інформатика

В. М. Томашевський

МОДЕЛЮВАННЯ СИСТЕМ

ПІДРУЧНИК

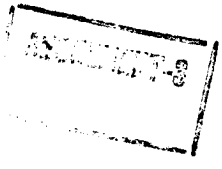
ДЛЯ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ

bhv ПИТЕР

В. М. ТОМАШЕВСЬКИЙ

МОДЕЛЮВАННЯ СИСТЕМ

Затверджено Міністерством освіти і науки України як підручник для студентів вищих навчальних закладів, які навчаються за напрямами «Комп'ютерні науки», «Комп'ютеризовані системи, автоматика і управління», «Комп'ютерна інженерія», «Прикладна математика»



Серія «ІНФОРМАТИКА»
За загальною редакцією академіка НАН України
М. З. Згуровського

Київ
Видавнича група BHV
2005

ББК 32.973-018я73
Т-56
УДК 004.94(075.8)

Рецензенти: В. М. Михайленко, доктор технічних наук, професор, директор Українського науково-дослідного інституту системних досліджень, завідувач кафедри математичних досліджень Європейського університету фінансів, інформаційних систем менеджменту і бізнесу;

А. Ф. Верлань, доктор технічних наук, професор, член-кореспондент Академії педагогічних наук України, завідувач відділу Інституту проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України.

Гриф надано Міністерством освіти і науки України, лист № 14/18.2-1566 від 08.07.2004 р.

Томашевський В. М.

Т-56 Моделювання систем. — К.: Видавнича група BHV, 2005. — 352 с.: іл.
ISBN 966-552-120-9

У підручнику викладено основи теорії моделювання і базові принципи моделювання складних технічних і економічних систем. Докладно описуються моделі систем масового обслуговування, мережі Петрі та програмні генератори випадкових величин і процесів. Особливу увагу приділено технології імітаційного моделювання, зокрема методології проектування і програмної реалізації імітаційних моделей, плануванню і проведенню імітаційних експериментів, обробці та інтерпретації результатів моделювання. Розглянуто також програмні засоби і приклади моделювання виробничих та обчислювальних систем, бізнес-процесів.

Підручник призначено для студентів, які навчаються за напрямом «Комп'ютерні науки» і вивчають сучасні інформаційні технології в рамках дисципліни «Моделювання систем». Книжка також може бути корисна для викладачів зазначеної дисципліни.

ББК 32.973-018я73

Серія підручників «Інформатика» виходить у світ за підтримки видавництв «Издательский дом "Питер"», м. Санкт-Петербург та «Освітня книга», м. Київ.

ISBN 966-552-120-9

Стислий зміст

Зміст

Передмова

Моделювання як одну з найважливіших категорій процесу пізнання неможливо відокремити від розвитку людства. Ще з дитинства людина пізнає світ, спочатку через іграшки та ігри, і відображає, або моделює, дійсність. Згадаємо комп'ютерні ігри, в яких ми, сидячи в літаку або космічному кораблі, здійснюємо політ так, нібито дійсно знаходимося там. З роками людина використовує більш складні моделі, що дають можливість «програвати» будь-які життєві та виробничі ситуації й отримувати такі рішення, що дозволяють знайти найкращий спосіб вирішення проблеми. У таких випадках є можливість аналізувати за допомогою моделі будь-які ситуації, включаючи ті, за яких реальна система вийшла б із ладу. Це дозволяє моделювати катастрофи, рідкісні випадки, та навіть такі явища і процеси, яких не існує насправді, тобто віртуальну реальність.

Методи комп'ютерного моделювання широко застосовуються в усіх сферах діяльності людини – від конструювання моделей технічних, технологічних та організаційних систем до вирішення проблем розвитку людства та всесвіту. Класичними об'єктами моделювання є інформаційні, виробничі, транспортні та інші логістичні системи, які в більшості випадків застосовуються для розв'язання задач проектування, реконструкції та довгострокового планування, а також використання моделей у контурі керування, тобто в реальному масштабі часу. Найважливішим завданням моделювання є оцінка показників функціонування таких систем.

Коли слід використовувати комп'ютерне моделювання? Завжди, як тільки ми ставимо запитання: «Що буде, якщо ...?», тобто для прийняття рішень. У розвинених країнах перед інвестуванням коштів у будь-який проект можливості його реалізації перевіряються на імітаційних моделях. Практично всі транснаціональні компанії мають моделі розвитку виробництва, більш того, вони вкладають значні кошти у дослідження цих моделей. Наприклад, щодо автомобільної промисловості Німеччини існує рішення до 2005 року приймати до розгляду технічну документацію тільки за умови її відповідності концепції Digitale Fabrik (комп'ютерне виробництво). Важливу роль у цій концепції відіграють 3D-моделі всіх елементів виробничого процесу, що замінюють собою звичайні CAD-креслення. У вигляді 3D-моделей повинні зображуватись усі засоби виробництва: устаткування і робочі місця, окремі цехи і підприємство в цілому, а також вироблена продукція – готові вироби з їх докладною технічною документацією. Зрозуміло також, що демонстрація будь-яких динамічних процесів можлива лише за умови, що ними керуватимуть відповідні імітаційні моделі. Треба чітко уявляти собі, що поряд із традиційними для імітаційного моделювання моделями процесів із дискретними подіями існують кінематичні 3D-моделі устаткування і робочих місць, ергономічні 3D-моделі, моделі типу Digital MockUp тощо.

Моделювання як технологія розв'язання задач усередині специфічного середовища широко застосовується під час аналізу і проектування інформаційних

систем для перевірки вимог до їх ефективності, до використаних ресурсів і оцінки пропускної спроможності систем. Однак розробка і застосування імітаційних моделей інформаційних систем – це не прості завдання. Етап формулювання абстрактної моделі та етап конструювання моделі часто включають тривалі й дорогі процедури. Абстрактна модель інформаційної системи звичайно створюється фахівцем із моделювання, який може отримувати знання у потрібній галузі від проектувальників і аналітиків. Модель може мати математичний характер (наприклад, системи формування черг, ланцюги Маркова або мережі Петрі), але для того щоб вона підлягала аналізу, навіть за допомогою комп'ютера, при її формулюванні роблять деякі узагальнення. Програмна реалізація моделі потім здійснюється фахівцями з моделювання, які можуть використовувати універсальну мову програмування (типу C++ або Java) або спеціалізовані засоби моделювання (такі як GPSS або iThink). Для цього часто залучаються програмісти, які є проміжною ланкою між аналітиком і людиною, що приймає рішення. Наявність такої ланки може призводити до появи помилок і неточностей не тільки під час побудови моделі, але й під час програмування.

Моделювання – складний процес, що потребує багато часу, незважаючи на те, йде мова про окремого фахівця з моделювання чи цілої групи фахівців, впродовж роботи якої потрібні постійний зв'язок і координація. Зазначені причини виправдовують зусилля, докладені для розробки методів, що допомагають прискорити процес моделювання шляхом автоматизації деяких процесів. Сучасні програмні засоби моделювання використовують графічний інтерфейс і дво- або тримірну анімацію, що значно полегшує сприйняття результатів моделювання неспеціалістом.

Програми реалізації моделей взагалі складно писати й налагоджувати. Для того щоб перевірити правильність і достовірність імітаційної моделі та її відповідність цілям моделювання, необхідно мати вичерпну інформацію щодо області застосування системи, методології моделювання і мови програмування. Таку роботу зазвичай виконує експерт. У якісній моделі повинні враховуватися всі можливі варіанти вихідних даних, і починати моделювання можна лише, отримавши позитивні результати. Після огляду числових результатів моделювання може виникнути потреба у внесенні деяких змін в абстрактну модель і (або) програмну реалізацію моделі, що може призвести до повторного виконання деяких або всіх операцій на різних етапах моделювання. Таким чином, жоден серйозний проект з моделювання не може бути успішно реалізований без участі експерта. Великі за обсягом моделі створює, як правило, команда розробників, і хоча б один з її членів має виконувати при цьому роль експерта. Експерт повинен:

- володіти базовими інженерними знаннями, необхідними для розуміння принципів функціонування визначених класів систем;
- володіти методами системного аналізу і керування проектами, необхідними для коректної постановки задачі моделювання і організації робіт з реалізації й використання моделей;
- володіти методами математичного та імітаційного моделювання незалежно від того, які програмні засоби моделювання використовуються;

- знати і вміти застосовувати одну або декілька імітаційних систем і мов програмування;
- бути обізнаним із сучасними інформаційними технологіями, що забезпечують інтеграцію моделей у системи проектування, планування і керування;
- бути спроможним приймати рішення за результатами моделювання;
- знати основні класи математичних моделей і методи моделювання систем, а також принципи побудови імітаційних моделей процесів функціонування систем, методи та етапи їх формалізації та алгоритмізації;
- вміти вибирати та використовувати методи математичного моделювання при проектуванні та експлуатації складних систем управління, розробляти схеми алгоритмів для імітаційного моделювання технічних, технологічних, організаційних, інформаційних систем та їх об'єктів, реалізовувати моделюючі програми на комп'ютері;
- мати уявлення про сучасний стан і перспективи розвитку методів моделювання в галузі інформаційних технологій, систем управління та систем обробки інформації з використанням сучасних програмних систем, таких як програмні генератори, інтерактивні, інтелектуальні та візуальні системи моделювання.

Для того щоб стати досвідченим експертом і професіоналом, необхідно також мати досвід роботи в проектах з моделювання.

З 1952 року існує всесвітнє добровільне товариство міжнародного комп'ютерного моделювання – SCS (www.scs.org), основними завданнями якого є вивчення, розповсюдження, використання й удосконалення методів комп'ютерного моделювання для вирішення реальних проблем, що існують у світі. До SCS входять професіонали, діяльність яких пов'язана з розробленням методології та застосуванням сучасних технологій і методів моделювання. Регіональні ради SCS існують у США, Канаді, країнах Європи (www.scs-europe.net), включаючи Східну Європу, в Китаї, Мексиці та інших країнах.

Щороку SCS проводить конференції з проблем моделювання, публікує доповіді та випускає журнали (www.scsorg/pubs/pubsinfo.html). В Європі існує федерація європейських товариств моделювання – EUROSIM (www.eurosim.info), товариство моделювання та технології імітації – EUROSIS (http://biomath.rug.ac.be/~eurosis/index.html), а також інститути науки моделювання – McLeod. Комп'ютерне моделювання активно застосовується у дослідницьких центрах в усьому світі. Тільки у Великій Британії існує близько десяти груп дослідників в університетах, що працюють у цій галузі,– в Лондонській школі економіки, Імперіал-коледжі, Університеті Варвік, Університеті Ланкастера, Саутамптонському університеті тощо. Американське і європейські товариства моделювання регулярно проводять конференції та публікують їх матеріали.

Добре відомі праці з імітаційного моделювання Р. Шеннона, Дж. Шрайбера, Дж. Клейнена, А. Прицкера, надруковані російською мовою. Протягом більш ніж 15 років не було перекладено жодної книжки іноземного автора відповідної тематики ні в Росії, ні в Україні, хоча тільки в США щорічно видається або перевидається чимало таких книжок. Серед останніх слід відзначити монографії, які

широко використовуються в учбовому процесі багатьох університетів: A. M. Law, W. D. Kelton. Simulation Modeling and Analysis (у 2004 році перекладена російською мовою); J. Banks, J. S. Carson, B. L. Nelson. Discrete-Event System Simulation.

На жаль, активне застосування методів імітаційного моделювання за кордоном не викликало поки що значного його поширення у нас в країні, незважаючи на проведення Інститутом кібернетики НАН України фундаментальних робіт у цій галузі. Пояснити те, що відбувається, мабуть, можна двома причинами: по-перше, пануванням у певні часи принципу витратної економіки, за якої імітаційні моделі були не потрібні; по-друге, необхідністю перебудови стереотипу мислення у процесі розробки імітаційних моделей, який суттєво відрізняється від процесу проектування традиційних програмних засобів для автоматизації систем управління.

У зв'язку з розвитком ринкової економіки та переходом до ринкових моделей ситуація почала змінюватись. Це підтверджує і поява в мережі Інтернет за останні два роки портала www.simulation.org.ua в Україні, www.gpss.ru – в Росії та сайта www.gpss-forum.narod.ru в Росії.

Призначення цієї книги – подати всебічне і сучасне трактування всіх важливих аспектів моделювання, включаючи формальні моделі систем масового обслуговування, системної динаміки та мереж Петрі, технологію і програмне забезпечення моделювання, перевірку достовірності та правильності моделей, методи моделювання випадкових чисел, величин і процесів, планування експериментів й аналіз результатів моделювання з наступним прийняттям рішень.

У підручнику узагальнено більш ніж двадцятип'ятирічний досвід роботи автора у галузі імітаційного моделювання та практичного використання систем АЛСИМ, СТАМ-КЛАСС, GPSS, CSS для трьох поколінь комп'ютерів, починаючи з БЕСМ, ЄС ЕОМ та закінчуючи персональними комп'ютерами, а також досвід розробки власних систем моделювання ICIM'95, ISS 2000 і викладання дисципліни «Моделювання систем» на кафедрі автоматизованих систем обробки інформації та управління в Національному технічному університеті України «Київський політехнічний інститут».

З усіх видів моделювання – а це перш за все математичне і графічне – в підручнику основна увага приділяється імітаційному моделюванню. Огляд науково-дослідницьких робіт показує, що імітаційне моделювання є чи не найпопулярнішим, за використанням на практиці його випереджають лише методи математичного програмування. Головна цінність імітаційного моделювання полягає в тому, що в основу його покладена методологія системного аналізу. Вона дозволяє досліджувати проектовану або аналізовану систему методами операційного аналізу, який включає такі взаємопов'язані етапи: змістовна постановка задачі, розробка концептуальної моделі, розробка та програмна реалізація імітаційної моделі, перевірка адекватності моделі та оцінка точності результатів моделювання, планування і проведення експериментів та прийняття рішень. Завдяки цьому можна застосовувати імітаційне моделювання як універсальний підхід під час прийняття рішень в умовах невизначеності та врахування у моделях тих факторів, які важко формалізувати, а також використовувати основні принципи системного підходу для виконання практичних завдань.

Упровадження імітаційних моделей обробки даних висвітлило низку проблем, з якими доводиться мати справу як розробникам моделей, так і користувачам. Іноді навіть кваліфіковані фахівці в галузі традиційного програмування не можуть засвоїти прийоми роботи з системами моделювання через нерозуміння прихованого механізму роботи моделюючої програми. З подібними серйозними труднощами стикаються і студенти на початковому етапі вивчення методів імітаційного моделювання, а потім і під час переходу від розробки програм на традиційних мовах програмування до розробки імітаційних програм. Тому було поставлено за мету, базуючись на системному підході, розкрити суть імітаційного моделювання ззовні (з боку користувача) та зсередини (з боку розробника моделей).

У підручнику зроблена спроба відповісти на традиційне, дискусійне питання щодо вибору засобів програмування для реалізації імітаційної моделі. Під час побудови складних імітаційних моделей не може йтися про алгоритмічні процедурні мови як основу моделі, бо в цьому разі доводиться відтворювати весь прихований механізм мов моделювання. Останні служать для навчальних та «іграшкових» моделей, що ілюструють можливості імітаційного моделювання. Для складних моделей використовуються спеціалізовані засоби моделювання, які дозволяють автоматизувати процеси створення моделі. Особлива увага приділяється мові дискретно-подійного імітаційного моделювання GPSS, яку, незважаючи на її солідний вік (понад 40 років), досі застосовують для програмних реалізацій моделей. Ця мова проста й ефективна при розробленні більшості простих моделей, навчитися будувати які можна за дуже короткий час. Розглядаючи принципи побудови алгоритмів для реалізації блоків і керуючої програми моделювання мови GPSS, можна легко зрозуміти, яким чином будуються складні імітаційні системи. Тому в багатьох розділах книги розглядаються приклади, пов'язані з цією мовою. Більш того, аналіз застосування мов моделювання для проведення практичних, лабораторних занять і курсового проектування у вищих навчальних закладах України і країн СНД (www.gpss.ru) підтверджує популярність мови GPSS.

Враховуючи те, що моделювання систем – прикладна наука, поданий у книжці матеріал має практичну спрямованість, і наведені математичні формулювання і докази не завжди є строгими. Особливо це стосується теорії масового обслуговування, яка викладається з позицій операційного аналізу.

Імітаційне моделювання неперервних систем і пов'язаних з ними комбінованих систем моделювання у книжці не висвітлене в повній мірі. Це пояснюється тим, що в галузі інформатики, економіки, обробки інформації та автоматизованих систем управління неперервні системи використовуються тільки на рівні управління потоками (матеріальними, фінансовими та ін.). В останніх розділах посібника описується технологія автоматизації проектування імітаційних моделей та проведення експериментів, а також прискорення процесу моделювання, наведені приклади побудови складних імітаційних проектів.

У книжці розглянуто сучасний стан засобів моделювання та тенденції їх розвитку; серед них на особливу увагу заслуговують засоби автоматизації створення моделей, а саме візуальне середовище моделювання, яке дозволяє будувати імітаційні моделі за принципом «що бачу, те і відображаю у моделі».

Для успішного засвоєння матеріалу книги необхідно мати базові знання з теорії ймовірностей та математичної статистики, основ дискретної математики, лінійної алгебри або теорії матриць і програмування.

Книга призначена для студентів, які навчаються за програмою бакалаврів з напряму «Комп'ютерні науки», магістрам, що навчаються за спеціальностями «Інформаційні управляючі системи та технології», «Прикладна математика», «Інформаційні системи у менеджменті», крім того, вона буде корисною аспірантам і усім, хто цікавиться проблемами імітаційного і комп'ютерного моделювання та його застосуванням на практиці.

Від видавництва

Ваші зауваження, пропозиції та запитання надсилайте за адресою електронної пошти pg@bhv.kiev.ua та за адресою http://www.osvita.info. На цьому ж сайті ви познайомитеся з детальною інформацією про інші видання серії «Інформатика».

Інформацію про книжки Видавничої групи BHV ви можете знайти на сайті http://www.bhv.kiev.ua.

Розділ 1

Загальні положення та визначення

- Визначення моделі та системи
- Взаємозв'язок моделі та системи
- Класифікація моделей і види моделювання
- Принципи і методи побудови моделей
- Технологія моделювання

Моделювання — це спосіб дослідження будь-яких явищ, процесів або об'єктів шляхом побудови та аналізу їх моделей. У широкому розумінні моделювання є однією з основних категорій теорії пізнання і чи не єдиним науково обґрунтованим методом наукових досліджень систем і процесів будь-якої природи в багатьох сферах людської діяльності.

На сьогоднішній день моделюванню приділяється велика увага. Невипадково найпотужніший у світі суперкомп'ютер NEC Vector SX6 (Earth-Simulator), за даними останньої версії рейтингу TOP500 (http://www.top500.org), встановлено в Центрі моделювання Землі в Йокогамі (Японія). Цей комп'ютер призначено для моделювання основних властивостей складових кліматичної системи Землі: атмосфери, океану, кріосфери, поверхні суші і біосфери, а також зовнішніх і внутрішніх факторів у системі, яка визначає глобальний клімат і його зміни.

У цьому розділі розглядаються основні відомості загальної теорії систем і моделювання.

1.1. Поняття системи

Основними поняттями в теорії і практиці моделювання об'єктів, процесів і явищ є «система» та «модель».

У перекладі з грецької «systema» — ціле, яке складається із частин; об'єднання. Термін «система» існує вже більш ніж два тисячоліття, проте різні дослідники визначають його по-різному. На сьогодні існує понад 500 визначень терміну «система». Однак, використовуючи будь-яке з них, у першу чергу потрібно мати на увазі ті завдання, які ставить перед собою дослідник. Системою може бути і один комп'ютер, і автоматизована лінія або технологічний процес, в яких комп'ютер

є лише одним із компонентів, і все підприємство або кілька різних підприємств, які функціонують як єдина система в одній галузі промисловості. Те, що один дослідник визначає як систему, для іншого може бути лише компонентом більш складної системи.

Для всіх визначень системи загальним є те, що система – це цілісний комплекс взаємопов'язаних елементів, який має певну структуру і взаємодіє із зовнішнім середовищем. *Структура* системи – це організована сукупність зв'язків між її елементами. Під таким зв'язком розуміють можливість впливу одного елемента системи на інший. *Середовище* – це сукупність елементів зовнішнього світу, які не входять до складу системи, але впливають на її поведінку або властивості. Система є *відкритою*, якщо існує зовнішнє середовище, яке впливає на систему, і *закритою*, якщо воно відсутнє або з огляду на мету досліджень не враховується.

Одне з перших визначень системи (1950 рік) належить американському біологу Л. фон Берталанфі, згідно з яким система складається з деякої кількості взаємопов'язаних елементів. Оскільки між елементами системи існують певні взаємозв'язки, то мають бути структурні відношення. Таким чином, система – це щось більше, ніж сукупність елементів. Аналізуючи систему, потрібно враховувати оцінку системного (синергетичного) ефекту. Властивості системи відмінні від властивостей її елементів, і залежно від властивостей, якими цікавляться дослідники, та ж сама сукупність елементів може бути системою або ні.

Багато дослідників визначають систему як *цілеспрямовану* множину взаємопов'язаних елементів будь-якої природи. Згідно з цим визначенням система функціонує для досягнення деякої мети. Це визначення є правильним для соціологічних і технічних систем, але не підходить для систем навколишньої природи (наприклад, біологічних), мета функціонування яких не завжди відома.

Одне з важливих визначень системи пов'язане з абстрактною теорією систем, у рамках якої, на відміну від інших рівнів опису систем [28], використовуються такі рівні абстрактного опису:

- символічний, або лінгвістичний;
- теоретико-множинний;
- абстрактно-алгебричний;
- топологічний;
- логіко-математичний;
- теоретико-інформаційний;
- динамічний;
- евристичний.

Найвищий рівень абстрактного опису систем – *лінгвістичний*; грунтуючись на ньому, можна одержати всі інші рівні. На цьому рівні вводиться поняття предметної області, для опису якої застосовуються алгебричні моделі, з якими пов'язана деяка мова. Для опису предметної області цією мовою використовуються два рівні формальних мов [28], за допомогою яких будують логіко-алгебричну модель предметної області. На цій моделі підтверджуються методи дослідження за допо-

могою формального апарату, яким можуть бути теорії, побудовані у вигляді істинних висловлювань з усієї множини висловлювань.

Таким чином, система – це окремий випадок теорії, описаний формальною мовою, яка уточнюється до мови об'єктів. Для визначення деякого поняття використовують певні символи (алфавіт) і встановлюють правила оперування ними. Сукупність символів і правил користування ними утворює абстрактну мову. Поняття, висловлене абстрактною мовою [14], означає будь-яке речення (формулу), побудоване за граматичними правилами цієї мови. Припускають, що таке речення містить варійовані змінні, так звані *конституенти*, які, маючи тільки певні значення, роблять дане висловлювання істинним.

Якщо існує множина висловлювань G, але лише V із них істинні, то вважають, що має місце теорія L відносно множин G. Якщо припустити, що конституенти в цих висловлюваннях є формально визначеними величинами, то такі висловлювання називаються правильними. Тоді, за визначенням М. Месаровича, система – це множина правильних висловлювань. Всі висловлювання поділяються на два типи: *терми*, які вказують на предмети (об'єкти), і *функтори*, які визначають відношення між термами (об'єктами). Використання термів і функторів дає змогу показати, як, базуючись на лінгвістичному рівні, можна утворити інші рівні абстрактного опису системи.

Наприклад, за допомогою термів і функторів можна показати, як із лінгвістичного рівня абстрактного опису системи виникає теоретико-множинний, якщо вважати, що терми – це множини X_S, за допомогою яких перелічують елементи або, інакше, підсистеми досліджуваних систем, а функтори встановлюють характер відношень між задіяними в описі множинами.

Під час подальшого викладення змісту цієї книги будемо користуватись *теоретико-множинним* визначенням системи (А. Холл і Р. Фейджін та Ф. Фейджін), згідно з яким *система* – це множина об'єктів, між якими існують певні відношення, та їх атрибути. Під *об'єктами* розуміють компоненти системи. Це, наприклад, підсистеми (тобто може існувати ієрархія підсистем) або окремі об'єкти системи. *Атрибути* – це властивості об'єктів. *Відношення* задають певний закон, за яким визначається деяке відображення в одній і тій самій множині об'єктів. За цим визначенням поняття *множина* та *елемент* є аксіоматичними.

Таким чином, система S задається парою елементів:

$$S = \langle X_s, R_s \rangle,$$

де X_s, R_s – множини відповідно елементів і відношень між ними. Відношення визначають взаємодію між об'єктами. У загальному випадку n – відношення R^n у множинах $X_1, X_2, ..., X_n$ є деякою підмножиною декартового добутку[1] $X_1 \times X_2 \times ... \times X_n$, який зіставлено з n-вимірних наборів (кортежів) виду $(x_1, x_2, ..., x_n)$, де $x_i \in X_i$, $i = 1, 2, ..., n$.

[1] Декартовим добутком множин $A \times B$ називається сукупність будь-яких пар виду $\langle a, b \rangle$, така що $A \times B = (\langle a, b \rangle \mid a \in A, b \in B)$.

Якщо відношення R^n в окремому випадку задається, наприклад, деякою функцією, що визначає зв'язок між певним елементом $x \in X$ і певною підмножиною Y, то $f : X \to Y$, тобто вважаємо, що функція f перетворює значення із множини X у значення підмножини Y. Для функції f множина X – це область визначення, а підмножина Y – область значень функції. Функцію f можна подати як множину впорядкованих пар елементів (x, y).

Що стосується атрибутів системи, то вони подібні до функцій, визначених у підмножині об'єктів. Відмінність атрибутів від функцій полягає в тому, що два різних атрибути з погляду на поняття функції можуть бути однаковими. Атрибут A задається парою елементів – (i, f), де i – ім'я атрибута, а f – функція, визначена на підмножині об'єктів. У динамічних об'єктів атрибут також може бути функцією від часу t.

Наприклад, у разі дослідження пропускної здатності ділянок доріг об'єктами системи можуть бути перехрестя, розв'язки, поворот і прямолінійні ділянки доріг (статичні об'єкти) та автомобілі (динамічні об'єкти). Властивості (атрибути) динамічних об'єктів, на відміну від властивостей статичних, змінюються в часі. Наприклад, гальмовий шлях автомобіля змінюється залежно від швидкості руху та погодних умов, а прискорення може бути додатним (під час розгону) або від'ємним (під час гальмування). Відношення в цій системі задаються згідно з правилами дорожнього руху.

Вивчаючи систему більш глибоко, усвідомлюємо, що вона може складатися з підсистем або бути однією з елементів більшої системи, тобто може існувати ієрархія систем. Наприклад, двигун є підсистемою автомобіля, який у свою чергу є підсистемою транспортного потоку магістралі.

На теоретико-множинному рівні абстрактного опису системи можна отримувати досить загальні відомості про реальні системи, а для конкретніших цілей потрібні інші моделі, які давали б змогу більш детально аналізувати різні властивості реальних систем. Для цього потрібні нижчі рівні абстрактного опису систем, які є окремими випадками опису теоретико-множинного рівня. Так, якщо зв'язки між елементами розглядуваних множин установлюються за допомогою деяких однозначних функцій, що відображають елементи множини в саму відправну множину, то має місце *абстрактно-алгебричний* рівень опису систем. У таких випадках вважають, що між елементами множин встановлено нульарні, унарні, бінарні, тернарні та інші відношення.

Якщо ж на множинах, які розглядаються, визначено деякі багатозначні функції, то мають місце *топологічні абстрактні* моделі, записані мовою загальної топології або її гілок, які називаються гомологічною топологією, алгебричною топологією тощо. Вибір потрібного рівня абстрактного опису під час вивчення тієї або іншої реальної системи є завжди найвідповідальнішим і найважчим кроком у теоретико-системних побудовах. Цей процес майже не піддається формалізації і багато в чому залежить від досвіду та знань дослідника, його фахової належності, цілей дослідження тощо.

Можна показати, як від систем із топологічним рівнем опису перейти до узагальнених динамічних. Щоб дати строге математичне визначення поняттю *динамічна система*, її наділяють властивістю мати «входи» й «виходи», тобто визначають як

структурований об'єкт, куди в певні моменти часу можна вводити речовину, енергію та інформацію, а в інші моменти – виводити їх. Динамічні системи можна зобразити і як системи, де процеси відбуваються неперервно, і як системи, в яких усі процеси протікають лише в дискретні моменти часу.

Інші абстрактні рівні опису систем пов'язані з розвитком інформаційних і програмних систем, а також систем штучного інтелекту.

Елементи системи і зв'язки між ними в різних випадках можуть мати різну природу (фізичну, інформаційну, технологічну, біологічну, соціальну), тому аналізом систем займаються представники різних галузей науки і техніки. Науковий напрям під назвою загальна теорія систем, який з'явився наприкінці 50-х – на початку 60-х років XX сторіччя, пов'язаний із розробленням сукупності філософських, методологічних, наукових і прикладних методів аналізу та синтезу систем довільної природи. Ця теорія є загальною, оскільки має дедуктивний характер, об'єднує інші теорії, а саме: теорії керування, самоорганізації, навчання тощо, і розроблена для вивчення поведінки абстрактних систем. Основне її призначення – пояснити, яким чином з окремих елементів утворюється складна єдність цілого, нова сутність. Загальна теорія систем тісно пов'язана з формальною і є певною мірою математичною. Основна процедура теорії систем і системного аналізу – *побудова моделі* системи, яка відображала б усі фактори, взаємозв'язки і реальні ситуації. Займаються цим спеціалісти із системного аналізу – системотехніки або системні аналітики.

1.2. Поняття моделі

Науковою основою моделювання як методу пізнання і дослідження різних об'єктів і процесів є *теорія подібності*, в якій головним є поняття *аналогії*, тобто схожості об'єктів за деякими ознаками. Подібні об'єкти називаються аналогами. Аналогія між об'єктами може встановлюватись за якісними і (або) кількісними ознаками.

Основним видом кількісної аналогії є *математична подібність*, коли об'єкти описуються за допомогою рівнянь і функцій. Функції та незалежні змінні називаються схожими, якщо вони співпадають з точністю до деякої константи. Окремими видами математичної подібності є *геометрична подібність*, яка встановлює подібність геометричних образів, і *часова*, що визначає подібність функції часу, для якої константа часу (масштаб) показує, в яких відношеннях знаходяться параметри функцій, такі як період, часова затримка тощо.

Іншим видом кількісної аналогії, який слід відзначити, є *фізична подібність*. Критерії фізичної подібності можна отримати, не маючи математичного опису об'єктів, наприклад на основі значень фізичних параметрів, які характеризують досліджуваний процес у натурі й на моделі. За типом процесу розрізняють види подібності, для яких розроблено відповідні критерії – гідравлічні, електричні, аеродинамічні та ін.

Вивчення переходу від властивостей реальних об'єктів до властивостей системи є найважливішим завданням теорії систем. У загальній теорії систем визнається об'єктивність їх існування. Згідно з цією теорією, якщо реально існують

взаємозв'язки між об'єктами, то існують і системи, які їм відповідають. Ця теорія ґрунтується на постулаті функціонально-структурного ізоморфізму об'єктів і явищ природи.

Якщо структура однієї системи і зовнішні функції її елементів ізоморфні структурі іншої системи і зовнішнім функціям її елементів, то зовнішні властивості цих систем не розрізняються в області їх ізоморфізму. У теорії систем цей постулат має не менше значення, ніж закони збереження матерії у фізиці або аксіоми в математиці. Разом з іншими постулатами він є підґрунтям для логічного, доказового розгортання теорії і дає можливість пояснити єдність закономірностей природи для об'єктів, які здаються несхожими і незалежними один від одного. Ізоморфізм реальних систем є основою і логічним наслідком вищезазначеного постулату.

У теорії систем існує ще один важливий для моделювання постулат, який визначає, що описом структури і функцій деякої системи може бути інша ізоморфна стосовно неї система. Ця ізоморфність (подібність) двох систем стосується структур систем і функцій їх елементів. Одна з таких систем є *моделлю* іншої (*оригіналу*) і навпаки. Таких ізоморфних систем може бути безліч. Виникає проблема вибору або побудови системи, яка може бути моделлю досліджуваної системи.

Теорія подібності дає змогу встановити відношення еквівалентності (відповідності, схожості) між двома розглядуваними системами за деякими ознаками. Будь-яка з цих систем може існувати реально або бути абстрактною. Якщо система існує реально, то її можна вивчати, досліджуючи, яким чином пов'язані вхідні впливи з виходами системи. На основі результатів досліджень будується деяка *абстрактна система*, де відношення еквівалентності визначають тільки ті істотні властивості та аспекти поведінки, які у вихідній та абстрактній системах мають бути однаковими. М. Месарович відзначає, що, базуючись на спостереженнях і дослідженнях однієї системи, можна робити висновки про властивості та поведінку іншої. Здебільшого на практиці абстрактна система *простіша* за вихідну, якщо не враховувати тих аспектів, що визначають відношення еквівалентності.

Таким чином, можна перейти до визначення терміна «модель». У філософській літературі терміном «модель» позначають «деяку реально існуючу систему або ту, що представляється в думках, яка, заміщаючи і відображаючи в пізнавальних процесах іншу систему-оригінал, знаходиться з нею у відношенні схожості (подібності), завдяки чому вивчення моделі дає змогу отримати нову інформацію про оригінал» [39]. У цьому визначенні закладено генетичний зв'язок моделювання з теорією подібності, принципом аналогії. Таким чином, *моделлю* можна називати систему, яку використовують для дослідження.

Термін «модель» походить від латинського слова «modulus», тобто зразок, пристрій, еталон. У широкому значенні – це будь-який аналог (уявний, умовний: зображення, опис, схема, креслення тощо) певного об'єкта, процесу, явища («оригіналу» даної моделі), що використовується як його «замінник». Цей термін можна застосовувати також для позначення системи постулатів, даних і доведень, формального опису деякого явища або стану речей. Словник Вебстера визначає модель як «спрощений опис складного явища або процесу».

У сучасній теорії керування використовуються моделі двох основних типів. Для технологічних об'єктів цей поділ відповідає «феноменологічним» і «дедуктивним» моделям [52]. Під феноменологічними моделями розуміють переважно емпірично поновлені залежності вихідних даних від вхідних, як правило, з невеликою кількістю входів і виходів. Дедуктивне моделювання передбачає з'ясування та опис основних фізичних закономірностей функціонування всіх компонентів досліджуваного процесу і механізмів їх взаємодії. За допомогою дедуктивних моделей описується процес у цілому, а не окремі його режими.

Перший тип моделей – *моделі даних*, які не потребують, не використовують і не відображають будь-яких гіпотез про фізичні процеси або системи, з яких ці дані отримано. До моделей даних належать усі моделі математичної статистики. Останнім часом ця сфера моделювання пов'язується з експерементально-статистичними методами і системами, що істотно розширює методологічну базу для прийняття рішень під час розв'язання завдань аналізу даних і керування.

Другий тип моделей – *системні моделі*, які будуються в основному на базі фізичних законів і гіпотез про те, як система структурована і, можливо, як вона функціонує. Використання системних моделей передбачає можливість працювати в технологіях віртуального моделювання – на різноманітних тренажерах і в системах реального часу (операторські, інженерні, біомедичні інтерфейси, різноманітні системи діагностики і тестування тощо). Саме системні моделі будуть ядром моделювання на сучасному етапі.

Таким чином, модель є абстракцією системи і відображає деякі її властивості. Цілі моделювання формулює дослідник. Значення цілей моделювання неможливо переоцінити. Тільки завдяки їм можна визначити сукупність властивостей модельованої системи, які повинна мати модель, тобто від мети моделювання залежить потрібний ступінь деталізації моделі.

1.3. Співвідношення між моделлю та системою

З огляду на вищеописане модель – це абстракція; вона відображає лише частину властивостей системи, і мета моделювання – визначення рівня абстрактного опису системи, тобто рівня детальності її подання.

Модель і система знаходяться в деяких відношеннях, від яких залежить ступінь відповідності між ними. На міру відповідності між системою та моделлю вказують поняття *ізоморфізму* та *гомоморфізму*. Система та модель є ізоморфними, якщо існує взаємооднозначна відповідність між ними, завдяки якій можна перетворити одне подання на інше. Строго доведений ізоморфізм для систем різної природи дає можливість переносити знання з однієї галузі в іншу. За допомогою теорії ізоморфізму можна не тільки створювати моделі систем і процесів, але й організовувати процес моделювання.

Однак існують і менш тісні зв'язки між системою та моделлю. Це так звані гомоморфні зв'язки, які визначають однозначну відповідність лише в один бік – від моделі до системи. Система та модель є ізоморфними тільки в разі *спрощення* системи, тобто скорочення множини її властивостей (атрибутів) і характеристик поведінки, які впливають на *простір станів системи*. Зазвичай модель простіша

за систему. На рис. 1.1 схематично зображено різницю ізоморфної та гомоморфної залежностей між системою та моделлю для простору станів системи Z_s і моделі Z_m. Множину станів моделі Z_m визначають з огляду на мету моделювання та обраний рівень абстрактного опису.

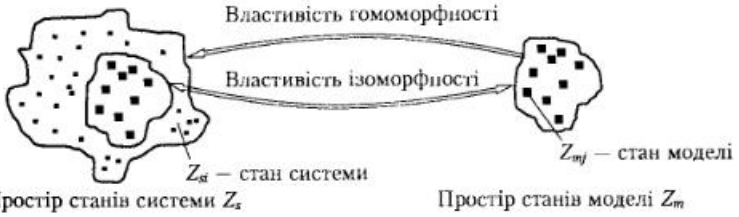


Рис. 1.1. Схематичне зображення відношень між системою та моделлю

Отже, *аналогія, абстракція та спрощення* – це основні поняття, які використовуються під час моделювання систем. Розглянемо відношення між системою та моделлю, враховуючи, що ці відношення відповідають цілям моделювання та обмеженням досліджуваної системи. Під час використання поняття множини можливих станів системи Z_s і моделі Z_m розрізняють такі типи відношень.

1. Детерміновані відношення, коли стан системи однозначно визначає стан моделі:

$$P[Z_s = Z_{si} | Z_m = Z_{mj}] = P[Z_m = Z_{mj} | Z_s = Z_{si}] = 0 \vee 1,$$

де P – ймовірність; Z_{si}, Z_{mj} – конкретні стани відповідно системи та моделі для скінченної множини значень i, j.

У цьому разі розглядається детермінована дискретна модель зі скінченною множиною можливих станів. Прикладом реалізації такої моделі може бути скінченний автомат або мережа Петрі.

2. Імовірнісні відношення зі скінченною множиною станів. У цьому випадку стан системи однозначно визначає стан моделі, але стан моделі визначає стан системи лише з деякою ймовірністю. Зазначені відношення для конкретних станів Z_{si}, Z_{mj} можна подати в такому вигляді:

$$P[Z_s = Z_{si} | Z_m = Z_{mj}] = 1 \vee 0,$$

$$P[Z_m = Z_{mj} | Z_s = Z_{si}] \le 1,$$

тобто розглядається дискретна стохастична модель зі скінченною множиною можливих станів. Прикладом реалізації подібної моделі може бути ймовірнісний автомат.

3. Імовірнісні відношення з нескінченною множиною станів, коли стани системи та моделі визначають стани одне одного лише з деякою ймовірністю:

$$P[Z_s = Z_{si} | Z_m = Z_{mj}] \le 1;$$

$$P[Z_m = Z_{mj} | Z_s = Z_{si}] \le 1.$$

Це так звані стохастичні моделі, до яких, наприклад, належать марківські моделі та моделі систем масового обслуговування.

1.4. Класифікація моделей

Для того щоб визначити види моделей, перш за все потрібно окреслити ознаки класифікації. У сучасній літературі описано сотні визначень поняття «модель» та їх класифікацій. Одну з перших, досить повних, класифікацій моделей було запропоновано Дж. Форрестером [65] у 1961 році. Інші класифікації наведено у працях [43, 54, 55], але в жодній з них немає відомостей про ознаки, за якими їх складено.

Якщо враховувати, що моделювання — це метод пізнання дійсності, то основною ознакою класифікації можна назвати *спосіб подання* моделі. За цією ознакою розрізняють *абстрактні* та *реальні* моделі (рис. 1.2). Під час моделювання можливі різні *абстрактні конструкції*, проте основною є *віртуальна* (*уявна*) модель, яка відображає ідеальне уявлення людини про навколишній світ, що фіксується в свідомості через думки та образи. Вона може подаватись у вигляді *наочної* моделі за допомогою графічних образів і зображень.

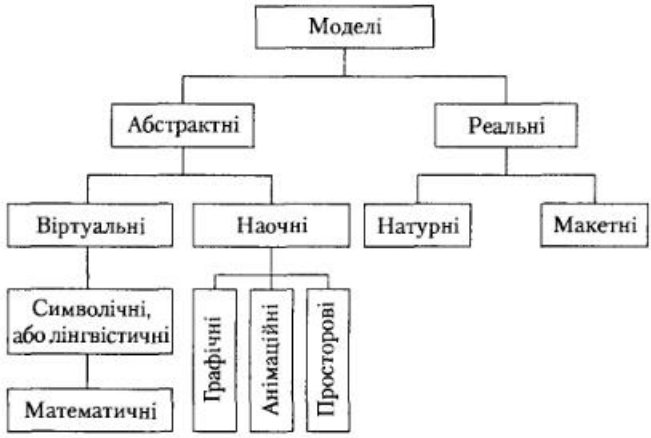


Рис. 1.2. Основні типи моделей

Наочні моделі залежно від способу реалізації можна поділити на дво- або тримірні графічні, анімаційні та просторові. Графічні та анімаційні моделі широко використовуються для відображення процесів, які відбуваються в модельованій системі. Графічні моделі застосовуються в системах автоматизованого проектування (computer-aided design, CAD). Для відтворення тримірних моделей за допомогою комп'ютера існує багато графічних пакетів, найбільш поширені з яких Corel DRAW, 3D Studio Max і Maya. Графічні моделі є базою всіх комп'ютерних ігор, а також застосовуються під час імітаційного моделювання для анімації.

Щоб побудувати модель у *формальному вигляді*, створюють *символічну*, або *лінгвістичну*, модель, яка відповідала б найвищому рівню абстрактного опису, як це було зазначено вище. На базі неї отримують інші рівні опису.

Основним видом абстрактної моделі є математична модель. *Математичною* називається абстрактна модель, яка відображає систему у вигляді математичних

відношень. Як правило, йдеться про систему математичних співвідношень, що описують процес або явище, яке вивчається; у загальному розумінні така модель є множиною символічних об'єктів і відношень між ними. Як відзначає Г. І. Рузавін у праці [51], «до сих пор в конкретных приложениях математики чаще всего имеют дело с анализом величин и взаимосвязей между ними. Эти взаимосвязи описываются с помощью уравнений и систем уравнений», через що математична модель звичайно розглядається як система рівнянь, в якій конкретні величини замінюються математичними поняттями, постійними і змінними величинами, функціями. Як правило, для цього застосовуються диференціальні, інтегральні та алгебричні рівняння. Розвиток нових розділів математики, пов'язаних з аналізом нечислових структур, досвід їх використання під час проведення досліджень свідчать, що потрібно розширювати уявлення про мову математичних моделей. Тоді математична модель визначатиметься як будь-яка математична структура, де об'єкти, а також відношення між ними можна буде інтерпретувати по-різному, наприклад як функції або функціонали.

На відміну від абстрактних, *реальні* моделі існують у природі, й з ними можна експериментувати. Реальні моделі – це такі, в яких хоча б один компонент є фізичною копією реального об'єкта. Залежно від того, в якому співвідношенні знаходяться властивості системи та моделі, реальні моделі можна поділити на натурні та макетні.

Натурні (фізичні) моделі – це існуючі системи або їх частини, на яких проводяться дослідження. Натурні моделі повністю адекватні реальній системі, що дає змогу отримувати високу точність і достовірність результатів моделювання. Суттєві недоліки натурних моделей – це неможливість моделювання критичних і аварійних режимів їх роботи та висока вартість.

Макетні моделі – це реально існуючі моделі, які відтворюють модельовану систему в певному масштабі. Іноді такі моделі називаються масштабними. Параметри моделі та системи відрізняються між собою. Числове значення цієї різниці називається масштабом моделювання, або коефіцієнтом подібності. Ці моделі розглядаються в рамках теорії подібності, яка в окремих випадках передбачає геометричну подібність оригіналу й моделі для відповідних масштабів параметрів. Найпростіші макетні моделі – це пропорційно зменшені копії існуючих систем, які відтворюють основні властивості системи або об'єкта залежно від мети моделювання. Макетні моделі широко використовуються під час вивчення фізичних та аеродинамічних процесів, гідротехнічних споруд і багатьох інших технічних систем.

За можливістю змінювати в часі свої властивості моделі поділяються на *статичні* та *динамічні*. Статичні моделі, на відміну від динамічних, не змінюють своїх властивостей у часі. Динамічні моделі також називаються імітаційними.

Залежно від того, яким чином відтворюються в часі стани моделі, розрізняють *дискретні*, *неперервні* й *дискретно-неперервні* (комбіновані) моделі. За відношеннями між станами системи й моделі розрізняють *детерміновані* й *стохастичні* моделі. Останні, на відміну від детермінованих моделей, враховують імовірнісні явища й процеси.

1.5. Вимоги до моделей

У загальному випадку під час побудови моделі потрібно враховувати такі вимоги:

- *незалежність результатів* розв'язання задач від конкретної фізичної інтерпретації елементів моделі;
- *змістовність*, тобто здатність моделі відображати істотні риси і властивості реального процесу, який вивчається і моделюється;
- *дедуктивність*, тобто можливість конструктивного використання моделі для отримання результату;
- *індуктивність* – вивчення причин і наслідків, від окремого до загального, з метою накопичення необхідних знань.

Оскільки модель створюється для вирішення конкретних завдань, розробник моделі має бути впевненим, що не отримає абсурдних результатів, а всі одержані результати відображатимуть необхідні для дослідника характеристики та властивості модельованої системи. Модель повинна дати можливість знайти відповіді на певні запитання, наприклад: «що буде, якщо ...», оскільки вони є найбільш доцільними під час глибокого вивчення проблеми. Не варто забувати, що системні аналітики використовують модель для прийняття рішень і пошуку найкращих способів створення модельованої системи або її модернізації. Завжди потрібно пам'ятати, що користувачем інформації, отриманої за допомогою моделі, є замовник. Недоцільно розробляти модель, якщо її не можна буде використовувати. Більш того, робота з моделлю має бути автоматизованою для замовника до такого ступеня, щоб він міг працювати з нею в межах своєї предметної області. Таким чином, між моделлю і користувачем повинен бути розвинутий інтерфейс, який звичайно створюється за допомогою системи меню, налагодженої на застосування моделі в певній галузі.

Ступінь деталізації моделі потрібно вибирати з огляду на цілі моделювання, можливості отримання необхідних вхідних даних для моделі та з урахуванням наявних ресурсів для її створення. Відсутність кваліфікованих фахівців може звести роботи зі створення моделі нанівець. З іншого боку, чим детальніше розроблено модель, тим вона стійкіша до вхідних впливів, які не були передбачені під час проектування, і на більшу кількість запитань може дати правильні відповіді.

1.6. Основні види моделювання

Єдина класифікація видів моделювання неможлива через багатозначність поняття моделі в науці, техніці, суспільстві. Широко відомими видами моделювання є комп'ютерне, математичне, імітаційне та статистичне. На жаль, різні джерела по-різному трактують ці поняття.

Комп'ютерне моделювання визначимо як реалізацію моделі за допомогою комп'ютера. Особливістю комп'ютерного моделювання є його *інтерактивність*, що дає змогу користувачу втручатися в процес моделювання та впливати на результати завдяки узгодженості дій користувача і моделі, яка відтворює об'єкти реального

середовища або гіпотетичні події та процеси. Під час комп'ютерного моделювання може бути задіяно реальні об'єкти (наприклад, кабіна пілота), віртуальні об'єкти, згенеровані комп'ютером, які відтворюють реальні об'єкти (наприклад, потоково-конвеєрна лінія для збирання автомобілів). Інтерактивне комп'ютерне моделювання широко застосовується в навчальних системах, наприклад для побудови тренажерів і в ситуаційних іграх.

Що стосується математичних моделей, або математичного моделювання, то слід відзначити, що під час їх використання багато чого залежить від способу подання як моделі, так і результатів моделювання. Розглянемо простий приклад. Нехай на деякому підприємстві для водопостачання використовується резервуар, об'єм якого становить W тисяч літрів. Рівень споживання – $V_п$ тисяч літрів, а швидкість наповнення резервуара – $V_з$ тисяч літрів за добу. Необхідно знайти час T, за який буде заповнено резервуар. Схему цієї системи зображено на рис. 1.3, де резервуар позначено прямокутником, а вхідний і вихідний потоки – стрілками з «вентилями», які регулюють ці потоки. Хмарки позначають необмежені потоки. Такі ідеограми широко використовуються під час побудови моделей неперервних процесів.

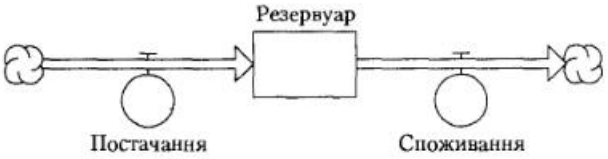


Рис. 1.3. Схема системи водопостачання

Знайдемо час заповнення резервуара:

$$T = \frac{W}{V_з - V_п}. \tag{1.1}$$

Ця математична модель процесу наповнення резервуара є надто ідеалізованою, тому що всі її параметри вважаються незмінними в часі, зовнішні впливи на систему не враховуються. Завдяки такій ідеалізації маємо дуже просту модель, яка дає змогу розв'язати задачу аналітично. Однак за допомогою такої моделі можна отримати відповідь тільки на одне конкретне запитання – за який час буде заповнено резервуар.

Якщо задачу наблизити до практики, то, будуючи модель, необхідно враховувати, що потреби підприємства у водопостачанні постійно змінюються, більш того, можливі перебої в роботі насосів під час подавання води. Розв'язок задачі в частково замкнутому вигляді можна записати як

$$W = (V_з - V_п)T. \tag{1.2}$$

За рахунок неявного запису отримано більш придатну для дослідження та аналізу реальних процесів математичну модель. Час заповнення резервуара об'ємом W залежить від параметрів моделі $V_з$, $V_п$. Використання цієї моделі дає можливість

вивчити відношення між величинами $V_п$ і $V_з$, якщо задавати різні початкові значення для них, і побудувати графік наповнення резервуара (рис. 1.4).

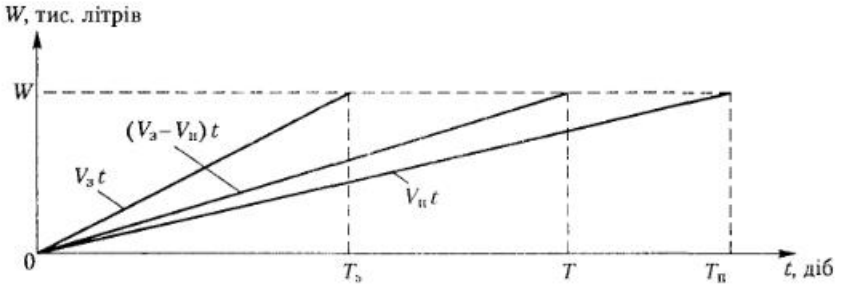


Рис. 1.4. Графік наповнення резервуара

Реалізувати цю модель можна за допомогою і чисельних методів. Змінюючи у формулі (1.2) значення t від 0 з деяким кроком Δt до такого, що буде виконуватись рівність, отримаємо динамічну характеристику заповнення резервуара. Чим менший крок Δt, тим точніший отримаємо результат, але тим довше буде вирішуватись задача моделювання.

Термін «моделювання» відповідає англійському слову «modeling», тобто побудові моделі та її аналізу. Англійський термін «simulation» відповідає прийнятому терміну «імітаційне моделювання», але часто вони використовуються разом, коли йдеться про технологічні або системні етапи моделювання, пов'язані з прийняттям рішень за допомогою моделей.

Імітаційне моделювання – це метод конструювання моделі системи та проведення експериментів. Однак під таке визначення підпадають майже всі види моделювання. Тому потрібно виділити суттєві особливості імітаційного моделювання.

