А) було виявлено, що деякі патерни можливо замінити одним патерном. Зокрема до таких патернів належать патерни утворення паралельних потоків, у яких виконуються завдання чи екземпляри одного завдання, а також патерни у ділянках злиття таких паралельних потоків.

При побудові моделей програмних систем насамперед потрібен інструмент, який у явній формі дозволить відображати процеси управління системою та відділити їх від потоків даних, які обробляються системою. Таке відображення дозволяє розробити інструментарій на основі інтерпретацій

мереж Петрі [12, 17, 60]. Так, у пропонованому інструментарії потоки управління передаються через дуги, відображаються у вершинах місць відповідною розміткою. При використанні управляючих векторів є можливість описати передачу сигналів управління від інших систем чи підсистем, що дозволяє не ускладнювати модель, а разом з тим наочно відображати та аналізувати особливості її функціонування.

Ці важливі властивості інструментарію моделювання добре сполучаються з сучасними компонентно-орієнтованими технологіями розробки програмних систем [60, 61], зокрема інструментарій на основі мереж Петрі природнім чином (за своїм визначенням) підтримує здатність модулів системи відновлювати свій початковий стан (властивість збережуваності [15], WF-інтерпретація мереж Петрі побудована саме за цією вимогою), а також дозволяє використовувати один і той же компонент (модуль) багаторазово, забезпечуючи гнучкість на масштабованість проекту.

При описі патернів не завжди вдавалося застосовувати певну інтерпретацію мереж Петрі, наприклад, тільки оціночні чи тільки безпечні мережі Петрі. Але при відображенні тих патернів, які побудовані за правилами WF-мереж, тобто мають одну вхідну та одну вихідну вершину місця, враховувалась властивість збережуваності, тобто якщо у вхідній вершині місця знаходиться одна мітка, то після повного циклу відпрацювання патерну у вихідній вершині теж має вміщуватись одна мітка (наприклад у патерні 15). Властивість збережуваності для інших патернів можливо перевірити тільки при побудові з них моделі, в який дотримаються правила побудови WF-мереж.

Для аналізу описані у попередньому розділі патерни розділимо на дві групи (додаток Б), при цьому збережемо нумерацію, яка застосовувалась у WF-патернах для зручності аналізу. До першої групи віднесемо патерни з однією вхідною та однією вихідною вершинами, назвемо їх «PN-патерни у послідовностях». Таких патернів тринадцять, це патерни 1, 9, 10, 13, 14, 15, 17, 21, 22, 30, 34, 39, 40 [46].

До другої групи патернів включимо патерни, які мають різне число вхідних вершин, назвемо їх «PN-патерни у паралельних структурах». У другій групі патернів можна виділити такі три підгрупи:

2.1) патерни, залежні від зовнішніх впливів (патерни 19, 20, 23, 24, 25, 26, 27, 38*),

2.2) патерни у розгалуженнях (патерни 2, 4, 6, 12, 16, 42),

2.3) патерни у точках злиття потоків (патерни 3, 5, 7, 8, 11, 28, 29, 31, 32, 33, 35, 36, 37, 38*, 41, 43).

Патерн 38, який відмічений «*» належить і до першої і третьої підгруп другої групи. Патерн 18 не увійшов у попередні групи чи підгрупи, він відображає вплив одного процесу на інший в середині моделі. Ці патерни є виключеннями з наведеної класифікації.

Патерни першої групи досить просто вбудовуються у модель програмної системи, це може робитись, як показано на рис. 3 чи через додатковий зв'язуючи перехід (рис. 38), який виконує роль проміжної обробки (чи підготовки) даних.

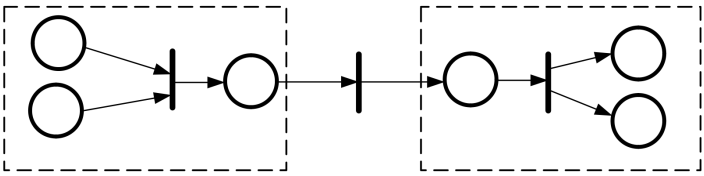


Рис. 38. Варіант з'єднання патернів у моделі.

Патерни цієї групи відображають послідовну обробку (патерн 1), вибір задачі, яка обробляється найшвидше (патерн 9), різні види циклів (патерни 10 та 21), рекурсію (патерн 22), часткове з'єднання (патерни 30 та 34), обробку у критичному розділі (патерн 39), псевдопаралельну маршрутизацію (патерни 17 та 40), а також моделювання паралельної обробки кількох екземплярів одного завдання (патерни 13, 14, 15). Патерни 13 та 14 мають однакове структурне відображення, вони

дозволяють описати обробку відзначеної заздалегідь кількості екземплярів одного класу. Розрізняються вони тільки параметрами, які задаються для певних вершин патерну, тому можна вважати їх одним патерном «Кількість екземплярів з встановленими характеристиками», у якому характеристики по створенню визначеної кількості екземплярів будуть відображатись структурно (патерн 13), а характеристики по часу відпрацювання завдань (активностей) – задаватись у вигляді часових обмежень (умов) у відповідних вершинах переходів. Патерн 15 схожий на 13 та 14 патерни, але дозволяє утворювати нові екземпляри процесу під час обробки екземплярів завдань цього процесу, про які не було відомо до його початку. Таким чином, цей патерн має додатковий механізм утворення непередбачених заздалегідь патернів і має відображатися окремою структурою, його назва 15* «Кілька екземплярів зі змінюваними характеристиками».

Патерни 17 та 40 теж мають спільні риси, у них кожне завдання обробляється послідовно і така обробка забезпечується контролюючою вершиною місця, хоча самі завдання розташовані у паралельних гілках (іноді у паралельних гілках послідовно кілька завдань). Оскільки механізм контролю псевдопаралельної обробки однаковий і патерн 40 є частковим випадком патерну 17, то ці патерни також можна структурно відобразити у вигляді патерну 17* «Псевдопаралельна маршрутизація».

При аналізі патернів 30 «Структуроване часткове з'єднання» та 35 «Часткове статичне об'єднання для кількох екземплярів» злиття результатів перших m виконаних завдань чи екземплярів завдання відбувається за однаковими правилами, інші результати ($n - m$) завдань чи екземплярів завдання ігноруються. Будемо відображати ці патерни одним патерном 34* «Часткове з'єднання».

Патерн 39 «Критичний розділ» є також різновидом даних патернів, але його краще відображати окремою конструкцією

патерна для зручнішого наочного аналізу моделі. Таким чином у першій групі патернів можемо залишити вісім патернів з десяти: 1, 9, 10, 13* (13-14), 15*, 17* (17, 40), 21, 22, 34* (30 та 34), 39* (39 та 40), серед яких об'єднані патерни, та патерни зі зміною назви:

1) паетрн 13* «Кілька екземплярів з апріорними характеристиками» замість патернів 13 та 14;

2) патерн 15* «Кілька екземплярів зі змінюваними характеристиками» замість патерну 15 «Кілька екземплярів без апріорних знань про час виконання»;

3) патерн 17* «Псевдопаралельна маршрутизація» замість патернів 17 та 40;

4) патерн 34* «Часткове з'єднання» замість патернів 30 та 34;

5) патерн 39* «Критичний розділ» замість патернів 39 та 40.

У другій групі зібрані патерни, які часто застосовуються для моделювання паралельних чи конкуруючих процесів. Такі процеси можуть мати перебіг у певній досліджуваній моделі (патерни другої та третьої підгруп) та при моделюванні зовнішніх впливів на модель, або впливів одних модулів системи на інші, що часто використовується при проектуванні та створенні сучасних програмних систем за об'єктно- чи компонентно-орієнтованими технологіями розробки [12, 61]. Це патерни для опису розгалужень та злиття потоків (з синхронізацією та без синхронізації).

