МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

А. В. Погудін, О. К. Погудіна, С. В. Губін

МОДЕЛЮВАННЯ СИСТЕМ

Навчальний посібник до самостійної роботи

Харків «ХАІ» 2018

УДК 519.876.5:004.94 (075.8)
П43

Рецензенти: д-р техн. наук, проф. Г. А. Кучук,
д-р техн. наук, проф. О. М. Пігнастий

Погудін, А. В.

П43 Моделювання систем [Текст] : навч. посіб. до самост. роботи / А. В. Погудін, О. К. Погудіна, С. В. Губін. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2018. – 104 с.

ISBN 978-966-662-594-9

Розглянуто класифікацію моделей і типові математичні схеми, що використовуються для вивчення складних об'єктів за допомогою інформаційних систем. Кожен тип схеми містить завдання для самостійної роботи. Наведено огляд існуючого програмного забезпечення для проведення імітаційного моделювання. Визначено порядок планування й проведення експериментів з моделями і вимоги до оформлення результатів моделювання. Подано варіанти індивідуальних завдань студентів для виконання розрахунково-графічної роботи.

Для студентів, що навчаються в галузі знань 12 «Інформаційні технології» при самостійній підготовці до лекцій, повторенні й вивченні основного матеріалу курсу «Моделювання систем».

Іл. 42. Табл. 19. Бібліогр.: 32 назви

УДК 519.876.5:004.94 (075.8)

ISBN 978-966-662-594-9

ЗМІСТ

ВСТУП

Неможливо уявити собі сучасну науку без широкого застосування математичного та / або імітаційного моделювання. Суть цієї методології полягає в заміні вихідного об'єкта його "образом" і подальшому вивченні моделі за допомогою інформаційних систем.

Різні аспекти моделювання систем описано в багатьох трудах, що використовуються як у матеріалах лекцій курсу «Моделювання систем» [1-3], так і при виконанні лабораторних робіт [4,5]. При написанні цього навчального посібника автори ставили собі за мету відібрати й викласти підходи до побудови моделей, загальних для різних галузей знань, уникаючи громіздкого й суворого опису, а звертаючи увагу студентів на ідеї та приклади, що їм відповідають. Тому ці матеріали можуть бути використані для самостійної підготовки студентів.

Стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології», спеціальністю 122 «Комп'ютерні науки та інформаційні технології» [6,7] містить розділи, в яких логічно впорядковано матеріал курсу «Моделювання систем». Саме тому навчальний посібник було сформовано за розділами: моделі систем, імітаційне моделювання, імітаційне моделювання виробничих і комп'ютерних систем.

Самостійна робота з курсу «Моделювання систем» складається з вивчення теоретичних питань першого розділу цього навчального посібника, літературних джерел і програмної документації, а також виконання:

- індивідуальних поточних завдань, що закріплюють теоретичний матеріал першого розділу;
- позааудиторної частини розрахунково-графічної роботи, порядок виконання та звітні матеріали якої наведено у другому розділі, а варіанти завдань – у третьому.

1. МОДЕЛІ СИСТЕМ

1.1. Загальні положення й визначення

Слово «модель» (від лат. modelium) означає «міра», «спосіб», «схожість з якоюсь річчю». Термін «модель» широко використовується в різних сферах людської діяльності й має безліч смислових значень. Під «моделлю» будемо розуміти певний матеріальний об'єкт, який під час дослідження заміщає об'єкт-оригінал так, що його безпосереднє вивчення дає нові знання про об'єкт-оригінал.

Модель – це об'єкт або опис об'єкта, системи для заміщення (при певних умовах, пропозиціях, гіпотезах) однієї системи (тобто оригіналу) іншою з метою кращого вивчення оригіналу або відтворення будь-яких його властивостей [8, 9].

Модель – це результат відображення однієї структури (вивченої) на іншу (маловивчену). Будь-яка модель будується і досліджується при певних припущеннях, гіпотезах.

Модель слід будувати так, щоб вона найбільш повно відтворювала ті якості об'єкта, які необхідно вивчити відповідно до поставленої мети. У всіх відношеннях модель має бути простішою і зручнішою для вивчення об'єкта. Таким чином, для одного й того ж об'єкта можуть існувати різні моделі, класи моделей, які відповідають різним цілям його вивчення. Необхідною умовою моделювання є подібність об'єкта і його моделі. У цьому випадку будемо говорити про адекватність моделі об'єкта-оригіналу. Якщо результати моделювання підтверджуються і можуть бути основою для прогнозування процесів, що відбуваються у досліджуваних об'єктах, то говорять, що модель є адекватною об'єкту. При цьому адекватність моделі залежить від мети моделювання і прийнятих критеріїв. Під адекватною моделлю розуміють модель, яка з певним ступенем наближення, на рівні розуміння, моделюється розробником моделі й відображає процес її функціонування в зовнішньому середовищі.

Під адекватністю (від лат. adaequatus – прирівняний) будемо розуміти ступінь відповідності результатів, отриманих за розробленою моделлю, даним експерименту або тестового завдання.

Якщо система, для якої розробляється модель, існує, то порівнюють вихідні дані моделі й цієї системи. У тому випадку, коли два набори даних виявляються подібними, модель існуючої системи вважається адекватною. Чим більше спільного між існуючою системою і її моделлю, тим більше впевненість у правильності моделі системи.

Перевірка адекватності моделі є необхідною для того, щоб переконатися в правдивості сукупності гіпотез, сформульованих на першому етапі розроблення моделі, і точності отриманих результатів, а також щоб визначити, чи відповідає модель точності, необхідній технічним завданням.

Для моделей, призначених для приблизних розрахунків, задовільною вважається точність 10 – 15 %, а для моделей, призначених для використання в керуючих і контролюючих системах, – 1 – 2 % [10].

Будь-яка модель має такі властивості:

- скінченність – модель відображає оригінал лише в скінченному числі його відносин;
- спрощеність – модель відображає тільки істотні сторони об'єкта;
- приблизність – модель відображає дійсність грубо або приблизно;
- адекватність – модель успішно описує систему, що моделюється;
- інформативність – модель містить достатню кількість інформації про систему в межах гіпотез, прийнятих при побудові моделі.

Процес побудови, вивчення і застосування моделей називатимемо моделюванням. Іншими словами, моделювання – це метод вивчення об'єкта шляхом побудови і дослідження його моделі, яке здійснюється з певною метою і полягає в заміні експерименту з оригіналом експериментом на моделі.

Моделювання базується на математичній теорії подібності, згідно з якою абсолютна подібність може мати місце лише при заміні одного об'єкта іншим, таким самим. При моделюванні більшості систем (за винятком, можливо, моделювання одних математичних структур іншими) абсолютна подібність є неможливою. Тому основна мета моделювання – побудова моделі, яка досить добре відображатиме функціонування модельованої системи.

1.1.1. Класифікація моделей

У загальному випадку всі моделі незалежно від областей і сфер їх застосування можуть бути трьох типів: пізнавальними, прагматичними й інструментальними.

Пізнавальна модель – форма організації і подання знань, засіб об'єднання нових і попередніх знань. Пізнавальна модель зазвичай відображає реальність і є теоретичною моделлю.

Прагматична модель – засіб організації практичних дій, робочого подання цілей системи для її керування. Реальність у них відображається згідно з певною прагматичною моделлю. Це зазвичай прикладні моделі.

Інструментальна модель – засіб побудови, дослідження та / або використання прагматичних і / або пізнавальних моделей.

Пізнавальні моделі відображають існуючі, а прагматичні – хоча й не існуючі, але бажані й, можливо, виконані відносини і зв'язки. Всі інші моделі класифікують відносно об'єкта-оригіналу, методів вивчення і т. ін.

За ступенем абстрагування від оригіналу (рис. 1.1) моделі можна поділити на матеріальні (фізичні) й ідеальні.

До матеріальних моделей належать такі, при яких дослідження ве-

деться на основі моделі, що відтворює основні геометричні, фізичні, динамічні й функціональні характеристики досліджуваного об'єкта. Основними різновидами фізичних моделей є [11] натурні, квазінатурні, масштабні й аналогові.

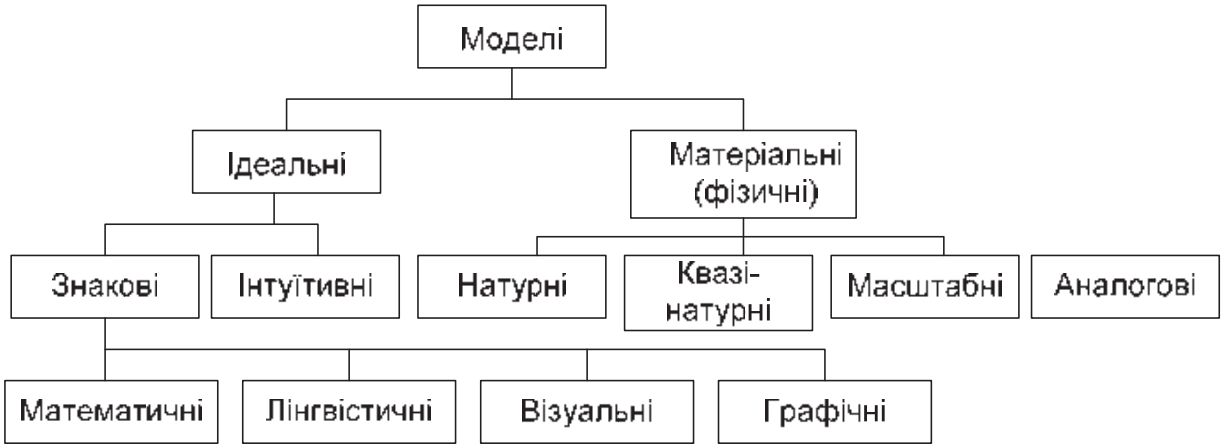


Рис. 1.1. Схема класифікації моделей за ступенем абстрагування від об'єкта-оригіналу

Натурні – це реальні досліджувані системи, які є макетами й дослідними зразками. Натурні моделі мають повну адекватність із системою-оригіналом, що забезпечує високу точність і достовірність результатів моделювання. Іншими словами, модель є натурною, якщо вона – матеріальна копія об'єкта моделювання. Наприклад, глобус – натурна географічна модель земної кулі.

Квазінатурні моделі (від лат. «квазі» – майже) – це сукупність натурних і математичних моделей. Цей вид моделей використовується у випадках, коли математична модель частини системи не є задовільною або коли частину системи необхідно дослідити у взаємодії з іншими частинами, але їх ще не існує або їх включення в модель є ускладненим або дорогим.

Масштабні моделі – це системи тієї ж фізичної природи, що й оригінал, але відрізняються від нього розмірами. Основою масштабних моделей є математичний апарат теорії подібності, який передбачає дотримання геометричної подібності оригіналу і моделі й відповідних масштабів їхніх параметрів. Прикладом масштабного моделювання є будь-які розроблення макетів будинків, а іноді й цілих районів при проведенні проектних робіт під час будівництва. Масштабне моделювання також використовується при проектуванні великих об'єктів у літакобудуванні й суднобудуванні.

Аналогове моделювання ґрунтується на аналогії процесів і явищ, що мають різну фізичну природу, але однаково описуються формально (одними й тими ж математичними рівняннями, логічними схемами і т. ін.). Як аналогові моделі використовують механічні, гідравлічні, пневматичні сис-

теми, але найбільш широкого застосування набули електричні та електронні аналогові моделі, в яких сила струму або напруги є аналогом фізичних величин і має іншу природу. Наприклад, відомо, що математичне рівняння коливання маятника має еквівалент при запису рівняння коливань струму.

Ідеальне моделювання має теоретичний характер. Розрізняють два типи ідеального моделювання: інтуїтивне й знакове. Під інтуїтивним будемо розуміти моделювання, основане на інтуїтивному уявленні про об'єкт дослідження, що не піддається формалізації або не має потреби в ній. У цьому розумінні, наприклад, життєвий досвід кожної людини може вважатися його інтуїтивною моделлю навколишнього світу.

Знаковим називається моделювання, при якому як моделі використовуються знакові перетворення різного виду: схеми, графіки, креслення, формули, набори символів тощо, які містять сукупність законів, за якими можна оперувати з вибраними знаковими елементами. Знакові моделі підрозділяються на лінгвістичні, візуальні, графічні й математичні.

Модель є лінгвістичною, якщо її подано деяким лінгвістичним об'єктом, формалізованою мовною системою або структурою. Іноді такі моделі називають вербальними, наприклад, правила дорожнього руху – мовна структурна модель руху транспорту і пішоходів на дорогах.

Модель є візуальною, якщо вона дає змогу візуалізувати відносини і зв'язки модельованої системи, особливо в динаміці. Наприклад, на екрані комп'ютера часто користуються візуальною моделлю об'єктів, клавіатури в програмі-тренажері з навчання роботі на клавіатурі.

Модель є графічною, якщо її подано геометричними образами і об'єктами (наприклад, макет будинку – натурна геометрична модель, що будується).

Найважливішим видом знакового моделювання є математичне моделювання. Класичним прикладом математичного моделювання є опис і дослідження основних законів механіки Ньютона засобами математики.

Математичні моделі класифікують:

- за належністю до ієрархічного рівня;
- характером відображуваних властивостей об'єкта;
- засобом подання властивостей об'єкта;
- засобом отримання моделі;
- формою подання властивостей об'єкта.

За належністю до ієрархічного рівня математичні моделі поділяють на моделі мікрорівня, макрорівня, метарівня. Математичні моделі мікрорівня відображають фізичні процеси, що відбуваються, наприклад, при різанні металів. Вони описують процеси на рівні переходу (проходу). Математичні моделі макрорівня відображають технологічні процеси. Математичні моделі метарівня описують технологічні системи (дільниці, цехи, підприємство в цілому).

За характером відображуваних властивостей об'єкта моделі можуть бути структурними й функціональними.

Модель є структурною, якщо її подано структурою даних або структурами даних і відносинами між ними; наприклад, структурною моделлю може бути опис (табличний, графовий, функціональний або інший) трофічної структури екосистеми. У свою чергу, структурна модель може бути ієрархічною або мережною.

Модель називається функціональною, якщо її подано у вигляді системи функціональних співвідношень (наприклад, закон Ньютона і модель виробництва товарів).

За засобом подання властивостей об'єкта моделі поділяють на аналітичні, числові, алгоритмічні й імітаційні [11].

Аналітичні математичні моделі являють собою явні математичні вирази вихідних параметрів як функцій від параметрів вхідних і внутрішніх і мають єдині розв’язки за будь-яких початкових умов (наприклад, процес різання (точіння) з точки зору діючих сил, квадратне рівняння, що має один або кілька розв’язків).

Модель називається числовою, якщо вона має розв’язок при конкретних початкових умовах (диференційні та інтегральні рівняння).

Модель є алгоритмічною, якщо її описано деяким алгоритмом або комплексом алгоритмів, що визначає її функціонування й розвиток. Уведення цього типу моделей (дійсно, здається, що будь-яка модель може бути описана алгоритмом її дослідження) цілком є обґрунтованим, оскільки не всі моделі можна дослідити або реалізувати алгоритмічно. Наприклад, моделлю обчислення суми нескінченного спадного ряду чисел може бути алгоритм обчислення скінченної суми ряду до деякого заданого степеня точності. Алгоритмічною моделлю кореня квадратного з числа X може бути алгоритм обчислення його наближеного, як завгодно точного значення за відомою рекурентною формулою.

Модель є імітаційною, якщо її призначено для дослідження або вивчення можливих шляхів розвитку й поведінки об'єкта шляхом варіювання деяких або всіх параметрів моделі (наприклад, модель економічної системи виробництва товарів двох видів). Таку модель можна використовувати як імітаційну з метою визначення й варіювання загальної вартості залежно від тих чи інших значень обсягів вироблених товарів.

За засобом отримання *моделі поділяють на теоретичні та емпіричні.*

Теоретичні математичні моделі будуються за результатом дослідження об'єктів (процесів) на теоретичному рівні. Наприклад, існують вирази для сил різання, отримані на основі узагальнення фізичних законів. Однак ці вирази є неприйнятними для практичного використання, оскільки вони дуже громіздкі й не зовсім адаптовані до реальних процесів оброблення матеріалів.

Емпіричні математичні моделі створюються внаслідок проведення експериментів (вивчення зовнішніх проявів властивостей об'єкта шляхом вимірювання його параметрів на вході й виході) і оброблення їх результатів методами математичної статистики.

За формою подання властивостей об'єкта *моделі поділяють на логічні, теоретико-множинні й графові.*

Модель є логічною, якщо її подано предикатами, логічними функціями (наприклад, сукупність двох логічних функцій може бути математичною моделлю однорозрядного суматора).

Модель називається теоретико-множинною, якщо її подано за допомогою деяких множин і відносин власності до них і між ними.

Модель є графовою, якщо її подано графом або графами і відносинами між ними.

Класифікація моделей за ступенем стійкості. Усі моделі можна поділити на стійкі й нестійкі.

Сталою є така система, яка, будучи виведеною зі свого вихідного стану, прагне набути цього стану. Вона може коливатися деякий час поблизу вихідної точки, подібно звичайному маятнику, приведеному в рух, але збурення в ній з часом згасають і зникають.

У нестійкій системі, яка перебуває спочатку в стані спокою, збурення, що виникло, посилюється, спричинає збільшення вимірюваного параметра.

Відносно зовнішніх чинників *моделі можна поділити на відкриті й замкнуті.*

Замкнутою моделлю є модель, яка функціонує поза зв'язком із зовнішніми (екзогенними) змінними. У замкнутій моделі зміни значень змінних в часі визначаються внутрішньою взаємодією самих змінних. Замкнута модель може виявити поводження системи без введення зовнішньої змінної. Наприклад, інформаційні системи зі зворотним зв'язком є замкнутими системами. Це самоналагоджувальні системи, і їх характеристики випливають із внутрішньої структури і взаємодій, які відображають введення зовнішньої інформації.

Модель, пов'язана із зовнішніми (екзогенними) змінними, називається відкритою.

Відносно тимчасового фактора моделі поділяють на динамічні й статичні (рис. 1.2).

Модель є статичною, якщо серед параметрів, що беруть участь в її описі, немає часового параметра. Статична модель у кожен момент часу дає лише «фотографію» системи, її зріз. Одним з видів статичних моделей є структурні моделі.

Модель є динамічною, якщо серед її параметрів є часовий параметр,тобто вона відображає систему (процеси в системі) у часі.

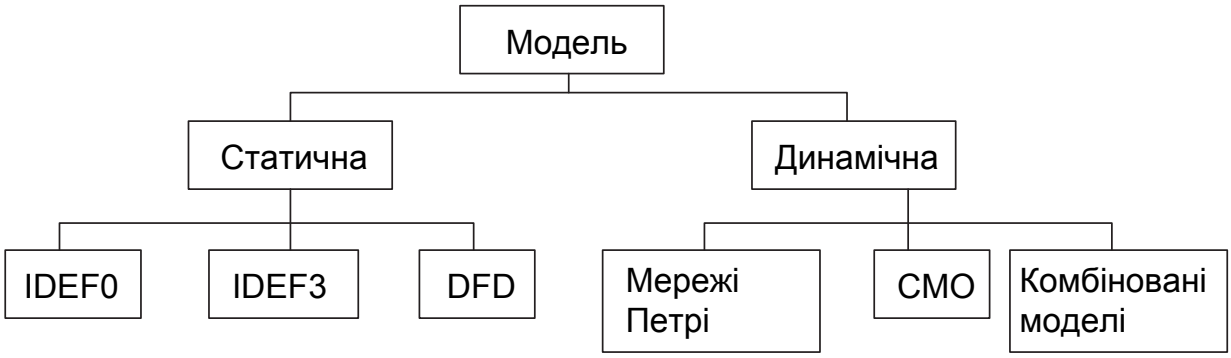


Рис. 1.2. Схема класифікації математичних моделей відносно часу

1.1.2. Типові математичні схеми

Математична модель у широкому розумінні – це приблизний опис будь-якого класу явищ зовнішнього світу, виражений за допомогою математичної символіки. Що стосується завдання дослідження якості системи, математична модель повинна забезпечувати адекватний опис впливу параметрів і умов функціонування на показники її якості. Що стосується точності моделі, то її рівень має забезпечувати достовірне порівняльне оцінювання й ранжування за рівнем якості альтернативних варіантів.

Основою вивчення й моделювання процесів функціонування технічних систем завжди є експеримент – реальний або логічний. Суть реального експерименту полягає в безпосередньому вивченні конкретного фізичного об'єкта. Під час логічного експерименту властивості об'єкта досліджуються не на самому об'єкті, а за допомогою його математичної або змістовної (словесної) моделі, ізоморфної об'єкту з точки зору досліджуваних властивостей.

Подаючи на вхід системи різні вхідні процеси і вимірюючи процес на її виході, дослідник отримує можливість установити і записати математично існуючий між ними зв'язок у вигляді рівняння, що зв'язує для кожного інтервалу часу значення вхідних і вихідних впливів і називається рівнянням «вхід-вихід». Крім того, для адекватного відображення зв'язку між входом і виходом системи в системотехніці вводиться поняття «стан». За своїм змістом стан z(т) являє собою сукупність істотних властивостей (характеристик) системи, знання яких у сьогоденні (у момент часу т) дає змогу визначити її поводження в майбутньому (у моменти часу t > т). Завдяки цьому поняттю рівняння "вхід-вихід" набирає вигляду

$$Y_T = A(T, z(\tau), X_T), \qquad (1.1)$$

де X_T, Y_T – вхідний і вихідний процеси на інтервалі часу T; $A(*)$ – оператор виходів.

Згідно з (1.1) вихідний процес повністю визначається вхідним процесом і початковим станом і не залежить від того, яким чином систему було переведено в цей стан. Звідси зрозуміло, що рівняння (1.1) обмежує клас даних систем тільки такими системами, функціонування яких у цей момент не залежить від того, як вони функціонували в минулому.

Для повного опису процесу функціонування системи необхідно задати умови визначення стану системи. Для цього вводиться поняття рівняння стану, яке можна записати так:

$$z(t) = B(\tau t, z(\tau), X_{\tau t}), \qquad (1.2)$$

де B(*) – оператор, що встановлює однозначну залежність z(t) від пари (z(т), Xтt), яку задано на інтервалі τ_t і яка має назву оператора переходу.

Рівняння (1.1) і (1.2) мають досить логічне узагальнення і в багатовимірному випадку, коли кожна з компонент рівнянь має векторний вигляд:

$$X \to \overline{X}, Y \to \overline{Y}, Z \to \overline{Z}.$$

Таким чином, модель функціонування системи повинна забезпечувати прогнозування процесу функціонування на всьому інтервалі функціонування T (множина часу) за заданим вектором початкового стану $\overline{Z}(\tau)$, записаним у векторному вигляді вхідного процесу $\overline{X}(T)$. Відповідно до викладеного вище для розв'язання цієї задачі досить задати множину допустимих значень вхідних X і вихідних Y процесів, а також множину можливих станів системи Z і оператори виходу A і переходу B. Модель функціонування системи без передісторії є кортежем

$$MF = < T, X, Y, Z, A, B > . \qquad (1.3)$$

Якщо всі компоненти в (1.3) є відомими, то модель функціонування повністю визначено і її можна використати для опису й вивчення властивих системі процесів функціонування. Множини й оператори, які складають загальносистемну модель (1.3), можуть мати різні властивості, сукупність яких дає змогу конкретизувати характер функціонування системи:

N – безперервність;

L – лінійність;

C – стаціонарність;

P – стохастичність (імовірність).

Надаючи системі тих чи інших властивостей загальносистемна модель конкретизується до системної моделі.

Системні властивості:

1. Якщо інтервал функціонування системи $T = [\varepsilon, \eta]$ є відрізком осі дійсних чисел, заданим початком ε і кінцем η, то система функціонує в безперервному часі. Якщо, крім того, оператори A і B є безперервними, то система називається безперервною.

2. Стосовно реакції на зовнішній вплив об'єкти поділяють на лінійні й нелінійні. Лінійним називається такий об'єкт, реакція якого на спільний

вплив двох будь-яких зовнішніх збурень дорівнює сумі реакцій на кожний з цих впливів, прикладених до системи окремо:

$F^0(x_1(t)+x_2(t))=F^0(x_1(t))+F^0(x_2(t))$ – принцип суперпозиції;

$F^0(0)=0$ (початковий стан системи),

де F^0 – оператор об'єкта, який установлює зв'язок між входом і виходом.

Для лінійних систем виконується принцип суперпозиції.

3. Оскільки стаціонарна система при фіксованому початковому стані $Z(t_0)$ однаково реагує на еквівалентні вхідні впливи, що відрізняються тільки зрушенням за часом, то накладення інтервалу t_0, t на осі часу не впливає на процес функціонування системи. Модель M для стаціонарних систем не містить в явному вигляді інтервал функціонування T.

4. Якщо в моделі M оператори A і B кожній парі $(X,V,Z(t_0))$ (вхід, стан) ставлять у відповідність єдині значення Y і Z, то система, що описується цією моделлю, називається детермінованою. Для стохастичної (імовірнісної) системи Y і Z випадкові величини задаються за законами розподілу.

Загальносистемна й системні моделі функціонування (у подальшому термін «модель функціонування» для стислості може замінюватися терміном «модель» зі збереженням вихідного значення) мають виключно високий ступінь спільності. Вони, безумовно, є необхідними для теоретичних досліджень і корисними, оскільки виявляють загальні закономірності, притаманні досить широкому класу систем. Але в повсякденній практичній діяльності інженери традиційно використовують так звані конструктивні моделі, які є набагато менш загальними, але дають змогу виконувати конкретні обчислення. Конструктивні моделі по суті являють собою алгоритми, користуючись якими, можна визначити значення одних змінних, що характеризують певну систему, за заданими або виміряним значенням інших змінних. Однак між системними й конструктивними моделями немає суперечності. У міру накопичення знань про систему, уточнення і конкретизації її властивостей і характеристик системна модель природним чином перетворюється на конструктивну. Отже, конструктивна модель може і повинна закономірно виростати з більш загальної системної моделі. Такий суто системотехнічний підхід здається більш обґрунтованим, ніж апріорне задання конструктивної моделі дослідником, що використовує для цього лише свою інтуїцію і суб'єктивні уявлення про можливості тих чи інших математичних схем.

Таким чином, найбільш важливі й принципові етапи побудови моделі функціонування системи визначаються процесом реалізації системотехнічного ланцюжка перетворень «загальносистемна модель $\rightarrow$ системна модель $\rightarrow$ конструктивна модель $\rightarrow$ машинна модель».

Моделювання процесів функціонування конкретної системи має по-

чинатися із запису всіх компонент загальносистемної моделі (1.3), визначення їх змістовного сенсу і областей зміни. Згідно з моделлю (1.3) необхідно визначити: інтервал часу, на якому треба дослідити функціонування системи; множину вхідних і вихідних впливів і області їх можливих змін; множину характеристик стану системи і область їх можливих змін.

Під час побудови моделей функціонування систем застосовують підходи (математичні схеми моделювання) [12], наведені в табл. 1.1.

Таблиця 1.1

Математичні схеми

Процес функціонування системи	Типова математична схема	Позначення
Безперервно-детермінований підхід	Стандартні диференціальні рівняння	D-схема
Дискретно-детермінований підхід	Кінцеві автомати	F-схема
Дискретно-стохастичний підхід	Імовірнісні автомати	P-схема
Безперервно-стохастичний підхід	Система масового обслуговування	Q-схема
Узагальнений (універсальний) підхід	Агрегативна система	A-схема

Математичні схеми моделювання систем дають змогу розглядати математику не як метод розрахунку, а як засіб формулювання понять, що є найбільш важливим при переході від словесного опису до формалізованого подання процесу її функціонування у вигляді деякої математичної моделі. При використанні математичної схеми дослідника системи в першу чергу має цікавити питання про адекватність відображення у вигляді конкретних схем реальних процесів у досліджуваній системі, а не можливість отримання відповіді (результату) на конкретне питання дослідження.

Математичну схему можна визначити як ланку при переході від змістовного до формалізованого опису процесу функціонування системи з урахуванням впливу зовнішнього середовища, тобто має місце ланцюжок «описова модель – математична схема – імітаційна модель».

1.1.3. Завдання

На основі розглянутої класифікації моделей за різними ознаками сформувати Mindmapping (деревоподібну карту) за допомогою одного з вибраних інструментів [13]:

- XMind;
- MindJet Mindmanager;

- iMindMap; Mindmeister;
- Mind Node;
- FreeMind.

1.2. Безперервно-детерміновані моделі (D-схеми)

Розглянемо особливості безперервно-детермінованого підходу на прикладі, використовуючи диференціальні рівняння як математичну модель.

Диференціальними рівняннями називаються такі рівняння, в яких функції однієї або декількох змінних є невідомими, причому до рівняння входять не тільки їх функції, але і їх похідні різних порядків.

Якщо невідомими є функції багатьох змінних, то рівняння називаються рівняннями в частинних похідних. Якщо невідомими є функції однієї незалежної змінної, то – звичайними диференціальними рівняннями (ДР).

Математичне співвідношення для детермінованих систем у загальному вигляді

$$\overline{y}^{l} = \overline{f}(\overline{y}, t)\overline{y}(t_0) = \overline{y}_0. \qquad (1.4)$$

1.2.1. Приклади D-схем

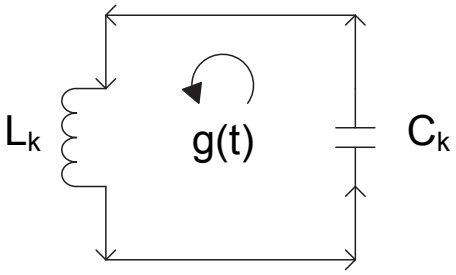


Рис.1.3. Схема електричного контуру

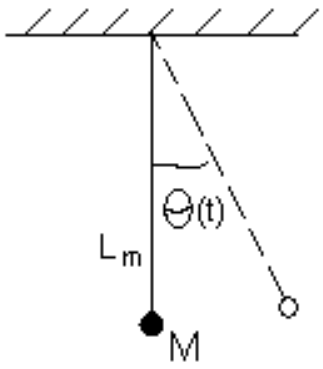


Рис.1.4. Схема коливань маятника

Процеси в електричному коливальному контурі (рис. 1.3) на базі ДР описуються формулою

$$L_k \frac{d^2 q(t)}{dt^2} + \frac{q(t)}{C_k} = 0, \qquad (1.5)$$

де L_k – індуктивність, C_k – ємність конденсатора, $q(t)$ – заряд конденсатора в момент часу t. Тоді $T = 2\pi\sqrt{L_k C_k}$ – період коливань.

Процес малих коливань маятника (рис. 1.4) можна описати звичайним диференціальним рівнянням

$$m_M l_M^2 \frac{d^2\Theta(t)}{dt^2} + m_M g l_M = 0,$$

де m_M, l_M – маса, довжина підвіски маятника, $\Theta(t)$ – кут відхилення маятника від стану рівноваги. Тоді $T_n = 2\pi\sqrt{\frac{l_M}{g}}$ – період коливань.

Введемо позначення: h_1, h_2, h_3 – внутрішні параметри системи. Отже,

$$\begin{aligned} h_0 &= L_k = m_M l_M^2, \\ h_1 &= 0, \\ h_2 &= \frac{1}{C_k} = m_M g l_M, \\ \Theta(t) &= q(t) = z(t). \end{aligned} \tag{1.6}$$

Отримаємо звичайне диференціальне рівняння другого порядку, що описує поводження замкнутої системи (1.6):

$$h_2 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_0 z(t) = 0, \tag{1.7}$$

де $z(t)$ – стан системи в момент часу t.

Якщо система (1.6) взаємодіє із зовнішнім середовищем, то виникає вхідний вплив $x(t)$ і безперервно детермінована система має вигляд

$$h_0 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_2 z(t) = x(t). \tag{1.8}$$

Стан системи S у цьому випадку можна розглядати як вихідну характеристику, тобто вважати, що $y = x$. Отже, використання D-схем дає змогу формалізувати процес функціонування безперервно-детермінованих систем і оцінити їх характеристики, застосовуючи аналітичний або імітаційний підхід.

Системна динаміка як метод імітаційного моделювання включає структуризацію об'єкта; побудову системної діаграми об'єкта, де вказуються зв'язки між елементами; визначення змінних для кожного елемента і темпів їх зростання; прийняття гіпотез про залежність кожного темпу зростання від змінних і формальний опис цих гіпотез; оцінювання введених параметрів за допомогою наявної статистики.

Програмні засоби, що реалізують модель Форрестера: AnyLogic, VenSim, PowerSim, Stella, ModelMaker та ін. Для побудови моделей в них використовується графічне подання залежностей змінних у вигляді так званих «stock and flow diagrams».

1.2.2. Завдання

Вивчити модель поширення нового продукту за Бассом [14] і побудувати імітаційну модель в AnyLogic.