Перш за все слід подати в моделі *структуру* системи, тобто загальний опис елементів і зв'язків між ними, потім визначити засоби відтворення в моделі поведінки системи. Здебільшого поведінку системи описують за допомогою станів і моментів переходів між ними. Стан системи в момент часу t визначають як безліч значень певних параметрів системи у цей самий момент часу t. Будь-яку зміну цих значень можна розглядати як перехід до іншого стану. І врешті-решт, імітаційна модель має відобразити властивості середовища, в якому функціонує досліджувана система. Зовнішнє середовище задають вхідними впливами на модель.

Вся інформація про імітаційну модель загалом має логіко-математичний характер і подається у вигляді сукупності алгоритмів, які описують процес функціонування системи. Отже, здебільшого імітаційною моделлю є її програмна реалізація на комп'ютері, а імітаційне моделювання зводиться до проведення експериментів з моделлю шляхом багаторазового прогону програми з деякою множиною даних – середовищем системи. Під час імітаційного моделювання може бути задіяно не тільки програмні засоби, але й технічні засоби, люди та реальні системи.

З математичної точки зору імітаційну модель можна розглядати як сукупність рівнянь, які розв'язують з використанням чисельних методів у разі кожної зміни модельного часу. Окремі рівняння можуть бути простими, але їх кількість і частота розв'язання – дуже великими. Розв'язання таких рівнянь під час імітаційного

моделювання означає *встановлення хронологічної послідовності подій*, які виникають у системі і відображають послідовність її станів. Отже, імітаційна модель функціонує так само, як система.

Якщо повернутись до процесу наповнення резервуара (рис. 1.5), то за допомогою імітаційної моделі весь процес можна відтворити з використанням рівняння (1.2). Позначимо через W_i поточний стан резервуара, який відтворюється в певні моменти модельного часу, що змінюється з постійним кроком Δt:

$$W_i = (V_з - V_п)t_i, \tag{1.3}$$

де $t_i = t_{i-1} + \Delta t$ $(i = 1, 2, ...)$, $t_0 = 0$. Така модель є детермінованою. Процес моделювання закінчується, якщо на деякому кроці виконується умова $W \le W_i$, тобто розв'язок отримуємо за один прогін імітаційної моделі. Точність результату буде залежати від значення Δt.

Рис. 1.5. Динамічна характеристика наповнення резервуара

За наявності в моделі випадкових факторів виникає необхідність статистичного оцінювання результатів моделювання, що виконується за допомогою методу *статистичного моделювання* (методу Монте-Карло). Статистичне моделювання є самостійним видом моделювання, яке включається в імітаційне моделювання тільки за необхідності моделювання ймовірнісних систем і процесів.

Побудуємо більш реальну модель системи, яка розглядалась вище. Припустимо, що рівень споживання води на підприємстві має імовірнісний характер і змінюється згідно з рівномірним розподілом імовірності в межах $V_п \pm \Delta V_п$. Тоді значення $V_п$ у деякий момент часу t_i будемо визначати як

$$2\Delta V_п r_i + V_п - \Delta V_п,$$

де r_i – випадкове число, рівномірно розподілене в інтервалі [0,1]. Результати роботи імітаційної моделі наведено на рис. 1.6. У цьому випадку після кожного прогону моделі отримаємо випадкові значення T_j, де j – кількість прогонів, $j = 1, 2, 3, ...$. Для кожного прогону потрібно задавати свою послідовність випадкових чисел r_i. Як видно на рис. 1.6, отримані значення T_j будуть відрізнятись від середнього значення T, знайденого за допомогою детермінованої моделі. Таким чином, щоб оцінити час T наповнення резервуара, потрібно задати точність оцінювання $\varepsilon = \Delta T$ і рівень довіри α. Звичайно $\alpha = 0{,}95$, тобто є гарантія, що в 95 випадках із 100 середнє значення часу T буде знаходитись у межах $T \pm \Delta T$.

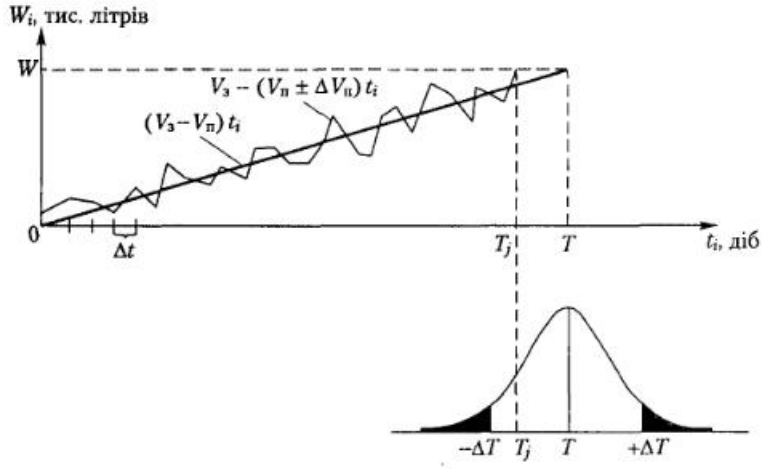


Рис. 1.6. Графік реалізації стохастичної моделі

Із вищенаведеного прикладу видно, що стохастичне моделювання використовується під час імітаційного моделювання тільки за необхідності врахування випадкових факторів.

1.7. Декомпозиція систем і простір станів

Як правило, під час побудови моделі система спрощується, тобто провадиться її декомпозиція, або розкладення на підсистеми. Якщо систему задати множиною відношень n-го порядку $R[x_1, x_2, ..., x_n]$, то загальний метод декомпозиції можна описати за допомогою операції добутку відношень. Відношення R є добутком відношень R_1 і R_2, якщо виконується умова

$$(Y\,R\,Z) \leftrightarrow [(Y\,R_1\,Z) \cap (Z\,R_2\,X)],$$

де X, Y, Z – деякі множини. Завдання дослідника полягає у визначенні відношень R_1 та R_2.

Якщо ці відношення знайдені, то систему можна подати як сукупність двох підсистем:

$$R_1[x_1, x_2, ..., x_j, Z] \text{ і } R_2[Z, x_{j+1}, x_{j+2}, ..., x_n],$$

де $x_j \in X$, $j = 1, 2, ..., n$.

У літературі [42] зазначено, що відношення n-го порядку можна розкласти на $n - 2$ тримісних відношення. З огляду на дослідження систем найважливішим є наслідок цієї теореми, пов'язаний з уведенням поняття стану системи. Розглянемо систему, яка задається відношенням

$$Y\,R\,X\,(t). \tag{1.4}$$

Другий елемент відношення, $X(t)$, є функцією часу, тобто деякою множиною. Припустимо, що множина $X(t)$ скінченна і містить n елементів.

Згідно з наслідком теореми відношення (1.4) має порядок $n + 1$ і не може бути розкладене на відношення нижче третього порядку. Нехай елементи $X(t)$ упорядковані в часі:

$$X(t) = [x(t_1), x(t_2), ..., x(t_n)].$$

Тоді відношення (1.4) має вигляд

$$Y R [x(t_1), x(t_2), ..., x(t_n)].$$

Розглянемо підмножину всіх елементів $x(t)$ з індексом, більшим за j:

$$X^j(t) = [x(t_{j+1}), ..., x(t_n)].$$

Відношення (1.4) буде еквівалентне відношенню

$$Y R [X^j(t), X^{jr}(t)],$$

де $X^{jr}(t)$ складається з членів $x(t)$, які залишились:

$$X^{jr}(t) = [x(t_1), ..., x(t_j)].$$

Якщо подати відношення R у вигляді добутку відношень R_1 і R_2, то система складатиметься з двох підсистем:

$$R_1 [X^j(t), Z^j] \quad \text{і} \quad R_2 [Z^j, X^{jr}(t)]. \tag{1.5}$$

Терм Y залежить тільки від проміжного терму Z^j і не залежить від елементів $x(t_j)$, в яких індекс менший за j. Можна стверджувати, що елемент Z^j описує *стан системи*. Якщо систему поділено на дві відповідно до виразу (1.5), то терм Y залежить тільки від стану системи в момент $\tau = t_j$ та всіх майбутніх елементів x і не залежить від усіх попередніх елементів. Стан системи в момент часу τ називається початковим і позначається через $z_0(\tau)$. Наведені міркування правильні й для нескінченних множин.

Таким чином, під час моделювання системи або процесу немає необхідності запам'ятовувати всі стани системи до моменту часу τ, тобто алгоритм моделювання «забуває», що було раніше. Якщо реалізувати алгоритм за допомогою комп'ютера, то не потрібно зберігати всі стани в пам'яті. Винятком є необхідність анімаційного або графічного відтворення станів системи в часі та можливість її «програвання» у прямому й зворотному напрямках.

Якщо потрібно зменшити порядок відношення системи шляхом усунення залежності від будь-яких елементів певної підмножини X^r, то нове відношення має бути хоча б тримісним, три терми його є входами X, виходами Y і станами Z. Рівняння

$$z(t > \tau) = z(z(\tau); x[\tau, t)) \tag{1.6}$$

будемо називати рівнянням станів системи, а функцію z — перехідною функцією станів системи. Таким чином, вхідні впливи X перетворюються у виходи системи Y за допомогою рівняння станів (1.6), і саму систему S можна подати у вигляді «чорного ящика», зображеного на рис. 1.7, де зовнішні відношення пов'язують

елементи системи із зовнішнім середовищем за допомогою входів системи. Під час проведення досліджень системи можна впливати на її входи та спостерігати за її виходами. Вхідні змінні, які дослідник може змінювати, проводячи експерименти, називаються змінними, якими керують, а ті, що неможливо змінювати, – змінними, за якими спостерігають. Під час моделювання звичайно можна змінювати всі вхідні змінні.

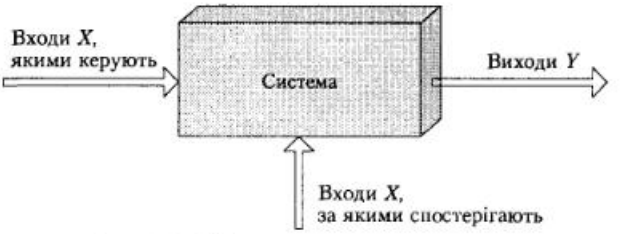


Рис. 1.7. Кібернетична модель системи

Розглядаючи простір станів, або фазовий простір, і зміни станів системи в часі, можна описати її поведінку (функціонування). Поняття стану вже давно є одним з найважливіших у техніці. У теорії систем стан системи визначається як точка фазового простору, який містить всю інформацію про передісторію системи, суттєву для визначення її поведінки в майбутньому. Через стани системи можна пов'язати виходи системи з її входами.

У разі введення множини T як певної впорядкованої множини позитивних дійсних чисел t, які визначають плин часу, пару елементів $\langle t, z\rangle$, де $t \in T, z \in Z$, називають *станом* або *фазою* системи S, а множину $T \times Z$ – *простором станів* або фазовим простором системи, де $\times$ – декартовий добуток. Перехідна функція z або її графік у просторі станів визначає поведінку системи або її траєкторію руху у фазовому просторі на певному проміжку часу $t \in [\tau, t)$. Поняття простору станів не повинне викликати труднощів. Можна уявити звичайний простір, в якому не три, а довільна кількість осей координат, а стан – це точка в цьому просторі, що характеризує об'єкт у поточний або довільний момент часу подібно тому, як координати звичайного простору характеризують просторове розташування. Під фазовим простором розуміється простір, в якому визначено не тільки статичні координати точки, координати її положення, але й міститься вся інформація, потрібна для визначення її поведінки в майбутньому.

Важливість поняття стану полягає в можливості, використовуючи його як деякий параметр, пов'язати з кожною вхідною змінною єдину вихідну змінну. Якщо зміна станів системи відбувається неперервно в часі, то динамічна система належить до класу неперервних систем. Якщо ж функцію (1.6) визначено на дискретній множині моментів часу t, то розглядають клас дискретних динамічних систем. В окремому випадку дискретні моменти часу можуть задаватись у момент настання деяких подій, які призводять до зміни станів системи.

Отже, щоб відтворити функціонування системи або, іншими словами, її траєкторію у фазовому просторі, потрібно задати рівняння станів системи (1.6). Під час моделювання системи таке рівняння називають також *функцією дії*. Цю функцію можна задати в явному вигляді, наприклад за допомогою диференціального

рівняння, або у вигляді алгоритму моделювання, який визначає стан системи в кожний момент часу t, або шляхом задання таблиці станів, як це виконується, наприклад, для дискретних автоматів.

Таким чином, *процес*, який під час моделювання системи описує її функціонування, визначається послідовністю станів, зв'язок між якими задається функцією дії та початковим станом системи. Отже, послідовність розташованих у порядку збільшення часу пар $\langle z(t), x[\tau,t)\rangle$ визначає процес і описує поведінку системи.

У разі побудови моделей динамічних систем ці системи описуються у вигляді множини деяких *реалій* (рис. 1.8), які можна описувати та моделювати за допомогою властивостей, що змінюють стани системи. Зміна станів системи викликає *події*, яким відповідають певні *умови*. Виникнення певних умов приводить до *дій*, які утворюють конкретні *процеси*.

Рис. 1.8. Схема опису динамічних систем

Процес можна також розглядати як послідовність взаємопов'язаних дій за умовами визначення початку і закінчення дії.

1.8. Формальні методи побудови моделей

Розглядаючи сфери застосування моделей, можна констатувати, що за допомогою моделі можна досягти *двох основних цілей* [67]: *описової*, якщо модель призначена для пояснення і кращого розуміння об'єкта, або *приписуючої*, коли модель дає змогу передбачити або відтворити характеристики об'єкта або визначити його поведінку.

Таким чином, *модель є описовою*, якщо вона призначена зображувати поведінку (функціонування) або властивості існуючої або типової системи (наприклад, масштабна модель або письмовий опис, що дає змогу знайомити потенційних покупців із фізичними і робочими характеристиками комп'ютера). Протилежність – *приписуюча модель*, яка відображає необхідну поведінку або властивості запропонованої системи (наприклад, масштабна модель або письмовий опис, представлений постачальнику комп'ютерів, з фізичними і робочими характеристиками потрібного замовнику комп'ютера).

Приписуюча модель може бути описовою, але не навпаки. Тому існує різний ступінь корисності моделей, які використовуються в технічних і соціальних науках. Це значною мірою залежить від методів, і засобів, застосовуваних під час побудови моделей, а також від кінцевої мети. У соціальних науках моделі призначено для пояснення існуючих систем, а в техніці вони є допоміжними засобами для створення нових або більш досконалих моделей. Модель, що придатна для досягнення цілей розроблення системи, має також пояснювати її.

Під час побудови моделей застосовуються фундаментальні закони природи, варіаційні принципи, аналогії, ієрархічні ланцюжки. Процес створення моделі включає такі етапи.

1. Словесно-смисловий опис об'єкта або явища – формулювання *описової* моделі, призначеної для сприяння кращому розумінню об'єкта моделювання.
2. Числове вираження модельованої реальності для виявлення кількісної міри і границь відповідних якостей; з цією метою провадиться математико-статистична обробка емпіричних даних, пропонується кількісне формулювання якісно встановлених фактів і узагальнень.
3. Перехід до вибору або формулювання моделей явищ і процесів (варіаційного принципу, аналогії тощо) і його запису у формалізованій формі; це рівень структурних теоретичних схем, таких як системи масового обслуговування, мережі Петрі, скінченні або ймовірнісні автомати, діаграми фонд–потік тощо.
4. Завершення формулювання моделі її «оснащенням» – задання початкового стану і параметрів об'єкта.
5. Вивчення моделі за допомогою доступних методів (у тому числі із застосуванням різних підходів і обчислювальних методів).

У результаті дослідження моделі досягається поставлена мета. У цьому разі має бути встановлена всіма можливими способами (шляхом порівняння з практикою, порівняння з іншими підходами) її адекватність – відповідність об'єкта сформульованим припущенням.

1.8.1. Кібернетичний підхід

Систему можна вивчати та аналізувати, змінюючи вхідні впливи і спостерігаючи за виходами. Це кібернетичний підхід, згідно з яким система розглядається як «чорний ящик». Метод «чорного ящика» широко використовується під час моделювання систем, коли для дослідника важливо отримати інформацію про поведінку системи, а не про її будову. Дослідник не може зробити однозначний висновок про структуру «чорного ящика», спостерігаючи тільки за його входами та виходами, бо поведінка модельованої системи нічим не відрізняється від поведінки ізоморфних їй систем.

Для побудови моделі використовуються методи теорії ідентифікації. У загальному випадку завдання ідентифікації формулюється так: на основі результатів спостереження за вхідними та вихідними змінними системи потрібно побудувати

оптимальну в деякому розумінні математичну модель. Основними етапами ідентифікації є такі:

1. Вибір класу і структури моделі та мови її опису.
2. Вибір класу і типів вхідних впливів X.
3. Обгрунтування критеріїв подібності системи та моделі.
4. Вибір методу ідентифікації та розроблення відповідних алгоритмів оцінювання параметрів системи.
5. Перевірку адекватності отриманої в результаті ідентифікації моделі.

Залежно від обсягу апріорної інформації про клас і структуру системи вирізняють завдання ідентифікації в широкому та вузькому розумінні [43]. Завдання ідентифікації у широкому розумінні виконується в умовах апріорної невизначеності структури моделі системи («чорний ящик»). Клас і структура математичної моделі вибираються на основі результатів теоретичного аналізу з використанням загальних закономірностей процесів, які протікають у системі, або на основі загальної інформації про подібні системи. У цьому випадку для побудови математичної моделі можна використовувати непараметричні методи. Їх розроблено для тих ситуацій, які досить часто виникають на практиці, коли дослідник нічого не знає про параметри досліджуваної системи (звідси і назва методів — *непараметричні*).

Завдання ідентифікації у вузькому розумінні полягає в оцінюванні параметрів і станів системи, якщо відома структура моделі («сірий ящик»). Завданням ідентифікації є кількісне оцінювання певних параметрів. Для цього використовується параметрична ідентифікація математичної моделі. Прикладами таких моделей можуть бути диференціальні та різницеві рівняння, моделі типу «вхід–стан–вихід».

На рис. 1.9 зображено загальну схему ідентифікації системи. Вхідні впливи X на систему та модель однакові, виходи системи Y_s і моделі Y_m у загальному випадку відрізняються. Для їх порівняння потрібно сформувати критерій подібності та мінімізувати його, тобто налагодити модель.

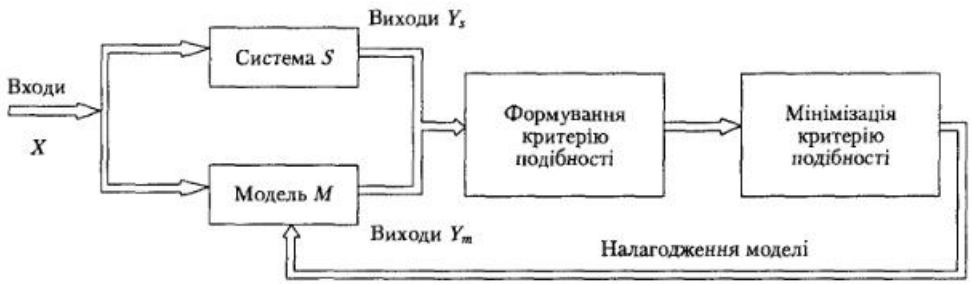


Рис. 1.9. Загальна схема ідентифікації системи

Прикладами моделей, створених на основі експериментальних даних, можуть бути моделі авторегресії різних порядків, ковзного середнього та моделі типу «вхід–вихід», побудовані за допомогою методу найменших квадратів.

1.8.2. Системна динаміка

Для формального представлення моделей неперервних систем Дж. Форрестер у 1960 році запропонував підхід, названий *системною динамікою*, який дає змогу будувати моделі динамічних взаємопов'язаних систем за допомогою причинних діаграм циклів і схем виду «фонд–потік». Він же запропонував для числового моделювання таких систем мову Динамо. Модель будується як система диференціально-різницевих рівнянь, а мова Динамо дає можливість автоматизувати процес їх написання. Практично всі сучасні засоби неперервного та неперервно-дискретного моделювання базуються на цій мові для побудови моделей. На відміну від математичного розв'язання системи таких рівнянь у замкненому вигляді використовується числове розв'язання з дискретним кроком часу, що дає змогу моделювати на деякому проміжку часу динамічні зміни фондів, пов'язаних з точкою часу і потоків. Фонди та потоки пов'язані між собою через змінні.

Фонд можна трактувати як деяку кількість чого-небудь, що виміряється в певних одиницях (наприклад, фізичних, грошових та ін.). Фонди можуть акумулювати одиниці фонду. Найкраще їх уявляти як резервуари, ресурси або буфери. Фонди поповнюються через вхідні потоки та спорожнюються через вихідні. Як буфер фонд може використовуватись для забезпечення балансування швидкості накопичення та витрачання (наприклад, у задачі про водопостачання, яка розглядалась у розділі 1.4).

Потік – це процес, що протікає неперервно в часі, оцінити який можна в деяких кількісних одиницях за певний проміжок часу. Залежно від характеристики використання потоки поділяються на обмежені та необмежені, одно- та двоспрямовані, конвертовані та неконвертовані. Потік, як правило, обмежується фондом. Потоком можна керувати, тобто збільшувати або зменшувати його інтенсивність за допомогою деяких алгебричних виразів.

Існує багато різних способів зв'язувати у динамічних моделях причини та наслідки, не розглядаючи конкретні методи. В їх основі лежить кілька підходів. Розглянемо три з них, які наведено на рис. 1.10.

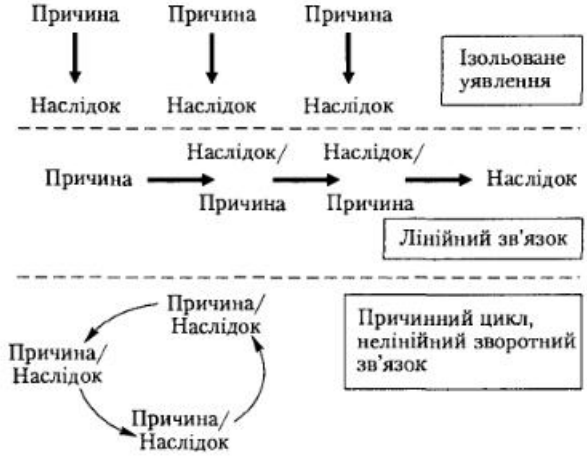


Рис. 1.10. Три підходи до зв'язування причин і наслідків для побудови моделі

Перший підхід полягає в тому, що наслідок виникає з причини і взаємозв'язок між різними причинами відсутній. Такий підхід, наприклад, використовують бухгалтери під час розрахунків. Як правило, для цього застосовують статичні та статистичні моделі.

Другий підхід передбачає, що між причинами та наслідками існує лінійний зв'язок у вигляді ланцюжка. Такий підхід підтримують інженери та науковці, які вважають, що всі події у всесвіті залежать одна від одної. Маючи достатню кількість інформації, можна побудувати залежності в часі для всіх подій у майбутньому. Системні мислителі, які застосовують цю парадигму, користуються діаграмами впливу та моделями лінійних рівнянь і вважають, що завжди можна логічно прослідкувати, «що є на вході і що буде на виході».

Згідно з третім підходом всесвіт розглядається як система зі зворотними зв'язками, тобто ланцюжки причин і наслідків циклічно пов'язані між собою. Таке уявлення підтримують кібернетики, прихильники нелінійної динаміки та хаосу. Вони вважають, що всесвіт значною мірою хаотичний, і передбачити майбутнє з огляду на його минуле неможливо. Ці системні мислителі використовують циклічні причинні моделі, нелінійні рівняння в кінцевих різницях. Часто поведінка таких моделей далека від реальності та інтуїтивного уявлення і може бути дещо несподіваною для дослідника.

На рис. 1.11 зображено найпростішу причинну циклічну модель для деякої популяції, яка має два цикли. Лівий цикл, позитивний, свідчить про приріст популяції в разі збільшення народжуваності, що у свою чергу збільшує народжуваність. Правий цикл, негативний, свідчить про зменшення популяції у випадку збільшення смертності, що у свою чергу зменшує смертність. Такі пари причинних циклів можуть використовуватись під час побудови більш складних динамічних моделей.

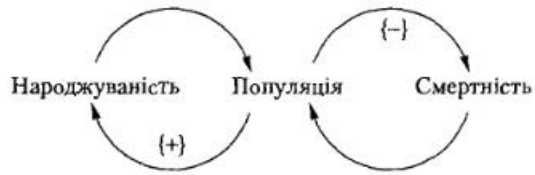


Рис. 1.11. Найпростіша причинна модель циклу популяції

Побудова складних динамічних моделей з використанням причинних циклів включає такі етапи.

1. Абстрагування від фізичної структури.
2. Концентрація на процесах для визначення траєкторій, за якими система починає та закінчує працювати.
3. Використання простих диференціально-різницевих рівнянь для опису процесів у системі:
   - $dx/dt = kx$ — показова функція, яка визначає швидкість зміни фонду в часі, де x — фонд (для прикладу з водопостачанням — це швидкість наповнення бака);
   - $dx/dt = ax - bx^2$ — сигмаїдальна, або логістична, крива, або S-крива;

або системи рівнянь:

- $dx/dt = k_1x - k_2x^2 - k_3y$ (наприклад, x – кількість травоїдних тварин);
- $dx/dt = k_4y - k_5y^2 - k_6x$ (наприклад, y – кількість хижаків).

Такі системи рівнянь відомі як рівняння Ланкастера. Їх можна використовувати для дослідження складних взаємозв'язків, конкуренції або конфліктів. За допомогою комп'ютерів подібні рівняння можна подати в числовому вигляді. Для цього використовують прості рівняння рекурсії:

$$\frac{dx}{dt} = f(x),$$

$$\frac{[x(n+1) - x(n)]}{dt} = f(x(n)),$$

$$x(n+1) = dt\, f(x(n)) + x(n).$$

Якщо описати дані рівняння словами, то наступний рівень дорівнюватиме попередньому плюс невеличка зміна протягом короткого проміжку часу. Таким способом можна будувати складні динамічні моделі за допомогою створення простих блоків у вигляді відношень і рівнів. У сучасних пакетах моделювання цей процес запису рівнянь автоматизовано із застосуванням ідеографічних схем (див. рис. 1.3).

Причинні діаграми циклів дають змогу провадити *якісне* моделювання, а діаграми «фонд–потік» – *кількісне*. Щоб пояснити явище, потрібно знайти «причини» його виникнення. Припустимо, що таку причину визначено і наслідок може спостерігатись кожного разу, коли ця причина присутня. Якщо описують ці концепції системного мислення звичайними словами, то використовують слова або фрази «тому що», «завдяки тому, що», «якщо ..., то» та ін. З погляду математики, якщо розглядають функціональну концепцію з однією незалежною змінною, ця змінна – *причина*, а залежна змінна – *наслідок*.

У разі кількісного моделювання таких систем модельовані об'єкти – це об'єкти, параметри яких можна виміряти і між якими існують функціональні залежності. Якщо розглядати систему «хижаки–зайці», то в кількісній моделі знищення хижаками деяких зайців – це не знищення тварин, а зменшення їх кількості. Тобто в системі є суттєва різниця між зайцями як тваринами та їх кількістю. Наприклад, вовк може знищити зайців, але кількість вовків не може знищити дещо, а може тільки вплинути на кількість зайців.

Вищенаведені моделі динамічних систем широко використовуються для побудови спеціальних засобів моделювання – мов і пакетів неперервного та неперервно-дискретного імітаційного моделювання.

1.8.3. Теоретико-множинний підхід

Згідно з теоретико-множинним підходом [60] формальна модель динамічної системи має такий вигляд:

$$M = \langle T, X, Y, Z, z(t), P \rangle, \tag{1.7}$$

де T – модельний час; X, Y – множина відповідно вхідних і вихідних змінних; Z – простір станів моделі; $z(t)$ – функція станів, $t \in T$; P – множина процесів, яка

визначається як множина впорядкованих у часі пар елементів $\langle x, z[\tau, t)\rangle$, де $t \in T$, а τ – початковий момент модельного часу для процесу $p \in P$. Таке визначення задає модель системи у вигляді *схеми процесів*, у якій множини процесів можуть існувати паралельно в модельному часі T.

Вважається, що деяка подія з множини подій C зумовлює зміну стану системи, якщо починається певний процес $p_i \in P$ або закінчується деякий процес $p_j \in P$. У протилежному випадку стан системи не змінюється. Тоді можна задати *подійну схему* моделі:

$$M = \langle T, X, Y, Z, z(t), C \rangle, \tag{1.8}$$

де C – множина подій, що визначається як множина впорядкованих у часі пар елементів $\langle j, d[\tau, t_j] \rangle$, де $c \in C$, $d[\tau, t_j]$ – функція дії для процесу $p_j \in P$; $t \in T$, а τ – початковий момент модельного часу T. У цій схемі процес моделювання описується як послідовність подій, що відбуваються в моделі.

Припустимо, що завдяки виконанню деякої умови u з множини U почне виконуватись певна дія $d[\tau, t_j]$ з множини D для деякого процесу $p_j \in P$. Тоді можна задати модель системи у вигляді *схеми дій*:

$$M = \langle T, X, Y, Z, z(t), D \rangle. \tag{1.9}$$

У цій схемі процес моделювання описується як перевірка всіх умов у разі кожної зміни модельного часу $t \in T$, щоб знайти умову, яка почне певну дію з множини D. Зміна часу t може відбуватись з постійним або змінним від події до події кроком. Схеми моделей (1.7)–(1.9) широко застосовуються під час побудови алгоритмів моделювання і мов дискретного імітаційного моделювання.

Якщо припустити, що виконання деякої множини процесів P може призвести до зміни станів $z \in Z$ і виникнення нових процесів, що спричинить появу деякої множини ситуацій L, тобто $z(t) : Pz \to L$, то отримаємо ситуаційну або причинно-наслідкову схему:

$$M = \langle T, X, Y, Z, z(t), L \rangle, \tag{1.10}$$

в якій потрібно описати множину ситуацій та множину правил (алгоритмів), за якими визначають процес, що має виконуватись. Поведінка моделі в таких системах зображується у вигляді ланцюга

$$\{\text{ситуація}\} \to \{\text{правило}\} \to \{\text{процес}\}.$$

Якщо модель здатна конструювати нові правила на основі існуючих, то вона перетворюється на модель зі штучним інтелектом.

Під час ситуаційного моделювання, як правило, повний опис усіх можливих ситуацій замінюється деякою множиною узагальнених ситуацій, кожна з яких з певною мірою ймовірності відтворює один із можливих станів системи. Для кожної ситуації існує набір правил дії. Вибір того або іншого правила може здійснюватись за деяким критерієм або за допомогою таблиць прийняття рішень, а в простіших випадках – згідно із заданою ймовірністю. Моделювання виконується шляхом програвання різних ситуацій за певним сценарієм, яким у окремому

випадку може бути алгоритм моделювання. Таким чином створюють різні ігри, наприклад ділові, військові, економічні, розважальні. *Гра* – це спрощене відтворення реального процесу, яке здебільшого використовується для навчання, прийняття рішень, проведення досліджень або розваг.

Визначити систему можна не тільки як сукупність елементів, але і як сукупність відношень, спостерігаючи за їх змінами. Перш за все це стосується взаємодії між різними динамічними системами, кожна з яких досить складна. Прикладом можуть бути екологічні та соціальні системи. Під час вивчення таких систем дослідник, базуючись на системному аналізі, вивчає та описує впливи однієї системи на іншу.

1.9. Принципи побудови моделей

Розглянемо коротко основні принципи моделювання, які відображають достатньо багатий досвід, накопичений на даний час у галузі розроблення і використання моделей.

- **Принцип інформаційної достатності.** За повної відсутності інформації про систему модель побудувати неможливо. За наявності повної інформації про систему її моделювання недоцільне. Існує деякий критичний рівень апріорних відомостей про систему (рівень інформаційної достатності), після досягнення якого можна побудувати її адекватну модель.
- **Принцип доцільності.** Модель створюється для досягнення деяких цілей, які визначають на первинному етапі формулювання проблеми моделювання.
- **Принцип здійсненності.** Модель, яка створюється, має забезпечувати досягнення мети дослідження з урахуванням граничних ресурсів з імовірністю, суттєво відмінною від нуля, і за кінцевий час. Звичайно задають деяке граничне значення P (ступінь ризику) ймовірності досягнення мети моделювання $P(t)$, а також сам граничний термін t досягнення мети. Модель вважають здійсненною, якщо $P(t) \geq P$.
- **Принцип миожинності моделей.** Модель, яка створюється, має відображати в першу чергу ті властивості реальної системи (або явища), які впливають на вибраний показник ефективності. Відповідно під час використання будь-якої конкретної моделі пізнаються лише деякі складові реальності. Для повного її дослідження необхідно мати ряд моделей, які дали б змогу відобразити певний процес з різних боків і з різним ступенем детальності.
- **Прицип агрегації.** У більшості випадків складну систему можна подати такою, що складається з агрегатів (підсистем), для адекватного формального описування яких придатними є деякі стандартні математичні схеми. Принцип агрегації дає змогу досить гнучко перебудовувати модель залежно від завдань дослідження.
- **Принцип параметризації.** У ряді випадків модельована система має у своєму складі деякі відносно ізольовані підсистеми, які характеризуються певними параметрами, у тому числі векторними. Такі підсистеми можна замінювати

в моделі відповідними числовими величинами, а не описувати процес їх функціонування. У разі необхідності залежність значень цих величин від ситуації може задаватись у вигляді таблиць, графіків або аналітичних виразів (формул), наприклад за допомогою регресійного аналізу. Принцип параметризації дає змогу скоротити обсяг і тривалість моделювання, але слід мати на увазі, що параметризація знижує адекватність моделі.

Потреба в моделюванні виникає як на етапі проектування систем для оцінювання правильності прийнятих рішень, так і на етапі експлуатації – для оцінювання наслідків внесення змін у системи. У цьому випадку на різних етапах проектування (технічний або робочий проект) з уточненням вихідних даних і виявленням нових суттєвих факторів ступінь деталізації процесу в системі зростає, що має відобразитись у моделі. Отже, у моделі можуть водночас знаходитись блоки з різним ступенем деталізації, які моделюють одні й ті самі компоненти проектованої системи. Іншими словами, під час побудови моделі потрібно застосовувати методологію *ітераційного багаторівневого* моделювання.

Розроблення моделі доцільно починати зі створення простої вихідної моделі, яку в процесі уточнення вхідних даних і характеристик системи ускладнюють і коригують, тобто *адаптують* до нових умов. Водночас модель має залишатись досить *наочною*, тобто її структура має відповідати структурі модельованої системи, а рівень деталізації моделі повинен вибиратись з урахуванням мети моделювання, ресурсних обмежень (наприклад, час, кваліфіковані людські ресурси і кошти, виділені на проектування) і можливості отримання вхідних даних.

Отже, модель має бути багаторівневою, адаптивною, наочною, цільовою, розвиватись ітераційним способом, ускладнюватись і коригуватись у процесі утворення, що можливо тільки за умови побудови її блоковим (модульним) способом. Програмування та налагодження моделі доцільно провадити поетапно, з наступним збільшенням програмних модулів.

Один із способів підвищення ефективності моделювання полягає в тому, щоб не будувати заново модель для кожної нової системи, а вирізняти окремі класи систем і створювати уніфіковані програмні моделі для класів у цілому. Узагальнені програмні моделі дають змогу моделювати будь-яку систему із заданого класу без додаткових витрат на програмування. Така методологія забезпечує єдиний системний підхід до розроблення програмних реалізацій моделей і використовується під час об'єктно-орієнтованого програмування у вигляді бібліотеки класів моделей.

Розглянутий підхід можна реалізувати також у вигляді спеціалізованої мови або пакета моделювання, що дає змогу створювати узагальнені моделі шляхом уведення засобів розмноження підмоделей, реорганізації зв'язків між ними та їх параметричного налагодження. Цей спосіб орієнтовано на фахівців, добре обізнаних з мовою моделювання. Інший спосіб реалізації цього підходу полягає в розробленні діалогових інтелектуальних систем моделювання з використанням банку моделей та бази знань, які користувач може налагоджувати на конкретну реалізацію [29, 32, 44]. У цьому разі етап програмування можна повністю виключити під час програмної реалізації імітаційної моделі завдяки використанню ефективних методів взаємодії з базами даних і застосуванню засобів генерації моделей.

1.10. Технологія моделювання

Основою моделювання є методологія системного аналізу. Це дає змогу досліджувати систему, яка проектується або аналізується, за технологією операційного дослідження, включаючи такі взаємопов'язані етапи:

1. Формулювання проблеми та змістове поставлення задачі.
2. Розроблення концептуальної моделі.
3. Розроблення програмної реалізації моделі (зазвичай застосовується комп'ютерна модель), яка включає:
   а) вибір засобів програмування, за допомогою яких буде реалізовано модель;
   б) розроблення структурної схеми моделі та складання опису її функціонування;
   в) програмна реалізація моделі.
4. Перевірка адекватності моделі.
5. Організація та планування проведення експериментів, яке включає оцінювання точності результатів моделювання.
6. Інтерпретація результатів моделювання та прийняття рішень.
7. Оформлення результатів дослідження.

На першому етапі замовник формулює проблему. Організовуються зустрічі керівника проекту із замовником, аналітиками з моделювання та експертами з проблеми, яка вивчається. Визначаються цілі дослідження та спеціальні питання, відповіді на які буде одержано за результатами дослідження; встановлюються критерії оцінювання роботи, які використовуватимуться для вивчення ефективності різних конфігурацій системи; розглядаються такі показники, як масштаб моделі, період дослідження і необхідні ресурси; визначаються конфігурації модельованої системи, а також потрібне програмне забезпечення.

На цьому ж етапі провадиться цілеспрямоване дослідження модельованої системи, залучаються експерти з проблеми, що вирішується, які володіють достовірною інформацією. Збирається інформація про конфігурацію системи і способи експлуатації для визначення параметрів моделі і вхідних розподілів імовірностей.

На другому етапі розробляється *концептуальна модель* – абстрактна модель, яка дає змогу виявити причинно-наслідкові зв'язки, властиві досліджуваному об'єкту в межах, визначених цілями дослідження. По суті, це формальний опис об'єкта моделювання, який відображає концепцію (погляд дослідника на проблему). Вона включає в явному вигляді логіку, алгоритми, припущення й обмеження.

Згідно з цілями моделювання визначаються вихідні показники, які потрібно збирати під час моделювання, ступінь деталізації, необхідні вхідні дані для моделювання.

Рівень деталізації моделі залежить від таких чинників: цілі проекту; критерії оцінювання показників роботи; доступність даних; достовірність результатів; комп'ютерні можливості; думки експертів з проблеми, що вирішується; обмеження, пов'язані з часом і фінансуванням. Провадиться структурний аналіз концептуальної моделі, пропонується опис допущень, які обговорюються із замовником, керівником проекту, аналітиками та експертами з проблеми, яка вирішується.

Розробляються моделі вхідних даних, провадиться їх статистичний аналіз, за результатами якого визначають розподіли ймовірностей, регресійні, кореляційні та інші залежності. На цьому етапі для попереднього аналізу даних широко застосовують різні статистичні пакети (наприклад, Statistica).

Для динамічних систем провадиться пооперaційний аналіз функціонування модельованої системи з детальним описуванням роботи елементів системи. За результатами такого аналізу можна з'ясувати, чи можна вирішити проблему без застосування засобів моделювання. Детально опрацьована концептуальна модель дає змогу замовнику з іншого боку поглянути на роботу системи та, наприклад, визначити вузькі місця системи, які спричиняють зниження її пропускної здатності.

Одна з найскладніших проблем, з якою має справу аналітик моделювання, полягає у визначенні, чи адекватна модель системі. Якщо імітаційна модель «адекватна», її можна використовувати для прийняття рішень щодо системи, яку вона представляє, тобто ніби вони приймались на основі результатів проведення експериментів з реальною системою. Модель складної системи може тільки приблизно відповідати оригіналу, незалежно від того, скільки зусиль затрачено на її розроблення, тому що абсолютно адекватних моделей не існує.

Оскільки модель завжди має розроблятись для певної множини цілей, то модель, яка є адекватною для однієї мети, може не бути такою для дослідження іншої. Слід відзначити, що адекватна модель не обов'язково є достовірною, і навпаки. Модель може бути достовірною, але, в цьому разі, не використовуватись для прийняття рішень. Наприклад, достовірна модель не може бути адекватною з політичних або економічних причин.

Під час розроблення програмної реалізації моделі визначаються засоби для програмування, тобто мови програмування або пакети. Наприклад, можуть використовуватись мови програмування загального призначення, такі як C чи PASCAL, або спеціалізовані засоби для моделювання (наприклад, Arena, AutoMod, Extend, GPSS, iThink). Перевага використання мов програмування полягає в тому, що, як відомо, вони мають невисоку закупівельну вартість, і на виконання моделі з їх допомогою затрачається менше часу. Натомість використання програмного забезпечення моделювання сприяє зменшенню тривалості програмування і вартості всього проекту.

Серед спеціалізованих пакетів для моделювання слід відзначити MATLAB з інтерактивним модулем Simulink. Пакет MATLAB є всесвітньо визнаним універсальним відкритим середовищем, і мовою програмування водночас, в якому інтегровані засоби обчислень, візуалізації, програмування та моделювання.

Здійснюється програмування моделі та її налагодження, виконуються тестові прогони моделі на основі контрольних даних, провадиться аналіз чутливості, щоб визначити, які фактори в моделі суттєво впливають на робочі характеристики системи і мають моделюватись дуже точно.

Після кожного з вищезазначених етапів перевіряється достовірність моделі. Перевірку умовно можна розділити на два етапи: перевірка правильності створення концептуальної моделі, тобто задуму – *валідація*; перевірка правильності її реалізації – *верифікація* [38]. Під час перевірки достовірності потрібно відповісти на запитання про відповідність моделі модельованій системі, тобто визначити,

наскільки ізоморфні система та модель. Як правило, у разі моделювання вимога ізоморфізму об'єкта та моделі надмірна, бо в цьому разі складність моделі має відповідати складності об'єкта. Через те будують гомоморфні моделі, в яких виконується вимога однозначної відповідності моделі об'єкту.

На етапі *верифікації* розглядають, чи правильно перетворено концептуальну модель (модельні припущення) на комп'ютерну програму, тобто виконують налагодження програми моделювання. Це складне завдання, оскільки може існувати безліч логічних шляхів.

Етап перевірки правильності реалізації моделі включає перевірку еквівалентності перетворення моделі на кожному з етапів її реалізації та порівняння станів. У цьому разі модель зазнає таких змін: концептуальна модель–математична модель–алгоритм моделювання–програмна реалізація моделі.

Валідація – це процес, який дає змогу встановити, чи є модель (а не комп'ютерна програма) точним відображенням системи *для конкретних цілей дослідження* [33].

Розробляється план проведення експериментів з моделлю для досягнення поставлених цілей. Основна мета планування експериментів – вивчення поведінки модельованої системи при найменших витратах під час експериментів. Зазвичай провадять такі експерименти:

- порівнюють середні значення і дисперсії різних альтернатив;
- визначають важливість урахування впливу змінних та обмежень, які накладаються на ці змінні;
- визначають оптимальні значення з деякої множини можливих значень змінних.

Проведення експериментів планують для пошуку незначущих факторів. У випадку оптимізації якого-небудь числового критерію формулюють гіпотези щодо вибору найкращих варіантів структур модельованої системи або режимів її функціонування, визначають діапазон значень параметрів (режимів функціонування) моделі, у межах якого знаходиться оптимальне рішення. Визначають кількість реалізацій та час прогону моделі кожної реалізації. Провадять екстремальний експеримент, за результатами якого знаходять оптимальне значення критерію і відповідні значення параметрів. Для оцінювання точності стохастичних моделей будують довірчі інтервали для одержуваних вихідних змінних.

Далі аналізують та оцінюють результати. Наводять результати комп'ютерних експериментів у вигляді графіків, таблиць, роздруківок, а також визначають якісні і кількісні оцінки результатів моделювання. Для унаочнення моделі використовують анімацію. Обговорюють процес створення моделі та її достовірність, щоб підвищити рівень довіри до неї.

За отриманими результатами формулюють висновки з проведених досліджень і визначають рекомендації щодо використання моделі й прийняття рішень.

Вищенаведені етапи моделювання взаємопов'язані, а сама процедура створення моделі ітераційна. Це пояснюється тим, що після виконання кожного етапу перевіряється правильність і достовірність моделі та в разі невідповідності моделі об'єкту здійснюється повернення до попередніх етапів з метою коригування та підстроювання моделі. Залежно від характеру внесених змін повертаються

безпосередньо до попереднього етапу або до більш ранніх етапів. Детальніше технологія моделювання розглядається в наступних розділах.

На останньому етапі моделювання документально оформлюють усі результати дослідження і готують програмну документацію для використання їх під час розроблення поточних і майбутніх проектів.

Висновки

- Система – це цілісний комплекс взаємопов'язаних елементів, який має певну структуру і взаємодіє із зовнішнім середовищем.
- Модель – це реально існуюча або уявна система, яка, заміщаючи і відображаючи в пізнавальних процесах іншу систему-оригінал, знаходиться з нею у відношенні подібності.
- Моделювання – це спосіб дослідження будь-яких явищ, процесів або об'єктів шляхом побудови та аналізу їх моделей.
- Імітаційне моделювання – це метод конструювання моделі системи та проведення експериментів над моделлю.

Контрольні запитання та завдання

1. Що таке система? Як впливає на систему зовнішнє середовище? Чому існує багато визначень системи?
2. Назвіть кілька статичних і динамічних об'єктів, дій, процесів, атрибутів, подій та змінних станів для таких систем:
   а) станція технічного обслуговування автомобілів;
   б) магазин самообслуговування;
   в) станція швидкої допомоги;
   г) кафе;
   д) таксомоторний парк.
3. Яким чином динамічна поведінка системи пов'язана з поняттям стану системи?
4. Що розуміють під абстрактною системою?
5. Що розуміють під моделлю? У яких відношеннях перебувають об'єкт моделювання та модель? Чи може система бути моделлю?
6. Яку роль відіграють поняття стану та процесу в моделюванні? Опишіть роботу банку з двома касирами, до яких стоїть одна черга. Виділіть основні стани цієї системи.
7. Наведіть приклади задач, які можна розв'язати за допомогою моделювання. В яких випадках задачі можна розв'язати лише у такий спосіб?
8. Виконайте критичний аналіз різних видів класифікацій моделей та видів моделювання. Чому неможлива єдина класифікація? Запропонуйте іншу класифікацію моделей.

9. Покажіть, яким чином можна провести декомпозицію для нескінченних множин (див. розділ 1.7).
10. Порівняйте міркування щодо процесів Маркова з дискретними і безперервними станами з тими, що наведені в розділі 1.7. Для яких дискретних і безперервних розподілів часу перебування процесу в даному стані виконуються умови без післядії (без запам'ятовування)?
11. Порівняйте числовий метод розв'язання задачі про водопостачання з методом імітаційного моделювання. Що є між ними спільного?
12. Сформулюйте завдання ідентифікації в широкому та вузькому розумінні для задачі про водопостачання.
13. Яким чином задається час моделювання в задачах про водопостачання? Чи можливо так задати час моделювання для цієї задачі, щоб він залежав від деяких подій? Наведіть приклади моделювання таких подій.
14. Дайте ситуаційний опис переходу пішоходом дороги. Розгляньте всі можливі ситуації.

Розділ 2

Моделі систем масового обслуговування

- Класифікація систем масового обслуговування
- Основні характеристики систем масового обслуговування
- Математичні моделі потоків вимог
- Одноканальні та багатоканальні системи масового обслуговування
- Імітаційне моделювання систем масового обслуговування
- Мережі систем масового обслуговування
- Операційний аналіз мереж систем масового обслуговування

У теорії і практиці моделювання систем важливе місце посідають моделі систем масового обслуговування (СМО). Такі системи зустрічаються нам щоденно. Це процеси обслуговування в черзі на заправній станції, у магазині, бібліотеці, кафе, також різні служби ремонту і медичної допомоги, транспортні системи, аеропорти, вокзали тощо. Черги виникають і за потреби скористатись телефонним зв'язком або передати повідомлення по Інтернету. Більше того, будь-яке виробництво також можна подати як послідовність таких систем. Особливого значення СМО набули в інформатиці. Це передусім комп'ютерні системи, мережі передавання інформації, бази і банки даних.

Існує розвинутий математичний апарат теорії масового обслуговування (науковці західних країн цю теорією називають теорією черг), що дає змогу аналізувати ефективність функціонування СМО певних типів і визначати залежність між характеристиками потоку вимог, кількістю каналів (пристроїв для обслуговування), їх продуктивністю, правилами роботи СМО та її ефективністю.

Перші теоретичні результати внаслідок вирішення проблем, пов'язаних із функціонуванням систем обслуговування, було отримано датським ученим, співробітником Копенгагенської телефонної компанії А. К. Ерлангом у період 1908–1922 років. Ці результати стосувались практичних завдань підвищення якості обслуговування абонентів і визначення кількості телефонних ліній. У подальшому з'ясувалось, що отримані теоретичні результати є настільки загальними, що їх можна використовувати для визначення оптимальної кількості кас і продавців на торговельних підприємствах, для розрахунків запасів у магазинах, достатніх для їх безперебійної роботи, тощо. Однак більшість результатів було отримано для систем, в яких

процеси надходження та обслуговування вимог є марківськими або напівмарківськими. У цьому разі завдання аналізу СМО можна описати звичайними диференціальними рівняннями і в явному вигляді обчислити основні характеристики системи.

На практиці часто виникають задачі, пов'язані з чергами, які неможливо розв'язати із застосуванням існуючих методів теорії масового обслуговування. Це зумовило інтенсивний розвиток методів дослідження СМО за допомогою засобів імітаційного моделювання. У цьому випадку характеристики СМО оцінюються наближено шляхом обробки результатів моделювання системи.

У даному розділі розглядаються обидва підходи до аналізу СМО, причому основна увага зосереджується на методах імітаційного моделювання. Зазначається, яким чином потрібно будувати алгоритми моделювання та їх програмні реалізації для дискретно-подійних систем і як збирати інформацію про показники їх роботи.

2.1. Характеристики систем масового обслуговування

Функціонування будь-якої СМО полягає в обслуговуванні потоку вимог, які одна за одною або групами надходять до неї в деякі, як правило, випадкові моменти часу. Вимоги, які надійшли до СМО, обробляються протягом певного часу, після чого залишають систему.

У будь-якій системі обслуговування передбачена наявність *пристроїв для обслуговування* (інші назви: *прилади для обслуговування, сервери, канали*) і *вимог* (інші назви: *заявки, виклики, клієнти*), які потребують обслуговування. Правила або алгоритми взаємодії пристроїв і вимог називатимемо *дисциплінами поставлення в чергу* та *обслуговуванням*.

Для кожної СМО задається режим роботи. Слід відзначити, що для вимоги може бути потрібно кілька обслуговувань одним або кількома пристроями. Звичайно термін *«пристрій для обслуговування»* (англійською – «server») використовується для відносно простих моделей, в яких кожна вимога може обслуговуватись тільки одним пристроєм. Якщо ж вимоги обслуговуються кількома пристроями в певній послідовності, переміщаючись за заданим маршрутом, то має місце «мережа обслуговування» (англійською – «queueing network»). Іншими словами, мережа – це складна СМО.

Зазвичай за допомогою методів теорії масового обслуговування розв'язують задачі з проектування та експлуатації однотипних елементів обслуговування – наприклад, розраховують кількість контрольно-пропускного обладнання, місць для ремонту, бензоколонок, обслуговуючого персоналу, ліній зв'язку, одиниць обладнання обчислювальної техніки тощо.

Окремим типом завдань у теорії масового обслуговування є визначення місць накопичування вимог у системі обслуговування, наприклад визначення місць на стелажах на складі або в багатоповерховому гаражі, кількості пристроїв введення-виведення інформації комп'ютера, кількості місць у палатах госпіталю та ін.

Найчастіше ефективність функціонування будь-якої СМО визначається за такими показниками:

- середня кількість вимог, які система може обслужити за одиницю часу;
- середній відсоток вимог, які не були обслужені;
- ймовірність того, що вимогу, яка надійшла до системи, буде прийнято для обслуговування;
- середній час очікування вимоги в черзі;
- закон розподілу часу очікування;
- середня кількість вимог у черзі;
- закон розподілу числа вимог у черзі;
- коефіцієнт завантаження пристрою для обслуговування;
- середня кількість пристроїв, зайнятих обслуговуванням.