У першій підгрупі другої групи патернів містяться конструкції, функціонування яких залежить від зовнішніх впливів. Патерн 19 «Скасувати завдання» спрацьовує при надходженні управляючого сигналу (чи відповідної мітки) з інтерфейсів інших модулів чи від інших завдань, які передбачають таке скасування.

Патерн 20 «Скасувати процес» та патерн 25 «Скасувати набір завдань» реалізовані у двоваріантному вигляді, який дозволяє припиняти виконання запущеного на виконання завдання і переходити до наступних завдань при надходжен-

ні відповідного управляючого сигналу. Для їх відображення у моделі можливо використовувати один патерн (20*), різниця використання буде полягати у тому, що з елементів процесу, який виконується, будуть вилучатися всі мітки активних завдань, а дуги для скасування бути проведені до всіх завдань, натомість при скасуванні набору завдань також мітки будуть вилучатись з активованих процесів, для яких будуть передбачені двоваріантні конструкції, передбачені у патерні, та проведені відповідні дуги (це будуть дуги до вибраних завдань).

Патерни 23 «Перехідний тригер» та 24 «Стійкий тригер» можливо представити одним патерном, який буде потребувати простого налагодження. Якщо патер буде відображати звичайний тригер, який може очікувати мітку з інших процесів, то його можна застосовувати по замовчуванню, управляючий вектор матиме нульове значення, інакше, якщо потрібно відобразити перехідний тригер, який не буде очікувати закінчення іншого процесу, то управляючий вектор прийматиме значення «1» (рис. 39).

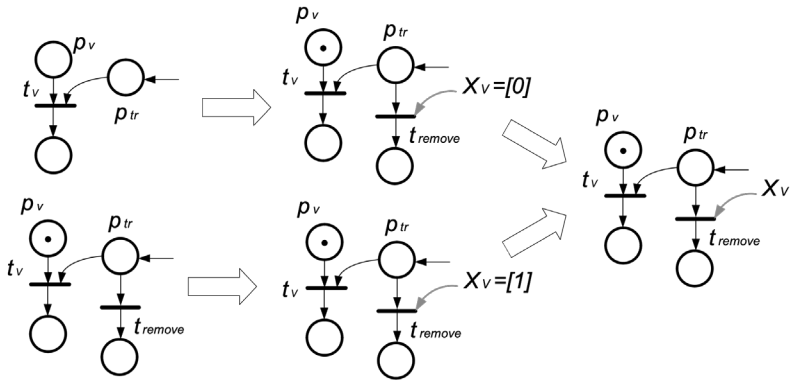


Рис. 39. Патерн 23* «Тригер», похідний від патернів 23 «Перехідний тригер», 24 «Стійкий тригер»

Патерни 26, 27 та 38 мають особливості, які відображаються в їх конструкціях, їх недоцільно об'єднувати. Таким чином, у другій групі патернів у першій підгрупі вісім патернів можливо замінити шістьма патернами (19, 20* (20 та 25), 23* (23 та 24), 26, 27, 38):

1) патерн 20* «Скасувати кілька завдань» замість патернів 20 та 25;

2) патерн 23* «Тригер» замість патернів 23-24 перехідного-стійкого тригерів.

У другій підгрупі другої групи особливо велика кількість патернів описує злиття потоків (дев'ятнадцять патернів), що обумовлено численними випадками, які потребують особливих правил керування такими потоками. При аналізі даних правил виявлені подібні, таким чином кількість патернів цієї підгрупи можна зменшити, згрупувавши їх за подібними правилами, які можна реалізувати на одній структурній конструкції патерну. Так для патернів 2 «Паралельне розщеплення» та 6 «Множинний вибір», в яких відбувається вибір всіх або кількох наступних гілок для виконання завдань, можливо такий вибір відобразити у єдиному за структурою патерні, який матиме на кожен описаний випадок відповідний управляючий вектор (рис. 40). Якщо всі елементи вектора X_V є ненульовими (налагоджується за замовчуванням), то маємо відображення патерну 2 «Паралельне розщеплення», якщо хоч один з компонентів вектора X_V буде нульовим, то реалізований варіант відображення патерну 6 «Множинний вибір». Можливий також варіант, коли тільки один компонент вектора X_V буде ненільовим, тоді це фактично буде відображення патерну 4 «Виключний вибір», але таке відображення патерну 4 складніше сприймається для візуального аналізу, ніж його основне відображення (рис. 4), тому основне відображення патерну 4 є більш пріоритетним при побудові моделі. Також на зображенні патерну 6 (рис. 40) показаний вектор Y_V, який передає інформацію для відповідного патерну злиття потоків.

Патерн 42 «Розділення ниток» фактично функціонує так, як патерн 2, за виключенням того, що його застосовують для екземплярів одного завдання, але оскільки правила його виконання однакові з патерном 2, то він є надлишковим, тобто замість нього застосовується патерн 2. Таким чином для патернів 2, 6, 42 будемо використовувати конструкцію патерну 2 та загальну назву цих патернів 2* «Кероване розщеплення» (рис. 40).

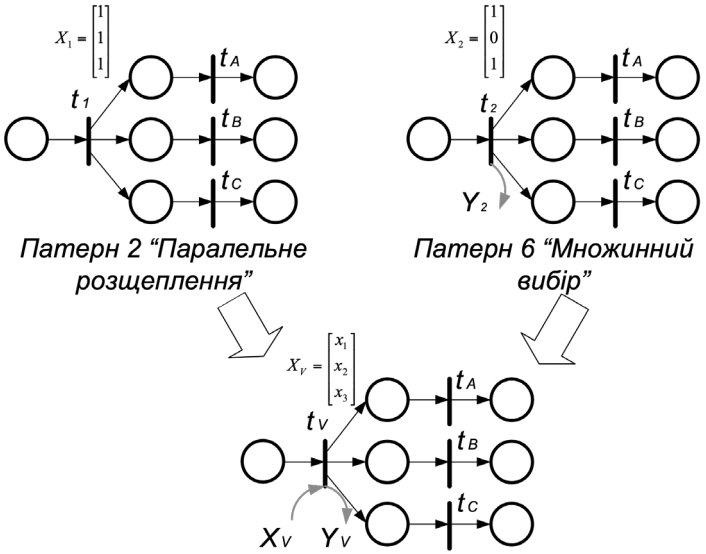


Рис. 40. Патерн 2* «Кероване розщеплення» з управляючим вектором

При аналізі патернів 4 та 16 теж бачимо багато спільного, розгалуження в них починаються від вершини місця і передбачають вибір однієї з гілок. Різниця у даних патернах полягає у тому, що у патерні 4 «Виключний вибір» вибір однієї з гілок відбувається «миттєво», тобто у наступний момент модельного часу, за умовою, яка виявилась істинною, а у патерні 16

«Відкладений вибір» вибір гілки відбувається через певний час очікування, який встановлюється для цього патерну (у налаштуваннях моделі) чи передається з іншого елементу моделі (як керуючий сигнал).

Якщо передбачити у властивостях вершини місця p_1 патерну 16 можливість задавати параметри управляючого вектора X_{p1} у вигляді умов, які обмежують час очікування (така мережа Петрі буде сполучувати у собі властивості безпечних і часових чи оціночних і часових мереж Петрі), то можливо патерни 4 та 16 представити одним за структурою патерном з відповідними характеристиками ключової вершини місця у розгалуженні (рис. 41).

Так у випадку моделювання патерну 4 «Виключний вибір» достатньо буде задати параметри вектора X_{pi} (який має розмірність, що відповідає кількості вихідних дуг), він вміщує умови вибору для кожної гілки, як у класичному варіанті патерну. Якщо потрібно відобразити патерн 16, то у векторі X_{pi} задаємо в умові час очікування (для конкретної гілки), а також умови вибору інших вихідних гілок, які перевіряються, коли час очікування збіг. Такі налаштування можливо задати, якщо передбачити, що кожна компонента вектору X_{pi} буде містити не одну, а кілька сполучуваних умов.