Модель Басса (рис. 1.5) визначає процес поширення продукту. Спочатку продукт нікому не відомий, і для того, щоб люди почали його купува-

ти, його рекламують. Унаслідок цього певна частка людей купує продукт під впливом реклами. Люди також купують товар унаслідок спілкування з тими, хто цей продукт уже придбав. Процес придбання нового продукту під впливом переконання його власників чимось схожий на поширення епідемії.

Спочатку проаналізуємо модель, щоб вирішити, як її можна описати в термінах системної динаміки. Визначимо ключові змінні моделі й те, як вони впливають одна на одну, а потім створимо потокову діаграму моделі. При створенні потокової діаграми слід ураховувати, які змінні мають бути подані накопичувачами, які – потоками, а які – динамічними змінними.

Накопичувачі (їх також називають рівнями або фондами) являють

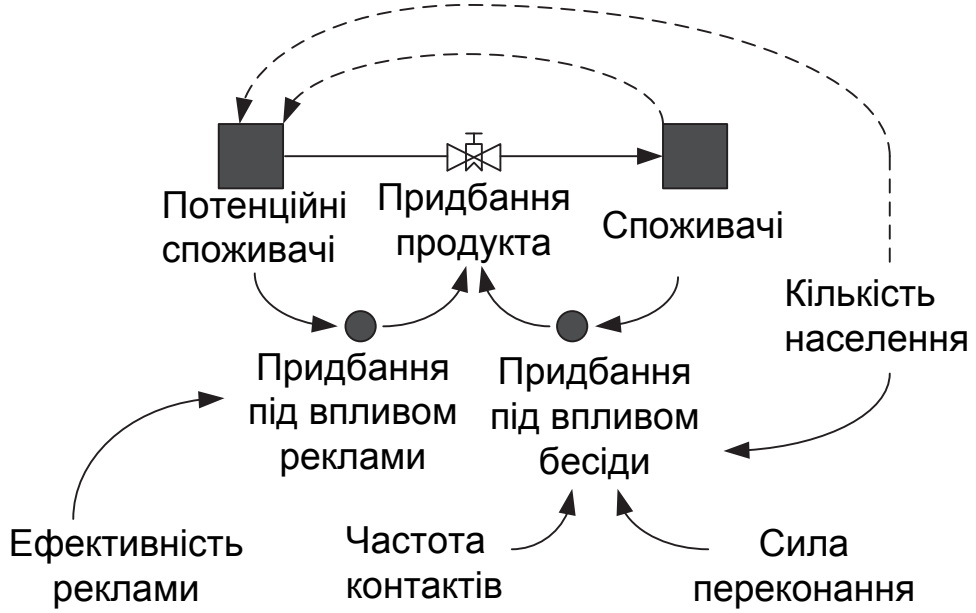


Рис.1.5. Модель поширення нового продукту за Бассом

собою такі об'єкти реального світу, в яких зосереджуються деякі ресурси; їхні значення змінюються безперервно.

Потоки – це активні компоненти системи, які змінюють значення накопичувачів. У свою чергу, накопичувачі системи визначають значення потоків.

Динамічні змінні допомагають перетворювати одні числові значення на інші. Вони можуть довільно змінювати свої значення або бути константами.

При створенні потокової діаграми слід виявити змінні, які накопичують значення протягом певного часу. У моделі кількості споживачів потенційні споживачі продукту є накопичувачами, а процес придбання продукту – потоком.

Системно-динамічне подання цієї моделі показано на рис. 1.7. Накопичувачі позначаються прямокутниками, потік – вентилем, а динамічні

змінні – кружками. Стрілки позначають причиново-наслідкові залежності в моделі.

Для додавання елементів існує палітра інструментів, зображених на рис. 1.6.

Зв'язки мають додатну або від'ємну полярність.

Додатний зв'язок означає, що два елементи системної динаміки змінюють свої значення в одному напрямку. Таким чином, якщо значення елемента, з якого спрямовано зв'язок, зменшується, то значення іншого елемента теж зменшується. Аналогічно якщо значення одного елемента збільшується, то значення елемента, який від нього залежить, теж збільшується.

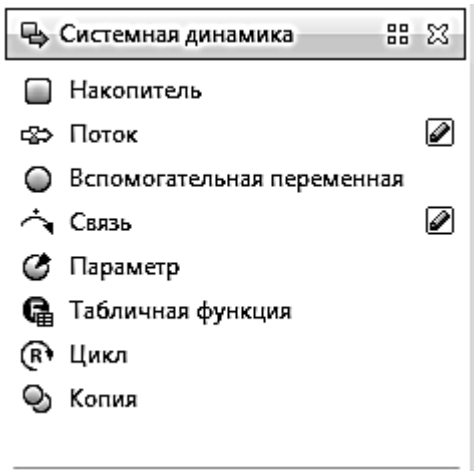


Рис.1.6. Палітра інструментів

Від'ємний зв'язок означає, що два елементи системної динаміки змінюють свої значення в протилежних напрямках, тобто якщо значення елемента, з якого спрямовано зв'язок, зменшується, то значення іншого елемента збільшується, і навпаки.

Можна поруч зі зв'язками додати мітки, які будуть позначати полярність цих зв'язків. Зазвичай полярність позначається за допомогою символів «+/-» поруч зі стрілкою зв'язку. Таким чином можна показати, як змінюється залежна змінна при зміненні незалежної змінної.

Очевидно, що модель Басса містить два цикли зі зворотним зв'язком: компенсувальний і підсилювальний.

Компенсувальний цикл (рис. 1.7, а) зі зворотним зв'язком впливає на потік придбання продукту, спричинений рекламою. Потік придбання продукту зменшує кількість потенційних споживачів, що, в свою чергу, призводить до зниження інтенсивності придбання продукту.

Підсилювальний цикл (рис. 1.7, б) зі зворотним зв'язком впливає на потік придбання продукту, спричинений спілкуванням зі споживачами продукту. Потік придбання продукту збільшує кількість споживачів продукту, що приводить до зростання інтенсивності придбання продукту під впливом спілкування з споживачами продукту, і, отже, до зростання інтенсивності придбання продукту.

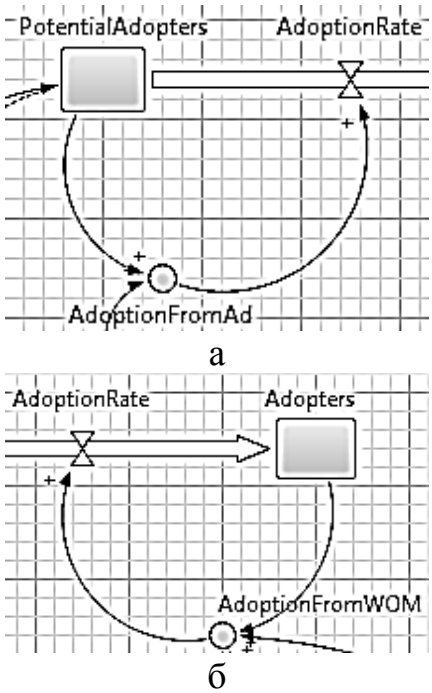


Рис. 1.7. Види циклів:
а – компенсувальний;
б – посилювальний

Щоб визначити, чи є цикл підсилюваль-

ним або врівноважувальним, можна почати з припущення, що, наприклад, значення змінної А збільшується, і простежити за зміненням значень, які входять до циклу змінних.

Цикл є:

- підсилювальним, якщо після проходження за циклом результат залишається таким самим, як і при початковому припущенні (у цьому випадку значення збільшилося).

- врівноважувальним, або компенсувальним, якщо результат суперечить початковому припущенню.

Таким чином, діаграма повинна мати вигляд, як показано на рис. 1.8.

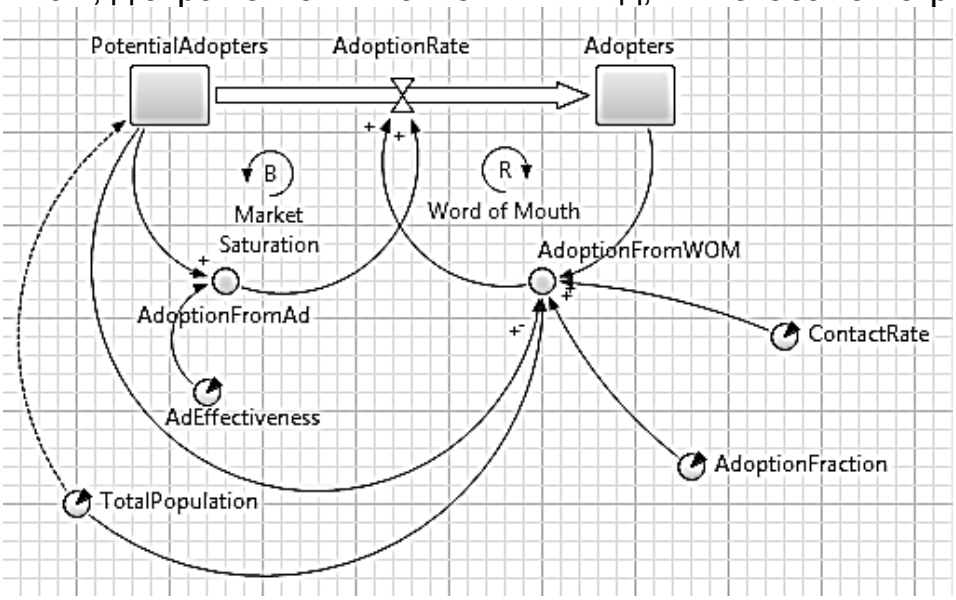


Рис. 1.8. Діаграма моделі Басса

Під час пуску моделі має бути видно, що при впровадженні нового продукту на ринок, коли кількість споживачів дорівнює нулю, реклама буде єдиним джерелом продажу. Найбільший рекламний ефект спостерігається на початку процесу поширення продукту; він неухильно зменшується у міру зменшення кількості потенційних споживачів

1.3. Дискретно-детерміновані моделі (F-схеми)

Дискретно-детерміновані моделі (ДДМ) є предметом розгляду теорії автоматів (ТА). ТА – розділ теоретичної кібернетики, що вивчає пристрої, які переробляють дискретну інформацію і змінюють свій внутрішній стан лише в допустимі моменти часу.

Кінцевий автомат має множину внутрішніх станів і вхідних сигналів, що є скінченними множинами. Автомат задається F-схемою:

$$F =< z, x, y, \varphi, \psi, z_0 >, \tag{1.9}$$

де z, x, y – відповідно скінченні множини вхідних, вихідних сигналів (алфа-

вітів) і скінченна множина внутрішніх станів (алфавіту); $z_0 \in Z$ – початковий стан; $\varphi(z,x)$ – функція переходу; $\psi(z,x)$ – функція виходу. Автомат функціонує в дискретному автоматному часі, моментами якого є такти, тобто однакові інтервали часу, які межують один з одним і кожному з яких відповідають сталі значення вхідного, вихідного сигналів і внутрішнього стану. Абстрактний автомат має один вхідний і один вихідний канали.

У момент t автомат, що перебуває в стані $z(t)$, здатний сприйняти сигнал $x(t)$ і видати сигнал $y(t) = \psi[z(t), x(t)]$, переходячи в стан $z(t+1) = \varphi[z(t), x(t)]$, $z(t) \in Z$; $y(t) \in Y$; $x(t) \in X$. Абстрактний КА в початковому стані z_0 приймає сигнали $x(0), x(1), x(2)$ ... і видає сигнали $y(0), y(1), y(2)$ ... (вихідне слово).

Існують F-автомати:

– першого роду (Міля), що функціонують за схемою

$$z(t+1) = \varphi[z(t), x(t)],\ t = 0,1,2... , \tag{1.10}$$

$$y(t) = \psi[z(t), x(t)],\ t = 0,1,2... ; \tag{1.11}$$

– другого роду, які функціонують за схемою

$$z(t+1) = \varphi[z(t), x(t)],\ t = 0,1,2... , \tag{1.12}$$

$$y(t) = \psi[z(t), x(t-1)],\ t = 1,2,3... ; \tag{1.13}$$

– другого роду, для яких

$$y(t) = \psi[z(t)],\ t = 0,1,2... \tag{1.14}$$

Отже, функція виходів, яка не залежить від вхідної змінної x(t), називається автоматом Мура. Оскільки рівняння (1.10) – (1.14) повністю задають F-автомат, то окремим випадком є рівняння

$$\overline{z}(t) = \Phi(\overline{z}_0, \overline{x}, \overline{\nu}, \overline{h}, \overline{t}), \tag{1.15}$$

де $\overline{z}$ – вектор стану, $\overline{x}$ – вектор незалежних вхідних змінних, $\overline{\nu}$ – вектор впливів зовнішнього середовища, $\overline{h}$ – вектор власних внутрішніх параметрів системи, $\overline{z}_0$ – вектор початкового стану, t – час; і рівняння

$$\overline{y}(t) = F(\overline{z}, t), \tag{1.16}$$

коли система є S-денормірованою і на її вхід надходить дискретний сигнал x.

За кількістю станів кінцеві автомати бувають з пам'яттю і без пам'яті. Автомати з пам'яттю мають більше одного стану, а автомати без пам'яті (комбінаційні або логічні схеми) – лише однин стан. При цьому згідно з (1.11) робота комбінаційної схеми полягає в тому, що вона ставить у відповідність кожному вхідному сигналу $x(t)$ певний вихідний сигнал $y(t)$, тобто реалізує логічну функцію вигляду

$$y(t) = \psi[z(t)],\ t = 0,1,2...$$

Ця функція має назву булевої, якщо алфавіти X і Y, яким належать значення сигналів x і y, складаються з двох букв.

За характером відліку часу (дискретного) F-автомати поділяють на синхронні й асинхронні. У синхронних автоматах моменти часу, в які вони "зчитують" вхідні сигнали, визначаються примусово-синхронізуючими сигналами. Реакція автомата на кожне значення вхідного сигналу закінчується за один такт синхронізації. Асинхронний F-автомат «зчитує» вхідний сигнал безперервно, і тому, реагуючи на досить довгий вхідний сигнал сталої величини x, може, як це випливає з (1.10) – (1.14), кілька разів змінити свій стан, видаючи відповідну кількість вихідних сигналів, поки набуде стійкого стану.

Для задання F-автомата необхідно описати всі елементи множини $F =< z, x, y, \varphi, \psi, z_0 >$, тобто вхідний, внутрішній і вихідний алфавіти, а також функції переходів і виходів. Для задання роботи F-автоматів найбільш часто використовують табличний, графічний і матричний способи.

При табличному способі задання використовують таблиці переходів і виходів, рядки яких відповідають вхідним сигналам автомата, а стовпці – його станам. При цьому зазвичай 1-й стовпець зліва відповідає початковому стану z_0. На перетині і-го рядка і j-го стовпця таблиці переходів розміщується відповідне значення $\varphi(z_k, x_i)$ функції переходів, а в таблиці виходів $\psi(z_k, x_i)$ – функції виходів. Для F-автомата Мура обидві таблиці можна поєднати, отримавши так звану таблицю переходів, в якій над кожним станом z_k автомата, що позначає стовпець таблиці, стоїть згідно з (1.14) вихідний сигнал $\psi(z_i)$, що відповідає цьому стану.

Роботу F-автомата Мілі задано таблицями переходів φ і виходів ψ (табл. 1.2), а роботу F-автомата Мура – таблицею переходів (табл. 1.3).

Таблиця 1.2

Автомат Мілі

xj	zk			
	z0	z1	…	zk
Переходи				
x1	φ(z0,x1)	φ(z1,x1)	…	φ(zk,x1)
x2	φ(z0,x2)	φ(z1,x2)	…	φ(zk,x2)
……				
xl	…	…	…	…
Виходи				
x1	ψ(z0,x1)	ψ(z1,x1)	…	ψ(zk,x1)
…… …				
xl	ψ(z0,xl)	ψ(z1,xl)	…	ψ(zk,xl)

Таблиця 1.3

Автомат Мура

xi	ψ(zk)			
	ψ(z0)	ψ(z1)	…	ψ(zk)
	z0	z1	…	zk
x1	φ(z0,x1)	φ(z1,x1)	…	φ(zk,x1)
x2	φ(z0,x2)	φ(z1,x2)	…	φ(zk,x2)
……				
xl	φ(z0,xl)	φ(z1,xl)	…	φ(zk,xl)

Приклади табличного способу задання F-автомата Мілі F_1 з трьома станами, двома вхідними і двома вихідними сигналами наведено в табл. 1.4, а для F-автомата Мура F_2 – в табл. 1.5.

При іншому способі задання кінцевого автомата використовується поняття напрямленого графа. Граф автомата являє собою набір вершин, що відповідають різним станам автомата, і зв'язків вершин дуг графа, які відповідають тим чи іншим переходам автомата. Якщо вхідний сигнал x_k спричиняє перехід зі стану z_i у стан z_j, то на графі автомата дуга, що з'єднує вершину z_i з вершиною z_j, позначається x_k. Для того щоб задати функцію переходів, дуги графа необхідно позначити відповідними вихідними сигналами.

Таблиця 1.4

Автомат Мілі

xj	z0		
	z0	z1	z2
Переходи			
x1	z2	z0	z0
x2	z0	z2	z1
Виходи			
x1	y1	y1	y2
x2	y1	y2	y1

Таблиця 1.5

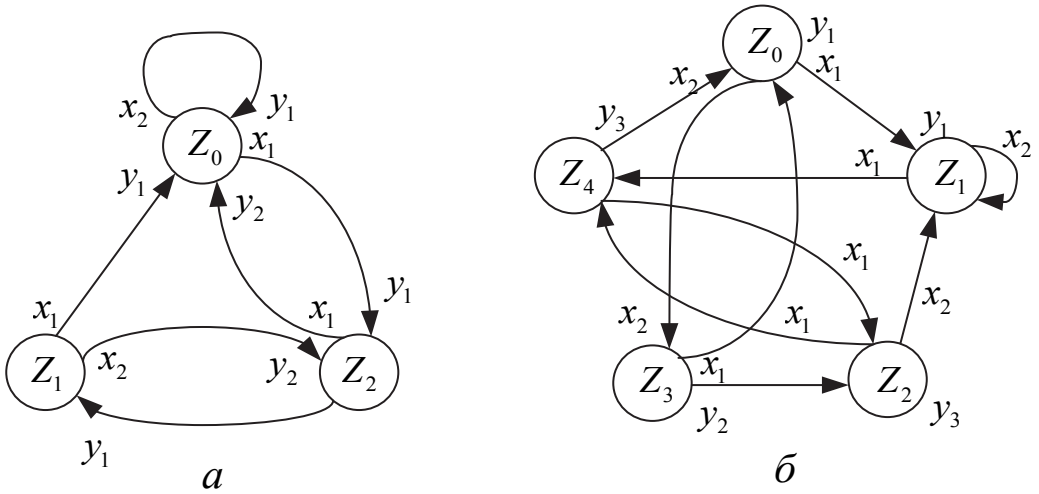
Автомат Мура

xi	Y				
	y1	y1	y3	y2	y3
	z0	z1	z2	z3	z4
x1	z1	z4	z4	z2	z2
x2	z3	z1	z1	z0	z0

Для автоматів Мілі цю розмітку виконують так: якщо вхідний сигнал діє на стан z_i, то відповідно до сказаного вище одержують дугу x_k, яка виходить із z_i, цю дугу додатково позначають вихідним сигналом $y = \psi(z_i, x_k)$.

Для автомата Мура розмітка графа є такою: якщо вхідний сигнал x_k, діючи на деякий стан автомата, спричиняє перехід у стан z_j, то дугу, спрямовану в z_j і позначену x_k, додатково позначають вихідним сигналом $y = \psi(z_j, x_k)$. На рис. 1.9 наведено задані раніше таблицями F-автомати Мілі F_1 і Мура F_2 відповідно.

Часто більш зручною формою розв’язання задач моделювання є матричне задання кінцевого автомата. При цьому матриця з'єднань автомата є квадратною матрицею $C = \left\| c_{ij} \right\|$, рядки якої відповідають вихідним станам, а стовпці – станам переходу. Елемент $c_{ij} = \dfrac{x_k}{y_S}$ для автомата Мілі відповідає вхідному сигналу x_k, що спричиняє перехід зі стану z_i у стан z_j, і вихідному сигналу y_S, який видається при цьому переході.



а *б*

Рис. 1.9. Графи автоматів Мілі (а) і Мура (б)

Для автомата Мілі, розглянутого вище, матриця з'єднань має вигляд

$$C_1 = \begin{pmatrix} \frac{x_2}{y_1} & - & \frac{x_1}{y_1} \\ \frac{x_1}{y_1} & - & \frac{x_2}{y_2} \\ \frac{x_1}{y_2} & \frac{x_2}{y_1} & - \end{pmatrix}.$$

Якщо перехід зі стану z_i у стан z_j відбувається під дією декількох сигналів, то елемент матриці c_{ij} є множиною пар "вхід / вихід" для цього переходу, з'єднаних знаком диз'юнкції.

Для F-автомата Мура елемент c_{ij} дорівнює множині вхідних сигналів на переході ($z_i z_j$), а вихід описується вектором виходів:

$\overline{y} = \begin{vmatrix} \varphi(z_0) \\ \varphi(z_1) \\ \dots \\ \varphi(z_k) \end{vmatrix}$, і-та компонента якого є вихідним сигналом, що позначає стан z_i.

Приклад. Для розглянутого раніше автомата Мура F_2 запишемо матрицю станів і вектор виходів:

$$C_2 = \begin{pmatrix} - & x_1 & - & x_2 & - \\ - & x_2 & - & - & x_1 \\ - & x_2 & - & - & x_1 \\ x_2 & - & x_1 & - & - \\ x_2 & - & x_1 & - & - \end{pmatrix}; \qquad \overline{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_2 \\ y_3 \end{bmatrix}.$$

Для детермінованих автоматів переходи є однозначними. Що стосується графічного способу задання F-автомата, це означає, що в графі F-автомата з будь-якої вершини не можуть виходити два ребра і більше, позначені одним і тим самим вхідним сигналом. Аналогічно цьому в матриці з'єднань автомата три в кожному рядку будь-який вхідний сигнал не повинен зустрічатися більше одного разу.

Розглянемо, який має вигляд таблиця переходів і графа асинхронного кінцевого автомата. Для F-автомата стан z_k називається стійким, якщо для будь-якого входу $x_i \in X$, $\varphi(z_k, x_i) = z_k$. Таким чином F-автомат називається асинхронним, якщо кожний його стан $z_k \in Z$ є стійким.

На практиці автомати завжди є асинхронними, а стійкість їх станів забезпечується тим чи іншим способом, наприклад введенням сигналів синхронізації. На рівні абстрактної теорії часто зручно оперувати з синхронними кінцевими автоматами.

1.3.1. Стейтчарт як F-схема

У системі Anylogic кінцеві автомати відображаються у вигляді систем, що здатні реагувати, – стейтчартів. Система, яка здатна реагрувати (reactive system), – це система, що постійно очікує зовнішні або внутрішні події і реагує на них. Системи, які здатні реагувати, є типовими системами дискретно-подієвого типу: події відбуваються в дискретні моменти часу, реакція системи на події полягає в зміні змінних стану цих систем і формально така реакція є миттєвою.

При описі систем, що реагують, виявилося зручним використання стейтчартів (або карт станів). Стейтчарти є графічною мовою діаграм. Мова стейтчартів у цей час широко застосовується при специфікації, моделюванні й прототипуванні протоколів комунікації, систем керування в авіації, науковій і побутовій електроніці.

Стейтчарти складаються зі станів і переходів між ними. Система може перебувати в кожен момент часу тільки в одному стані. Переходи зі стану в стан трапляються, якщо відбувається подія, пов'язана з цим переходом, і умову, пов'язану з переходом, виконано. На діаграмі (рис. 1.10) система перейде зі стану A в стан B, якщо настане подія a й при цьому умову P буде виконано. Подією може бути, наприклад, завершення тайм-ауту, перемикання в істину предиката (умови), певного на змінних моделі й т. ін. Графічно стани відображаються прямокутниками або овалами, а переходи – дугами. Коротка стрілка-покажчик, що

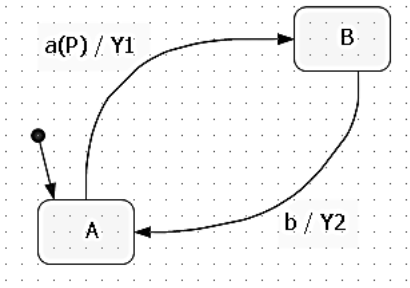


Рис. 1.10. Діаграма станів

показує стан A, свідчить про те, що цей стан є початковим: у початковий момент часу система буде перебувати саме в цьому стані. Очевидно, що система може мати лише один початковий стан. З кожним переходом може бути пов'язана деяка дія: змінення, посилання сигналу тощо. З кожним станом також можуть бути пов'язані дії. Одна дія виконується в момент входу в цей стан, інша – при виході зі стану. На рис. 1.10 Y_1 і Y_2 умовно позначено дії, що виконуються при спрацьовуванні відповідних переходів.

У загальному випадку в стейтчартах можна використовувати розширення цієї найпростішої моделі переходів: ієрархічні стани (гіперстани), історичні стани, умовні переходи й деякі інші можливості.

Ієрархічний стан, або гіперстан, вводять для того, щоб об'єднати кілька станів, що мають одну й ту саму реакцію на подію. На рис. 1.11 праворуч гіперстан D дає змогу спростити граф переходів, зображений зліва: перехід у стан A при надходженні події b відбувається незалежно від того, в якому зі станів (B або C) перебувала система. Іншими словами, два стейтчарти на рис. 1.11 є еквівалентними.

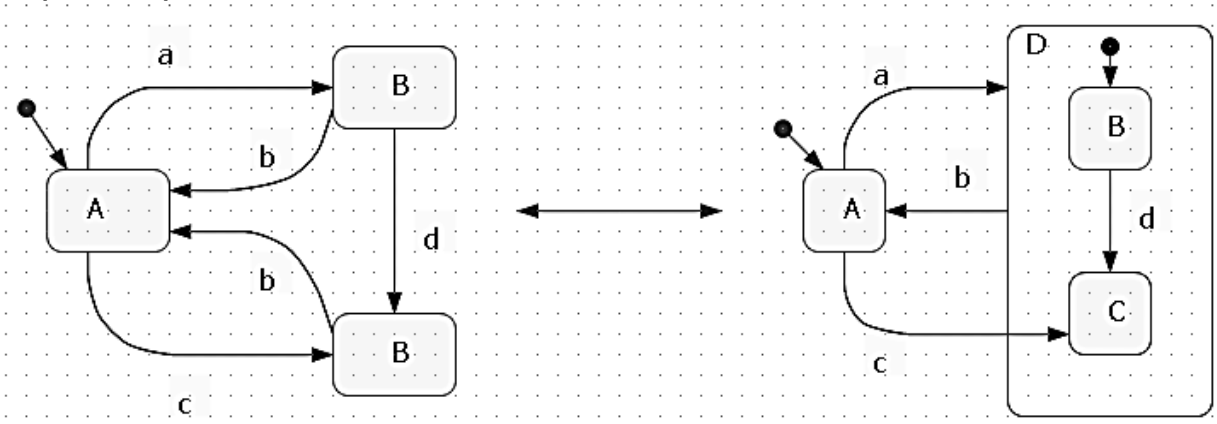


Рис. 1.11. Використання гіперстану, що спрощує граф переходів

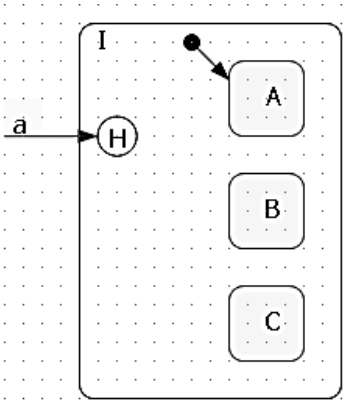


Рис. 1.12. Історичний стан

Кожний гіперстан потребує, щоб точно один з включених у нього станів був позначений як початковий. Це дає змогу трактувати перехід зі стану A при настанні події a в гіперстан D як перехід зі стану A в елементарний стан B. На рис.1.11 початковим станом системи є стан A, а стан B є початковим тільки для множини станів $\{B,C\}$, що входять у гіперстан D.

Історичний стан зберігає той стан усередині даного гіперстану, в якому система перебувала останній раз.

На рис. 1.12 при настанні події a система перейде до того стану з множини станів $\{A,B,C\}$, в якому вона була останній раз (незалежно від того, якими переходами

зв'язані ці стани). Історичні стани є зручними, наприклад, для опису продовження функціонування системи після переривань.

Умовні стани дають змогу відкласти перевірку логічної умови. Таке відкладання перевірки є зручним, наприклад, у тому випадку, коли подальші дії системи можна визначити тільки після реакції на подію.

1.3.2. Завдання 1

Як приклад розглянемо специфікацію процесу доступу до середовища протоколу IEEE 802.12 обміну повідомленнями у високошвидкісній локальній мережі 100VG-AnyLAN (рис. 1.13). У мережі працює безліч станцій, і в кожній з них активізовано свій процес доступу до мережі. Усі процеси є ідентичними.

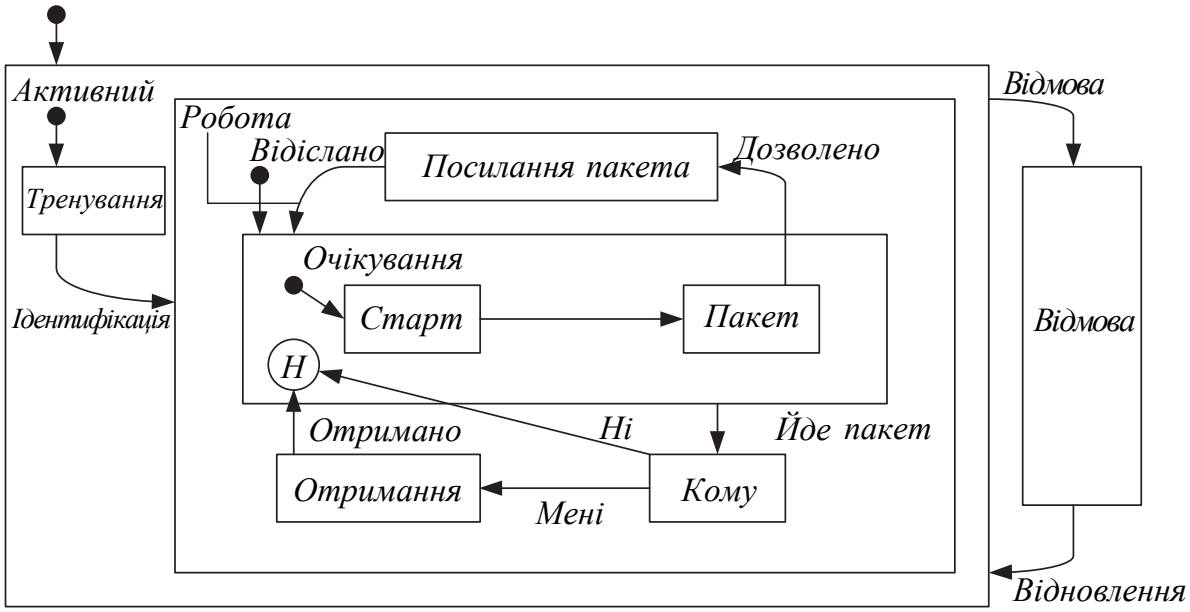


Рис. 1.13. Стейтчарт процесу доступу до середовища протоколу IEEE 802.12

Кожен процес починає свою роботу в елементарному стані тренування гіперстану *Активний.* У стані тренування виконуються операції з ідентифікації певної робочої станції, перевірки каналу і верхнього рівня стека протоколів. Якщо ідентифікацію було виконано без помилок, процес переходить у стан *Старт*, в якому очікує черговий пакет, генерований користувачем даної робочої станції для передачі. Коли пакет для передачі сгенеровано, процес переходить у стан *Пакет*, посилаючи сигнал верхнього рівня про наявність пакета. У цьому стані процес чекає від верхнього рівня дозволу послати пакет у канал, і коли дозвіл отримано, процес пере-

ходить у стан *Посилання* пакета. По завершенні пересилання пакета процес повертається у стан *Старт*. У будь-якому з двох станів (*Старт* і *Пакет*) гіперстану *Очікування* процес може бути перерваний приходом сигналу *Йде пакет* від іншої робочої станції. Цей сигнал сповіщає кожну станцію локальної мережі про те, що в мережі почав передаватися пакет, адресату необхідно підготуватися до його прийняття. Оскільки ім'я адресата знаходиться в заголовку пакета, пакет починають приймати всі процеси. Після прийняття пакета (або після виявлення, що пакет є чужим після прийняття заголовка) процес повертається в той стан, включений у гіперстан *Очікування*, з якого він був перерваний сигналом *Йде пакет*. У будь-якому з обговорюваних раніше станів процес може бути перерваний сигналом *Відмова*, який змушує процес перейти в стан *Відмова*. Після відновлення процес входить у нормальну роботу через стан тренування.