Щоб визначити ці параметри, потрібно охарактеризувати СМО, тобто описати та задати такі характеристики:

- вхідний потік вимог (вимоги, які надходять до системи для обслуговування);
- дисципліни постановки вимог у чергу та вибору вимог із неї;
- правила, за якими здійснюється обслуговування;
- вихідний потік вимог (вимоги, які залишають систему);
- режими роботи системи.

Розглянемо ці характеристики більш детально.

2.1.1. Вхідний потік вимог

Для визначення вхідного потоку вимог потрібно зазначити моменти часу їх надходження до системи (закон надходження) і кількість вимог, які можуть надійти одночасно. Закон надходження може бути детермінованим (наприклад, вимога або вимоги надходять до системи у фіксовані моменти часу) або ймовірнісним (проміжки часу між моментами надходження вимог до системи мають рівномірний, експоненціальний або інший заданий закон розподілу).

У загальному випадку вхідний потік вимог описується розподілом імовірностей проміжків часу між моментами надходження до системи двох сусідніх вимог. Здебільшого припускається, що ці проміжки часу є незалежними і мають однаковий розподіл випадкових величин і, таким чином, вимоги утворюють стаціонарний вхідний потік. У класичній теорії масового обслуговування, як правило, розглядається так званий *пуассонівський* (найпростіший) потік вимог, в якому кількість вимог k для будь-якого проміжку часу t має розподіл Пуассона:

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \quad k \geq 0, \quad t \geq 0, \tag{2.1}$$

де λ – інтенсивність потоку вимог (кількість вимог, які надійшли до системи за одиницю часу).

Існує кілька теорем теорії ймовірностей, за допомогою яких можна довести, що більшість потоків вимог є пуассонівськими. Це, зокрема, теореми про суперпозицію незалежних потоків малої інтенсивності, розрідження випадкового потоку та ін. На практиці вважають, що вхідний потік має розподіл Пуассона, якщо за визначений проміжок часу вимоги надходять до системи від великої кількості незалежних джерел. Прикладами таких потоків можуть бути дзвінки абонентів у телефонній мережі або запити до централізованої бази даних від користувачів комп'ютерної мережі.

Для задання вхідного потоку вимог, крім закону розподілу, потрібно визначати кількість вимог, які надходять до системи одночасно. Вимоги можуть надходити до системи по одній або групами, наприклад у метро або до стадіону через вхідні турнікети одночасно можуть зайти кілька чоловік. Системи, до яких вимоги надходять пакетами (більше ніж з однією вимогою), будемо називати системами з груповим потоком вимог.

Кількість вимог, які надходять до системи від якогось джерела (під час моделювання це джерело відтворює генератор вимог), може бути необмеженою або обмеженою. Прикладом системи з обмеженою кількістю вимог є система з відмовами та відновленням (ремонтом) обладнання виробничої дільниці, де в разі відмови обладнання подається вимога на ремонт обладнання до бригади ремонтників. Якщо в такій системі є M одиниць обладнання, то максимально можлива кількість вимог у системі дорівнюватиме M.

Прикладом системи з необмеженою кількістю вимог може бути телефонна мережа, кількість абонентів якої визначити практично неможливо. Якщо кількість вимог безмежна і вони незалежні, то це свідчить про те, що вхідний потік вимог є пуассонівським.

Моделювання пуассонівського потоку

Покажемо, як під час моделювання систем масового обслуговування можна задати пуассонівський потік вимог. Для цього розглянемо найпростіший потік з інтенсивністю λ і позначимо моменти надходження вимог на осі $(0, t)$, як показано на рис. 2.1. Визначимо, який розподіл мають проміжки часу T між моментами надходження двох сусідніх вимог.

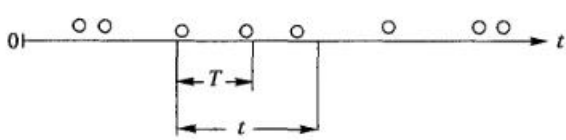


Рис. 2.1. Моменти надходження вимог для пуассонівського потоку

Очевидно, що проміжки часу T – випадкові величини. Знайдемо закон їх розподілу. Функція розподілу $F(t)$ визначає ймовірність того, що випадкова величина T набуде значення, яке менше за t, тобто

$$F(t) = \mathrm{P}, \quad T < t.$$

Нехай t_0 – початок проміжку часу T. Знайдемо ймовірність того, що випадкова величина T буде меншою за t. Для цього потрібно, щоб на проміжок довжиною t, який починається з точки t_0, потрапила хоча б одна вимога. Обчислимо функцію $F(t)$ через імовірність протилежної події, тобто через імовірність P_0 того, що за проміжок часу t до системи не надійде жодної вимоги:

$$F(t) = 1 - P_0.$$

Значення ймовірності P_0 знайдемо за формулою (2.1) за умови, що $\lambda t = 0$:

$$P_0 = \frac{(\lambda t)^0}{0!} e^{-\lambda t} = e^{-\lambda t}.$$

Тоді функція розподілу випадкової величини T матиме вигляд:

$$F(t) = 1 - e^{-\lambda t}, \quad t > 0.$$

Щоб знайти функцію щільності розподілу $f(t)$ випадкової величини T, продиференціюємо функцію $F(t)$ за t:

$$f(t) = \lambda e^{-\lambda t}, \quad t > 0.$$

Це і є функція щільності показникового або експоненціального закону розподілу. Її графік і графік функції розподілу за умови $\lambda = 1$ зображено на рис. 2.2.

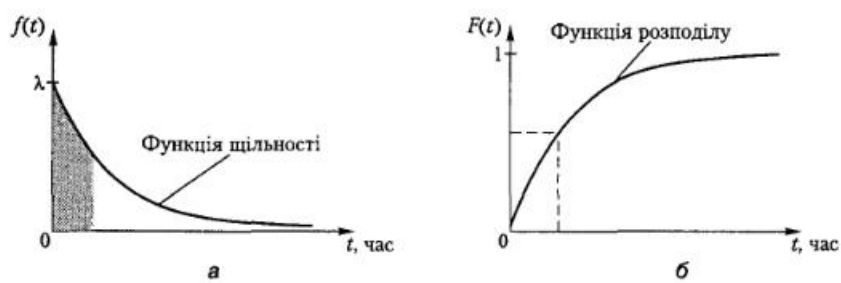


Рис. 2.2. Графіки функцій щільності (*а*) та розподілу (*б*) експоненціального закону: $\lambda = 1$; $p = 0.5$

Отже, щоб отримати пуассонівський потік вхідних вимог, які надходять до системи, достатньо обчислити випадкову величину з експоненціальним розподілом.

Властивості пуассонівського потоку

У теорії масового обслуговування найпростіший потік відіграє таку ж роль, як нормальний закон розподілу випадкових величин у теорії ймовірностей. Випадковий потік вимог, який за своїми характеристиками наближений до найпростішого, утворюється в разі додавання випадкових потоків.

Основними властивостями найпростішого потоку вимог є:

- стаціонарність;
- відсутність післядії;
- ординарність.

Потік є *стаціонарним*, якщо ймовірність надходження певної кількості вимог за деякий проміжок часу τ залежить тільки від довжини цього проміжку τ і параметра λ, і не залежить від місця розташування проміжка на осі часу (рис. 2.3). Відрізки часу τ не повинні перетинатися.

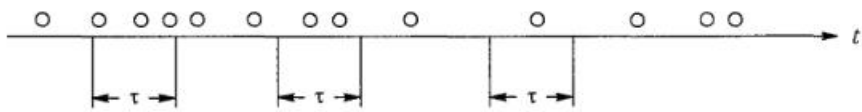


Рис. 2.3. Відображення стаціонарного потоку вимог

У потоці *відсутня післядія*, якщо ймовірність надходження визначеної кількості вимог за деякий проміжок часу τ не залежить від кількості вимог, які надійшли до системи, тобто не залежить від передісторії. Те, що проміжки часу не перетинаються, свідчить про взаємну незалежність протікання процесів у часі.

Потік є *ординарним*, якщо в один і той самий момент часу неможливе надходження двох або більше вимог.

Пуассонівський потік вимог є окремим випадком більш загального потоку Ерланга. Потік Ерланга *r*-го порядку можна отримати шляхом просіювання пуассонівського потоку (рис. 2.4).

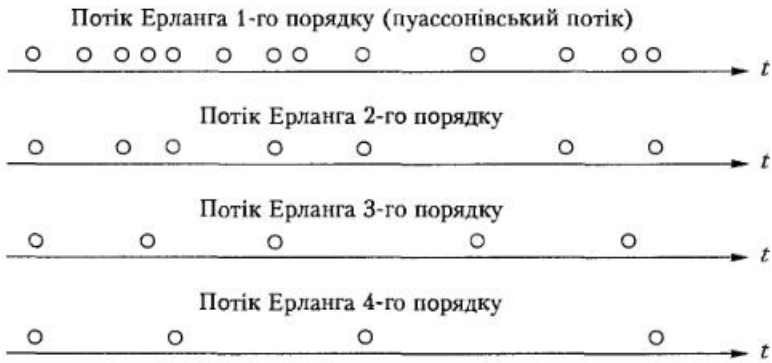


Рис. 2.4. Потік Ерланга *r*-го порядку

На рисунку видно, що для того щоб отримати потік Ерланга *r*-го порядку, досить підсумувати *r* випадкових експоненціально розподілених величин.

2.1.2. Організація черги

Дисципліни постановки вимог у чергу та вибору вимог із неї для обслуговування визначають порядок, за яким вимоги стають у чергу, якщо пристрій для обслуговування зайнятий, та порядок їх виходу з черги для обслуговування – якщо пристрій для обслуговування вільний.

Найпростіша дисципліна обслуговування передбачає поставлення вимог у чергу за порядком їх надходження. Вона має назву *перший прийшов – першим обслужили* (ППО), в англомовній літературі – FIFO (First In First Out). Прикладом черги з такою дисципліною може бути черга до телефону-автомата.

Існує також інший спосіб організації черги, коли для обслуговування вибираються останні в черзі вимоги (*останній прийшов – першим обслужили* (ОППО)), в англомовній літературі – LIFO (Last In First Out)). Цей спосіб також називається стеком або «магазином». Прикладом черги з такою дисципліною обслуговування може бути паром, на якому перевозять авто, – автомобіль, який заїхав на паром перший, виїжджає з нього останнім.

Що стосується правила вибору вимог із черги, то вибір може бути *випадковим* (в англомовній літературі – RANDOM), наприклад вибір куль із барабана для гри в лото. Під час вибору вимог із черги може враховуватись їх пріоритет.

Черга може мати обмеження за довжиною або за часом перебування вимог у ній. Наприклад, якщо в черзі знаходиться більше трьох вимог, то нова вимога, яка надійшла, залишає систему, або вимога залишає систему, якщо час перебування її в черзі становить понад 2 хв. Прикладом черги з обмеженою кількістю місць є бункер, в який надходять заготовки, перш ніж їх буде оброблено верстатом. Буфери даних широко використовуються в комп'ютерній техніці. Під час обміну інформацією між пристроями, які мають різну швидкість обробки даних, інформація накопичується в буфері, а потім використовується пристроєм, що має меншу швидкість. Такі буфери організовуються в системах введення-виведення даних і мультиплексорах. У комп'ютерних мережах буфери створюють для організації черг повідомлень або пакетів.

2.1.3. Правила обслуговування вимог

Правила обслуговування вимог характеризуються часом обслуговування (розподілом часу обслуговування), кількістю вимог, які обслуговуються одночасно, і дисципліною обслуговування.

Обслуговування може бути організоване за допомогою одного або кількох ідентичних пристроїв. У першому випадку система називається *одноканальною*, у другому – *багатоканальною*. Час обслуговування вимог може бути детермінованим або заданим за ймовірнісним законом розподілу.

Якщо пристрої для обслуговування об'єднані в ланцюжок, то система називається багатофазовою, тому що вимоги в ній послідовно проходять кілька фаз обслуговування (наприклад, складальний конвеєр автомобільного заводу).

Дисципліна обслуговування визначає, за яких умов припиняється обслуговування вимог, як обирається для обслуговування наступна вимога, а також що станеться із частково обслуженою вимогою.

Розрізняють *безпріоритетні* і *пріоритетні* дисципліни обслуговування. У разі безпріоритетного обслуговування порядок обслуговування визначається за дисципліною вибору вимоги з черги, наприклад ППО. До безпріоритетних належить циклічна дисципліна обслуговування, яка часто використовується в комп'ютерних системах. Вимога (програма) багаторазово використовує пристрій для обслуговування (процесор) перед тим, як його залишити. Після закінчення кожного циклу обслуговування вимога знову надходить до черги для додаткового обслуговування.

Під час пріоритетного обслуговування для кожної вимоги задається деякий числовий параметр, значення якого визначає її *пріоритет*. Значення пріоритету

може бути незмінним (*статичний* пріоритет) або являти собою функцію, яка залежить від часу перебування вимоги в системі (*динамічний* пріоритет).

Пріоритет може бути також *відносним* або *абсолютним*. Відносний пріоритет передбачає, що надходження вимоги з вищим пріоритетом не перериває обслуговування менш пріоритетної вимоги (обслуговування без переривання). Вимоги з однаковими пріоритетами можуть утворювати черги.

Якщо в системі задається абсолютний пріоритет, то поява вимоги з більш високим пріоритетом переривається обслуговування менш пріоритетної вимоги (обслуговування з перериванням). У таких системах можуть утворюватися вкладені переривання, коли обслуговування вимоги, яка витиснула менш пріоритетну, буде перервано більш пріоритетною вимогою і т. д. Іноді в таких системах обмежують глибину переривання. Перервані вимоги можуть або залишати систему, або знову ставати в чергу для додаткового обслуговування.

Зрозуміло, що дисципліни обслуговування з абсолютними пріоритетами можуть використовуватись тільки для систем з одним пристроєм для обслуговування.

2.1.4. Вихідний потік вимог

Вихідний потік – це потік вимог, які залишають систему, до того ж вони можуть бути як обслуженими, так і необслуженими. Структура формування потоку вихідних вимог більш важлива для багатофазових систем, де вихідний потік одного пристрою для обслуговування (фази обслуговування) є вхідним для іншого. Ймовірнісні характеристики розподілу вимог вихідного потоку в часі залежать від щільності вхідного потоку та параметрів роботи пристроїв для обслуговування.

З теорії масового обслуговування відомо, що вихідний потік вимог СМО з M пристроями з очікуванням для найпростішого вхідного потоку з параметром λ і експоненціального розподілу часу обслуговування з параметром μ є найпростішим потоком з параметром $\lambda' = \min\{\lambda, M\mu\}$. Це дає можливість аналізувати багатофазові системи і мережі СМО, в яких вихідний потік вимог одних систем обслуговування є вхідним для інших. У всіх інших випадках розподіл імовірності вихідних потоків вимог СМО має складнішу ймовірнісну природу і може вивчатись тільки шляхом спостереження за функціонуванням цих СМО під час моделювання.

2.1.5. Режими роботи системи масового обслуговування

На практиці часто доводиться вивчати режими роботи СМО, за допомогою яких описується деякий виробничий процес або система обробки інформації. Якщо в системі пристрої для обслуговування час від часу виходять з ладу, то вводиться поняття *режим відмови*. Під час дослідження деяких систем треба брати до уваги ще один режим – *блокування обслуговування*, пов'язаний з тимчасовим перериванням або сповільненням процесу обслуговування.

Зміна режиму роботи СМО може бути зумовлена зовнішнім впливом (наприклад, тимчасовою відсутністю деталей у технологічному процесі, ремонтом обладнання тощо) або виходом із ладу деякого пристрою системи (наприклад, блока живлення в комп'ютері).

2.2. Типи моделей систем масового обслуговування

У теорії систем масового обслуговування розглядаються тільки такі СМО, параметри ефективності яких можна отримати аналітично в замкненому або числовому вигляді. Для позначення таких моделей СМО часто використовують запис, запропонований Канделом – $X/Y/Z$, де X – розподіл часу прибуття надходження вимог, Y – розподіл часу обслуговування, а Z – кількість пристроїв для обслуговування.

Найпоширенішою моделлю, яка розглядається в теорії масового обслуговування, є модель типу $M/M/1$. Ця модель має тільки один пристрій для обслуговування (цифра 1), і в ній процеси розподілу часу надходження (перша буква M) та обслуговування (друга буква M) є марківськими. Для такої моделі час між двома надходженнями вимог до системи і час їх обслуговування мають експоненціальні розподіли. Модель типу $M/M/1$ може використовуватися, наприклад, для моделювання роботи однопроцесорної системи або стандартного пристрою для введення-виведення інформації (магнітного диска, принтера тощо).

Модель типу $D/D/1$ – детермінована, а модель $D/M/1$ – змішана. Якщо відомостей про систему мало, її модель позначають як $G/G/m$ – модель з будь-якими розподілами ймовірностей випадкових величин і m пристроями для обслуговування.

У теорії масового обслуговування аналітичні результати отримано тільки для моделей типів $D/D/1$, $M/M/1$ і $M/G/1$. Для визначення характеристик моделей з іншими значеннями параметрів СМО потрібно використовувати методи імітаційного моделювання.

У реальних системах не завжди можна описати закони розподілу вхідних потоків вимог і часу їх обслуговування. Для їх визначення потрібно вміти оцінювати характер робочого навантаження системи. Зокрема, у разі моделювання комп'ютерної системи треба знати, коли до системи надходять нові завдання, скільки часу потрібно для виконання процесором кожного з них, як часто програма звертається до пристроїв для введення-виведення інформації. Для цього треба розробити діаграму роботи системи, на якій можна зобразити потоки вхідних завдань у систему, ресурси, до яких вони направляються, а також час обслуговування завдань на цих ресурсах.

Якщо графік робочого навантаження має періодичній характер, то під час спостереження за роботою комп'ютера є можливість отримати представницьку вибірку, яку можна застосовувати для аналізу показників ефективності СМО. Проте моделювання з використанням такого опису робочого навантаження (сценарію) дасть змогу відтворити тільки результати роботи комп'ютерної системи в минулому. На підставі цього не можна визначити, як працюватиме система в майбутньому.

Для того щоб спрогнозувати поведінку системи в майбутньому, потрібно виявити закономірності та визначити один або кілька розподілів імовірностей, а не використовувати необроблені дані, отримані шляхом вимірювання. Якщо для визначення робочого навантаження на систему використовуються розподіли ймовірності, а для аналізу результатів моделювання – відповідні статистичні методи, то отримані результати можна поширити на більший діапазон робочих навантажень, ніж під час використання визначеного сценарію.

2.3. Формула Литтла

У теорії масового обслуговування важливе значення має формула Литтла (*закон збереження стаціонарної черги*), яка дозволяє обчислювати середню кількість вимог, що знаходяться в системі. Щоб отримати формулу Литтла, розглянемо СМО загального виду, яку зображено на рис. 2.5 у вигляді «чорного ящика», і будемо спостерігати за її вхідними та вихідними потоками вимог.

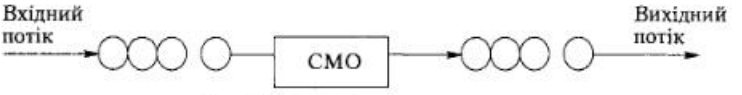


Рис. 2.5. СМО загального виду

Процес $\alpha(t)$ – деякий випадковий процес надходження вимог до системи за проміжок часу $(0, t)$. Процес $\delta(t)$ визначає вихідний потік вимог із системи на цьому ж проміжку. Відобразимо обидва випадкових процеси у вигляді графіків, наведених на рис. 2.6.

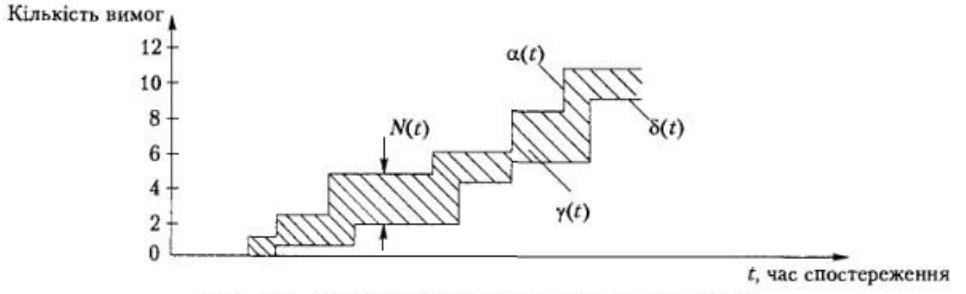


Рис. 2.6. Вхідні та вихідні випадкові процеси в СМО

Кількість вимог, що знаходяться в системі в будь-який момент часу t, можна знайти як $N(t) = \alpha(t) - \delta(t)$.

Заштрихована площа між двома кривими $\gamma(t)$ визначає загальну роботу (добуток кількості вимог на час перебування їх у СМО) на проміжку часу $(0, t)$, яка вимірюється у вимогах за секунду.

Інтенсивність надходження вимог до СМО за час спостереження $(0, t)$ можна визначити як

$$\lambda_t = \frac{\alpha(t)}{t}, \tag{2.2}$$

а середній час перебування вимог у системі за той же проміжок часу як

$$T_t = \frac{\gamma(t)}{\alpha(t)}. \tag{2.3}$$

Середня кількість вимог, що перебували в системі за проміжок часу $(0, t)$,

$$\overline{N}_t = \frac{\gamma(t)}{t}. \tag{2.4}$$

Використовуючи вирази (2.2)–(2.4), отримаємо таку формулу:

$$\overline{N}_t = \lambda_t T_t. \tag{2.5}$$

Для того щоб СМО була в стані рівноваги, потрібно, щоб середній час перебування вимог у системі був більшим за середній час їх обслуговування. Припустимо, що для СМО, яка розглядається, $\lambda = \lim\limits_{t\to\infty} \lambda_t$ і $T = \lim\limits_{t\to\infty} T_t$, де λ – інтенсивність надходження, а T – середній час перебування вимог у системі. У цьому випадку існує також межа для середньої кількості вимог, які знаходяться в системі, тобто $\overline{N} = \lim\limits_{t\to\infty} \overline{N}_t$.

Тоді з формули (2.5) отримаємо формулу Литтла у такому вигляді:

$$\overline{N} = \lambda T.$$

Отже, для будь-якого закону розподілу проміжків часу між двома моментами надходження вимог і будь-якого розподілу часу їх обслуговування, кількості пристроїв для обслуговування та дисципліни обслуговування середню кількість вимог, що знаходяться в СМО, визначають через інтенсивність надходження та середній час перебування вимог у системі.

Інтуїтивне доведення формули Литтла базується на тому, що кількість вимог $\overline{N}$ у системі в момент надходження нової вимоги, буде такою ж, як і в момент, коли вимога залишає систему. Це свідчить про те, що СМО перебуває в стані рівноваги або сталому стані, тобто вимоги не можуть знаходитись у системі нескінченно довго і завжди залишають її. Як бачимо, під час виведення формули Литтла ніяких обмежень на тип СМО немає. Можна, наприклад, вважати, що СМО складається тільки з однієї черги або з одного пристрою для обслуговування.

2.4. Одноканальні системи масового обслуговування

Розглянемо одноканальну СМО з одним пристроєм для обслуговування S і чергою до нього q, яку зображено на рис. 2.7.

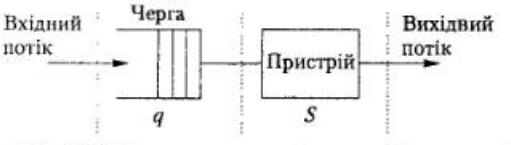


Рис. 2.7. СМО з одним пристроєм для обслуговування

Якщо позначити через w середній час перебування вимоги в черзі, то з формули Литтла можна отримати середню кількість вимог у черзі:

$$\overline{N}_q = \lambda w.$$

Якщо позначити середній час обслуговування вимоги в пристрої через $\overline{x}$ і розглядати СМО як таку, що має один пристрій, то, використовуючи формулу Литтла, можна знайти середню кількість вимог у пристрої для обслуговування:

$$\overline{N}_s = \lambda \overline{x}.$$

Для СМО з одним пристроєм для обслуговування завжди має місце рівність $T = w + \bar{x}$, де T – середній час перебування вимоги в системі.

Коефіцієнт завантаження (коефіцієнт використання) пристрою для обслуговування ρ можна визначити, якщо розділити інтенсивність надходження вимог до системи (λ) на швидкість обслуговування цих вимог у пристрої (μ), тобто

$$\rho = \frac{\lambda}{\mu},$$

де

$$\mu = \frac{1}{\bar{x}}.$$

Тоді

$$\rho = \lambda\bar{x}, \qquad 0 \le \rho \le 1.$$

Покажемо, що ρ – це ймовірність того, що під час надходження вимоги до системи пристрій для обслуговування буде зайнятим. Розглянемо як задовгий проміжок часу τ. Відповідно до закону великих чисел, очікувана кількість вимог на проміжку часу τ з імовірністю 1 буде приблизно дорівнювати $\lambda\tau$. Позначимо через p_0 імовірність того, що в деякий випадковий момент часу вимога, що надійшла, застане пристрій вільним. Тоді можна стверджувати, що на проміжку часу t пристрій буде зайнятим $\tau(1 - p_0)$. Таким чином, з імовірністю 1 кількість обслужених за цей проміжок часу вимог буде дуже близькою до величини $\tau(1 - p_0)/\bar{x}$. Якщо всі вимоги, які надійшли за проміжок часу τ, вважати обслуженими, то

$$\lambda\tau \approx \frac{\tau(1 - p_0)}{\bar{x}}.$$

Таким чином, маємо $\lambda\bar{x} = 1 - p_0$, якщо $\tau \to \infty$, або $\rho = 1 - p_0$, тобто з імовірністю ρ вимога застає пристрій для обслуговування зайнятим, або ρ дорівнює частці часу, протягом якого пристрій був зайнятим.

Введемо коефіцієнт варіації C як відношення стандартного відхилення $\sigma_{\bar{x}}$ від середнього значення до середнього значення $\bar{x}$:

$$C = \frac{\sigma_{\bar{x}}}{\bar{x}}.$$

Для експоненціального закону розподілу коефіцієнт варіації $C = 1$, оскільки $\bar{x}$ і $\sigma_{\bar{x}}$ для цього закону дорівнюють λ. Для регулярного детермінованого закону розподілу $C = 0$ ($\sigma_{\bar{x}} = 0$). Таким чином, для моделі СМО типу $G/G/1$ з одним пристроєм і при довільних законах надходження та обслуговування вимог середня кількість вимог визначається як

$$\bar{N} = \rho + \frac{\rho^2(1 + C^2)}{2(1 - \rho)}.$$

Цей вираз можна отримати [21], якщо відзначити, що кількість вимог, які залишилися в системі після виходу деякої вимоги, є напівмарківським процесом з укладеним ланцюгом Маркова, який визначено в моменти виходу вимоги.

Використовуючи результат Хінчина–Полячека, можна отримати середній час перебування вимог у одноканальній СМО за формулою

$$T = \bar{x}\left[1 + \frac{\rho(1+C^2)}{2(1-\rho)}\right]. \qquad (2.6)$$

Із формули (2.6) видно, що середній час перебування вимоги в системі залежить тільки від математичного сподівання і стандартного відхилення часу обслуговування. Таким чином, час чекання визначається як

$$w = \frac{\bar{x}\rho(1+C^2)}{2(1-\rho)}.$$

Зазвичай потрібно знати нормований час чекання:

$$\frac{w}{\bar{x}} = \frac{\rho(1+C^2)}{2(1-\rho)}.$$

Для моделей типу $M/M/1$

$$\frac{w}{\bar{x}} = \frac{\rho}{(1-\rho)}.$$

Для моделей типу $M/D/1$

$$\frac{w}{\bar{x}} = \frac{\rho}{2(1-\rho)}.$$

Таким чином, система з регулярним законом обслуговування характеризується середнім часом чекання, удвоє меншим, ніж для системи з показовим законом обслуговування. Це закономірно, оскільки час перебування вимог у системі та їх кількість пропорційні дисперсії часу обслуговування.

2.5. Багатоканальні системи масового обслуговування

Аналіз багатоканальних СМО (з кількома однаковими пристроями для обслуговування) (рис. 2.8), на відміну від одноканальних, є набагато складнішим. За допомогою теорії масового обслуговування можна отримувати аналітичні залежності в замкнутому вигляді для розрахунків характеристик роботи багатоканальної СМО в стаціонарному режимі роботи, однак лише для моделі типу $M/M/m$. Для СМО з іншими законами розподілу часу надходження та обслуговування вимог використовують числові методи.

Рис. 2.8. Багатоканальна СМО

Для системи, яка складається з m однакових пристроїв для обслуговування, коефіцієнт завантаження $\rho = \lambda\bar{x}/m$. Для багатоканальної СМО значення ρ можна трактувати як математичне сподівання частки зайнятих пристроїв.

Приклад 2.1

Як приклад розглянемо часову діаграму роботи багатоканальної СМО з двома пристроями (ПР1 і ПР2) і двома позиціями 1, 2 для чекання в черзі (рис. 2.9). Час надходження вимоги до системи і час, коли вона залишила систему, наведено поряд із номером вимоги відповідно в нижній і верхній частинах рис. 2.9. Час спостереження за СМО ($T_{сп}$) становить 55 хв.

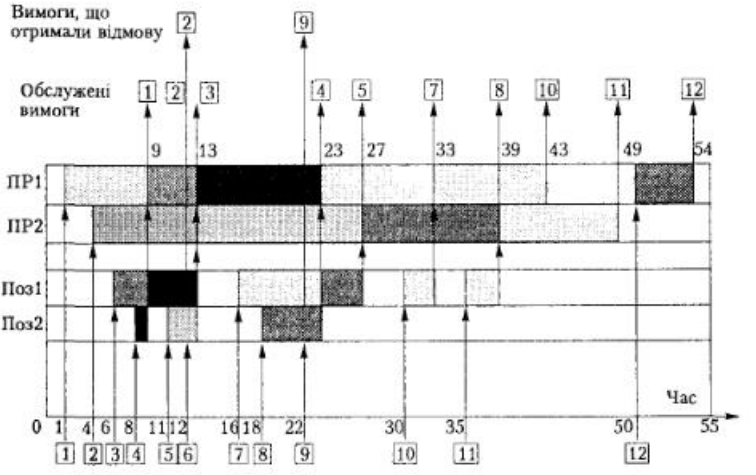


Рис. 2.9. Діаграма роботи багатоканальної СМО

На основі цієї діаграми розрахуємо значення деяких характеристик ефективності роботи СМО.

1. Імовірність обслуговування вимоги:

$$P_{об} = \frac{N_{об}}{N} = \frac{10}{12} = 0{,}83,$$

де $N_{об}$ і N — відповідно кількість обслужених вимог і загальна кількість вимог.

2. Пропускна здатність СМО:

$$X = \frac{N_{об}}{T_{сп}} = \frac{10}{55} = 0{,}18 \text{ вимоги/хв},$$

де $T_{сп}$ — час спостереження за системою.

3. Імовірність відмови в обслуговуванні:

$$P_{від} = \frac{N_{від}}{N} = \frac{2}{12} = 0{,}166,$$

де $N_{від}$ — кількість вимог, яким відмовлено в обслуговуванні.

4. Імовірність того, що вимога застане обидва пристрої вільними:

$$P_0 = \frac{T_{вільн}}{T_{сп}} = \frac{3}{55} = 0{,}05,$$

де $T_{вільн}$ — час, протягом якого обидва пристрої були вільними.

5. Імовірність того, що обслуговуванням зайнятий тільки один пристрій із двох:

$$P_1 = \frac{T_з^1 + T_з^2}{T_{сп}} = \frac{7+6}{55} = 0{,}236,$$

де $T_з^1, T_з^2$ — час, протягом якого були зайнятими відповідно перший і другий пристрої.

6. Імовірність того, що обслуговуванням зайняті обидва пристрої:

$$P_2 = \frac{T_з^{1+2}}{T_{сп}} = \frac{39}{55} = 0{,}709,$$

де $T_з^{1+2}$ — час, протягом якого обидва пристрої були зайнятими.

7. Середня кількість пристроїв, зайнятих обслуговуванням:

$$N_{пр} = 0 \cdot P_0 + 1 \cdot P_1 + 2 \cdot P_2 = 1 \cdot \frac{13}{55} + 2 \cdot \frac{39}{55} = 1{,}654.$$

8. Імовірність того, що в черзі відсутні вимоги:

$$P_{чер}^0 = \frac{T_{чер}^0}{T_{сп}} = \frac{(6-0)+(16-13)+(30-27)+(35-33)+(55-39)}{55} = \frac{30}{55} = 0{,}545,$$

де $T_{чер}^0$ — час, протягом якого в черзі не було вимог.

9. Імовірність того, що в черзі є лише одна вимога:

$$P_{чер}^1 = \frac{T_{чер}^1}{T_{сп}} = \frac{(6-4)+(11-9)+(18-16)+(27-23)+(33-30)+(39-35)}{55} = \frac{17}{55} = 0{,}309,$$

де $T_{чер}^1$ — час, протягом якого в черзі перебувала лише одна вимога.

10. Імовірність того, що в черзі знаходяться дві вимоги:

$$P_{чер}^2 = \frac{T_{чер}^2}{T_{сп}} = \frac{(9-8)+(13-11)+(23-18)}{55} = \frac{8}{55} = 0{,}145,$$

де $T_{чер}^2$ — час, протягом якого в черзі було дві вимоги.

11. Середня кількість вимог у черзі:

$$N_{чер} = 0 \cdot P_{чер}^0 + 1 \cdot P_{чер}^1 + 2 \cdot P_{чер}^2 = 0 + 1 \cdot \frac{17}{55} + 2 \cdot \frac{8}{55} = 0{,}6.$$

12. Середній час перебування вимог у черзі:

$$t_{чер} = \frac{\sum_{i=1}^{10} t_i^{чер}}{N_{об}} = \frac{0+0+3+5+2+7+9+3+11+0}{10} = \frac{40}{10} = 4 \text{ хв},$$

де $t_i^{чер}$ — час перебування i-ї вимоги в черзі ($i = 1, 2, \ldots$).

13. Середній час перебування вимог у черзі без урахування вимог, які не чекали:

$$t_{чер} = \frac{\sum_{i=1}^{7} t_i^{чер}}{N_{об}(-0)} = \frac{3+5+2+7+9+3+11}{7} = \frac{40}{7} = 5{,}714 \text{ хв},$$

де $N_{об}(-0)$ — кількість вимог, які чекали в черзі.

14. Середній час обслуговування вимоги пристроями:

$$t_{об} = \frac{\sum_{i=1}^{10} t_i^{об}}{N_{об}} = \frac{92}{10} = 9{,}2 \text{ хв},$$

де $t_i^{об}$ — час обслуговування i-ї вимоги в СМО ($i = 1, 2, ...$).

15. Загальний середній час перебування вимоги в СМО:

$$T = t_{об} + t_{чер} = 4 + 9{,}2 = 13{,}2 \text{ хв}.$$

16. Середня кількість вимог у системі обслуговування:

$$\overline{N} = N_{пр} + N_{чер} = 1{,}654 + 0{,}6 = 2{,}254.$$

На рис. 2.10 зображено гістограму часу надходження вимог до СМО та його апроксимацію за експоненціальним законом розподілу (значення критерію Колмогорова–Смирнова для даної вибірки дорівнює 0,1345275). На гістограмі видно, що кількість вимог, які надійшли до системи, є недостатньою для статистичного оцінювання. Тому гіпотезу про експоненціальний закон розподілу надходження вимог до СМО потрібно відхилити.

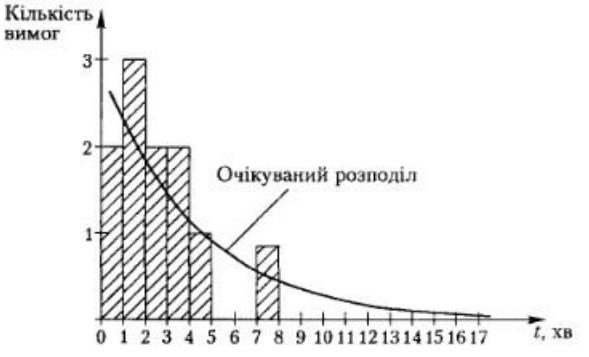


Рис. 2.10. Гістограма часу надходження вимог до СМО

Отримані числові значення показників СМО мають ілюстративний характер і вказують на те, які статистичні дані необхідно фіксувати в разі моделювання СМО, щоб визначити її характеристики ефективності.

У теорії масового обслуговування розглядаються багатоканальні СМО типу $M/M/m$ з m пристроями для обслуговування двох типів:

- з відмовами, коли зайняті всі m пристроїв і вимога отримує відмову в обслуговуванні;
- з чеканням, коли зайняті всі m пристроїв і вимога чекає в черзі (кількість місць чекання s), якщо в системі знаходиться $m + s$ вимог і надходить нова вимога, то вона отримує відмову.

Для кожного з цих двох випадків можна побудувати системи диференціальних рівнянь, які описують усі стани СМО, а потім розв'язати ці рівняння за умови, що система функціонує в стаціонарному режимі. Наведемо основні формули, потрібні для розрахунків параметрів СМО типу $M/M/m$ [41].

1. Імовірність того, що всі пристрої для обслуговування вільні:

$$P_0 = \frac{1}{\sum_{k=0}^{m-1} \frac{(\lambda\bar{x})^k}{k!} + \frac{(\lambda\bar{x})^m}{(m-1)!\,(m-\lambda\bar{x})}}, \quad \text{якщо} \quad \lambda\bar{x} < 1.$$

2. Імовірність того, що зайнято обслуговуванням k пристроїв або в системі знаходиться k вимог:

$$P_k = \frac{(\lambda\bar{x})^k}{k!} P_0, \quad 1 \le k < m.$$

3. Імовірність того, що всі пристрої зайняті ($k \ge m$). Позначимо цю ймовірність через π:

$$\pi = \frac{(\lambda\bar{x})^m P_0}{(m-1)!\,(m-\lambda\bar{x})}, \quad \frac{\lambda\bar{x}}{m} < 1.$$

4. Імовірність того, що всі пристрої зайняті обслуговуванням і s вимог знаходяться в черзі:

$$P_{m+s} = \frac{(\lambda\bar{x})^{m+s}}{m!\,m^s} P_0, \quad s > 0.$$

5. Імовірність того, що час перебування вимог у черзі перевищує деяку задану величину t:

$$\mathrm{P}(w > t) = \pi e^{-\frac{1}{\bar{x}}(m-\lambda\bar{x})t}.$$

6. Середня довжина черги:

$$\bar{N}_q = \frac{\lambda\bar{x} P_m}{m\left(1 - \frac{\lambda\bar{x}}{m}\right)^2}.$$

7. Середня кількість вільних від обслуговування пристроїв:

$$N_{\text{вільн}} = \sum_{k=0}^{m-1} \frac{m-k}{k!} (\lambda\bar{x})^k P_0.$$

8. Середня кількість зайнятих обслуговуванням пристроїв:

$$N_{\text{пр}} = m - N_{\text{вільн}}.$$

9. Середній час чекання вимогою початку обслуговування в системі:

$$w = \frac{\pi\bar{x}}{m - \lambda\bar{x}}, \quad \frac{\lambda\bar{x}}{m} < 1.$$

Використовуючи вищенаведені формули, можна розрахувати параметри СМО типу $M/M/m$, а потім порівняти їх з результатами імітаційного моделювання.

2.6. Основи дискретно-подійного моделювання систем масового обслуговування

Вивчаючи роботу СМО та розглядаючи алгоритми їх моделювання, можна визначити, як побудовані базові конструкції дискретно-подійних мов моделювання, наприклад такі, як генератори вимог, пристрої для обслуговування, термінатори вимог і хронологічні списки подій. Усі програмні засоби імітаційного моделювання дискретних систем включають засоби моделювання СМО будь-якої складності. Одна з перших мов імітаційного моделювання GPSS спочатку також розроблялась як мова моделювання СМО.

Під час моделювання СМО необхідно відтворити її роботу в модельному часі та організувати збір статистичних даних, потрібних для обчислення показників ефективності системи. Алгоритми моделювання можна побудувати, використовуючи формальні моделі, описані в розділі 1.8.3, тобто шляхом імітації станів модельованої системи. Дискретно-подійне моделювання ґрунтується на принципі просування модельного часу від події до події, якщо ці події упорядковані у модельному часі. Для реалізації цього принципу використовуються списки подій, де кожній події відповідає підпрограма обробки події, яка викликається в разі її настання.

2.6.1. Деякі визначення, потрібні під час моделювання систем масового обслуговування

Визначимо основні поняття і терміни, які використовуються під час моделювання СМО.

Об'єкт – будь-який елемент або компонент СМО, який має бути заданим явно в моделі СМО (наприклад, пристрій для обслуговування, клієнт, машина).

Список – множина (постійна або тимчасова) пов'язаних між собою об'єктів, упорядкована згідно з певним логічним правилом (наприклад, усі вимоги, які знаходяться в певний час у черзі, упорядковані за принципом «перший прийшов – першим обслужили» або за пріоритетами).

Подія – миттєва зміна стану системи, наприклад прибуття нової вимоги, або закінчення обслуговування вимоги в системі.

Повідомлення про подію – інформація про подію, яка сталась або станеться, і дані, необхідні для обробки події (запис про подію має включати інформацію про тип і час події).

Список подій – перелік намічених майбутніх подій, упорядкованих за часом їх виникнення, відомий також як список майбутніх подій (СМП).

Дія – операція, яка виконується протягом зазначеного проміжку часу (наприклад, час обслуговування або час між надходженнями вимог), для якої відомі час початку і закінчення (хоча цей час може бути визначено в термінах статистичного розподілу).

Затримка – тривалість невизначеного проміжку часу, для якого невідомо заздалегідь, коли він закінчується (наприклад, затримка вимоги в черзі за правилом «останній прийшов – першим обслужили», для якого початок обслуговування залежить від майбутніх надходжень).

Модельний час – позитивна зростаюча величина, яка відображає перебіг часу в імітаційній моделі.

Годинник – змінна, яка відображає зміну модельного часу, у прикладах – *годинник* (CLOCK).

Дискретно-подійне моделювання – це моделювання роботи системи в дискретні моменти часу, коли настають певні події, які відображають послідовність змін станів системи в часі. Розглянуті системи є динамічними, тобто змінюються в часі. Тому стан системи, властивості об'єкта і число активних об'єктів, параметрів, дій і затримок – функції часу, які постійно змінюються в процесі моделювання.

Для СМО з одним пристроєм для обслуговування події відбуваються в момент надходження вимоги до системи і в кінці її обслуговування пристроєм. Початок обслуговування – це умовна подія, яка залежить від стану пристрою (зайнятий або вільний) і числа вимог, що знаходяться в черзі. Затримку іноді називають умовним очікуванням, а дію – безумовним. Дії в такій системі характеризуються часом між надходженнями вимог і часом їх обслуговування пристроєм. Завершення дії – *первинна* подія, для керування якою в СМП уміщується повідомлення. Керування затримкою пов'язане з уміщенням об'єкта в інший список, який, можливо, відтворює чергу, де має місце затримка до того часу, коли умови, що склались у системі, дають змогу обробити вимоги. Закінчення затримки іноді називають *умовною* або *вторинною* подією, але такі події не зазначаються у відповідних повідомленнях про події та не з'являються в СМП.

2.6.2. Простір станів системи масового обслуговування

На рис. 2.11 зображено діаграму станів СМО з одним пристроєм для обслуговування. На вхід системи в моменти часу $t_j^{в}$ ($j = 1, 8, ...$) надходять вимоги. Момент початку обслуговування вимоги пристроєм позначено як $t_j^{п}$, а *момент закінчення* – $t_j^{к}$. Момент виходу вимоги із системи $t_j^{вих}$ співпадає з моментом закінчення обслуговування вимоги пристроєм $t_j^{к}$. Якщо в момент надходження вимоги до системи пристрій для обслуговування зайнятий, то вона змушена буде чекати обслуговування в черзі, як, наприклад, вимоги під номерами 2, 4, 5, 6, 7, 8. Стан черги визначається кількістю вимог і позицією кожної з них у черзі. Під час звільнення пристрою перша вимога з черги надходить для обслуговування. Таким чином, простір станів СМО визначається станами пристрою для обслуговування та черги. Зміна станів системи пов'язана з подіями надходження нової вимоги та зайняттям або звільненням пристрою для обслуговування. Стан черги також залежить від стану пристрою для обслуговування. З кожною вимогою j асоційовано кілька подій: поява вимоги – $c_j^{в}$, початок обслуговування – $c_j^{п}$, кінець обслуговування – $c_j^{к}$. Тоді впорядкована множина подій $C = \{c_j^{в}, c_j^{п}, c_j^{к}\}$, $j = 1, 2, ...$, у модельному часі t описує поведінку СМО.

На рис. 2.12 процес обслуговування вимог у СМО з одним пристроєм для обслуговування зображено більш детально. Оскільки будь-яка послідовність станів, упорядкована в часі, є процесом, то обслуговування вимоги в СМО визначається як послідовність станів СМО на проміжку часу від моменту надходження вимоги – $t_j^{в}$ до моменту $t_j^{к}$, коли вона залишає систему.

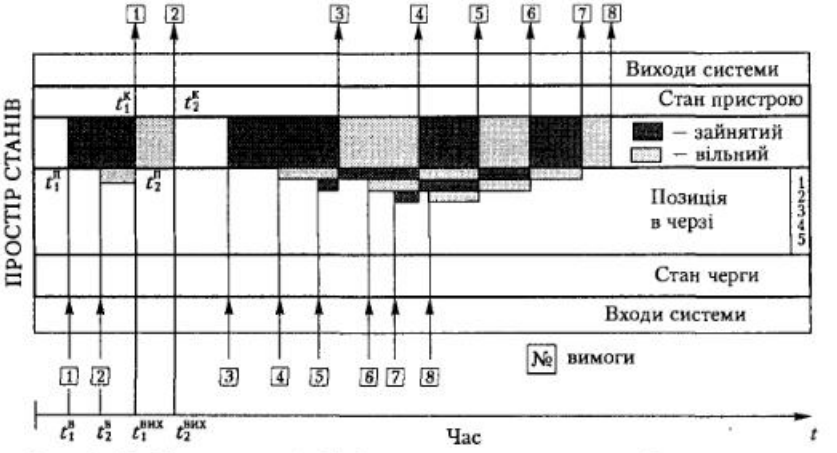


Рис. 2.11. Діаграма станів СМО з одним пристроєм для обслуговування

Рис. 2.12. Процес обслуговування вимог у СМО з одним пристроєм для обслуговування

Перевірка умови зайнятості пристрою $u_з$ у момент надходження вимоги j визначає можливість її негайного обслуговування. Якщо пристрій зайнятий, то вимога змушена чекати в черзі деякий час $t_j^{ч} = t_j^{п} - t_j^{в}$ (функція дії ЧЕКАННЯ – $d(t_j^{ч})$, яка залежить від умови зайнятості пристрою $u_з$). Час обслуговування визначається як $t_j^{об} = t_j^{к} - t_j^{п}$ (функція дії ОБСЛУГОВУВАННЯ – $d(t_j^{об})$, яка пов'язана з умовою звільнення пристрою $u_{зв}$). Загальний час перебування вимог у СМО $t_j^{пер} = t_j^{ч} + t_j^{об}$. Тоді тривалість чекання в черзі є функцією дії для черги, а тривалість обслуговування пристроєм залежить від функції дії пристрою. Упорядковані пари елементів $\langle t_j^{ч}, t_j^{об} \rangle$, $j = 1, ..., 8$, визначають процес обслуговування вимоги в СМО, тобто процес можна визначити як *«життєвий цикл»* вимоги в СМО.

Аналогічно, наприклад, можна визначити процес появи вимог у СМО як упорядковані пари значень $\langle j, t_j^{появ} \rangle$, де j – номер вимоги, яка надійшла до системи для обслуговування, а $t_j^{появ} = t_j^{в} - t_{j-1}^{в}$ (функція дії НАДХОДЖЕННЯ – $d(t_j^{появ})$, яка пов'язана з умовою появи вимоги $u_{появ}$). Умовою початку дії є умова перевірки стану системи (пристрій зайнятий або вільний, надійшла вимога або ні). Отже,

процес починається тільки тоді, коли змінюється стан системи та виконуються функції дії. Зміна стану визначається настанням певної події, а початок виконання функції дії – здійсненням певних умов.

Якщо під час зміни модельного часу щоразу перевіряти множину умов $U = \{u_{появ}, u_з, u_{зв}\}$ для кожної вимоги j, щоб почати деяку дію із множини $D = \{d_j^{появ}, d_j^{ч}, d_j^{об}\}$, $j = 1, 2, ...$, то пари $\langle U, D \rangle$ теж описуватимуть поведінку СМО.

2.6.3. Приклад побудови моделі системи масового обслуговування

Уявімо магазин з одним продавцем, який одночасно є і касиром. Покупця, що прийшов у магазин і застав продавця вільним, негайно обслуговують. Продавець переходить у стан «зайнятий». Якщо під час обслуговування приходить інший покупець (покупці), то він стає в чергу. Закінчивши обслуговування, продавець переходить у стан «вільний».

Якщо в черзі є покупці, то продавець «вибирає» для обслуговування першого з них. Черга зменшується на одиницю. Всі інші покупці в черзі, якщо вони є, зміщуються на одну позицію вперед. За відсутності покупців у черзі продавець залишається в стані «вільний» до приходу наступного покупця.

У табл. 2.1 наведено проміжки часу між приходами покупців у магазин і проміжки часу, потрібні для їх обслуговування продавцем, а на рис. 2.11 представлене «ручне» графічне моделювання СМО з одним пристроєм.

Таблиця 2.1. Параметри СМО

Назва параметра	Значення за номером покупця							
	1	2	3	4	5	6	7	8
Час надходження $t_j^{вх}$	2	2	7	3	2	3	1	1,5
Час обслуговування $t_j^{об}$	4	2	6	5	3	2	2	1

2.6.4. Алгоритм моделювання систем масового обслуговування

Основою побудови моделі є об'єктний підхід, який передбачає створення моделі системи як множини об'єктів, що взаємодіють між собою. У цій моделі можна виділити об'єкти – вимоги (*покупці*) і деякий ресурс (R) – пристрій для обслуговування (статичний об'єкт *продавець*). Якщо вимога претендує на ресурс, зайнятий у даний момент часу, то вона стає в чергу. Черга також є окремим об'єктом – списком, пов'язаним з ресурсом. Для системи обслуговування введемо також правило обслуговування «перший прийшов – першим обслужили».

Під час моделювання для кожної пари «вимога–ресурс» потрібно з'ясувати, як довго вимога j буде використовувати ресурс R, тобто треба зазначити проміжок часу між моментами призначення ресурсу R вимозі j і звільнення цього ресурсу. Однак перш ніж ресурс буде призначено вимозі j, на нього повинен надійти запит. У загальному випадку вимога може чекати в черзі до призначення ресурсу.

Опишемо алгоритм роботи системи обслуговування з погляду на «життєвий цикл» покупця, тобто від моменту його приходу до магазину до моменту виходу з нього. Оскільки покупці приходять до магазину безперервно протягом деякого періоду часу спостереження за системою (час, упродовж якого моделюється система), то потрібно відтворити потік покупців шляхом «створення» їх у моделі (генерування в деякі моменти часу – моментів їх приходу до магазину). Для створення об'єктів *покупець* використовується спеціальна підпрограма генерування (у мові GPSS [68] цій підпрограмі відповідає блок GENERATE). Наведемо алгоритм її роботи.

1. Створити динамічний об'єкт *покупець*. Такий об'єкт – це структура даних, яка включає в себе такі поля: номер покупця – j, момент його приходу – $t_j^{\text{вх}}$, а також, якщо необхідно, властивості покупця або його атрибути, (наприклад, пріоритет покупця). Потрібно також запланувати подію приходу покупця j на момент часу $t_j^{\text{вх}}$, тобто записати повідомлення про подію в СМП.
2. Запланувати наступну подію для покупця j – запит-призначення ресурсу R (*продавець*) на момент часу $t_j^{\text{вх}}$. Запланувати прихід наступного об'єкта – *покупець* $j+1$, тобто визначити подію приходу наступного покупця $t_{j+1}^{\text{вх}} + t_j^{\text{вх}}$.