Вплив на даний патерн зовнішніх умов, наприклад, коли наслідки виконання завдання у іншому модулі набули певних значень, можливо передати з допомогою управляючих векторів $X_A \ldots X_C$ до вершин переходів. Ці вектори задаються за потреби, у стані «по замовчуванню» вони не впливають на режим спрацьовування вершин переходів, тобто функціонування моделі відбувається за правилами оціночних чи безпечних мереж Петрі. При налагодженні вхідних управляючих векторів $X_A \ldots X_C$ в них задають певні параметри, наприклад «в очікування сигналу» від процесу (ідентифікатор процесу), або час очікування (у вигляді умови), тоді конструкція буде відображати патерн 16.

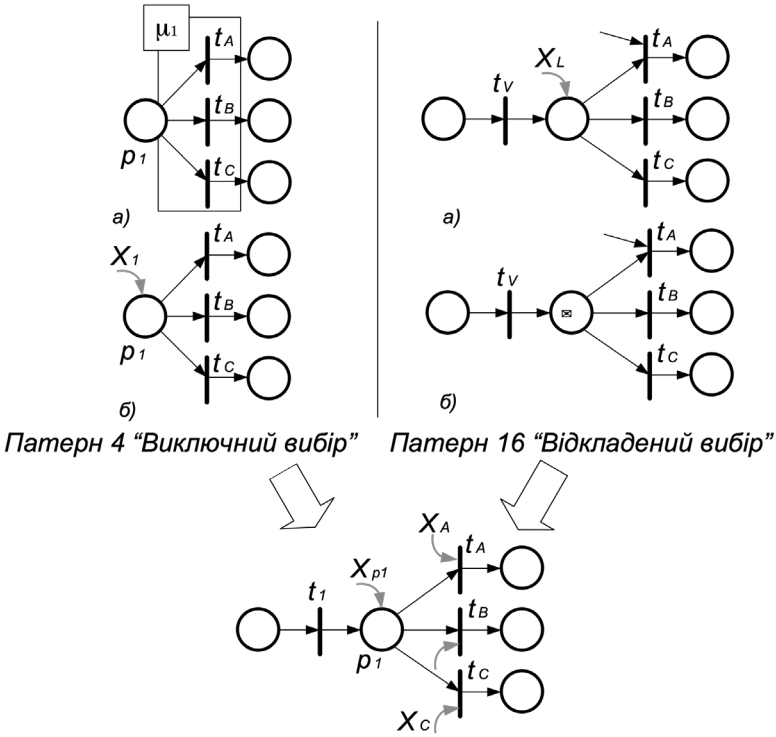


Рис. 41. Керований патерн виключного вибору у розгалуженні

Механізм очікування, реалізований у даному патерні також буде корисним при реалізації патерна 38 «Загальна синхронізація злиття», оскільки якщо у злитті не спрацювала певна гілка і немає мітки у відповідній вершині місця, то можливо доцільно застосувати механізм обмеження часу очікування події (сигналу), тоді буде можливим завершити роботу моделі коректно.

Таким чином, у другій підгрупі другої групи патернів шість патернів у розгалуженні можливо замінити трьома патернами (2* (2, 6 та 42), 12 та 16* (4 та 16)):

1) патерн 2* «Кероване розщеплення» (рис. 4.38) замість патернів 2, 6 та 42;

2) патерн 12 «Кілька екземплярів без синхронізації»;

3) патерн 16* «Керований виключний вибір» (рис. 4.39) замість патернів 4 та 16.

При аналізі патернів злиття (з синхронізацією чи без синхронізації), які об'єднані у третю підгрупу другої групи патернів, теж є конструкції, які схожі за структурним відображенням, але потребують певного налагодження елементів (рис. 42, а). Це, насамперед, патерни 3, 7, 33 та 41, для їх коректної роботи всі паралельні гілки повинні завершити виконання задач у активізованих гілках і передати інформацію про гілки, які були активізовані у вершину злиття (через управляючий вектор X_V) на відміну від процесів у разі застосування патерну 38, де передбачено, що деякі з активізованих процесів можуть не завершитись. Саме 38 патерн дозволяє описати правила коректного завершення всієї модель (чи окремого модуля моделі) у такому випадку. Патерн 37 є окремим випадком патерну 38, тобто може бути представлений конструкцією цього патерну.

Якщо проаналізувати патерни 5 «Просте з'єднання», 8 «Множинне з'єднання» та 43 «Явне припинення», то можна помітити, що їх теж структурно відображає однаковий патерн 5* «Неповне з'єднання» (рис. 42, б), ключова вершина переходу якого τ_V може не налагоджуватись або потребує налагодження. Ці патерни характеризуються тим, що вершина переходу τ_V, в якій зливаються потоки, може бути активована міткою з будь-якого з них чи з кількох з них. Якщо вектор X_τ не заданий, то патерн функціонує як «Просте з'єднання», якщо вектор X_τ заданий – як «Множинне з'єднання». Злиття потоків у патерні «Явне припинення» може проводитись за правилами будь-якої з цих конструкцій.

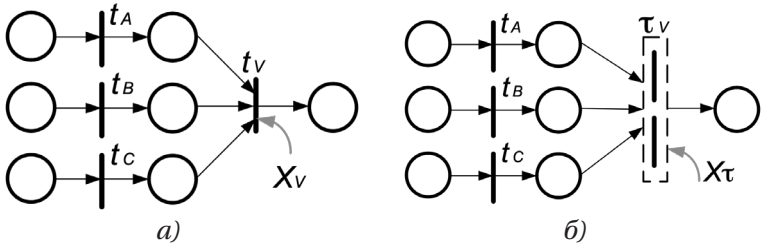


Рис. 42. Патерни злиття:
***а* – 3* «Синхронізуюче з'єднання», *б* – 5* «Неповне з'єднання»**

Патерни 9 «Дискримінатор» та 28 «Блокуючийо дискримінатор» є дещо відмінними за функціонуванням, так, патерн 9 спрацьовує під дією мітки з гілки, завдання якої виконалось першим (інші ігноруються), а у патерні 28 послідовне спрацьовування вершини злиття τ_V відбувається від мітки з кожної з гілок у з'єднанні. Тому зображати ці патерни однією конструкцією недоцільно. Патерн 28 «Блокуючий дискримінатор» можна представити, наприклад, як відображено на рис. 43.

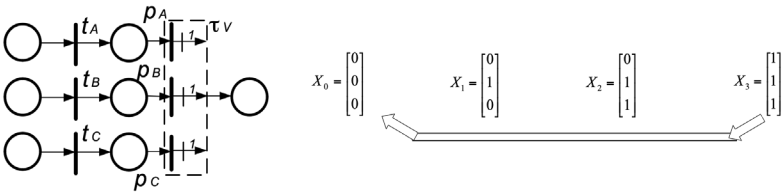


Рис. 43. Варіант відображення патерну 28
«Блокуючий дискримінатор»

Функціонування патерну 28 полягає у тому, що при спрацьовуванні [26] вершини макропереходу τ_V від мітки однієї з вхідних вершин місць дана гілка блокується до того часу, коли з кожної сусідньої гілки у з'єднанні не надійде по одній мітці для спрацьовування цього макропереходу, після чого в управляючому векторі всі значення скидаються у нуль.

Для патернів 32 «Скасування часткового з'єднання» та 35 «Скасування часткового об'єднання для кількох екземплярів» механізм скасування окремих завдань чи окремих екземплярів одного завдання теж однаковий, тому будемо відображати їх патерном 35* «Скасування часткового з'єднання».