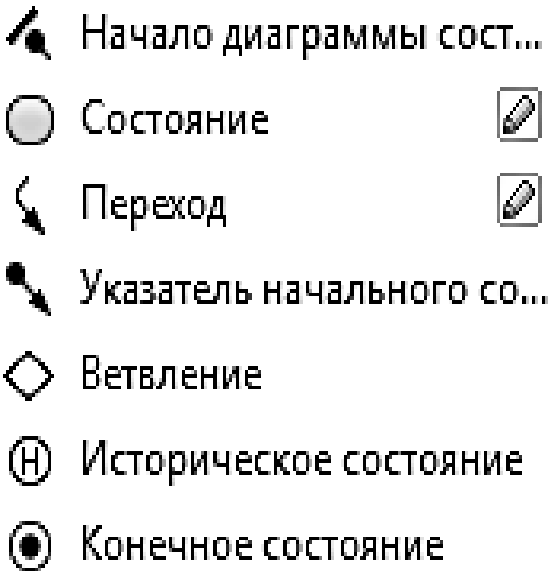


Рис. 1.14. Інструменти для створення діаграми станів

Щоб побудувати розглянутий кінцевий автомат, потрібно створити стейтчарт AnyLogic за допомогою інструментів палітри «Діаграма станів» (рис. 1.14).

Для створення діаграми станів потрібно використовувати три основні інструменти: «Початок діаграми» – позначає початкову точку оброблення стейтчарта; «Стан» – задає стан діаграми; «Перехід» – використовується для з'єднання станів; «Покажчик початкового стану» – застосовується для позначення стану, з якого починається оброблення визначеної послідовності станів; «Кінцевий стан» – позначає точку завершення оброблення станів.

Визначення початку діаграми є обов'язковим.

1.3.3. Завдання 2

Розробіть кінцевий автомат, який моделює роботу світлофора, що керує рухом автотранспорту [15].

Побудуйте нову модель з самого початку. Додайте в поле класу Main моделі три логічні змінні. Ці змінні фіксують стан світлофора: red – червоний сигнал; yellow – жовтий сигнал; green – зелений сигнал.

Побудуйте стейтчарт так, як показано на рис. 1.15.

Об'єднати стани можна за допомогою інструмента «Перехід». При правильному з'єднанні станів кінцеві точки переходу будуть помічені зеленим кольором.

Налаштування стейтчарта має відповідати табл. 1.6 і 1.7, а, налаштування переходів – табл. 1.7.

Перед налаштуванням переходів дайте початку стейтчарта ім'я p_0, а покажчику початкового композитного стану – ім'я p_1.

Примітка. Щоб ім'я переходу з'являлося на діаграмі, необхідно встановити його властивість «Відображати ім'я» в активний стан. Добитися потрібного розташування імені переходу на діаграмі можна після виділення лінії переходу мишею, захопивши ім'я переходу і відбуксирувавши його в потрібне місце.

Наступний крок полягає в розміщенні в моделі зображення світлофора. Використовуючи панель інструментів «Презентація», розмістіть три овали.

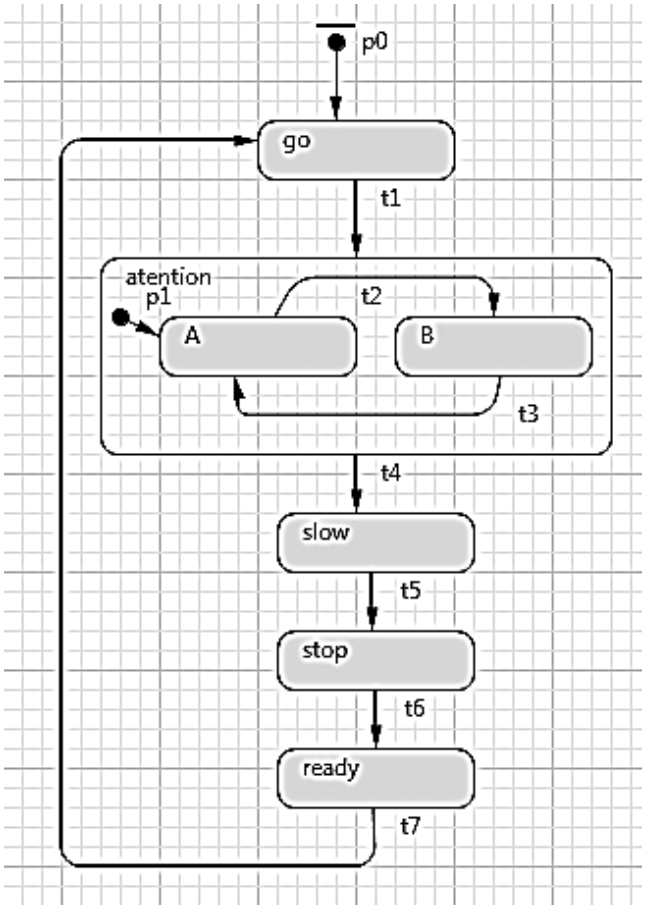


Рис. 1.15. Стейтчарт світлофора

Таблиця 1.6

Налаштування станів

Ім'я	Дія при вході	Дія при виході
go	green=true	green=false
atention		
A		
B	green=true	green=false
slow	yellow=true	yellow=false
stop	red=true	red=false
ready	red=true; yellow=true;	red=false; yellow=false;

Таблиця 1.7

Налаштування переходів

Ім'я	Тип	Період
t1	За тайм-аутом	25
t2	За тайм-аутом	1
t3	За тайм-аутом	1
t4	За тайм-аутом	5
t5	За тайм-аутом	5
t6	За тайм-аутом	25
t7	За тайм-аутом	5

Для відображення сигналів потрібно на вкладці «Динамічні» овалів задайте код Java для властивості «Колір заливки»:

Червоний сигнал: red? Color.red: Color.gray
Жовтий сигнал: yellow? Color.yellow: Color.gray
Зелений сигнал: green? Color.green: Color. gray

Налаштуйте модельний час експерименту моделі – об'єкт Simulation: одиниці модельного часу – секунди; зупинити – ні.

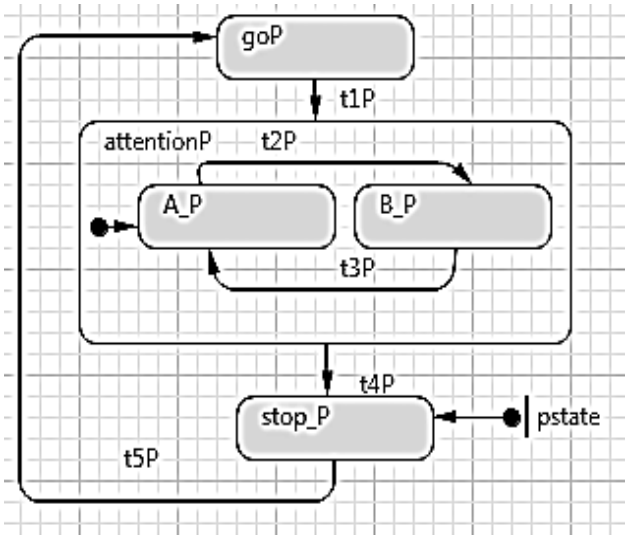


Рис. 1.16. Стейтчарт світлофора пішохідного переходу

При появі червоного сигналу на світлофорі руху автотранспорту повинен вмикатися зелений сигнал світлофора пішохідного переходу. При появі сигналу червоного кольору на світлофорі пішохідного переходу має з'явитися зелений сигнал на світлофорі руху. Процес повторюється циклічно.

Введіть у модель дві змінні логічного типу: redP, greenP.

Побудуйте стейтчарт, який відповідає рис. 1.16 .

Налаштування переходів і станів має відповідати табл. 1.8 і 1.9.

Таблиця 1.8

Налаштування переходів пішохідного стейтчарта

Ім'я	Тип	Період
t1p	За тайм-аутом	15
t2p	За тайм-аутом	1
t3p	За тайм-аутом	1
t4p	За тайм-аутом	15
t5p	За тайм-аутом	1

Таблиця 1.9

Налаштування станів стейтчарта пішохідного переходу

Ім'я	Дія при вході	Дія при виході
goP	greenP=true	greenP=false
B_P	greenP=true	greenP=false
Stop_P	redP=true	redP=false

Для організації взаємодії між стейтчартами в стейтчарт світлофора керування рухом внесіть такі зміни:

1. Відредагуйте перехід t5, додавши у властивість «Дія» код Java pstate.fireEvent ("ПІШОХОДИ"). Цей оператор передає стейтчарту світлофора пішохідного переходу контрольне повідомлення для ввімкнення зеленого сигналу.

2. У стейтчарті світлофора пішохідного переходу змініть налаштування переходу t5P: він відбувається при отриманні повідомлення, якщо повідомлення одне – «Пішоходи».

3. Після закінчення циклу роботи світлофора пішохідного переходу подайте сигнал світлофора, який керує рухом. У властивість «Дія» для переходу t4P введіть оператор Java p0.fireEvent ("Трафік").

4. У стейтчарті світлофора, який керує рухом, для переходу t6 змініть налаштування властивостей: він відбувається при отриманні повідомлення, якщо повідомлення одне – "Трафік".

5. Поруч зі стейтчартом керування роботою світлофора пішохідного переходу розмістіть його зображення. Додайте два кола: верхнє – для показу червоного сигналу, нижнє – зеленого. Для змінення кольору заливки кіл введіть код Java:

```
redP ? Color.red : Color.gray
greenP ? Color.green : Color.gray
```

6. У модель введіть логічну змінну waiting. Коли пішохід натискає кнопку, то їй присвоюється значення true.

7. У стейтчарт керування світлофором руху введіть композитний стан go. Перехід T повинен спрацювати при натисканні пішоходом кнопки «ЧЕКАЮ».

8. Після закінчення циклу роботи світлофора пішохідного переходу для виконання переходу t1P змінній waiting присвойте значення false.

1.4. Дискретно-стохастичні моделі (P-схеми)

Суть дискретизації часу в цих моделях є аналогічною суті побудови кінцевих автоматів. Розглянемо вплив фактора стохастичності на різновиди кінцевих автоматів, наприклад, на ймовірнісні (стохастичні) автомати (probabilistic automate).

Імовірнісний автомат – це дискретний потактний перетворювач інформації, функціонування якого в кожному такті залежить тільки від стану пам'яті в ньому і який може бути заданий статистично.

Введемо множину G, яка складається з пар (x_i, z_j), де $x_i \in X$, $z_j \in Z$. Тоді функція переходів $\varphi - G \to Z$, а функція виходів $\psi - G \to Y$. Кажуть, що $F =< Z, X, Y, \varphi, \psi >$ визначає автомат детермінованого типу.

Введемо множину Ф, що складається з пар (z_i, y_j), де y_j – елемент вихідної підмножини $y \in Y$. Нехай будь-який елемент множини G індукує на множині Ф деякий закон розподілу такого вигляду (табл. 1.10).

Таблиця 1.10

Закон розподілу на множині Ф

Елементи з Ф	(z_1, y_1)	(z_1, y_2)		(z_n, y_m)
(x_k, z_j)	b_{11}	b_{12}	…	b_{nm}

При цьому виконується умова $\sum_{i=1}^{n}\sum_{j=1}^{m} b_{ij} = 1$, де b_{ij} – імовірність переходу автомата в стан z_i і появи на виході сигналу y_j, якщо він був у стані z_l і на його вхід у цей момент часу надійшов сигнал x_k.

Кількість таких розподілів, поданих у вигляді таблиць, дорівнює кількості елементів множини G. Позначимо множину цих таблиць через B. Тоді набір елементів $P = < Z, X, Y, B >$ буде називатися ймовірнісним автоматом.

Введемо за аналогією з детермінованим автоматом визначення ймовірнісних автоматів Мілі й Мура. Нехай кожен елемент множини G індукує на множинах Z і Y деякий закон розподілу (табл. 1.11).

Таблиця 1.11

Закон розподілу на множині Y

Елементи з Y	y_1	y_2	…	y_{l-1}	y_l
(x_j, z_s)	q_1	q_2	…	q_{l-1}	q_l

Таблиця 1.12

Закон розподілу на множині Z

Елементи з Z	z_1	z_2	…	z_{k-1}	z_k
(x_i, z_s)	p_1	p_2	…	p_{k-1}	p_k

При цьому $\sum_{k=1}^{K} p_k = 1$ і $\sum_{l=1}^{L} q_l = 1$, де p_k і q_l – ймовірності переходу Р-автомата в стан z_k і появи вихідного сигналу y_l за умови, що автомат знаходився в стані z_s і на його вхід надійшов вхідний сигнал x_i.

Якщо для всіх k і l виконується умова $p_k q_l = b_{kl}$, то такий автомат називається ймовірнісним автоматом Мілі. Ця умова означає незалежність розподілів для нового стану автомата і його вихідного сигналу.

Нехай визначення вихідного сигналу залежить тільки від того стану, в якому автомат перебуває на даному такті роботи. Це означає, що кожен елемент множини Y індукує розподіл імовірностей виходів (табл. 1.13).

Таблиця 1.13

Розподіл імовірностей виходів елементів множини Y

Елементи з Y	y_1	y_2	…	y_{l-1}	y_l
z_s	r_1	r_2	…	r_{l-1}	r_l

Тут $\sum_{i=1}^{I} r_j = 1$, де r_i – ймовірність появи вихідного сигналу y_i за умови, що Р-автомат перебуває в стані z_k.

Якщо для всіх k та і має місце співвідношення $p_k r_i = b_{ki}$, то такий автомат називається ймовірнісним автоматом Мура.

Окремим випадком автоматів, що задаються як $P =< Z, X, Y, B >$, є автомати, в яких або перехід у новий стан, або вихідний сигнал визначаються детерміновано. Якщо вихідний сигнал Р-автомата визначається детерміновано, то такий автомат називається Y-детермінованим. Якщо перехід у новий стан є детермінованим, то Р-автомат називається Z-детермінованим.

Способи задання ймовірнісних автоматів такі самі, як і для детермінованих автоматів: табличний, графічний, матричний.

Розглянемо способи задання ймовірносних автоматів на прикладі.

Приклад 1. Нехай є Y-детермінований Р-автомат, який задається таблицею переходів (табл. 1.14) і таблицею виходів (табл. 1.15).

Таблиця 1.14

Переходи Y-детермінованого Р-автомата

z_k	z_k				
	z_1	z_2	…	z_{k-1}	z_k
z_1	p_{11}	p_{12}	…	$p_{1(k-1)}$	p_{1k}
z_2	p_{21}	p_{22}	…	$p_{2(k-1)}$ $p_{2(k-1)}$	p_{2k}
…	…	…	…	…	…
z_k	p_{k1}	p_{k2}	…	$p_{k(k-1)}$	p_{kk}

Тут p_{ij} – імовірність переходу Р-автомата зі стану z_i в стан z_j. При цьому, як і раніше, $\sum_{j=1}^{k} p_{ij} = 1$.

Таблиця 1.15

Виходи Y-детермінованого Р-автомата

Z	z_1	z_2	…	z_k
Y	y_{i1}	y_{i2}	…	y_{ik}

Для виходів імовірності не задаються, оскільки автомат є Y-детермінованим.

Дані табл. 1.14 можна подати у вигляді матриці, яку називають матрицею переходів:

$$P_p = \begin{Vmatrix} p_{11} & p_{12} & \dots & p_{1k} \\ p_{21} & p_{22} & \dots & p_{2k} \\ \dots & \dots & \dots & \dots \\ p_{k1} & p_{k2} & \dots & p_{kk} \end{Vmatrix}.$$

Для опису Y-детермінованого Р-автомата також необхідно задати початковий розподіл імовірностей (табл. 1.16).

Таблиця 1.16

Розподіл імовірностей Y-детермінованого Р-автомата

Z	z_1	z_2	…	z_k
D	d_1	d_2	…	d_k

Тут d_k – ймовірність того, що на початку роботи Р – автомат перебуває в стані Z_K. При цьому $\sum_{k=1}^{K} d_k = 1$.

Будемо вважати, що до початку роботи (до нульового такту часу) Р-автомат завжди перебуває в стані Z_0 і в нульовий такт часу змінює стан відповідно до розподілу D . Подальша зміна станів Р-автомата визначається матрицею переходів P_p. Інформацію про початковий стан Р-автомата зручно внести в матрицю P_p, збільшивши розмірність до $(k+1)(k+1)$. При цьому перший рядок такої матриці, який зіставляється зі станом z_0, матиме вигляд $(0, d_1, d_2, ..., d_k)$, а перший стовпець буде нульовим.

Описаний Y-детермінований Р-автомат можна задати у вигляді орієнтованого графа, вершини якого відповідають станам автомата, а дуги – можливим переходам з одного стану в інший. Дуги мають ваги, які відповідають можливостям переходу P_{ij}, а біля вершин графа пишуть значення вихідних сигналів, індукованих цими станами.

З точки зору математичного апарату задання У-детермінованого Р-автомата є еквівалентним заданню деякого дискретного марковського ланцюга і скінченною множиною станів. Тому для аналітичних розрахунків апарат марковських ланцюгів є основним при використанні Р-схем. Розглянемо застосування апарату марковських ланцюгів для Р-автоматів на прикладі.

Приклад 2. Нехай є Р-автомат, який задано матрицею переходів,

$$p=\begin{Vmatrix} 0 & 0,5 & 0 & 0 & 0,5 \\ 0 & 0 & 0 & 1,0 & 0 \\ 0 & 0 & 0,75 & 0 & 0,25 \\ 0 & 0 & 0,4 & 0 & 0,6 \\ 0 & 1,0 & 0 & 0 & 0 \end{Vmatrix}.$$

Відповідна таблиця виходів матиме такий вигляд (табл. 1.17).

Таблиця 1.17

Виходи Р-автомата, який задано матрицею переходів

Z	z_0	z_1	z_2	z_3	z_4
y	0	0	1	1	0

На рис. 1.17 показано граф переходів цього автомата.

Потрібно оцінити ймовірність видачі на виході одиниці. Одиниці відповідають фінальній імовірності перебування автомата в стані Z_2 або Z_3.

На значення фінальних імовірностей початковий стан не впливає, тому матриця фінальних імовірностей матиме вигляд

$$\bar{C}=\begin{vmatrix} 0 & 0 & 1,0 & 0 \\ 0 & 0,75 & 0 & 0,25 \\ 0 & 0,4 & 0 & 0,6 \\ 1,0 & 0 & 0 & 0 \end{vmatrix}=(C_1,C_2,C_3,C_4),$$

де C_k – фінальна ймовірність перебування Р-автомата в стані Z_k.

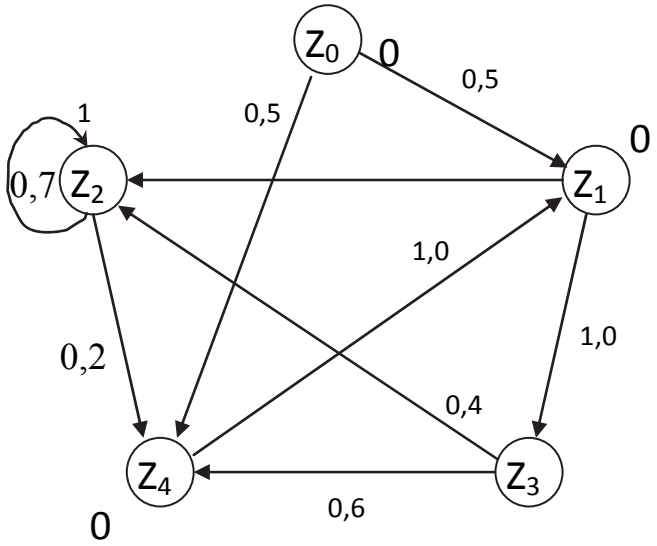


Рис. 1.17. Граф переходів імовірнісного Y-детермінованого автомата

При використанні аналітичного підходу можна записати відомі співвідношення з теорії марковських ланцюгів і отримати систему рівнянь для визначення фінальних імовірностей:

$$C_1 = C_4,$$
$$C_2 = 0,75C_2 + 0,4C_3;$$
$$C_4 = 0,25C_2 + 0,6C_3;$$
$$C_3 = C_1.$$

Додамо до цих рівнянь умову нормування $C_1 + C_2 + C_3 + C_4 = 1$.

Тоді, розв'язуючи систему рівнянь, отримаємо $C_1 = \frac{5}{23}$, $C_2 = \frac{8}{23}$, $C_3 = \frac{5}{23}$, $C_4 = \frac{5}{23}$. Таким чином, $C_2 + C_3 = \frac{13}{23} = 0,5652$. Іншими словами, при нескінченній роботі заданого в цьому прикладі Y-детермінованого P-автомата на його виході формується двійкова послідовність з імовірністю появи одиниці, що дорівнює 0,5652.

Для оцінювання різних характеристик досліджуваних систем, які подаються у вигляді P-схем, крім розглянутого випадку аналітичних моделей можна застосовувати й імітаційні моделі, які реалізуються, наприклад, методом статистичного моделювання.

1.5. Безперервно-стохастичні моделі (Q-схеми)

Теорія масового обслуговування є одним із розділів теорії ймовірностей. У цій теорії розглядаються ймовірнісні задання функцій і математичні моделі (до цього було досліджено детерміновані математичні моделі). Нагадаємо, що детермінована математична модель відображає поводження об'єкта (системи, процесу) з позицій повної визначеності в сьогоденні й майбутньому.

Імовірнісна математична модель враховує вплив випадкових факторів на поводження об'єкта (системи, процесу) і, отже, оцінює майбутнє з позицій імовірності тих чи інших подій, тобто в цій моделі, як у теорії ігор, задання об'єкта розглядається в умовах невизначеності.

Наведемо спочатку деякі поняття, які характеризують стохастичну невизначеність. Це коли невизначені фактори, що входять до задачі, являють собою випадкові величини (або випадкові функції), імовірнісні характеристики яких або є відомими, або їх можна отримати з досліду. Таку невизначеність ще називають сприятливою, доброякісною.

Поняття випадкового процесу

Строго кажучи, випадкові збурення притаманні будь-якому процесу. Простіше навести приклади випадкового, ніж невипадкового процесу. Навіть, наприклад, процес ходу годинника (начебто це суворо вивірена робота – «працює як годинник») може випадково змінюватися (догляд наперед, відставання, зупинення). Але доти, доки ці збурення є неістотними, мало впливають на необхідні параметри, можна ними нехтувати і процес розглядати як детермінований, невипадковий.

Нехай є деяка система S (технічний пристрій, група таких пристроїв, технологічна система – верстат, дільниця, цех, підприємство, галузь промисловості). У системі S відбувається випадковий процес, якщо вона протягом часу змінює свій стан (переходить з одного стану в інший), причому заздалегідь невідомим випадковим чином.

Марковський випадковий процес. Випадковий процес, що відбувається в системі, називається марковським, якщо для будь-якого моменту часу t_0 імовірнісні характеристики процесу в майбутньому залежать тільки від його стану в ций момент t_0 і не залежать від того, коли і як система перейшла в цей стан.

Нехай у момент t_0 система перебуває в певному стані S_0. Відомо характеристики стану системи в реальному часі $t_0 \to S_0$ і все, що було при $t < t_0$ (передісторію процесу). Чи можна передбачити, що буде при $t > t_0$? Точно – ні, але якісь ймовірнісні характеристики процесу в майбутньому знайти можна (наприклад, імовірність того, що через деякий час τ система S набуде стану S_1 або залишиться в стані S_0 і т. ін.

Приклад. Система S – група літаків, що беруть участь у повітряному бою. Нехай x – кількість «червоних» літаків, у – кількість «синіх» літаків. До моменту часу t_0 кількість збережених (незбитих) літаків відповідно становила x_0 і y_0. Слід визначити ймовірність того, що в момент часу $t_0 + \tau$ кількісно переважатимуть «червоні» літаки. Ця ймовірність залежить від того, в якому стані перебувала система в момент часу t_0, а не від того, коли і в якій послідовності гинули збиті до моменту t_0 літаки.

На практиці марковські процеси в чистому вигляді зазвичай не існують. Але є процеси, для яких впливом «передісторії» можна знехтувати, і при вивченні таких процесів можна застосовувати марковські моделі (у теорії масового обслуговування розглядаються і немарковські системи масового обслуговування, але математичний апарат, що їх описує, є набагато складнішим).

У дослідженні операцій велике значення мають марковські випадкові процеси з дискретними станами і безперервним часом.

Процес називається процесом з дискретним станом, якщо його можливі стани S_1, S_2, ... можна заздалегідь визначити і перехід системи зі стану в стан відбувається «стрибком», майже миттєво.

Процес називається процесом з безперервним часом, якщо моменти можливих переходів зі стану в стан не фіксовані заздалегідь, а невизначені, випадкові й можуть статися в будь-який момент.

Далі розглянемо тільки процеси з дискретним станом і безперервним часом.

Приклад. Технологічна система (ділянка) S складається з двох верстатів, кожен з яких у випадковий момент часу може вийти з ладу (відмови-

ти), після чого миттєво починається ремонт вузла, який триває заздалегідь невідомий випадковий час. Можливі такі стани системи:

S_0 – обидва верстати є справними;

S_1 – перший верстат ремонтують, другий є справним;

S_2 – другий верстат ремонтують, перший є справним;

S_3 – обидва верстати ремонтують.

Переходи системи S зі стану в стан відбуваються майже миттєво в випадкові моменти виходу з ладу того чи іншого верстата або після закінчення ремонту.

При аналізі випадкових процесів з дискретними станами зручно користуватися геометричною схемою – графом станів. Вершини графа – стани системи. Дуги графа – можливі переходи зі стану в стан.

Потоки подій

Потік подій – послідовність однорідних подій, які настають одна за одною в якісь випадкові моменти часу.

У попередньому прикладі це потоки відмов і відновлень. Інші приклади: потік викликів на телефонній станції, потік покупців до магазину і т. д.

Потік подій можна наочно зобразити біля точок на осі часу $0-t$ (рис. 1.18).

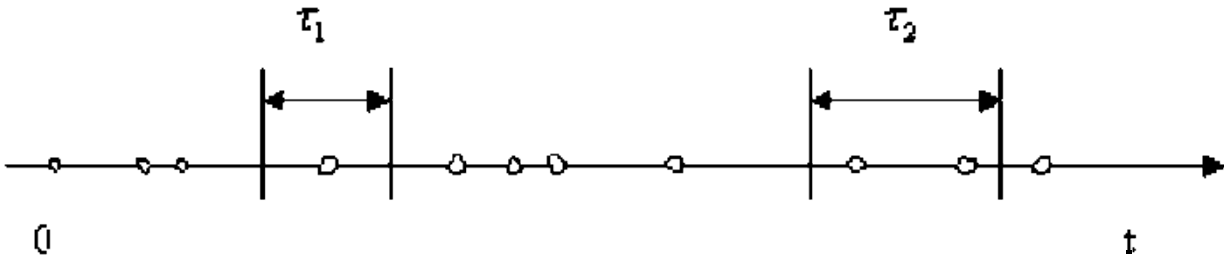


Рис. 1.18. Зображення потоку подій на осі часу

Положення кожної точки є випадковим, і тут зображено лише якусь одну реалізацію потоку.

Інтенсивність потоку подій (λ) – це середня кількість подій, що припадає на одиницю часу.

Розглянемо деякі властивості (види) потоків подій.

Потік подій називається стаціонарним, якщо його ймовірнісні характеристики не залежать від часу.

Зокрема, інтенсивність λ стаціонарного потоку є постійною. Потік подій неминуче має скупчення або розрідження, але вони не є закономірни-

ми, і середня кількість подій, що припадає на одиницю часу, є сталою і від часу не залежить.

Потік подій називається потоком без наслідків, якщо для будь-яких двох непересічних інтервалів часу τ_1 і τ_2 (рис.1.19) кількість подій, що потрапляють на один з них, не залежить від того, скільки подій потрапило на інший. Це означає, що події, що утворюють потік, виникають в ті чи інші моменти часу незалежно одна від одної і через свої власні причини.

Потік подій називається ординарним, якщо події в ньому вникають поодинці, а не групами по кілька одразу.

Потік подій називається найпростішим (або стаціонарним пуасонівським), якщо він є стаціонарним, ординарним і не має наслідків.

Найпростіший потік має простий математичний опис. Він відіграє серед потоків таку саму особливу роль, як і закон нормального розподілу серед інших законів розподілу, тобто, при накладенні досить великої кількості незалежних, стаціонарних і ординарних потоків (порівнянних між собою за інтенсивністю) формується потік, близький до найпростішого.

Для найпростішого потоку з інтенсивністю інтервал Т між сусідніми подіями має так званий показниковий (експоненціальний) розподілі з щільністю $f(t) = \lambda e^{-\lambda t}$, де λ – параметр експоненціального закону.

Для випадкової величини Т, що має показниковий розподіл, математичне сподівання m_T є величиною, оберненою до параметра, а середнє квадратичне відхилення σ_T дорівнює математичному сподіванню $m_T = \sigma_T = \frac{1}{\lambda}$.

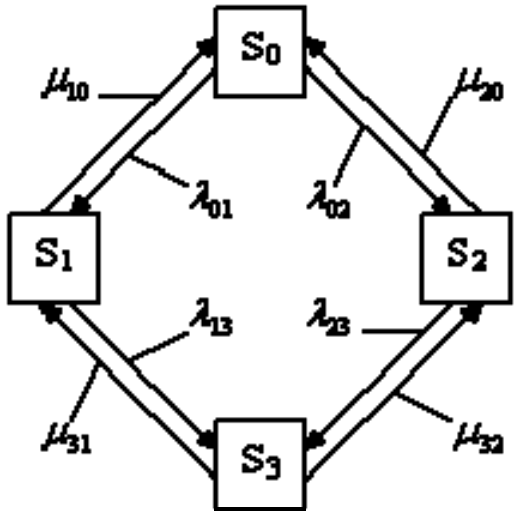


Рис.1.19. Розмічений граф станів системи

Рівняння Колмогорова для ймовірностей станів. *Фінальні ймовірності станів*

Під марковськими процесами з дискретними станами і безперірвним часом розуміється що всі переходи системи S зі стану в стан відбуваються під дією найпростіших потоків подій (потоків викликів, потоків відмов, потоків відновлень і т. д.). Якщо всі потоки подій, що переводять систему S зі стану в стан, є найпростішим, то процес, що відбу-

вається в системі, буде марковським.

Отже, на систему, що перебуває в стані S_i, діє найпростіший потік подій. При виникненні першої події цього потоку відбувається «перескок» системи зі стану S_i в стан S_j (на графі станів по стрілці $S_i \to S_j$).

Для наочності на графі станів системи біля кожної дуги вказують інтенсивності того потоку подій, який переводить систему по цій дузі (позначають стрілкою). λ_{ij} – інтенсивність потоку подій, що переводить систему зі стану S_i в стан S_j. Такий граф називається розміченим. Для розглядуваного прикладу розмічений граф наведено на рис. 1.19, де λ_{ij} – інтенсивності потоку відмов; μ_{ij} – інтенсивності потоку відновлень.

Припускаємо, що середній час ремонту верстата не залежить від того, ремонтується один верстат чи обидва відразу, тобто ремонтом кожного верстата займається окремий фахівець.

Нехай система перебуває в стані S_0. У стан S_1 її переводить потік відмов першого верстата. Його інтенсивність

$$\lambda_{01} = \frac{1}{T_{сер.роб.1}}, од.часу^{-1},$$

де $T_{сер.роб.1}$ – середній час безвідмовної роботи першого верстата.

Зі стану S_1 у стан S_0 систему переводить потік «закінчень ремонтів» першого верстата. Його інтенсивність

$$\mu_{10} = \frac{1}{T_{сер.рем.1}}, од.часу^{-1},$$

де $T_{сер.рем.1}$ – середній час ремонту першого верстата.

Аналогічно обчислюються інтенсивності потоків подій, що переводять систему по всім дугам графа. На основі розміченого графа станів системи будується математична модель цього процесу.

Нехай ця система S має $S_1, S_2, ..., S_n$ можливі стани. Ймовірність i-го стану $p_i(t)$ – це ймовірність того, що в момент часу t система буде перебувати в стані S_i. Очевидно, що для будь-якого моменту часу сума всіх імовірностей станів дорівнює одиниці:

$$\sum_{i=1}^{n} p_i(t) = 1.$$

Для знаходження всіх імовірностей станів $p_i(t)$ як функцій часу необхідно скласти й розв'язати рівняння Колмогорова (це особливий вид рів-

нянь, в яких невідомими функціями є ймовірності станів). Правило складання цих рівнянь наведемо без доказів. Але спочатку пояснимо, що означає поняття «фінальна ймовірність стану».

Що буде відбуватися з імовірностями станів при $t \to \infty$? Чи будуть $p_1(t), p_2(t), \ldots$ прямувати до якихось меж? Якщо ці межі існують і не залежать від початкового стану системи, то вони називаються фінальними ймовірностями станів:

$$\lim_{t\to\infty} p_i(t) = p_i, i = \overline{1,n},$$

де n – скінченна кількість станів системи.