Процес обробки вимоги ресурсом доцільно розподілити між трьома підпрограмами.

Перша – це підпрограма запиту і призначення ресурсу R вимозі j (у мові GPSS цій підпрограмі відповідає блок SEIZE). Алгоритм її роботи такий:

1. Якщо ресурс R (об'єкт *продавець*) може бути відразу призначеним для вимоги j, то змінити стан ресурсу R на «зайнятий». Запам'ятати момент початку обслуговування вимоги $t_j^{\text{п}}$ і передати керування підпрограмі обслуговування вимоги j.
2. Якщо ресурс R зайнятий, то поставити вимогу j в чергу до ресурсу R.

Друга – це підпрограма обслуговування вимоги j (у мові GPSS цій підпрограмі відповідає блок ADVANCE). Алгоритм її роботи дуже простий. Вона повинна визначити подію, яка настає після закінчення обслуговування вимоги j як $t_j^{\text{к}} = t_j^{\text{п}} + t_j^{\text{об}}$ (де $t_j^{\text{об}}$ – час обслуговування в пристрої), тобто створити повідомлення про подію в СМП, для того щоб передати керування підпрограмі звільнення ресурсу R вимогою j.

Третя – підпрограма звільнення ресурсу R вимогою j (у мові GPSS цій підпрограмі відповідає блок RELEASE). Алгоритм її роботи такий.

1. Змінити стан ресурсу R на «вільний». Передати керування підпрограмі знищення вимоги.
2. Перевірити наявність у черзі вимог до ресурсу R. Якщо вони є, то вибрати вимогу з черги і запланувати для неї подію запиту і призначення ресурсу R.

Підпрограма знищення вимоги (у мові GPSS цій підпрограмі відповідає блок TERMINATE) потрібна для знищення структури даних, яка створюється для кожної вимоги.

Крім вищенаведених підпрограм потрібна програма керування всім процесом моделювання (ПКМ), яка запускає процес моделювання і контролює пересування кожної вимоги («життєвий цикл») під час моделювання шляхом виклику зазначених підпрограм обробки окремих подій. Інше призначення цієї програми –

вести список упорядкованих у часі подій – СМП і просувати годинник модельного часу від події до події. У мові GPSS функції ПКМ виконує програма інтерпретатор.

Список майбутніх подій містить інформацію про всі події, які мають відбутись. Моделювання з використанням СМП гарантує, що всі події відбуватимуться в хронологічному порядку.

Планування настання кожної майбутньої події означає, що на початку кожної дії обчислюється (або встановлюється, наприклад на основі заданого статистичного розподілу) її тривалість. Це необхідно для того, щоб визначити час закінчення дії і занести цю інформацію (у першу чергу тривалість дії) у СМП. У реальному світі заздалегідь запланувати настання більшості подій неможливо, вони просто відбуваються як, наприклад, випадкові відмови устаткування або випадкові приходи покупців. Під час моделювання для кожної такої події потрібно зазначити кінцевий момент пов'язаної з нею деякої дії.

У будь-який визначений час моделювання $t^{\text{м}}$ СМП містить дані про всі попередньо заплановані події та пов'язані з цими подіями моменти часу $t_1^{\text{м}}, t_2^{\text{м}}, \ldots$. У СМП події упорядковані в хронологічному порядку, тобто моменти часу настання подій задовольняють умови

$$t^{\text{м}} < t_1^{\text{м}} < t_2^{\text{м}} < t_3^{\text{м}} < \ldots < t_n^{\text{м}}.$$

Час $t^{\text{м}}$ – це поточне значення часу моделювання (показання годинника модельного часу). На початку моделювання подія, пов'язана з моментом часу $t_1^{\text{м}}$, є майбутньою подією, тобто ця подія відбуватиметься першою. Після того як показання годинника, що відображає моменти зміни станів системи під час моделювання, зміниться з $t^{\text{м}}$ на $t_1^{\text{м}}$, запланована для майбутнього виконання подія вилучається з СМП і виконується підпрограма події. Виконання підпрограми майбутньої події означає, що відображено новий стан системи в момент часу $t_1^{\text{м}}$, який формується на основі попереднього стану моделі в момент часу $t^{\text{м}}$ і характеру майбутньої події. У момент $t_1^{\text{м}}$ зазначити всі майбутні події неможливо, але протягом всього процесу моделювання будь-яка запланована подія одразу ж поміщається в СМП.

Після того як нове відображення стану системи в момент часу $t_1^{\text{м}}$ було модифіковано, годинник модельного часу просувається до моменту часу $t_2^{\text{м}}$ настання наступної майбутньої події і виконується підпрограма цієї події. Цей процес повторюється до закінчення моделювання. Послідовність дій, які потрібно виконати для того, щоб годинник модельного часу був переведений і відображався новий стан системи, називається алгоритмом, який планує події або перебіг часу від події до події. На рис. 2.13 зображено структурну схему імітаційної моделі.

На початку моделювання (час моделювання $t_0^{\text{м}} = 0$) ПКМ передає керування підпрограмі генерування об'єктів, що визначає момент приходу першого покупця і намічає подію запиту і призначення ресурсу R у СМП на момент часу $t_1^{\text{м}} = t_1^{\text{вх}}$. Оскільки інших подій у системі не зафіксовано, модельному часу присвоюється значення $t_1^{\text{м}}$, викликаються підпрограма генерування об'єктів і підпрограма запиту і призначення ресурсу R.

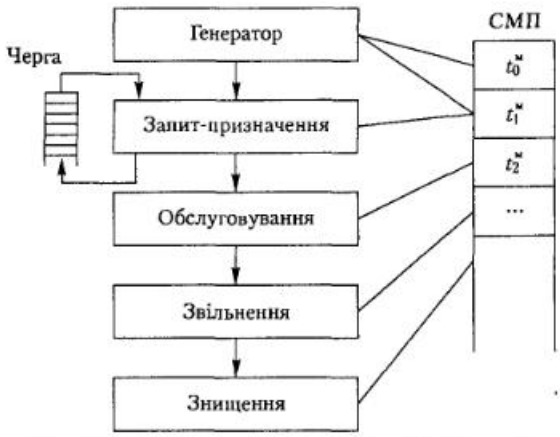


Рис. 2.13. Структурна схема імітаційної моделі

Підпрограма генерування об'єктів визначає майбутню подію – момент приходу другого покупця $t_2^{вх}$ і призначає в СМП настання цієї події на момент часу $t_1^{вх} + t_2^{вх}$.

Підпрограма запиту і призначення перевіряє стан ресурсу R (*продавця*). Якщо ресурс R вільний, то він призначається першому покупцю, і стан ресурсу R змінюється на «зайнятий». Фіксується момент початку обслуговування вимоги $t_1^{п}$. Керування передається підпрограмі обслуговування першого покупця.

Підпрограма обслуговування визначає подію закінчення обслуговування першого покупця як $t_1^{к} = t_1^{п} + t_1^{об}$. Для цього в СМП створюється повідомлення про майбутню подію і керування передається підпрограмі звільнення ресурсу R першою вимогою в момент часу $t_1^{п}$.

Таким чином, тепер у СМП є два елементи: один – з наміченою подією появи другого покупця на момент часу $t_2^{вх}$, другий – з наміченою подією закінчення обслуговування першого покупця в момент часу $t_1^{к}$. Якщо $t_2^{вх} < t_1^{к}$, то годинник модельного часу буде переведено на час $t_2^{м} = t_2^{вх}$, тобто буде викликано заново підпрограму генерування об'єктів і згенеровано появу другого покупця. У цей же момент часу $t_2^{м}$ викликається підпрограма генерування об'єктів, яка намітить у СМП появу третього покупця на час $t_3^{вх}$, та виклик підпрограми запиту і призначення ресурсу R, але оскільки ресурс зайнятий обслуговуванням першого покупця, то другого покупця буде поставлено в чергу до ресурсу R.

У СМП знову буде два елементи: один – з наміченою подією появи третього покупця на час $t_3^{вх}$, другий – з наміченою подією закінчення обслуговування першого покупця на час $t_1^{к}$. Якщо $t_3^{вх} > t_1^{к}$, то годинник модельного часу буде встановлено на час $t_3^{м} = t_1^{к}$, тобто буде викликано підпрограму звільнення ресурсу R першим покупцем. Вона змінить стан ресурсу R із «зайнятий» на «вільний» і передасть керування підпрограмі знищення вимоги. Потім перевірить, чи є вимоги в черзі до ресурсу R і вибере другого покупця з черги, намітить для нього подію для підпрограми запиту і призначення ресурсу R для другого покупця.

Викликана в цей же момент модельного часу $t_3^{м}$ підпрограма знищення ліквідує структуру даних (посилання на адресу) першого покупця. Ця ж підпрограма, якщо необхідно, може обчислювати час перебування першого покупця в системі

($t_3^{\text{пер}} = t_3^{\text{м}} - t_1^{\text{вх}}$) для подальшого статистичного оцінювання часу перебування покупців у системі.

Надалі життєвий цикл інших покупців у системі відбуватиметься за описаним вище алгоритмом.

Під час побудови алгоритму потрібно зазначати час закінчення процесу моделювання за допомогою одного із трьох способів.

1. Моделювання закінчити після того, як через модель пройдуть усі покупці, згенеровані підпрограмою генерування об'єктів, наприклад 100. У цьому випадку в СМП після обслуговування останнього, сотого, покупця не буде жодної запланованої події.
2. Якщо потік покупців від генератора необмежений (наприклад, генерується необмежений пуассонівський потік), моделювання можна закінчити після проходження через модель визначеної кількості покупців, наприклад 1000. Для цього в підпрограмі знищення треба поставити лічильник покупців і припинити моделювання після проходження 1000 покупців. У мові GPSS такий лічильник організовується за допомогою команди START, за якою починається процес моделювання.
3. Потрібно змоделювати роботу системи протягом заданого періоду часу, наприклад 480 хв. У цьому випадку під час зміни модельного часу $t^{\text{м}}$ можна провадити порівняння поточного часу зі значенням 480 хв. Як тільки значення модельного часу буде більше або дорівнюватиме 480 хв, моделювання слід припинити. Однак цей спосіб може дуже сповільнити роботу моделі, оскільки потребує постійної перевірки умови закінчення процесу моделювання. Тому звичайно використовують такий спосіб. Генерують спеціальну вимогу-таймер за допомогою ще однієї підпрограми генерування об'єктів з наміченим часом входу в модель $t^{\text{вх}} = 480$ хв. Вимога-таймер після генерування відразу ж направляється ще в одну підпрограму знищення, в якій ставлять лічильник вимог на одиницю. За лічильником припиняють моделювання. У цьому випадку в СМП весь час буде знаходитись елемент списку для вимоги-таймера з моментом часу настання події 480 хв. Як тільки цю подію буде намічено наступною, показання годинника модельного часу переводиться на час 480 хв, від лічильника вимог віднімається одиниця і він приймає значення нуль, що і є ознакою закінчення моделювання.

У процесі моделювання збирається статистична інформація про роботу моделі під час кожного просування модельного часу. Це можуть бути відомості про довжину черги, час перебування в черзі та пристрої для обслуговування, завантаження або стан пристрою тощо. Для збору інформації створюється спеціальна підпрограма, яка накопичує її, а після закінчення моделювання видає у вигляді стандартного статистичного звіту.

У мові GPSS статистична інформація накопичується в стандартних числових атрибутах і доступна в процесі моделювання тільки для зчитування. Доступ до стандартних числових атрибутів дає можливість керувати процесом руху вимог, наприклад обмежувати довжину черги або час перебування в черзі.

2.7. Мережі систем масового обслуговування

Вище у цьому розділі розглядались методи аналізу найпростіших СМО. Проте існують системи зі складнішою структурою – мережі, кожним вузлом яких є окремі СМО. За допомогою мереж СМО моделюють багато типів транспортних, технологічних та обчислювальних систем, процеси надання медичної допомоги, обслуговування пасажирів та ін. Особливий внесок у розвиток математичних методів у теорії мереж СМО в 70-х роках XX сторіччя зробили фахівці, які займались моделюванням обчислювальних систем.

Аналіз мереж СМО є набагато складнішим, ніж окремих СМО. Отримати результати в замкненому вигляді можна лише для мереж з кількістю вузлів не більше трьох. За більшої кількості вузлів використовуються чисельні методи, що значно ускладнює розрахунки. Крім того, аналіз мереж СМО можливий лише в тих випадках, коли ймовірнісні процеси в мережах є ергодичними, незалежними і протікають за відомими законами розподілу ймовірностей.

Усі потрібні для розрахунків величини (наприклад, параметри розподілів і рівняння, які пов'язують ці величини) також мають бути визначені кількісно. Однак на практиці зробити це не завжди вдається. Наприклад, шляхом вимірювань параметрів функціонування реальної системи неможливо визначити, що тривалість обслуговування запитів – це вибірка значень з послідовності незалежних експоненціально розподілених випадкових величин.

Операційний аналіз, який розглядається далі, вперше було запропоновано для обчислювання показників роботи комп'ютерів і комп'ютерних систем [76]. Він надає математичний апарат для аналізу технічних і економічних систем багатьох типів і дозволяє легко визначити показники їх роботи. Операційний аналіз з успіхом можна застосовувати для валідації імітаційних моделей та пошуку допустимих рішень під час оптимального планування проведення експериментів.

2.7.1. Загальні відомості про мережі СМО

У загальному випадку мережу СМО можна зобразити у вигляді графа, вершинами якого є одноканальні або багатоканальні СМО (дуги визначають потоки пересування вимог).

Найпростіша мережа утворюється шляхом послідовного з'єднання кількох СМО (рис. 2.14). Таку мережу ще називають багатофазовою СМО. Розрізняють *замкнені* та *розімкнені* мережі. Для замкненої стохастичної мережі не існує зовнішніх джерел вимог, тобто в ній завжди знаходиться однакова кількість вимог. Замкнена мережа ізольована від зовнішнього середовища. У розімкненій мережі (рис. 2.14) існують джерела і стоки вимог.

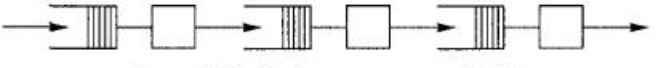


Рис. 2.14. Розімкнена мережа СМО

Найпростіша замкнена мережа, яку зображено на рис. 2.15, має тільки два вузли. Перший вузол містить M пристроїв для обслуговування, а другий – N. Така

мережа є відомою моделлю СМО з відмовами та відновленням. Пристрої для обслуговування M можуть виходити з ладу та відновлюватись із заданими інтенсивностями у випадкові моменти часу. У цій мережі постійно знаходяться M вимог, які з'являються в разі відмови пристроїв обслуговування. Якщо пристрій виходить з ладу, до бригади з N ремонтниками надходить вимога на його ремонт, після завершення якої пристрій відновлює свою роботу. На рис. 2.15 це позначено зворотним зв'язком від N пристроїв. Дана мережа може використовуватись і для моделювання комп'ютерної системи, яка працює в режимі «запит–відповідь». У такій системі користувач не надсилає нового запиту до системи доти, доки не отримає відповіді на попередній запит. Запити обробляють будь-які з N комп'ютерів. Прикладами таких систем можуть бути автоматизовані системи продажу квитків, системи передавання транзакцій від касирів у банку та ін.

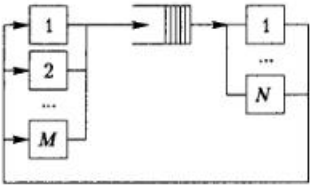


Рис. 2.15. Найпростіша замкнена мережа СМО

Мережа СМО (рис. 2.16) містить K вузлів, а також N вимог. Кожний вузол може містити один або кілька однакових пристроїв для обслуговування. З імовірністю (або частістю) q_{0j} вимоги надходять до будь-якого вузла в мережі СМО, а з імовірністю q_{kj} $(j = \overline{1,K})$ вимога, яка залишає вузол k, прямує до вузла j. Таким чином, кожна вимога в процесі обслуговування в мережі проходить кілька вузлів.

Зовнішнє середовище позначається як вузол 0 мережі. Якщо мережа замкнена, то вимоги від виходу надходять до входу (рис. 2.16, пунктирна лінія) і кількість вимог N у мережі не змінюється.

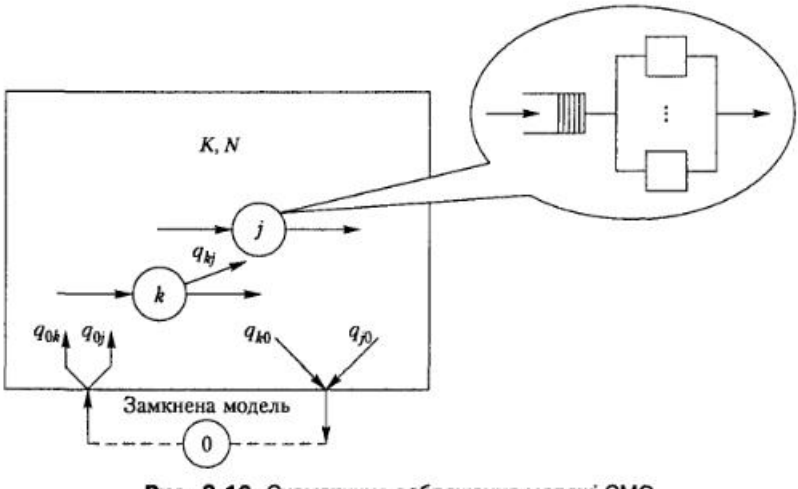


Рис. 2.16. Схематичне зображення мережі СМО

У сталому режимі роботи мережі для потоків вимог справедливі закони про сумарні потоки (рис. 2.17):

$$\sum_{i=1}^{N} q_i = 1, \quad \lambda = \sum_{i=1}^{N} \lambda_i.$$

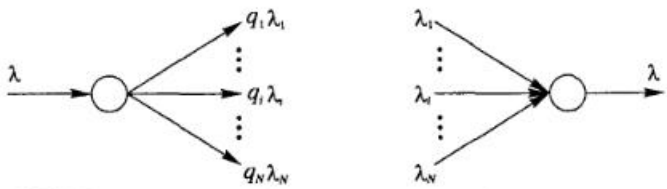


Рис. 2.17. Розгалуження та підсумовування потоків вимог вузла мережі

Для розрахунків мереж СМО використовується теорія стохастичних мереж, яка ґрунтується на марківських або напівмарківських процесах [21], але більшість результатів отримано тільки для експоненціальних законів розподілу надходження та обслуговування вимог. Операційний аналіз [30], на відміну від теорії масового обслуговування базується на моделюванні логіки роботи системи. Це дає змогу встановити прості залежності між параметрами і показниками роботи системи, не абстрагуючись від процесів її функціонування.

2.7.2. Операційний аналіз мереж систем масового обслуговування

Операційний аналіз стохастичних мереж базується на таких положеннях:

- усі припущення щодо властивостей вхідних і вихідних змінних системи можна перевірити шляхом вимірювань впродовж кінцевого проміжку часу параметрів функціонування реальної системи або її моделі;
- у системі повинен існувати баланс потоків вимог: кількість вимог, які залишили систему протягом деякого періоду спостереження, має дорівнювати кількості вимог, що надійшли до системи за цей же період;
- пристрої для обслуговування мають бути однорідними, надходження вимог від одного вузла до іншого не повинні залежати від довжини черг у вузлах і часу закінчення обслуговування пристроями.

Основне завдання операційного аналізу стохастичних мереж полягає у визначенні таких показників, як середній час перебування вимог в окремих вузлах мережі, середній час завантаження пристроїв у вузлах, середні довжини черг до вузлів тощо.

Більшість результатів операційного аналізу стосується замкнених мереж, коли вимоги, які залишають мережу, знову повертаються до неї. Моделі замкнених мереж можна застосовувати для дослідження систем, які працюють з перевантаженням, тобто в яких завжди є черга вимог. У цьому разі можна вважати, що замість вимоги, яка залишила систему, до системи надходить інша вимога з такими ж параметрами.

Операційні змінні

Уведемо операційні змінні, значення яких можна отримати шляхом безпосереднього вимірювання параметрів реальної системи або в процесі її імітаційного моделювання:

- q_{0j} – імовірність (частість) надходження зовнішніх вимог до будь-якого вузла мережі, $j = \overline{1,K}$, де K – загальна кількість вузлів;
- q_{kj} – імовірність надходження вимог від вузла k до вузла j, $k = \overline{1,K}$, $j = \overline{1,K}$;
- q_{k0} – імовірність того, що після закінчення обслуговування у вузлі k вимоги залишать мережу;
- A_k – кількість вимог, які надійшли до вузла k, $k = \overline{1,K}$;
- C_{kj} – кількість вимог, які залишили вузол k і надійшли до вузла j, $k = \overline{1,K}$, $j = \overline{1,K}$;
- B_k – загальний час обслуговування вимог у вузлі k, $k = \overline{1,K}$;
- T – загальний час спостереження за системою або час моделювання.

Зовнішнє середовище мережі позначимо як вершину з номером 0. Тоді параметри A_{0j}, C_{k0} набуватимуть відповідно значень кількості вимог, які надійшли до вузла j ззовні, та вимог, які залишили вузол k і мережу.

Вузол вважається зайнятим, якщо в ньому є хоча б одна вимога. Уведемо додаткові позначення:

$$C_k = \sum_{j=1}^{K} C_{kj}, \quad A_0 = \sum_{j=1}^{K} A_{0j}, \quad C_0 = \sum_{j=1}^{K} C_{j0}.$$

Для замкненої мережі виконується умова $A_0 = C_0$.

Уведені змінні називаються *основними операційними змінними*. Шляхом найпростіших операцій над ними отримують операційні змінні, що виводяться (наприклад, інтенсивність надходження вимог до вузла k визначається як $\lambda_k = A_k/T$). Серед цих операційних змінних найчастіше застосовують

- коефіцієнт використання вузла k:

$$U_k = \frac{B_k}{T}; \tag{2.7}$$

- середній час обслуговування у вузлі k:

$$S_k = \frac{B_k}{C_k}; \tag{2.8}$$

- інтенсивність вихідного потоку вимог від вузла k:

$$X_k = \frac{C_k}{T}; \tag{2.9}$$

- відносну частість переміщення вимог між вузлами k і j:

$$q_{kj} = \begin{cases} \dfrac{C_{kj}}{C_k}, & k = \overline{1,K}, \\ \dfrac{A_{0j}}{A_j}, & k = 0, \end{cases} \qquad \sum_{k=1}^{K} q_{kj} = 1.$$

Використовуючи вирази (2.7)–(2.9), маємо

$$U_k = X_k S_k. \tag{2.10}$$

Вираз (2.10) – це закон *коефіцієнта використання вузла*, який виконується за умови, що $A_k = C_k$ протягом усього періоду спостереження T (у цьому випадку $\lambda_k = X_k$).

Операційні залежності

Основні результати операційного аналізу формуються у вигляді співвідношень між операційними змінними. Ці співвідношення ґрунтуються на гіпотезі про баланс потоків у мережі: кількість вимог, що надійшли до деякого вузла протягом тривалого періоду часу T, дорівнює кількості вимог, які залишили цей вузол. Ця гіпотеза визначає умови роботи мережі СМО в сталому режимі, тобто вважається, що вимоги завжди залишають вузли мережі, або розглядається досить довгий період часу T. Баланс потоків вимог існує тільки для деякого періоду спостереження за системою, але це дуже непогане наближення у разі тривалого періоду часу T, тому що відношення $(A_k - C_k)/C_k$ зазвичай незначне.

Гіпотеза про баланс дає змогу визначати залежність між операційними змінними для кожного вузла мережі, а також записати рівняння балансу потоків вимог:

$$X_j = \sum_{k=0}^{K} X_k q_{kj}, \qquad j = \overline{0,K}. \tag{2.11}$$

Справедливість виразу (2.11) випливає з припущення про баланс потоків вимог у мережі, тобто $A_j = C_j$, бо $\sum_{k=0}^{K} C_{kj} = C_j = A_j$ але за умови, що $q_{kj} = C_{kj}/C_k$, знаходимо $C_j = \sum_{k=0}^{K} C_{kj} q_{kj}$. Поділивши останнє співвідношення (ліву та праву його частини) на загальний час спостереження T, отримаємо вираз (2.11). Рівняння (2.11) буде мати єдиний розв'язок для замкненої мережі у разі заданого значення X_0. Для розімкненої мережі рівняння (2.11) будуть лінійнозалежними, однак і в цьому випадку вони дають корисну інформацію про динаміку потоків мережі.

За допомогою виразу (2.10) знаходимо продуктивність вузла, тобто інтенсивність, з якою вимоги залишають вузол k:

$$X_k = \frac{U_k}{S_k}.$$

Визначаємо коефіцієнт відвідування вузла k вимогами:

$$V_k = \frac{X_k}{X_0}. \tag{2.12}$$

Рівняння балансу потоку можна записати як еквівалентну систему рівнянь, в якій замість інтенсивності потоків використовуються коефіцієнти відвідування кожного вузла мережі.

Поділимо ліву і праву частини виразу (2.11) на X_0:

$$V_0 = 1, \quad V_j = q_{0j} + \sum_{k=1}^{K} V_k q_{kj}, \quad j = \overline{1,K}. \tag{2.13}$$

Вираз (2.13) справедливий, якщо справедливе рівняння (2.11).

Зв'язок коефіцієнтів відвідування та продуктивності вузла визначаємо за формулою

$$\frac{X_k}{X_j} = \frac{V_k}{V_j}.$$

Обчислимо середній час R перебування вимог у стохастичній мережі. Позначимо час перебування вимог у окремих вузлах через R_k. Введемо ще одну операційну змінну W_k, яка дорівнює сумарному часу чекання та часу обслуговування вимог у вузлі k протягом часу T:

$$R_k = \frac{W_k}{C_k}. \tag{2.14}$$

Середній час перебування вимог у системі R можна знайти через R_k і коефіцієнти відвідування окремих вузлів вимогами, тобто

$$R = \sum_{k=1}^{K} V_k R_k. \tag{2.15}$$

Це загальний *закон часу перебування*, який справедливий і в тому випадку, коли гіпотеза про баланс потоків не виконується.

Знайдемо середню кількість вимог у мережі N, яка визначається через середню кількість вимог у кожному вузлі n_k,

$$N = \sum_{k=1}^{K} n_k,$$

де n_k – операційна змінна, яку можна отримати з основних операційних змінних:

$$n_k = \frac{W_k}{T}. \tag{2.16}$$

Для середнього часу перебування вимог у мережі справедливий закон Литтла, тобто середній час перебування вимог у k-му вузлі визначається через середню кількість вимог у ньому та інтенсивність потоку:

$$R_k = \frac{n_k}{X_k}. \tag{2.17}$$

Обґрунтувати формулу Литтла можна також із застосуванням операційного аналізу.

З виразу (2.16) знаходимо

$$W_k = n_k T.$$

Підставляємо отриману операційну змінну в рівняння (2.14):

$$R_k = \frac{n_k T}{C_k} = \frac{n_k}{C_k/T} = \frac{n_k}{X_k}.$$

Закон Литтла справедливий також для всієї мережі в цілому. Підставивши формули (2.12) і (2.17) у вираз (2.15), отримаємо

$$R = \sum_{k=1}^{K} \frac{n_k}{X_k} \frac{X_k}{X_0} = \frac{1}{X_0} \sum_{k=1}^{K} n_k = \frac{N}{X_0}. \qquad (2.18)$$

Покажемо, як можна використовувати основні співвідношення операційного аналізу для визначення часу перебування вимог у замкненій мережі (рис. 2.18).

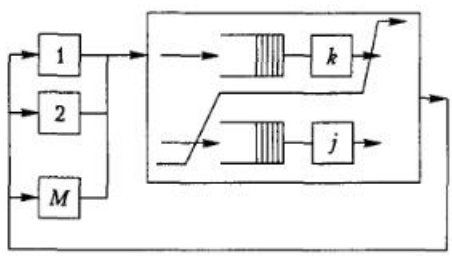


Рис. 2.18. Замкнена мережа СМО

Нехай є M пристроїв, час обслуговування вимоги кожним із них – Z. Середній час перебування вимоги в мережі визначаємо за формулою

$$R = \frac{M}{X_0} - Z. \qquad (2.19)$$

Вираз (2.19) випливає з таких міркувань. Середній час одного циклу взаємодії, який включає час обслуговування вимоги в зовнішній мережі та перебування в одному з M пристроїв, визначається сумою $(Z + R)$. Якщо припустити, що виконується гіпотеза про баланс потоків, то для заданого циклу справедлива формула Литтла. Тому величина $M = (Z + R)X_0$ має визначати середню кількість зайнятих пристроїв або середню кількість працюючих пристроїв для системи з відмовами.

Продемонструємо використання наведених співвідношень операційного аналізу на прикладах. Зображені в них моделі мереж стосуються моделювання обчислювальних систем, але зазначені розрахунки мають загальний характер і демонструють можливості операційного аналізу.

Приклад 2.2

Розглянемо замкнену мережу, яка має $M = 20$ пристроїв. Середній час обслуговування вимоги кожним пристроєм $Z = 25$ с (рис. 2.19). Для вузлів мережі l, g, n ймовірність переміщення вимог до вузла t становить відповідно: $q_{lt} = 0{,}5$; $q_{gt} = 0{,}7$; $q_{nt} = 0{,}85$, а коефіцієнти відвідування цих вузлів – $V_l = 12$, $V_g = 17$, $V_n = 19$. Вузол t завантажений на 50 %, середній час обслуговування вузлом t вимог, які надходять, становить 25 мс. Необхідно знайти середній час перебування R і середню кількість вимог у мережі N.

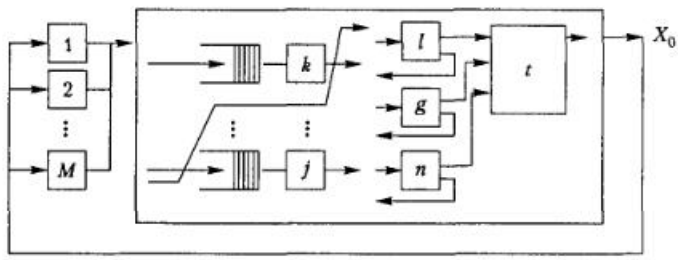


Рис. 2.19. Приклад мережі СМО

Визначаємо коефіцієнт відвідування вузла t, використовуючи рівняння балансу потоків вимог (2.13), записані через коефіцієнти відвідування вузлів:

$$V_t = V_l q_{lt} + V_g q_{gt} + V_n q_{nt};$$

$$V_t = 12 \cdot 0{,}5 \cdot 17 \cdot 0{,}7 + 19 \cdot 0{,}85 = 34{,}05.$$

Знаходимо інтенсивність X_0 надходження вимог у мережі:

$$X_0 = \frac{X_t}{V_t} = \frac{U_t}{V_t S_t}.$$

У цей вираз входять відомі з початкових умов операційні змінні: $U_t = 50$ % і $S_t = 0{,}025$ с. Тоді отримаємо

$$X_0 = \frac{0{,}5}{34{,}05 \cdot 0{,}025} = 0{,}587 \text{ вимоги/с}.$$

З виразу (2.18) знаходимо середній час перебування вимог у мережі:

$$R = \frac{20}{0{,}587} - 25 = 9{,}072 \text{ с}.$$

Для визначення середньої кількості вимог у мережі скористаємося формулою Литтла:

$$N = RX_0;$$

$$N = 9{,}072 \cdot 0{,}587 = 5{,}33 \text{ вимоги}.$$

Таким чином, для даної мережі знайдено середній час перебування і середню кількість вимог у мережі.

Приклад 2.3

Розглянемо мережу (рис. 2.20), до якої надходять вимоги як від пристроїв для обслуговування (замкнена частина мережі, яка, наприклад, моделює роботу терміналів обчислювальної системи), так і ззовні (пакетні завдання, що надходять до обчислювальної системи).

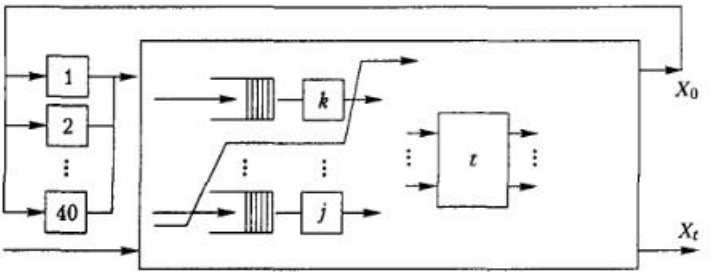


Рис. 2.20. Мережа СМО, яка має замкнену та розімкнену частини

Нехай мережа має 40 пристроїв для обслуговування ($M = 40$). Середній час обслуговування вимог кожним пристроєм $Z = 15$ с. Про мережу відомо такі дані (наприклад, у результаті дослідження реальної обчислювальної системи):

- середній час перебування вимог, які надходять до мережі від 40 пристроїв для обслуговування, дорівнює 5 с;
- середній час обслуговування будь-якої вимоги у вузлі t становить 40 мс;
- кожна вимога, яка надходить від кожного із M пристроїв для обслуговування, породжує 10 вимог, що надходять до вузла t;
- кожна вимога, яка надходить до системи ззовні, породжує 5 вимог, що надходять до вузла t;
- завантаження вузла t становить 90 %.

Потрібно визначити нижню межу часу перебування у мережі вимог, які надходять від M пристроїв для обслуговування з інтенсивністю вхідного потоку X_0 і від зовнішнього джерела вимог з інтенсивністю X_t, тобто визначаються пропускною здатністю вузла t.

Під час розв'язання цієї задачі змінні, що стосуються вимог, які надходять від M пристроїв для обслуговування, позначатимемо зірочкою.

З виразу (2.19) знаходимо $X_0^* = M/(Z + R^*)$, де R^* — середній час перебування вимог, які надійшли до мережі від 40 пристроїв для обслуговування. Тоді $X_0^* = 40/(15 + 5) = 2$ вимоги/с.

Інтенсивність потоку вимог до вузла t визначаємо як суму інтенсивностей потоків вимог від пристроїв для обслуговування та інтенсивності потоку зовнішніх вимог, тобто $(X_t^* + X_t)$. Тоді, згідно з виразом (2.11), можна записати $X_0^* + X_t = U_t/S_t$, або $(X_0^* + X_t) = 0{,}9/0{,}44 = 22{,}5$ вимоги/с.

Використовуючи формулу (2.12), знаходимо $X_t^* = V_t^* X_0^*$ і $X_0^* = 10 \cdot 2 = 20$ вимоги/с.

Звідси $X_t = 2{,}5$ вимоги/с.

Тепер можна знайти інтенсивність вхідного потоку зовнішніх вимог до мережі:

$$X_0 = \frac{X_t}{V_t},$$

$$X_0 = \frac{2{,}5}{5} = 0{,}5 \text{ вимоги/с}.$$

Припустимо, що початкові умови змінилися та інтенсивність вхідного потоку зовнішніх вимог збільшилася втричі, тобто $X_0 = 1{,}5$ вимоги/с. Тоді $X_t = V_t X_0 = 7{,}5$ вимоги/с. Якщо середній час обробки вимог у вузлі t не змінився, то при завантаженні вузла t на 100 % максимально можлива інтенсивність обслуговування вимог у вузлі t становитиме $1/S_t = 25$ вимоги/с. Таким чином, інтенсивність обслуговування вимог у вузлі t (пристроями для обслуговування, які знаходяться у вузлі t) не може перевищувати $(25 - 7{,}5) = 17{,}5$ вимоги/с.

З огляду на це маємо

$$X_0^* = \frac{X_t^*}{V_t^*} \le \frac{17,5}{10} = 1,75 \text{ вимоги/с.}$$

Отже, згідно з виразом (2.18), нижня межа часу перебування вимог у мережі, які надходять від 40 пристроїв для обслуговування, становлять

$$R^* = \frac{M}{X_0^*} - Z \ge \frac{40}{1,75} - 15 = 7,9 \text{ с.}$$

Таким чином, збільшення інтенсивності потоку зовнішніх вимог у 3 рази призведе до збільшення середнього часу перебування вимог у мережі, які надходять від 40 пристроїв для обслуговування, на 2,9 с.

2.7.3. Аналіз вузьких місць у мережі

Пошук вузьких місць у мережі є важливим аспектом аналізу її роботи. Вузьке місце утворюється тим вузлом мережі, коефіцієнт завантаження якого наближається до одиниці. У цьому вузлі створюється велика черга вимог, яка за умови $U \ge 1$ стає нескінченною, тому мережа переходить у нестійкий режим роботи. Такий вузол стає «насиченим» вимогами. За впливом вузьких місць у мережі визначають її пропускну здатність. Тому під час аналізу роботи мережі потрібно особливу увагу приділяти пошуку таких місць.

Покажемо на простому прикладі, як вузьке місце впливає на пропускну здатність мережі. Розглянемо трубопровід, в якому є труби різного діаметра, що доставляють воду споживачу. Зрозуміло, що споживач не отримає води більше, ніж її може пропустити вузька труба. Це так званий ефект вузької шийки. Тому під час аналізу таких систем велике значення має балансування потоків у мережі, тобто знаходження такого балансу потоків у вузлах, при якому середній час перебування в мережі є мінімальним або її пропускна здатність максимальна.

Наведемо співвідношення, яке пов'язує коефіцієнти використання вузлів з коефіцієнтами відвідування цих вузлів:

$$\frac{U_k}{U_j} = \frac{V_k S_k}{V_j S_j}, \quad j,\ k = \overline{1, K}.$$

Якщо пристрій k буде «насиченим» вимогами, тобто його коефіцієнт використання становитиме приблизно 1, то під час виконання гіпотези про баланс потоків інтенсивності вихідного потоку та обслуговування будуть практично збігатись, тобто

$$X_k = \frac{1}{S_k}, \quad \text{якщо} \quad X_k < \frac{1}{S_k}, \quad U_k < 1.$$

У разі збільшення кількості вимог, які одночасно обслуговуються в мережі, першим досягне насичення той вузол d, що буде мати максимальне значення величини $V_i S_i$, $i = \overline{1, K}$ тобто

$$V_d S_d = \max \{V_1 S_1, \ldots, V_k S_k\}.$$

У випадку збільшення кількості вимог значення коефіцієнта використання U_d дорівнює приблизно 1 і $X_d = 1/S_d$. Оскільки $X_0/X_d = 1/V_d$, то

$$X_0 = \frac{1}{V_d S_d}.$$

Таким чином, у разі великої кількості вимог N вихідний потік вимог від мережі повністю визначається вузлом d, що є вузьким місцем.

Визначимо мінімальний середній час перебування вимоги R_0, якщо в мережі є лише одна вимога, через коефіцієнти відвідування окремих вузлів і час обслуговування у вузлі:

$$R_0 = \sum_{k=1}^{K} V_k S_k.$$

На рис. 2.21 зображено графік залежності продуктивності мережі від кількості вимог. У разі збільшення N інтенсивність X_0 монотонно зростає до граничної асимптоти $1/V_d S_d$, тобто доти, доки на цю інтенсивність не почне впливати потенційне вузьке місце вузла d. На рис. 2.21 через N^* позначено кількість вимог, за якої вузьке місце ще не впливає на пропускну здатність мережі.

Для замкненої мережі з кількістю пристроїв $M = 1$ час перебування вимоги в мережі $R' = R_0$. У разі збільшення M потік вимог від мережі зростатиме, але не перевищуватиме $X_0 = 1/V_d S_d$. Таким чином,

$$R' \geq MV_d S_d - Z \geq MV_k S_k - Z, \quad k = \overline{1, K}.$$

Отже, у випадку збільшення M середній час перебування вимоги в мережі має асимптоту $MV_d S_d - Z$. На рис. 2.22 зображено залежність середнього часу перебування вимоги в замкненій мережі від кількості пристроїв M. Асимптота, яка створює вузьке місце в мережі, перетинає вісь абсцис у точці $M_d = Z/V_d S_d$.

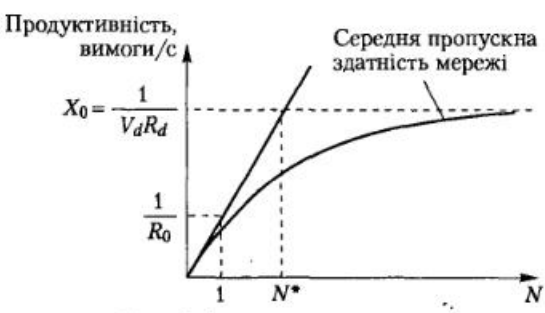


Рис. 2.21. Графік залежності продуктивності мережі від кількості вимог

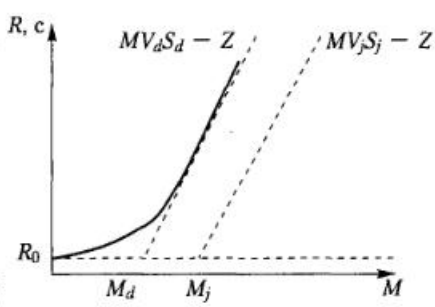


Рис. 2.22. Залежність часу перебування вимог від кількості пристроїв у замкненій мережі

Вищезазначений підхід до пошуку вузьких місць у мережі є досить простим. Покажемо це на прикладах.

Приклад 2.4

Розрахуємо характеристики замкненої мережі, зображеної на рис. 2.23, де наведено значення операційних змінних S_k, q_{kj} і Z.

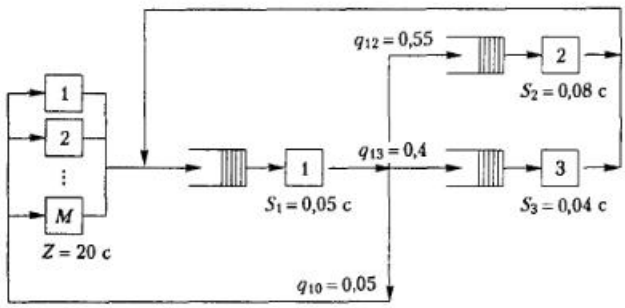


Рис. 2.23. Приклад мережі СМО

Запишемо рівняння балансу потоків вимог для коефіцієнтів відвідування цієї мережі:

$$V_0 = 1 = q_{10}V_1 = 0,05\ V_1;$$
$$V_1 = V_0 + V_2 + V_3;$$
$$V_2 = q_{12}V_1 = 0,55\ V_1;$$
$$V_3 = q_{13}V_1 = 0,4\ V_1.$$

Розв'язавши цю систему рівнянь, отримаємо

$$V_1 = 20, \quad V_2 = 11, \quad V_3 = 8.$$

Обчислимо значення V_kS_k для кожного з вузлів мережі:

$$V_1S_1 = 20 \cdot 0,05 = 1 \text{ с},$$
$$V_2S_2 = 11 \cdot 0,08 = 0,88 \text{ с},$$
$$V_3S_3 = 8 \cdot 0,04 = 0,32 \text{ с}.$$

Таким чином, мінімальний середній час перебування однієї вимоги в мережі $R_0 = 1 + 0,88 + 0,32 = 2,2$ с. Оскільки $V_1S_1 > V_2S_2 > V_3S_3$, то потенційним вузьким місцем у мережі є перший вузол.

За допомогою методу операційного аналізу можна знайти відповідь, наприклад, на такі запитання.

1. Яка середня кількість пристроїв M для обслуговування взаємодіє з мережею протягом усього часу спостереження? Нехай за допомогою вимірювань визначено, що $X_0 = 0,715$ вимоги/с, а середній час перебування вимоги в мережі становить $R = 5,2$ с.

   Згідно з формулою (2.19) маємо

   $$M = (R + Z)X_0 = (5,2 + 20) \cdot 0,715 = 18 \text{ пристроїв}.$$

2. Чи можна забезпечити середній час перебування вимог у мережі, який дорівнюватиме 8 с, якщо кількість пристроїв для обслуговування становить 30? Чому повинен дорівнювати максимальний середній час обслуговування вимоги в першому вузлі, щоб це стало можливим?

   Згідно з формулою (2.19) маємо:

   $$R \geq \frac{M}{X_0} - Z = \frac{30}{1} - 20 = 10 \text{ с}.$$

Таким чином, у разі взаємодії з мережею 30 пристроїв для обслуговування середній час перебування вимоги в ній становитиме понад 10 с.

Позначимо через S_1^* допустимий середній час обслуговування вимоги. Тоді можна записати

$$MV_1S_1^* - Z \le 8 \text{ с}; \quad S_1^* \le 0{,}047 \text{ с},$$

тобто максимально можливий середній час обслуговування вимоги у вузлі *1* становить 0,047 с.

Для цього випадку на рис. 2.24 зображено графіки для асимптоти середнього часу обслуговування вимог.

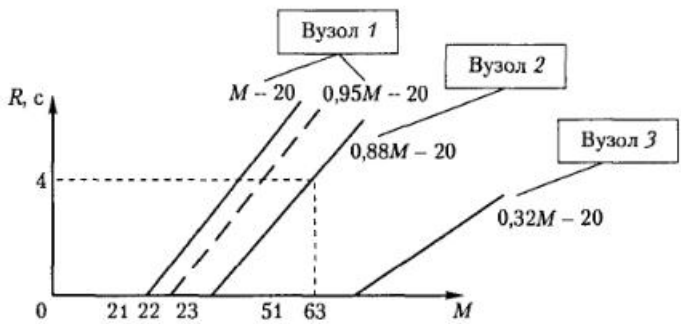


Рис. 2.24. Графіки для середнього часу обслуговування вимог

Приклад 2.5

Припустимо, що в мережу, крім вимог від пристроїв для обслуговування, надходять ще і вимоги від вузла *3* (рис. 2.25). Параметри, позначені штрихом, характеризують вимоги вузла *3*. Вимірювання, які провадились для реальної обчислювальної системи, працюючої в режимі «запит–відповідь», свідчать, що вузол *3* завантажений практично повністю, а час відповіді системи дорівнює 7 с. Як у цих умовах завантажений вузол *1* і якого значення набуває величина X_0'?

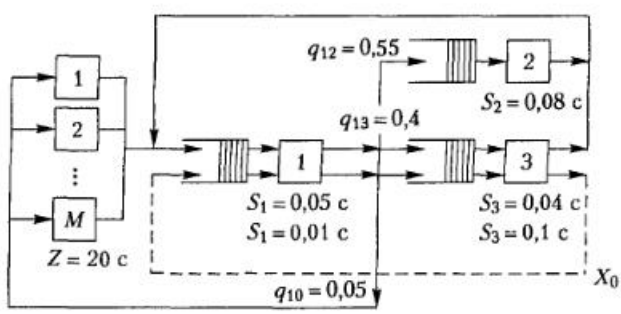


Рис. 2.25. Приклад мережі СМО

На рис. 2.25 видно, що $V_1' = V_3' = 1$. Звідси $V_1'S_1' = 0{,}01$ с, $V_3'S_3' = 0{,}1$ с. Таким чином, потенційно вузьким місцем для вимог вузла *3* є сам вузол *3*.

$$X_0 = \frac{M}{R+Z} = \frac{25}{7+20} = 0{,}924 \text{ вимоги/с}.$$

Оскільки $X_0 = X_3/V_3 = X_1/V_1$, то $X_3 = 7{,}392$ вимоги/с, $X_1 = 18{,}48$ вимоги/с.

Завантаженість вузла *3*, що створюється вимогами від пристроїв для обслуговування, має дві складові від різних потоків вимог. Для першого потоку

$$U_3 = X_3 S_3 = 7{,}392 \cdot 0{,}04 = 0{,}296.$$

Для другого потоку завантаженість цього вузла $U_3 = 0{,}704$.

Оскільки вимоги від вузла *3* циркулюють у замкненому контурі, то за умови замкнутості мережі маємо

$$X_1' = X_3' = X_0' = \frac{U_3'}{S_3'} = 7{,}04 \text{ вимоги/с.}$$

Визначимо коефіцієнт використання вузла *1*:

$$U_1 + U_1' = X_1 S_1 + X_1' S_1' = 18{,}48 \cdot 0{,}05 + 7{,}04 \cdot 0{,}01 = 0{,}994.$$

Таким чином, розраховано завантаженість вузла *1* та інтенсивність потоку X_0'.

Висновки

- Теорія масового обслуговування визначає залежність між параметрами потоків вимог, кількістю пристроїв обслуговування і їх продуктивністю, а також між режимами функціонування СМО та її ефективністю.
- Основні теоретичні результати теорії масового обслуговування отримано для систем, в яких процеси надходження та обслуговування вимог є марківськими та напівмарківськими.
- Замкнена мережа СМО – це ізольована від зовнішнього середовища мережа, для якої не існує зовнішніх джерел і стоків вимог.
- Розімкнена мережа СМО – це мережа, в якій вимоги, що обробляються пристроями обслуговування, можуть надходити із зовнішнього середовища або спрямовуватись до нього.
- Операційний аналіз є методом дослідження мереж СМО, в основі якого лежать операційні змінні. Основні результати операційного аналізу формуються у вигляді співвідношень між операційними змінними.
- Умови функціонування мережі СМО визначає гіпотеза про баланс потоків у мережі: кількість вимог, що надійшли до деякого вузла протягом тривалого періоду часу, дорівнює кількості вимог, які залишили цей вузол.

Контрольні запитання та завдання

1. Розгляньте потік Ерланга r-го порядку. Як розподіляються проміжки часу між сусідніми вимогами? Поясніть, чому. До якого розподілу буде наближатись розподіл Ерланга у разі збільшення параметра r? Який розподіл матиме час надходження вимог, якщо r прямує до нескінченності?

2. Розгляньте K незалежних джерел вимог, для кожного з яких проміжки часу між сусідніми вимогами розподілені за експоненціальним законом з параметром λ_k, тобто кожне джерело є пуассонівським. Розгляньте сумарний потік від усіх джерел і доведіть, що цей потік теж є пуассонівським з параметром $\Lambda = \sum_{k=1}^{K} \lambda_k$.
3. Розгляньте об'єднаний потік вимог із попереднього завдання і припустіть, що тепер його потрібно розбити на кілька потоків. Нехай p – це ймовірність того, що деяка вимога з об'єднаного потоку буде віднесена до підпотоку з номером k. Якщо інтенсивність сумарного потоку дорівнює Λ вимог за одиницю часу і ймовірності p вибираються довільно, то кожний з підпотоків створює пуассонівський процес з інтенсивністю λp. Покажіть це.
4. Поясніть, чому система типу $D/D/1$ за умови $\rho = 1$ працює в сталому режимі й не має черги, а система типу $M/M/1$ за умови $\rho = 1$ переходить у нестаціонарний режим, і черга росте до нескінченності. Що буде за умови накладення обмежень на довжину черги для системи типу $M/M/1$? У цьому випадку при будь-якому співвідношенні значень для інтенсивності надходження вимог λ і швидкості обслуговування μ ця система працюватиме в сталому режимі. Покажіть це.
5. З огляду на життєві ситуації наведіть кілька прикладів, які ілюструють роботу СМО. Розгляньте такі варіанти:
   а) вимоги надходять до системи обслуговування по одній;
   б) вимоги надходять до системи обслуговування пакетами;
   в) порядок надходження відомий заздалегідь;
   г) надходження повністю випадкове;
   д) джерело вимог невичерпане або обмежене;
   е) існує тільки одна черга або кілька черг;
   ж) час перебування в черзі обмежений;
   з) обслуговування виконується у дві фази.

   Назвіть для наведених вами прикладів три–п'ять важливих характеристик, які потрібно було б розрахувати або виміряти для цих систем. За яких умов можна вважати вхідний потік вимог пуассонівським?
6. Опишіть, яким чином потрібно змінити роботу алгоритмів моделювання, описаних у розділі 2.6.4, щоб урахувати:
   а) обмеження, накладені на довжину черги;
   б) обмеження, накладені на час перебування в черзі;
   в) наявність m однакових пристроїв для обслуговування вимог.

   Під час реалізації алгоритму для третього варіанта розгляньте визначення одного із вільних пристроїв у порядку наданих їм номерів 1, 2, ..., m і у випадковому порядку. Поясніть, чому для першого способу визначення вільних приладів їх завантаження буде різним.
7. Продовжіть моделювання процесів обслуговування в магазині методом, описаним у розділі 2.6.2, до 20 вимог, задавши довільне значення часу надходження

та часу обслуговування вимог. Побудуйте за допомогою програми Excel або пакета STATISTICA гістограму часу перебування вимог у черзі.