Таким чином, у третій підгрупі другої групи патернів шістнадцять патернів злиття потоків можливо замінити дванадцятьма патернами (3* (3, 7, 33 та 41), 5* (5, 8 та 43), 11, 28, 29, 31, 35* (32 та 35), 36, 38* (37 та 38), серед яких об'єднані патерни::

1) патерн 3* «Синхронізуюче з'єднання» (рис. 4.41, а) замість патернів 3, 7, 33 та 41;

2) патерн 5* «Неповне з'єднання» (рис. 4.41, б) замість патернів 5, 8, та 43;

3) патерн 35* «Скасування часткового з'єднання» замість патернів 32 та 35;

4) патерн 38* «Загальна синхронізація» замість патернів 37 та 38.

Таким чином, перелік PN-патернів, які застосовують при моделюванні програмних систем з паралелізмом наведений у таблиці 1 (додаток А), графічне представлення класифікації PN-патернів – у додатку Б.

6. ПРИКЛАД ЗАСТОСУВАННЯ ПАТЕРНІВ ПРИ ПОБУДОВІ МОДЕЛІ ПРОГРАМНОЇ СИСТЕМИ

Побудова моделі системи з паралелізмом з патернів та інших елементів проводиться за методом структурної безконфліктності [15] для зменшення кількості помилок. Використання патернів дозволяє прискорити побудову моделі та надає можливість використовувати відлагоджені рішення, що має сприяти зменшенню кількості логічних помилок у побудованих моделях.

Для зручного використання патернів впродовж формування моделі, при встановленні параметрів їх елементів для уникнення взаємоблокувань від зовнішніх систем, передбачено налаштування «за замовчуванням». Такий підхід до налаштувань дозволяє перевірити працездатність системи при базових налаштуваннях, а також дозволяє звернути увагу на налагодження управляючих векторів, які відповідають за завершення часткових процесів у моделі.

На рис. 44 побудована модель системи керування реабілітацією хворих після серцево-судинних захворювань, які потребують контролю динамічних навантажень на тренуваннях ($p_{16} - t_{15} - p_{17} - t_{16}$) в залежності від основних медичних показників ($p_3 \dots p_{13}$). Модель побудована з використанням патернів (табл. 1, додаток А) 2* «Кероване розщеплення», 17* «Псевдопаралельна маршрутизація», 21 «Структурований цикл», 28 «Блокуючий дискримінатор» та інших конструкцій комбінованих мереж Петрі.

Для аналізу моделі при імітаційному моделюванні задана початкова розмітка [26], яка є ненульовою у двох

вершинах: $\mu_0(p_1) = 1$ та $\mu_0(p_{23}) = 1$. У кінцевій розмітці моделі ненульовими є вершини місць p_{23} та p_{33}: $\mu_{end}(p_{23}) = 1$ та $\mu_{end}(p_{33}) = 1$.

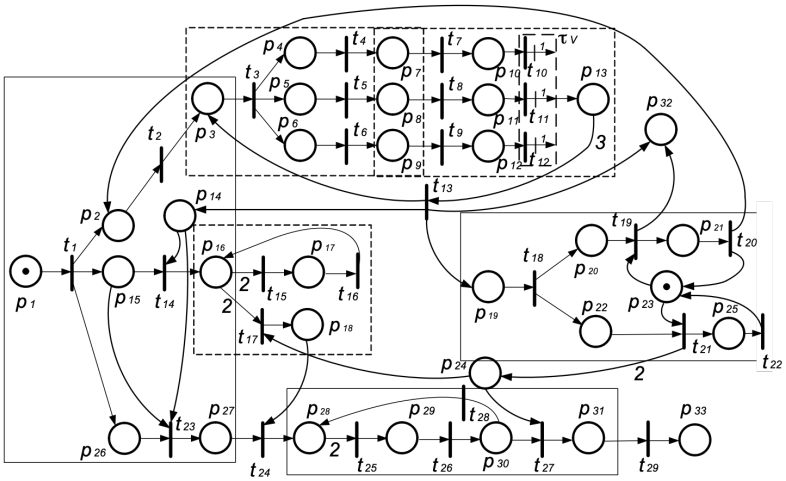


Рис. 44. Модель системи керування реабілітацією хворих після серцево-судинних захворювань на основі PN- патернів

При імітаційному моделюванні побудованої моделі системи керування реабілітацією хворих після серцево-судинних захворювань підтверджена коректність функціонування моделі при контролі початкової та кінцевої розміток, зокрема у початковій розмітці розмічена початкова вершина p_1 та контролююча вершина p_{23}, а у кінцевій розмітці одна з міток перемістилиась у вершину місця p_{33}, а інша повернулась у контролюючу вершину p_{23}. Таким чином, дотримані правила перевірки розміток, які є складовою методу структурної безконфліктності для моделювання систем з паралелізмом.

Сформована модель є підґрунтям для аналітичного дослідження динамічних характеристик модельованої систе-

ми з використанням комбінованого підходу до імітаційного моделювання систем з паралелізмом [5]. Зокрема вона дозволяє побудувати матрицю інцидентності W', сформувати основне рівняння мереж Петрі та на їх основі обчислити Т- та Р-інваріанти, за аналізом яких визначаються властивості обмеженості, повторюваності, несуперечливості, збережуваності [14]. За рангом матриці інцидентності та розмірністю інваріантів визначається властивість керованості, що є важливим показником передбачуваності функціонування досліджуваної системи. Таким чином, побудована модель системи керування реабілітацією хворих після серцево-судинних захворювань створює надійне підґрунтя для подальшого аналізу динамічних властивостей програмної системи.

ВИСНОВКИ

У монографії описані методологічні основи побудови інструментальних засобів для моделювання систем з паралелізмом, до яких належать програмні системи. Програмні системи розглянуті, як недетерміновані динамічні системи, які при моделюванні потребують аналізу структури та динамічних процесів, серед яких часто зустрічаються асинхронні паралельні та конкуруючі процеси.

Аналіз властивостей формальних мов, що породжуються мережами Петрі, дозволив обрати інтерпретації мереж Петрі для комбінованого інструментарію моделювання систем з паралелізмом. У пропонованому інструментарії виключені класи мереж Петрі з інгібіторними дугами, оскільки доказ основних властивостей для цих класів мереж Петрі з допомогою матричних методів аналізу є дуже ускладненим.

Описане підґрунтя для формування інструментальних засобів побудови компонентних моделей програмних систем зі застосуванням PN-патернів, що описують типові конструкції при моделюванні програмних засобів.

За основу формування комбінованих інструментальних засобів імітаційного моделювання систем з паралелізмом взяті формальні мови мереж Петрі L-типу з елементами мов G-типу. Для мов L-типу є розв'язуваною проблема бездефектності, що важливо при аналізі динамічних властивостей моделі. Цей клас мов мереж Петрі також є замкнутим до нескінченної конкатенації, що дозволяє забезпечити стійкість моделі до змін та масштабованість. Але мови L-типу мають досить жорсткі обмеження, що знижує потужність моделювання. Тому для зменшення обмеженості у пропонованому інструментарії застосовувались властивості класу вільних мов мереж Петрі, якими є мови G-типу, що однозначно описуються та аналізуються

матричними методами. Використання елементів формальних мов G-типу для розширення потужності опису моделей мовами L-типу, дозволило контролювати початкову та кінцеву розмітки. Також, на відміну від мов L-типу, запропоноване сполучення мов L_G-типу дозволяє створити інструментарій з можливістю розпочати моделювання системи з будь-якого етапу функціонування, що важливо при аналізі та тестуванні моделей програмних засобів з паралелізмом. Можливість застосування принципів побудови ієрархічних мереж Петрі, які також описуються мовами G-типу, дозволяє дотримуватись вимог стійкості до змін та масштабованості для програмних систем, що є важливими при розробці сучасних програмних систем.