Фінальні ймовірності станів – це вже не змінні величини (функції часу), а сталі числа. Очевидно, що

$$\sum_{i=1}^{n} p_i = 1.$$

Фінальна ймовірність стану S_i – це по суті середній відносний час перебування системи в цьому стані.

Наприклад, система S має три стани – S_1, S_2 і S_3. Їх фінальні ймовірності дорівнюють 0,2; 0,3 і 0,5 відповідно. Це означає, що система в граничному стаціонарному стані в середньому 2/10 часу перебуває в стані S_1, 3/10 – у стані S_2 і 5/10 – у стані S_3.

Правило складання системи рівнянь Колмогорова: у кожному рівнянні системи в лівій його частині знаходиться фінальна ймовірність стану p_i, помножена на сумарну інтенсивність у всіх потоків, що ведуть з цього стану, а в правій його частині – сума інтенсивностей всіх потоків, що входять в i-й стан, і ймовірностей тих станів, з яких ці потоки виходять.

Користуючись цим правилом, напишемо систему рівнянь для розглядуваного прикладу:

$$\begin{cases} (\lambda_{01} + \lambda_{02}) p_0 = \mu_{10} p_1 + \mu_{20} p_2; \\ (\mu_{10} + \lambda_{13}) p_1 = \lambda_{01} p_0 + \mu_{31} p_3; \\ (\mu_{20} + \lambda_{23}) p_2 = \lambda_{02} p_0 + \mu_{32} p_3; \\ (\mu_{31} + \mu_{32}) p_3 = \lambda_{13} p_1 + \lambda_{23} p_2. \end{cases} \quad (1.17)$$

Систему чотирьох рівнянь (1.17) з чотирма невідомими p_0, p_1, p_2, p_3, здавалося б, можна розв'язати цілком, але ці рівняння є однорідними (не мають вільного члена) і, отже, визначають невідомі тільки з точністю до довільного множника. Однак можна скористатися нормувальною умовою $p_0 + p_1 + p_2 + p_3 = 1$ і з її допомогою розв'язати систему. При цьому одне (будь-яке) з рівнянь можна відкинути (воно випливає як наслідок з інших).

Продовження прикладу. Нехай значення інтенсивностей потоків

$$\lambda_{01} = \lambda_{23} = 1,\ \lambda_{13} = \lambda_{02} = 2,\ \mu_{10} = \mu_{32} = 2,\ \mu_{20} = \mu_{31} = 3.$$

Четверте рівняння відкидаємо, додаючи замість нього нормувальну умову:

$$\begin{cases} 3p_0 = 2p_1 + 3p_2; \\ 4p_1 = p_0 + 3p_3; \\ 4p_2 = 2p_0 + 2p_3; \\ p_0 + p_1 + p_2 + p_3 = 1; \end{cases}$$

$$p_0 = \frac{6}{15} = 0,4; \quad p_1 = \frac{3}{15} = 0,2; \quad p_2 = \frac{4}{15} \cong 0,27; \quad p_3 = \frac{2}{15} \cong 0,13.$$

Іншими словами, у граничному й стаціонарному режимах система S в середньому 40 % часу буде перебувати в стані S_0 (обидва верстати є справними), 20 % – у стані S_1 (перший верстат ремонтують, другий працює), 27 % – у стані S_2 (другий верстат ремонтують, перший працює), 13 % – у стані S_3 (обидва верстати ремонтують). Знання цих фінальних імовірностей може допомогти оцінити середню ефективність роботи системи і завантаження ремонтних органів.

Нехай система S у стані S_0 (повністю є справною) дає за одиницю часу дохід 8 умовних одиниць, у стані S_1 – дохід 3 умовні одиниці, у стані S_2 – дохід 5 умовних одиниць, в стані S_3 – не дає доходу. Тоді в граничному й стаціонарному режимах середній дохід за одиницю часу буде $W = 0,4 \cdot 8 + 0,2 \cdot 3 + 0,27 \cdot 5 = 5,15$ умовних одиниць.

Верстат 1 ремонтують частку часу, що становить $p_1 + p_3 = 0,2 + 0,13 = 0,33$. Верстат 2 ремонтують частку часу, що дорівнює $p_2 + p_3 = 0,27 + 0,13 = 0,4$. Постає питання оптимізації. Нехай можна зменшити середній час ремонту першого або другого верстата (або обох), але на це треба використати певну суму. Чи окупить збільшення доходу, пов'язане з прискоренням ремонту, підвищені витрати на ремонт? Потрібно буде розв'язати систему чотирьох рівнянь з чотирма невідомими.

1.5.1. Система масового обслуговування як модель

Система масового обслуговування (СМО) – одна з основних моделей, які використовують інженери системотехніки. Коротко її опишемо.

Заявки (вимоги) на обслуговування надходять через постійні або випадкові інтервали часу. Прилади (канали) використовуються для обслуговування цих заявок. Обслуговування триває деякий час, що задано постійним або випадковим параметром. Якщо в момент надходження заявки всі прилади зайняті, заявка поміщається в комірку буфера і чекає там початку

обслуговування. Заявки, що знаходяться в буфері, складають чергу на обслуговування. Якщо всі комірки буфера зайняті, заявка отримує відмову в обслуговуванні й втрачається. Імовірність втрати заявки (ймовірність відмови) – одна з основних характеристик СМО. Інші характеристики: середній час очікування початку обслуговування, середня довжина черги, коефіцієнт завантаження приладу (частка часу, протягом якого прилад займається обслуговуванням) і т. д.

Залежно від обсягу буфера розрізняють СМО з відмовами, де немає буфера, СМО з очікуванням, де буфер є необмеженим (наприклад, черга до магазину на вулиці), і СМО змішаного типу, де буфер має скінченну кількість заявок. У СМО з відмовами немає черги, у СМО з очікуванням немає втрат заявок, у СМО змішаного типу і те, й інше є можливим.

Іноді заявки розрізняють за їх пріоритетом, тобто за важливістю. Заявки високого пріоритету обслуговуються в першу чергу. Абсолютний пріоритет дає право перервати обслуговування менш важливої заявки і зайняти її місце в приладі (або буфері, якщо всі прилади зайняті настільки ж важливими заявками). Витіснена заявка або втрачається, або поміщається в буфер, де чекає початку обслуговування. Іноді доводиться відновлювати обслуговування витісненої заявки з початку, а не продовжувати з точки переривання. Якщо заявку витіснено з буфера, вона, природно, втрачається. Прикладом заявки з абсолютним пріоритетом є судно, яке отримало пробоїну і потребує термінового розвантаження. В обчислювальних системах абсолютний пріоритет мають команди оператора. Відносний пріоритет дає право першочергового заняття приладу, що звільнився, але не дає права на витіснення заявки з приладу або буфера. Особи, які користуються пільгами при обслуговуванні в касі, у лікаря тощо, зазвичай мають відносний пріоритет. Абсолютний і відносний пріоритети розрізняються і моментом дії: абсолютний реалізується в момент надходження, а відносний – у момент звільнення приладу.

Розрізняють фіксовані й динамічні пріоритети. Фіксовані пріоритети частіше називають дисципліною обслуговування.

Дисципліна обслуговування задає порядок вибору з черги заявок однакового пріоритету в прилад, що звільнився. Виділимо такі дисципліни: FIFO (First Input - First Output): першим прийшов – першим обслужений, LIFO (Last Input - First Output): останнім прийшов – першим обслужений, RAND (Random): випадковий вибір з черги. У побуті зазвичай діє дисципліна FIFO. Дисципліна LIFO реалізується в буфері, організованому за принципом стека. Така дисципліна може виявитися доцільною, наприклад, при передачі інформації, якщо її цінність швидко падає з часом.

У теорії масового обслуговування важливим є поняття раптового потоку як деякої послідовності подій, що відбуваються у випадкові моменти часу.

Випадковий потік можна задати функцією розподілу величини проміжку (інтервалу) часу між моментами настання подій $\tau_j = t_j - t_{j-1}$, $P(\tau_j \le t)$.

Якщо величини τ_j є незалежними в сукупності, то потік – обмежений післядією.

У випадку $P(\tau_j \le t) = P(\tau \le t)$ для всіх $j \ge 2$ потік є рекурентним. Рекурентний потік, для якого $P(\tau \le t) = 1 - \exp(-\lambda t)$, називається пуасонівським. Для такого потоку ймовірність настання за проміжок часу $[0,t]$ n подій є $P_n(t) = \frac{(\lambda t)^n}{n!}\exp(-\lambda t)$, а математичне сподівання кількості подій, що настали за час t, λt, де λ – середня кількість подій, що відбуваються за одиницю часу.

Пуасонівський потік характеризується відсутністю післядії.

Якщо крім цього виконуються умови стаціонарності й ординарності, то пуасонівський потік буде найпростішим.

Нагадаємо, що для стаціонарного потоку розподілення залежить тільки від тривалості, а не від положення інтервалу τ на осі часу. Відсутність післядії означає незалежність кількості подій у неперекривних інтервалах. Властивість ординарності полягає в тому, що ймовірність появи більше однієї події на нескінченно малому інтервалі має порядок малості вищий, ніж вірогідність появи однієї події на цьому інтервалі.

Величину λ в разі пуасонівського потоку називають інтенсивністю потоку подій. Якщо $\tau_j = \tau = const$, то потік є регулярним, або детермінованим. Для позначення типу СМО Кендалл і Башарін [16,17] запропоновали систему позначень, що мають вигляд $\Delta|\Theta|\Xi|\Omega$, де Δ – позначення закону розподілу ймовірність для інтервалів надходження заявок, Θ – позначення закону розподілу ймовірностей для часу, Ξ – кількість каналів обслуговування, Ω – кількість місць у черзі.

Закони розподілу в позиціях Δ і Θ позначають зазвичай такими літерами:

M – експоненціальний;

E_k – ерлангівського порядку k;

R – рівномірний;

D – детермінований (стала величина);

G – довільний (будь-якого вигляду) і т. д.

Якщо кількість місць у черзі є необмеженою, то позиція Ξ не вказується. Наприклад, $M|M|1$ означає найпростішу СМО (обидва розподіли є експоненціальними, канал обслуговування один, черга є необмеженою), а позначення $R|D|2|100$ відповідає СМО з рівномірним розподілом інтервалів надходження вимог, фіксованим часом їх обслуговування, двома кана-

лами і 100 місцями в черзі. У цій СМО заявки, які надходять у моменти, коли всі місця в черзі зайнято, залишають систему (тобто втрачаються).

Якщо в СМО надходить n потоків заявок (кожний потік має пріоритет), то Δ і Θ приписують число n у вигляді індексу. Наприклад, $M_2 \mid M_2 \mid 1$ позначає СМО з двома потоками заявок, які на вході мають експоненціальний розподіл з експоненціальним часом обслуговування, своїм для кожного потоку. У системі $M_2 \mid M \mid 1$ час обслуговування всіх заявок має один і той самий розподіл. У разі декількох вхідних потоків, що мають різні пріоритети, необхідно додатково указати типи пріоритетів – абсолютні, відносні.

1.5.2. Одноканальна однорідна експоненціальна СМО

Одноканальна експоненціальна СМО визначається такими властивостями: має канал, до неї приходять заявки, якщо СМО є порожньою (немає заявок), то заявка, що приходить, займає канал.

Заявка, що приходить у непорожню СМО, стає в чергу останньою. Будь-яка заявка, яка зайняла канал, обслуговується, звільняє канал і йде з СМО. Якщо в момент відходу черга є непорожньою, то перша в ній заявка виходить з черги і займає канал. Колом (рис. 1.20) позначено канал К, трьома прямокутниками – чергу.

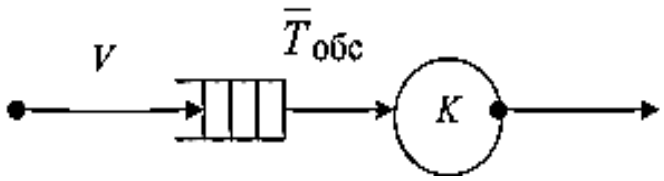


Рис. 1.20. Позначення в СМО

Стрілки вказують напрямок руху заявок, точки біля стрілок – вхід і вихід СМО. Парафії заявок прогнозують пуасонівський потік подій. Це означає, що час між парафіями будь-яких двох послідовних заявок є незалежною випадковою величиною з експоненціальною функцією розподілу ймовірностей $F(X) = 1 - e^{-VX}$.

Параметр V є інтенсивністю потоку заявок, тобто середнім числом заявок, що приходять за одиницю часу, яке дорівнює V. Надалі інтенсивність приходу заявок у СМО будемо позначати через λ. Час обслуговування заявки також є незалежною випадковою величиною з експоненціальною функцією розподілу ймовірностей вигляду (1.1). Але параметр V у цьому випадку має інше значення. Будемо позначати його через μ. Величину $\frac{1}{\mu}$, яка дорівнює середньому часу обслуговування заявки, позначимо через $T_{обс.}$.

У вигляді одноканальної експоненціальної СМО можна промоделювати, наприклад, периферійний пристрій мультипрограмної обчислювальної системи. Тоді парафії заявок будуть відповідати зверненнями програм

до пристрою для виконання операції введення або виведення інформації, λ буде інтенсивністю таких звернень, $T_{обс}$ – середнім часом виконання необхідної операції.

Одноканальна експоненціальна СМО задається параметрами λ, $T_{обс}$. Мета її аналізу – розрахунок показників, найважливішими з яких є такі:

- коефіцієнт завантаження ρ;
- середня довжина черги L;
- середнє число заявок у СМО M;
- середній час очікування обслуговування $T_{оч}$;
- середній час перебування заявки в СМО $T_{пер}$.

Коефіцієнт завантаження розраховується за формулою

$$\rho = \lambda \overline{T}_{обс}. \tag{1.18}$$

Якщо виконується умова

$$\rho \le 1, \tag{1.19}$$

то існує стаціонарний режим функціонування СМО. У стаціонарному режимі всі ймовірні характеристики системи є сталими в часі величинами. Події, які самі відбуваються в СМО, залишаються при цьому випадковими. Якщо (1.3) не виконується, то стаціонарного режиму в СМО не існує.

У стаціонарному режимі середнє число M заявок у СМО є сталим, тому середнє число заявок, що приходять в СМО за одиницю часу, дорівнює середньому числу заявок, що йдуть зі СМО, за одиницю часу. Отже, у стаціонарному режимі інтенсивність потоку заявок, які йдуть, дорівнює λ. Коефіцієнт завантаження ρ у стаціонарному режимі є:

а) середнім значенням тієї частини одиниці часу, протягом якої канал є зайнятим;

б) імовірністю того, що канал є зайнятим;

в) середнім числом заявок у каналі (у подальшому будемо говорити тільки про стаціонарні значення параметрів.

Середня довжина черги (середнє число заявок в черзі) в одноканальній експоненціальній СМО розраховується за формулою

$$L = \frac{\rho^2}{1-\rho}. \tag{1.20}$$

Середнє число M заявок у СМО дорівнює сумі середнього числа L заявок у черзі й середнього числа ρ заявок у каналі:

$$M = \frac{\rho}{1-\rho}. \tag{1.21}$$

Заявка переміщається в черзі в середньому зі сталою швидкістю. Середнє число переходів заявки в черзі на одне місце вперед за одиницю часу однієї λ. При такій швидкості переміщення L переходів відбудеться за час

$$\overline{T}_{оч} = \frac{\overline{T}_{обс}\rho}{1-\rho}. \qquad (1.22)$$

Формула (1.22) дає середній час проходження заявки через чергу. Це є середнім часом очікування. Середній час перебування заявки в СМО є сумою середнього часу очікування і середнього часу обслуговування заявки:

$$\overline{T}_{пер} = \frac{\overline{T}_{обс}}{1-\rho}. \qquad (1.23)$$

Імовірність наявності в системі k вимог визначається за допомогою геометричного закону розподілу у вигляді

$$(1-\rho)\rho^k, k = 0,1,2,...$$

Характеристики (1.18) – (1.23) можуть давати цінну інформацію про моделювання у вигляді СМО систем. Нехай, наприклад, СМО зображує периферійний пристрій обчислювальної системи. Тоді ρ дорівнює коефіцієнту використання пристрою, а $(1-\rho)$ – коефіцієнту простою. Необхідно, щоб коефіцієнт використання був досить великим. Величина $T_{оч}$ характеризує середній час, протягом якого програми очікують звільнення пристрою. У цей час програми фактично «простоюють». Бажано, щоб він був досить малим.

Багатоканальна експоненціальна СМО відрізняється від одноканальної тим, що вона має більше одного каналу. Якщо всі канали є зайнятими, то заявка, що приходить, стає в чергу. В іншому випадку заявка займає вільний канал.

1.5.3. Багатоканальна експоненціальна СМО

Багатоканальна експоненціальна СМО задається трьома параметрами: інтенсивністю V приходу заявок, середнім часом $T_{обс}$ обслуговування заявок і кількістю К каналів (рис. 1.21).

Формули для розрахунку показників багатоканальної експоненційної СМО є трохи складнішим за (1.18) – (1.23). Коефіцієнт завантаження визначається за формулою

$$\rho = \frac{\lambda \overline{T}_{обс}}{K}. \qquad (1.24)$$

Його значення має відповідати умовній стаціонарності (1.19). Середня довжина черги в блоці очікування

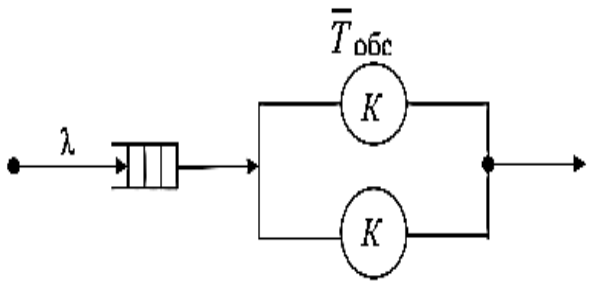


Рис. 1.21. Двоканальна СМО

$$L = \beta_0 \frac{(\lambda \overline{T}_{обс})^{K+1}}{K! K \left(1 - \frac{\lambda \overline{T}_{обс}}{K}\right)^2}, \quad (1.25)$$

де β_0 – стаціонарна ймовірність того, що в СМО немає заявок,

$$\beta_0 = \frac{1}{\frac{(\lambda \overline{T}_{обс})^K}{K! \left(1 - \frac{\lambda \overline{T}_{обс}}{K}\right)} + \sum_{m=0}^{K-1} \frac{(\lambda \overline{T}_{обс})^m}{m!}}. \quad (1.26)$$

Інші характеристики можна обчислити через параметри СМО таким чином:

$$M = L + Kp; \quad (1.27)$$

$$\overline{T}_{ож} = \frac{L}{\lambda}; \quad (1.28)$$

$$\overline{T}_{пр} = \overline{T}_{ож} + \overline{T}_{обс}. \quad (1.29)$$

Багатоканальну СМО можна поставити у відповідність, наприклад, багатопроцесорному блоку обчислювальної системи, яка має загальну пам'ять для всіх процесорів і, отже, загальну чергу завдань.

1.5.4. Мережі масового обслуговування

Мережа масового обслуговування (ММО) являє собою сукупність скінченного числа N обслуговуючих вузлів, в якій циркулюють заявки, що переходять відповідно до маршрутної матриці з одного вузла в інший. Вузел завжди є розімкнутою СМО (СМО може бути будь-якого класу). При цьому окремі СМО відображають функціонально самостійні частини реальної системи, зв'язки між СМО – структуру системи, а вимоги, які циркулюють у ММО, – складові матеріальних потоків (повідомлення (пакети) в комунікаційній мережі, завдання в мультипроцесорних системах, контейнери вантажопотоків і т. ін.). Для наочного уявлення ММО використовується граф, вершини якого (вузли) відповідають окремим СМО, а дуги відображають зв'язки між вузлами. Перехід заявок між вузлами відбувається миттєво відповідно до перехідних можливостяей p_{ij}, $i, j = \overline{1, N}$; p_{ij} − імовірність того, що заявка після обслуговування у вузлі i перейде у вузол j. Природно, якщо вузли безпосередньо не пов'язані між собою, то $p_{ij} = 0$. Якщо з і-го вузла є перехід тільки в один який-небудь вузол j, то $p_{ij} = 1$. ММО класифікують за кількома ознаками (рис. 1.22).

Мережа називається лінійною, якщо інтенсивності потоків заявок у

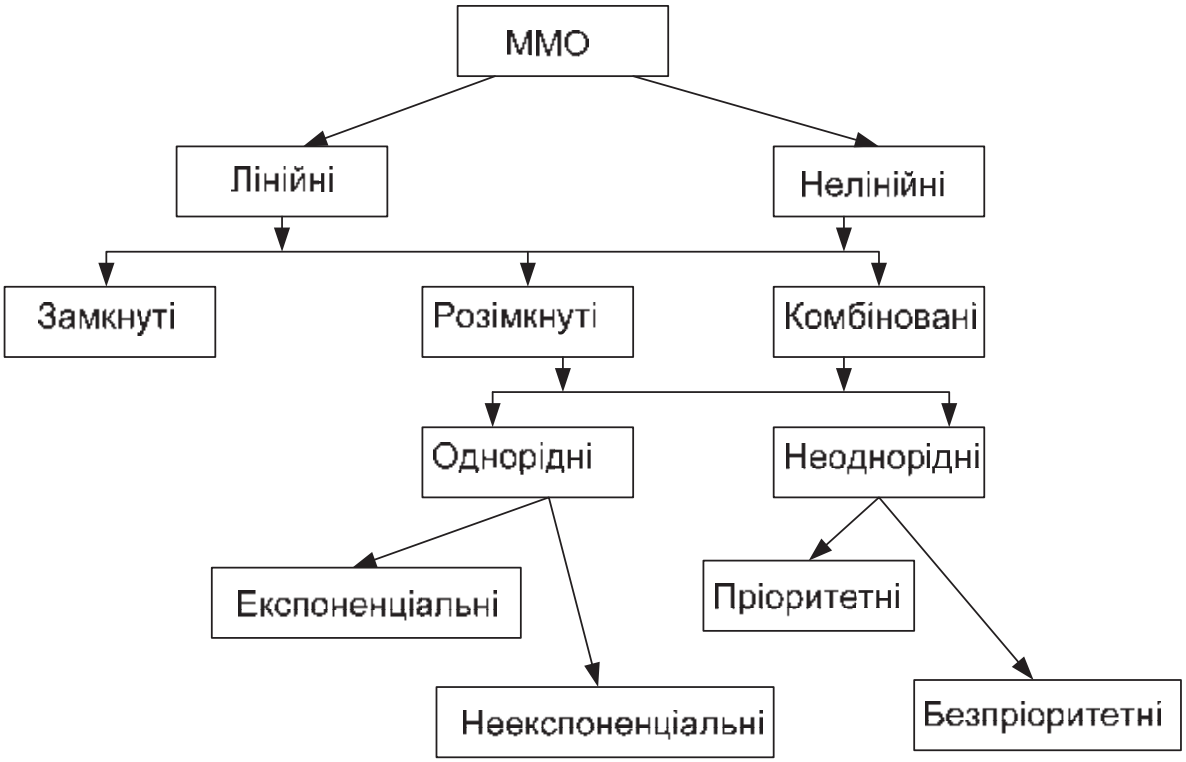


Рис. 1.22. Класифікація мереж масового обслуговування

вузлах пов'язані між собою лінійною залежністю $\lambda_j = \alpha_{ij}\lambda_i$, де α_{ij} – коефіцієнт пропорційності, або відносно джерела $\lambda_j = \alpha_j\lambda_0$. Коефіцієнт α_j називають коефіцієнтом передачі, який характеризує частку заявок, що надходять в j-й вузол від джерела заявок, або середнім числом проходжень заявкою через певний вузол за час перебування заявки в мережі. Якщо інтенсивності потоків заявок у вузлах мережі пов'язані нелінійною залежністю (наприклад, $\lambda_j = \sqrt{\alpha_j\lambda_0}$), то мережу називають нелінійною.

Мережа завжди є лінійною, якщо в ній заявки не втрачаються і не розмножуються. Розімкнута мережа – це така відрита мережа, в яку заявки надходять із зовнішнього середовища і йдуть після обслуговування з мережі в зовнішнє середовище. Іншими словами, особливістю розімкнутої ММО (РММО) є наявність одного або декількох незалежних зовнішніх джерел, які генерують заявки, що надходять в мережу, незалежно від того, скільки заявок вже находиться в мережі. У будь-який момент часу в РММО може знаходитися довільна кількість заявок (від 0 до ∞). У замкнутій ММО (ЗММО) циркулює фіксована кількість заявок, а зовнішнє незалежне джерело відсутнє. Виходячи з фізичних міркувань, у ЗММО вибирається зовнішня дуга, на якій зазначається псевдонульова точка, відносно якої можуть вимірюватися тимчасові характеристики. Комбінована мережа – це мережа, в якій постійно циркулює певна кількість заявок і є заявки, що надходять від зовнішніх незалежних джерел. У однорідній мережі циркулю-

ють заявки одного класу. І, навпаки, в неоднорідній мережі можуть знаходитися заявки кількох класів. Заявки належать до різних класів, якщо вони відрізняються хоча б одним з таких атрибутів: законом розподілу тривалості обслуговування в вузлах; пріоритетами; маршрутами (шляхами руху заявок у мережі).

У експоненціальній мережі тривалості обслуговування у всіх вузлах розподілені за експоненціальним законом, і потоки, що надходять у розімкнуту мережу, є найпростішими (пуасонівськими). У всіх інших випадках мережа є неекспоненціальною. Якщо хоча б в одному вузлі здійснюється пріоритетне обслуговування, то це пріоритетна мережу. Пріоритет – це ознака, що визначає черговість обслуговування. Якщо обслуговування заявок у вузлах здійснюється в порядку надходження, то така мережа є безпріоритетною. Таким чином, експоненціальною будемо називати ММО, що відповідає вимогам: вхідні потоки ММО є пуасонівськими; у всіх N СМО час обслуговування заявок має експонентну функцію розподілу ймовірностей і заявки обслуговуються в порядку приходу; перехід заявки з виходу i-ї СМО на вхід j-ї є незалежною випадковою подією, що має ймовірність p_{ij} $i, j = 1 \;\; N_i$; p_{i0} – ймовірність розгляду заявки з ММО. Якщо заявки приходять в мережу і не йдуть з неї, то мережа називається розімкнутою. Якщо заявки не приходять в мережу і з неї не йдуть, мережа називається замкнутою. Кількість заявок у замкнутій мережі є постійною.

1.5.5. Властивості розімкнутої експоненціальної ММО

Повторимо визначення мережі масового обслуговування. ММО називають сукупність СМО, в якій заявки з виходів одних СМО можуть надходити на входи інших. Вхідним потоком заявок СМО будемо називати потік заявок, що приходять на вхід окремої СМО із зовнішнього середовища ММО, тобто не з виходу будь-якої СМО. У загальному випадку кількість вхідних потоків ММО дорівнює кількості СМО. Розімкнута експоненціальна ММО задається такими параметрами:

1) кількістю N СМО;

2) кількістю $K_1,...,K_N$ каналів у СМО $1,...,N$;

3) матрицею $P = \left\| p_{ij} \right\|$ ймовірностей передач, $i = 1,...,N$; $j = 0,...,N$;

4) інтенсивністю $I_1,...,I_N$ вхідних потоків заявок;

5) середнім часом обслуговування $\overline{T}_{обс1},...,\overline{T}_{обсN}$ заявок у СМО.

Наприклад, ММО (рис. 1.23) задано чисельно в такому вигляді:

1) $N = 3$;

2) $K_1 = 1$; $K_2 = 1$; $K_3 = 2$;

3) $P = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}\begin{bmatrix} 0,1 & 0 & 0,5 & 0,4 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$;

4) $I_1 = 1;\ I_2 = 0;\ I_3 = 0;$

5) $T_{обс1} = 0,07;\ T_{обс2} = 0,06; T_{обс3} = 0,35.$

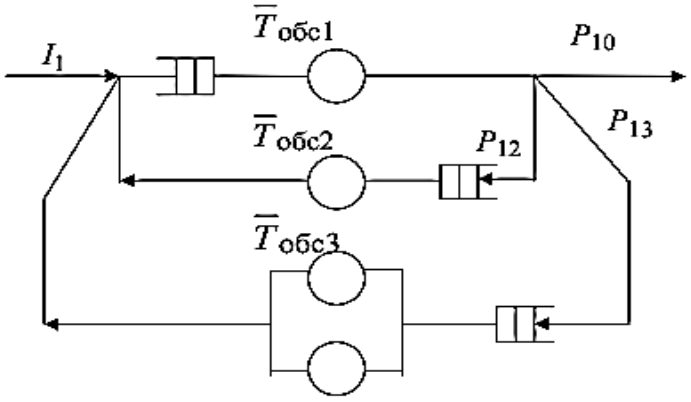


Рис. 1.23. Чисельне задання ММО

У експоненційній ММО потік заявок на вході СМО складається з вхідного потоку ММО (можливо, має нульову інтенсивність) і потоків, що надходять з виходів СМО. Вхідний потік СМО в експоненціальній ММО в загальному випадку є непуасонівським. Це означає, що СМО в ній у загальному випадку є неекспоненціальним. Проте досить часто вважають, що СМО поводяться в ній як експоненціальні. Зокрема, характеристики СМО відповідають (1.18) – (1.23). Тому для їх розрахунку в заданій ММО досить знайти інтенсивності $\lambda_1,...,\lambda_N$ вхідних потоків СМО. Знаходження інтенсивностей $\lambda_1,...,\lambda_N$ здійснюється на основі рівнянь балансу мережі з урахуванням простих властивостей злиття і розгалуження потоків. При злитті n потоків заявок з інтенсивностями $\lambda_1,...,\lambda_n$ утворюється потік, що має інтенсивність $\lambda = \lambda_1 +,...,+\lambda_n$. При розгалуженні потоку з інтенсивністю λ на n напрямків, імовірності переходу заявки в які дорівнюють $p_1,...,p_n$, утворюється n потоків з інтенсивностями $\lambda p_1,...,\lambda p_n$ відповідно. У стаціонарній ММО середнє число заявок у будь-якій її фіксованій частини є сталим. Звідси випливає, що сумарна інтенсивність потоків, які входять у цю частину дорівнює сумарній інтенсивності потоків, що виходять з неї. Запис цього закону в математичній формі називається рівнянням балансу. Виділяючи різні частини в ММО і складаючи для них рівняння балансу, можна отримати систему рівнянь, що зв'язує невідомі інтенсивності $\lambda_1,...,\lambda_N$ з відомими $I_1,...,I_N$. Зазвичай при цьому як окремі частини ММО виділяють усі СМО. У цьому випадку для N невідомих є N рівнянь. Можна додати до них рівняння балансу для вхідних і вихідних потоків усієї ММО. Тоді отримаємо $N+1$ рівняння, і одне з них можна використовувати як перевірне.

Наприклад, баланс інтенсивностей у мережі для рис. 1.23 можна врахувати, позначаючи інтенсивності на входах і виходах СМО і ММО так,

як показано на рис.1.24. Застосовуючи властивості злиття і розгалуження потоків, запишемо:

$$\begin{aligned} \lambda_1 &= I_1 + \lambda_2 + \lambda_3; \\ I_1 &= P_{10}\lambda_1; \\ \lambda_2 &= P_{12}\lambda_1; \\ \lambda_3 &= P_{13}\lambda_1. \end{aligned} \qquad (1.30)$$

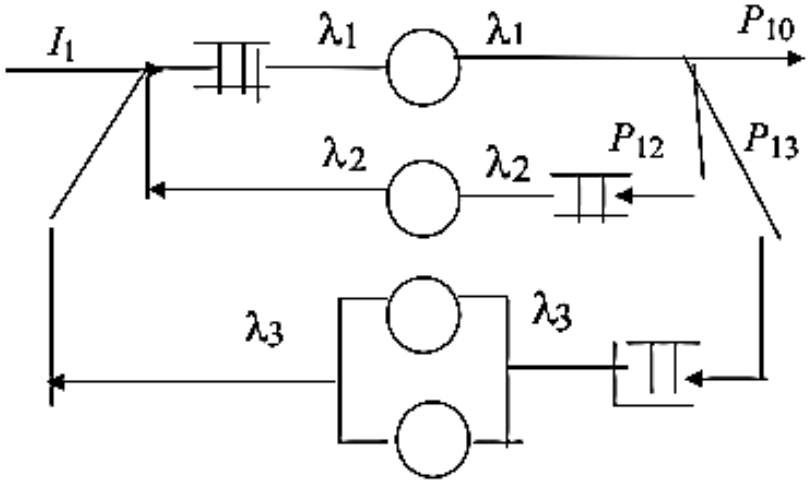


Рис. 1.24. Баланс інтенсивностей

При відомих $I_1 = 1$, $p_1 = 0,1$; $p_2 = 0,5$; $p_3 = 0,4$ з останніх трьох рівнянь знаходимо $\lambda_1 = 10$, $\lambda_2 = 5$, $\lambda_3 = 4$. Використовуючи перше рівняння в (1.30) для перевірки, підставляємо в нього знайдені значення інтенсивностей і отримуємо тотожність 10=1+5+4, що підтверджує правильність зроблених обчислень.