8. У якому випадку потік вимог з мережі (див. рис. 2.14) буде пуассонівським? Поясніть чому.
9. Які складності виникають під час розрахунків мереж СМО?
10. Для розрахунків яких систем можна застосовувати операційний аналіз?
11. Сформулюйте гіпотезу про баланс потоків у мережі СМО. Поясніть, як її можна використовувати під час розрахунків різних СМО.
12. На рис. 2.26 зображено графік спостереження за роботою СМО з одним пристроєм для обслуговування протягом 20 с її роботи. Проведені вимірювання показали, що $A_i = 7$ вимогам; $B_i = 16$ с; $C_i = 10$ вимог. Як видно на графіку, $n_i(0) = 3$ вимог і $n_i(20) = n_i(0) + A_i + C_i = 0$. Потрібно знайти вихідні операційні змінні: коефіцієнт завантаження U_i, продуктивність X_i, сумарний час чекання W_i та обслуговування S_i вимог, середню довжину черги N і середній час перебування вимоги в СМО.

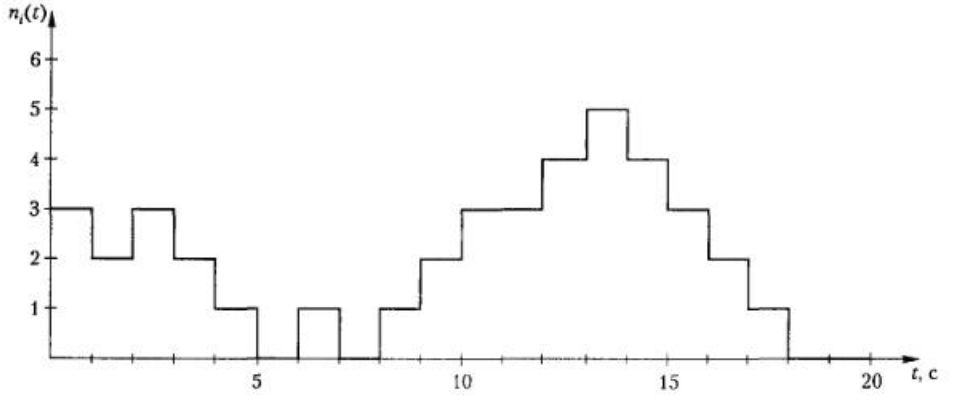


Рис. 2.26. Графік спостереження за роботою СМО

13. Деяка виробнича дільниця має L верстатів, які працюють 24 год на добу H діб. Будь-який з верстатів може вийти з ладу в будь-який час. Якщо верстат зламався, його замінюють іншим, резервним, а зламаний направляють для ремонту в майстерню. Відремонтований верстат повертають до роботи.

    У майстерні є три спеціалізовані дільниці для ремонту верстатів. Технологічний цикл ремонту починається на дільниці діагностики, де визначаються причина виходу з ладу верстатів та необхідний вид ремонту. Діагностика триває $A_1 \pm B_1$ год (розподіл – рівномірний). Ремонт виконується в механічних і електричних майстернях. За статистичними даними, отриманими за результатами аналізу виходу з ладу верстатів, 75 % випадків становить відмова електричного обладнання верстатів, а 25 % – механічного. Діагностикою зайнято m_1 робітників, ремонтом механічного обладнання – m_2, а ремонтом електричного обладнання – m_3 робітників.

Досвід експлуатації свідчить, що на діагностику витрачається $A_1 \pm B_1$ год, на ремонт електричного обладнання верстата – $A_2 \pm B_2$ год, а на ремонт механічного обладнання – $A_3 \pm B_3$ год (розподіл – рівномірний). Якщо верстат використовується у виробництві, час напрацювання на відмову має експоненціальний розподіл з параметром T год. Час для перевезення верстатів із цеха в майстерню та в зворотному напрямку незначний, і його не враховують. Між робітниками в майстерні немає ніяких відмінностей, як і між верстатами.

Керівнику необхідно визначити, скільки робітників треба найняти для роботи в майстерні на кожній дільниці, щоб мінімізувати час ремонту обладнання. Час спостереження за системою – H діб. Необхідні дані наведено в табл. 2.2.

Таблиця 2.2. Параметри моделювання роботи майстерні

L	T	$A_1 \pm B_1$	$A_2 \pm B_2$	$A_3 \pm B_3$	H
50	160	2 ± 1	30 ± 10	45 ± 5	360

14. У касовому залі з продажу квитків на літаки працює 20 касирів, кожний за своїм терміналом. Термінали підключено до двохпроцесорного кластерного сервера, який обробляє запити клієнтів на будь-якому з двох процесорів. Середній час обробки запиту процесором становить 10 с. Кожний касир працює із системою в режимі «запит–відповідь», тобто не посилає новий запит, доки не отримає відповідь на попередній. Середній час введення запиту касиром становить 30 с. До касирів постійно є черга клієнтів. Знайдіть середній час відповіді системи на запит клієнтів. Побудуйте залежність цього часу від кількості працюючих касирів, яка може змінюватись від 1 до 20 чоловік.

Розділ 3
Мережі Петрі

- Формалізоване визначення мереж Петрі
- Моделювання динамічних систем за допомогою мереж Петрі
- Розширення можливостей елементів мереж Петрі для моделювання

Мережі Петрі – це математична модель, яка широко використовується для моделювання динамічних систем багатьох типів. Мережі Петрі запропонував Карл Адам Петрі у своїй дисертації в 1962 році, і з часом вони стали однією з найбільш зручних математичних конструкцій для представлення моделей складних причинно-наслідкових систем. За допомогою мереж Петрі досить легко будувати моделі будь-яких асинхронних, паралельних і розподілених систем.

Високий рівень формалізації мереж Петрі [26, 45] дає змогу легко будувати на їх основі алгоритми і програми моделювання. Як математичний інструмент мережі Петрі можна використовувати для опису рівнянь станів, алгебричних рівнянь та інших математичних моделей, що керують поведінкою систем. Крім того, вони є досить зручним інструментом для аналізу та автоматизації побудови програм імітаційного моделювання.

В Європі існує два провідних центри, де займаються проблемами моделювання систем на основі мереж Петрі, – у Данії та Німеччині. У центрах провадять теоретичні дослідження, розробляють програмні засоби моделювання, а також впроваджують стандарти стосовно використання мереж Петрі (http://www.daimi.au.dk/PetriNets/standard/). Ознайомитися з базою інструментальних засобів моделювання мереж Петрі можна в Інтернеті на сайті http://www.daimi.au.dk/ PetriNets/tools/quick.html.

3.1. Прості мережі Петрі

Мережа Петрі є орієнтованим дводольним графом, який має чотири базових елементи: *вузли*, або *місця* (places), *переходи* (transitions), *дуги* (arcs) і *маркери* (tokens). Нагадаємо, що дводольним називається граф, який має дві множини вузлів і не має ребер, з'єднуючих вузли однієї множини. Вузли позначаються кружками і визначають стан, в якому може знаходитись мережа або її частина. Переходи – це активні елементи мережі, які позначають дії, виконувані під час спрацювання переходів. Для того щоб перехід міг спрацювати, необхідне виконання певних

умов, які визначаються наявністю маркерів у вузлах мережі, з'єднаних з переходом. Якщо умови настання подій виконано, то вважають, що перехід збуджений. Переходи позначаються короткими вертикальними або горизонтальними лініями.

Вузли та переходи з'єднуються орієнтованими ребрами (дугами). Вузли, з яких виходять дуги до певного переходу, називаються вхідними вузлами переходу, а вузли, до яких ведуть дуги з певного переходу, – вихідними. Два вузли або два переходи з'єднуватись дугами не можуть. Кожний перехід може бути з'єднаним з вузлом тільки однією дугою (вхідною або вихідною).

Функціонування мережі Петрі можна описати різними способами. Наприклад, можна розглядати вузли як певні умови, а переходи – як події. Таким чином, стан мережі в кожний момент часу задається системою умов. Для зручності задання умов у мережі Петрі вводяться маркери (фішки), які зображуються крапками всередині вузлів. Виникнення певної комбінації маркерів у вузлах приводить до настання деякої події, яка у свою чергу викликає зміну стану умов мережі. Стан маркування або стан мережі Петрі визначається сукупністю маркерів кожного окремого вузла мережі.

На рис. 3.1 зображено просту мережу Петрі з одним переходом, трьома вузлами, два з яких містять маркери.

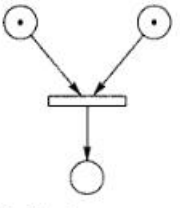


Рис. 3.1. Проста мережа Петрі

Перехід, в якого всі вхідні вузли містять маркери, називається *збудженим* (рис. 3.1). Збуджений перехід може *спрацювати*, після чого всі маркери із вхідних вузлів переходу перемістяться у вихідні (рис. 3.2). Таким чином, настає *подія*, яка змінює стан мережі.

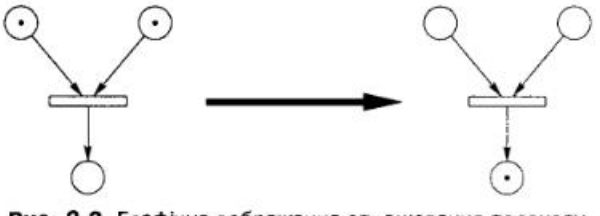


Рис. 3.2. Графічне зображення спрацювання переходу

Якщо одночасно збуджуються кілька переходів мережі, виникає невизначеність, тому одночасне спрацювання кількох переходів у мережі Петрі неможливе, тобто переходи спрацьовують послідовно, миттєво. Незважаючи на те, що маркери змінюють своє положення у вузлах, прості мережі Петрі – це статичні моделі, в яких не враховується динаміка в часі (зміна станів мережі не залежить від моментів часу).

Для того щоб за допомогою мережі Петрі відтворити динаміку роботи деякої детермінованої динамічної системи в часі, необхідно зазначати моменти спрацьовування переходів. Такі можливості мають тільки розширення мереж Петрі, в яких спрацювання переходів здійснюється в задані моменти модельного часу з деяким постійним кроком Δt.

3.1.1. Розмітка мережі Петрі

Розмітка M мережі Петрі — це функція, яка ставить у відповідність маркерам вузлів цілі додатні числа. Суть розмітки полягає в приписуванні кожному вузлу певної кількості маркерів. Наприклад, якщо позначити через N саму мережу Петрі, через P — множину вузлів у мережі N, через $n(P)$ — кількість вузлів, то кожному вузлу цієї мережі можна поставити у відповідність число із послідовності $\{1, 2, ..., n(P)\}$. Таким чином, розмітку M можна зобразити за допомогою вектора з $n(P)$ елементів, в якому i-й елемент визначає кількість маркерів у i-му вузлі. У загальному випадку кількість маркерів у вузлі може бути більшою за одиницю.

На рис. 3.3 зображено мережу Петрі, яка може перейти в такий стан, коли жоден з переходів не буде збудженим. Мережа в такому стані називається *заблокованою*. Розмітка заблокованої мережі задається такими елементами: $M = [2, 0, 0]$; $P = \{1, 2, 3\}$; $n(P) = 3$; a, b, c, d — переходи. Переходи a, b можуть бути збудженими, а переходи c, d — ні. У результаті збудження та спрацьовування переходу a отримуємо розмітку $M^1 = [1, 1, 0]$, за якої збуджуються переходи a, b і c. У разі спрацювання переходу c отримуємо мережу з новою розміткою $M^2 = [2, 0, 0]$. У мережах Петрі паралельне спрацювання переходів обмежується тільки кількістю маркерів у вузлах, тобто для спрацьовування переходів мають бути маркери у всіх вхідних вузлах.

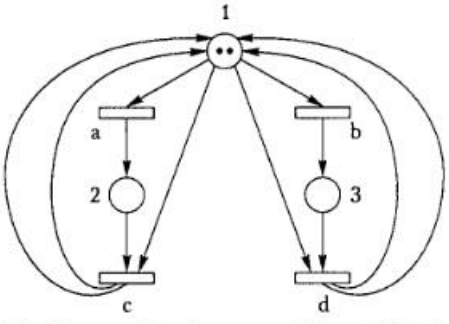


Рис. 3.3. Мережа Петрі, яка може бути заблокованою

При початковій розмітці мережі, яку зображено на рис. 3.3, переходи a і b можуть збуджуватись одночасно. Однак збудження двох переходів в один і той же момент часу не означає, що вони можуть одночасно спрацьовувати. Наприклад, переходи a та b мережі з розміткою $M = [1, 0, 0]$ можуть одночасно тільки збуджуватись, але не спрацьовувати. Причина в тому, що в разі спрацювання як переходу a, так і переходу b повинні одночасно зникнути обидва маркери з першого вузла, що неможливо.

Перехід, в якого немає жодного вхідного вузла, завжди є збудженим і може видавати (*генерувати*) маркери. Генерування маркерів можна розглядати як послідовне виконання функції розмітки M, тобто послідовність зміни розміток M^0, M^1, M^2, ..., де M^{i+1} отримане із M^i у результаті збудження деякого переходу. Тут M^0 – початкова розмітка мережі Петрі.

Перехід, який не має жодної вихідної дуги і має тільки одну вхідну дугу, збуджений завжди тільки в тому випадку, якщо вхідний вузол містить маркер. Такий перехід може *знищувати* маркери.

У загальному випадку для заданої мережі N з початковою розміткою M^0 є багато можливих послідовностей спрацювання переходів. Мережа Петрі знаходиться в *«живому»* стані (живуча розмітка), коли кожний з переходів може збуджуватись нескінченну кількість разів.

Мережа Петрі знаходиться в *«мертвому»* стані (заблокована), якщо вона не має жодного збудженого переходу. Заблокована мережа є стійкою, тобто її маркери не можуть змінювати свої позиції у вузлах. На рис. 3.3 зображено мережу з початковою розміткою $M^0 = [2, 0, 0]$. У разі спрацювання переходів a і b початкова розмітка замінюється на розмітку $M^2 = [0, 1, 1]$. Мережа з такою розміткою стане заблокованою, і жодний з переходів не зможе бути збудженим. Отже, розмітка M^0 не є живучою. Дійсно, для цієї мережі не існує живучих розміток. Незалежно від кількості маркерів у вузлі *1* переходи a і b будуть спрацьовувати доти, доки всі маркери не перейдуть у вузли *2*, *3*. Коли у вузлі *1* не залишиться маркерів, мережа заблокується.

Вважають, що два переходи *конфліктують*, якщо вони не можуть бути збудженими одночасно. Незважаючи на те, що умови конфлікту формально точно не визначені, зрозуміло, що суть конфлікту тісно пов'язана з типом тупикової ситуації, яка мала місце в мережі, зображеній на рис. 3.3, де переходи c і d конфліктують при розмітці M^0.

Теорем, якими задаються умови блокування та скінченності мереж Петрі, не існує. Подібні теореми доведено тільки для *розмічених графів*, які є простішими за мережі Петрі. Для графів тільки певних типів доведено, що перехід, який не може бути збудженим, існує лише в тому випадку, якщо в графі існує орієнтований цикл без маркерів. Для мереж Петрі аналогічного твердження довести не вдалось. У розміченому графі, зображеному на рис. 3.4, перехід a мертвий, оскільки дуги *1* і *2* створюють орієнтований цикл, який не має маркерів.

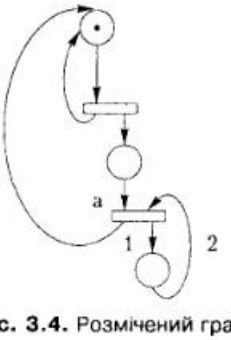


Рис. 3.4. Розмічений граф

3.1.2. Формальне визначення мереж Петрі

Існує кілька способів формального опису мереж Петрі, які відрізняються способами задання елементів і зв'язків. Згідно з працею [26] будемо визначати мережу Петрі трьома елементами: $\langle P, T, F\rangle$, де P – непорожня множина елементів мережі, названих вузлами; T – непорожня множина елементів мережі, названих переходами; F – *функція інцидентності*, що задає зв'язок між елементами множин P і T. Для мережі Петрі, визначеної елементами $\langle P, T, F\rangle$, повинні виконуватись такі умови.

1. $P \cap T = \varnothing$, тобто множини вузлів і переходів не перетинаються.
2. $(F \neq \varnothing) \wedge (\forall x \in P \cap T, \exists y \in P \cup T: xFy \vee yFx)$, тобто будь-який елемент мережі інцидентний хоча б одному елементу іншого типу.
3. Якщо для довільного елемента мережі $x \in X$ позначити через $^*x$ множину його вхідних елементів, а через x^* – множину його вихідних елементів, то

   $\forall p_1, p_2 \in P: (^*p_1 = {}^*p_2) \wedge (p^*_1 = p^*_2) \Rightarrow (p_1 = p_2)$, тобто мережа не міститиме пари вузлів, які інцидентні до тієї ж множини переходів.

На основі поняття мережі Петрі, яка описує тільки статичну топологію модельованого процесу або системи, вводяться динамічні мережні структури, в яких вузлам приписуються спеціальні розмітки для моделювання виконання умов. З мережею пов'язують поняття її функціонування, яке описується зміною її розмітки (умови) у результаті спрацювання переходів. До динамічних мереж відносяться мережі Петрі та їх різні узагальнені варіанти. Таким чином, мережа Петрі не є динамічною системою, в якій стан моделі змінюється в часі, він змінюється тільки в разі спрацювання переходів і зміни розмітки.

Зміну позицій маркерів у вузлах можна визначити, якщо задати мережу Петрі як структуру $P_N = \langle P, T, F, M^0\rangle$, де $\langle P, T, F\rangle$ – скінченна мережа (множини P і T скінченні), а M^0 – початкова розмітка мережі, яка ставить у відповідність будь-якому вузлу $p_i \in P$ деяке число $M^0(p) = n$.

Функціонування мережі Петрі описується за допомогою множини послідовностей спрацьовувань і множини досяжних у мережі розміток. Ці поняття визначаються через правила спрацювання переходів мережі.

Розмітка мережі P_N є функцією $M: P \Rightarrow N$ (N – множина натуральних чисел). Якщо припустити, що всі вузли мережі строго впорядковані деяким чином, тобто $P = (p_1, p_2, ..., p_n)$, то розмітку M мережі (у тому числі і початкову розмітку) можна задати як вектор чисел $M = [m_1, m_2, ..., m_n]$ такий, що для будь-якого i $(1 \le i \le n)$ $m_i = M(p_i)$. Якщо $P' = (p_{i1}, p_{i2}, ..., p_{ik})$ – підмножина вузлів з P, то умовимося через $M(P')$ позначати множину розміток $\{M(p_{i1}), ..., M(p_{ik})\}$. Якщо P' представити як вектор $P' = (p_{i1}, ..., p_{ik})$, то $M(P')$ позначає вектор проекції розмітки M на P'.

Перехід $t \in T$ може спрацювати при деякій розмітці M мережі P, якщо $\forall p \in {}^*t M(p) \ge F(p, t)$, тобто кожний вхідний вузол p переходу t має розмітку, не меншу, ніж кратність дуги, що з'єднує p і t. Цю умову можна записати як $M \ge {}^*F(p, t)$.

Для ординарної мережі Петрі (тобто мережі, в якій дуги мають кратність, рівну одиниці) умова спрацьовування переходу означає, що будь-який вхідний вузол цього переходу містить хоча б один маркер, тобто має ненульову розмітку.

Спрацьовування переходу t розміткою M породжує розмітку M' за таким правилом:

$$\forall p \in P : M'(p) = M(p) - F(p, t) + F(t, p),$$

тобто

$$M' = M - {}^*F(t) + F^*(t).$$

На множині розміток уводять відношення [> безпосереднього проходження розміток:

$$M \; [> \; M' \Leftrightarrow \exists t \in T : (M \geq {}^*F(t)) \; \& \; (M' = M - {}^*F(t) + F^*(t)),$$

тобто розмітка M' безпосередньо отримана за розміткою M, якщо знайдеться такий перехід t, який може спрацювати в разі розмітки M, і розмітка M' є результатом спрацьовування цього переходу в разі розмітки M.

Розмітка M' досяжна від розмітки M, якщо існує послідовність розміток M, M^1, M^2, ..., M' і слово $\tau = t_1 t_2 \ldots t_k$ в алфавіті T такі, що

$$M [t_1 > M^1 [t_2 > M^2 \ldots [t_k > M'.$$

Слово τ у цьому випадку називається послідовністю перемикань, які ведуть від M до M'.

Множину розміток, досяжних у мережі P_N від розмітки M, позначимо через $R(P_N, M)$. Якщо $R(P_N) = R(P_N, M^0)$, то множину всіх розміток, досяжних у мережі P_N від початкової розмітки M^0, називають *множиною досяжних розміток* мережі. Властивості мережі Петрі визначають, досліджуючи можливі послідовності спрацьовувань переходів і множини досяжних у мережі розміток.

3.2. Моделювання систем за допомогою мереж Петрі

Прості мережі Петрі містять лише три основних елементи: вузли, переходи та маркери. Тому побудова за їх допомогою моделей складних динамічних систем, в яких протікає велика кількість взаємодіючих паралельних і асинхронних процесів та існує багато інформаційних і матеріальних потоків, стає досить складною та громіздкою процедурою. Це помітно звужує клас моделей систем, які можна побудувати на основі простих мереж Петрі. У таких випадках застосовують розширення простих мереж Петрі, які дають можливість значно спростити побудову складних моделей і їх графічне зображення.

3.2.1. Розширення простих мереж Петрі

Розширення мережі Петрі — це така їх модифікація, яка збільшує можливості мережі стосовно опису та моделювання систем. Існують різні розширення мереж Петрі, орієнтовані на моделювання систем різних типів: стохастичних, динамічних, предикатних та ін. Кольорові мережі Петрі дають змогу значно зменшити

розміри мереж, які використовуються, наприклад, для опису моделей складних паралельних обчислювальних систем.

Необхідність дослідження динаміки ресурсних потоків у системах змусила шукати шляхи включення до мереж Петрі в явному вигляді часових параметрів. Крім того, у деяких моделях необхідно розрізняти порядок надходження маркерів до вузлів та вибору маркерів з них, що дає змогу впорядкувати події в часі.

На практиці широко використовуються проблемно-орієнтовані розширення мереж Петрі, серед яких найбільш відомі *E*-мережі, комбі-мережі, *FIFO*-мережі, *M*-мережі та ін.

Нижче наведено класифікацію та коротку характеристику деяких розширень мереж Петрі, які часто використовуються для формалізації опису обчислювальних комплексів, систем і мереж. Більшість матеріалу, викладеного в цьому розділі, базується на документації користувача системи POSES++, розробленої німецькою компанією GPC mbH (http://www.gpc.de/), і містить опис засобів, які розширюють можливості мереж Петрі в разі моделювання складних розподілених і паралельних систем.

У простих мережах Петрі допускається наявність у вузлі лише одного маркера, тоді як у розширених мережах кожний вузол може містити кілька маркерів (наприклад, вузол *1* на рис. 3.3). Кількість маркерів у вузлі позначає число поряд з вузлом. Відповідно для цих вузлів змінюються і правила маркування:

- перехід збуджується тільки тоді, коли число, яке визначає кількість маркерів у кожному вхідному вузлі, більше або дорівнює одиниці;
- якщо збуджений перехід спрацьовує, то число маркерів у всіх вхідних вузлах, які містять маркери, зменшується на одиницю, а в усіх вихідних вузлах – збільшується на одиницю. Певна річ, кількість маркерів не може бути від'ємним значенням.

Мереж Петрі достатньо для опису причинно-наслідкових подій, які виникають у системах, однак мережі не забезпечують повну спільність з логічними операціями. Зрозуміло, що проста мережа Петрі відтворює роботу тільки логічного елемента «І» («AND»), тому за допомогою цієї мережі не можна змоделювати збудження переходу, коли вхідний вузол не має маркерів (логічний оператор заперечення «НІ»). Для цього в розширених мережах уводяться дуги заперечення, які зображуються лініями з кружечком на кінці замість стрілки (рис. 3.5) і не мають дугової ваги (дугова вага визначає пропускну здатність дуги). Дуги, визначені раніше, розглядаються як позитивні. Вони завжди направлені від деякого вузла до переходу і не можуть мати зворотного напрямку. Наявність дуги заперечення змінює правила маркування на такі:

- перехід збуджується лише тоді, коли число, яке визначає кількість маркерів кожного вхідного вузла з позитивною дугою, більше або дорівнює одиниці, та коли кількість маркерів кожного вхідного вузла з дугою заперечення дорівнює нулю;
- якщо збуджений перехід спрацьовує, то число маркерів усіх вхідних вузлів з позитивною дугою зменшується на одиницю, у той час як кількість маркерів

вхідних вузлів з дугами заперечення залишається незмінною. Кількість маркерів усіх вихідних вузлів збільшується на одиницю.

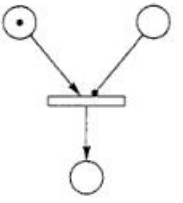


Рис. 3.5. Введення дуги заперечення

Для кожної позитивної дуги можна задати певний ваговий коефіцієнт (або вагу), рівний одиниці або більший за неї (рис. 3.6). За замовчуванням ваговий коефіцієнт дуги дорівнює одиниці. Тоді збудження та спрацьовування переходу відбувається за такими правилами:

- перехід збуджується тільки тоді, коли кількість маркерів у кожному вхідному вузлі більше ваги дуги або дорівнює їй, а для дуги заперечення дорівнює нулю;
- у разі перемикання переходу кількість маркерів кожного вхідного вузла зменшується на відповідну вагу вхідної дуги та залишається незмінною для дуг заперечень. Кількість маркерів кожного вихідного вузла збільшується на вагу відповідної вихідної дуги.

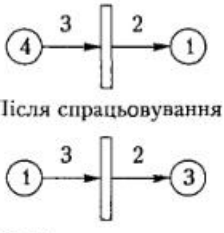


Рис. 3.6. Введення ваги для дуги

3.2.2. Формалізоване зображення моделі за допомогою мережі Петрі

Один із найскладніших етапів створення моделі — це вибір методу її формалізації. Зазвичай існує кілька підходів до зображення моделі системи і мережі Петрі. Продемонструємо на прикладах, як перейти від змістовної постановки задачі до формальної моделі.

Приклад 3.1

Покажемо, як за допомогою мережі Петрі можна синхронізувати кілька асинхронних обчислювальних процесів, що виконуються в пам'яті комп'ютера. Розглянемо роботу комп'ютерної системи, в якій два обчислювальні процеси, $П_1$ і $П_2$, намагаються одночасно записати дані в область пам'яті ($П_1$) і зчитати з неї дані ($П_2$). Проблема, яку потрібно вирішити — синхронізація доступу процесів до пам'яті. Щоб запобігти суперечності, необхідно заборонити одночасний доступ процесів до одної і тої ж області пам'яті. Це означає, що активним може бути тільки один процес, який виконує операцію доступу до пам'яті.

Для формалізації модельованої системи скористаємося семафором, який буде розділяти доступ до області пам'яті. Побудуємо просту мережу Петрі (рис. 3.7), яку можна буде використати для синхронізації цих процесів, і розглянемо її роботу.

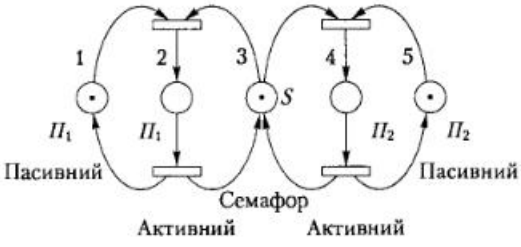


Рис. 3.7. Застосування мережі Петрі для синхронізації асинхронних процесів

Кожний обчислювальний процес може перебувати в одному з двох станів: активному або пасивному, які відповідно позначені двома вузлами. Розташування маркерів у мережі, зображеній на рис. 3.7, показує, що в даний момент обидва процеси знаходяться в пасивному стані. Кожний процес може змінити свій стан за допомогою спрацювання переходу. Зміна стану мережі залежить від вузла семафора — S. Розмітка вихідного стану мережі $M = [1, 0, 1, 0, 1]$. Якщо один процес, наприклад $П_1$, намагається змінити свій стан на активний (записати дані), то він збуджує свій перехід і змінює розмітку мережі на $M = [0, 1, 0, 0, 1]$. Таким чином, збудження переходу процесу $П_2$ (зчитування даних з тієї ж області) не станеться доти, доки процес $П_1$ не перейде в пасивний стан і розмітка мережі знову не зміниться на $M = [1, 0, 1, 0, 1]$.

Приклад 3.2

Покажемо, як за допомогою мереж Петрі можна описати роботу світлофора, що керує дорожнім рухом. Модель світлофора можна використовувати для моделювання систем керування рухом транспорту. Ця модель має відтворювати зміну станів світлофора: зелене світло, жовте світло, червоне світло і червоно-жовте світло. Для формалізації модельованої системи скористаємося простою мережею Петрі, вузли якої будуть визначати стани світлофора, а положення чорного маркера — поточний його стан (рис. 3.8).

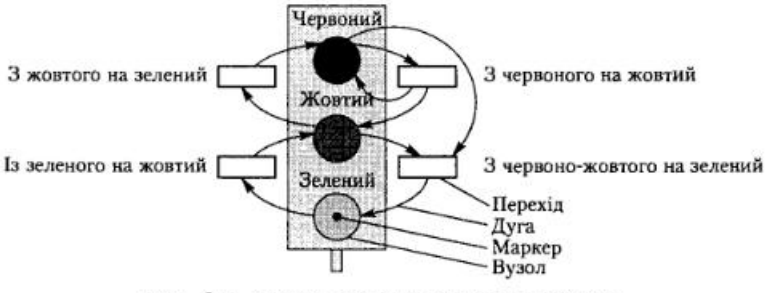


Рис. 3.8. Модель функціонування світлофора

На рис. 3.8 зображено модель роботи світлофора, стан якої відповідає ситуації, коли горить зелене світло (цей стан позначено маркером). Динаміку роботи світлофора, тобто зміну його світла і відповідно положення маркера в мережі Петрі, можна відтворити за допомогою переходів, для яких мають виконуватись правила збудження та спрацьовування. У мережі на рис. 3.8 збудженим є тільки перехід, який відповідає зміні зеленого світла

на жовте. У разі спрацювання переходу маркер із зеленого вузла переміщується в жовтий. Зміна положення маркера приводить до нового стану моделі, що відповідає ситуації, коли горить жовте світло. Із цього стану можливий перехід тільки до стану, який відповідає ситуації, коли горить червоне світло. Таким чином, ця модель описує всі стани світлофора, які змінюються в певній послідовності.

Для формалізації моделі світлофора можна скористатися й іншим підходом, коли всі стани світлофора та переходи можна зобразити графічно за допомогою *скінченного автомата*. Скінченний автомат — це граф станів (вершин) і переходів (дуг). Стан такого автомата позначається маркером в одній із його вершин. Модель функціонування світлофора у вигляді скінченного автомата зображено на рис. 3.9.

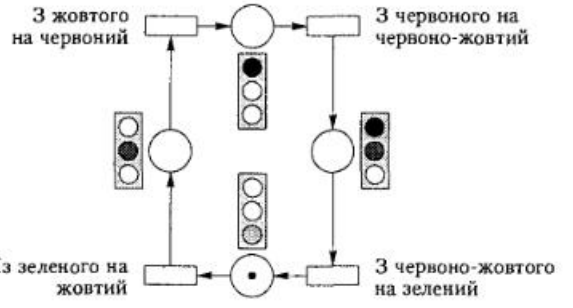


Рис. 3.9. Модель функціонування світлофора у вигляді скінченного автомата

Поточний стан моделі (рис. 3.9) позначається тільки одним маркером (слід звернути увагу, що в мережі Петрі для позначення стану моделі світлофора, в якому одночасно горить червоне і жовте світло, використовувалось два маркери). Множину можливих станів такої моделі можна розширити, наприклад ввести стан світлофора, коли горить зелене і жовте світло. Кількість необхідних маркерів моделі в цьому разі не зміниться, але модель стане більш складною — з'являться ще один вузол і перехід. Таким чином, виникають запитання: чи потрібно всі стани моделі відображати окремими вузлами; чи можна позначати поточний стан моделі за допомогою маркерів різного кольору, як це зроблено на рис. 3.10. Таким чином, можна перейти ще до одного підходу до формалізації моделі світлофора.

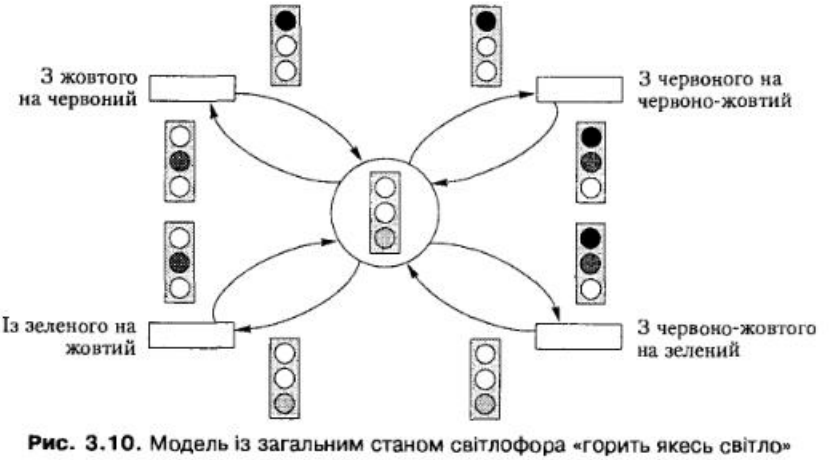


Рис. 3.10. Модель із загальним станом світлофора «горить якесь світло»

Такий підхід передбачає, що стани світлофора в мережі Петрі, які зображувались раніше кількома різними вузлами, зображуються лише одним вузлом, в якому знаходиться маркер відповідного кольору (зелений, жовтий, червоний і червоно-жовтий). На рис. 3.10 зображено мережу Петрі, поточний стан якої визначає зелений маркер, що знаходиться в її вузлі. Стан мережі змінюється тільки після спрацювання переходу від зеленого світла до жовтого. Для збудження цього переходу повинні виконуватись обидві умови збудження переходу:

- у вхідному вузлі має знаходитись зелений маркер (значення виразу дуги, спрямованої від вузла стану світлофора до переходу від зеленого до жовтого світла, графічно зображується зеленим маркером);
- у разі спрацьовування переходу жовтий маркер може переміститись у вузол стану світлофора (значення виразу дуги, спрямованої від вузла стану світлофора до переходу від зеленого до жовтого світла, графічно зображується жовтим маркером).

Відображення станів моделі за допомогою маркерів різного кольору – один з можливих підходів до формальної побудови моделі, що відрізняється від моделі скінченного автомата, в якому кожний допустимий стан моделі зображується окремим вузлом.

Під час розроблення імітаційної моделі, яка повинна якомога точніше відображати дійсність, перед проектувальником стоїть надзвичайно складне завдання – пошук допустимих станів, дій та взаємозалежностей, які існують у модельованій системі, та опис їх за допомогою елементів мережі Петрі. Таким чином, розширюючи можливості базових елементів мережі Петрі (вузлів, дуг і переходів), можна створити модель практично будь-якої дискретно-подійної системи. Такі можливості надають програмні засоби для моделювання систем за допомогою розширених мереж Петрі.

3.2.3. Розширення можливостей вузлів під час моделювання

Подальше розширення можливостей мереж Петрі для виконання завдань моделювання пов'язане з переходом від використання вузлів з маркерами і переходів до використання сховищ даних (вузол з деякою структурою даних) і потоків даних.

Вище зазначалось, що в мережі Петрі всі допустимі стани моделі позначаються вузлами з маркерами. У загальному випадку вузли виступають як сховища даних заданого об'єму, а переходи – як потоки даних. Можливості вузлів мережі Петрі можна значно розширити, якщо маркерам призначати різні типи даних, наприклад рядки символів, цілі або дійсні числа, множини, структури, як це робиться в мовах програмування. Тоді у разі зображення вузлів з такими маркерами необхідно вказувати типи даних і визначати максимальну кількість маркерів кожного типу, які можуть знаходитись у вузлі.

Існує ще одна можливість розширення функцій вузлів – зазначити *режим доступу* до маркерів, тобто задати, яким чином маркери (дані) надходять до вузлів та як вони з них вилучаються. Це дає змогу формувати у вузлах черги маркерів подібно тому, як створюються черги вимог у СМО. У табл. 3.1 наведено основні режими доступу до вузлів.

Таблиця 3.1. Основні режими доступу до вузлів у розширеннях мереж Петрі

Режими доступу	Опис
RAM	Принцип випадкового доступу. Маркер, який надійшов до вузла, розміщується в черзі випадково. У разі спрацювання переходу маркер, який вилучається з черги, вибирається випадково
FIFO	Принцип «першим прийшов — першим покинув». Маркер, який надійшов до вузла, розміщується в черзі останнім. У разі спрацювання переходу вилучається перший маркер черги
LIFO	Принцип «останнім прийшов — першим покинув». Маркер, який надійшов до вузла останнім, розміщується в черзі першим. У разі спрацювання переходу вилучається перший маркер черги
FIFORAM	Принцип «прийшов випадково — першим покинув». Маркер, який надійшов, розміщується в черзі випадково. У разі спрацювання переходу з черги вилучається перший маркер
LIFORAM	Принцип «прийшов випадково — останній першим покинув». Маркер, який надійшов, розміщується в черзі випадково. У разі спрацювання переходу з черги вилучається останній маркер

Вибір режимів формування черги і вилучення із неї маркерів залежить від того, в якій послідовності потрібно перевіряти маркери у вузлах. Тому навряд чи можна дати загальну пораду відносно того, які режими доступу використовувати. У простих мережах Петрі автоматично використовується режим довільного доступу (порядок поставлення маркерів у чергу та порядок їх вилучення для них не має значення).

Розширення можливостей вузлів мереж Петрі є досить зручним засобом для моделювання матеріальних та інформаційних потоків у виробничих системах.

3.3.3. Розширення можливостей дуг під час моделювання

Зміни стану мережі Петрі, тобто переміщення маркерів, спричинені діями переходів, у разі спрацювання яких маркери вилучаються із одних вузлів і поміщаються в інші; таким чином формуються потоки даних. Способи вилучення і розміщення маркерів визначаються *спрямованими дугами*, які з'єднують вузли та переходи. У системах моделювання з кожною дугою можна пов'язати не тільки певний ваговий коефіцієнт (вагу), а й будь-яку функцію, що визначається як вираз дуги. У загальному випадку *вираз дуги* — це функція або значення (числове, символьне і т. ін.), згідно з яким збуджується перехід і змінюється положення маркера під час моделювання.

Якщо під час моделювання значення виразу дуги не змінюється, то для спрацювання переходу необхідна наявність у вузлі маркера, який співпадав би із значенням виразу дуги (рис. 3.11).

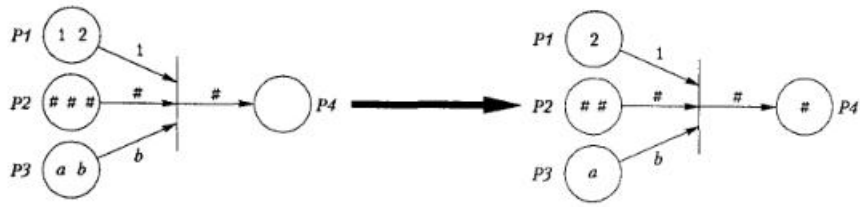


Рис. 3.11. Дуги з постійними виразами

Наприклад, лівий перехід, зображений на рис. 3.11, спрацьовує тільки за умови, що у вузлі *P1* знаходиться хоча б один маркер 1, у вузлі *P2* – хоча б один маркер # (у системі моделювання POSES++ символом # позначається чорний маркер) і у вузлі *P3* – один маркер типу *b*. Після спрацювання переходу зазначені *маркери вилучаються із вузлів P1, P2 та P3, а у вузол P4 поміщається маркер #.*

Дуги, вираз яких не є константою, найчастіше використовуються для моделювання фіксованих потоків даних. Дуги зі змінними виразами застосовуються в тих випадках, коли заздалегідь неможливо задати значення параметрів моделі, наприклад під час пошуку даних і перевірки деяких умов, що змінюються в процесі моделювання. Вираз дуги такого типу називається *змінною відповідності*. Змінні відповідності – це дані особливого типу, і їм не можна присвоїти значення явно, як звичайним змінним. Свої значення вони можуть отримувати тільки в разі збудження переходів.

На рис. 3.12 видно, що змінна x є значенням виразу дуги, яка з'єднує вузол *P1* і перехід *T1*. Оскільки для вузла заданий режим FIFORAM і маркер 1 є першим у черзі вузла *P1*, то змінній відповідності x присвоюється значення 1. Це ж значення присвоюється і виразу дуги, яка з'єднує перехід з вузлом *P2*, для якого теж встановлений режим FIFORAM. Для вузла *P3* задано режим FIFO, тому змінній y присвоюється значення *a*. Таким чином, після спрацювання переходу у вузлі *P4* знаходяться маркери 1 та *a*.

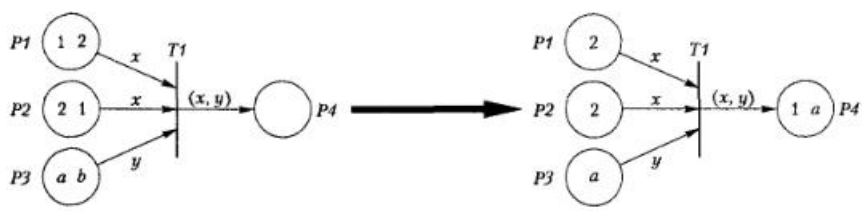


Рис. 3.12. Дуги з використанням змінних відповідності

Правило спрацювання переходу можна описати також таким чином. У вузлі *P1* знайти перший маркер і присвоїти його значення змінній відповідності x, тобто $x = 1$. Потім у вузлі *P2* знайти маркер із значенням, ідентичним $x = 1$. Після цього у вузлі *P3* знайти перший маркер у черзі (у даному випадку маркер *a*) і присвоїти його значення змінній відповідності y, тобто $y = a$. Після спрацювання переходу у вузлах *P1* і *P2* знищити маркер 1, а у вузлі *P3* – маркер *a*. Згенерувати у вузлі *P4* маркери, які складаються з комбінації значень (x, y).

Якщо для переходу не задане обмеження стосовно кількості паралельних спрацювань, то перехід зразу ж може спрацювати повторно. У цьому випадку перевіряються маркери, які залишились у вхідних вузлах переходу, тобто маркери 2 та *b*. Ці значення присвоюються змінним відповідності дуг. У цьому разі значення *x*, які використовувались під час попереднього спрацювання, не враховуються.

Застосування змінних відповідності пов'язане з послідовністю перегляду вхідних дуг переходу. Якби в наведеному вище прикладі першою вхідною дугою переходу *T1* була дуга, що йде від переходу *P2*, а не від *P1*, то після спрацювання переходу у вузлі *P4* знаходились би маркери 2 і *b*.

Покажемо, яким чином можна спростити мережу Петрі, використовуючи вирази для дуг.

Приклад 3.3

Розглянемо ще один приклад — проблему філософів, що обідають, відому з літературних джерел як задачу Эдсгера В. Дейкстри. Це класичний приклад задачі, яку часто використовують для пояснення паралельних процесів і пов'язаних з ними умов блокування, чи взаємовиключення («зависання»), або для пояснення стратегії використання глобального ресурсу в моделях. Одночасно приклад ілюструє, наскільки важкі іноді для розуміння процеси, які виникають у дуже простих паралельних структурах.

За одним круглим столом (рис. 3.13) сидять п'ять філософів. Перед кожним із них стоїть тарілка з їжею, а між тарілками лежать палички для їжі (всього п'ять паличок). Деякий час кожний філософ зайнятий роздумами, а як зголодніє, він починає втамовувати голод. Для їжі кожному філософу потрібно взяти дві палички — спочатку ту, що зліва, а потім ту, що справа. Втамувавши голод, філософ кладе палички на місце і знову переходить до роздумів. Цей процес може тривати досить довго. Щоб поїсти, філософи конкурують між собою за палички для їжі, і, таким чином, палички для їжі є глобальним ресурсом у системі.

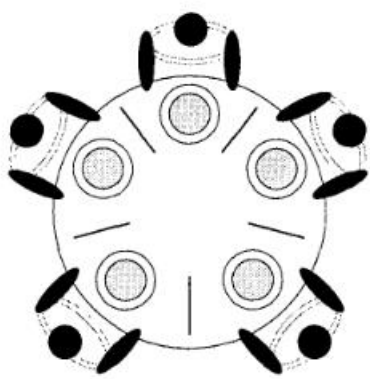


Рис. 3.13. Ілюстрація проблеми філософів, що обідають

Завдання полягає в тому, щоб побудувати таку модель, яка дала б змогу синхронізувати дії всіх філософів таким чином, щоб не виникло взаємного блокування моделі. Блокування може виникнути тоді, коли всі філософи сядуть за стіл і кожен з них намагатиметься одночасно взяти дві палички. У цьому разі виникає ситуація, коли кожен із філософів візьме лише одну паличку, і таким чином, жоден з них не зможе їсти. Такий розподіл глобального ресурсу називається зависанням мережі.

Під час побудови моделі слід ураховувати той важливий факт, що філософ повинен брати спочатку ліву паличку, а потім праву. Обидві палички брати одночасно заборонено. Неузгоджений доступ до паличок визначає критичний процес, який має бути глобально синхронізованим таким чином, щоб уникати конфліктів між філософами, наприклад у випадку, коли два філософи одночасно намагаються взяти одну й ту ж паличку.

Для формалізації моделі за допомогою мережі Петрі спочатку потрібно визначити об'єкти моделі і задати для них структури даних. Об'єктами моделі є п'ять філософів і п'ять паличок для їжі. Для кожного філософа необхідно визначити, які палички для їжі знаходяться зліва й справа від нього. Це можна легко зробити, якщо присвоїти номери філософам і паличкам (від 0 до 4). Паличці, яка лежить зліва від філософа, слід присвоїти той же номер, що й філософу, а паличці справа — на одиницю більший. Виняток буде зроблено тільки для четвертого філософа. Зліва від нього буде розташована паличка під номером 4, а справа — під номером 0. Таким чином, нумерацію паличок можна записати як

$$N_{л.п} = N_{ф},$$

$$N_{п.п} = (N_{ф} + 1) \bmod 5,$$

де $N_{л.п}$, $N_{п.п}$, $N_{ф}$ — номери лівої, правої палички та філософа відповідно.

Очевидно, що одночасно можуть їсти тільки два філософи, тому що п'яти паличок для їжі вистачить тільки для двох.

Далі треба визначити стани об'єктів. Філософ може бути зайнятий роздумами або їжею. Ці можливі стани філософів позначимо як роздуми та їжа. Позначимо також можливі стани паличок як вільна і зайнята. Палички для їжі вважаються зайнятими, якщо філософ знаходиться в стані їжа. На початку моделювання всі філософи знаходились у стані роздуми, а палички для їжі — у стані вільна.

Після того, як визначено всі об'єкти моделі та їх можливі стани, можна задати процеси і дії, які будуть виконувати ці об'єкти (рис. 3.14).

Процес взяти:

- філософ бере спочатку ліву, а потім праву паличку.

Процес покласти:

- філософ кладе обидві палички на місце.

Щоб запобігти блокуванню моделі, кожний філософ повинен переходити від стану роздумів до їжі й навпаки в довільні моменти часу. Отже, для вузлів роздуми і їжа необхідно встановити режим доступу RAM.

Синхронізація доступу до паличок (єдиного глобального ресурсу системи), так само як і розв'язання проблеми блокування моделі, здійснюється автоматично під час перевірок правил дуг для спрацювання переходів. Незалежно від того, виконуються чи ні правила спрацювання переходів, модель ніколи не заблокується.

Отже, дії філософів можна описати такими правилами, номери яких позначено в дужках:

Процес взяти:

- якщо паличка, номер якої збігається з номером філософа, перебуває в стані вільна (1),
- паличка поряд з філософом, номер якої обчислюється як $(N_{ф} + 1) \bmod 5$, теж перебуває в стані вільна (2)
- і відповідний філософ з номером знаходиться в стані роздуми (3),
- то дві палички для їжі, що знаходяться справа і зліва від філософа, позначити як такі, що знаходяться в стані зайняті (1),(2),
- і позначити філософа як такого, що перейшов від стану роздуми до стану їжа (3),(4).

Процес покласти:

- якщо філософ з певним номером перебуває в стані їжа (5),
- змінити його стан від їжа до роздуми (5),(6)
- і палички для їжі, номери яких збігаються з номером філософа та з номером, що обчислюється за формулою $(N_ф + 1) \bmod 5$, перевести в стан вільна (7),(8)

За допомогою наведених правил можна сформулювати вирази дуг у моделі мережі Петрі. На рис. 3.14 зображено мережу Петрі для цієї моделі.

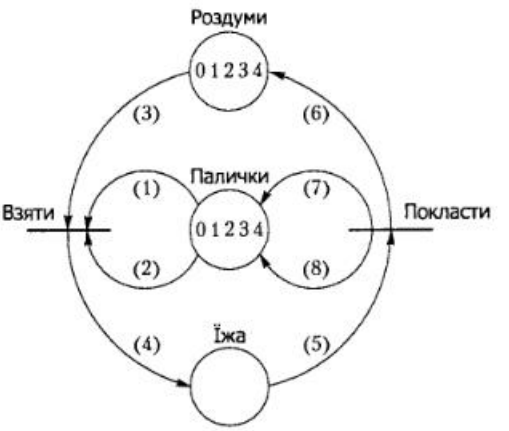


Рис. 3.14. Застосування мережі Петрі для вирішення проблеми філософів, що обідають

Незважаючи на складність такої проблеми, модель досить проста, і перетворення її у програму для моделювання доволі формальне.

3.3.4. Розширення можливостей переходів під час моделювання

Переходи – це активні елементи мережі, які можуть спрацьовувати паралельно. Чи може перехід спрацьовувати паралельно, чи буде він спрацьовувати раніше ніж інші переходи – це повинно бути точно вказано у визначенні переходу під час моделювання. Порядок спрацьовування переходів визначає програма керування моделюванням. Щоб відтворити динаміку роботи системи, котра моделюється, переходи повинні спрацьовувати у відповідні моменти модельного часу.

Перехід може збуджуватися тільки тоді, коли виконано правила збудження. Для цього програма керування моделюванням відшукує маркери у вхідних вузлах кожного переходу, щоб перевірити вимоги виконання правил, визначених вхідними дугами. Якщо необхідна сукупність маркерів у вхідних вузлах переходу знайдена, він збуджується. Перехід спрацьовує тоді, коли виконуються всі чотири умови:

- пропускну здатність паралельності все ще не вичерпано (тобто, як часто залежний від часу перехід може спрацьовувати паралельно самому собі);
- вхідні вузли переходу містять достатню кількість маркерів з необхідними атрибутами, які відповідають виразам вхідних дуг;

- можуть бути виконані логічні умови, сформульовані для атрибутів маркерів вузлів;
- пропускна ємність вихідних вузлів цього переходу така, що вони здатні включити таку кількість маркерів, яка може бути задана числами пропускної здатності вихідних дуг, тобто числами, які визначають кількість маркерів, що потраплять у вихідні вузли.

Потім за допомогою програми моделювання перевіряються маркери у вхідних вузлах, які знищуються в поточний момент модельного часу (для вхідних дуг може бути заданий більш пізній час знищення маркерів), і маркери, що будуть створені для вихідних вузлів, генеруються після того, як закінчиться час переходу (для вихідних дуг може установлюватися час на дузі відносно часу переходу).

Розглянемо мережу Петрі, зображену на рис. 3.15 (чорні маркери позначено символом #). Під час перегляду сукупності маркерів у лівій частині мережі за допомогою програми керування моделюванням перевіряється правило збудження переходу *T1*. Перехід може спрацювати, вилучаючи три чорних маркери з вхідного вузла *P1*, генеруючи один чорний маркер у вихідному вузлі *P2* і п'ять чорних маркерів у вихідному вузлі *P3*. Наступна сукупність маркерів не дає змогу переходу *T1* спрацювати знову, тому що вузол *P1* повинен мати принаймні три чорних маркери. Така сукупність маркерів може виникнути, наприклад, коли збуджується перехід, для якого вузол *P1* є вихідним вузлом.