У роботі розглянуті та проаналізовані 20 основних та 23 додаткових WF-патернів, які широко застосовуються для моделювання управління робочими процесами. WF-патерни описуються формальними мовами L-типу і є достатньо жорстко обмеженими. На їх основі сформовані еквівалентні PN-патерни, що дозволяють розширити можливості застосування WF-патернів, і належать до класу формальних мов L_G-типу, дозволяють коректно будувати та аналізувати моделі програмних систем, не порушуючи правила класичних мереж Петрі та їх інтерпретацій. При адаптації розглянутих WF-патернів до моделювання програмних систем 43 проаналізовані WF-патерни зведені до 28 PN-патернів, чому слугувало застосування правил та елементів управляючих мереж Петрі до оціночних та безпечних мереж з часовими характеристиками.

Пропоновані інструментальні засоби дозволяють однозначно відобразити недетермінований характер функціонування моделей ПЗ, використовуючи правила структурної безконфліктності, та створюють підґрунтя для автоматизованої перевірки моделі [5, 15] при побудові й моделюванні функціонування програмних засобів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Супруненко О. О. Парадигми імітаційного моделювання при дослідженні складних систем з паралелізмом. // Восточно-европейский журнал передовых технологий. – 2013. – № 5/4 (65). – С. 63–67.
2. Соммервилл И. Инженерия программного обеспечения, 6-е изд.: Пер. с англ. – М.: Изд. дом «Вильямс», 2002. – 624 с.
3. Святний В. А. Проблеми паралельного моделювання складних динамічних систем // Наукові праці Донецького національного технічного університету. Серія «Проблеми паралельного моделювання складних динамічних систем» – Донецьк: ДонНТУ. – 1999. – С. 6–14.
4. Kuzmuk V. V., Suprunenko O. A. The means for the description of information flows in dynamic models of medical hardware-software systems // Theoretical and Applied Science. – Vol. 15. – Nr. 7. – 2014. – PP. 11–18.
5. Suprunenko, O. Combined approach architecture development to simulation modeling of systems with parallelism. Eastern-European Journal of Enterprise Technologies, 4(4(112)), 2021. – 74–82. doi: https://doi.org/10.15587/1729-4061.2021.239212.
6. P. Bourque and R. E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society. Available at: www.swebok.org (accessed 11 January 2022).
7. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. – СПб.: БХВ-Петербург, 2005. – 400 с.
8. Boccara N. Modeling Complex Systems. – Springer-Verlag New York, Inc., 2004. – 397 p.
9. Панкова, Л. А. Способы создания универсального инструментария для компьютерного моделирования [Текст] / Л. А. Панкова, В. А. Пронина. // Проблемы управления (Control Sciences). – 2006. – № 6. – С. 2–5.
10. Лавріщева К. М., Слабоспицька О. О. Підхід до побудови об'єктно-компонентної моделі сімейства програмних продуктів // Проблеми програмування. – 2013. – № 4. – С. 14–24.
11. Чередніченко Ю.О. Аналіз компонентно-орієнтованих методів розробки програмного забезпечення для електронного бізнесу /

О. Ю. Чередніченко, М. Ю. Гонтар, О. В. Іващенко, М. А. Вовк // Вісник Вінницького політехнічного інституту. – 2018. – № 2. – С. 80–88.

12. Краліна Г. С., Смілій Е. Р. Компонентно-орієнтоване програмування сучасний підхід до розробки складних програмних систем // Тези IX міжнародної науково-практичної конференції «Інформаційно-комп'ютерні технології 2018». – Житомир: Вид. Житомирського держ. ун-ту. «Житомирська політехніка». – С. 27.

13. Брауде Э. Технология разработки программного обеспечения. – СПб: Питер, 2004. – 655 с.

14. Супруненко О. О., Онищенко Б. О., Гребенович Ю. Є. Аналітичний підхід при дослідженні властивостей графової моделі програмної системи // Праці міжнародної науково-практичної конференції «Математичне моделювання процесів в економіці та управлінні проектами і програмами» (ММП-2020), Коблево, 14–18 вересня 2020 р. – Харків: ХНУРЕ, 2020. – С. 110–113.

15. Супруненко О. О. Комбінований підхід до імітаційного моделювання динаміки програмних систем на основі інтерпретацій мереж Петрі. // KPI Science News (Наукові вісті НТУУ КПІ). – 2019. – № 5–6. – С. 43-53. (0,98 друк.арк.) doi: 10.20535/kpi-sn.2019.5-6.174596.

16. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. 2-е изд. – М.: Вильямс, 2002. – 528 с.

17. Питерсон Дж. Теория сетей Петри и моделирование систем. / Пер. с англ. – М.: Мир, 1984. – 264 с. (Peterson J. L. Petri net theory and modeling of systems. Prentice-Hall, 1981.)

18. Kopczynski E. Complexity of Problems for Communicative Grammars. 2010. Available at: https://arxiv.org/pdf/1003.4105.pdf (Accessed 11.12.2021).

19. Васильев В. В., Кузьмук В. В. Сети Петри, параллельные алгоритмы и модели мультипроцессорных систем – К.: Наукова думка, 1990. – 216 с.

20. Jancar P. Decidability questions for bisimilarity of Petri nets and some related problems. Proc. of STACS'94. Lecture Notes in Computer Science, 775, 1994, pp. 581–592.

21. Гломодза Д.К. Застосування методу інваріантів до аналізу кольорових мереж Петрі із дедлоками // Вісник НТУУ КПІ. Інформатика, управління та обчислювальна техніка, 2016. № 64. – С. 38-46. [Електронний документ]. Режим доступу: http://ekmair.ukma.edu.ua/

bitstream/handle/123456789/10870/ Hlomozda_Zastosuvannia_metodu_invariantiv.pdf. Перевірено 21.11.2021.

22. Кривий С.Л., Матвєєва Л.Е. Формальные методы анализа свойств систем. // Кибернетика и системный анализ. – 2003. – № 2. – С. 15–36.

23. Will van der Aalst, Kees van Hee. Workflow management: models, methods and systems. The MIT Press Cambridge, Massachusetts, London, England, 2002,359 p.

24. K. van Hee, N. Sidorova, M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best (Eds.) Application and Theory of Petri Nets 2003. Lecture Notes in Computer Science. Vol. 2679. P. 335–354. Springer-Verlag, Berlin, 2003.

25. W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and implementation of the YAWL system. 16th International Conference on Advanced Information Systems Engineering (CAiSE'2004). Copyright Springer Verlag. Available at: https://eprints.qut.edu.au/379/1/aalst_yawls.pdf (Accessed 04.01.22).

26. Кузьмук В. В., Супруненко О. О. Модифицированные сети Петри и устройства моделирования параллельных процессов: Монография. – К.: Маклаут, 2010. – 252 с.

27. Kappe T., Arbab F., Talcott C. L. A compositional framework for preference-aware agents. In: Proc. Workshop on Verification and Validation of Cyber-Physical Systems (V2CPS), 2016, pp. 21–35 (https://arxiv.org/pdf/1708.00072.pdf).

28. Лук'янова О.О. Про компонентне моделювання систем із паралелізмом. // Наукові записки НаУКМА, 2012. Т. 138. – С. 47–52.

29. Супруненко О.О. Комбінований підхід до імітаційного моделювання динаміки програмних систем на основі інтерпретацій мереж Петрі. // KPI Science News. – 2019. – № 5-6. – С. 43–53.

30. W. Reisig, G. Rozenberg. Informal Introduction to Petri Nets. Lectures on Petri Nets I: Basic Models. Advances in Petri Nets. Series: Lecture Notes in Computer Science. Vol. 1491. Springer-Verlag, Berlin, 1998. P. 1–12.