Перевірка стаціонарності ММО

ММО є стаціонарним, якщо стаціонарними є всі СМО, тобто якщо

$$\rho_j \le 1, \ \overline{j = 1, N}. \qquad (1.31)$$

Перевірити ці умови після того, як визначено λ_j, дуже легко. Наприклад, для ММО (рис. 1.24) умова (1.31) виконується, оскільки

$$\rho_1 = \lambda_1 \overline{T}_{обс1} = 10 \cdot 0,07 = 0,7, \ \rho_2 = \lambda_2 \overline{T}_{обс2} = 5 \cdot 0,06 = 0,3,$$

$$\rho_3 = \frac{\lambda_3 \overline{T}_{обс3}}{2} = \frac{4 \cdot 0,35}{2} = 0,7.$$

Для стаціонарної експоненціальної ММО з відомими інтенсивностями λ_j розрахунок локальних характеристик зводиться до застосування формул (1.18) – (1.23). Наприклад, для ММО (рис. 1.24) знаходимо, що $\rho_1 = 0,7$; $L_1 = 1,63$; $M_1 = 2,33$; $\overline{T}_{оч1} = 0,163$; $\overline{T}_{пер1} = 0,233$; $\rho_2 = 0,3$; $L_2 = 0,13$;

$M_2 = 0,43$; $\overline{T}_{оч2} = 0,026$; $\overline{T}_{пер2} = 0,086$; $\rho_1 = 0,7$; $\beta_0 = 0,176$; $L_3 = 0,402$; $M_3 = 1,802$; $\overline{T}_{ож3} = 0,1$; $\overline{T}_{пр3} = 0,45$.

1.5.6. Розрахунок системних характеристик експоненціальних ММО

Характеристики ММО зазвичай визначаються на рівні середніх значень і поділяються на локальні й системні. До локальних характеристик ММО належать всі характеристики, що входять до неї. Системні характеристики відображають властивості мережі в цілому, що розглядається як єдина, неподільна на частини система. Найбільш важливими системними характеристиками ММО є:

1. Середній час $\overline{T}_{пер}$ перебування в мережі. Часом перебування в мережі називається час між приходом заявки в мережу і її видходом з мережі.

2. Передавальні коефіцієнти $\alpha_{ij}, i, j = \overline{1, N}$. Нехай заявка входить у мережу з і-го вхідного потоку. Її маршрут у мережі є випадковим, тому випадковими є і число парафій в j-ї ММО за час перебування в мережі. Середнє значення α_{ij} цього числа парафій будемо називати передавальним коефіцієнтом. Він однозначно визначається для будь-яких і, j матрицею Р імовірностей передач.

3. Вхідний середній час $F_1, ..., F_N$ перебування в мережі. Величина F_j визначається як середній час перебування в мережі заявки, що надходить з j-го вхідного потоку $(j = \overline{1, N})$.

4. Умовні пропускні здатності $B_1, ..., B_N$. Припустимо, що в заданій ММО значення інтенсивності I_j замінено на максимальне значення, при якому мережа є стаціонарною. Це значення B_j будемо називати умовною пропускною здатністю по j-му входу. При заданих I_k $(k \neq j)$ мережа є стаціонарною для будь-яких значень $I_j \leq B_j$.

5. Абсолютні пропускні здатності A_j. Припустимо, що в заданій ММО інтенсивності всіх вхідних потоків, крім j-го, замінено на нульові, а I_j змінено на граничне значення, при якому мережа є стаціонарною. Це значення A_j будемо називати абсолютною пропускною здатність по j-му входу. Якщо $I_j \geq A_j$, то мережа є нестаціонарною, якими б не були інтенсивності інших вхідних потоків.

6. Запаси $D_1, ..., D_N$ за пропускними здатностями. Запас $D_j = B_j - I_j$,

$j = \overline{1, N}$, показує, наскільки можна збільшити інтенсивність приходу заявок на j-му вході (при заданих інших) без порушення умови стаціонарності. Якщо у вигляді ММО моделюється певна реальна система, то характеристики 1 – 6 можуть дати цінну інформацію про властивості цієї реальної системи. Наприклад, якщо ММО зображує обчислювальну систему реального часу, то середній час перебування E характеризує середній час відповіді системи, а запаси D_j показують готовність системи продовжувати стійке функціонування при збільшенні навантаження (інтенсивності запитів) з того чи іншого входу.

1.5.7. Завдання

За допомогою класів бібліотеки Enterprise Library [18, 19] побудуйте ММО, зображену на рис. 1.24. Порівняйте параметри аналітичної та імітаційної моделей.

Для роботи потрібні такі класи бібліотеки Enterprise Library.

Source – джерело заявок. Зазвичай використовується як початкова точка потоку заявок або як генератор ресурсів, транспортерів, і т. ін.

Sink – знищує заявки, що надійшли. Зазвичай використовується як кінцева точка потоку заявок. Об'єкт Sink автоматично підраховує вхідні заявки і вираховує середню інтенсивність вхідного потоку.

Combine чекає надходження двох заявок у порти $input_1$ і $input_2$ (у довільному порядку), а потім створює нову заявку і направляє її на вихідний порт. Клас нової заявки задається користувачем. Заявка, що надходить першою, зберігається всередині об'єкта, поки не прийде інша. Як тільки надходить інша заявка, створена заявка відразу ж залишає об'єкт.

SelectOutput приймає заявку, а потім залежно від заданої умови передає її на один з двох вихідних портів. Умова може залежати від самої заявки або від якоїсь іншої інформації. Заявка, яка надійшла, залишає об'єкт у той самий момент часу.

Queue моделює чергу і зберігає заявки, які надходять в певному порядку: FIFO (заявки поміщаються в чергу в порядку надходження), LIFO (заявки поміщаються в чергу в порядку, зворотному їх надходженню), RANDOM (заявки поміщаються в будь-які місця черги) або PRIORITY (заявки поміщаються в чергу відповідно до значення своїх полів priority).

Delay затримує заявки на заданий час. Одночасно може бути затримано відразу кілька заявок (не більше заданої місткості об'єкта capacity). На відміну від об'єкта Server заявки затримуються незалежно одна від одної. Час затримки обчислюється окремо для кожної заявки. Як тільки час затримки закінчується, заявка відразу залишає об'єкт.

1.6. Мережі Петрі

Мережі Петрі використовуються для моделювання асинхронних систем, що функціонують як сукупність паралельних взаємодіючих процесів. Аналіз мереж Петрі дає змогу отримати інформацію про структуру й динамічну поведінку модельованої системи.

Причиново-наслідковий зв'язок подій в асинхронних системах задається множиною відносин виду "умови – події".

Побудова моделей систем у вигляді мереж Петрі полягає в такому:

1. Процеси, що моделюються, описуються множиною подій (дій) і умові визначають можливість настання цих подій, а також причиново-наслідкових відносин, що встановлюються на множині пар "події – умови".

2. Визначаються події-дії, послідовність виконання яких керується станами системи. Стан системи задається множиною умов, які формуються у вигляді предикатів. Кількісно умови характеризуються величиною, яка виражається числами натурального ряду.

3. Умови залежно від значень їх кількісних характеристик можуть виконуватися або ні. Виконання умов забезпечує можливість реалізації подій. Умови, з фактом виконання яких пов'язується можливість реалізації подій, називаються передумовами. Реалізація події забезпечує можливість виконання інших умов, які перебувають з передумовою в причиново-наслідковому зв'язку. Ці умови називаються постумовами.

У мережах Петрі умови – це позиції, а події – переходи. Відповідно до цього граф мережі Петрі є дводольним орієнтованим мультиграфом. Зображення позиції і переходу на графі показано на рис. 1.25.

Рис. 1.25. Граф: а − зображення позицій; б − зображення переходів

Орієнтовані дуги можуть з'єднувати тільки позиції і переходи в прямому і
зворотному напрямах (властивість дводольних). Мережа Петрі є мультиграфом, оскільки допускається кратність дуг між позиціями і переходами (вершинами графа). Приклад графа мережі Петрі наведено на рис.1.26.

У графах мережі Петрі кількісні характеристики умов (числа натурального ряду) прийнято зображувати кількістю міток у відповідних позиціях.

Послідовності подій відображаються спрацюваннями переходів. Виконання будь-якої умови пов'язано з появою однієї або декількох міток у позиції, що відповідає цій умові. Угоди про правила спрацювання переходів є способом вираження причиново-наслідкових зв'язків між умовами і подіями в системі.

При моделюванні гнучких виробничих систем позиції відображають окремі операції виробничого процесу (наприклад, транспортування заготовки до конвеєра, пересування заготовки до верстата конвеєром, оброблення деталі) або стан компонентів гнучкої виробничої системи (наприклад, робота конвеєра, верстата). Наявність мітки в одній з позицій відповідає стану виконання деякої з технологічних операцій або стану, в якому перебувають деякі з компонентів гнучкої виробничої системи.

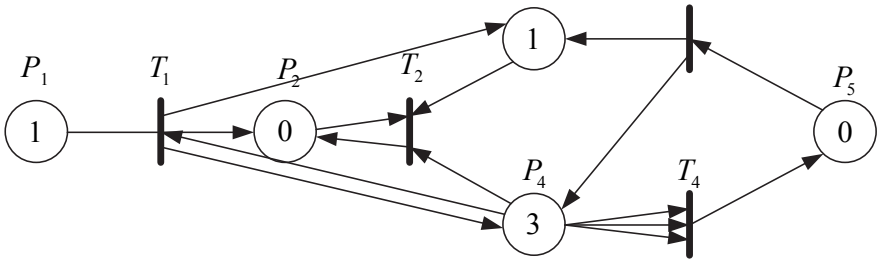


Рис. 1.26. Приклад графа мережі Петрі

Переходи відповідають подіям, що відображають початок або завершення модельованих операцій. Наприклад, перехід інтерпретується як подія, пов'язана із завершенням операції транспортування заготовки роботом і її установлення на конвеєрі, а також з початком операції переміщення заготовки конвеєром до верстата.

Присвоєння міток позиціям мережі Петрі називають маркуванням мережі. Динаміка мереж Петрі пов'язана з механізмом зміни маркувань позицій і угодами про правила спрацювання переходів. Перехід спрацьовує, якщо в кожній вхідній позиції (передумові) кількість міток не є меншою за кількість дуг, що виходять з позиції в цей перехід. Такі переходи називають збудженими, їх спрацювання може відбутися через будь-який скінченний проміжок часу після збудження. Унаслідок спрацювання з усіх вхідних позицій переходу виключається кількість міток, яка дорівнює кількості дуг, що виходять з відповідної позиції в перехід, а до вихідних позицій цього переходу додається кількість міток, яка дорівнює кількості дуг, що виходять з переходу в відповідну вихідну позицію. На рис. 1.27, а показано збуджений перехід, а на рис.1.27, б – незбуджений.


а б

Рис. 1.27. Перехід: а − збуджений; б – незбуджений

Маркування на рис. 1.27, б отримано внаслідок спрацювання збудженого переходу на рис. 1.27, а. Зауважимо, що спрацювання переходу вважається неподільним актом, тобто зміна маркувань всіх пов'язаних з цим переходом вхідних і вихідних позицій здійснюється миттєво.

1.6.1. Механізми синхронізації процесів

Для більш глибокого розуміння суті використання апарату мереж Петрі при описі спільної роботи сукупності паралельних взаємодіючих процесів розглянемо основні типи взаємодії і синхронізації процесів. Взаємодія процесів потребує розподілу ресурсів між ними. Щоб система працювала правильно, розподілом ресурсів необхідно керувати. Це керування в основному зводиться до задач синхронізації взаємодії процесів. При цьому деякі задачі синхронізації стали класичними:

- задача про взаємне виключення;
- задача про виробника / споживача;
- задача про мудреців, що обідають;
- задача про читання / запис;
- p - і V-операції над семафором.

Інші механізми синхронізації процесів містять розв'язання перелічених задач. Знання цих задач і набуття навичок їх застосування дає змогу успішно справлятися з досить непростою справою опису функціонування систем мереж Петрі.

Задача про взаємне виключення. Ця задача виникає в умовах, коли кілька процесів поділяють загальну змінну. Кожен процес може спочатку вважатися значенням змінної (рис. 1.28), а потім обчислити нове значення і записати його на те саме місце. Якщо два процеси одночасно будуть виконувати зазначену послідовність дій, то це може спричинити неправильну роботу системи. Наприклад, обидва процеси враховують по черзі попереднє значення змінної, а

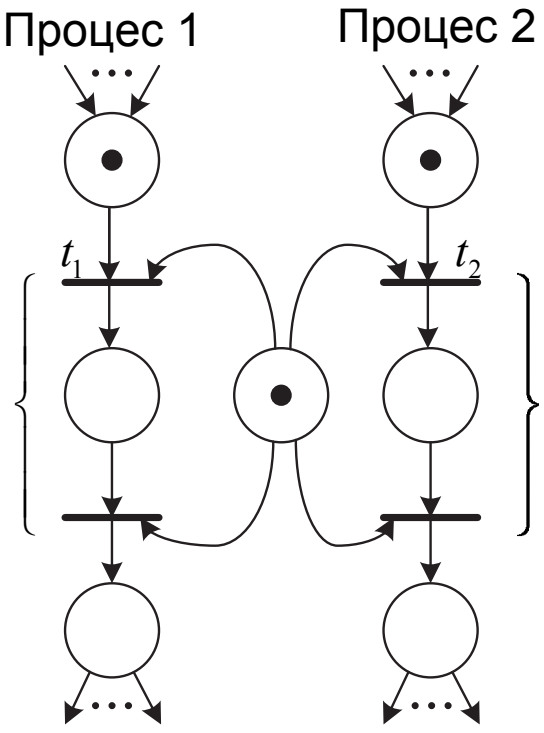


Рис. 1.28. Механізм взаємного виключення

при запису перше з нових значень буде знищено. Таким чином, результат роботи одного з двох процесів буде втрачено.

Для коректної роботи системи необхідно забезпечити механізм взаємного виключення, при якому тільки один процес має доступ до розділеної змінної. Частина програмного коду процесу, який виконує доступ до розділеної змінної і при цьому надає захист від доступу до розділеної змінної іншого процесу, називається критичною секцією. Механізм взаємного виключення працює таким чином, що при виконанні критичної секції будь-якого процесу критичні секції інших процесів блокуються. Взаємодія двох процесів через механізм взаємного виключення критичних секцій подано у вигляді мережі Петрі на рис. 1.28. Переходи t_1 і t_2 перебувають в конфлікті за мітку в позиції m. Запуск, наприклад, переходу t_2 змушує перший процес чекати доти, доки другий процес вийде зі своєї критичної секції і поверне мітку в позицію m .

Задача про виробника / споживача. У цій задачі відокремлювальним об'єктом є буфер. Процес-виробник породжує компоненту інформації і розміщує її в буфер. Процес-споживач чекає доти, доки компонента інформації розміститься в буфері, після цього він може її видалити з буфера і використовувати. Зазвичай застосовується буфер обмеженої місткості, тобто є можливість розмістити в ньому не більше n компонентів даних. У цьому випадку швидкість заповнення буфера процесами-виробниками і швидкість отримання даних процесами-споживачами мають бути порівнянними. На рис.1.29 показано мережу Петрі, яка відображає дію процесів.

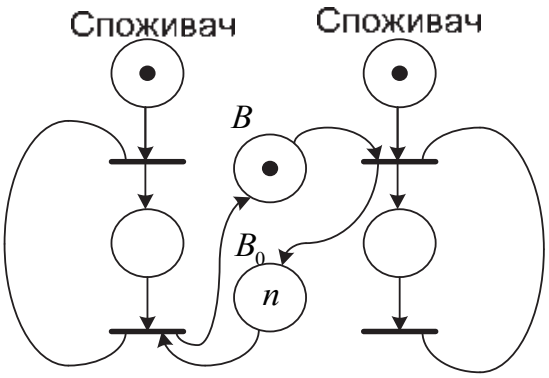


Рис.1.29. Механізм обміну даними через буфер

Буферу зіставлено дві позиції: B – кількість компонентів даних, розміщених у буфері, але ще не використаних; B_0 – кількість вільних місць у буфері для розміщення компонентів даних. Спочатку позиція B має n міток, а позиція B_0 не має. Якщо буфер буде повністю заповнений, то в позиції B буде n міток, а в B_0 – 0 міток. Таким чином, доступ процесу-виробника в заповнений буфер є неможливим, оскільки в позиції B буде 0 міток.

Задача про мудреців, що обідають*.* Мудреці сидять за круглим столом, кожен з них може перебувати в одному з двох станів: думає або обідає. На столі стоять страви китайської кухні, а між мудрецями лежить по одній паличці. Для вживання їжі мудрець повинен взяти паличку зліва і

паличку справа. У цьому випадку сусіди мудреця, які обідають, можуть тільки думати і чекати, коли звільняться палички, щоб розпочати обідати. На рис.1.30 у вигляді мережі Петрі зображено взаємодію трьох процесів (трьох мудреців), стани яких відображено позиціями O_i і D_i: O_i − обідає, D_i − думає. Позиціями P_i є палички для їжі (колективні ресурси). У вихідному маркуванні всі мудреці думають, а всі палички є вільними. При такому вихідному маркуванні будь-який з трьох мудреців може розпочати обідати, захопивши палички зліва і справа.

Задача про читання і запис.

Розглядається взаємодія процесів двох типів: читання й запису. Всі процеси спільно використовують загальний файл або змінну чи елемент даних. Процеси читання об'єкта не змінюють, а процеси запису – змінюють. Тому процеси запису повинні взаємно виключати (рис. 1.31) всі інші процеси читання і запису, тоді як кілька процесів читання можуть мати доступ до даних, що розділяються одночасно. Взаємодію процесів необхідно організовувати так, щоб не могла виникнути безвихідна ситуація і було забезпечено механізм взаємного виключення з боку процесів запису.

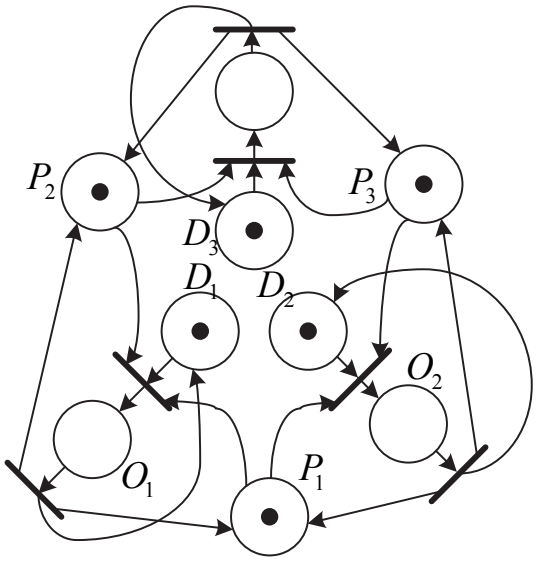


Рис.1.30. Механізм розділу ресурсів

На рис.1.31 зображено мережу Петрі, що імітує взаємодію процесів читання і запису за умови, що кількість процесів читання і запису обмежено величиною n. Це означає, що і при необмеженій кількості процесів читання одночасно можуть виконуватися не більше n процесів. Початкове маркування мережі передбачає наявність S процесів читання й t процесів запису. З цієї мережі видно, що процеси перебувають в конфлікті за мітку в позиції m. При захопленні міток процесом запису позиція m залишається без міток, тим самим блокується запуск процесів читання доти, доки

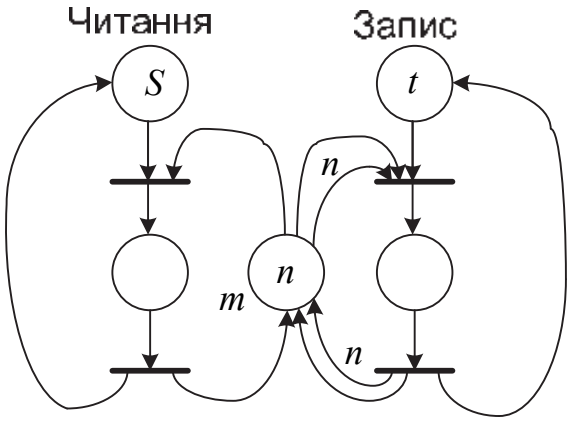


Рис.1.31. Механізм взаємодії процесів читання, запису

процес запису не поверне n міток у позицію m.

Слід звернути увагу на те, що в цій мережі позиція m після завершення процесів читання завжди буде мати n міток, що дає змогу запускати процес запису.

У тих випадках, коли кількість процесів, що одночасно читають, є необмеженою, моделювання взаємодії процесів читання і запису за допомогою мережі Петрі виявляється неможливим. Проблема полягає у відсутності механізму блокування необмеженої кількості процесів, що одночасно читають, при виконанні процесів запису. Для зберігання кількості тих процесів, що читають, можна ввести спеціально позицію «лічильник». При запуску кожного процесу читання в лічильник додається одиниця, а після його закінчення – віднімається. Однак це не вирішує проблеми, оскільки для запуску процесу запису необхідно, щоб у лічильнику не було міток. Механізму перевірки необмеженої позиції на нуль у мережах Петрі не існує. Це свідчить про те, що можливості мереж Петрі для відображення взаємодії процесів є небезмежними.

P- і V-операції над семафором. Одним з поширених механізмів синхронізації процесів є P- і V-операції над семафором. Семафор – це змінна, яка може мати тільки невід'ємні цілі значення. V-операція збільшує значення на одиницю, а P-операція зменшує його на одиницю. P-операція може виконуватися тільки в тому випадку, коли отримане значення семафора залишається невід'ємним. Таким чином, якщо значення семафора дорівнює 0, то P-операція повинна чекати, поки якийсь інший процес не виконає V-операція. P- і V-операції визначено як примітивні, тобто ніяка інша операція не може змінювати значення семафора одночасно з ними. На рис.1.32 показано, як такі операції моделюються мережею Петрі. Кожен семафор моделюється позицією, кількість міток r в позиції показує значення семафора. P-операції використовують позицію семафора як вхід, а V-операції – як вихід.

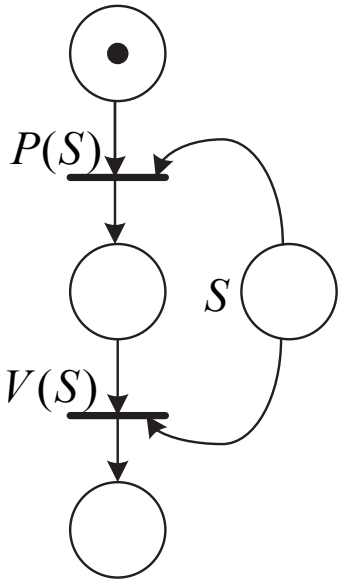


Рис.1.32. Механізм семафора

1.6.2. Формалізований опис мереж Петрі

Для опису й математичного аналізу процесів з точками розгалуження і синхронізації взаємодії розроблено апарат мереж Петрі. Структура мережі являє собою орієнтований дводольний граф. Множина вершин графа

розбивається на дві підмножини T і P, $V = T \cup P$, $T \cap P \neq 0$. Дугами можуть зв'язуватися вершини з множин P і T. Динаміка розвитку процесів відбивається у вершинах P мітками (марками). Розподіл міток по вершинах називають маркуванням мережі. Кожне маркування відповідає певному стану мережі.

Мережа Петрі визначається п'ятіркою множин:

$$N = \{P, T, J, O, M_0\},$$

де $P = \{p_i\}$, $i = 1,2,...,n$ – множина позицій; $T = \{t_j\}$, $j = 1,2,...,m$ – множина переходів; $J : T * P \to \{0,1\}$ – функція проходження; $O : P * T \to \{0,1\}$ – функція передування; $M_0 : P \to Z_0$ – початкове маркування (стан) мережі; Z_0 – множина додатних цілих чисел.

Функції J і O задають множину дуг $\{t_j, p_i\}$ і $\{p_i, t_j\}$ відповідно.

Дуги, що передують позиції P_i, позначимо множиною $J(p_i) = \{(t_j, p_i) \mid J(t_j, p_i)\} = 1$, а дуги, що передують переходу t_j, – множиною $J(t_i) = \{(p_i, t_j) \mid O(p_i, t_j)\} = 1$.

Тут запис $J(t_j p_i) = 1$ означає наявність дуги $(t_j p_i)$, а запис $O(p_i, t_j) = 1$ – дуги (p_i, t_j). Аналогічно дуги, які випливають з P_i і t_j, подамо множинами $O(p_i) = \{(p_i, t_j) \mid O(p_i, t_j)\} = 1$ і $O(t_j) = \{(t_j, p_i) \mid J(t_j, p_i)\} = 1$.

Вхідні позиції переходу t_j об'єднуються в множині його попередників $\Pr e(t_j) = \{p_i \in P \mid O(p_i, t_j) = 1\}$, а вихідні позиції – у множині позицій-послідовників $Post(t_j) = \{p_i \in P \mid J(t_j, p_i) = 1\}$.

Маркувати мережі рекомендується вектором $M = \{m(p_i)\}$, де $m(p)$ – кількість міток у позиції p_i. Перехід t_j є збудженим при маркуванні M і може спрацювати, якщо виконується умова $\forall p_i \in \Pr e(y_j)[m(p_i) - O(p_i, t_j) \geq 0]$, тобто кількість міток $m(p)$ більша або дорівнює кількості дуг (p_i, t_j), що відповідає $m(p_i) - O(p_i, t_j) \geq 0$.

Спрацювання переходу t_j призводить до того, що кожна позиція $p_i \in \Pr e(t_j)$ втрачає $O(p_i, t_j)$ міток, а кожна з позицій $Post(t_j)$ отримує $J(t_j, p_i)$ міток.

Якщо при маркуванні M збуджено кілька переходів, то порядок їх спрацювання є невизначеним, отже, може бути подано кілька послідовностей переходів, що спрацьовують.

1.6.3. Приклади побудови мережі Петрі

Приклад 1. Розглядається опис двох взаємодіючих процесів на основі механізму семафорів. Нехай у загальній пам'яті виділено ділянку довжиною одиниць, яка використовується для зберігання запитів на виконання завдань. Процеси запису запитів у чергу і вибору їх з черги можуть відбуватися незалежно (асинхронно). Описувач черги містить чотири елементи: n і l – кількість заповнених і вільних елементів черги; α і β – адреси початкового й кінцевого елементів. Алгоритми запису в чергу і вибору запитів з черги наведено на рис.1.33. Запис вигляду $[\beta]=R$ являє собою операцію розміщення кількості елементів з черги з номером β, а запис вигляду $R=[\alpha]$ – зчитування вмісту комірки α і використання його як інформації про запит.

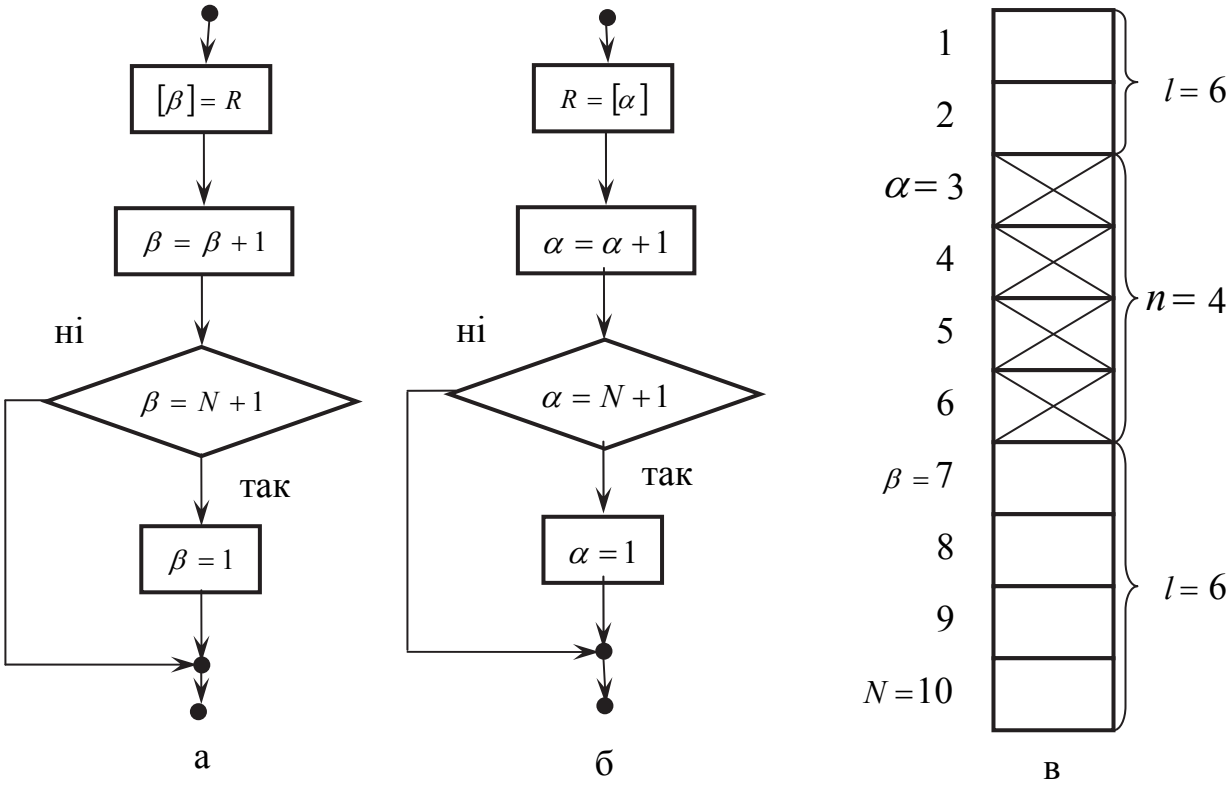


Рис. 1.33. Взаємодіючі процеси: а – алгоритм запису запиту в чергу; б – алгоритм вибору запиту з черги; в – приклад стану черги в буфері з $N=10$ елементів

З чергою можуть працювати кілька програм «у чергу» і «з черги». Для організації коректного використання черги необхідно виключити одночасний доступ програм до області збереження запитів. Інакше можуть бути помилки. Наприклад, два споживача вибирають один і той саме запит, не вибравши з черги наступного. Для виключення подібної ситуації паралель-

ні процеси «в чергу» і «з черги» можуть синхронізуватися з допомогою семафорних примітивів P і V. Мережу Петрі, яка описує ці паралельні процеси, наведено на рис. 1.34. Зміст позицій і переходів відображено у цих елементах мережі.

Позиція $S=\{0,1\}$ – семафор, що обмежує доступ процесу до черги. Операції над $S: V(s)$ – змінна S збільшується на одиницю, якщо $S=0$; $P(s)$ – змінна S зменшується на одиницю, якщо $S=1$, в іншому випадку процес блокується, чекаючи, коли стан семафора буде $S=1$. Примітив $P(s)$ забороняє доступ процесу до критичної області, а $V(s)$ – дозволяє.

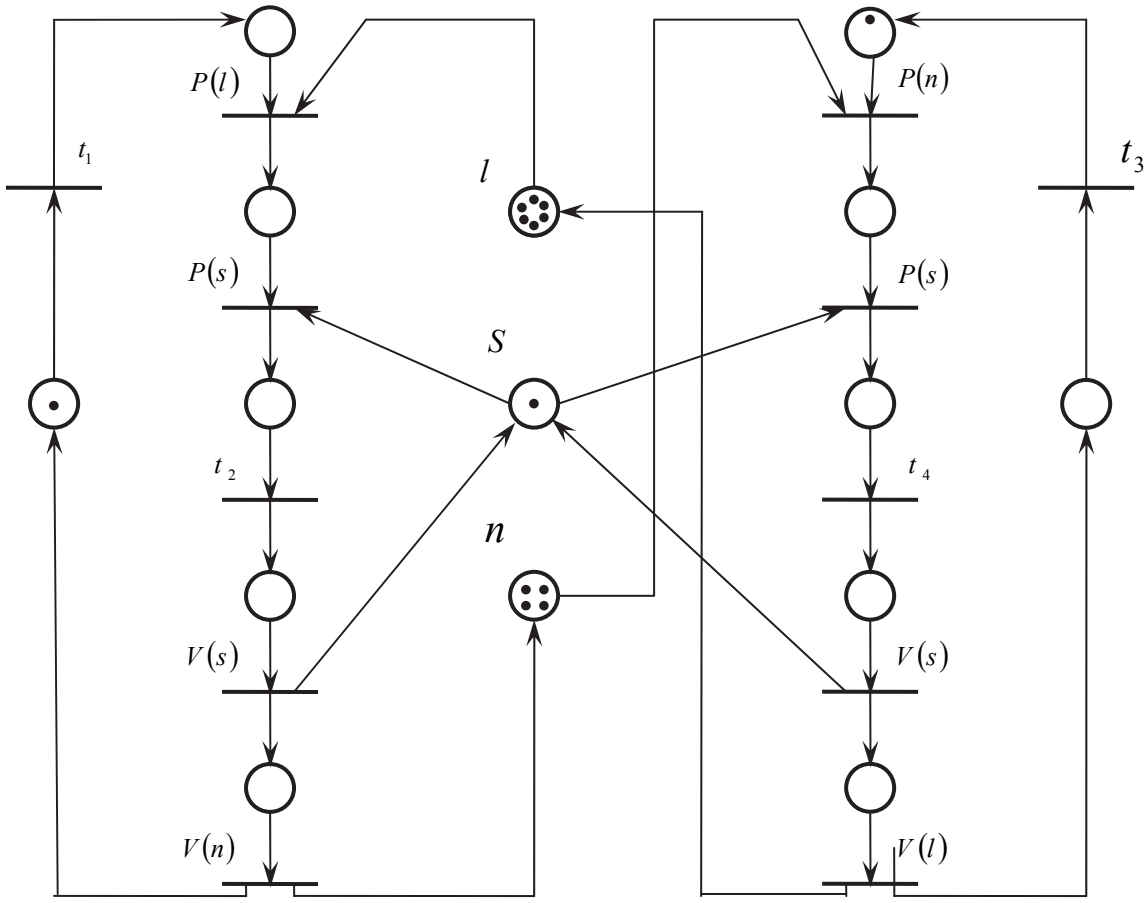


Рис. 1.34. Мережа Петрі, яка описує паралельні процеси керування:
t_1 – створення запиту; t_2 – запис запиту в чергу; t_3 – виконання запиту;
t_4 – вибір запиту із черги

Аналогічно визначаються операції над семафорами l і n, що прочитуються: $P(l)$ – якщо $l>0$, то l зменшується на одиницю і процес триває,

інакше він блокується в поточній точці; $V(l)$ – якщо $l < N$, то l збільшується на одиницю і процес триває.