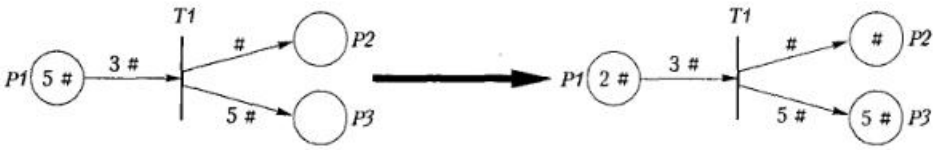


Рис. 3.15. Ілюстрація до правила спрацювання переходів

Слід зробити одне попередження щодо конструкцій циклу. Відповідно до вищезгаданого правила перехід *T1* на рис. 3.16 не буде збудженим, якщо пропускна здатність вузла *P1* дорівнює одиниці. Така пропускна здатність недостатня для передавання маркера, тому що у вузлі *P1* уже є інший маркер. Однак цей маркер знищується вхідною дугою на початку спрацювання так, щоб точно в цей же момент часу пропускна здатність вузла була б достатньою для прийому іншого маркера. Інтуїтивне припущення, що перехід *T1* міг би спрацьовувати, буде дійсним, якщо вхідна дуга перевіряється за допомогою програми керування моделюванням раніше, ніж вихідна.

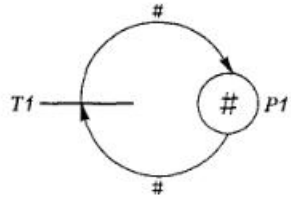


Рис. 3.16. Конструкція циклу

У пакетах моделювання мереж Петрі це так і реалізується. Принцип циклу також працює, коли вхідні й вихідні дуги мають різні значення часу збудження, і час збудження вхідної дуги менший, ніж час збудження вихідної дуги, або дорівнює йому. Бажано уникати конструкцій циклу із залежними від часу дугами.

Спрацювання переходу зазвичай приводить до того, що у вихідних вузлах утворюється нова сукупність маркерів. Таким способом може бути представлено потік даних. Щоб потік даних міг бути випущеним переходом, його слід обробити і представити за допомогою опису переходу.

Переходи, які спрацьовують у модельному часі

Для того щоб моделювати динамічні системи, потрібно мати можливість пов'язати моменти спрацювання переходів з модельним часом (рис. 3.17). Вирізняють переходи кількох типів – *миттєвий, експоненціальний* і *детермінований*. Миттєвий перехід не пов'язаний з моментом часу, і його перемикання здійснюється так, як було описано вище. Такий перехід відображається зазвичай на схемі мережі Петрі як штрих або вузький прямокутник. Якщо час спрацьовування переходу розподілений за експоненціальним законом, то такий перехід зображують незафарбованим прямокутником, а якщо час спрацьовування переходу детермінований – зафарбованим прямокутником. Мережу Петрі, в якій час спрацьовування переходів задано через імовірнісний розподіл, називають стохастичною мережею Петрі.

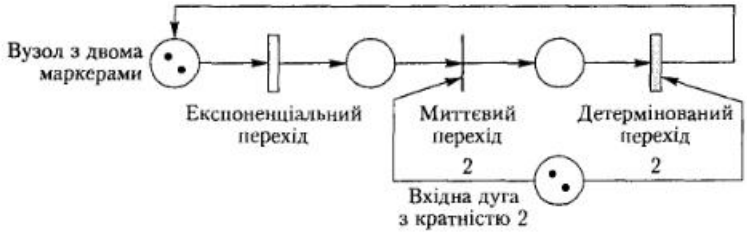


Рис. 3.17. Мережа Петрі з різними типами переходів

Час спрацьовування переходу може бути фіксованим або розрахованим. Він залежить від *масштабу часу*, прийнятому в моделі, і *такту часу*, який визначає розробник моделі. Модельний час просувається з деяким фіксованим значенням такту часу, наприклад кожну секунду. Похибки округлення часу не впливають на точність, якщо час моделювання задається як цілочисловий тип даних.

Детерміноване значення часу спрацьовування може бути безпосередньо призначене атрибуту переходу, тоді як довільне значення часу має бути визначене (точніше, розраховане) у процесі моделювання, тому що воно може залежати від значень маркера. Таким чином, нове довільне значення часу для переходу встановлюється для кожного такту модельного часу, як і для спрацювання переходу.

Фрагмент мережі Петрі, зображений на рис. 3.18, ілюструє окремі можливості спрацювання переходів. Переходи *T1* і *T2* починають збуджуватись у момент часу 0. Після моделювання протягом 10 с перехід *T1* генерує маркер, а перехід *T2* генерує маркер після моделювання протягом 20 с. Після цього жоден з переходів не може

збуджуватися знову, тому що вузол *P1* більше не забезпечує достатньої кількості маркерів.

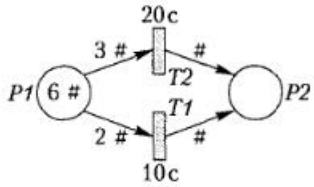


Рис. 3.18. Фрагмент мережі Петрі з часовими переходами

Більшість систем моделювання підтримує різні закони розподілу часу для спрацювання переходів. Наприклад, система POSES ++ підтримує рівномірний, нормальний, логарифмічний розподіли, гамма-розподіл, показовий і пуассонівський розподіли. У разі необхідності інші функції розподілу може бути додано за допомогою мови С.

Слід зауважити, що визначення часу спрацьовування переходу може бути пов'язане з вузлом, тобто вузол *P* може бути пов'язаний з часом δ. У цьому випадку, якщо маркер з'явиться у вузлі *P* у час *t*, то перехід, для якого *P* є вхідним вузлом, може спрацювати тільки в час $t + \delta$.

Паралельність спрацьовування переходів

За достатньої кількості маркерів у вхідних вузлах переходи можуть спрацьовувати паралельно кілька разів, доки в них не вичерпаються маркери. Це дає змогу описувати досить складні реальні процеси і системи. Кількість спрацьовувань переходу задається значенням паралельності в системі, яка моделюється. Наприклад, стандартне значення паралельності в системі POSES ++ дорівнює 2^{32}, тобто може змінюватись від 1 до 4294967295, що є достатнім для досягнення практичних цілей.

Перехід *T1* (рис. 3.19) тричі спрацьовує одночасно, тому що обмеження паралельності не визначене. Якщо задати значення паралельності переходу 2, то він міг би спрацьовувати паралельно тільки двічі. Обмеження паралельності не впливає на переходи, для яких не вказаний час спрацьовування. Коли цей час визначено (наприклад, фіксований час 10 с), функціонування переходу відзначається в статистиці результатів моделювання, наприклад: «У час 0 два спрацьовування починаються одночасно і продовжуються 10 с». У модельний час 10 с починається третє спрацьовування і продовжується воно протягом 10 с (доки тривалість модельного часу не досягне 20 с).

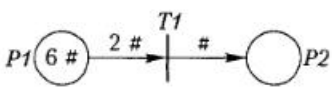


Рис. 3.19. Паралельне спрацьовування переходу

Таких же функціональних можливостей можна досягти, якщо змінити структуру моделі. Для цього є маркери, що діють як ресурси (resources), і використову-

ються, щоб забезпечити максимальну кількість паралельного спрацьовування переходів (на рис. 3.20 два маркери у вузлі *P5* – це ресурси, наприклад пристрої для обслуговування).

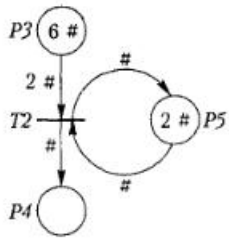


Рис. 3.20. Маркери, що відображають ресурси

Пріоритети переходів

Оскільки для спрацьовування переходу треба перевірити правила збудження, виникає ситуація, коли окремі переходи конкурують між собою за володіння маркерами. Доки не задані ніякі пріоритети, переходи, що конкурують, вирішують проблеми довільно, тобто генератор випадкових чисел з однаковою ймовірністю для всіх переходів визначає, який з них повинен перевірятися спочатку. Якщо для цього переходу виконується правило, то він збуджується раніше за тих, які могли б також збуджуватись.

Приклад конкуруючих переходів наведено на рис. 3.21.

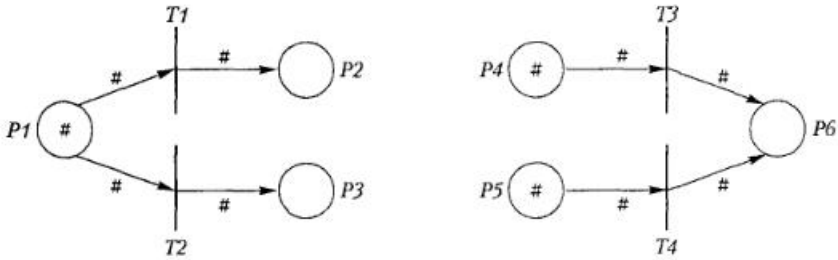


Рис. 3.21. Конкуруючі переходи

Якщо, наприклад, потрібно, щоб переходи *T1* і *T3* спрацьовували раніше, ніж переходи *T2* і *T4*, необхідно встановити їх пріоритети. Стандартний пріоритет переходу в системі POSES ++ має значення 100. Максимально можливий пріоритет дорівнює 1. Усі інші пріоритети відповідно нижчі. Найнижчий можливий пріоритет 2^{32}, тобто пріоритети можуть у системі POSES ++ приймати значення від 1 до 4294967295. У будь-якому випадку програма керування моделюванням перевіряє всі переходи, що мають більш високий пріоритет, і вони спрацьовують раніше, ніж переходи з більш низьким пріоритетом. Якщо окремі переходи мають однаковий пріоритет, програма керування моделюванням вибирає один перехід довільно (відповідно до принципу рівномірного розподілу), і він збуджується.

Приклад 3.4

Розглянемо, як можна використовувати розширені мережі Петрі для моделювання СМО. Нехай необхідно побудувати модель розімкнутої мережі СМО, в якій є два пристрої для обслуговування, до яких надходять вимоги кожні 30 хв. Час обслуговування першим одноканальним пристроєм рівномірно розподілений в інтервалі від 20 до 30 хв. Часові переходи можуть мати позиачки, які вказують розподіли ймовірностей для їх тривалості.

Після обслуговування 80 % вимог надходять до другого одноканального пристрою із середнім часом обслуговування 10 хв, розподіленим за експоненціальним законом, а 20 % вимог уникають обслуговування другим пристроєм і залишають систему. На рис. 3.22 зображено мережу Петрі, яка моделює цю систему.

Маркери використовуються для зображення вимог, ресурсів (пристроїв для обслуговування вимог) і перевірки стану пристроїв. Маркери-вимоги генеруються деяким процесом надходження вимог. Маркери-ресурси застосовуються для початкового маркування вузлів-ресурсів, які повинні використовувати маркери-вимоги. Маркери перевірки використовуються для задання будь-яких спеціальних умов, що можуть впливати на обслуговування вимог.

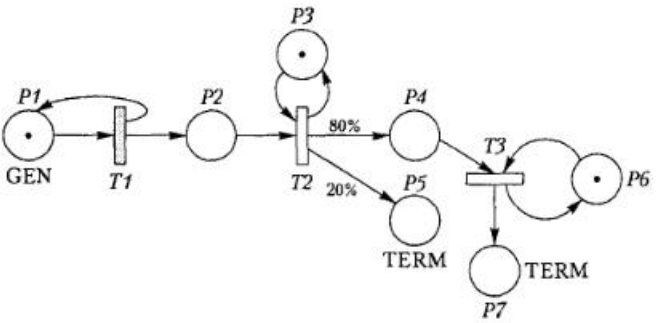


Рис. 3.22. Застосування мережі Петрі для моделювання СМО

У моделі використовуються різні типи вузлів. Вузол *P1*, який генерує вимоги, позначений як GEN, вузли *P5*, *P7*, що знищують вимоги, – як TERM. Вузол *P1* є місцем введення і виведення маркерів для часового переходу *T1*, який збуджується кожні 30 хв. Таким чином, маркери-вимоги надходять кожні 30 хв до вузла *P2*, який моделює чергу для першого одноканального пристрою.

Висновки

- Мережа Петрі є орієнтованим дводольним графом, який має чотири базових елементи: вузли або місця (places), переходи (transitions), дуги (arcs) і маркери (tokens).
- Вузли визначають стан, в якому може знаходитись мережа або її частина.
- Переходи – це активні елементи мережі, які позначають дії, що виконуються під час спрацьовування переходів.
- Розмітка M мережі Петрі – це функція, яка ставить у відповідність маркерам вузлів цілі додатні числа. Суть розмітки полягає в приписуванні кожному вузлу певної кількості маркерів.

- Спрацьовування переходу – це подія, яка змінює розмітку мережі.
- Мережа Петрі знаходиться в *«живому»* стані (живуча розмітка), коли кожен із переходів може збуджуватись нескінченну кількість разів.
- Мережа Петрі знаходиться в *«мертвому»* стані (заблокована), якщо вона не має жодного збудженого переходу.
- Розширення мережі Петрі – це така її модифікація, яка збільшує можливості мережі стосовно опису та моделювання систем.

Контрольні запитання та завдання

1. Побудуйте просту мережу Петрі для роз'вязання задачі про філософів, що обідають (див. приклад 3.2).
2. Технологічна лінія на виробництві створює напівфабрикат з визначених матеріалів і працює періодично. На початку кожного періоду завантажується певна кількість матеріалів, після чого починається основний процес – виробництво, під час якого може виникнути збій. Тоді матеріали, які знаходились у лінії (завантажені на початку періоду) вилучаються з технологічного процесу. Завантажується нова порція матеріалів, і основний процес починається спочатку. На рис. 3.23 наведено модель цього процесу в термінах мережі Петрі. Пріоритети переходів *T2*, *T3* вважати рівними.

   Задайте функцію початкової розмітки M^0 цієї мережі $\langle P, T, F, M^0 \rangle$ і вагу зазначених дуг (J, K, H).

   Варіанти відповіді:

   1) $M^0 = [1, 0, 1, 0, 0, 1, 0]$, $(J = 1, K = 2, H = 1)$;
   2) $M^0 = [1, 1, 2, 0, 0, 2, 2]$, $(J = 1, K = 1, H = 2)$;
   3) $M^0 = [1, 1, 2, 0, 0, 1, 1]$, $(J = 1, K = 1, H = 2)$;
   4) $M^0 = [1, 1, 1, 0, 0, 2, 2]$, $(J = 1, K = 1, H = 1)$.

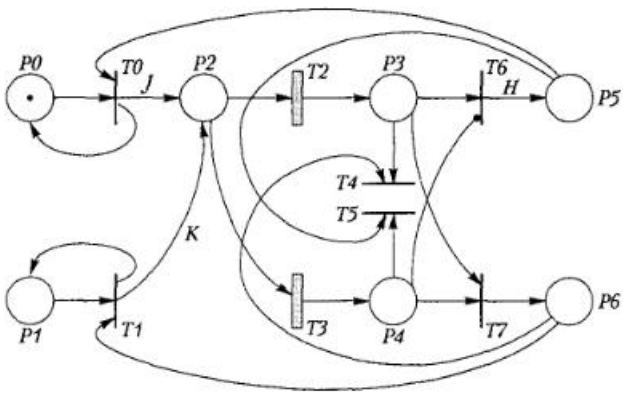


Рис. 3.23. Модель технологічного процесу

3. Розглянемо процес, який є прикладом «нецікавої» гри. Дехто сидить у кімнаті, в який є закрита ваза і відкрита скринька, в обох містяться чорні і червоні кулі. Швидко взявши будь-які дві кулі з вази, він звіряє їх кольори: якщо вони різні, то кладе їх у скриньку, інакше кулі повертаються у вазу і туди ж зі скриньки додається куля того ж кольору, що і дві попередні. Процес повторюється доти, доки в одній із ємностей не буде вистачати куль для продовження гри. На рис. 3.24 наведено модель цього процесу в термінах мереж Петрі. Введені позначення: R – червона куля, B – чорна куля.

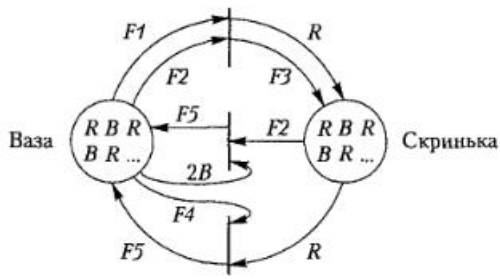


Рис. 3.24. Модель процесу гри

Необхідно встановити вагу дуг F_i, $i = \overline{1, 5}$.

Варіанти відповіді:

$F1 = R, F2 = B, F3 = R, F4 = 2B, F5 = 2B, F6 = 2R;$

$F1 = B, F2 = R, F3 = B, F4 = 2B, F5 = 2R, F6 = 2B;$

$F1 = B, F2 = R, F3 = B, F4 = 2R, F5 = 3B, F6 = 3R;$

$F1 = R, F2 = B, F3 = B, F4 = 2R, F5 = 3B, F6 = 3R.$

4. Клініка має відділення хірургії, де на зміні працює один хірург і три медсестри. Коли з'являється пацієнт, якому роблять операцію, у ній бере участь весь присутній персонал. Після проведення операції пацієнт розміщується в палаті реабілітації, де його доглядає одна з медсестер. Періодично його оглядають один хірург і одна медсестра, доки не з'явиться новий пацієнт.

На рис. 3.25 наведено модель даного процесу в термінах мереж Петрі.

Введені позначення: S – хірург, N – медсестра, P – пацієнт.

Необхідно встановити вагу дуг X_i, $i = \overline{0, 5}$.

Варіанти відповіді:

$X0 = \{P\}, \quad X1 = \{S, N\}, \quad X2 = \{S, 3N\}, \quad X3 = \{S, N\}, \quad X4 = \{N\}, \quad X5 = \{N\};$

$X0 = \{2P\}, X1 = \{S, 2N\}, X2 = \{S, 3N\}, \quad X3 = \{S, N\}, \quad X4 = \{N\}, \quad X5 = \{N\};$

$X0 = \{2P\}, X1 = \{S, N\}, \quad X2 = \{S, 3N\}, \quad X3 = \{S, 2N\}, X4 = \{2N\}, X5 = \{2N\};$

$X0 = \{P\}, \quad X1 = \{S, 2N\}, X2 = \{S, 2N\}, \quad X3 = \{S, 2N\}, X4 = \{2N\}, X5 = \{N\}.$

Чи правильно вибрано функцію початкової розмітки M^0 цієї мережі $[P, T, F, M^0]$ згідно з рис. 3.25?

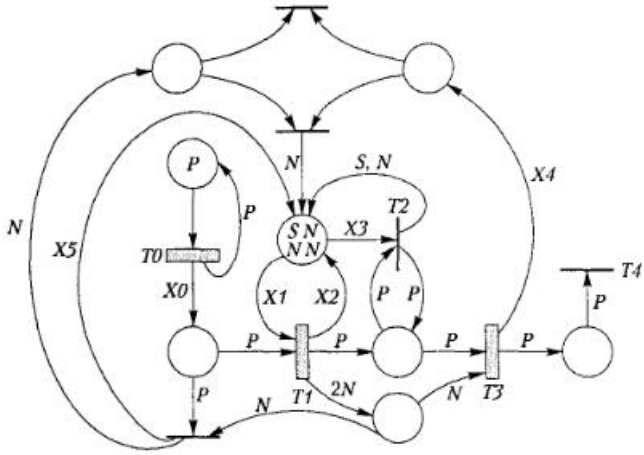


Рис. 3.25. Модель процесу операції

5. Розглянемо процес організації конференції. Спочатку учасників повідомляють, що їм необхідно представити матеріали для реєстрації. Після цього (або в той же час) починається процес прийому матеріалів різних авторів, який триває лише певний період часу. Потім матеріали реєструються. У кожному конкретному випадку приймається рішення про запрошення автора. Можна прийняти лише 80 % матеріалів для доповідей. Яка з моделей цього процесу, зображених на рис. 3.26, *а–в* в термінах мереж Петрі, є правильною?

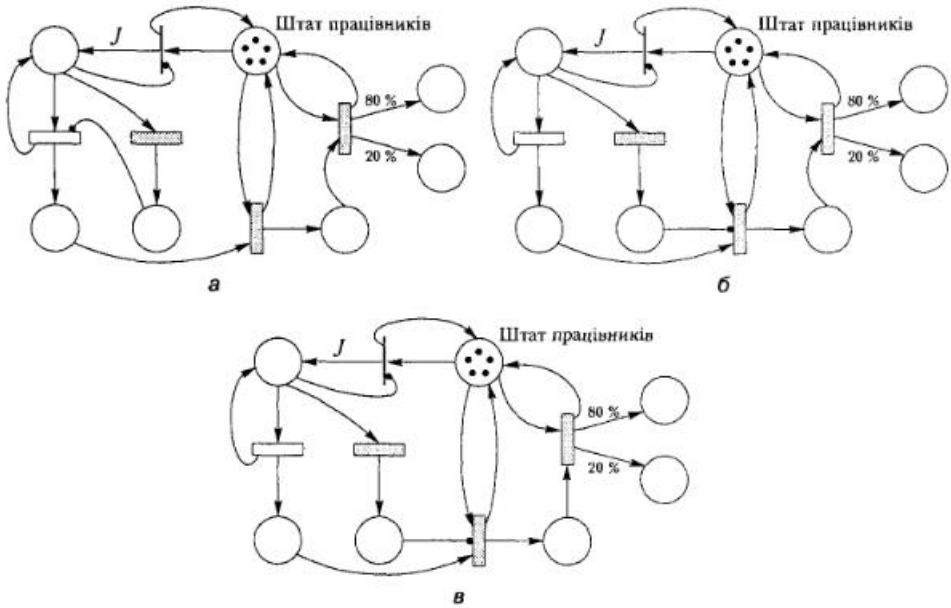


Рис. 3.26. Моделі процесів організації конференції: варіант 1 (*а*); варіант 2 (*б*); варіант 3 (*в*)

Розділ 4

Імовірнісне моделювання

- Метод статистичних випробувань
- Методи генерування випадкових чисел
- Програмні генератори випадкових чисел
- Моделювання випадкових величин із заданими законами розподілу
- Моделювання випадкових процесів і векторів
- Аналіз результатів моделювання

4.1. Метод статистичних випробувань

Метод статистичних випробувань – це числовий метод математичного моделювання випадкових величин, який передбачає безпосереднє включення випадкового фактора в процес моделювання і є його істотним елементом. Вплив випадкових факторів на систему моделюється за допомогою випадкових чисел. Результатом моделювання є випадкові процеси або величини, які характеризують систему, що моделюється. Щоб їх імовірнісні характеристики (імовірність деяких подій, математичне сподівання, дисперсія випадкових величин, імовірності попадання випадкової величини в задану область та ін.) співпадали з аналогічними параметрами реальної системи або процесу під час моделювання потрібно отримати велику кількість реалізацій випадкових величин або процесів. Таким чином, метод полягає в багатократному проведенні випробувань побудованої ймовірнісної моделі і подальшій статистичній обробці результатів моделювання з метою визначення шуканих характеристик розглядуваного процесу у вигляді оцінок його параметрів. Точність оцінок цих параметрів визначає ступінь наближення розв'язку задачі до ймовірносних характеристик.

На практиці метод статистичних випробувань доцільно використовувати в таких випадках, коли:

- розв'язувати задачу цим методом простіше, ніж будь-яким іншим;
- досліджується система, функціонування якої визначається багатьма ймовірнісними параметрами елементарних явищ;
- важко або неможливо побудувати аналітичну ймовірнісну модель системи.

Важливою властивістю цього методу є те, що для звичайних числових методів обсяг обчислень зростає в разі збільшення розмірності задачі приблизно як по-

казникова функція розмірності задачі, а для методу статистичних випробувань – *лише як лінійна функція розмірності.*

Незалежно від типу досліджуваної моделі системи, застосовуючи метод статистичних випробувань, необхідно виконати такі кроки.

1. Визначити, що являтиме собою кожне випробування і зазначити, яке випробування буде успішним, а яке – ні.
2. Обчислити кількість випробувань, які необхідно провести, щоб отримати результати із заданою точністю, і провести ці випробування.
3. Виконати статистичну обробку результатів випробувань та обчислити оцінки необхідних статистичних характеристик.
4. Проаналізувати точність отриманих статистичних характеристик.

Така послідовність кроків є обов'язковою під час розв'язування будь-якої задачі за допомогою методу статистичних випробувань. Однак конкретний зміст цих кроків залежить від поставленого завдання та типу досліджуваної системи. У цьому разі метод завжди потребує використання генераторів випадкових чисел із заданим законом розподілу.

У методі статистичних випробувань особливе значення відіграють випадкові числа, рівномірно розподілені в інтервалі [0, 1]. Найважливіша їх властивість полягає в тому, що за їх допомогою можна отримати вибіркові значення, які мають будь-який інший розподіл, або промоделювати випадковий процес з різними статистичними властивостями.

Отже, для використання методу статистичних випробувань необхідні певні можливості, а саме:

- ✦ генерувати випадкові числа, рівномірно розподілені в інтервалі [0, 1];
- ✦ описувати модельовані випадкові явища функціями розподілу ймовірностей та ймовірнісними процесами;
- ✦ мати методи отримання випадкових величин функцій розподілу ймовірностей (дискретних і неперервних), які базуються на випадкових числах, рівномірно розподілених у інтервалі [0, 1];
- ✦ оцінювати статистичні характеристики випадкових величин з отриманих за допомогою методу статистичних випробувань чисел вибіркової послідовності;
- ✦ визначати точність отриманих статистичних оцінок як функцій від числа випробувань.

Випадкові числа, рівномірно розподілені в інтервалі [0, 1], мають дві основні властивості:

1. Якщо r_i ($i = 1, 2, 3, ...$) – випадкові числа, рівномірно розподілені в інтервалі [0, 1], то їх кумулятивний розподіл F (за визначенням $F(r_i) = P, (r_i < r)$), задовольняє співвідношенням:

$$F(r_i) = \begin{cases} r_i, & \text{якщо} \quad 0 \le r_i \le 1, \quad \forall i, \\ 0, & \text{якщо} \quad r_i < 0, \quad \forall i, \\ 1, & \text{якщо} \quad r_i > 1, \quad \forall i. \end{cases}$$

Слід зауважити, що теоретично ці випадкові числа повинні бути вибірковими значеннями неперервної величини з функцією щільності, визначеною таким чином:

$$f(r) = \begin{cases} 1, & \text{для всіх } 0 \le r \le 1, \\ 0, & \text{для всіх інших значень.} \end{cases}$$

Насправді ж під час комп'ютерного моделювання використовуються тільки дискретні значення, в яких після десяткової коми є фіксована кількість десяткових знаків.

2. Випадкові числа $r_1, r_2, ..., r_n$ є незалежними, якщо їх сумісний кумулятивний розподіл G можна подати як добуток окремих функцій розподілу:

$$G(r_1, r_2, ..., r_n) = F_1(r_1)\, F_2(r_2) \dots F_n(r_n),$$

або, враховуючи, що n випадкових чисел мають однакові розподіли, можна записати:

$$G(r_1, r_2, ..., r_n) = F(r_1)\, F(r_2) \dots F(r_n),$$

Методи генерування випадкових чисел, рівномірно розподілених у інтервалі [0, 1], буде описано нижче.

Розглянемо кілька задач, для розв'язування яких можна застосувати метод статистичних випробувань.

Приклад 4.1

Необхідно знайти площу фігури (рис. 4.1), обмежену функцією $y = f(x)$ та осями координат $0x$ і $0y$.

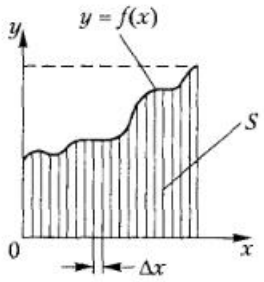


Рис. 4.1. Схема обчислення інтеграла

У числових методах для інтегрування використовується наближене зображення інтеграла у вигляді квадратурної формули. Одним із найпростіших є метод прямокутників. У разі використання методу прямокутників інтеграл апроксимується такою формулою:

$$S = \sum_{i=1}^{n} f(x_i)\, \Delta x.$$

Ця формула і є формулою числового інтегрування. Чим більша кількість інтервалів n і менший крок Δx, тим точніше можна обчислити площу S.

Тепер покажемо, як можна розв'язати цю задачу за допомогою методу статистичних випробувань. Спочатку пронормуємо функцію $y = f(x)$ так, щоб уписати її в одиничний квадрат[1]. Припустимо, що ξ — деяка випадкова величина, рівномірно розподілена в інтервалі $[0, 1]$. Тоді ймовірність попадання значення ξ в будь-який відрізок $[a, b] \in [0, 1]$ буде залежати тільки від довжини відрізка $[a, b]$, а не від місця його розташування в інтервалі $[0, 1]$, тобто ймовірність того, що вибіркове значення випадкової величини ξ потрапить у деякий відрізок $0 \le a \le b \le 1$, дорівнюватиме довжині цього відрізка: $P(a \le \xi \le b) = \int_a^b d\xi = b - a$.

Будемо використовувати одне значення випадкової величини ξ для визначення координати x_i, а друге — для визначення координати y_i. Таким чином, пара значень випадкової величини ξ задаватиме на площині точку з координатами (x_i, y_i). Ймовірність попадання цієї точки в деяку область одиничного квадрата пропорційна площі цієї області та не залежить від місця розташування області в одиничному квадраті.

Проведемо N випробувань. Випробування будемо вважати успішним, якщо точка з координатами (x_i, y_i) потрапить в область під кривою $y = f(x)$ або на неї. Підрахуємо кількість успішних випробувань, позначимо їх через m і визначимо частість успішних випробувань — m/N. На рис. 4.1 видно, що у разі збільшення кількості випробувань ця величина наближається до ймовірності попадання точки в заштриховану область $P = S/S_{\text{од.кв}} = S \approx m/N$, де $S_{\text{од.кв}}$ — площа одиничного квадрата. Таким чином, згідно з теоремою Бернуллі

$$\lim_{N \to \infty} P\left(\left|\frac{m}{N} - p\right| < \varepsilon\right) = 1.$$

У разі прямування кількості випробувань N до нескінченності частість успішних випробувань буде відрізнятись від імовірності p на нескінченно малу величину ε. Отже, можна вважати, що m/N — наближене значення шуканої площі S.

Цей приклад демонструє те, як метод статистичних випробувань може бути використано під час розв'язування детермінованих задач. На практиці такий підхід використовується для знаходження площ або об'ємів деяких багатовимірних фігур, які утворюються у випадку перетину різних геометричних тіл. У цьому разі число випробувань N, які необхідно провести для обчислення площі або об'єму, не залежить від кратності визначеного інтеграла.

Приклад 4.2

Припустимо, що чотири стрільці одночасно стріляють у рухому ціль. Імовірність влучення в ціль кожним стрільцем дорівнює 0,5. Ціль вважається враженою, якщо в неї влучило два або більше стрільців. Потрібно знайти ймовірність ураження цілі.

Використовуючи методи теорії ймовірностей, цю задачу досить легко розв'язати аналітично. Дійсно, імовірність ураження цілі одним пострілом $p_{\text{вр}} = 1 - p_{\text{невр}}$, де $p_{\text{невр}}$ — імовірність того, що ціль не буде вражена взагалі, визначається за формулою

$$p_{\text{невр}} = 0{,}5^4 + C_4^1 \cdot 0{,}5^1 \cdot 0{,}5^3 = 0{,}5^4 + 4 \cdot 0{,}5^1 \cdot 0{,}5^3 = 0{,}3125.$$

Звідси ймовірність ураження цілі

$$p_{\text{вр}} = 1 - p_{\text{невр}} = 1 - 0{,}3125 = 0{,}6875.$$

---

[1] Таке нормування необхідне для того, щоб спростити процедуру випробувань. Усі сучасні мови моделювання та програмування загального призначення мають вбудовані засоби генерування рівномірно розподілених незалежних випадкових величин у інтервалі $[0, 1]$.

Тепер покажемо, як розв'язати цю задачу за допомогою методу статистичних випробувань. Процедуру розіграшу можна реалізувати, одночасно підкидаючи чотири монети. Для моделювання підкидання однієї монети використовується одне значення r_i. Якщо $r_i < p$, вважаємо, що монета падає лицевим боком, і, таким чином, стрілець влучив у ціль. Інакше вважаємо, що стрілець промахнувся. Одне випробуваиня — це підкидання чотирьох монет. Зробимо N випробувань і позначимо через m число успішних випробувань (дві або більше монет упали лицевим боком, що свідчить про те, що в ціль улучило два або більше стрільців). Тоді, згідно з теоремою Бернуллі, $p_{вр} \approx m/N$. У разі значного збільшення числа випробувань N і при будь-якому значенні є частість враження пілі буде збігатись до ймовірності $p_{вр} \to 0{,}6875$.

Приклад 4.3

Розглянемо більш складну задачу, яку розв'язати аналітично досить важко. Нехай є деяка ціль довільної форми загальною площею S, на яку бомбардувальники скидають n бомб. Площа враження кожної бомби — це круг з радіусом R (рис. 4.2). Ціль вважається враженою, якщо зруйновано K відсотків її площі S. Необхідно знайти ймовірність ураження цілі.

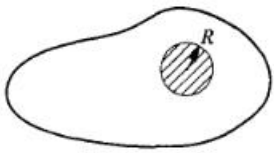


Рис. 4.2. Схематичне зображення враження цілі бомбою

Для цього розглянемо область улучення бомб. За допомогою генератора випадкових чисел отримаємо координати падіння n бомб. Біля кожної точки падіння опишемо коло радіусом R (див. рис. 4.3) і визначимо площу враження, яку заштриховано на рисунку. Площу враження можна легко обчислити, використовуючи методи геометрії. Якщо площа враження становить K відсотків (або більше) загальної площі цілі S, то ціль вважається враженою, а випробування — успішним. Інакше ціль вважається неураженою, а випробування — неуспішним.

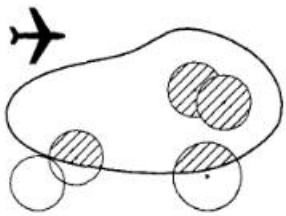


Рис. 4.3. Схематичне вирішення задачі

Проведемо N випробувань, моделюючи кожного разу координати n точок падіння бомб, і підрахуємо кількість випробувань, під час яких ціль була враженою. Тоді, згідно з теоремою Бернуллі, ймовірність враження цілі визначається за формулою $p_{вр} \approx m/N$, де m — кількість випробувань, за яких ціль була вражена. Оцінку математичного сподівання площі враження цілі можна визначити як

$$\bar{S} = \frac{1}{N}\sum_{i=1}^{N} S_i,$$

де S_i — площа враження під час i-го випробування. Згідно із законом великих чисел, якщо $N \to \infty$, то оцінка буде наближатись до математичного сподівання.

За допомогою методу статистичних випробувань можна обчислити будь-які характеристики випадкових величин і процесів. Крім того, цей метод можна застосовувати для розв'язування не тільки ймовірнісних, а й детермінованих задач. Але під час його застосування слід пам'ятати, що для отримання результату з наперед заданою точністю необхідно провести велику кількість випробувань, для чого потрібні довгі послідовності випадкових чисел.

4.2. Генератори випадкових чисел

Найбільше прикладів генерування випадкових чисел можна знайти в ігровому бізнесі. Це номери в спортивних лотереях, числа, які випадають на рулетці, варіанти розкладу карт тощо. Більшість комп'ютерних ігор теж базується на випадкових числах.

4.2.1. Типи генераторів

Без комп'ютера використання випадкових чисел, передбачене методом статистичних випробувань, не має сенсу, тому генератори випадкових чисел повинні бути безпосередньо з'єднані з комп'ютером. Це можна зробити за допомогою апаратних приставок до комп'ютера (апаратні методи) або спеціальних програм (програмні методи). Крім того, під час моделювання можна використати готові таблиці випадкових чисел, які слід розміщати в пам'яті комп'ютера або на зовнішньому накопичувачі.

Апаратні методи генерування випадкових чисел базуються на використанні деяких фізичних явищ (наприклад, шумів електронних приладів, радіоактивного випромінення та ін.). Під час застосування апаратних генераторів випадковий електричний сигнал перетворюють у двійковий код, який уводиться в комп'ютер за допомогою спеціальних аналого-цифрових перетворювачів. Один з найбільш поширених методів – це використання шумів електронних приладів. Якщо на підсилювач не подавати ніякого сигналу та увімкнути його на повну потужність, то буде чутно шипіння (шум). Це і є шум електронних елементів підсилювача, який є випадковим процесом. Цей неперервний сигнал можна перетворити в дискретний. Існують різні схеми перетворення випадкового сигналу в послідовність двійкових цифр [47]. У більшості випадків його підсилюють і встановлюють граничне значення напруги шумового сигналу, перевищення якого можна вважати значенням двійкової одиниці на деякому малому проміжку часу t. У протилежному випадку отримуємо двійковий нуль. Для отримання m-розрядного випадкового двійкового числа провадиться m вимірювань неперервного сигналу у фіксовані моменти часу $t_1, t_2, ..., t_m$.

Вбудовані в комп'ютери апаратні генератори випадкових чисел останнім часом часто використовуються в системах захисту інформації. Прикладом застосування таких генераторів для забезпечення конфіденційності, цілісності та достовірності електронної інформації, яка зберігається в комп'ютері або передається по мережі, є пристрій для шифрування даних PadLock, інтегрований у деякі моделі

процесорів, розроблених компанією Intel. Пристрій має інтерфейс прикладного рівня, що дає змогу розробникам програмного забезпечення отримувати випадкові числа без використання програмних драйверів. Такий спосіб отримання високоякісних випадкових послідовностей простіший та ефективніший, ніж використання апаратно-програмної RNG (Random Number Generator) архітектури і суто програмних генераторів, що особливо важливо під час побудови захищених і криптографічних програм.

Табличний метод. У 1955 році корпорація «Ренд» опублікувала таблиці випадкових чисел, які мали мільйон значень. Для заповнення цих таблиць застосовувались апаратні методи. Дані цих таблиць можна використовувати під час моделювання систем за допомогою методу статистичних випробувань. У сучасних комп'ютерах ці таблиці можна зберігати на зовнішніх носіях або навіть в основній пам'яті. Головним недоліком табличного методу є те, що під час його використання витрачаються значні об'єми основної пам'яті комп'ютера.

Найбільш розповсюдженими на практиці є *програмні генератори*, які дають змогу отримувати послідовності *випадкових чисел* за рекурентними формулами. Якщо бути абсолютно точним, то числа, які виробляють програмні генератори, насправді є *псевдовипадковими* («псевдо» у перекладі з грецької – *нібито*). Так їх називають тому, що алгоритми їх отримання завжди є детермінованими.

Загалом же програмні генератори повинні задовольняти таким вимогам:

- генерувати статистично незалежні випадкові числа, рівномірно розподілені в інтервалі [0, 1];
- мати можливість відтворювати задані послідовності випадкових чисел;
- затрати ресурсів процесора на роботу генератора повинні бути мінімальними;
- легко створювати незалежні послідовності випадкових чисел (потоки).

Слід звернути увагу на те, що більшість програмних генераторів виробляють випадкові числа, рівномірно розподілені в інтервалі [0, 1]. Необхідність моделювання таких чисел обумовлена тим, що на їх основі можна отримати випадкові числа практично будь-яких розподілів. Потрібно також мати на увазі, що випадкові числа, які виробляють програмні генератори, є *квазірівномірно* розподіленими («квазі» у перекладі з латинської – *майже*). Причина в тому, що вони створюються комп'ютером, кількість двійкових розрядів якого обмежена, і за його допомогою можна зобразити тільки дискретні (а не неперервні) значення з діапазону від 0 до 1.

Якість роботи генераторів визначається статистичними властивостями послідовностей випадкових чисел, які він виробляє, – незалежністю і випадковістю. Властивості послідовностей перевіряються за статистичними критеріями, детально описаними нижче.

Здатність відтворювання послідовності випадкових чисел полягає в тому, що за однакових початкових умов і параметрів генератор повинен відтворювати одні й ті ж послідовності псевдовипадкових чисел. Ідентичні послідовності випадкових чисел рекомендується використовувати у випадку, коли потрібно порівняти альтернативні варіанти систем, що моделюються, і налагодити програми. Однак можливість відтворення не завжди бажана під час моделювання систем і в комп'ютерних іграх (як це було в перших версіях відомої гри «Тетріс», коли кожна гра

починалась з тієї ж послідовності фігур). Для усунення такого недоліку початкові значення величин, необхідних для запуску програмного генератора, рекомендується брати з таймера комп'ютера.

Під час дослідження складних систем виникає необхідність у моделюванні послідовностей випадкових чисел великої довжини. Для їх створення потрібні швидкодіючі алгоритми генерування з мінімальними вимогами до ресурсів комп'ютера. Інколи дослідники з невеликим досвідом роботи використовують під час моделювання складних систем один і той же генератор, звертаючись до нього з різних місць програми. Однак це часто призводить до того, що процес моделювання швидко вироджується у зв'язку з виходом псевдовипадкової послідовності за межі аперіодичності. Тому для моделювання різних випадкових факторів бажано мати окремі послідовності (набори значень), які відтворювались би одним і тим же генератором, але за різних значень параметрів.

У більшості генераторів псевдовипадкових чисел x_i використовується рекурентна процедура $x_{i+1} = f(x_i)$. Найпростішим та найдавнішим серед таких генераторів є генератор фон Неймана та Метрополіса, робота якого базувалась на методі середин квадратів. Пояснімо суть цього методу.

Зобразимо умовно довільне чотирирозрядне десяткове число як x x x x. Піднесемо це число до квадрату і в отриманому результаті відкинемо по дві цифри зліва і справа:

$$\begin{array}{rl} & \times\ \underline{\mathrm{x\ x\ x\ x}} \\ & \underline{\mathrm{x\ x\ x\ x}} \\ \mathrm{x\ x} \mid & \mathrm{x\ x\ x\ x} \mid \mathrm{x\ x} \\ & \times\ \underline{\mathrm{x\ x\ x\ x}} \end{array}$$

Чотири цифри, які залишились, і є новим випадковим числом. Якщо результатом множення є число з кількістю цифр менше восьми, зліва дописуються додаткові нулі. Реалізувати програмно цей генератор дуже просто, але він має ряд вад:

- якщо початкове число парне, то може відбутися виродження послідовності, тобто починаючи з деякого значення всі наступні дорівнюватимуть нулю (спробуйте взяти як початкове число 4500);
- числа, які виробляє генератор, є сильно корельованими.

4.2.2. Лінійні конгруентні генератори

Зважаючи на ці вади, на практиці використовують більш складні програмні генератори. У більшості сучасних програмних генераторів використовується властивість *конгруентності*, яка полягає в тому, що два цілих числа A і B є конгруентними за модулем m, якщо їх різниця $(A - B)$ є числом, яке ділиться на m без остачі (тобто є кратним m). Записується це так:

$$A = B \,(\mathrm{mod}\ m).$$

Наприклад, щоб знайти число, конгруентне з числом 134 за модулем 10, необхідно знайти цілочислову *остачу* від ділення 134 на 10, яка дорівнює 4.

Наведемо кілька прикладів обчислення конгруентних значень для різних m:

$$12 \equiv 5 \pmod 7;\ 35 \equiv 5 \pmod{10};\ 125 \equiv 5 \pmod{10}.$$

Серед методів генерування випадкових чисел найбільш поширеним є лінійний мультиплікативний конгруентний метод:

$$x_{i+1} = (ax_i + c) \bmod m, \tag{4.1}$$

де $i = 1, 2, ...$; a, c і m – цілі константи. Щоб отримати нове число, необхідно взяти псевдовипадкове число x_i (або задати вихідне x_0), помножити його на коефіцієнт a, додати константу c і взяти модуль отриманого числа за m, тобто розділити на m, і отримати остачу. Ця остача і буде наступним псевдовипадковим числом x_{i+1}. У разі правильного підбору параметрів цей генератор повертає випадкові числа від 0 до $m - 1$.

Отримані за формулою (4.1) значення x_{i+1} належать до діапазону $0 < x_{i+1} < m - 1$ і мають рівномірний дискретний розподіл. Для того щоб отримати випадкове значення r_{i+1} з інтервалу [0, 1], необхідно число x_{i+1} розділити на m. У цьому разі всі значення m, c, a, x_0 повинні бути додатними й задовольняти умовам: $0 < m$; $a < m$; $c < m$; $x_0 < m$. Отримана за формулою (4.1) послідовність називається *лінійною конгруентною послідовністю*.

Однією із вад лінійних конгруентних генераторів є те, що отримані випадкові числа x_{i+1} суттєво залежать від значень m, c, a, x_0 і обчислюються за однією й тією ж формулою (4.1), тобто не є абсолютно випадковими. Але незважаючи на те що алгоритм їх отримання є детермінованим, за умови відповідного вибору констант m, c, a послідовність чисел x_{i+1}, на основі яких отримують значення r_{i+1}, повністю задовольнятиме більшості статистичних критеріїв.

Ще одна вада цих генераторів стосується того, що випадкові числа r_{i+1}, отримані за допомогою генератора, можуть приймати тільки дробово-раціональні значення – $0; 1/m; 2/m; ...; (m-1)/m$. Більше того, числа r_{i+1} можуть приймати лише деякі з указаних значень залежно від вибраних параметрів m, c, a і x_0, а також від того, як реалізується операція ділення чисел з плаваючою комою на число m у комп'ютері, тобто залежно від типу комп'ютера і системи програмування. Наприклад, якщо $m = 10, x_0 = a = c = 7$, то отримаємо послідовність 7; 6; 9; 0; 7; 6; 9; 0, ..., яка не є випадковою. Це свідчить про важливість правильного вибору значень констант m, c, a і x_0. Правильно підібрані значення іноді називають *магічними числами*.

Наведений приклад ілюструє й те, що конгруентна послідовність завжди є циклічною, тобто вона починає повторюватися через певну кількість випадкових чисел. Кількість значень, після яких випадкові числа починають повторюватися, називається *повним періодом* генератора і є основним його параметром. Значення повного періоду залежать від розрядності комп'ютера, а також від значень m, c, a і x_0. Існує теорема, яка визначає умови існування повного періоду генератора, а саме:

- числа c і m повинні бути взаємно простими, тобто мати взаємний дільник 1;
- значення $b = a - 1$ має бути кратним q для кожного простого q, бути дільником m;
- значення b має бути кратним 4, якщо m кратне 4.

Достатність цих умов уперше було доведено Халлом (Hull) і Добеллом (Dobell). Повне доведення теореми можна знайти в літературі [23].

Якщо $c > 0$, то генератор називається *мішаним*, а якщо $c = 0$ — *мультиплікативним*.

Розглянемо, як потрібно вибирати параметри лінійного конгруентного генератора, щоб отримати послідовність з повним періодом. Для отримання такої послідовності необхідно вибирати значення $m = 2^g - 1$, де g — довжина розрядної сітки комп'ютера. Для 32-розрядного комп'ютера m — найбільше ціле число, яке може бути відтворене в ньому, дане число дорівнює $2^{31} - 1 = 2147483647$ (один розряд відводиться під знак числа). У цьому разі ділення x_{i+1}/m виконувати не обов'язково. Якщо в результаті роботи генератора буде отримане число x_{i+1}, яке більше ніж те, що може бути відтворене в комп'ютері, виникне переповнення розрядної сітки. Це призведе до втрати крайніх лівих двійкових знаків цілого числа, які перевищили допустимий розмір. Однак розряди, що залишились, саме і є значеннями $x_{i+1} \pmod{2^g}$. Таким чином, під час генерування замість операції ділення можна скористатись переповненням розрядної сітки.

Стосовно константи c теорема стверджує, що для отримання послідовності з повним періодом генератора значення c повинне бути непарним, і, крім того, $a - 1$ має ділитися на 4. Для такого генератора початкове значення x_0 може бути довільним і лежати в діапазоні від 0 до $m - 1$. Якщо $c = 0$, то отримуємо мультиплікативний конгруентний метод, який передбачає використання таких рекурентних виразів:

$$x_i = ax_i \pmod{m}. \tag{4.2}$$

Цей метод більш швидкодіючий, ніж попередній, але він не дає послідовності з повним періодом. Дійсно, з виразу (4.2) видно, що значення $x_{i+1} = 0$ може з'явитись тільки в тому випадку, якщо послідовність вироджується в нуль. Взагалі, якщо d — будь-який дільник m і x_i кратне d, всі наступні елементи мультиплікативної послідовності $x_{i+1}, x_{i+2}, \ldots$ будуть кратними d. Таким чином, якщо $c = 0$, потрібно, щоб x_i і m були взаємно простими числами з m і знаходились між 0 і m.

Що стосується вибору a, то у випадку, коли $m = 2^g$, де $g \geq 4$, викладені в праці [23] умови приводять до єдиної вимоги стосовно значення a:

$$a \equiv 3 \pmod{8} \quad \text{або} \quad a \equiv 5 \pmod{8}.$$

У цьому випадку четверта частина всіх можливих значень множників дає довжину періоду, що дорівнює $m/4$, яка й буде максимальним періодом генератора.

Наведемо одну з найкращих програм генерування послідовностей рівномірно розподілених випадкових чисел.

```
/* Програма мовою C для TT800: версія від 8 липня 1996 р. */
/* M. Matsumoto, email: matumoto@math.keio.ac.jp */
/* genrand() генерує одне псевдовипадкове число */
/* з подвійною точністю */
/* рівномірно розподілене в інтервалі [0, 1] */
/* для кожного запиту. Можна вибирати будь-які */
/* 25 початкових значень */
/* окрім всіх нулів. */
```

```
/* Див.: ACM Transactions on Modelling and Computer Simulation, */
/* Vol. 4, No. 3, 1994, pages 254-266. */

#include
#define N 25
#define M 7

double
genrand()
{
 unsigned long y;
 static int k = 0;
 static unsigned long x[N]={ /* Початкові 25 значень за бажанням */
 0x95f24dab, 0x0b685215, 0xe76ccae7, 0xaf3ec239, 0x715fad23,
 0x24a590ad, 0x69e4b5ef, 0xbf456141, 0x96bc1b7b, 0xa7bdf825,
 0xc1de75b7, 0x8858a9c9, 0x2da87693, 0xb657f9dd, 0xffdc8a9f,
 0x8121da71, 0x8b823ecb, 0x885d05f5, 0x4e20cd47, 0x5a9ad5d9,
 0x512c0c03, 0xea857ccd, 0x4cc1d30f, 0x8891a8a1, 0xa6b7aadb
 };
 static unsigned long mag01[2]={
 0x0, 0x8ebfd028 /* Це чарівний вектор `a', що не змінюється*/
 };
 if (k==N) { /* Генерування N слів одночасно */
 int kk;
 for (kk=0;kk< N-M; kk++) {
 x[kk] = x[kk+M] ^ (x[kk] >> 1) ^ mag01[x[kk] % 2];
 }
 for (; kk< N-1; kk++) {
 x[kk] = x[kk+(M-N)] ^ (x[kk] >> 1) ^ mag01[x[kk] % 2];
 }
 x[N-1] = x[M-1] ^ (x[N-1] >> 1) ^ mag01[x[N-1] % 2];
 k=0;
 }
 y = x[k];
 y ^= (y << 7) & 0x2b5b2500; /* s і b, чарівні вектори*/
 y ^= (y << 15) & 0xdb8b0000; /* t і c, чарівні вектори */
 y &= 0xffffffff; /* вилучіть цей рядок, якщо довжина слова = 32 */
/*  Наступний рядок був доданий в версії 1996 р. для зменшення кореляції. */
 y ^= (y >> 16); /* Додати для версії 1994 р. */
 k++;
 return( (double) y / (unsigned long) 0xffffffff);
}
/* main() виводить перші 50 згенерованих чисел */
main()
{ int j;
 for (j=0; j<50; j++) {
   printf("%5f ", genrand());
   if (j%8==7) printf("\n");
}
print("\n");
}
```

Існують й інші конгруентні методи генерування випадкових чисел, серед яких слід відзначити адитивний.

Найпростіший генератор, послідовність якого x_{i+1} залежить більше ніж від одного з попередніх значень, — це генератор, що використовує числа Фібоначчі:

$$x_{i+1} \equiv (x_i + x_{i-1}) \bmod m.$$

Цей генератор широко використовувався в 50-ті роки XX століття, але, як показали подальші дослідження, статистичні властивості його досить низькі. Більш складні методи генерування випадкових чисел можна знайти в праці [23].

Розглянуті в цьому розділі методи генерування випадкових чисел не стосуються дослідників, які використовують мови або пакети моделювання. Ці засоби містять вбудовані генератори випадкових чисел, якість яких залежить від розробників цих програмних засобів. На жаль, є ще багато пакетів моделювання, в яких застосовуються генератори досить низької якості [18]. Що стосується мов програмування загального призначення, то засоби генерування випадкових чисел, вбудовані в них, взагалі не задовольняють будь-яким статистичним критеріям. Тому їх не слід використовувати під час проведення відповідальних досліджень.