31. Suprunenko O. A. Combined tools simulation Workflow based graph models // Theoretical and Applied Science. – Vol. 23. – Nr. 3. – 2015. – PP. 153–158. doi: http://dx.doi.org/10.15863/TAS.2015.03.23.26.

32. W.M.P. van der Aalst. Three good reasons for using a Petri-net-based workflow management system. Eindhoven University of Technology. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?-doi=10.1.1.33.960&rep=rep1&type=pdf (Accessed 19.01.22).

33. Башкин В. А. О разрешимости бездефектности для сетей потоков работ с неограниченным ресурсом [Текст]/ В.А. Башкин, И.А. Ломазова // Моделирование и анализ информационных систем. – Т. 20, № 4. – 2013. – С. 23–40.

34. Ломазова И. А. Адаптивное и динамическое моделирование потоков работ на основе взаимодействующих сетей Петри / Методы и средства обработки информации. Труды третьей Всероссийской научной конференции. / Под ред. Л. Н. Королева. – М.: Издательский отдел факультета вычислительной математики и кибернетики МГУ. – 2009. – Т.2 – С. 32-37.

35. Нестеренко Б. Б., Новотарський М. А. Формальні засоби моделювання паралельних процесів та систем // Праці Інституту математики НАН України. – Т. 90. – Київ: Інститут математики НАН України, 2012. – 334 с.

36. Супруненко О. О. Розробка візуально-аналітичних засобів управління програмним проектом. // Восточно-европейский журнал передовых технологий. – 2012. – № 2/3 (56). – С. 46–49.

37. Кузьмук В. В., Супруненко О. А. Модифицированные сети Петри для формирования параллельных процессов в системах управления. // Восточно-европейский журнал передовых технологий. – 2010. – № 6/8 (48). – С. 50–53.

38. Кузьмук В. В., Супруненко О. А., Кузьмук А. В. Оценочные управляющие сети Петри. // Управління розвитком складних систем. – 2011. – № 8. – С. 25–27.

39. Вендеров А. М. Современные технологии создания программного обеспечения. [Электронный документ]. Режим доступа: http://citforum.ru/programming/application/program/index.shtml#v. Проверено: 03.08.2021.

40. Jancar P., Kucera A., Moller F. Simulation and Bisimulation over One-Counter Processes // Proc. of STACS'2000. 2000. LNCS 1770. P. 334–345.

41. Супруненко О. О. Блочний інструментарій моделювання динамічних процесів у системах з паралелізмом / Матеріали всеукраїнської науково-практичної конференції «Інформаційні та моделюючі технології» ІМТ-2015. – Черкаси: ЧНУ, 2015. – С. 58.

42. Shatz, Sol M., Shengru Tu, Tadao Murata and Sastry Duri. An Application of Petri Net Reduction for Ada Tasking Deadlock Analysis. IEEE Trans. Parallel Distrib. Syst. 7, 1 December 1996, PP 1307-1322.

43. Крывый С. Л. О вычислении минимального множества инвариантов сети Петри / С. Л. Крывый // Искусственный интеллект. – 2001. – № 3. – С. 199–206.

44. Вил ванн дер Аалст. Анализ процессов – мост между BI и BPM. // Открытые системы. – 2012. – № 2. [Электронный документ]. Режим доступа: http://www.osp.ru/os/2012/02/13014099/. Проверено 09.01.22.

45. Ломазова И. А. Об одном подходе к моделированию распределённых алгоритмов управления мультиагентными системами. / Труды Международной конференции «Интеллектуальное управление: новые интеллектуальные технологии в задачах управления» (ICIT'99) [Электронный документ]. Режим доступа: http://www.botik.ru/~raai/Resource/ICIT99.ru.shtml. Проверено 27.12.21.

46. N. Russell, A. H. M. ter Hofstede, W.M.P. van der Aals, N. Mulyar. Workflow Control-Flow Patterns: A Revised View. *BPM Center Report BPM-06-22* , BPMcenter.org. Available at: http://www.workflowpatterns.com/patterns/control/ (accessed 05 January 2022).

47. Супруненко О. О., Братко О. В. Case-засоби автоматизованого аналізу програм на основі мереж Петрі. // Вісник Черкаського державного технологічного університету. – 2009. – № 3. – С. 12–15.

48. W.M.P. van der Aalst, A.H.M. ter Hofstede, Kiepuszewski, B., et al. Workflow Patterns. Distributed and Parallel Databases 14 (3), 5-51, 2003. https://doi.org/10.1023/A:1022883727209.

49. Башкин В. А., Ломазова И. А. Эквивалентность ресурсов в сетях Петри. – М.: Научный мир, 2008. – 208 с.

50. Кузьмин Е. В., Вполне структурированные системы помеченных переходов [Электронный ресурс] / Кузьмин Е. В., Соколов В. А. – М.: ФИЗМАТЛИТ, 2005. – 176 с. – ISBN 5-9221-0598-1 – Режим доступа: http://www.studentlibrary.ru/book/ISBN5922105981.html. Проверено 09.01.2022

51. Кузьмук В. В., Супруненко О.О. Расширение функциональных возможностей алгоритмического аппарата сетей Петри при моделировании параллельных процессов / Кузьмук В.В., Супруненко О. О. // Электронное моделирование. – 2009. – Т.31. – №5. – С. 65–73.

52. Супруненко О. О. Моделювання процесів скасування та блокування задач при проектуванні програмних засобів. // Інформаційні моделюючі технології, системи та комплекси (ІМТСК-2021). Третя міжнародна науково-практична конференція, 27–28 травня 2021 р., Черкаси, Україна. – Черкаси: ЧНУ, 2021. – С. 19–22.

53. Зайцев Д. А., Инварианты функциональных подсетей. // Труды Одессой национальной академии святи им. А.С. Попова.- 2003. – № 4. – С. 57–63.

54. Кузьмук А. В., Кузьмук В. В., Супруненко О. А. Применение управляющих сетей Петри для моделирования параллельных процессов с многовариантным выбором. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб.наук.пр. – К.: Век+, – 2011.– №54. С. 26–30.

55. Михеев А., Орлов М. Перспективы WorkFlow-систем. Обзор. Часть 2. // PC Week. – 2004. – № 28 (442). – [Электронный документ]. Режим доступа: https://www.itweek.ru/idea/article/detail.php?ID=68049. Проверено: 21.01.2022.

56. Kiepuszewski, B., A.H.M. ter Hofstede, W.M.P. van der Aalst, Fundamentals of Control Flow in Workflows. Acta Informatica, 39(3), 143–209, 2003.

57. Control-flow patterns. [E-resource] Available at: http://www.workflowpatterns.com/patterns/control/index.php(Accessed 09.03.21)

58. W.M.P. van der Aalst. Exterminating the Dynamic Change Bug: A Concrete Approach to Support Workflow Change. // Information Systems Frontiers, 2001. P. 297-317.

59. Van der Aalst, W.M.P., J. Desel, E. Kindler. On the semantics of EPCs: A vicious circle. / In M. Rump and F.J. Nuttgens, editors, Proceedings of the EPK 2002: Business Process Management using EPCs, Trier, Germany, 2002. Gesellschaft fur Informatik. P. 71–80.

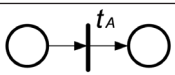
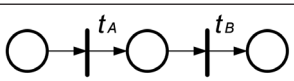
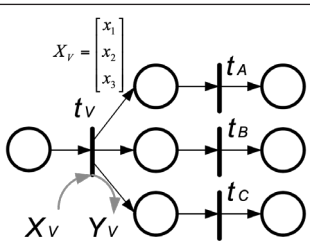
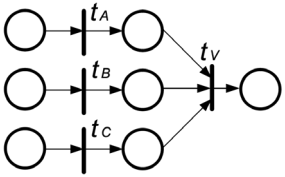
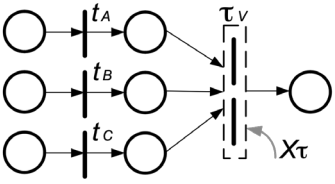
60. Онищенко Б. О., Супруненко О. О. Управляючі мережі Петрі, як засіб моделювання та автоматизованого аналізу алгоритмічних конструкцій. // Вісник запорізького національного університету. – 2009. – № 1. – С. 163–169.