Приклад 2. Розглядається обчислювальна система (ОС) (рис. 1.35), що містить процесор (П), пристрій зв'язку з об'єктом (ПЗО) і запам'ятовуючий пристрій (ЗП). ПЗО і П можуть працювати паралельно.

Процеси в цій ОС породжуються періодичними та ініціативними запитами від ОК і ініціативними запитами від оператора. Розглянемо чергу запитів оператора, які включають виконання на П двох операцій. Перша з них використовує поточні дані із ЗП. Друга операція виконується після першої і завершення роботи ПЗО з об'єктом керування (ОК) і ЗП. Також П взаємодіє з оперативним запам'ятовуючим пристроєм (ОЗП). Мережу Петрі, що описує цей процес, наведено на рис.1.36.

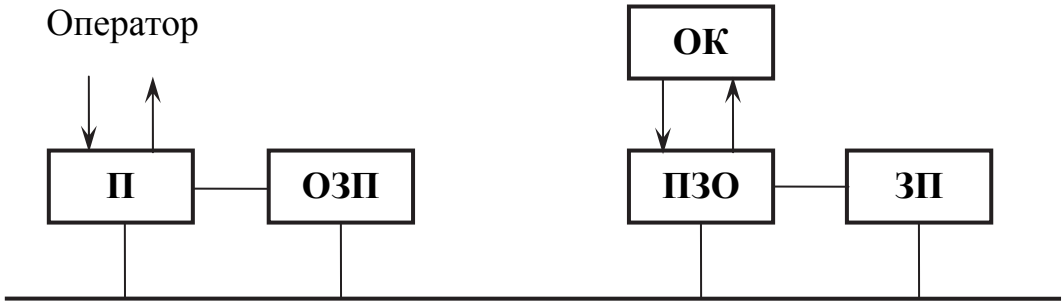


Рис. 1.35. Однопроцесорна ОС з ПЗО і ЗП

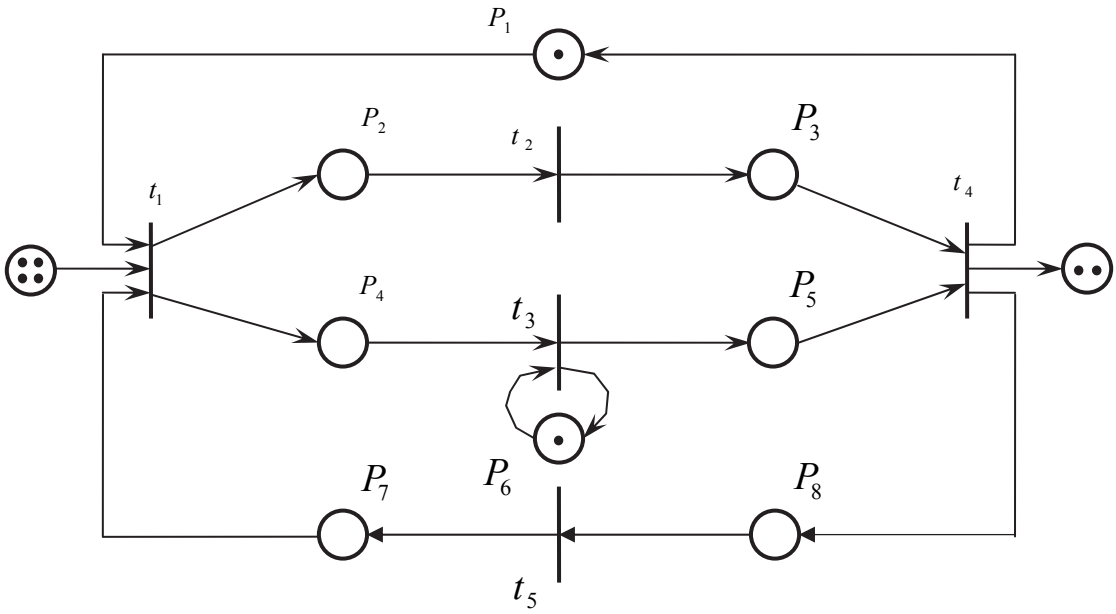


Рис. 1.36. Модель процесу у вигляді мережі Петрі

Переходи: t_1 – введення даних із ЗП в ОЗП; t_2 – операція 1; t_3 – робота ПЗО з ОК і ЗП; t_4 – операція 2; t_5 – підготовка ЗП до роботи (установлення головок).

Позиції: P_1 – П є вільним; P_2, P_4 – введення даних із ЗП в ОЗП виконано; P_3 – операцію 1 виконано; P_5 – роботу ПЗО закінчено; P_6 – ПЗО є вільнім; P_7 – ЗП готовий до роботи; P_8 – операцію 2 виконано.

Перехід t_1 спрацьовує, якщо є запит, вільний процесор і готовий до роботи ЗП. Можливість паралельної роботи процесора ПЗО відображено паралельними гілками з t_2 і t_3. Операція 2 (перехід t_4) виконується після завершення операції 1 і роботи ПЗО. Після спрацювання переходу t_4 звільняються П, ЗП і фіксується мітка про завершення процесу. Після підготовки ЗП (перехід t_5) може спрацювати наступний запит оператора.

У наведених прикладах мережі Петрі було описано:

- паралелізм і синхронізацію процесів;
- взаємодію процесів і ресурсів, поданих мітками;
- незалежні дії, подані переходами;
- накопичувачі інформації (елементи і буфери) і стани процесів, подані позиціями.

Однак у цьому вигляді мережі Петрі не дають змоги адекватно подавати багато властивостей систем. Так, одним з недоліків мереж Петрі є відсутність часу у визначенні їх динамічного функціонування. У більшості теоретичних досліджень мереж Петрі час також не розглядається. При врахуванні часу істотно ускладнюється аналіз мереж, тобто відбувається перехід від алгебраїчних рівнянь до диференціальних.

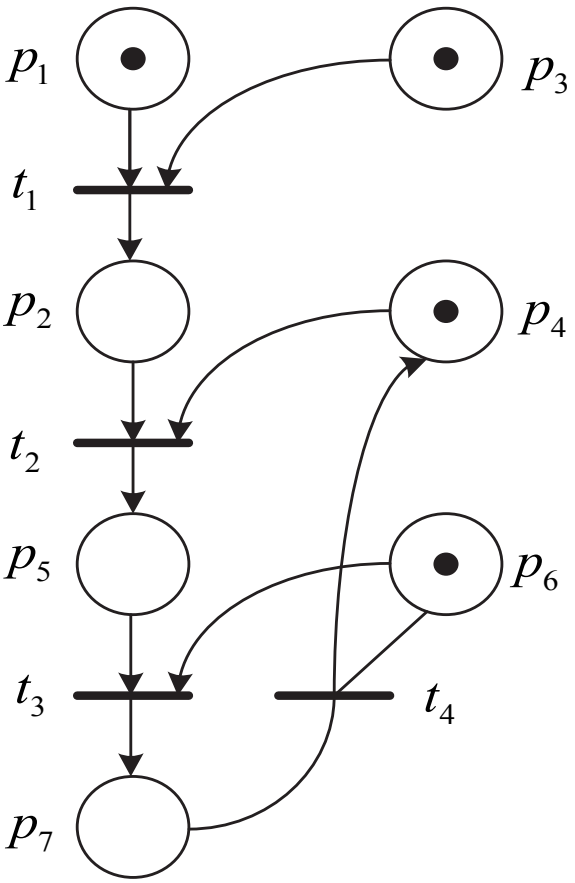


Рис. 1.37. Мережа Петрі для задання процесу розвантаження судна

1.6.4. Завдання

Побудувати приклад моделі в середовищі AnyLogic для моделювання мережі Петрі, яка описує процес розвантаження судна (рис. 1.37).

Розглянемо найпростішу модель

порту, що містить одне місце біля причалу і в якому обслуговуються лише судна внутрішнього каботажу, тобто судна, які не потребують митного огляду. Мережу Петрі, яка відповідає такій моделі, зображено на рис. 1.37.

Позиції p_1, p_2, p_3, p_5, p_7 відповідають можливим станам судна в порту, p_4 і p_6 – станам причалу і перевантажувального комплексу. Переходи t_1, t_2, t_3 і t_4 описують тимчасові події, такі, як візит і розвантаження судна, звільнення причалу. Початковим станом системи буде відсутність судів у межах порту і наявність судна на вході в порт. Початкове маркування – мітки в позиціях p_1, p_3, p_4, p_6. Завершенню одного конвеєрного процесу буде відповідати наявність міток у позиціях p_4, p_6, а всього конвеєрного процесу – наявність міток у позиціях p_3, p_4, p_6 і відсутність у всіх інших.

Модель в середовищі AnyLogic матиме вигляд, як показано на рис. 1.38.

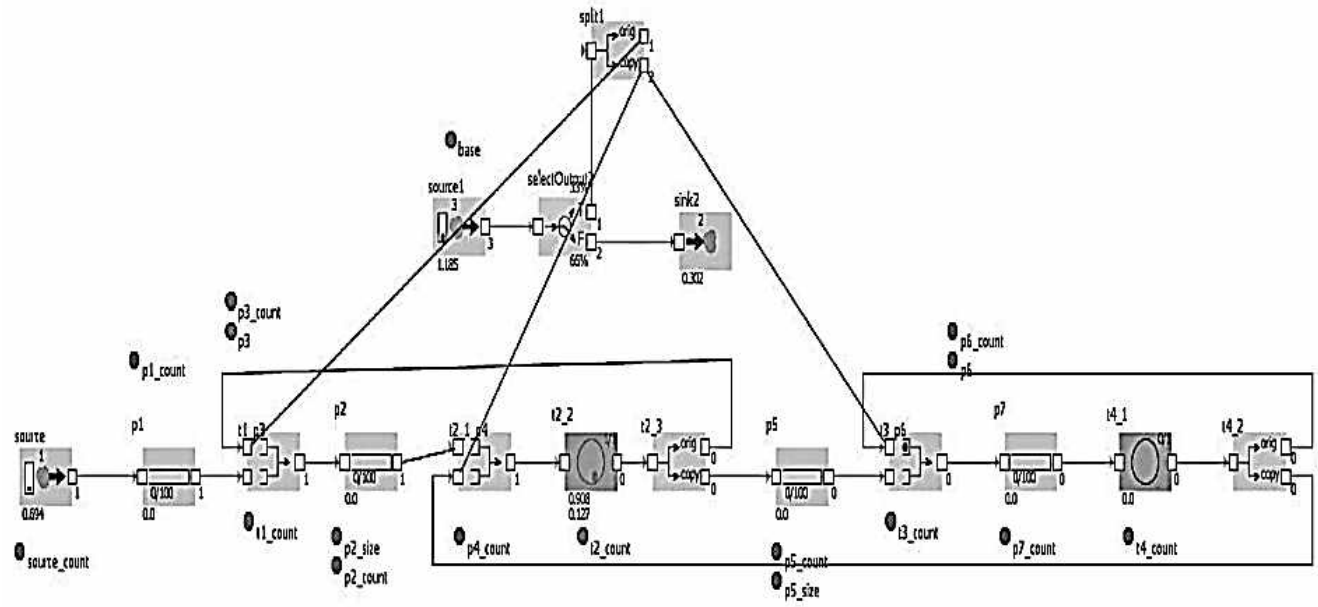


Рис. 1.38. Модель у середовищі AnyLogic, що імітує мережу Петрі

Елемент Source повинен імітувати появу мітки в початковому стані p_1. Стани мережі Петрі моделюються за допомогою черг Queue, довжина яких залежить від допустимої кількості міток в одному стані. Переходи мають бути реалізовані за допомогою елементів об'єднання і копіювання заявок для реалізації переходів з декількома вхідними і декількома вихідними дугами відповідно, а також елементів, що забезпечують затримку при переході на заданий час. Блок, що містить елемент Source1, буде забезпечувати початкове розставлення міток у мережі.

Модель і її подальший розвиток детально описано в роботі [20].

2. ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

2.1. Загальні визначення імітаційного моделювання

У першому розділі було проаналізовано існуючі математичні моделі й розглянуто приклади проведення кількох експериментів з ними в середовищі AnyLogic. Експериментування з моделлю називають імітацією (імітація – це розуміння суті явищ, не застосовується при експериментах на реальному об'єкті). Таким чином, імітаційне моделювання – це частковий випадок математичного моделювання. Існує клас об`єктів, для яких з різних причин не розроблено аналітичні моделі або методи розв'язання отриманої моделі. У цьому випадку аналітична модель замінюється імітатором або імітаційною моделлю.

Наприклад, при розгляді типових математичних схем було вказано агрегативну систему (A-схему). Ця система передбачає об'єднання інших схем (D-, F-, P- і Q-схем) в одній моделі, що аналітично є неможливим.

Другим прикладом є агентне моделювання – відносно новий (1990 – 2000-ті рр.) напрям у імітаційному моделюванні, що використовується для дослідження децентралізованих систем, динаміка функціонування яких визначається не глобальними правилами й законами (як в інших парадигмах моделювання), а навпаки, коли ці глобальні правила й закони є результатом індивідуальної активності членів групи. Мета агентних моделей – отримати уявлення про ці глобальні правила, загальну поведінку системи, виходячи з припущень про індивідуальну, приватну поведінку окремих активних об'єктів і взаємодію цих об'єктів у системі. Агент – якась суть, що є активною, автономно поводиться, може приймати рішення відповідно до деяких правил набору, взаємодіяти з оточенням, а також самостійно змінюватися.

У цьому розділі описано існуючі системи моделювання, сформовано план проведення експериментів імітаційного моделювання й дано пояснення щодо змісту звіту з результатами експерименту. Цей звіт є частиною розрахунково-графічної роботи студента, варіанти якої наведено в третьому розділі.

2.2. Програмне забезпечення імітаційного моделювання

2.2.1. ARIS

ARIS – це методологія, або, як називають автори, концепція [22], на якій базується сім'я програмних продуктів, розроблених компанією IDS Scheer AG (Німеччина) для структурованого опису, аналізу, подальшого вдосконалення бізнес-процесів підприємства й керування ними, а також для підготовки до впровадження складних інформаційних систем.

ARIS Toolset має сильний графічний інтерфейс, а також велику кількість стандартних об'єктів для опису бізнес-процесів. Загалом, методологія ARIS дає змогу вирішити широкий комплекс завдань з організаційного проектування, розроблення й супроводу технічного проекту.

ARIS підтримує чотири типи моделей, що відображають різні аспекти досліджуваної системи:

- організаційні моделі, що являють собою структуру системи;
- ієрархію організаційних підрозділів, посад, різноманіття зв'язків між ними, а також територіальну прив'язку структурних підрозділів;
- функціональні моделі, які містять ієрархію цілей, що стоять перед апаратом управління, із сукупністю дерев функцій, необхідних для досягнення поставлених цілей;
- інформаційні моделі, що відображають структуру інформації, необхідної для реалізації всієї сукупності функцій системи;
- моделі управління, що являють собою комплексний погляд на реалізацію ділових процесів у межах системи.

Сім'ю програмних продуктів наведено в табл. 2.1.

Таблиця 2.1

Програмний модуль	Призначення
ARIS Toolset	Виконання проектів з реінжинірингу і вдосконалення бізнес-процесів
ARIS Easy Design	Навчання розробленню й реінжинірингу бізнес-процесів
ARIS Web Designer	Розроблення бізнес-процесів з використанням Інтернету
ARIS Server	Мережний інсталяції програмних продуктів ARIS
ARIS BSC	Моделювання стратегічного управління компанією
ARIS ABC (Activity Based Cost)	Функціонально-вартісний аналіз бізнес-процесів
ARIS Web Publisher	Публікація інформації про бізнес-процеси в Інтернеті
ARIS Process Performance Manager	Вимірювання, аналіз і оптимізації бізнес-процесів
ARIS Quality Management Scout	Виконання проектів зі створення системи управління якістю
ARIS Process Risk Scout	Оцінювання ризиків виконання бізнес-процесів

2.2.2. ITHINK

ITHINK являє собою інструмент управління й планування, є засобом експертного аналізу ситуації, розробленим компанією High Performance Systems (США) [23] .У ITHINK реалізовано ідеї структурного проектування Е. Йордана, структурного аналізу Т. Демарко, системного аналізу С. Гейне і Т. Сарсона.

За допомогою пакета ITHINK можна змоделювати весь виробничо-збутовий цикл підприємства, починаючи від закупівлі сировини й закінчуючи виробництвом і реалізацією продукта. До сфери потенційного застосування належать також транспорт (у тому числі трубопровідний), мережі газо- і водопостачання й інші розподільні системи.

Система ITHINK дає змогу вирішувати такі завдання системного аналізу:

- розроблення моделей різноманітних систем;
- побудова моделей систем середньої складності як сукупності простих;
- визначення механізму взаємодії компонентів системи з метою поведінкового опису в термінах її основних характеристик;
- використання й розроблення формальних ефективних процедур імітації відповідних поведінкових описів для отримання статистичних оцінок і визначення якісних характеристик;
- формування процедур генерації представницьких тестових наборів, необхідних для дослідження нетривіальної поведінки;
- цілеспрямоване вивчення поведінки моделей для складання попередніх розкладів роботи з компонентами системи.

У ITHINK існує чотири основні стандартні блоки: потік, конвеєр, накопичувач, розподільник.

2.2.3. Powersim Studio

Пакет PowerSim Studio було створено фірмою Powersim Software AS (Норвегія) [24]. Використовувану методологію побудовано на базі класичних методів системної динаміки, створених Дж. Форрестером. Пакет є додатком таких відомих систем, як SAP, SEM, BPS, але також може використовуватися як автономний додаток. Пакет має розвинені засоби візуального програмування й різні розширені можливості, у тому числі вбудовані блоки аналізу ризиків та оптимізації бізнес-процесів. PowerSim дає змогу моделювати як дискретні, так і неперервні процеси.

До розширених можливостей системи можна віднести: ієрархічні способи розроблення програм (аналог використання підпрограм мовами високого рівня), можливість написання алгоритмів вбудованою мовою Visual Basic, комбінація аналізу ризиків та оптимізації, що дає змогу знайти найкраще (оптимальне) рішення, якщо параметри змінюються за випадковим законом.

Пакет PowerSim Studio має розвинені засоби використання зовнішніх даних з інформаційного середовища підприємства. Він має вбудовані механізми роботи зі звичайними текстовими файлами, файлами Excel і сховищами даних SAP BW. При використанні розширених можливостей є можливою інтеграція пакета в будь-яке інформаційне середовище. Пакет

PowerSim Studio також має потужний вбудований оптимізатор PowerSim Solver 2.5.

Системи, побудовані на базі моделей у середовищі PowerSim Studio, можна використовувати для такого:

- створення єдиних виробничо-економічних моделей підприємства;
- розрахунок різних виробничих і економічних показників;
- сценарні розрахунки і визначення впливу керуючих впливів і аналіз їх ефективності;
- аналіз ризиків;
- оптимізація діяльності підприємства;
- формування оптимальної стратегії на різних рівнях планування.

2.2.4. Extend

Extend – універсальний пакет імітаційного моделювання процесів модернізації й обслуговування, розроблений компанією Imagine That, Inc. (США) [25].

Пакет Extend має потужний графічний інтерфейс, який дає змогу створювати схеми процесів і виробляти імітаційні експерименти. Є можливість перегляду моделей у вигляді графіків, а також з використанням 2D- і 3D-анімації.

Деякі можливості 3D-аніматора:

- створення віртуального середовища, що являє собою необхідний процес переходу 3D-зображення;
- вивчення об’єкта, переглядом з розширеними можливостями (обертання об'єктів і моделей під будь-яким кутом);
- використання вбудованої бібліотеки 3D-об'єктів;
- інтеграція зі створеною моделлю.

Extend може використовуватися для моделювання як дискретних, так і безперервних систем.

Моделювання процесів в Extend потребує часткового написання коду при заданні властивостей блоків (пакет Extend містить внутрішню мову для задання нових блоків). Цей пакет має підтримку імпорту та експорту даних з Microsoft Visio.

Extend – пакет моделювання дискретних процесів, що використовується для моделювання, аналізу й поліпшення будь-якого процесу, який можна подати у вигляді блок-схем. Extend дає змогу створювати динамічні моделі різнорідних процесів і систем у термінах предметної області, оптимізувати побудовану модель.

Цей пакет дає змогу створювати стохастичні динамічні моделі будь-якого підприємства. Динамічні моделі дають можливість оптимізувати, прогнозувати, планувати діяльність підприємств, а також проводити аналіз йо-

го діяльності на основі отриманих моделей і видавати рекомендації щодо поліпшення роботи підприємства.

Пакет Extend складається з п'яти типів елементів: блоки (дії); стрілки (шляхи переміщення динамічних об'єктів (тегів)); ромби (розгалуження шляхів); черги тегів; точки введення тегів у модель.

2.2.5. GPSS

GPSS/H – мова для моделювання дискретних систем, розроблена Wolverine Software (США). Основою цього програмного продукту є мова імітаційного моделювання GPSS (General Purpose Simulating System – загальна мета системи моделювання) [26].

Основне призначення GPSS – це моделювання систем масового обслуговування, хоча наявність додаткових вбудованих засобів дає змогу моделювати і деякі інші системи (наприклад, розподіл ресурсів між споживачами). Мова GPSS має блочну структуру і може бути легко пристосована для структурно-функціонального моделювання не надто складних систем.

Основними поняттями мови GPSS є транзакт, блок і оператор. Транзакт GPSS – це динамічний об'єкт, під яким може матися на увазі клієнт, вимога, виклик або заявка на обслуговування приладом. Транзакти в GPSS можуть створюватися (вводитися), знищуватися (виводитися), затримуватися, розмножуватись, зливатися, накопичуватися і т. д. Блок GPSS являє собою деякий самостійний елемент системи, що моделюється. Кожен блок реалізує одну або кілька операцій над транзактом, групою транзактів або параметрами транзактів, а сукупність блоків є моделювальною.

Існуюча зараз версія містить понад 70 типів блоків, керуючих операторів, що дають змогу організувати цикли; підтримує подання часу як числа з плаваючою комою, має графічні засоби маніпулювання з блок-схемами і гнучкий інтерфейс зв'язку з C ++.

У 1980 році було представлено два програмних пакети для користування в GPSS / H, обидва з можливостями анімації.

TESS – розширена система моделювання для використання в компанії Pritsker & Associates і автосимулятор AutoGram. Ці анімаційні пакети стали нецікавими користувачам GPSS / H з моменту, коли Wolverine Software в 1990 році розробив власний анімаційний пакет Proof Animation. В автосимуляторі також була програма AutoMod – препроцесор для автоматичної генерації текстів GPSS / H програм. У 80-ті роки дослідники також використовували RESQ для швидкого введення моделей GPSS / H (Mathewson, 1989). У 1993 році Елніскі представив свою програму, яка прискорює введення текстів моделей, оформлену як оболонка GPSS / H для прогону моделей на комп'ютерах типу IBM PC (Wolverine, 1993). На

початку 90-х років MOGUL від High Performance Software було використано для генерації GPSS / H кодів при моделюванні систем зв'язку (Rodrigues, 1993).На початку 90-х років фірма GfL з Аахена (Німеччина) також реалізувала GPSS / H EDITOR – прискорювач введення GPSS / H програм в основному простим натисканням на кнопки з текстом даних блоків, але без справжнього графічного інтерфейсу, тобто без меню з символами блок-діаграм блоків (Knepper and Krönchen, 1993). Починаючи з 1994 року, разом з кожною версією GPSS / H Professional поставляється програма UniFit II, що дає змогу користувачеві підбирати найбільш придптні ймовірнісні розподіли для своїх даних. У середині 90-х років було також розроблено систему SIMSTAT, яка читала й аналізувала вихідні дані GPSS / H (Crain, 1996).

Мова GPSS / H є досить простою в освоєнні, а наявність у неї функцій, змінних, стандартних атрибутів, графіків і статистичних блоків істотно розширює її можливості.

GPSS World – сучасна версія GPSS для персональних ЕОМ і ОС Windows. Система GPSS World – це середовище комп'ютерного моделювання загального призначення, розроблене для професіоналів в області моделювання. Це комплексний моделювальний інструмент, який використовується як для дискретного, так і для безперервного комп'ютерного моделювання, що має найвищий рівень інтерактивності й візуального подання інформації.

GPSS World було розроблено з метою досягнення тісної інтерактивності навіть у багатозадачному середовищі з використанням віртуальної пам'яті. У GPSS World існує кілька анімаційних можливостей. Рівень їх реалізму змінюється від абстрактної візуалізації, яка не потребує ніяких зусиль, до високореалістичних динамічних зображень, що містять складні елементи, створені користувачем.

GPSS World є об'єктно-орієнтованою мовою. Її можливості візуального подання інформації дають змогу спостерігати й фіксувати внутрішні механізми функціонування моделей. Її інтерактивність дає можливість одночасно досліджувати й керувати процесами моделювання. За допомогою вбудованих засобів аналізу даних можна легко обчислити допустимі інтервали і провести дисперсійний аналіз. Крім того, тепер є можливість автоматично створювати й виконувати складні відсіювання й оптимізувальні експерименти.

Можливості GPSS World:

- об'єктно-орієнтований інтерфейс користувача, який містить об'єкти (модель, процес моделювання, звіт і текст);
- високопродуктивний транслятор моделей;
- програмні експерименти з автоматичним аналізом даних;
- багатозадачність, яка дає змогу спільно запускати кілька процесів моделювання та експериментів;

- збереження і продовження виконання запущених процесів моделювання;

- використання механізму віртуальної пам'яті, що дає змогу моделям реально досягати розміру мільярда байт;

- введення / виведення даних під час виконання процесу моделювання;

- понад 20 вбудованих імовірнісних розподілів;

- 17 різних графічних вікон для спостереження за процесом моделювання, що виконується;

- автоматичне інтегрування звичайних диференціальних рівнянь будь-якого порядку;

- швидке і зручне налагодження з використанням графічного інтерфейсу;

- автоматичні генератори відсіювальних і оптимізувальних експериментів;

- пакетний режим з контрольованою процедурою виходу з додатку;

- діалогові вікна введення блоків;

- настроювання інтервалів табуляції;

- можливість динамічного виклику функцій із зовнішніх файлів.

Основне призначення GPSS – це моделювання систем масового обслуговування, хоча наявність додаткових вбудованих засобів дає змогу моделювати і деякі інші системи.

Остання версія GPSS містить 53 типи блоків і 25 команд, а також більш ніж 35 системних числових атрибутів, які забезпечують поточні змінні стану, доступні в будь-якому місці моделі. Основними об'єктами GPSS є: модель, процес моделювання, звіт і текст.

GPSS World сумісний з GPSS / PC і видає результати, які статистично не відрізняються від результатів, які видає GPSS / PC. Цей рівень сумісності можна досягти виправленням деяких відмінностей і запуском процесу моделювання.

Крім того, є доступним ще більш високий рівень сумісності, який називається режимом сумісності з GPSS / PC. У більшості випадків можна досягти точного повторення результатів. Проте GPSS World використовує нову виконувану бібліотеку.

2.2.6. SIMPROCESS

SIMPROCESS – це ієрархічний пакет імітаційного моделювання бізнес-процесів, який дає змогу будувати схему (карту) модельованого процесу, підтримує дискретно-подієве моделювання і функціонально-вартісний аналіз з використанням ABC-методу. Компанія-розробник SIMPROCESS – CACI Products Company (США)[27].

Цей пакет призначено для організацій, яким необхідно аналізувати різні сценарії розвитку підприємства і зменшувати ризики відповідно до мінливих умов діяльності.

SIMPROCESS підтримує дискретно-подієве моделювання. Анімований інтерфейс дає змогу візуалізувати динаміку моделювання «вузьких місць» у моделі.

SIMPROCESS має такі компоненти: процеси (Processes), підпроцеси (Alternative Sub-Processes), дії (Activities), сутності (Entities), ресурси (Resources), з'єднувачі (Connectors), майданчики (Pads).

SIMPROCESS потребує часткового написання програмного коду і підтримує використання UML / XML-технологій, XML / XRDL-процесів, підтримує діаграми Dot, SOAP (Simple Object Acces Protocol), а також експорт і імпорт даних з ODBC і Java, Java RMI, C4ISR / DoDAF, TOGAF, 6 Sigma. SIMPROCESS має модуль «дистанційного керування». За допомогою цього налаштування, що використовує Java в PMMO-технології, користувач може задати параметри методу, що будуть посилатися на об'єкт.

2.2.7. AllFusion Process Modeler (BPWin)

AllFusion Process Modeler (раніше – BPWin) – інструмент візуального моделювання бізнес-процесів, який потребує написання програмного коду і дає можливість наочно подати будь-яку діяльність або структуру у вигляді моделі, що дає змогу оптимізувати роботу організації, перевірити її на відповідність стандартам ISO 9000, спроектувати оргструктуру, знизити витрати, виключити непотрібні операції, підвищити гнучкість і ефективність [28].

AllFusion Process Modeler, розроблений фірмою Computer Associates (США), пропонується для використання компаніям, які прагнуть до оптимальності та ефективності власного бізнесу або бізнесу замовників. AllFusion Process Modeler має інтуїтивно зрозумілий графічний інтерфейс, швидко і легко освоюється, що дає змогу зосередитися на аналізі самої предметної області, не відхиляючись на вивчення інструментальних засобів. AllFusion Process Modeler допомагає швидко створювати й аналізувати моделі з метою оптимізації ділових і виробничих процесів.

Простий у використанні інтерфейс дає можливість заповнювати моделі, але репрезентативні властивості BPWin є низькими, немає стандартних об'єктів для опису бізнес-процесів.

AllFusion Process Modeler підтримує три стандартні нотації: IDEF0 (функціональне моделювання), DFD (моделювання потоків даних) і IDEF3 (моделювання потоків робіт). Ці три основні ракурси дають змогу комплексно описати предметну область.

До основних недоліків цього програмного продукту можна віднести можливість розроблення тільки статичних моделей, ніяк не прив'язаних до часових параметрів реальних процесів.

Інтегрований з ERwin (для проектування БД), Paradigm Plus (для моделювання компонентів ПЗ), із засобом імітаційного моделювання Arena, з численним ПЗ компанії CA-Platinum. За необхідності інформацію можна експортувати в інші додатки (наприклад, у Microsoft Word або Excel).

2.2.8. ProcessModel

ProcessModel – це інструмент для візуалізації, аналізу й удосконалення бізнес-процесів різних типів, включаючи оброблення транзакцій, обслуговування покупців, виробничі й складальні операції, транспортні послуги тощо, розроблений Process-Model, Inc. (США) [29].

ProcessModeler об'єднує просту технологію бізнес-діаграм з можливостями імітаційного моделювання і вбудованими засобами графічної анімації.

Анімація ProcessModeler – це візуальне динамічно обґрунтоване вікно результатів з ключовими індикаторами (показниками) продуктивності, що відображає зміну значень змінних на екрані, зміну графічних елементів моделі пад час моделювання і можливість імпорту графіка з вибраної користувачем програми.

ProcessModel дає змогу реалізувати такі опції:

- розклад роботи персоналу і графік робочого часу;
- пріоритетність і можливість переривання завдань;
- можливість вибору різних методів управління;
- планування місткості операції або процесу;
- можливість завдання розміру партії оброблення;
- розклад призначень;
- визначення послідовності робіт;
- виробничий розклад;
- підвищення продуктивності;
- скорочення циклу оброблення;
- зниження вартості;
- управління якістю;
- аналіз «вузьких місць»;
- оцінювання вартості за операціями і ресурсами;
- розклад доступності ресурсів при перервах у роботі й простоях.