4.3. Перевірка послідовностей випадкових чисел

Статистичні властивості всіх послідовностей випадкових чисел, які будуть використовуватись під час проведення досліджень, потрібно ретельно перевіряти. Для цього застосовують емпіричні та теоретичні критерії.

Емпіричні критерії – це звичайні тести, в яких під час обчислення статистичних даних використовують вибіркові значення r_i, що виробляються генератором.

Теоретичні критерії не є тестами в тому розумінні, в якому вони передбачаються в математичній статистиці. У разі їх використання не потрібна послідовність випадкових значень r_i. Глобальна оцінка властивостей генератора формується на основі числових значень його параметрів. Теоретичні критерії визначають характеристики послідовності за допомогою методів, які ґрунтуються на рекурентних правилах створення послідовності.

Відомо одинадцять емпіричних критеріїв [23], що застосовуються для перевірки статистичних властивостей послідовностей дійсних чисел r_i, $i = 1, 2, ...$, які вважаються незалежними та рівномірно розподіленими в інтервалі [0, 1].

Для оцінювання наближеності отриманого розподілу до рівномірного застосовують чотири типи тестів:

- *частотний* – з використанням або критерію Колмогорова–Смирнова, або критерію χ^2;
- *автокореляційний* – з вимірюванням кореляції між x_n і x_{n+k}, де k – зсув по послідовності ($k = 1, 2, 3, ...$);
- *серіальний* – з фіксацією частоти появи всіх можливих комбінацій чисел (по 2, по 3, по 4 рази) і виконанням оцінювання за критерієм χ^2;
- *циклічний* – з перевіркою кількості циклів більше і менше деякої константи, за яку береться значення математичного сподівання із підрахунком істинного числа циклів різної довжини, що порівнюється за критерієм χ^2 з очікуваним числом циклів.

Серед інших критеріїв важливу роль відіграє *спектральний критерій*, який застосовується для перевірки конгруентних генераторів випадкових чисел. Вважається, що цей тест найбільш потужний. Спектральний тест використовують

для перевірки гіпотези про рівність сумісних розподілів t послідовних елементів випадкової послідовності. Якщо задано послідовність $\{r_i\}$ з періодом m, то для перевірки за цим тестом необхідно проаналізувати множину всіх m точок

$$\{r_i, r_{i+2}, \ldots, r_{i+t-1}\}, \quad 0 \le i < m.$$

у t-вимірному просторі.

4.4. Моделювання випадкових подій та дискретних величин

У разі дослідження складних систем методом статистичних випробувань необхідно мати можливість отримувати за допомогою комп'ютера вибіркові значення випадкових величин, які мають різні закони розподілу. Випадкові величини зазвичай моделюють за допомогою перетворення одного або кількох незалежних значень випадкової величини R, рівномірно розподіленої в інтервалі [0, 1], що позначаються як r_i, i =1, 2, 3, ... ($r_i \in [0, 1]$). Значення r_i генерують, як звичайно, за допомогою програмних генераторів випадкових чисел.

4.4.1. Незалежні випадкові події

Припустимо, що ймовірність настання деякої елементарної випадкової події A в одному випробуванні дорівнює $P(A) = p$. Вважається, що умови проведення кожного випробування однакові і його можна повторити нескінченну кількість разів. Якщо r_i – це значення рівномірно розподіленої в інтервалі [0, 1] величини, то можна стверджувати, що за умови $r_i \le p$ (рис. 4.4) настане подія A, а якщо $r_i > p$, то відбудеться подія $\overline{A}$.

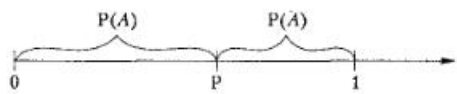


Рис. 4.4. Моделювання настання випадкових подій

Дійсно, якщо $f(r)$ – функція щільності рівномірно розподіленої випадкової величини r, то

$$P(r < p) = \int_0^P f(r)\,dr = p = P(A).$$

Ця модель добре описує такі події, як обслуговування вимоги в пристрої СМО, що може бути вільним або зайнятим, успішну або ні спробу виконання деякого завдання, влучення або ні в ціль, розгалуження потоків інформації у двох і більше напрямках. У деяких мовах для моделювання випадкової події використовується спеціальний блок (наприклад, у мові GPSS – блок TRANSFER, який працює в статистичному режимі [68]).

4.4.2. Група несумісних подій

Нехай є група несумісних подій $A_1, A_2, ..., A_k$, настання яких необхідно дослідити. Відомі ймовірності настання цих подій $p_1 = P(A_1)$, $p_2 = P(A_2)$, $p_3 = P(A_3)$, ..., $p_k = P(A_k)$. Якщо події несумісні, то $\sum_{i=1}^{k} p_i = 1$. Припустимо, що $p_0 = 0$. На відрізку [0, 1] числової осі відкладемо значення цих імовірностей (рис. 4.5).

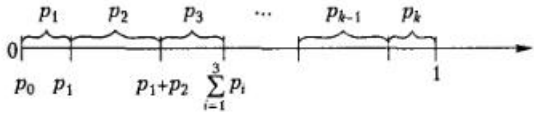


Рис. 4.5. Моделювання групи несумісних подій

Якщо отримане від генератора випадкових чисел значення r_i потрапляє в інтервал від $\sum_{k=0}^{i-1} p_k$ до $\sum_{k=0}^{i} p_k$, вважаємо, що відбулася подія A_i. Таку процедуру називають визначенням результату випробування за жеребом. Вона ґрунтується на формулі

$$P\left(\sum_{k=0}^{i-1} p_k < r_i \le \sum_{k=0}^{i} p_k\right) = p_i = P(A_i),$$

де $p_0 = 0$.

Ця модель часто використовується в теорії прийняття рішень і добре відтворює процеси вибору однієї з багатьох альтернатив у комп'ютерних іграх, розгалуження потоків інформації у вузлах мережі в кількох напрямках, вибір одного з багатьох пристроїв для обслуговування в СМО і т. ін.

4.4.3. Умовна подія

Умовна подія A – це подія, яка відбувається з імовірністю $P(A/B)$ тільки за умови, що настала подія B (рис. 4.6). У цьому разі має бути задана ймовірність $P(B)$ настання події B. Моделювання настання умовної події A провадиться таким чином. Спочатку випадкове число r_1, отримане від генератора випадкових чисел, використовується для моделювання настання події B. Подія B настає в тому випадку, якщо справджується нерівність $r_1 \le P(B)$. Настання події A моделюється за допомогою числа r_2. Для цього перевіряється умова $r_2 \le P(A)$, за виконання якої приймається рішення, що подія A відбулася. Якщо ж подія B не відбулася (тобто настає подія $\overline{B}$), то настання події A моделювати не потрібно. Таким чином, можна скоротити загальну кількість випробувань.

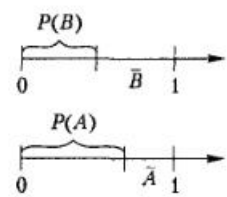


Рис. 4.6. Моделювання настання умовної події

4.4.4. Випадкова дискретна величина

Одне з основних понять теорії ймовірностей – дискретна випадкова величина X, яка набуває конкретних значень x_i з імовірністю p_i. Ці випадкові величини називають цілочисловими. Якщо можливі значення випадкової величини становлять скінченну послідовність, то розподіл імовірностей випадкової величини визначають, задаючи значення $x_1, x_2, ..., x_n$ і відповідних їм імовірностей $p_1, p_2, ..., p_n$. Моделювання випадкової дискретної величини виконується аналогічно моделюванню групи несумісних подій, тобто випадкову величину X подають як повну групу подій:

$$A_1 = (X = x_1),\ A_2 = (X = x_2),\ ...,\ A_n = (X = x_n).$$

Для моделювання дискретної випадкової величини X зручно використовувати дискретну кумулятивну функцію. Для цього аналізують можливі значення випадкової величини X і будують гістограму розподілу можливих значень.

Побудову і використання кумулятивної функції розглянемо на прикладі моделювання процесу введення даних під час роботи текстового терміналу. В табл. 4.1 наведено результати, які відображають результати спостереження за об'ємом інформації, яка вводиться з терміналу під час обробки одного повідомлення.

Таблиця 4.1. Результати спостереження за об'ємом введеної з терміналу інформації

Кількість символів	Розподіл (частка повідомлень зазначеної довжини)	Кумулятивний розподіл (частка повідомлень зазначеної або меншої довжини)
Менше 6	Відсутній	Відсутній
6–10	0,390	0,390
11–15	0,214	0,604
16–20	0,186	0,790
21–25	0,140	0,930
26–30	0,070	1,000
Більше 30	Відсутній	1,000

На рис. 4.7 і 4.8 зображено відповідно гістограму та кумулятивну функцію розподілу наведених у табл. 4.1 даних.

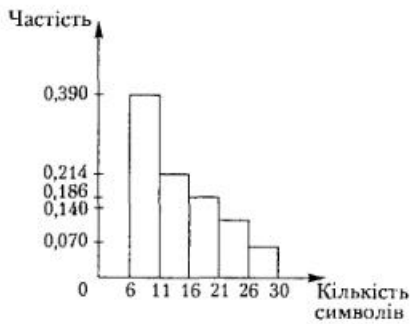


Рис. 4.7. Гістограма розподілу довжини повідомлень

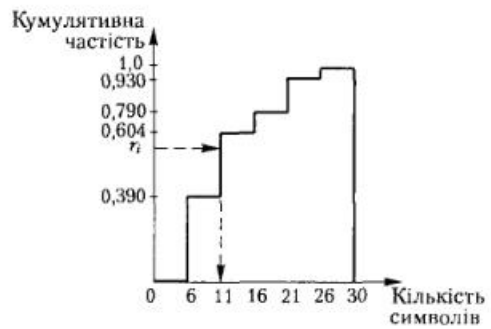


Рис. 4.8. Кумулятивна функція розподілу довжини повідомлень

Слід звернути увагу, що висота кумулятивної функції за заданих значень кількості символів дорівнює сумі значень, наведених на рис. 4.7. Для того щоб під час імітаційного моделювання роботи терміналу відтворити кількість символів, які вводяться з клавіатури, необхідно згенерувати випадкове число з діапазону від 0 до 1 (значення по вертикальній осі), а потім на горизонтальній осі визначити кількість уведених символів, які відповідають цьому числу. Наприклад, якщо випадкове число дорівнює 0,578, (див. рис. 4.8), то кількість символів, уведених з терміналу, можна прийняти таким, що дорівнює 11. Цей підхід ілюструє метод оберненої функції, згідно з яким спочатку генерується випадкове рівномірно розподілене число r_i, що задає значення кумулятивної функції розподілу, за яким потім визначається значення аргументу функції $x_i = F^{-1}(r_i)$, $i = 1, 2, ..., n$, де F^{-1} – обернена до F функція.

На практиці часто застосовують дискретні випадкові величини, що набувають лише невід'ємних значень $j = 0, 1, 2, ..., k, ..., n$ з імовірностями $p_0, p_1, p_2, ..., p_k, ..., p_n$, тобто функція розподілу дискретної величини x має вигляд

$$F(x) = \sum_{j=0}^{k} p(j).$$

У цьому випадку обернену функцію можна записати як

$$x_k = j \quad \text{для} \quad F(j-1) < r_k < F(j), \tag{4.3}$$

де згідно з умовою $F(-1) = 0$.

4.4.5. Геометричний розподіл

Для моделювання випадкової величини X з геометричним розподілом необхідно задати таблицю її значень та їх ймовірність (табл. 4.2).

Таблиця 4.2. Значення ймовірності геометричного розподілу випадкової величини

Значення X	0	1	...	n
Імовірність	p	$(1-p)\,p$		$(1-p)^n p$

Прикладом випадкової величини з таким розподілом може бути загальна кількість випробувань, які потрібно провести до першого успішного випробування, наприклад кількість пострілів, які потрібно виконати до першого влучення в ціль.

Загалом, ймовірність того, що випадкова величина приймає значення k, визначається за формулою

$$p_k = p(1-p)^k, \quad k = 0, 1, ..., n.$$

Для моделювання випадкової величини з геометричним розподілом можна скористатися табл. 4.2 або методом оберненої функції (4.3), але за великих n такі підходи потребують багато комп'ютерного часу. З цієї причини для отримання значення випадкової величини з геометричним розподілом використовують таку формулу [15]:

$$x_i = \left\lfloor \frac{\ln r_i}{\ln (1-p)} \right\rfloor, \quad 0 < p < 1.$$

У вищенаведеному виразі дужки $\lfloor\ \rfloor$ означають цілу частину виразу. Справді,

$$P(X = x) = P[k \le \ln r/\ln(1-p) < k+1] =$$
$$= P[-k \ln(1-p) \le -\ln r < -(k+1)\ln(1-p)] =$$
$$= P[(1-p)^{k+1} \le r < (1-p)^k] = (1-p)^k - (1-p)^{k+1} = p(1-p)^k,$$

так як випадкова величина r розподілена рівномірно в інтервалі [0, 1].

4.4.6. Біноміальний розподіл

Біноміальний розподіл, або розподіл Бернуллі, – це розподіл дискретної випадкової величини, яка приймає два і тільки два значення: 1 – «true», або «істина», та 0 – «false», або «хибність». Цей розподіл показує ймовірність настання деякої події за n незалежних повторних випробувань, у кожному з яких подія настає з імовірністю p, тобто ймовірність s успішних наслідків у n випробуваннях

$$f(s) = \frac{n!}{s!\,(n-s)!} p^s (1-p)^{n-s}.$$

Функція розподілу ймовірності має такий вигляд:

$$F(k) = \sum_{s=0}^{k} \frac{n!}{s!\ (n-s)!} p^s (1-p)^{n-s}, \quad k = 0, 1, \ldots, n.$$

Залежно від значення n можна вибрати один із двох способів моделювання випадкової величини з біноміальним розподілом. За невеликих n значення випадкової біноміально розподіленої величини визначається як кількість чисел у послідовності $\{r_i\}$ з n чисел, які не перевищують значення p. Припустимо, що потрібно отримати випадкову величину, яка належить біноміальному розподілу з параметрами $n = 7$ і $p = 0{,}3$. Для цього спочатку генеруємо послідовність із семи значень r_i: 0,0234; 0,1234; 0,7459; 0,0341; 0,8451; 0,1905; 0,5302, а потім рахуємо ті з них, які менші ніж p. У даному випадку в послідовності тільки чотири значення менші, ніж 0,3. Таким чином, значення випадкової величини, розподіленої за біноміальним законом, дорівнює 4.

За великих значень n і малих p можна діяти таким чином. Генеруємо рівномірно розподілені випадкові числа r_i доти, доки не виконається умова

$$r_i \le \sum_{j=0}^{n} u_j, \tag{4.4}$$

де u_0 та u_{j+1} задаються виразами

$$u_0 = (1-p)^n \quad \text{і} \quad u_{j+1} = u_j \frac{n-1}{j+1} \frac{p}{1-p}.$$

Значення випадкової величини з біноміальним розподілом дорівнює кількості випробувань n, які необхідно провести, доки не буде справджуватись умова (4.4).

4.4.7. Розподіл Пуассона

Випадкову величину з розподілом Пуассона можна отримати, якщо припустити, що кількість незалежних випробувань n у біноміальному розподілі прямує до нескінченності, а ймовірність успішного випробування p – до нуля, причому добуток np є незмінним і дорівнює λ. Функція щільності розподілу Пуассона задається виразом

$$f(s) = \frac{\lambda^s}{s!} e^{-\lambda}.$$

Таким чином, розподіл Пуассона є граничним випадком біноміального та описує випадкові події, які мають місце дуже рідко. На практиці згідно з біноміальним законом розподілені кількість дефектів у готовому виробі та кількість аварій на транспорті за деякий тривалий проміжок часу, кількість дзвінків у телефонній мережі за одиницю часу та ін.

Щоб отримати випадкову величину s з розподілом Пуассона, генеруємо послідовність рівномірно розподілених випадкових чисел r_i і знаходимо їх добуток, перевіряючи нерівність

$$\prod_{i=1}^{n} r_i < e^{-\lambda}. \qquad (4.5)$$

У разі виконання умови (4.5) число $n - 1$ і є випадковою величиною, що належить сукупності, розподіленій за законом Пуассона з математичним сподіванням λ. Якщо умові (4.5) відповідає перше із чисел r_i, то значення випадкової величини s дорівнює 0.

4.5. Моделювання неперервних випадкових величин

Існує кілька методів моделювання значень неперервних випадкових величин з довільним законом розподілу на основі випадкових чисел, рівномірно розподілених у інтервалі [0, 1]: метод оберненої функції, метод відсіювання, наближені методи тощо.

4.5.1. Метод оберненої функції

Розглянемо метод моделювання випадкової величини, яка має функцію щільності ймовірностей $f(x)$ і монотонно зростаючу функцію розподілу $F(x)$ (рис. 4.9). Суть методу така. За допомогою генератора випадкових чисел генеруємо значення випадкової величини r_i, якому відповідає точка на осі ординат. Значення випадкової величини x_i з функцією розподілу $F(x)$ можемо одержати з рівняння $F(x_i) = r_i$.

Дійсно, якщо на осі ординат відкласти значення r_i випадкової величини, розподіленої рівномірно в інтервалі [0, 1], і на осі абсцис знайти значення x_i випадкової

величини (рис. 4.9), при якому $F(x_i) = r_i$, то випадкова величина $X = F^{-1}(r)$ буде мати функцію розподілу $F(x)$. За визначенням функція розподілу $F(x)$ випадкової величини X дорівнює ймовірності $P(X < x)$:

$$P(X < x) = P(r < F(X)) = \int_0^{F(x)} f(r)\,dr = F(x).$$

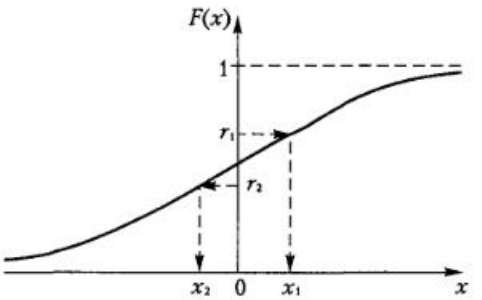


Рис. 4.9. Використання методу оберненої функції для генерування неперервної випадкової величини

Таким чином, послідовність випадкових чисел r_1, r_2, r_3, ... перетворюється на послідовність x_1, x_2, x_3, ..., яка має задану функцію щільності розподілу $f(x)$. Звідси випливає загальний алгоритм моделювання випадкових неперервних величин, що мають задану функцію розподілу ймовірностей:

- генерується випадкове число $r_i \in [0, 1]$;
- обчислюється випадкове число x_i, яке є розв'язком рівняння

$$r_i = \int_{-\infty}^{x_i} f(x)\,dx.$$

Приклади застосування методу наведені нижче.

4.5.2. Рівномірний розподіл

У загальному випадку випадкова величина X є рівномірно розподіленою на відрізку $[a, b]$, якщо її щільність розподілу ймовірностей має вигляд

$$f(x) = \begin{cases} 0, & x < a, \\ \dfrac{1}{b-a}, & a \le x \le b, \\ 0, & x > b. \end{cases}$$

Функцію розподілу ймовірностей можна знайти як

$$F(x) = \int_a^x \frac{1}{b-a}\,dx = \frac{x-a}{b-a},$$

тобто

$$F(x) = \begin{cases} 0, & x < a, \\ \dfrac{1}{b-a}, & a \le x \le b, \\ 1, & x > b. \end{cases}$$

Графіки функцій щільності $f(x)$ та ймовірності $F(x)$ зображено на рис. 4.10.

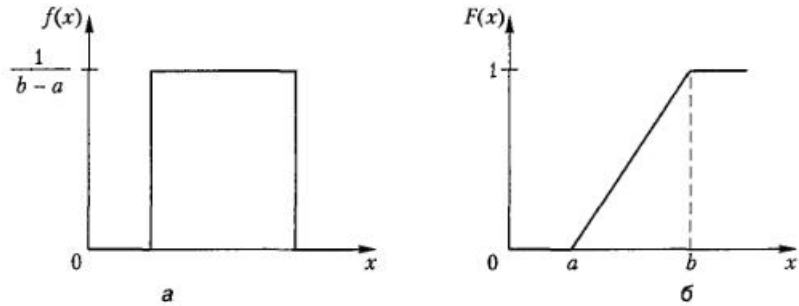


Рис. 4.10. Функції щільності (*а*) і розподілу (*б*) рівномірно розподіленої випадкової величини

Математичне сподівання та дисперсія випадкової величини X визначаються як

$$\mathrm{M}(X) = \frac{a+b}{2}, \quad \mathrm{D}(X) = \frac{b-a}{12}.$$

Для моделювання випадкової рівномірно розподіленої на відрізку $[a, b]$ величини можна скористатись методом оберненої функції. Обчислимо функцію розподілу випадкової величини та прирівняємо її до значення r_i:

$$r_i = \int_a^{x_i} \frac{dx}{b-a} = \frac{x_i - a}{b-a}.$$

Звідси знаходимо значення випадкової величини з функцією розподілу $f(x)$:

$$x_i = (b - a)\, r_i + a.$$

Цю формулу також можна отримати, якщо виконати лінійне перетворення інтервалу $[0, 1]$ у відрізок $[a, b]$. Для цього потрібно змінити масштаб функції рівномірного розподілу, помноживши її на $(b - a)$, а потім змістити її на величину a.

Функція рівномірного розподілу широко застосовується для моделювання випадкових величин, для яких функція розподілу невідома, а відоме лише її середнє значення. У такому випадку припускають, що відомими є середнє значення випадкової величини та деяке розсіювання $\pm\Delta$ її значень відносно середнього. Це дає змогу стверджувати, що дана випадкова величина має рівномірний розподіл. У мові GPSS такий розподіл часто використовується в блоках ADVANCE для моделювання затримки проходження інформації або під час генерування потоків транзактів у блоках GENERATE. Наприклад, щоб згенерувати потік транзактів, які надходять у модель кожні 5 ± 2 хв, використовується блок GENERATE 5,2.

Прикладами реальних задач, в яких виникає необхідність моделювання рівномірно розподілених випадкових величин, можуть бути аналіз помилок округлення під час проведення числових розрахунків (точність задається як кількість десяткових знаків), час переміщення головок у магнітних накопичувачів (мінімальний та максимальний час), відхилення від розкладу руху транспортних засобів (наприклад, метро).

4.5.3. Експоненціальний розподіл

Експоненціальний закон розподілу набув широкого використання в теорії надійності складних систем. Функція щільності експоненціального розподілу випадкової величини має вигляд

$$f(x) = \lambda e^{-\lambda x}.$$

Для її моделювання скористаємося методом оберненої функції. Маємо

$$r_i = \int_0^{x_i} f(x)\,dx = \int_0^{x_i} \lambda e^{-\lambda x}\,dx = 1 - e^{-\lambda x_i}. \qquad (4.6)$$

З виразу (4.6) знаходимо значення x_i:

$$x_i = -\frac{1}{\lambda}\ln(1 - r_i).$$

Можна показати, що випадкові величини $(1 - r_i)$ мають такий самий розподіл, що і величини r_i. Тоді, замінивши $1 - r_i$ на r_i, отримаємо

$$x_i = -\frac{1}{\lambda}\ln r_i.$$

Випадкові величини з експоненціальним розподілом широко застосовуються в задачах моделювання та аналізу СМО, наприклад під час моделювання процесів виходу з ладу та ремонту обладнання, які виникають у складних системах, у разі визначення інтервалів часу між послідовними викликами абонентів у телефонній мережі або замовлень від незалежних клієнтів у будь-якій мережі обслуговування (швидка допомога, служби ремонту, виклик таксі і т. ін.)

Покажемо ще один підхід до моделювання випадкової величини, розподіленої за експоненціальним законом, з використанням методу оберненої функції, який прийнятий у мові GPSS [68]. Цей підхід передбачає заміну функції розподілу ймовірностей кусково-лінійною апроксимуючою функцією.

Розглянемо найпростіший випадок, коли $\lambda = 1$. Апроксимуємо функцію експоненціального розподілу лінійними відрізками таким чином, щоб кожний відрізок можна було використовувати для моделювання за допомогою методу оберненої функції. У мові GPSS функція розподілу апроксимована 23 відрізками. Точки апроксимації x_i та значення функції $F(x_i)$ у цих точках наведені в табл. 4.3.

Таблиця 4.3. Вузли та значення апроксимованої функції

x_i	$F(x_i)$	x_i	$F(x_i)$	x_i	$F(x_i)$	x_i	$F(x_i)$	x_i	$F(x_i)$	x_i	$F(x_i)$
0	0	0,4	0,509	0,75	1,38	2,3	0,9	3,2	0,96	5,3	0,995
0,1	0,1	0,5	0,69	0,8	1,6	2,52	0,92	3,5	0,97	6,2	0,998
0,2	0,222	0,6	0,915	0,84	1,83	2,81	0,94	3,9	0,98	7	0,999
0,3	0,355	0,7	1,2	0,88	2,12	2,99	0,95	4,6	0,99	8	0,9998

На рис 4.11 зображено лінійну апроксимацію експоненціальної функції розподілу $F(x)$ з параметром $\lambda = 1$, а на рис. 4.12 – функцію, обернену до неї. Перша функція відображає задані в табл. 4.2 значення, а друга використовується під час моделювання випадкових величин з експоненціальним розподілом.

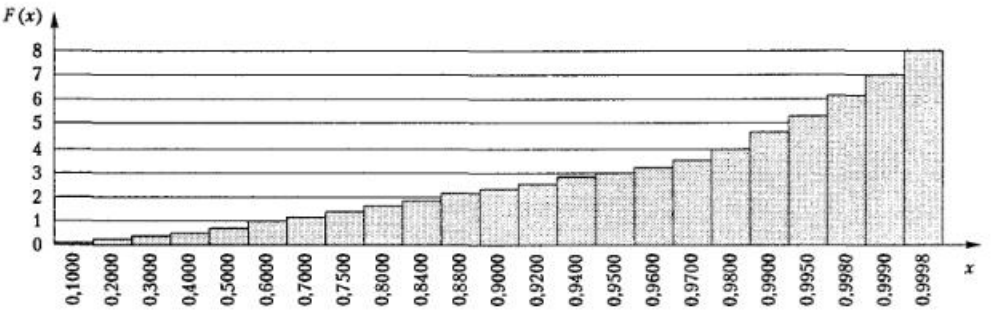


Рис. 4.11. Апроксимація експоненціальної функції розподілу

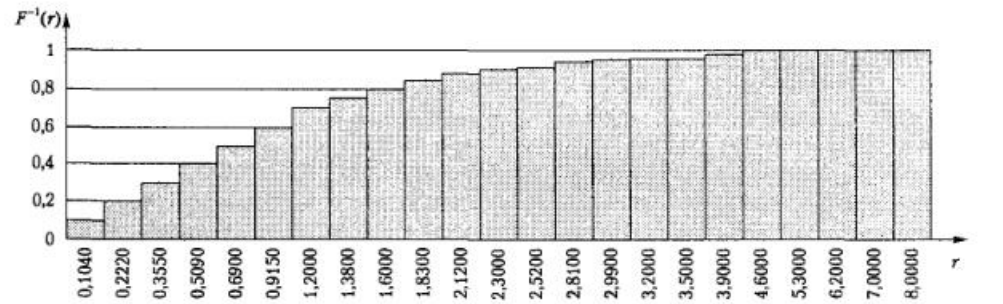


Рис. 4.12. Обернена функція до апроксимованої

За необхідності моделювання випадкової величини X, розподіленої за експоненціальним законом з математичним сподіванням $\lambda_x \neq 1$, діють таким чином:

- генерують значення випадкової величини r_i, яке використовують як аргумент оберненої функції (рис. 4.12) експоненціального розподілу з параметром $\lambda = 1$, і знаходять значення функції $F(r)$ (для підвищення точності оцінювання параметрів моделювання функцію розподілу $F(r)$ інтерполюють лінійними відрізками);
- знаходять добуток отриманого значення $F(r)$ на $1/\lambda_x$.

Наприклад, для моделювання часу затримки транзактів, що має експоненціальний закон розподілу з параметром $\lambda_x = 0{,}01$, у мові GPSS використовують блок

```
ADVANCE 100,FN$XPDIS
```

У цьому операторі функція XPDIS (див. табл. 4.3) задає експоненціальний розподіл з інтенсивністю $\lambda = 1$:

```
XPDIS FUNCTION RN1,C24; exponential distribution function
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/
.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/
.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/
.96,3.2.97,3.5/.98,3.9/.99,4.6/.995,5.3/
.998,6.2/.999,7/.9998,8
```

4.5.4. Пуассонівський потік

Розглянемо моделювання пуассонівського потоку з інтенсивністю λ, основна властивість якого полягає в тому, що ймовірність надходження k вимог протягом інтервалу довжиною t становить

$$p_k(t) = \frac{e^{-\lambda t}(\lambda t)^k}{k!}, \quad k = 1, 2, \ldots$$

Для пуассонівського потоку інтервали часу між надходженням двох сусідніх вимог мають експоненціальний закон розподілу (див. розділ 2.1). Тому для його моделювання достатньо отримати ряд чисел з таким розподілом. Це можна реалізувати за допомогою методу оберненої функції, якщо ряд випадкових чисел r_i, рівномірно розподілених у інтервалі [0, 1], перетворити згідно з функцією, оберненою до експоненціальної функції розподілу

$$t_j = F^{-1}(x) = -\overline{T} \ln(r_j),$$

де t_j – j-й проміжок часу між надходженнями двох сусідніх вимог; $\overline{T} = 1/\lambda$ – середнє значення проміжку часу між надходженнями двох сусідніх вимог; r_j – j-е число в послідовності випадкових чисел з рівномірним розподілом у інтервалі [0, 1].

У мові GPSS для моделювання пуассонівського потоку вимог з $\overline{T} = 2$ год (одиниця часу в моделі дорівнює 1 хв) використовується блок GENERATE 120,FN$XPDIS.

4.5.5. Нормальний розподіл

Випадкова величина X має нормальний розподіл (розподіл Гаусса), якщо її щільність розподілу ймовірностей описується виразом

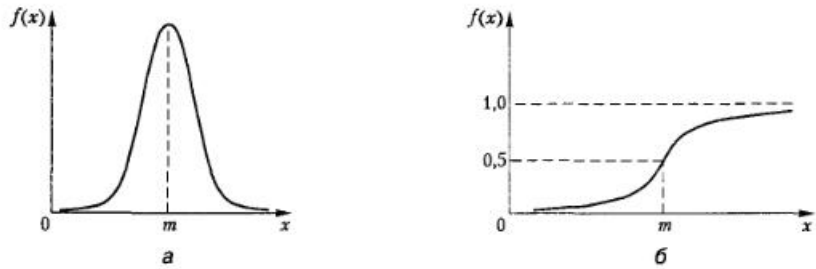
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-m)^2}{2\sigma^2}},$$

де m – математичне сподівання, а σ – середньоквадратичне відхилення.

Функція розподілу нормально розподіленої величини X має вигляд

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-m)^2}{2\sigma^2}}\, dx.$$

Графіки функцій щільності ймовірностей $f(x)$ і розподілу $F(x)$ зображено на рис. 4.13.



а б

Рис. 4.13. Графіки функції щільності ймовірностей (*а*) та розподілу (*б*) випадкової величини

Для моделювання випадкової величини з нормальним законом розподілу безпосередньо скористатися методом оберненої функції не можна, оскільки неможливо аналітично виконати перетворення виду $X = F^{-1}(r)$. Тому для моделювання слід скористатися методом згорток.

Метод згорток базується на центральній граничній теоремі – одному із найбільш видатних результатів теорії ймовірностей: за широких припущень відносно розподілів суми великої кількості взаємно незалежних малих випадкових величин має місце розподіл, який близький до нормального. Метод згорток передбачає зображення випадкової величини як суми незалежних однаково розподілених випадкових величин зі скінченними математичним сподіванням і дисперсією.

Центральна гранична теорема формулюється таким чином.

Якщо $X_1, ..., X_n$ – послідовність незалежних випадкових величин із скінченним математичним сподіванням $M[X_i] = a, i = \overline{1,n}$ і дисперсією $D[X_i] = \sigma^2, i = \overline{1,n}$, то у разі необмеженого збільшення значення n функція розподілу випадкової величини

$$\overline{X}^*(n) = \frac{1/n\,(X_1 + ... + X_n) - a}{\sigma/\sqrt{n}} = \frac{(\overline{X}(n) - a)\sqrt{n}}{\sigma}$$

наближається до функції розподілу стандартного нормального закону $\Phi(z)$ при всіх значеннях аргументу, тобто

$$F_{\overline{X}^*(n)} \xrightarrow[n\to\infty]{} \Phi(z),$$

де

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-u^2/2}\, du, \quad u = \frac{x-m}{\sigma}.$$

Функція $\Phi(z)$ називається функцією Лапласа, для якої є детальні таблиці.

Найпростіший метод отримання значення випадкової величини, що має заданий нормальний розподіл, передбачає виконання таких кроків. Спочатку формують послідовність r_i $(i = \overline{1,n})$ незалежних, рівномірно розподілених у інтервалі [0, 1]

величин і обчислюють суму – $Z = \sum_{i=1}^{12} r_i - 6$. Величина $n = 12$ є хорошим наближенням до нормально розподіленої випадкової величини з нульовим математичним сподіванням $m_z = 0$ і одиничним середньоквадратичним відхиленням $\sigma_z = 1$. Нормальний розподіл з параметрами $m_z = 0$ та $\sigma_z = 1$ називається стандартним.

Перейти від випадкової величини Z з нульовим математичним сподіванням і одиничним середньоквадратичним відхиленням до випадкової величини X, яка має математичне сподівання m_x і середньоквадратичне відхилення σ_x, дає змогу виконати лінійне перетворення

$$X = \sigma_x Z + m_x . \tag{4.7}$$

У системі моделювання GPSS [68] для моделювання випадкової величини з нормальним розподілом прийнято підхід, який базується на методі оберненої функції. За такого підходу функція нормального розподілу випадкової величини Z з параметрами $m_z = 0$ і $\sigma_z = 1$ наближається кусково-лінійною функцією. Як відзначили розробники інтерпретатора GPSS/PC [68], для цього достатньо 24 відрізки. У табл. 4.4 занесено відповідні значення аргументу z_i і функції $\Phi(z)$.

Таблиця 4.4. Вузли апроксимації і значення функції нормального розподілу

z_i	$\Phi(z)$	z_i	$\Phi(z)$	z_i	$\Phi(z)$	z_i	$\Phi(z)$	z_i	$\Phi(z)$
– 5	0	– 1,5	0,06681	– 0,4	0,34458	0,6	0,72575	2	0,97725
– 4	0,00003	– 1,2	0,11507	– 0,2	0,42074	0,8	0,78814	2,5	0,99379
– 3	0,00135	– 1	0,15866	0	0,5	1	0,84134	3	0,99865
– 2,5	0,00621	– 0,8	0,21186	0,2	0,57964	1,2	0,88493	4	0,99997
– 2	0,02275	– 0,6	0,27425	0,4	0,65542	1,5	0,93319	5	1

Для того щоб одержати нормально розподілену випадкову величину з математичним сподіванням $m_x \neq 0$ і середньоквадратичним відхиленням $\sigma_x \neq 1$, необхідно виконати обчислення за формулою (4.7). На рис. 4.14 зображено графік функції, отриманої в результаті апроксимації функції нормального розподілу $\Phi(z)$, а на рис. 4.15 – графік функції $\Phi^{-1}(r)$, якою зручніше користуватися під час моделювання (як аргумент використовують значення r_i від генератора випадкових чисел і отримують значення функції).

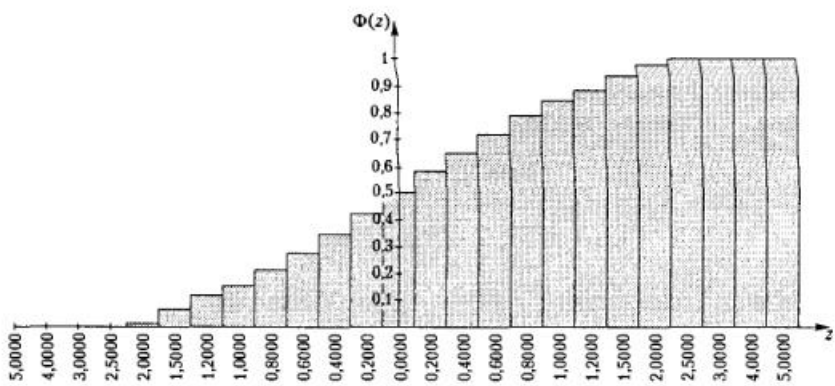


Рис. 4.14. Кусково-лінійна апроксимація функції нормального розподілу

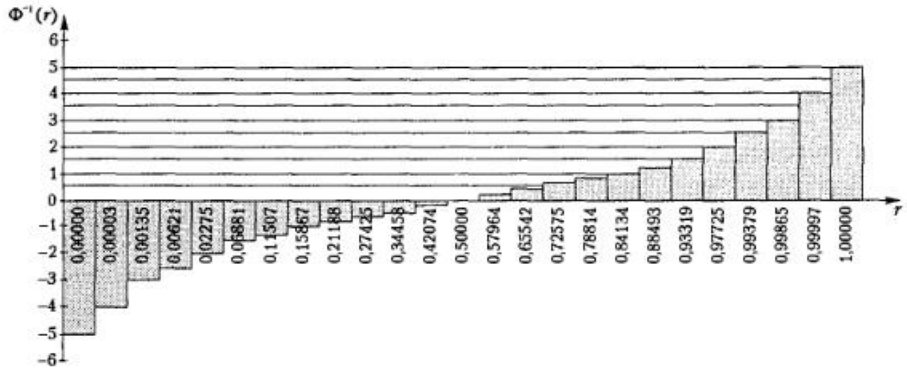


Рис. 4.15. Обернена функція до апроксимованої

У системі GPSS для моделювання нормально розподілених випадкових величин використовується функція NOR, яка апроксимує функцію стандартного нормального розподілу:

```
NOR FUNCTION RN1,C25
0,-5/.00003,-4/.00135,-3/.00621,-2.5/.02275,-2
.06681,-1.5/.11507,-1.2/.15866,-1/.21186,-.8/.27425,-.6
.34458,-.4/.42074,-.2/.5,0/.57926,.2/.65542,.4
.72575,.6/.78814,.8/.84134,1/.88493,1.2/.93319,1.5
.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
```

Наприклад, щоб отримати значення випадкової величини з параметрами $m_x = 60$ і $\sigma_x = 10$, потрібно задати змінну NORM1, яка викликає функцію NOR:

```
NORM1 FVARIABLE 60+10#FN$NOR
```

Якщо в моделі потрібно здійснити затримку транзактів на проміжок часу, що має нормальний розподіл (необхідно забезпечити невід'ємність значень, тобто потрібно, щоб виконувалась умова $m_x \geq 5\,\sigma_x$), використовують блок ADVANCE V$NORM1.

Недоліком розглянутих вище методів моделювання є те, що значення функції нормального розподілу, які лежать за межами $m_x \pm \sigma_x$, суттєво відрізняються від точних значень. Щоб зменшити загальну похибку моделювання, треба використовувати більш точні методи отримання значень функції нормального розподілу. Ці методи базуються на такій властивості. Якщо X_1 і X_2 є незалежними нормально розподіленими випадковими величинами з нульовим математичним сподіванням і одиничним середньоквадратичним відхиленням, то величина кута між віссю абсцис і вершиною випадкового вектора з координатами (x_1, x_2) має рівномірний розподіл і не залежить від довжини вектора $\left(\sqrt{x_1^2 + x_2^2}\right)$ (рис. 4.16).

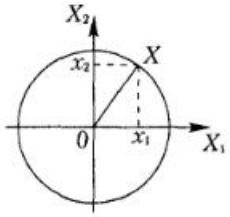


Рис. 4.16. Зображення вектора для моделювання нормального розподілу

Квадрат довжини вектора в цьому випадку має розподіл χ^2 з двома ступенями свободи і моделюється як окремий випадок показового розподілу з параметром $\lambda = 1/2$.

Існує два методи моделювання нормального розподілу, які використовують цю властивість:

Метод Бокса–Мюллера (Box–Muller). Генеруємо пару нормально розподілених чисел з $m_x = 0$ і $\sigma_x = 1$ за допомогою двох випадкових чисел r_1 і r_2:

$$x_1 = -2 \ln r_1 \cos(2\pi r_1); \quad x_2 = -2 \ln r_2 \cos(2\pi r_2).$$

Таким чином, отримуємо два числа x_1 і x_2 з нормальним розподілом.

Метод Марсальї–Брея (Marsaglia–Bray). Існує більш швидка модифікація цього методу. Генерують два випадкових числа r_1 і r_2, вважаючи, що $v_1 = -1 + 2r_1$, $v_2 = -1 + 2r_2$, обчислюють суму $s = v_1 + v_2$. Якщо $s \geq 1$, то повторюють процедуру, якщо $s < 1$, то одержують два нормально розподілених числа:

$$x_1 = v_1 \sqrt{\frac{-2 \ln s}{s}}, \quad x_2 = v_2 \sqrt{\frac{-2 \ln s}{s}}.$$

Щоб одержати за цим методом 100 пар нормально розподілених чисел, потрібно генерувати 127 пар випадкових чисел. Це простий та швидкий метод, у разі його застосування більша частина часу роботи алгоритму витрачається на обчислення логарифму.

4.5.6. Логарифмічно-нормальний розподіл

Логарифмічно-нормальний розподіл – це такий розподіл випадкової величини, в якої нормальний розподіл має натуральний логарифм її значень. Цей розподіл придатний для моделювання мультиплікативних процесів так само, як нормальний розподіл – для адитивних. Дійсно, використовуючи центральну граничну теорему, можна показати, що добуток незалежних додатних випадкових величин прямує до логарифмічно-нормального розподілу.

Логарифмічно-нормальна величина є результатом взаємодії великої кількості незалежних малих випадкових факторів. Внесок кожного фактора пропорційний уже досягнутому рівню досліджуваної величини, тобто характер впливу є мультиплікативним. Функція щільності логарифмічно-нормального розподілу має вигляд

$$f_\eta(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(\ln x - \ln a)^2}{2\sigma^2}}.$$

Щоб перевірити, чи мають емпіричні вибіркові дані логарифмічно-нормальний розподіл, потрібно обчислити логарифм від кожного елементу вибірки і перевірити її на нормальність, наприклад за допомогою критерію χ^2. Якщо трансформований набір даних має нормальний розподіл, то вхідні дані є логарифмічно-нормально розподіленими.

На відміну від нормального розподілу, значення випадкової величини x з логарифмічно-нормальним розподілом завжди додатне і використовується під час моделювання економічних, фізичних, біологічних систем багатьох типів. Вони добре описують процеси, в яких значення змінної, що спостерігається, є випадковою часткою значення попереднього спостереження.

Випадковими величинами з цим розподілом можуть бути тривалість безвідмовної роботи виробу в режимі спрацювання та старіння, розмір банківського вкладу, довжини слів певної мови та переданих повідомлень у мережі, розміри файлів, що зберігаються в комп'ютері.

Як і нормальний розподіл, логарифмічно-нормальний також визначається двома параметрами a і σ. Графіки функції щільності логарифмічно-нормального розподілу для різноманітних значень параметрів a і σ зображено на рис. 4.17.

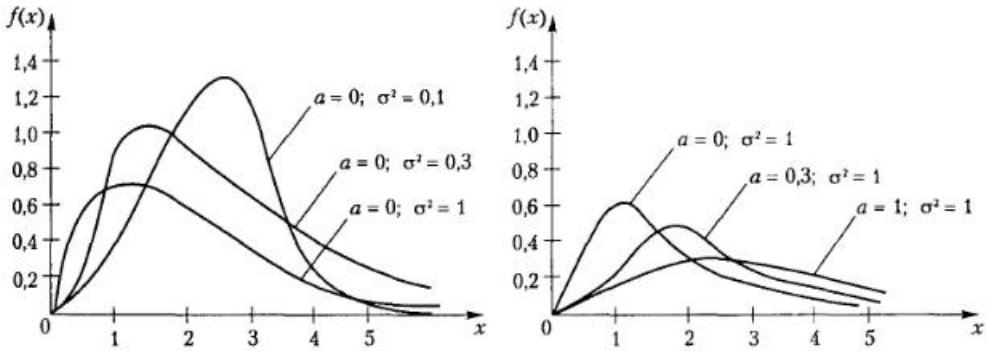


Рис. 4.17. Графіки функції щільності логарифмічно-нормального розподілу

Метод моделювання логарифмічно-нормального розподілу передбачає підставлення в рівняння $L = e^N$ значень з вибірки $N(m, \sigma^2)$, які мають нормальний розподіл з математичним сподіванням m і дисперсією σ^2, де L – логарифмічно-нормальний розподіл. Математичне сподівання для цього розподілу $M[x] = e^{m+\sigma^2/2}$ і дисперсія $D[x] = e^{2m+\sigma^2}(e^{\sigma^2} - 1)$.

4.5.7. Розподіл і потоки Ерланга

Випадкові величини з експоненціальним розподілом не завжди адекватно описують деякі реальні процеси та події, наприклад час обслуговування і моменти надходження вимог до СМО. Для більш точного моделювання таких процесів доцільніше використовувати гамма-розподілені випадкові величини або ті, що мають розподіл Ерланга. Розподіл Ерланга є результатом підсумовування взаємно незалежних і однаково розподілених експоненціальних випадкових величин і є окремим випадком гамма-розподілу.

Функція щільності розподілу Ерланга k-го порядку з інтенсивністю λ має такий вигляд:

$$f_\eta(x) = \frac{\lambda(\lambda x)^{k-1}}{(k-1)!} e^{-\lambda x}, \quad x \geq 0.$$

Математичне сподівання і дисперсія розподілу Ерланга визначаються як $M[x] = 1/k\lambda$, $D[x] = 1/k\lambda^2$. Графік функції щільності розподілу Ерланга зображено на рис. 4.18.

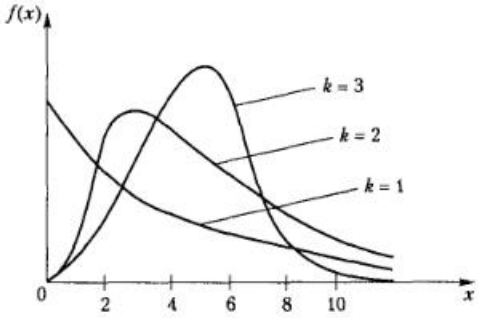


Рис. 4.18. Графік функції щільності розподілу Ерланга

Для моделювання розподілу Ерланга використовують метод згорток випадкових величин з експоненціальними функціями розподілу. Як зазначалось у розділі 2.1, для цього треба лише обчислити суму k експоненціально розподілених випадкових величин. Зі збільшенням k розподіл Ерланга наближається до нормального.

Мовою GPSS потік Ерланга другого порядку із середнім значенням часу надходження 180 можна задати таким чином:

```
  GENERATE,,,1
SDFG ADVANCE 90,FN$EXPDIS
  ADVANCE 90,FN$EXPDIS
  SPLIT 1,SDFG
  ...
  TERMINATE 1
```

Дамо пояснення до цієї програми. У початковий момент часу в моделі генерується один транзакт. Кожний блок ADVANCE імітує затримку транзакту на певний проміжок часу, який має експоненційальний закон розподілу. Блок SPLIT створює копію транзакту і направляє його до блока з міткою SDFG, транзакт-оригінал надходить до моделі і так далі.

Для дослідження властивостей розподілу Ерланга в системі GPSS можна скористатись таким кодом:

```
EXPDIS   FUNCTION RN1,C24
0,0/.100,.104/.200,.222/.300,.355/.400,.509
.500,.690/.600,.915/.700,1.200/.750,1.380
.800,1.600/.840,1.830/.880,2.120/.900,2.300
.920,2.520/.940,2.810/.950,2.990/.960,3.200
.970,3.500/.980,3.900/.990,4.600/.995,5.300
.998,6.200/.999,7/1,8
TPTABLE    X2,1,20,50 ;визначає таблицю
  GENERATE ...1
SDFG   ADVANCE 100,FN$EXPDIS
  ADVANCE   100,FN$EXPDIS
  ADVANCE   100,FN$EXPDIS
  SPLIT     1,SDFG  ;створює одну копію транзакту
  SAVEVALUE   2,C1  ;запам'ятовує час появи транзакту
```

```
SAVEVALUE   2-,X1   ;визначає інтервал часу між транзактами
SAVEVALUE   1,C1    ;запам'ятовує час появи попереднього
                    ;транзакту
TABULATE    TP      ;будує гістограму
TERMINATE   1
```

Оператор TABLE, блоки SPLIT, SAVEVALUE і TABULATE необхідні для збирання статистичних даних про інтервали надходження транзактів до моделі [63].

Результати моделювання в разі використання оператора START 100000000 наведено на рис. 4.19. Пропонується дослідити властивості розподілу Ерланга при різних значеннях k шляхом зміни кількості блоків ADVANCE у наведеній програмі.

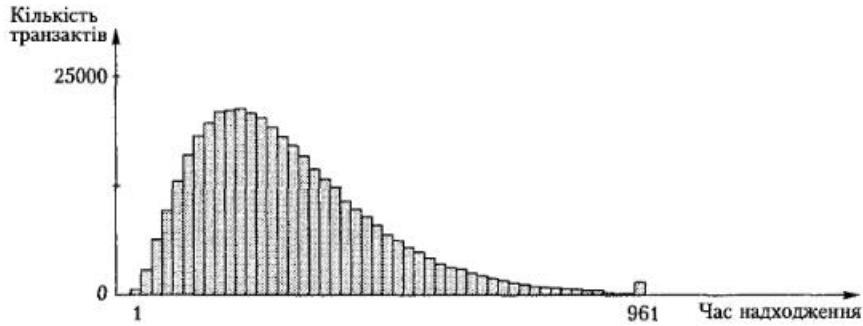


Рис. 4.19. Гістограма, отримана за результатами моделювання

4.5.8. Гамма-розподіл

Випадкова величина має гамма-розподіл з параметрами α та β, якщо її функція щільності має вигляд

$$f_\gamma(x) = \begin{cases} \dfrac{\beta^{-\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}}{\Gamma(\alpha)}, & 0 \le x < \infty; \\ 0, & x < 0, \end{cases} \tag{4.8}$$

де α – параметр форми розподілу, β – масштабний коефіцієнт, $\Gamma(\alpha)$ – гамма-функція або функція Ейлера, яка визначається як [18]

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x}\, dx, \quad z > 0.$$

Нагадаємо деякі властивості гамма-функції: $\Gamma(z+1) = z\Gamma(z)$ для $z > 0$; $\Gamma(k+1) = k!$ для від'ємних цілих k; $\Gamma(k+1/2) = \sqrt{\pi} \cdot 1 \cdot 3 \cdot 5 \ldots (2k-1)/2^k$ для додатних цілих k; $\Gamma(1/2) = \sqrt{\pi}$.

Математичне сподівання гамма-розподілу і дисперсія визначаються як $M_\gamma[x] = \alpha\beta$ та $D_\gamma[x] = \alpha\beta^2$. Вигляд функції гамма-розподілу значною мірою залежить від значень параметрів. Ця властивість дає змогу використовувати випадкові величини з цим розподілом для моделювання різноманітних фізичних та економічних

явищ і процесів. Графіки функції розподілу при різних значеннях параметрів α і β зображено на рис. 4.20.

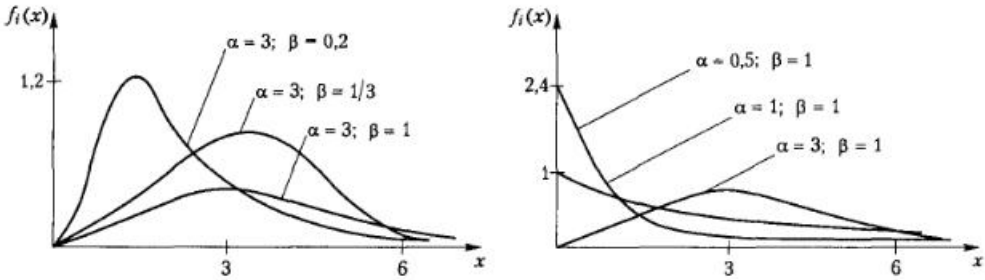


Рис. 4.20. Графіки функції щільності для гамма-розподілу

Гамма-розподіл є узагальненням розподілу Ерланга, коли кількість підсумованих експоненціальних величин не є цілим числом. Якщо α = 0,5 і β= 2, то випадкові величини з гамма-розподілом можна інтерпретувати також як суму квадратів нормально розподілених випадкових величин, тобто таких, які мають розподіл χ^2. Таким чином, розподіл χ^2, розподіл Ерланга та експоненційальний розподіл є окремими випадками гамма-розподілу.