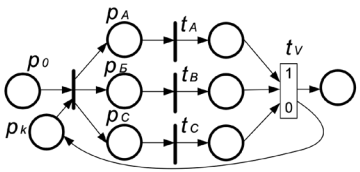
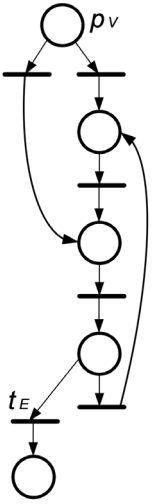
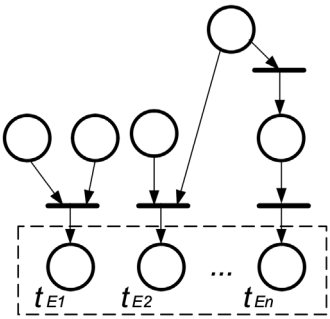
61. Чередніченко Ю. О. Аналіз компонентно-орієнтованих методів розробки програмного забезпечення для електронного бізнесу / О. Ю. Чередніченко, М. Ю. Гонтар, О. В. Іващенко, М. А. Вовк // Вісник Вінницького політехнічного інституту. – 2018. – № 2. – С. 80–88.

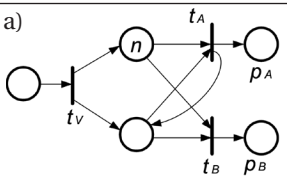
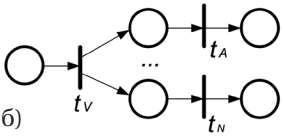
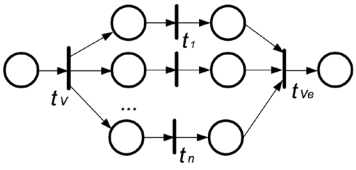
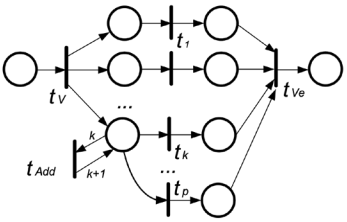
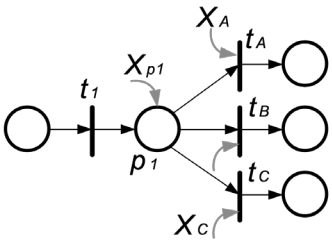
ДОДАТОК А

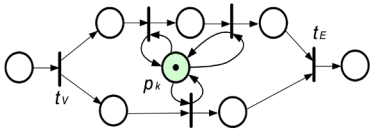
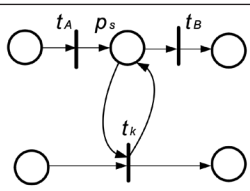
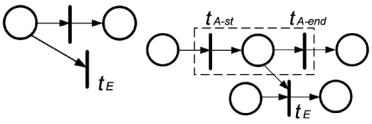
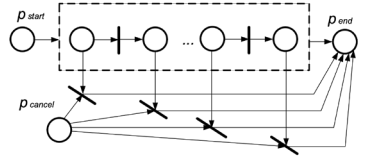
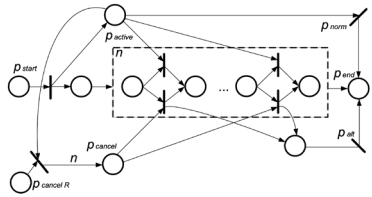
Таблиця 1

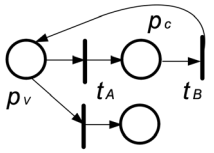
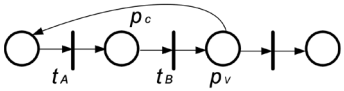
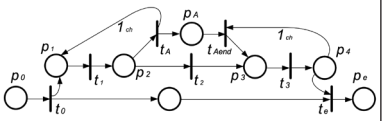
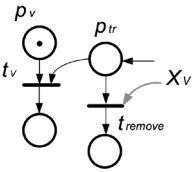
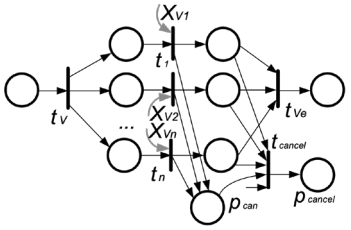
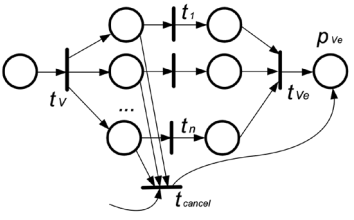
Графічне відображення PN-патернів комбінованого інструментарію для моделювання програмних систем з паралелізмом

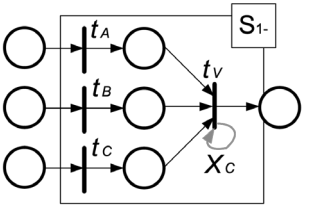
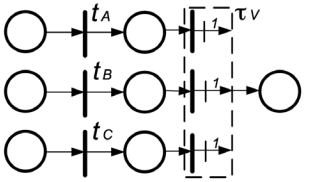
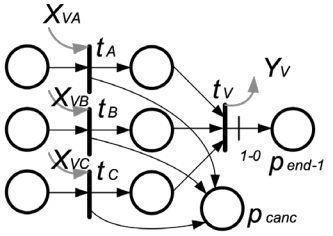
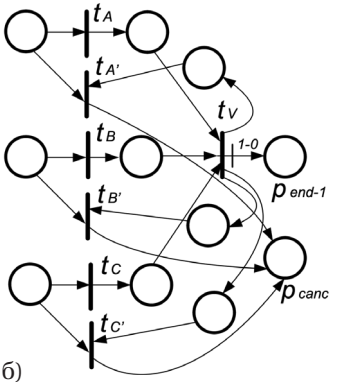
Назва PN-патерну	Відповідність WF-патерну [46]	Нотація PN-патерну
1	2	3
0. Дія	0. Дія	
1. Послідовність	1. Послідовність	
2*. Кероване розщеплення	2. Паралельне розщеплення. 6. Множинний вибір. 42. Розділення ниток	
3*. Синхронізуюче з'єднання	3. Синхронізація. 7. Синхронізуюче з'єднання. 33. Узагальнене and-з'єднання. 41. Об'єднання ниток	
5*. Неповне з'єднання	5. Просте з'єднання. 8. Множинне з'єднання. 43. Явне припинення	

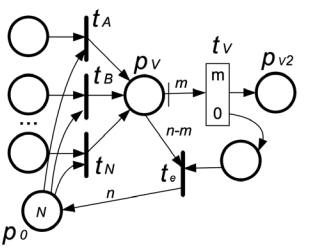
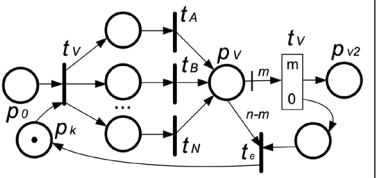
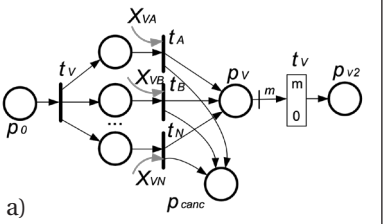
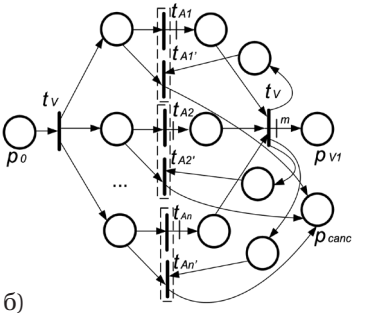
1	2	3
9. Структурований дискримінатор	9. Структурований дискримінатор	
10. Довільний цикл	10. Довільний цикл	
11. Неявне припинення	11. Неявне припинення	