ProcessModel дає змогу створювати ієрархічні моделі для кращої організації й керування великими проектами. За допомогою ProcessModel можна проводити аналіз «що буде, якщо».

Основні блоки ProcessModeller: Activity shape (функція), Decision shape (умова), Links (з'єднувальні лінії (зв'язку)) і Orjoin (логічне АБО), Split

Worksteps (точка злиття), Андеінс (логічне І).

Перевагами ProcessModel є: можливість його швидкого освоєння і легкого використання, особливо для фахівців, які володіють техніками моделювання та інтерактивною, контекстнозалежною системою допомоги, яка забезпечує необхідну інформацію для швидкого і легкого створення моделі предметної області. ProcessModel забезпечує підтримку зв'язків між моделями і об'єктами моделі, Excel, Access, SQL і т. д. Повна інтеграція з VBA (Visual BASIC for Application).

2.2.9. AnyLogic

AnyLogic – інструмент імітаційного моделювання, який дає змогу поєднувати всі існуючі підходи до моделювання. Розробник – Екс Джей Технолоджіс (Росія) [30].

AnyLogic дає можливість створювати інтерактивну 2D- і 3D-анімацію, візуально відображає результати роботи моделі в реальному часі. Засоби анімації дають змогу користувачеві створювати віртуальний світ (сукупність графічних образів, мнемосхему і т. ін.), керований динамічними параметрами моделі за законами, визначеним користувачем за допомогою рівнянь і логіки модельованих об'єктів. Області застосування програмного продукту AnyLogic: ринок і конкурентоспроможність, керування проектами, соціальні та екологічні системи, розвиток міст, переміщення людей і транспортних засобів у безперервному просторі, перехрестя, парковки, будівлі, музеї, черги, транспорт, перевезення, евакуація, виробничі процеси, охорона здоров'я та ін.

AnyLogic має істотну перевагу перед традиційними інструментами моделювання саме в тих проектах, розроблення яких потребує виходу за межі однієї єдиної парадигми моделювання.

AnyLogic застосовується в діапазоні від мікромоделей «фізичного» рівня, де важливими є конкретні розміри, відстані, швидкості, час, до макромоделей «стратегічного» рівня, на якому розглядаються глобальна динаміка зворотних зв'язків, тенденції на тривалих часових відрізках і оцінюються стратегічні рішення.

AnyLogic підтримує моделювання систем як з дискретними, так і безперервними подіями, а також дає змогу комбінувати їх.

Побудова моделі в AnyLogic не потребує написання програмного коду, але якщо стандартних засобів не вистачає (або їх використання є незручним), є можливість застосування мови Java. У найпростішому випадку це зводиться до опису дій, що здійснюються при переході в інший стан, спрацюванні таймера або приході повідомлення. Крім того, можна додавати власний код на Java до активного об'єкта, а також використовувати сторонні бібліотеки. Це робить систему AnyLogic такою, що легко розширюється.

Будь-який об'єкт моделі, що розробляється в AnyLogic, подається як клас Java. Користувач може додати до моделі свої класи, перевизначити методи базових класів, використати базові й розробити власні бібліотеки класів і т. ін. За моделлю, поданою в графічному редакторі, AnyLogic, написаний на Java-платформі, генерує Java-програму, з якою працює. При побудові моделі в AnyLogic розробник фактично створює Java-класи активних об'єктів і визначає відносини між ними. Під час виконання модель являє собою ієрархію екземплярів активних об'єктів.

Зібрана модель може працювати локально, на одному комп'ютері, або ж можна побудувати Java-аплет і запустити його під керуванням браузера.

Простота освоєння AnyLogic визначається знанням користувачем мови Java, яка використовується в комбінації з графічним середовищем розроблення моделей і дає AnyLogic величезну гнучкість і виразність, що водночас є перешкодою для розробників, які не володіють цією мовою.

За допомогою AnyLogic можна створювати моделі архітектури, а саме інтегрувати їх з офісним і корпоративним ПЗ, включаючи електронні таблиці, БД, ERP і CRM-системи, і модулями, написаними іншими мовами.

У системі AnyLogic введено поняття активного об'єкта – розширення класичних об'єктів, які використовуються в об'єктно-орієнтованих мовах програмування. Як і звичайні, активні об'єкти можуть мати властивості й методи. У AnyLogic активний об'єкт може містити також інші елементи:

- параметри – особливі властивості, що визначають параметри моделі;
- зовнішні змінні, які використовуються для зв'язку двох активних об'єктів, коли зміна змінної в одному об'єкті автоматично змінює її значення в іншому;
- порти, що застосовуються для обміну повідомленнями між активними об'єктами;
- стейтчарти (діаграми станів), які задають переходи між станами активних об'єктів;
- таймери, що дають змогу здійснювати періодичні дії, задавати тимчасові затримки;
- математичні функції, які задають функцію з використанням математичних формул.

2.2.10. Witness

Witness – програмний продукт для моделювання виробничих систем, розроблений компанією Lanner (США) [31].

Witness застосовується для такого:

- аналіз вхідних даних і результатів експериментальних даних;
- виявлення правил і структури даних;

- підвищення точності моделей.

Цей пакет підтримує блочне графічне моделювання. Witness охоплює понад 50 стандартних блоків. Основні блоки Witness: деталі, верстати, буфери, роботи.

В останніх версіях Witness додано новий компонент модуля, який дає змогу розробляти власний код в Witness-моделі, завдяки чому можна не тільки забезпечити зв'язок із запланованими COM-бібліотеками, а й автоматично створити функції в Witness для доступу до необхідних бібліотек.

Witness підтримує зв'язок баз даних (Oracle, SQL Server, Access і т. ін.), має прямий доступ до всіх електронних таблиць, за винятком форматів повідомлень XML, HTML, повністю інтегрований з 3D / VR views або Post Processed VR, підтримує зв'язок з Microsoft Visio, забезпечує спектр прямих графічних рішень CAD і багато іншого, а також дає змогу формувати звіти (документацію) процесів.

2.2.11. Arena

Arena – програмний продукт, розроблений компанією Systems Modeling Corporation (США) і призначений для імітаційного моделювання. Він дає змогу створювати моделі, використовуючи реальні системи, які можна адекватно подати. Система не потребує написання програмного коду і є виключно простою у використанні, але для її освоєння необхідно багато часу і досить глибоких знань теорії ймовірностей, математичної статистики, теорії систем масового обслуговування і мереж Петрі [32].

Для відображення результатів моделювання використовують анімаційну систему Cinema animation. Інтерфейс Arena має всілякі засоби для роботи з даними, у тому числі електронні таблиці, бази даних, ODBC, OLE, а також підтримує формат DXF.

Система імітаційного моделювання Arena містить:

- двовимірний графічний редактор;
- тривимірний графічний редактор (пакет 3D-player);
- редактори тимчасових шаблонів і розкладів;
- редактор символів і бібліотеку графічних заготовок.

Ця система підтримує зв'язок з бібліотекою графічних заготовок і буфером обміну Microsoft.

Область застосування системи імітаційного моделювання Arena:

- служби роботи з клієнтами, керування внутрішніми бізнес-процесами, такими, як виконання замовлень, обслуговування або керування простими виробничими потоками;

- складні великомасштабні проекти, що відрізняються високою чутливістю до змін у системах керування логістичними ланцюжками, виробничими процесами, логістикою, збутом, складуванням і системами обслуговування; шляхом створення спеціалізованих шаблонів для складної логіки

та логіки, що повторюється, можна спростити процеси і зменшити час розроблення моделей;

- вирішення оперативних і стратегічних питань розроблення пакувальних ліній, здійснення інвестицій у нове обладнання, розроблення чутливої логіки і конвеєрного оброблення, а також планування промислових процесів великосерійного виробництва в змішаних дискретно-безперервних системах;

- розроблення стратегій роботи з клієнтами, таких як перехід на нові сценарії оброблення викликів, використання віртуальних довідкових служб, переадресація на основі практичного досвіду і моделювання кадрового забезпечення;

- ефективний засіб постоброблення, що забезпечує можливість створення і перегляду тривимірних анімацій існуючих моделей Arena;

- інструмент оптимізації завдань, призначений і спеціально налаштований на аналіз результатів моделювання, виконаного за допомогою пакета Arena;

- поширення моделей Arena для перегляду і проведення експериментів.

Arena дає змогу використовувати дискретне, безперервне, а також дискретно-безперервне моделювання.

Цей програмний продукт підтримує взаємодію з пакетом VBA Visual Basic for Applications корпорації Microsoft; об'єктною моделлю ActiveX для зовнішнього керування, доступу через ADO / ODBC до баз даних (Oracle, Access, Excel, SQL), а також імпортування файлів з пакетів AutoCad (у форматі dxf), Visio, Blue Pumpkin Workforce і комунікацію між окремими процесами.

Arena містить 200 навчальних посібників і бібліотек SMART (залежно від версії), електронні керівництва і універсальні вбудовані довідкові системи, бази знань з Web-інтерфейсом.

2.3. Планування й проведення експериментів з моделями

Процес моделювання має ітераційний характер і проводиться у межах раніше сформульованих цілей з дотриманням принципів моделювання. Побудова моделей починається з вивчення (обстеження) реальної системи, її внутрішньої структури і змісту взаємозв'язків між її елементами, а також зовнішніх впливів і завершується розробленням цієї моделі.

Схему етапів моделювання подано на рис. 2.1.

Слід зазначити, що при розробленні конкретних моделей з певними цілями і межами моделювання необов'язково виконувати всі підетапи.

На етапі аналізу вимог і проектування формулюється концептуальна модель, будується її формальна схема і вирішується доцільність подальшого моделювання системи.

Рис. 2.1. Схема створення моделі

Концептуальна модель (КМ) – це абстрактна модель, яка визначає склад і структуру системи, властивості елементів і причиново-наслідкові зв'язки, властиві аналізованій системі й суттєві для досягнення цілей моделювання. У таких моделях зазвичай у словесній формі наводяться відомості про природу і параметри (характеристики) елементарних явищ досліджуваної системи, вид і ступінь взаємодії між ними, місце і значення кожного елементарного явища в загальному процесі функціонування системи. При створенні КМ майже паралельно формується область вихідних даних (інформаційний простір системи) – етап підготовки вихідних даних. На цьому етапі визначаються кількісні характеристики (параметри) функціонування системи і її елементів, числові значення яких є вихідними даними для моделювання.

Очевидно, що значна частина параметрів системи – це випадкові величини. Тому при формуванні вихідних даних особливе значення мають вибір законів розподілу випадкових величин, апроксимація функцій і т. д.

Після виявлення властивостей моделі й побудови концептуальної моделі необхідно перевірити адекватність моделі.

На етапі розроблення моделі уточнюється або вибирається програмний пакет моделювання. Програмні й технічні засоби моделювання вибираються з урахуванням кількох критеріїв. Обов'язковими умовами при цьому є достатність і повнота засобів для реалізації концептуальної моделі. Серед інших критеріїв можна назвати доступність, простоту і легкість освоєння, швидкість і коректність створення програмної моделі.

Після вибору середовища проектування концептуальна модель, сформульована на попередньому етапі, перетворюється на комп'ютерну, тобто вирішується проблема алгоритмізації і деталізації моделі.

Модель системи подається у вигляді сукупності частин (елементів, підсистем). До цієї сукупності додаються всі частини, які, з одного боку, забезпечують збереження цілісності системи, а з іншого – досягнення поставлених цілей моделювання (отримання необхідної точності й достовірності результатів при проведенні комп'ютерних експериментів над моделлю). Надалі проводяться остаточна деталізація, локалізація (виділення системи з навколишнього середовища), структуризація (указання і загальний опис зв'язків між виділеними елементами системи), укрупнений опис динаміки функціонування системи і її можливих станів.

Для задання модельного часу введемо поняття «модельний час». У комп'ютерній моделі змінна, що забезпечує поточне значення модельного часу, називається годинником модельного часу.

Існує два основні підходи до просування модельного часу: просування часу від події до події і просування часу з постійним кроком [14].

Просування часу в моделі від події до події застосовується у всіх основних комп'ютерних програмах і більшістю розробників, що створюють свої моделі універсальними мовами (рис. 2.2) [14].

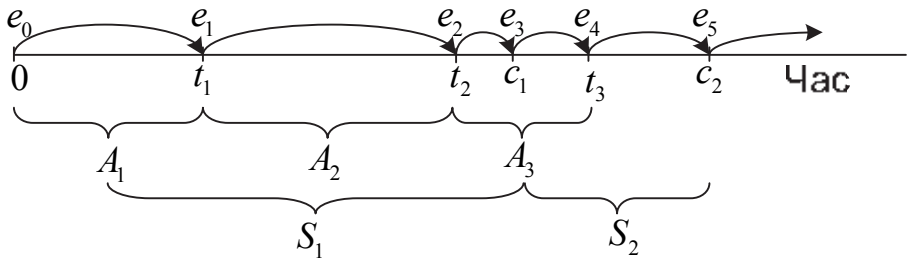


Рис. 2.2. Механізм просування модельного часу від події до події

При використанні просування часу від події до події годинник модельного часу в початковому стані встановлюється в 0 і визначається час виникнення майбутніх подій. Після цього годинник модельного часу переводять на час виникнення найближчої події, і в цей момент оновлюються стан системи з урахуванням події, що сталася, а також відомості про час виникнення майбутніх подій. Потім годинник модельного часу просуває-

ся до часу виникнення наступної нової найближчої події, оновлюється стан системи, визначається час майбутніх подій і т. д. Процес просування модельного часу від часу виникнення однієї події до часу виникнення іншої триває доти, доки не буде виконано певну умову або умову зупинки, вказану заздалегідь. Оскільки в дискретно-подієвій імітаційній моделі всі зміни відбуваються тільки під час виникнення подій, періоди бездіяльності системи просто пропускаються і годинник переводиться з часу виникнення однієї події на час виникнення іншої. При просуванні часу з постійним кроком такі періоди бездіяльності не пропускаються, що призводить до великих витрат комп'ютерного часу. Слід зазначити, що тривалість інтервалу просування модельного часу від однієї події до іншої може бути різною [14].

При просуванні часу з постійним кроком Δt годинник модельного часу просувається точно на Δt одиниць часу для будь-якого вибраного значення Δt. Після кожного оновлення годин виконується перевірка, щоб визначити, відбулися які-небудь події протягом попереднього інтервалу часу Δt чи ні. Якщо на цей інтервал заплановано одну або кілька подій, вважається, що ці події відбуваються в кінці інтервалу, після чого стан системи і статистичні лічильники відповідним чином оновлюються. Просування часу за допомогою постійного кроку показано на рис. 2.3, де вигнуті стрілки показують

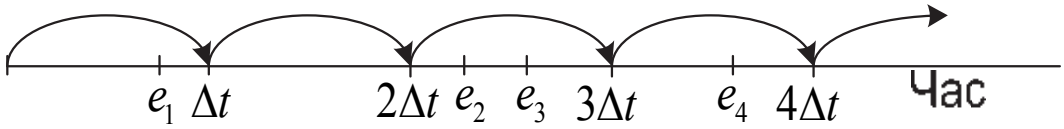


Рис. 2.3. Приклад просування модельного часу за допомогою постійного кроку

просування модельного часу, а e_i (i = 1, 2, ...) – це дійсний час виникнення події будь-якого типу, а не значення годин модельного часу. На інтервалі [0, Δt) подія відбувається в момент часу e_1, але вона розглядається як те, що сталося в момент часу Δt. На інтервалі [Δt, 2Δt) події не відбуваються, але все ж модель виконує перевірку, щоб переконатися в цьому. На інтервалі [2Δt, 3Δt) події відбуваються в моменти часу e_1 і e_3, проте вважається, що вони відбулися в момент часу 3Δt і т. д. У ситуаціях, коли прийнято вважати, що дві або кілька подій відбуваються в один і той самий час , необхідно застосувати кілька правил, які дають змогу визначити, в якому порядку обробляти події. Таким чином, просування часу за допомогою постійного кроку має два недоліки: виникнення помилок, пов'язаних з обробленням подій у кінці інтервалу, протягом якого вони відбуваються, а також необхідність вирішувати, яка подія оброблятиметься першою, якщо події,

що в дійсності відбуваються в різні часи, розглядаються як одночасні. Подібного роду проблеми можна частково вирішити, зробивши інтервали Δt менш тривалими, але тоді зросте кількість перевірок виникнення подій, що призведе до збільшення часу виконання завдання. Через цю обставину просування часу за допомогою постійного кроку не використовують у дискретно-подієвих імітаційних моделях, коли інтервали часу між послідовними подіями можуть значно відрізнятися за своєю тривалістю [14].

В основному цей підхід призначено для систем, в яких можна допустити, що всі події в дійсності відбуваються в один з моментів n часу Δt (n = 0, 1, 2, ...) для будь-якого вибраного Δt. Так, в економічних системах дані часто подаються за річні проміжки часу, тому природно в імітаційній моделі встановити просування часу з кроком, що дорівнює одному року.

Слід зауважити, що просування часу з постійним кроком можна виконати за допомогою механізму просування часу від події до події, якщо запланувати час виникнення подій через Δt одиниць часу, тобто цей підхід є різновидом механізму просування часу від події до події.

На етапі проведення експерименту, який є вирішальним, завдяки процесу імітації модельованої системи відбуваються збір необхідної інформації, її статистичне обробляння в інтерпретації результатів моделювання, унаслідок чого приймається рішення: дослідження продовжити або закінчити.

Якщо результат є відомим, то його можна порівняти з одержаним результатами моделювання. Отримані висновки часто сприяють проведенню додаткової серії експериментів, а іноді й зміні моделі. Основою для прийняття рішення є результати тестування і експериментів. Якщо результати не відповідають цілям моделювання (реальному об'єкту або процесу), то це означає, що помилки, допущені на попередніх етапах, або вхідні дані не є кращими параметрами у досліджуваній області. Тому розробнику слід повернутися до одного з попередніх етапів.

Аналіз результатів моделювання є всебічним аналізом отриманих результатів з метою формування рекомендацій з проектування системи або її модифікації.

На етапі підведення підсумків моделювання згідно з поставленою метою і завданням моделювання оцінюють виконану роботу, зіставляють поставлені цілі з отриманими результатами і створюють остаточний звіт про виконану роботу.

2.4. Оформлення результатів моделювання

Результати моделювання необхідно оформити у вигляді звіту, який має містити три розділи. Розглянемо їхній зміст.

2.4.1. Технічне завдання

Перший розділ пояснювальної записки повинен містити технічне завдання на створення імітаційної моделі й такі підрозділи:

1. «Призначення розробки». У цьому підрозділі необхідно описати модель із розділу 3.

2. «Вхідні дані». Тут слід навести вхідні дані (табл. 2.2).

Таблиця 2.2

Вхідні дані для проведення імітаційного моделювання

Найменування / Тип	Обмеження	Призначення

1. «Опис вихідних даних». Цей підрозділ має містити або аналогічну таблицю (табл. 2.2), або опис елементів анімації, які дають змогу зробити висновок про адекватність досліджуваної моделі (якщо модель містить графіки, то в записці потрібно вказати, яку залежність вони показують).

2. «Інтерфейс користувача». У цьому підрозділі треба описати анімацію моделі й навести рисунок вікна анімації при запущеній моделі.

3. «Вибір методу моделювання». Необхідно вказати, який вид моделювання було вибрано для побудови імітаційної моделі (дискретно-подієвий, агентний і т. д.) і чому.

4. «Мета моделювання». Слід указати, яку практичну користь і кому дасть ця модель.

2.4.2. Дослідження імітаційної моделі

Другий розділ «Дослідження імітаційної моделі» повинен містити опис принципу створення вибраної моделі й такі підрозділи:

1. «Аналіз предметної області». Якщо модель створюється на основі відомих математичних залежностей, то в цьому підрозділі треба їх навести. Якщо в процесі дослідження використовуються статистичні дані, то їх також необхідно навести.

2. «Структурний аналіз ». У цьому підрозділі треба описати всі елементи, наведені у вікні «Проект», їх призначення, параметри, що використовуються, рівняння і призначення додаткового коду класу, якщо такий є. Якщо для елементів моделі створюється опис її поведінки (statechart), необхідно навести його у записці й указати умови переходу з одного стану в інший

3. «Алгоритми роботи елементів моделі». Цей підрозділ має містити опис або блок-схеми алгоритмів, заданих у додатковому коді класу.

2.4.3. Проведення експерименту

У цьому підрозділі має бути описано не менше трьох експериментів. Для кожного експерименту слід навести вхідні параметри, рисунки з анімацією роботи моделі й результати у вигляді графіків, значень параметрів або словесного опису, а також зробити висновок про кількісну оцінку моделі.

Наприклад, для моделі терміналу аеропорту можна провести такий експеримент (рис. 2.4): закриття всіх пунктів контролю безпеки.

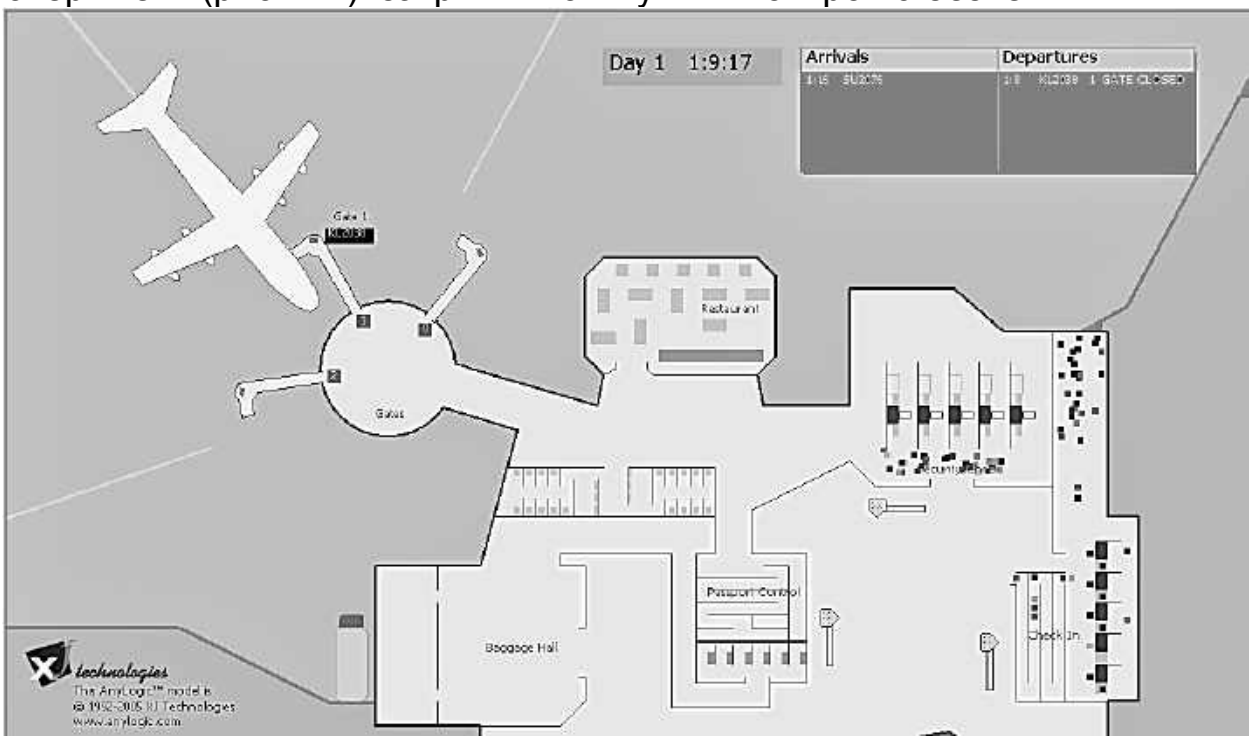


Рис. 2.4. Анімація моделі внаслідок проведення експерименту

Унаслідок проведення експерименту:

- черга пасажирів збільшується й після 34-ї хвилини система виходить з ладу;
- усі пункти реєстрації й паспортного контролю працюють повністю;
- за цей час прибуло два літаки.

При відкритті трьох пунктів паспортного контролю, трьох пунктів контролю реєстрації і одного пункту контролю безпеки система працює в оптимальному режимі, при цьому немає ніяких виходів з ладу системи.

Отже, якщо термінал аеропорту обслуговуватиме три злітно-посадкових смуги із середньою кількістю пасажирів на борту літака – 30 осіб, що прибувають через 10 – 20 хвилин, необхідно мати три пункти паспортного контролю, три пункти контролю реєстрації і один пункт контролю безпеки.

3. ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ВИРОБНИЧИХ І КОМПЮТЕРНИХ СИСТЕМ

Номер підрозділу відповідає варіанту самостійного завдання для виконання розрахунково-графічної роботи (РГР).

3.1. Модель "Виробництво мінеральної води і тоніка"

Модель, яка імітує технологічну лінію з виробництва мінеральної води і тоніка, включаючи складування й вивезення готової продукції. Посилання на працюючу версію цієї моделі можна знайти на сторінці http://www.xjtek.com/anylogic/demo_models/manufacturing_logistics/ під назвою Beverage Production (тут же описано, як запустити модель). Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Manufacturing and Logistics").

У діючому вигляді ця модель містить такі основні елементи:

- анімоване подання всіх елементів технологічного процесу виробництва мінеральної води і тоніка, у тому числі індикатори поточного стану кожного елемента;
- меню моделі, яке має сім змінюваних параметрів у вигляді повзунка, що дає змогу під час роботи моделі змінювати деякі умови роботи певного технологічного процесу (рецептуру мініральної води і тоніка, кількість мийних машин для пляшок, кількість підйомно-розвантажувальних машин і контейнерів, які вони можуть перевозити);
- текстові пояснення до виведеної інформації, а також опис того, як перемикати анімоване подання роботи певного виробництва на технологічну лінію або склад готової продукції.

При запуску моделі анімована схема технологічного процесу починає рухатися зліва направо. Порожні пляшки надходять з двох джерел: нові й вже раніше використані. Раніше використані пляшки спочатку потрапляють в мийку (можна задати кількість мийних машин). Потім пляшки з обох джерел проходять перевірку і роздвоюються на лінії підготовки мінеральної води і тоніка. Напої готуються з інгредієнтів, запаси яких задано в моделі: вода, вуглекислий газ, спирт і ромова есенція, що змішуються в заданих пропорціях (пропорції можна змінювати параметрами "рецептура"). Заповнені пляшки проходять через два апарати з наклеювання етикеток, потім складаються в ящики. Як тільки на виході технологічної лінії накопичується 5 ящиків напоїв, за ними приходить автомобіль, який доставляє їх контейнером на склад. На складі автомобілі розвантажуються зліва на анімованій схемі складу двома підйомно-розвантажувальними машинами. Контейнери з мінеральною водою накопичуються в нижній частині складу, а з тоніком – у верхній. Ще одна підйомно-розвантажувальна машина доставляє кон-

тейнери на праву частину складу, де вони забираються автомобілями покупців. Кількість щодня закуповуваних пляшок напою також задається в моделі.

Модель є прекрасною анімованої презентацією відповідного виробництва. Вона дає його керівникам просту для розуміння і компактну інформацію про стан усіх процесів / елементів технологічного ланцюжка в задані моменти часу. Щоб створити аналогічний рівень інформованості традиційними таблицями і графіками треба було б набагато більше зусиль як від керівників, так і від аналітиків. Параметри моделі, що настроюються (виконані у вигляді повзунків), дають змогу під час її роботи аналізувати наслідки змінення деяких факторів (наприклад, вихід з ладу підйомно-розвантажувальних машин, зміни в рецептурі напоїв і т. ін.). Кількість подібних параметрів моделей, що настроюються, можна за необхідності збільшити.

3.2. Модель "Відділення швидкої допомоги"

Модель, яка імітує функціонування відділення швидкої допомоги при великій лікарні, включаючи детальну статистику про використання ресурсів відділення. Посилання на працюючу версію цієї моделі можна знайти на сторінці http://www.xjtek.com/anylogic/demo_models/healthcare/ під назвою Emergency Department. Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Healthcare").

У діючому вигляді ця модель містить такі самі основні елементи, що і попередня модель. Однак тут передбачено тільки два параметри (два повзунки), що настроюються: кількість хворих, що приходять до відділення за одиницю модельного часу; відсоток хворих, що ходять, у їх загальному потоці.

Модель відділення швидкої допомоги імітує обслуговування вхідного потоку хворих з випадковим розподілом їх за видами травм і захворювань наявними ресурсами відділення. Ресурси відділення складаються з певної кількості різних фахівців, кількості спеціалізованих кабінетів і робочого часу фахівців.

Додатковим обмеженням є організація простору відділення, переміщення по якому хворих і персоналу потребує часу, спричиняючи затримки в обслуговуванні хворих, створюючи додаткові черги і т. ін. Для кожного хворого, що поступив залежно від виду його захворювання в моделі призначається певна схема обслуговування, реалізація якої візуалізується на анімованій презентації у вигляді переміщення хворого із кабінету в кабінет, у тому числі в супроводі персоналу відділення, очікування в чергах, коли звільниться необхідний фахівець або кабінет, і т. ін.

Зведена статистика також демонструє поточні показники використання всіх ресурсів відділення, середній час перебування хворих у чергах, загальний час, проведений хворими у відділенні, включаючи час, проведений у кабінетах, і т. ін.

Подібна комп'ютерна імітаційна модель наочно демонструє функціонування великих / складних організаційних систем, які майже неможливо подати традиційними способами. Складний організаційний механізм постає тут у вигляді реалістичної картини зміни стану елементів системи і зв'язків між ними. Керівник подібної організації може за допомогою моделі проаналізувати фактичні та / або можливі причини виникнення черг в обслуговуванні й проімітувати доступні йому варіанти поліпшення ситуації. Можливими є аналіз різних сценаріїв розвитку подій і пошук найкращих рішень (включаючи розв’язання оптимізаційних задач).

Подібна модель може бути корисним інформаційно-довідковим ресурсом для пацієнтів, якщо вона з'єднана з інформаційною системою організації і регулярно актуалізується на основі одержуваних від неї реальних даних (зайнятість фахівців, довжина черги пацієнтів та ін.). За допомогою такої моделі пацієнти можуть отримувати прогноз часу очікування в черзі, оцінювати загальний час і необхідну їм схему обслуговування і т. ін. Родичі хворих можуть переглядати аналогічну інформацію через Інтернет.

3.3. Модель "Глобальна конкуренція компаній з виробництва пульпи"

Модель, яка імітує функціонування шести транснаціональних компаній з виробництва деревної пульпи (компанії мають кілька підприємств, у тому числі розташованих на різних материках). Модель імітує конкуренції компаній на глобальному ринку пульпи, де перевагу мають компанії з меншою собівартістю продукції, що залежить від вартості деревини на континентах, де розташовуються їхні заводи. Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Marketplace and competition").

До параметрів моделі належать:

- світовий попит на пульпу, який змінюється повзунком внизу праворуч під гістограмою Demand and Price for Pulp;

- вартість деревини на кожному континенті, поточний рівень якої відображає кольоровий / цифровий індикатор, розташований поруч з кожним континентом (див. Cost of wood in ...); колір континенту відповідає поточній вартості деревини, встановленій на індикаторі, кольоровий індикатор перетворюється на повзунок і дає змогу змінити вартість деревини відповідного континенту, якщо зняти чекбокс у відповідного індикатора (див. приклад для індикатора євроазіатського континенту);

- період часу, через який компанія оцінює результати своєї роботи і за підсумками приймає рішення про зміну статусу своїх заводів (зупинення, реконструкція або закриття), значення, що змінюється повзунком у вікні відповідної компанії (див. Strategy review period), розташованим у лівій частині анімованої презентації моделі.

Кожен окремий завод поданий у моделі параметрами потужності й технологічного рівня. Собівартість виробленої заводом пульпи залежить від вартості деревини на континенті, де завод розташований, і технологічного рівня заводу. Щодо поточної світової ціни пульпи завод може виявитися прибутковим або ні.

Світова ціна пульпи встановлюється в моделі на рівні собівартості пульпи замикаючого заводу (заводи впорядковано за збільшенням собівартості), продукція якого ще є потрібною для задоволення світового попиту на пульпу. Замикаючий завод має нульову прибутковість. На гістограмі Demand and Price for Pulp зліва від червоної лінії, яка позначає встановлений рівень світового попиту на пульпу, знаходяться прибуткові заводи, а праворуч – збиткові.

За замовчуванням у моделі світовий попит на пульпу і вартість деревини на континентах змінюються за синусоїдою за умови, що ці параметри не змінюються вручну через відповідні повзунки.

Усі шість компаній реалізують в моделі однакову стратегію:

- якщо завод, який працює, стає неприбутковим, то він зупиняється;
- якщо зупинений завод може стати прибутковим, то він відновлює роботу, інакше він реконструюється;
- кілька разів реконструйовані заводи, якщо вони не стали прибутковими, закриваються;
- якщо світовий попит на пульпу не покривається, а компанія є не надто великою (її кількість заводів менша за встановлений в моделі ліміт), то вона створює нові заводи в регіонах з найдешевшою деревиною.