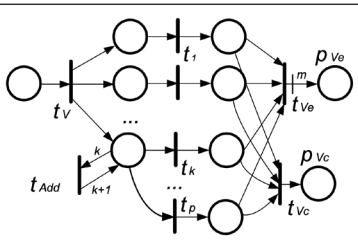
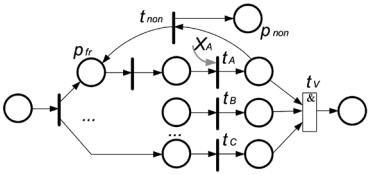
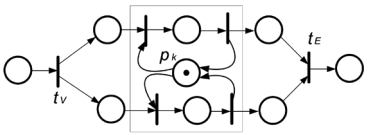
1	2	3
12. Кілька екземплярів без синхронізації	12. Кілька екземплярів без синхронізації	а)  б) 
13*. Кілька екземплярів із апріорними характеристиками	13. Кілька екземплярів із апріорними знаннями під час проектування. 14. Кількa екземплярів із апріорними знаннями про час виконання	
15*. Кілька екземплярів зі змінюваними характеристиками	15. Кілька екземплярів без апріорних знань про час виконання	
16*. Керований виключний вибір	4. Виключний вибір. 16. Відкладений вибір	

1	2	3
17*. Псевдо-паралельна маршрутизація	17. Interleaved-паралельна маршрутизація. 40. Interleaved-маршрутизація	
18. Етап (Milestone)	18. Етап (Milestone)	
19. Скасувати завдання	19. Скасувати завдання	 а) в)
20*. Скасувати кілька завдань	20. Скасувати процес. 25. Скасувати набір завдань	 а)  б)

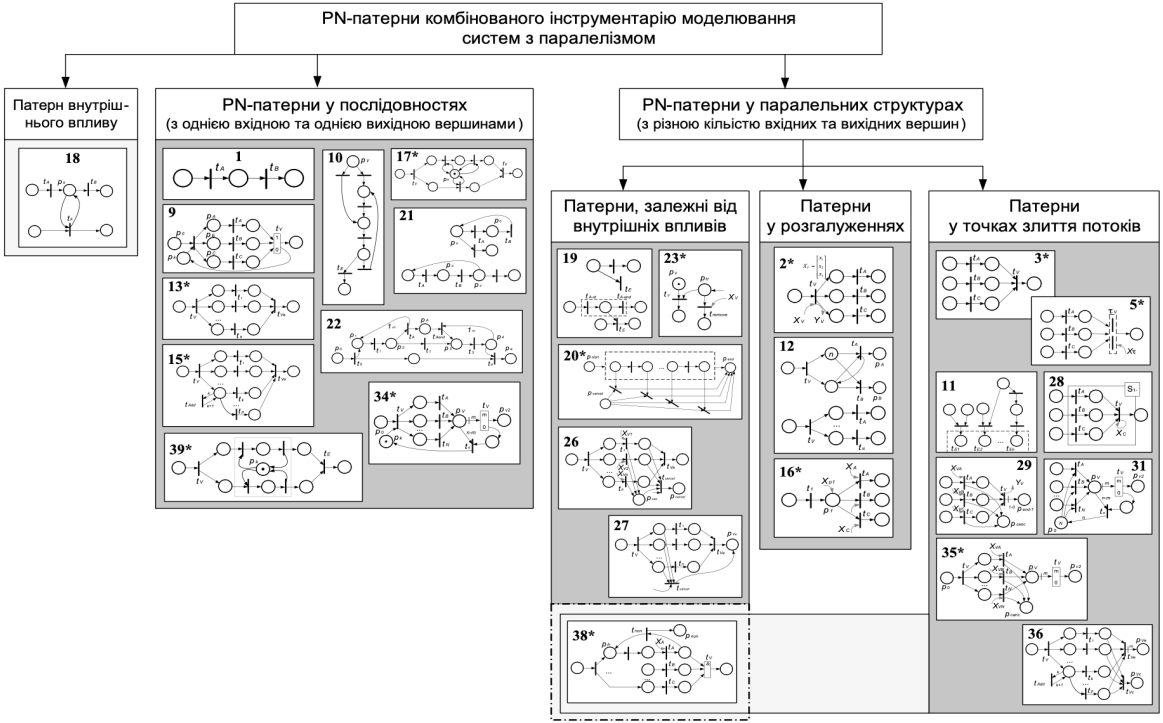
1	2	3
21. Структурований цикл	21. Структурований цикл	 а) цикл з передумовою  б) цикл з постумовою
22. Рекурсія	22. Рекурсія	
23*. Тригер	23. Перехідний тригер. 24. Стійкий тригер	
26. Скасувати багаторазову активність	26. Скасувати багаторазову активність	
27. Завершити багаторазову активність	27. Завершити багаторазову активність	

1	2	3
28. Блокуючий дискримінатор	28. Блокуючий дискримінатор	 а)  б)
29. Скасовуючий дискримінатор	29. Скасовуючий дискримінатор	 а)  б)

1	2	3
31. Блокування часткового з'єднання	31. Блокування часткового з'єднання	
34*. Часткове з'єднання	30. Структуроване часткове з'єднання. 34. Часткове статичне об'єднання для кількох екземплярів	
35*. Скасування часткового з'єднання	32. Скасування часткового з'єднання. 35. Скасування «часткове об'єднання для кількох екземплярів»	 а)  б)

1	2	3
36. Динамічне часткове об'єднання для кількох екземплярів	36. Динамічне часткове об'єднання для кількох екземплярів	
38*. Загальна синхронізація	37. Локальна синхронізація злиття. 38. Загальна синхронізація злиття	
39. Критичний розділ	39. Критичний розділ	

ДОДАТОК Б


PN-патерни комбінованого інструментарію моделювання систем з паралелізмом
Патерн внутріш-нього впливу
18
PN-патерни у послідовностях
(з однією вхідною та однією вихідною вершинами)
1
10
17*
9
21
13*
22
15*
34*
39*
PN-патерни у паралельних структурах
(з різною кількістю вхідних та вихідних вершин)
Патерни, залежні від внутрішніх впливів
19
23*
20*
26
27
38*
Патерни у розгалуженнях
2*
12
16*
Патерни у точках злиття потоків
3*
5*
11
28
29
31
35*
36

Наукове видання

О. О. Супруненко, Ю. Є. Гребенович

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ КОМПОНЕНТНО-ОРІЄНТОВАНОГО МОДЕЛЮВАННЯ ПРОГРАМНИХ СИСТЕМ ТА PN-ПАТЕРНИ

Монографія

Технічний редактор – Чабаненко Ю.
Комп'ютерна верстка – Зоря А.

Підписано до друку 21.04.2022.
Формат 60x84 1/16. Папір офсетний.
Умов. друк. арк. 8,75. Гарнітура Heuristica.
Зам. № 1256. Тираж 200 прим.

Видавець: Чабаненко Ю. А.
Свідоцтво про внесення
до Державного реєстру видавців
серія ДК № 1898 від 11.08.2004 р.
Україна, м. Черкаси, вул. О. Дашковича, 39
Тел: 0472/56-46-66, 0(93)788-99-99
E-mail: office@2upost.com

Друк ФОП Чабаненко Ю. А.
Україна, м. Черкаси, вул. О. Дашковича, 39
Тел: 0472/56-46-66, 0(93)788-99-99
E-mail: office@2upost.com