Подібна імітаційна модель є прикладом опису поведінки незалежних учасників з конфліктом інтересів (виграш одних означає програш інших). У разі якщо можливості учасників є однаковими, оптимальною стратегією їх поведінки стає необхідність домовлятися і погоджувати свої дії. Модель створює умови для координації дій між учасниками і може бути інструментом, що полегшує підтримку цієї системи незалежних дійових осіб у скоординованому стані.

3.4. Модель «Приклад V-подібного двигуна внутрішнього згоряння»

Модель, яка імітує геометричний і кінетостатичний аналіз плоских важільних механізмів Gas Engine. Повний вихідний код цієї моделі включе-

но в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Dynamic Systems").

Більшість плоских важільних механізмів з нижчими кінематичними парами (КП) (преси, насоси, компресори і т. ін.) складаються з кривошипа і однієї або декількох діад (дволанкових груп Ассура). У моделі показано кінематичну схему насоса двигуна внутрішнього згоряння. Цей механізм складається з кривошипа (одноланкової однорухомої структурної групи) і двох діад.

За допомогою побудованої моделі користувач має можливість:

- спостерігати механізм у русі,
- змінювати всі параметри механізму;
- отримувати графіки функцій $xE(t)$ і $yE(t)$ довільно вибраної точки спостереження E;
- досліджувати вплив параметрів механізму на його працездатність;
- спостерігати особливі (сингулярні) стани.

Можна також змінювати масштаб зображення, швидкість обертання кривошипа, положення механізму у вікні анімації.

3.5. Модель «Приклад CALL-CENTERS»

Модель, яке імітує роботу декількох CALL-центрів. Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Business Processes").

Дзвінки здійснюються в кожному з кількох взаємозалежних регіональних центрів оброблення викликів. Якщо жоден оператор не може відповісти на виклик, він поміщається в чергу. Якщо чергу заповнено, центр намагається маршрутизувати виклик в інший центр. Якщо є ще один центр, який здатний прийняти виклик, то він передається туди. В іншому випадку виклик скидається.

Заявки, у цьому випадку дзвінки, являють собою якісь пасивні об'єкти, які переміщаються, захоплюють і звільняють ресурси відповідно до потокових діаграм-схем, що описують досліджуваний процес.

3.6. Динаміка вживання алкоголю

У цій моделі, розробленій спільно з Research Triangle Institute International, досліджується ставлення людей до алкоголю, тривалість життя і пов'язані з цим витрати на охорону здоров'я.

Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Business Processes").

Розрізняють чотири стани людини: не вживає алкоголь взагалі (Never User), вживає час від часу (Recreational User), алкоголік (Addict) і кинув

(Quitter). Переходи між станами – це стохастичні тайм-аут. Наприклад, вік, коли людина починає пити ("ініціюється"), – це реалізація випадкової величини з розподілом Initiation Time Distribution. Розподіл побудовано на базі наявних статистичних даних, а саме ймовірностей ініціації для різного віку. Те ж стосується тривалості життя, розподіленої за законом Death Time Distribution, але цей розподіл може змінюватися залежно від ставлення людини до алкоголю. У цій моделі агенти не взаємодіють один з одним.

Розглядаємо дві групи людей: одна – з "природною" алкогольною динамікою (Normal), інша – що зазнала втручання (Intervened). Втручання (це можуть бути зміни в законодавстві, "соціальні" рекламні кампанії і т. ін.) моделюється як зміни в можливостях ініціації і відмови від алкоголю.

Приклад результатів моделювання (кількість тих, що не пьють, тих, що вживають алкоголь, алкоголіків і тих, хто кинув залежно від віку) показано у вигляді стекових графіків. Група, що зазнала втручання, тут має ймовірність ініціації в два рази нижчу, а ймовірність кинути у два рази вищу, ніж нормальна група. У групі Intervened люди живуть у середньому довше, і ресурсів на їх медичне обслуговування йде менше. Такі моделі використовуються для підтримки прийняття рішень (decision support) при розробленні федеральних, муніципальних або корпоративних політик.

3.7. Модель життєвого циклу продукту

Повний вихідний код побудови моделі життєвого циклу продукту включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Agent Based Modeling Tutorial Models"). Ця модель складається з кількох моделей: Product Life Cycle 1, Product Life Cycle 2, Product Life Cycle 3. Однак ці моделі не містять анімацію процесу моделювання, тому для проведення експериментів використовують модель Product Life Cycle 5 - Animation.alp.

Модель описує процес поширення продукту. Спочатку продукт є нікому не відомим, і для того, щоб люди почали його купувати, його рекламують. Унаслідок цього певна частка людей купує продукт під впливом реклами. Люди також купують товар унаслідок спілкування з тими, хто цей продукт вже придбав. Процес придбання нового продукту під впливом переконання його власників чимось схожий на поширення епідемій.

Головне завдання моделі поширення продукту – вивчення того, як швидко люди купують новий продукт. Тому в моделі є змінні, що підраховують кількість споживачів і потенційних споживачів продукту.

У другій моделі люди купують товар тільки під впливом реклами. Насправді, рекламний ефект відіграє значну роль лише в момент випуску продукту на ринок. Надалі все більше значення матиме спілкування людей з тими своїми знайомими, які цей продукт вже придбали. В основному лю-

ди купують нові продукти саме під впливом переконання своїх знайомих; цей процес чимось схожий на поширення епідемій.

Третя модель імітує ситуацію, коли з часом продукт може бути витраченим або непридатним, що спричиняє необхідність його повторного придбання. Промоделюйте повторні купівлі, вважаючи, що споживачі продукту знову стають потенційними споживачами, коли продукт, який вони придбали, стає непридатним.

3.8. Вивчення поширення декількох продуктів

Модель варіанта 3.7 вивчає динаміку поширення одного продукту в певному сегменті ринку. У цьому варіанті моделі передбачається, що продуктів буде кілька і кількість тих, які проживають у досліджуваній області, також буде непостійною.

Повний вихідний код побудови моделі життєвого циклу продукту включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Agent Based Modeling Tutorial Models"). Ця модель складається з моделей Product Life Cycle 8, Product Life Cycle 9. У моделі є змінна, яка буде визначати тип рекламованого продукту.

3.9. Модель «хижаки – жертви (Predator Prey)» – агентна версія

СД-модель «хижаки – жертви» (рисі та зайці) складається з пари диференціальних рівнянь, які описують динаміку популяцій хижаків і жертв (або паразитів-носіїв) в її найпростішому випадку (одна популяція хижаків, одна – жертв). Модель запропонували незалежно один від одного Альфред Лотка і Віто Вольтерра (Alfred Lotka і Vito Volterra) в 1920-х роках; вона характеризується коливаннями в розмірах обох популяцій, причому пік кількості хижаків трохи відстає від піку кількості жертв. У моделі прийнято такі спрощені припущення: жертви завжди мають достатню кількість ресурсів і гинуть, тільки будучи з'їденими хижаками; жертви – єдине джерело їжі для хижаків, і хижаки вмирають тільки від голоду; хижаки можуть з’їдати необмежену кількість жертв; простір проживання не має розмірності, тобто будь-який хижак може зустріти будь-яку жертву.

На основі розглянутої моделі було запропоновано агентну модель. Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Ecosystem Dynamics").

У агентній моделі: зайці (hares) і рисі (lynx) мають кінцеву тривалість життя, тобто вони вмирають також і від старості, а не тільки будучи з'їденими або від голоду; зайці й рисі живуть у двовимірному просторі (у термінології агентного моделювання "space-aware"); кількість зайців є обмеженою (наприклад, деяким харчовим ресурсом), так що зайці розмножуються, тільки якщо навколо досить вільного місця; рись може зловити зайця

тільки поблизу від місця її проживання; рись полює періодично; якщо під час полювання заєць залишається неспійманим, рись переміщається; і якщо рись так і не знаходить зайця протягом певного часу, вона вмирає.

Дії розгортаються на площині: видно атаки рисей, їх вимирання там, де з'їдені всі зайці, і швидке заповнення зайцями вільного від рисей простору. На агрегатному (кількісному) рівні модель показує коливальну поведінку, схожу на поведінку СД-моделі (піки популяції рисей ідуть за піками популяції зайців). Залежно від параметрів рисі можуть повністю вимерти (іноді разом із зайцями), чого ніколи не трапляється в СД-моделі через її безперервність. Осциляції є стохастичними через стохастичний характер моделі.

3.10. Модель зграй стрижів-бійців

Теоретичні розрахунки переконливо свідчать про те, що з появою стрижів у тривалій історії еволюції живих літальних апаратів було поставлено крапку. Краще за стрижа літати неможливо. Швидкість і траєкторію розраховано буквально з точністю до міліметра і мілісекунди. Агентну модель стрижів запропонував Джошуа Ліфтон (Joshua Lifton). Повний вихідний код цієї моделі включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Miscellaneous").

При спостереженні польоту стрижів було помічено: коли стриж самостійно здійснює свій політ, його траєкторія нагадує коло; коли стриж знаходиться в зграї, його політ ускладнюється. Було прийнято чотири правила щодо сусідніх птахів: стриж летить у тому ж напрямку, в якому летить його зграя; він має бути в центрі локального кластера, щоб уникнути зіткнення з іншими птахами; повинний «вивчати» область попереду, огинаючи інших птахів, які можуть загородити його кут огляду.

Модель нагадує гру, в якій на зграї (команди) стрижів, що випускають, можна робити ставки, яка команда переможе. Таким чином, стрижі є деякими монстрами, які змагаються і вбивають собі подібних з інших команд та ще й роблять це зграями. Модель алгоритмічно реалізовано непросто, але дуже красиво.

3.11. Транспортна модель міста

Оскільки організація керування транспортними потоками належить до такої області, де провести натурний експеримент важко або неможливо, імітаційне моделювання в багатьох випадках стає єдиним інструментом ефективного прийняття рішень в цій області. Одним з основних переваг цього методу є те, що на відміну від аналітичного імітаційне моделювання транспортних потоків дає змогу багаторазово відтворювати досліджувану систему і визначати оптимальний її стан. Повний вихідний код моделі

Urban Dynamics включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Social Dynamics").

Місто в моделі поділено на п’ять незалежних зон, які мають свої власні характеристики: кількість житлових приміщень і бізнес-офісів, комфорт житла і якість доріг. Сім'ї складаються з кількох людей, які можуть працювати на підприємствах. Сім'ї в моделі з'являються, зникають, об'єднуються або розділяються залежно від поведінки людей. Члени сім'ї можуть змінювати роботу (підприємства). Крім того, сім'ї можуть переїжджати з однієї зони в іншу. У цьому випадку витрати на транспорт і ціна оренди змінюються.

Результатом роботи моделі є пікове навантаження транспорту (час поїздки на роботу і назад), а також результати розрахунку викидів CO_2.

3.12. Модель завоювання ринку цукерок

Повний вихідний код моделі Candy Promotion Game включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Marketplace and Competition").

Це проста модель споживчого ринку і конкуренції. Споживачі (діти) живуть у п'яти різних містах. Цукерки випускаються три компанії, що конкурують. Кожна дитина має один улюблений вид цукерок (показує певну лояльність групі), але може переключитися на інший вид під впливом інших дітей, ціни, промоакцій або випадково. Діти можуть переміщатися з одного міста в інше і спілкуватися з іншими дітьми в тому ж місті. Модель виконано у вигляді гри: ви можете управляти стратегією компанії (ціноутворення і просування) під час моделювання.

3.13. Модель терміналу аеропорту

Модель терміналу аеропорту дає змогу оптимізувати його пропропускну здатність. Повний вихідний код моделі Airport Terminal включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Manufacturing and Logistics").

Змінюючи кількість контрольно-пропускних пунктів, можна впливати на потоки пасажирів терміналу. Кількість службовців на контрольно-пропускних пунктах змінюється динамічно керуючими елементами «повзунок». Інформаційне табло демонструє віртуальний час моделювання, час прибуття і відправлення рейсів.

Пасажири, які прийшли в аеропорт на посадку, проходять відділення перевірки паспортів і квитків, після чого оформляють свій багаж, який віддають на навантаження. Самі пасажири проходять додаткову перевірку

(рентген) і відправляються в зал очікування або в ресторан або справляти свої особисті потреби.

Пасажири, які прибули, чекають, коли з літака вивантажать багаж, потім виходять з літака, направляються до паспортного контролю, в разі успішного проходження забирають свої речі з багажного відділення, де, мабуть, працює контейнер, і просто йдуть (якщо речей з собою не брали).

Окремо в моделі розглядається багажне відділення, де завантажується автотранспорт, який здійснює перевезення у літак, існує анімація його руху до літака і розвантаження.

3.14. Модель сільськогосподарської логістики

Імітаційна модель збору врожаю (Examples \ Manufacturing and Logistics) демонструє роботу трьох різних машин (комбайна, вантажівки і фури). Її призначено для демонстрації логістичної динаміки, пов'язаної зі збором урожаю. Ця модель оптимізує взаємодію обладнання під час збору врожаю.

При запуску моделі починається збір зерна комбайном на полі. Як тільки ящик для зберігання зерна заповнено, комбайн зупиняється. Модель очікує від користувача керівних установок. Мабуть, вантажівці в цей момент необхідно під'їхати до комбайна і забрати зерно. Після цього вантажівка під'їжджає до фури і пересипає зерно в неї. Фура також очікує керівних установок. Тут у користувача є два варіанти: або «ганяти» напівпорожню фуру, або заповнити її повністю і потім відвезти зерно на елеватор. Елеватор теж виявляється не невичерпним.

Результатом моделювання буде кількість ходок автомобілів, отже, оптимізація витраченого пального, часу і зарплати робітників. Інтерес викликають параметр «врожайності» поля і можливість його зміни в моделі.

3.15. Модель павільйону метро

Модель, яка імітує рух пасажирів у наземному павільйоні метро (Examples \ Pedestrian Library Tutorial Models \ Subway Entrance - 5 - Statistics.alp.). Перед тим, як пройти до потягів метро, пасажири проходять через турнікети, перевіряючи наявність квитків. Ті пасажири, які не купили квитки заздалегідь, повинні будуть спочатку придбати їх у квитковій касі, яка знаходиться в павільйоні, і тільки потім вони зможуть пройти до поїздів. Ця модель демонструє, як промоделювати потік пішоходів і найпростіші сервіси в AnyLogicTM Pedestrian Library.

Бібліотека Pedestrian Library є високорівневою бібліотекою для моделювання руху пішоходів у фізичному просторі. Вона дає змогу моделювати будівлі, в яких рухаються пішоходи (станції метро, стадіони, музеї), вулиці, парки відпочинку і т. д. У моделях, створених у Pedestrian Library, пішоходи

рухаються в безперервному просторі, реагуючи на різні види перешкод у вигляді стін та інших пішоходів.

Модель містить турнікети для перевірки квитків у пасажирів. Є можливість змінювати інтенсивність надходження пасажирів у модель, щоб перевірити, наскільки модель є стійкою до можливих збільшень навантаження. Є інструменти, які дають змогу спостерігати: скільки пасажирів знаходиться в приміщенні в цей момент часу і скільки часу в середньому пасажир проводить у стінах павільйону.

3.16. Модель пересадкової станції метро

У продовженні вивчення Pedestrian Library (див. завдання 3.15) проаналізувати модель пересадкової станції метрополітену Examples \ Pedestrian Library Tutorial Models \ Subway Entrance - 3 - Escalators.alp. Ця модель має два потоки пасажтрів, які моделюються, пересідають на різні гілки метро. Імітується робота ескалаторів, що піднімають пасажирів на верхній поверх. Експерименти над цією моделлю дають змогу розглянути, яка інтенсивність пасажирів може вплинути на виникнення черги на ескалатор.

3.17. Модель банківського відділення

Це модель простої системи обслуговування, а саме модель банківського відділення (Examples \ Enterprise Library Tutorial Models \ Bank Department 9 - Activity-based costing.alp.). У банківському відділенні знаходяться банкомат і стійки банківських касирів, що дає змогу швидко та ефективно обслуговувати відвідувачів банку. Готівкові операції клієнти банку здійснюють за допомогою банкомату, а більш складні операції, такі, як оплата рахунків, – за допомогою касирів.

Модель дає можливість оцінити витрат на здійснення операцій і проаналізувати, скільки грошей витрачається на обслуговування одного клієнта, яку частину цієї суми становлять накладні витрати на оплату роботи персоналу банку, а яку – на обслуговування відвідувачів.

Елемент управління дає можливість змінювати кількість касирів під час роботи моделі. Відповідно можна зробити висновок про те, скільки службовців необхідно для нормальної роботи банківського відділення при заданій інтенсивності приходу клієнтів.

Enterprise Library надає інструменти для проведення оцінювання витрат на здійснення операцій. За допомогою методу оцінювання витрат на здійснення операцій (activity-based costing, ABC-метод) можна оцінити процес і ефективність операцій, визначити вартість обслуговування / виробництва і вказати можливості для вдосконалення продуктивності та ефективності процесу. За цим методом можно провести кількісне оціню-

вання вартості й продуктивності операцій, ефективності використання ресурсів і вартості об'єктів.

У моделі проводиться облік витрат на здійснення операцій для того, щоб зрозуміти, скільки в середньому коштує обслуговування одного клієнта і які накладні витрати пов'язані з обслуговуванням клієнтів, які очікують своєї черги.

3.18. Модель цеху підприємства

У цьому завданні пропонується проаналізувати роботу цеху підприємства, що здійснює складання виробів (Examples \ Enterprise Library Tutorial Models \ Shop Floor 9 - Equipment downtime.alp.). Цех працює за такою схемою:

- по конвеєру в цех надходять деталі двох типів;
- на обробній станції здійснюється оброблення деталей;
- далі деталі упорядковуються і по різних конвеєрах доставляються до складальної станції;
- на складальній станції здійснюється складання виробів;
- готові вироби вивозяться з цеху по конвеєру.

На складальній станції здійснюється складання виробів з двох деталей: червоної і синьої. Необхідно організувати процес складання, відсортувавши деталі, що надходять на складальну станцію. Крім цього, у моделі можна змінити інтенсивність поставки деталей, ураховуючи можливість виходу станції з ладу.

При проведенні експериментів необхідно зробити висновок про те, наскільки ефективно працює підприємство. Це можна легко зробити за допомогою статистики продуктивності

3.19. Модель відділення офтальмології

У цьому варіанті пропонується проаналізувати модель лікарняного відділення, в якому проводиться процедура офтальмоскопії (Examples \ Enterprise Library Tutorial Models \ Ophthalmology Department 3 - Calling a doctor.alp). Пацієнт, який прибув у відділення, спочатку проходить реєстрацію в приймальному покої. Потім він направляється для проведення процедури в зазначену процедурну кімнату. Якщо всі процедурні кімнати виявляються зайнятими, то пацієнт чекає в приймальному покої, поки якась із кімнат не звільниться. Тільки тоді медсестра відводить пацієнта в кімнату, яка звільнилася, і викликає туди офтальмолога. Лікар оглядає пацієнта за допомогою офтальмоскопа, який він спеціально приносить з кімнати зберігання інструментів. Після проведення процедури лікар відносить офтальмоскоп назад і відправляється в ординаторську, а пацієнт залишає відділення офтальмології.

3.20. Модель протиповітряної оборони

Повний вихідний код моделі Air Defense System включено в дистрибутив AnyLogic 5 (знаходиться в розділі "Приклади" у підрозділі "Miscellaneous").

Літаки противника намагаються проникнути на територію, яка захищається. Їх виявляють засоби розвідки повітряних цілей на основі одержуваних від них двох сигналів, розподілений незалежно. Якщо ціль виявлено, то її обстрілюють і знищують. Якщо ціль не знищено, то вона вражає сторону, що захищається, завдаючи шкоди. Огляд простору системами ППО проводиться рівномірно.

Мета системи – знищити літаки противника. Завдання моделювання – встановити, чи виконує система поставлені вимоги щодо знищення заданої частини літаків. Необхідно вибрати правильні параметри системи, щоб не допустити проникнення літаків (три експерименти повинні підтверджувати ефективність системи ППО для трьох значень інтенсивності появи літаків).

3.21. Гра «Життя»

Клітинний автомат, придуманий англійським математиком Джоном Конвеєм в 1970 році, знаходиться в підрозділі Miscellaneous розділу «Приклади» програми.

Місце дії цієї гри – «Всесвіт». Це розмічена на клітини поверхня або площина – безмежна, обмежена або замкнута (біля межі – нескінченна площина).

Кожна клітина на цій поверхні може перебувати в двох станах: бути "живою" або "мертвою" (порожньою). Клітина має вісім сусідів (оточуючих клітин).

Розподіл живих клітин на початку гри називається першим поколінням. Кожне наступне покоління розраховується на основі попереднього за такими правилами:

- порожня (мертва) клітка, поруч з якою є три живі клітини, оживає;
- якщо у живої клітини є дві або три живі сусідки, то ця клітина продовжує жити; в іншому випадку (якщо сусідів менше двох або більше трьох) клітина вмирає (від «самотності» або від «перенаселеності»).

Гра припиняється, якщо на полі не залишиться жодної "живої" клітини або якщо при черговому кроці жодна з клітин не змінить свого стану (складається стабільна конфігурація). Ці прості правила призводять до величезного розмаїття форм, які можуть виникнути в грі. Експериментатор не бере прямої участі в грі, а лише розставляє або генерує початкову конфігурацію «живих» клітин, які потім взаємодіють згідно з правилами вже без його участі (він є спостерігачем).

Вивчити, які шаблони (варіанти розташування живих клітин у першому поколінні) існують (вікіпедія вам на допомогу), і показати шаблонне розміщення і принтскрін експериментів (фото з екрана монітора).

3.22. Вибір лідера

Розглядається локальна ненадійна мережа (модель знаходиться в Telecom and IT Infrastructure). У мережі комп'ютери обмінюються повідомленнями. Мета протоколу – підтримати лідера (майстра) усіма машинами, що працюють протягом заданого часу. Синхронізація і вибір комп'ютерів базується на постійних і випадкових часових затримках.

На моделі можна вибрати досліджуваний комп'ютер, подивитися характеристики його роботи.

Процесор починає вважати себе лідером залежно від кількості з'єднань з іншими процесорами. Тільки один з «датчиків» вирішує бути лідером, він відправляє оточуючим сигнали, які пригнічують лідерські амбіції всіх інших.

3.23. Модель машин безперервного лиття заготовок

Як ілюстрацію роботи технології імітаційного моделювання пропонується розглянути спрощену модель конвертерного цеху, розміщену у відкритому доступі AnyLogic (Steel Converter Process Model). Демонстраційна модель відтворює алгоритм випадкового вибору крана при транспортуванні ковшів сталевоза. Відомо, що подібний режим є неефективним у деяких конфігураціях цеху. Мета моделювання – аналіз оптимальності вибраного режиму і його оптимізація для того, щоб надалі мінімізувати простої в роботі машин безперервного лиття заготовок (МБЛЗ).

Граничний час простою задається роботою машин безперервного лиття сталі, додаткові обмеження накладаються конвертером і режимом руху ковшів. Ефективність роботи алгоритму руху кранів показано на колірній шкалі: в ідеалі на шкалі не має бути червоних секцій, які вказують на прості МБЛЗ. Масштаб моделі можна легко змінити шляхом включення в модель будь-якої додаткової кількості конвертерів, кранів, ковшів різної місткості, зміни параметрів часу роботи, а також віддаленості один від одного структурних елементів цеху. На сайті розробника є тривимірний варіант моделі, з яким можна ознайомитися.

3.24. Механізм контролю надійності передачі повідомлення

У імітаційній моделі розглядаються PAR-протоколи. Її можна знайти в розділі Telecom and IT Infrastructure. Модуль TCP забезпечує захист від пошкодження, втрати, дублювання й порушення черговості отримання да-

них. Для виконання цих завдань всі октети в потоці даних наскрізним чином пронумеровано в порядку зростання. Тема кожного сегмента містить кількість октетів даних у сегменті й порядковий номер першого октету.

3.25. Моделювання мережі

Моделювання роботи мережі є яскравим і зрозумілим прикладом, де логіка роботи моделі пов'язана з анімацією. Ця модель знаходиться в розділі Telecom and IT Infrastructure.

Мережа складається з базових станцій (багаточастотних УКХ-приймачів), розподілених по всій зоні покриття мережі комутаторів. Стільниковий телефон прослуховує ефір, знаходить сигнал від базової станції і посилає їй унікальний IMSI SIM-карти, а також унікальний IMEI телефону. Оскільки телефони є «мобільними», у моделі вони повинні постійно переміщатися. Базові станції стоять на місці, і єдине, чим можна керувати, це їх потужністю і максимальним рівнем підтримуваних мобільних сервісів. Зверніть увагу на дію людей з мобільними телефонами в разі зменшення потужності станції.

Кількість телефонів у моделі можна змінювати, додаючи їх. Можна змінювати середній час між розмовами. В експериментах потрібно описати результати, які в моделі подано у вигляді графіків.

3.26. Моделювання терміналів із розвантаження танкерів

Приклад транспортування нафти в порти і її розвантаження розглядається у моделі Tanker Unloading з розділу Transportation.

Нафта доставляється танкерами в порт, де здійснюється її розвантаження. За допомогою суднових насосів нафта перекачується спочатку в сховище, а після – у цистерни. У цей час вивантаження нафтопродуктів з танкерів у більшості випадків проводиться за допомогою суднових насосів, що мають високу подачу (500 т / год і більше).

Термінали, які здійснюють розвантаження танкерів, можуть відрізнятися рівнем пропускної здатності, яка залежить від швидкості розвантаження (кт / д), обсягу сховища (кт), кількості цистерн, які можуть завантажуватися одночасно, швидкості заповнення цистерн. У порт надходять танкери із заданою інтенсивністю, регульованими об'ємом нафти і швидкістю. Із залізницею також укладаються договори про кількість цистерн, що поставляються в порт. У договорі вказується швидкість, з якою ці цистерни можуть пересуватися по залізничних коліях. Отже, логіст може зробити висновок про те, коли склад з цистернами зможе доставити нафту до кінцевого складу і повернутися.

Необхідно поставити правильні характеристики роботи порту, щоб не простоювали активи терміналів.

3.27. Моделювання роботи метрополітену

Пропонується оцінити роботу метро Санкт-Петербурга (модель знаходиться у розділі Traffic). Кожен пасажир у моделі створюється із зазначенням місця відправлення і призначення. Під час поїздки він, звичайно ж, може пересідати з однієї гілки на іншу, якщо це йому необхідно. Поява нових пасажирів залежить від часу доби. У години пік (ранок і вечір) пасажирів буде більше, ніж в обідній час. Модель демонструє можливість моделювання оброблення великої кількості заявок, оскільки пасажирів у метро Санкт-Петербурга протягом дня обслуговується близько мільйона.

Потяги також можуть змінювати свою інтенсивність появи в моделі. Таким чином, в експериментах можна спробувати вплинути на цю величину. Можна виконати моделювання збільшення кількості станцій у метро або появи нової гілки.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Струтинський, В. Б. Математичне моделювання процесів і систем : підруч. з грифом МОН України / В. Б. Струтинський, А. М. Гуржій, В. С. Кривцов. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т", 2011. – 672 с.
2. Томашевський, В. М. Моделювання систем : підруч. для студ. з грифом МОН України / В. М. Томашевський. - Київ : Вид. група BHV, 2005. – 352 с.
3. Прохоров, А. В. Моделирование сложных систем : учеб. пособие. Ч. 2 / А. В. Прохоров, К. В. Головань. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2005. – 57 с.
4. Прохоров, А. В. Моделирование сложных систем : учеб. пособие по лаб. практикуму. Ч. 1 / А. В. Прохоров, А. С. Садовничий. – Харьков: Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2003. – 63 с.
5. Моделирование распределенных систем обработки данных [Текст]: учеб. пособие по лаб. практикуму / Е. А. Дружинин, О. К. Погудина, И. Н. Бабак, С. А. Коба. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2015. – 56 с.
6. Стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 122 «Комп'ютерні науки та інформаційні технології» [Електронний ресурс] / Т. В. Ковалюк, О. І. Михальов, М. В. Новожилова та ін. - Режим доступу : http://mon.gov.ua/content/122-komp -texnologiyi.docx, свободный. − Загол. з экрана.
7. Галузевий стандарт вищої освіти України з напряму підготовки 6.050101 «Комп'ютерні науки» : зб. нормат. док. вищ. освіти / О. А. Павлов, Т. В. Ковалюк, А. І. Петренко та ін. – Київ : Вид. група BHV, 2011. − 85 с.
8. Гліненко, Л. К. Основи моделювання технічних систем : навч. посіб. з грифом МОН України / Л. К. Гліненко, О. Г. Сухоносов. - Львів : Бескид Біт, 2003. – 176 с.
9. Дубов, Ю. А. Многокритериальные модели формирования и выбора вариантов систем / Ю. А. Дубов, С. И. Травкин, В. Н. Якимец. – М. : Наука, 1986. - 296 с.
10. Зарубин, В. С. Математическое моделирование в технике : учеб. для студ. втузов / В. С. Зарубин ; под ред. В. С. Зарубина, А. П. Крищенко. – М. : МГТУ им. Н. Э. Баумана, 2001. – 496 с.
11. Моделювання інформаційно-вимірювальних систем: консп. лекцій для студ. / уклад.: І. В. Кравченко. – Київ : НТУУ «КПІ», 2017. – 79 с.
12. Замятина, О. М. Моделирование систем: учеб. пособие / О. М. Замятина. – Томск: Изд-во ТПУ, 2009. – 204 с.
13. Joeran, Beel Utilizing Mind-Maps for Information Retrieval and User Modelling \ Joeran Beel, Stefan Langer, Marcel Genzmehr, Bela Gipp

\\International Conference on User Modeling, Adaptation, and Personalization, UMAP, 2014. – P. 301 – 313.

14. Осоргин, А. Е. AnyLogic 6: лаб. практикум / А. Е. Осоргин. – М. : МГТУ им. Н. Э. Баумана, 2003. – 192 с.

15. Мезенцев, К. Н. Моделирование систем в среде AnyLogic 6 / К. Н. Мезенцев. – М. : МГТУ им. Н. Э. Баумана, 2005. – 232 с.

16. Башарин, Г. П. Анализ очередей в вычислительных сетях: Теория и методы расчета / Г. П. Башарин. – М. : Наука, 1986. - 126 с.

17. Kendall, D. G. Some Recent Work and Further Problems in the Theory of Queues // Теория вероятности и ее применение. – 1964. – Т. IX, № 1. – С. 3 – 15.

18. Киселева, М. В. Имитационное моделирование систем в среде Anylogic / М. В. Киселева. - Екатеренбург: УГТУ, 2009. – 492 с.

19. Маликов, Р. Ф. Практикум по имитационному моделированию сложных систем в среде AnyLogic 6 / Р. Ф. Маликов. – М. : ФГБОУ, 2013. – 320 с.

20. Скатков, А. В. Версійне моделювання структурно неоднорідних транспортно-виробничих систем / А. В. Скатков, А. В. Тарасова // Радіоелектронні й комп'ютерні системи. – 2012. – № 7. – С. 247 – 252.

21. Карпов, Ю. П. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 / Ю. П. Карпов. – СПб.: БХВ Питербург, 2005. – 250 с.

22. ARIS Express [Електронний ресурс]. – Режим доступу: http://www.ariscommunity.com/aris-express -

23. ITHINK [Електронний ресурс]. – Режим доступу: https://www.iseesystems.com/store/products/ithink.aspx

24. PowerSim Studio [Електронний ресурс]. – Режим доступу: http://www.powersim.com/

25. Extend [Електронний ресурс]. – Режим доступу: https://www.extendsim.com/

26. GPSS/H Software Products [Електронний ресурс]. – Режим доступу: http://www.wolverinesoftware.com/GPSSHProducts.htm

27. Simprocess [Електронний ресурс]. – Режим доступу: http://simprocess.com/

28. AllFusion Process Modeler [Електронний ресурс]. – Режим доступу: https://www.ca.com/us.html

29. ProcessModel [Електронний ресурс]. – Режим доступу: https://www.processmodel.com/

30. AnyLogic [Електронний ресурс]. – Режим доступу: https://www.anylogic.ru/

31. Witness [Електронний ресурс]. – Режим доступу: https://www.lanner.com/technology/witness-simulation-software.htm

32. Arena [Електронний ресурс]. – Режим доступу: https://www.arenasimulation.com/

Навчальне видання

Погудіна Ольга Костянтинівна
Погудін Андрій Володимирович
Губін Сергій Вікторович

МОДЕЛЮВАННЯ СИСТЕМ

Редактор А. М. Ємленінова

Зв. план, 2018
Підписано до друку 03.04.2018
Формат 60×84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 5,8. Обл.-вид. арк. 6,5. Наклад 50 пр.
Замовлення 154. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001