Гамма-розподіл має важливу властивість. Сума будь-якої кількості незалежних гамма-розподілених випадкових величин m з однаковим значенням параметра β теж підпорядковується гамма-розподілу, але з параметрами $(\alpha_1 + \alpha_2 + ... + \alpha_m)$ і β.

Метод моделювання випадкової величини з гамма-розподілом залежить від значень параметрів α та β. Якщо α = 1 і β = 1, гамма-розподіл перетворюється в експоненціальний розподіл, і тому для отримання випадкової величини з гамма-розподілом можна використати відповідні методи моделювання. Якщо α має ціле значення можна перейти до моделювання розподілу Ерланга. Якщо α = 0,5, гамма-розподіл перетворюється на розподіл χ^2, тому для його моделювання досить підносити до квадрату вибіркові значення нормально розподілених випадкових величин.

Від випадкової величини X, яка має гамма-розподіл з будь-яким значенням параметра α і значення β = 1, досить легко можна перейти до випадкової величини X' з параметрами α і β > 1. Для цього використовується перетворення виду $X' = \beta X$. Причому ефективність і швидкодія методу зростає зі збільшенням значення α.

Існує багато методів моделювання значень гамма-розподіленої випадкової величини [18, 46, 67]. Основна проблема, яка виникає під час її моделювання, – це обчислення гамма-функції. Справа в тому, що цей інтеграл обчислити аналітично неможливо. Тому для його обчислення зазвичай використовують числові методи.

Щоб отримати значення гамма-функції $\Gamma(z)$ або $1/\Gamma(z)$, можна скористатися такою формулою[26, 59]:

$$\frac{1}{\Gamma(z)} \approx z(1+z)(1+a_1 z + a_2 z^2 + ... + a_{13} z^{13}), \quad z \in [-1, 1],$$

де

$$a_1 = -0{,}422784335092; \quad a_2 = -0{,}233093736365; \quad a_3 = 0{,}191091101162;$$
$$a_4 = -0{,}024552490887; \quad a_5 = -0{,}017645242118; \quad a_6 = 0{,}008023278113;$$

$a_7 = -0{,}000804341335$; $a_8 = -0{,}000360851496$; $a_9 = 0{,}000145624324$;
$a_{10} = -0{,}000017527917$; $a_{11} = -0{,}000002625721$; $a_{12} = 0{,}000001328554$;
$a_{13} = -0{,}00000018122$.

Обчислення гамма-функції для різних значень x ускладнюється тим, що вона залежить від трьох аргументів – x, α та β. Тому на практиці під час моделювання у формулі функції щільності (4.8) використовується неповна гамма-функція

$$\Gamma(\alpha) = \int_0^x t^{\alpha-1} e^{-t}\, dt, \tag{4.9}$$

для обчислення якої за умови, що $\alpha < 1$ можна скористатися таким виразом [59]:

$$\Gamma(\alpha) = x^{\alpha} \sum_{i=0}^{\infty} \frac{(-1)^i x^i}{i!\,(\alpha + i)}.$$

Для $\alpha > 1$ інтеграл (4.9) можна легко обчислити за допомогою будь-яких формул числового інтегрування, наприклад квадратурних формул Ньютона–Котеса або Гаусса [59].

Отримана функція щільності гамма-розподілу використовується для перетворення випадкових незалежних рівномірно розподілених величин. Для цього область можливих значень випадкової величини X розбивається на n однакових інтервалів, кількість яких залежить від заданої точності апроксимації функції $f(x)$. Потім за допомогою значення r_i (методом розіграшу за жеребом) обирається один із n інтервалів, у якому отримують випадкові числа з функцією щільності розподілу $f(x)$.

Для оцінювання близькості функції щільності розподілу ймовірностей отриманих значень випадкової величини до функції щільності розподілу $f(x)$ використовують метод найменших квадратів. Цей метод передбачає задання максимально допустимої похибки (наприклад, $\varepsilon = 10^{-4}$).

Наведені вище формули та метод кускової апроксимації функції щільності можна використати, щоб задати таблицю значень функції гамма-розподілу при фіксованих значеннях параметрів α і β, як це зроблено для мови GPSS у генераторі програм ISS 2000 [61].

Наведемо ще кілька алгоритмів моделювання випадкової величини, яка має гамма-розподіл, при різних значеннях параметра α [46].

1. $0 < \alpha < 1$.

Генеруємо три числа: r_1, r_2 та r_3, які є незалежними реалізаціями випадкової величини, рівномірно розподіленої в інтервалі [0, 1]. Обчислимо значення: $X = r_1^{1/\alpha}$, $Y = r_2^{1/(1-\alpha)}$.

Якщо $X + Y \le 1$, обчислюємо $W = X(X + Y)$ і розраховуємо значення γ_i випадкової величини за формулою

$$\gamma_i = W(-\ln(r_3))\,\beta.$$

Або знаходимо нові значення X і Y і повторно перевіряємо умову $X + Y \le 1$.

2. $1 \le \alpha < 5$.

Позначимо через $a = \lfloor \alpha \rfloor$ цілу частину від α і через $b = \alpha - \lfloor \alpha \rfloor$. Обчислимо значення $X = \frac{\alpha}{a}\left[-\ln\left(\prod_{i=1}^{a} r_i\right)\right]$.

Якщо $r_{a+1} > \left(\frac{X}{\alpha}\right)^b e^{-\frac{X}{\alpha-1}}$, то обчислюємо нове значення X.

Або розраховуємо значення γ_i випадкової величини за формулою

$$\gamma_i = \pm X \beta.$$

3. $\alpha \ge 5$.

У цьому випадку провадиться зважений відбір значень послідовності, що має розподіл Ерланга.

Якщо $r_1 \ge \alpha - \lfloor \alpha \rfloor$, то γ_i визначається як значення випадкової величини з розподілом Ерланга з параметрами $\lfloor \alpha \rfloor$, β.

Якщо $r_1 < \alpha - \lfloor \alpha \rfloor$, то γ_i обчислюється як значення випадкової величини, що має розподіл Ерланга з параметрами $\lfloor \alpha \rfloor + 1$, β.

4.5.9. Бета-розподіл

Бета-розподіл визначений у скінченному інтервалі й при різних значеннях параметрів описується різноманітними кривими. Ці криві можуть бути симетричними, асиметричними, мати форму дзвону, U-подібну форму і т. ін. Одна із найпростіших різновидностей бета-розподілу – розподіл Парето, який часто використовується в економічних моделях для моделювання розподілу доходів або витрат.

Те, що бета-розподіл визначений лише в скінченному інтервалі, вносить обмеження на об'єкт моделювання (значення випадкової величини X лежить в інтервалі від 0 до 1). Прикладами можуть бути функції щільності оцінок імовірності, або частки чогось, експертні суб'єктивні ймовірності події, що нас цікавлять.

Функція щільності бета-розподілу має вигляд

$$f_\beta(x) = \begin{cases} \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\,\Gamma(\alpha_2)} x^{\alpha_1 - 1} (1-x)^{\alpha_2 - 1}, & 0 \le x \le 1; \\ 0, & \text{в інших випадках,} \end{cases}$$

де $0 < \alpha_1, \alpha_2 < \infty$.

Математичне сподівання

$$M_\beta[x] = \frac{\alpha_1}{\alpha_1 + \alpha_2},$$

а дисперсія

$$D_\beta[x] = \frac{\alpha_1 \alpha_2}{(\alpha_1 + \alpha_2)^2 (\alpha_1 + \alpha_2 + 1)}.$$

Види функцій розподілу залежно від параметрів α_1 та α_2, зображено на рис. 4.21.

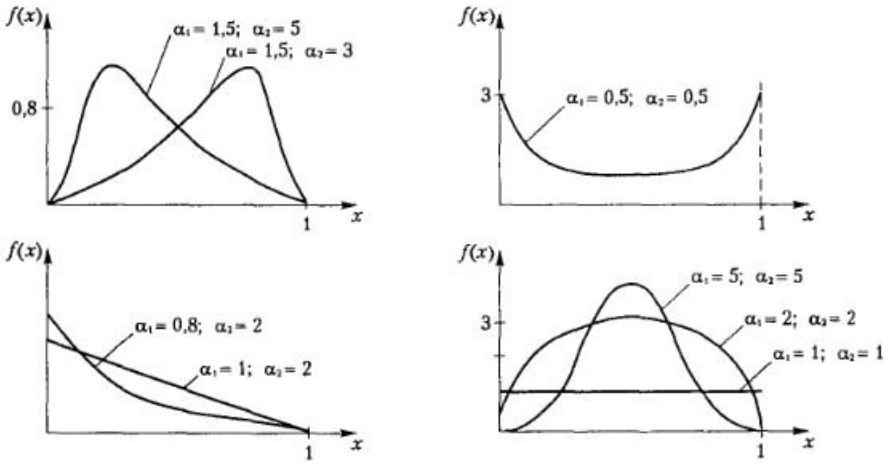


Рис. 4.21. Функції щільності бета-розподілу

Метод моделювання випадкової величини базується на такій властивості бета-розподілу: якщо γ_1 і γ_2 – дві незалежні гамма-розподілені випадкові величини з параметрами α_1, β та α_2, β відповідно, то значення $\gamma_1/(\gamma_1 + \gamma_2)$ підпорядковується бета-розподілу з параметрами α_1 і α_2. Тому метод моделювання передбачає перетворення випадкової величини за формулою

$$x_i = \frac{\gamma_1}{(\gamma_1 + \gamma_2)},$$

де γ_1 і γ_2 – дві незалежні гамма-розподілені випадкові величини з параметрами α_1, 1 та α_2, 1.

4.5.10. Розподіл Вейбулла

Розглянемо, якому розподілу підпорядковуються значення випадкової величини, які визначають тривалість безвідмовної роботи складної системи з кількох об'єктів, за умови, що з ладу можуть виходити окремі об'єкти. Через ξ позначимо тривалість безвідмовної роботи системи, через $F_\xi(t) = P(\xi < t)$ – неперервну і диференційовану функцію розподілу випадкової величини, а через $\lambda(t)$ – інтенсивність відмови елементів, що працювали до часу t:

$$\lambda(t) = -\frac{f_\xi(t)}{1 - F_\xi(t)} \approx \frac{n(t) - n(t + \Delta t)}{\Delta t\, n(t)}, \qquad (4.10)$$

де $n(t)$ – число об'єктів системи, які працювали безвідмовно до моменту часу t, а Δt – нескінченно малий відрізок часу. Інтенсивність відмов визначається як відношення кількості об'єктів системи, що вибули з ладу до моменту часу t, до

загальної кількості об'єктів системи, що працювали безвідмовно, $n(t)$. Розв'яжемо рівняння (4.10) відносно функції розподілу $F_\xi(t)$:

$$F_\xi(t) = 1 - e^{-\int_0^t \lambda(t)\,dt}. \qquad (4.11)$$

Із формули випливає, що конкретний вигляд функції $F_\xi(t)$ залежить від функції $\lambda(t)$. На рис. 4.22 зображено графік життєвого циклу складного виробу. У ньому можна виділити три періоди, кожному з яких відповідають три окремих відрізки графіка. Для кожного відрізка існує своя функція $\lambda(t)$ і, отже, свій закон розподілу часу безвідмовного функціонування системи $F_\xi(t)$. Для першого відрізка (період припрацьовування) системи параметр $\alpha < 1$, для другого (період нормальної експлуатації) $\alpha = 1$, для третього (період старіння) $\alpha > 1$. На графіку видно, чому в період припрацьовування не бажано продавати вироби або експлуатувати в критичних режимах складні системи.

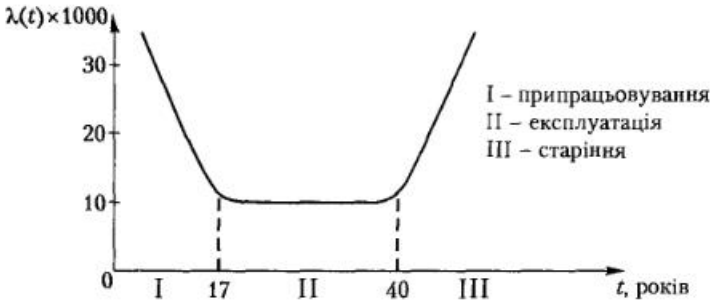


Рис. 4.22. Графік функції зміни інтенсивності відмов у часі для складної системи

Розглянемо випадок, коли функція $\lambda(t)$ має вигляд

$$\lambda(t) = \lambda_0 \alpha t^{\alpha-1},$$

де λ_0, $\alpha > 0$ – деякі числові параметри, які характеризують систему. Якщо підставити різні значення α у формули (4.10) та (4.11), отримаємо

$$F_\xi(t) = 1 - e^{-\lambda_0 t^\alpha}, \quad t \geq 0.$$

Відповідно, функція щільності ймовірності такої випадкової величини має такий вигляд:

$$f_\xi(t) = \lambda_0 \alpha\, t^{\alpha-1} e^{-\lambda_0 t^\alpha}.$$

Це і є розподіл Вейбулла. Математичне сподівання і дисперсія задаються виразами

$$M_\xi[t] = \lambda_0^{-\frac{1}{\alpha}} \Gamma\left(1+\frac{1}{\alpha}\right) \quad \text{та} \quad D_\xi[t] = \lambda_0^{-\frac{2}{\alpha}} \left[\Gamma\left(1+\frac{2}{\alpha}\right) - \Gamma^2\left(1+\frac{1}{\alpha}\right)\right].$$

Для моделювання випадкових величин w_i, розподілених по закону Вейбулла, використовуються незалежні рівномірно розподілені в інтервалі [0, 1] випадкові величини, що перетворюються за методом оберненої функції. Значення w_i отримують за формулою

$$w_i = \left(\frac{-\ln(r_i)}{\lambda_0}\right)^{1/\alpha},$$

де $1/\lambda_0$ – масштабний параметр; α – параметр крутизни.

4.5.11. Гіпер- і гіпоекспоненціальні розподіли

Експоненціально розподілені випадкові величини використовуються і для моделювання випадкових величин з гіпер- і гіпоекспоненціальним розподілом. Ці розподіли дають змогу замінити неекспоненціальний розподіл випадкової величини сумою незалежних зважених експоненціальних розподілів (такий спосіб називається *методом суперпозиції*). Вони широко використовуються в теорії масового обслуговування. Це дає можливість застосовувати методи теорії марківських процесів з неперервним часом для розрахунків характеристик систем, в яких випадкові процеси підпорядковуються неекспоненціальним законам розподілу.

Якщо потрібно отримати випадкову величину з неекспоненціальним розподілом з коефіцієнтом варіації[1] $C > 1$, можна скористатись гіперекспоненціальним розподілом, з функцією розподілу ймовірностей

$$F_t(x) = P(X \le x) = \sum_{i=1}^{k} \omega_i (1 - e^{-\mu_i x}); \quad \mu_i > 0, \quad \omega_i > 0; \quad \sum_{i=1}^{k} \omega_i = 1.$$

Математичне сподівання і дисперсія цієї випадкової величини задаються виразами

$$M[x] = \sum_{i=1}^{k} \frac{\omega_i}{\mu_i}, \quad D[x] = \sum_{i=1}^{k} \frac{2\omega_i}{\mu_i^2} - \left(\sum_{i=1}^{k} \frac{\omega_i}{\mu_i}\right)^2.$$

Легко показати, що коефіцієнт варіації визначається як

$$C = \frac{\sqrt{D(t)}}{\bar{x}} \ge 1.$$

Експоненціальний розподіл з коефіцієнтом варіації $C = 1$ отримуємо за умови, що $\mu_i = \mu$ для всіх i.

Розглянемо деякий обслуговуючий центр, обведений контуром на рис. 4.23, який має k паралельно з'єднаних пристроїв для обслуговування з імовірністю використання ω_i. Припустимо також, що в довільний момент часу може бути зайнято не більше одного пристрою з k, тобто нова вимога надходить до обслуговуючого центру тільки після того, як закінчиться обслуговування попередньої вимоги

[1] Коефіцієнт варіації C — це відношення стандартного відхилення до математичного сподівання випадкової величини.

і вона залишить центр. Тоді, якщо час обслуговування вимог на кожному пристрої підпорядковується експоненціальному закону розподілу з інтенсивністю μ_i, то час обслуговування вимог у центрі в цілому матиме гіперекспоненціальний розподіл. Схема з паралельними етапами обслуговування, наведена на рис. 4.23, використовується для моделювання гіперекспоненціального розподілу. Наприклад, під час моделювання обчислювальних систем такий розподіл добре описує функціонування центрального процесора комп'ютера [64].

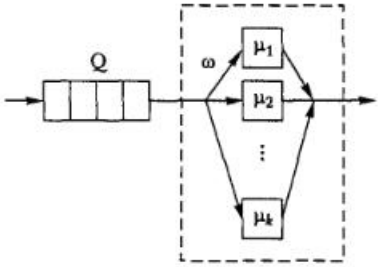


Рис. 4.23. Схема моделі для отримання гіперекспоненціального розподілу

Якщо необхідно отримати розподіл з коефіцієнтом варіації $C < 1$, можна скористатись гіпоекспоненціальним розподілом з функцією розподілу ймовірностей

$$F_t(x) = \mathrm{P}(X \le x) = 1 - \frac{\mu_2}{\mu_2 - \mu_1} e^{-\mu_1 x} - \frac{\mu_1}{\mu_1 - \mu_2} e^{-\mu_2 x}, \quad (\mu_1 \ne \mu_2).$$

Математичне сподівання, дисперсія і коефіцієнт варіації випадкової величини x задаються виразами

$$\mathrm{M}[x] = \frac{1}{\mu_1} + \frac{1}{\mu_2}; \quad \mathrm{D}[x] = \frac{1}{\mu_1^2 + \mu_2^2}; \quad C = \sqrt{1 - \frac{2\mu_1\mu_2}{(\mu_1 + \mu_2)^2}} < 1.$$

За умови, що всі коефіцієнти однакові ($\mu_k = \mu$), час перебування вимоги в обслуговуючому центрі (обведений на рис. 4.24) буде мати k-розподіл Ерланга:

$$F_t(x) = 1 - e^{-k\mu x} \sum_{i=0}^{k-1} \frac{(k\mu x)^2}{i!}.$$

Для моделювання пристрою СМО, час обслуговування якого є випадковою величиною з гіпоекспоненціальним розподілом, необхідно послідовно з'єднати k пристроїв для обслуговування, у кожному з яких час обслуговування має експоненціальний розподіл. Під час моделювання слід ураховувати, що в будь-який момент часу повинен бути зайнятим лише один пристрій (рис. 4.24), тобто нова вимога може надійти до першого пристрою для обслуговування тільки після того, як попередня вимога залишить останній пристрій.

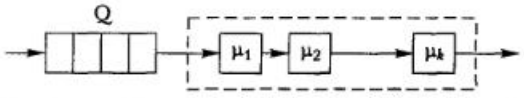


Рис. 4.24. Схема моделі для отримання гіпоекспоненціального розподілу

У разі моделювання обчислювальних систем випадкові величини з таким розподілом застосовують для визначення часу роботи пристроїв введення-виведення комп'ютера.

4.6. Моделювання випадкових векторів

Під час моделювання систем керування багатьох типів виникає необхідність генерувати багатовимірні випадкові вектори, які мають заданий сумісний розподіл або багатомірний розподіл. У цьому разі окремі компоненти вектора можуть бути незалежними.

Розглянемо моделювання неперервного випадкового вектора зі складовими X, Y. Нехай випадкові величини X, Y описуються спільною функцією щільності $f(x, y)$, яку може бути використано для визначення функції щільності $f(x)$ випадкової величини X:

$$f_\xi(x) = \int_{-\infty}^{+\infty} f(x, y)\, dy.$$

Маючи функцію щільності $f(x)$, можна знайти випадкове число x_i, а потім, якщо $x = x_i$, знайти умовний розподіл випадкової величини Y:

$$f_\eta(y_i \mid \xi = x_i) = f(x, y) \mid f_\xi(x_i).$$

З цього виразу для функції щільності можна визначити випадкову величину y. Тоді пара чисел (x_i, y_i) буде реалізацією неперервного випадкового вектора (X, Y). Такий спосіб реалізації двомірних векторів можна узагальнити і для моделювання багатомірних випадкових векторів. Однак слід мати на увазі: зі збільшенням числа компонентів вектора складність обчислень різко зростає, що перешкоджає широкому використанню цього методу на практиці.

4.7. Моделювання випадкових процесів

Випадковий процес – це процес (тобто зміна в часі стану деякої системи чи об'єкта), який розвивається під впливом якихось випадкових чинників і для якого задано ймовірнісні характеристики його протікання. До числа таких процесів можна віднести багато виробничих процесів, які супроводжуються випадковими флуктуаціями, а також процесів, з якими можна зустрітись у природничих науках, економіці, соціології тощо.

Моделювання будь-якого процесу, в тому числі і випадкового, полягає у відтворенні значень (величин) реалізації цього процесу. Випадковий стаціонарний процес задається значеннями математичного сподівання та автоковаріаційною або автокореляційною функцією. Для його моделювання скористаємось параметричними моделями авторегресії, які широко застосовуються для аналізу часових рядів [6].

Авторегресійний процес k-го порядку з постійними коефіцієнтами визначається рівнянням регресії

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_k y_{t-k} + e_t. \qquad (4.12)$$

Значення процесу (4.12) у будь-який момент часу t визначається через попередні значення та випадкове збурення e_t. На практиці звичайно використовують авторегресійні моделі процесів першого і другого порядку (процес Маркова і Юла–Уокера), автокореляційна функція яких є згасаючою (рис. 4.25).

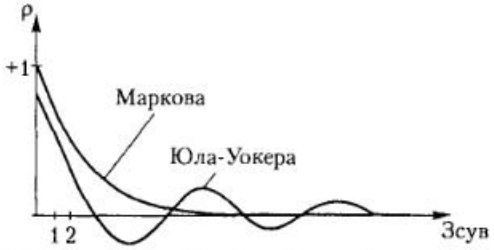


Рис. 4.25. Автокореляційна функція стаціонарного процесу

Параметри процесу $a_1, \ldots, a_k$ визначаються через коефіцієнти автокореляції. Так, для процесу Юла

$$a_0 = \mathrm{M}[y_t], \quad a_1 = \frac{\rho_1(1-\rho_2)}{1-\rho_1^2}, \quad a_2 = \frac{\rho_2 - \rho_1^2}{1-\rho_1^2},$$

де ρ_1, ρ_2 – значення автокореляційної функції при зсувах 1 та 2.

Під час побудови рівняння авторегресії висуваються дві гіпотези. Перша – про стаціонарність процесу, друга – про те, що збудження e_t є випадковим процесом у широкому розумінні слова з нормальною функцією розподілу, нульовим математичним сподіванням і дисперсією σ^2. На рис. 4.26, *а* зображено графік випадкового процесу і нормальний розподіл збурення e_t (рис. 4.26, *б*).

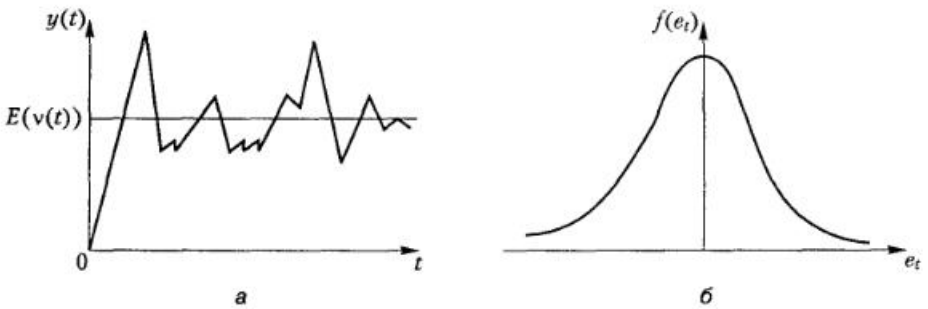


Рис. 4.26. Графіки випадкового процесу (*а*) і нормального розподілу збурення (*б*)

Процес y_t називається марківським, якщо для будь-яких моментів часу $t_1 < t_2 < t_3 < \ldots < t_n$ умовна ймовірність значення y_t залежатиме від y_{t-1} і не залежа-

тиме від того, в якому стані процес знаходився в попередні моменти часу. Під час моделювання марківського стаціонарного процесу з параметрами $M[y_t]$, ρ_1, σ^2 діють таким чином. За початковий член ряду можна взяти будь-яке значення випадкового процесу (тому що будь-яка частина стаціонарного процесу є повноцінним представником усього процесу і має ті ж імовірнісні характеристики), наприклад $y_{t-1} = 0$ або $y_{t-1} = M[y_t]$. За формулою (4.12) розраховуємо значення y_{t+1} при заданих $a_0 = M[y_t]$ і $a_1 = \rho_1$ без урахування збурення e_t і моделюємо значення нормально розподіленої випадкової величини з математичним сподіванням, що дорівнює нулю, і дисперсією σ^2. Отримане значення додаємо до y_{t-1} і таким чином отримуємо нове значення реалізації випадкового процесу. Повторюємо процедуру для обчислення інших значень за формулою (4.12), задаючи як початкове значення y_t, тобто моделюємо y_{t+1}, ..., y_{t+k}. Така методика дає змогу моделювати випадкові стаціонарні процеси з будь-якими автокореляційними та багатомірними функціями розподілів.

4.8. Статистична обробка результатів моделювання

Основою для обчислення статистичної оцінки параметра системи є реалізація випадкової величини, яка формується під час прогонів імовірнісної імітаційної моделі. Статистична оцінка також є функцією від випадкових величин, які дістають у результаті прогонів моделі, тому і згадана оцінка є випадковою величиною, закон розподілу якої залежить від закону розподілу досліджуваної випадкової величини та оцінюваного параметра. Чим більше реалізацій випадкової величини, тим точнішу статистичну оцінку параметра системи ми отримуємо. У таких умовах обробка результатів моделювання повинна провадитись лише з використанням методів і алгоритмів, які є оптимальними з погляду затрат часу та використання ресурсів комп'ютера. Під час вибору таких засобів необхідно враховувати, що всі статистичні оцінки мають бути ще й *незміщеними*, *ефективними* і *спроможними*.

Розглянемо деякі методи обчислення основних статистичних оцінок.

4.8.1. Оцінювання ймовірності

Оцінкою ймовірності p настання деякої події A є її частість:

$$\hat{p} = \frac{m}{N},$$

де m – кількість випробувань, під час за яких випадкова подія спостерігалась; N – загальна кількість випробувань. Для її використання зазвичай на програмному рівні організовують два лічильники, один з яких призначено для підрахунку загальної кількості випробувань N, а другий – кількості успішних випробувань m.

4.8.2. Оцінювання розподілу випадкової величини

Для оцінювання функції розподілу випадкової величини, як звичайно, будується гістограма. Під час її побудови область можливих значень випадкової величини розбивають на n діапазонів і підраховують кількість попадання значень випадкової величини в конкретний інтервал – m_k (k = 1, ..., n). Оцінка ймовірності попадання випадкової величини в k-й інтервал має такий вигляд:

$$\hat{p}_k = \frac{m_k}{N}.$$

Цю величину називають відносною частістю. У процесі моделювання під час підрахунку значень m_k кожному інтервалу ставлять у відповідність окремий лічильник.

4.8.3. Оцінювання математичного сподівання

Для оцінювання математичного сподівання випадкової величини використовується формула

$$\mathrm{M}(x) = \bar{x} = \frac{1}{N}\sum_{k=1}^{n} m_k x_k,$$

де x_k – значення випадкової величини, що належить k-му інтервалу; m_k – кількість попадань значень випадкової величини в інтервал; N – загальна кількість випробувань. У більш простому випадку для оцінювання математичного сподівання випадкової величини можна використати звичайне середнє арифметичне:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Щоб запобігти непотрібному завантаженню пам'яті, суму доцільніше підраховувати шляхом поступового накопичення.

4.8.4. Оцінювання дисперсії

Для оцінювання дисперсії випадкової величини можна використати формулу

$$S^2 = \frac{1}{N-1}\sum_{k=1}^{n}(x_k - \bar{x})^2 m_k$$

або в простому випадку

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2,$$

де S^2 – оцінка дисперсії випадкової величини x.

Безпосередньо використовувати ці формули в розрахунках дисперсії нераціонально, тому що зі збільшенням кількості значень x_i змінюється також середнє

значення, а для його обчислення потрібно запам'ятовувати всі N значень x_i. Тому доцільніше використовувати таку формулу:

$$S^2 = \hat{\sigma}^2 = \frac{1}{N-1}\left[\sum_{i=1}^{N} x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2\right].$$

У цьому випадку достатньо накопичувати тільки суми двох послідовностей – x_i і x_i^2. Але і такий спосіб має недолік – його використання може призвести до переповнення розрядної сітки комп'ютера. Щоб запобігти цьому, потрібно змінити послідовність дій при обчисленнях, використовуючи формулу

$$S^2 = \hat{\sigma}^2 = \sum_{i=1}^{N}\left(\frac{x_i}{N-1}\right) x_i - \left(\sum_{i=1}^{N} x_i \Big/ N\right)\left(\sum_{i=1}^{N} x_i \Big/ n-1\right).$$

4.8.5. Оцінювання кореляційного моменту

Для обчислення оцінки кореляційного моменту можна використовувати формулу

$$\hat{k}_{XY} = \frac{1}{N-1}\left[\sum_{k=1}^{N}(x_k - \bar{x})(y_k - \bar{y})\right]$$

або, більш зручну для обчислень,

$$\hat{k}_{XY} = \frac{1}{N-1}\left[\sum_{k=1}^{N} x_k y_k - \frac{1}{N}\sum_{k=1}^{N} x_k \sum_{k=1}^{N} y_k\right].$$

При обчисленнях за цією формулою теж доцільно змінити послідовність дій.

4.9. Визначення кількості реалізацій під час моделювання випадкових величин

Точність оцінок параметрів системи, які отримують під час обробки результатів моделювання, у першу чергу залежить від кількості випробувань N. Слід враховувати, що обсяг вибірки N завжди обмежений, тому вищезгадані оцінки матимуть різні похибки і дисперсії.

Якщо треба оцінити значення деякого параметра a за результатами моделювання x_i, то за його оцінку слід брати величину $\bar{x}$, яка є функцією від усіх значень x_i. Статистична оцінка $\bar{x}$ також є випадковою величиною, тому вона буде відрізнятись від a, тобто

$$|a - \bar{x}| < \varepsilon,$$

де ε – точність або похибка оцінки. Імовірність того, що ця нерівність виконується, позначимо через α:

$$P\left(|a-\bar{x}|<\varepsilon\right)\geq\alpha. \tag{4.13}$$

У теорії ймовірностей ε – це *довірчий інтервал* для α, довжина якого фактично дорівнює 2ε, а α – *довірчий рівень*, або *надійність оцінки*. Вираз (4.13) можна застосувати для визначення точності результатів статистичних випробувань.

4.9.1. Оцінювання ймовірності

Припустимо, що метою моделювання є оцінювання ймовірності настання деякої події A, яка визначає стан системи. У кожній з N реалізацій процесу настання події A є випадковою величиною ξ, що набуває значення $x_1 = 1$ з імовірністю p і $x_2 = 0$ з імовірністю $1-p$. Тоді можна визначити математичне сподівання і дисперсію відповідно за формулами

$$\mathrm{M}[\xi] = x_1 p + x_2 (1-p) = p, \tag{4.14}$$

$$\mathrm{D}[\xi] = (x_1 - \mathrm{M}[\xi])^2 p + (x_2 - \mathrm{M}[\xi])^2 (1-p) = p(1-p). \tag{4.15}$$

Як оцінку p використовують частість настання події A. Ця оцінка є незміщеною, спроможною та ефективною. За умови, що N задано, для отримання цієї оцінки достатньо накопичувати m:

$$\frac{m}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{4.16}$$

де x_i – настання події A в реалізації i.

За формулами (4.14)–(4.16) визначимо вибіркове математичне сподівання $\mathrm{M}[m/N] = p$ і дисперсію $\mathrm{D}[m/N] = p(1-p)/(N-1)$.

Згідно з центральною граничною теоремою (у даному випадку її можна взяти у вигляді теореми Хінчина) випадкова величина m/N буде мати розподіл, близький до нормального (рис. 4.27). Тому для кожного рівня достовірності α з таблиць нормального розподілу можна знайти таку величину t_α, при якій точність обчислюватиметься за формулою

$$\varepsilon = t_\alpha \sqrt{\mathrm{D}\,[m/N]}. \tag{4.17}$$

Якщо $\alpha = 0{,}05$, то $t_\alpha = 1{,}96$, а якщо $\alpha = 0{,}003$, то $t_\alpha = 3$.

Підставимо у формулу (4.17) вираз дисперсії:

$$\varepsilon = t_\alpha \sqrt{\frac{p(1-p)}{N-1}}.$$

Звідси

$$N = t_\alpha^2 \frac{p(1-p)}{\varepsilon^2} + 1. \tag{4.18}$$

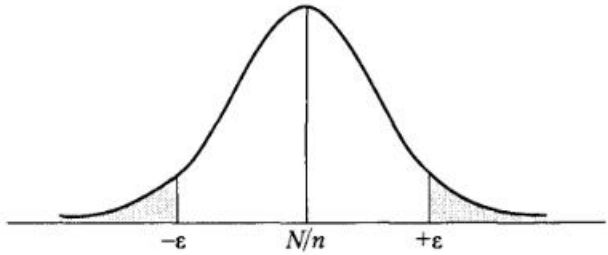


Рис. 4.27. Функція нормального розподілу для побудови довірчого інтервалу

З формули (4.18) видно, що при $p = 1$ або $p = 0$, кількість реалізацій, які необхідно провести для підтвердження того, що подія A настає (або ні), дорівнює одиниці. Але оскільки ймовірність p заздалегідь невідома, провадять випробування ($N = 50 ... 100$), оцінюють частість m/N і підставляють її значення у вираз (4.18) замість p, після чого визначають остаточну кількість реалізацій. Графік залежності числа реалізацій для $\alpha = 0{,}05$ і різних значень p, якщо $\varepsilon = 0{,}05$, наведено на рис. 4.29.

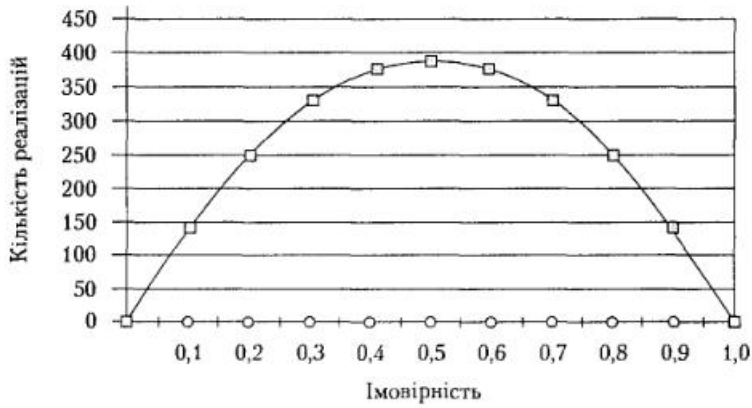


Рис. 4.28. Залежність числа реалізацій від значень імовірності

4.9.2. Оцінювання середнього значення

Нехай випадкова величина має математичне сподівання a і дисперсію σ^2. У i-й реалізації вона набуває значення x_i. Як оцінку математичного сподівання a використаємо середнє арифметичне:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \qquad (4.19)$$

Згідно з центральною граничною теоремою при великих значеннях N середнє арифметичне (4.19) буде мати нормальний розподіл з математичним сподіванням a і дисперсією $\sigma^2/(N-1)$. Тоді

$$\varepsilon = t_\alpha \frac{\sigma}{\sqrt{N-1}}.$$

Звідси

$$N = t_\alpha^2 \sigma^2 / \varepsilon^2 + 1. \qquad (4.20)$$

Оскільки дисперсія σ^2 випадкової величини невідома, потрібно провести кілька десятків (50 ... 100) випробувань і знайти оцінку $\hat{\sigma}^2$, а потім отримане значення підставити у формулу (4.20), щоб визначити необхідну кількість реалізацій N. У цьому випадку замість нормально розподіленої величини необхідно скористатись t-розподілом Стьюдента з $N - 1$ ступенями вільності для визначення t_α. Зауважимо, що за збільшення ступенів вільності t-розподіл наближається до нормального. З практичного погляду, якщо N більше 30, користуються нормальним розподілом.

Висновки

- Метод статистичних випробувань визначається як спосіб побудови і дослідження на комп'ютері моделі системи або процесу з використанням послідовностей випадкових чисел.
- Метод полягає в багатократному проведенні випробувань побудованої моделі й подальшій статистичній обробці результатів моделювання з метою визначення шуканих характеристик розглядуваного процесу у вигляді оцінок його параметрів.
- У методі статистичних випробувань особливе значення відіграють випадкові числа, рівномірно розподілені в інтервалі [0, 1], за допомогою яких можна отримати вибіркові значення з будь-якими статистичними властивостями.
- Для генерування випадкових чисел використовуються апаратні, табличні та програмні методи.
- Апаратні методи генерування випадкових чисел базуються на використанні деяких фізичних явищ і процесів – випадковий електричний сигнал перетворюють у двійковий код, який уводиться в комп'ютер за допомогою спеціальних аналого-цифрових перетворювачів.
- У разі використання табличного методу випадкові числа можна зберігати на зовнішніх носіях або навіть в основній пам'яті комп'ютера.
- Програмні генератори дають змогу отримувати послідовності випадкових чисел за рекурентними формулами. Більшість програмних генераторів виробляють випадкові числа, рівномірно розподілені в інтервалі [0, 1].
- Серед програмних генераторів найбільш розповсюдженим є лінійний мультиплікативний конгруентний генератор.
- Основним параметром програмного генератора є повний період – кількість чисел, після якої випадкові числа починають повторюватися.
- Для перевірки статистичних властивостей усіх послідовностей випадкових чисел, які будуть використовуватись під час проведення досліджень, застосовують емпіричні та теоретичні критерії.

- Емпіричні критерії – це звичайні статистичні тести, в яких використовують вибіркові значення, що виробляються генератором.
- Теоретичні критерії визначають характеристики послідовності за допомогою методів, які формуються на основі числових значень параметрів генератора.

Контрольні запитання та завдання

1. Доведіть, що незалежні випадкові величини r_i та $1 - r_i$, отримані з рівномірного розподілу в інтервалі [0, 1], мають однаковий розподіл.
2. Обчисліть імовірність того, що за 50 підкидань монети 10 разів випаде герб. Спочатку розв'яжіть задачу аналітично, а потім порівняйте результат з оцінкою, отриманою за допомогою методу статистичних випробувань.
3. Використовуючи метод статистичних випробувань, обчисліть площу круга. Зважаючи на те, що формула для визначення площі круга – $S = \pi r^2$, оцініть значення константи π.
4. Молодий пілот, виконуючи перший свій політ, не зміг здійснити вдалу посадку літака. Визначте число спроб зайти на посадку, які необхідно зробити пілоту, для того щоб політ благополучно закінчився. Ймовірність того, що за одну спробу стажист вдало завершить посадку літака, становить 0,1. Кількість палива в баках вважати необмеженою.
5. Скориставшись методом статистичних випробувань, знайдіть оцінку інтеграла $\int_0^1 x^{3/2}\,dx$ як площі під кривою $x^{3/2}$ на відрізку $0 \le x \le 1$.
6. Імовірність одержання заліку студентом, який не відвідував лекційні і практичні заняття, становить 0,13. Скориставшись методом статистичних випробувань, знайдіть оцінку ймовірності того, що студент одержить залік, якщо загальне число спроб здати залік не може перевищувати 3.
7. Знайдіть оцінку площі рівнобічного трикутника зі стороною 1 см. Визначіть рівняння прямих, які утворюють сторони трикутника і використайте ці вирази в процедурі оцінювання попадання «випадкової» точки в трикутник для методу статистичних випробувань.
8. Для одержання допуску до іспиту студенту необхідно отримати залік з лабораторних робіт (лабораторний курс складається з K лабораторних робіт). Припустимо, що ймовірність здавання однієї лабораторної роботи студентом становить 0,3 з однієї спроби. Оцініть кількість днів, потрібних студенту на одержання допуску до іспиту, якщо за один день він зможе здати не більше однієї лабораторної роботи (протягом семестру він не захистив жодної роботи).
9. Стрільцю необхідно вразити чотири мішені. Ймовірність ураження однієї мішені становить 0,14. Оцініть кількість набоїв, необхідних для того, щоб уразити всі чотири мішені.

10. Багдадський злодій ув'язнений у підземелля з трьома дверима [46]. Одні двері ведуть на волю, другі – у довгий тунель, а треті – у короткий. Потрапивши в один із тунелів, злодій знову опиняється в темниці. Він знову пробує вийти на волю, але не пам'ятає, в які двері входив минулого разу. Імовірність того, що злодій обере потрібні двері, дорівнює 0,3, ймовірність попадання в короткий тунель – 0,2; ймовірність попадання в довгий тунель – 0,5. Час перебування злодія в короткому тунелі – 3 хв, у довгому – 6 хв. Визначіть середній час пошуку шляху на волю. Побудуйте процедуру статистичних випробувань для розв'язування задачі.
11. Змоделюйте поведінку винищувача-бомбардувальника [67], який атакує об'єкт ракетами класу «повітря–земля». Кожна ракета наводиться індивідуально. Розміри об'єкта – 60×150 м. Заходження на атаку відбуваються в напрямі, який збігається з напрямом довгої осі цілі, точка прицілу – геометричний центр цілі. Фактичну точку влучення для кожної ракети можна визначити горизонтальним і вертикальним відхиленнями (рис. 4.29). Для відстані, з якої запускають ракети, обидва відхилення є незалежними, нормально розподіленими щодо точки прицілювання і мають нульове середнє значення.

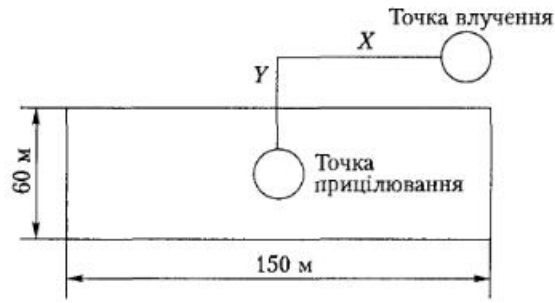


Рис. 4.29. Схема влучення ракети в ціль

Середньоквадратичне відхилення становить 60 м у напрямку X і 30 м – у напрямку Y. Бомбардувальник під час кожного заходження випускає шість ракет. Узявши обсяг вибірки в 10 заходжень, знайдіть оцінку середнього числа влучень для кожної атаки.

12. Змоделюйте випадкову двомірну дискретну величину (X, Y). Випадкова величина X може набувати значення 1, 2, 3, 4, а випадкова величина Y – 10, 20, 30, 40, у цьому разі кожній парі значень x_i, y_i відповідає ймовірність p_{ij} (табл. 4.5).

Таблиця 4.5. Ймовірність двомірної дискретної величини

Значення x_i	Значення y_i			
	10	20	30	40
1	0,02	0,05	0,1	0,05
2	0,03	0,1	0,05	0,03
3	0,01	0,15	0,1	0,02
4	0,04	0	0,15	0,1

13. Припустимо, що об'єм споживання води в місті є випадковою величиною, що має нормальний розподіл [67]. Знайдіть оцінку середнього споживання води в день так, щоб помилка не перевищувала ±6000 л з імовірністю 0,95. Відомо, що розумна область розкиду споживання води становить 120 000 л за день. Який обсяг вибірки необхідний для цього дослідження?
14. Використовуючи результати 20 імітаційних прогонів для оцінювання часу перебування відвідувачів у системі [46], що наведені в дужках (1,1; 2,8; 3,7; 1,9; 4,9; 1,6; 0,4; 3,8; 1,5; 3,4; 1,9; 2,1; 3,8; 1,6; 3,2; 2,9; 3,7; 2,0; 4,2; 3,3), обчисліть оцінки для вибіркового середнього, дисперсії і коефіцієнта варіації. Побудуйте гістограму, яка містить п'ять інтервалів (довжина кожного інтервалу дорівнює одиниці, нижня межа першого інтервалу дорівнює нулю).
15. Дано випадкові некорельовані змінні A, B, C. Змінна A має нормальний розподіл з математичним сподіванням 100 і середньоквадратичним відхиленням 20. Змінна B також розподілена нормально з математичним сподіванням 20 і середньоквадратичним відхиленням 5. Розподіл змінної C задано в табл. 4.6.

Таблиця 4.6. Функція розподілу випадкової величини

Значення C	10	20	30	40
Імовірність	0,10	0,25	0,50	0,15

Застосовуючи метод статистичних випробувань, оцініть середнє значення нової змінної D, що визначається так: $D = (A + B)/C$. Використайте вибірку із 10 значень, яку необхідно отримати за допомогою розподілу, що демонструється на рис. 4.15.

16. Проаналізуйте модель наземної протиповітряної ракетної установки. Скільки необхідно виконати випробувань, кожне з яких моделює одну відсіч повітряної атаки, щоб оцінка частоти поразки цілі відрізнялася від істинної ймовірності не більше ніж на 0,02 з імовірністю не меншою за 0,85?

Розділ 5
Імітаційне моделювання

- Методи проектування імітаційних моделей
- Формулювання проблеми та змістовна постановка задачі
- Побудова концептуальної моделі
- Автоматизація програмування
- Програмна реалізація імітаційної моделі
- Валідація та верифікація імітаційної моделі

У розділі 1.4 було подано визначення імітаційного моделювання та імітаційної моделі. Завдяки зниженню вартості комп'ютерної техніки та доступності спеціальних програмних засобів методи імітаційного моделювання широко використовуються для дослідження складних об'єктів і систем у найрізноманітніших галузях людської діяльності.

5.1. Доцільність використання імітаційного моделювання

Переваги застосування імітаційного моделювання найбільш помітно виявляються у разі моделювання виробничих і технологічних процесів, процесів матеріально-технічного забезпечення виробництва, у логістиці, а також під час проведення бізнес-планування, екологічних і соціологічних досліджень. Важливо, що імітаційне моделювання використовується, скоріше, як спосіб для осмислення проблеми і допомагає в цьому більше, ніж простий текстовий або математичний опис проблеми. Воно дає змогу глянути на складний процес ухвалення рішення більш масштабно, з погляду процесів, які відбуваються всередині системи, що моделюється.

Часто моделювання припиняють ще до того, як будуть отримані конкретні результати. Визначення моменту, в який зацікавлені сторони зрозуміють, що ж насправді відбувається в системі, уже може бути рішенням проблеми. Тому навіть не завжди потрібно провадити статистичну обробку результатів експерименту. Звичайно, це не є правилом, адже імітаційні моделі взагалі використовуються саме для експериментальних цілей, але безсумнівно те, що імітаційне моделювання — це технологічний процес, який проходить безліч стадій, вимагаючи від фахівців великих розумових і часових витрат.

Питання доцільності використання імітаційного моделювання розглядалось протягом багатьох років безліччю дослідників – від Ф. Мартина [38] до В. Келтона [18] та ін. Проаналізувавши ряд праць, можна зробити такі висновки.

1. Імітаційне моделювання дає змогу досліджувати внутрішні взаємодії у складних системах або підсистемах у межах складної системи, а також експериментувати з ними.
2. Моделюючи інформаційні, організаційні впливи і впливи зовнішнього середовища, можна оцінити ефекти цих впливів на поведінку (функціонування) системи.
3. На основі знань, отриманих під час проектування імітаційної моделі, можна визначити способи вдосконалення системи, яка моделюється.
4. Змінюючи вхідні дані під час моделювання і спостерігаючи за вихідними даними, можна виявити, які змінні найбільш важливі та як вони взаємодіють.
5. Імітаційне моделювання можна використовувати як метод для поліпшення рішень, отриманих під час аналітичного аналізу, а також для перевірки аналітичних рішень.
6. Імітаційне моделювання можна використовувати для проведення експериментів з новими проектами або стратегіями їх упровадження, щоб заздалегідь спрогнозувати результати.
7. Імітаційне моделювання можна застосовувати для визначення вимог, яким має відповідати пристрій або система.
8. Імітаційні моделі можна використовувати для навчання операторів складних технологічних процесів без зайвих затрат на придбання обладнання, яке може пошкоджуватись, і запобігаючи нещасним випадкам.
9. Для імітаційного моделювання можна використовувати засоби анімації, які дають змогу спостерігати за операціями, що моделюються.
10. Сучасне виробництво настільки складне, що взаємозв'язки в ньому можна інтерпретувати тільки шляхом проведення імітаційного моделювання.

Аналізуючи праці Р. Шенона [67] і Дж. Банкса [72], можна визначити ситуації, коли провадити імітаційне моделювання не варто, а саме:

- проблему можна вирішити шляхом логічного аналізу ситуації;
- проблему можна розв'язати аналітичними методами, наприклад за допомогою теорії СМО;
- результати можна отримати шляхом проведення прямих експериментів з об'єктом без втручання в технологічний процес, наприклад за допомогою хронометражу на робочих місцях;
- для розроблення імітаційного проекту за визначений строк немає достатньої кількості ресурсів;
- не можна отримати необхідні вхідні дані (імітаційне моделювання потребує великої кількості різноманітних даних, які досить важко збирати, більш того, вони можуть бути просто недоступними);

- менеджери організації, яка замовляє проект, бажають отримати забагато від імітаційного моделювання і дуже швидко;
- поведінка (режими функціонування) модельованої системи дуже складна або невизначена.

5.2. Методи проектування імітаційних моделей

Перш ніж розпочати побудову моделі, потрібно мати певну схему її проектування, за якою визначають основні принципи і методи розроблення імітаційної моделі. Сукупність правил виявлення та застосування системних принципів і методів визначає методологію проектування. Її можна розглядати на різних рівнях деталізації залежно від вибраних засобів розроблення програмних реалізацій імітаційної моделі. За допомогою вибраних програмних засобів визначають і можливі методи їх застосування. Так, наприклад, вибір за основу імітаційної моделі мереж СМО або Петрі заздалегідь визначає метод побудови її у вигляді формальних схем цих мереж.

Для створення імітаційних програм на рівні мовних засобів побудови моделей потрібно розробити алгоритми імітації, які можна подати у вигляді наборів типових обчислювальних схем. Під обчислювальною схемою імітаційного алгоритму розуміють спосіб його організації, який дає змогу відтворити в модельному часі динаміку функціонування системи [18].

Отже, перш ніж розпочати проектування імітаційної моделі, необхідно вибрати засоби програмування. Однак існують загальні методи побудови програмних реалізацій та проектування імітаційних моделей, які не залежать від вибраних програмних засобів.

5.2.1. Варіантний метод

Під час проектування імітаційної моделі варіантний метод є найпростішим та широко застосовуваним. Проектувальник послідовно крок за кроком створює імітаційну модель, опираючись тільки на свій досвід та інтуїцію. В процесі проектування розглядаються кілька варіантів кожної частини модельованої системи для її відображення в імітаційній підмоделі. Найдоцільніший варіант вибирається з урахуванням рішень, прийнятих відносно інших частин модельованої системи. Це так звана послідовна схема проектування, згідно з якою вибір варіанта імітаційної моделі є суб'єктивним і залежить від рівня знань проектувальника про систему.

Застосування варіантного методу рідко приводить до допустимих проектних рішень і не відповідає загальній схемі системного аналізу імітаційного моделювання. Така схема передбачає виконання ітераційної процедури, під час якої проектувальник не один раз повертається до вже розроблених частин імітаційної моделі і коригує їх, доки не буде впевненим, що модель відповідає цілям моделювання, або не відмовиться від неї.

5.2.2. Ітераційний метод

Суть цього методу полягає в тому, що шляхом багатьох ітерацій спроектована спочатку імітаційна модель перетворюється в таку, яка відповідає цілям моделювання. Цей метод є методом «проб і помилок», що передбачає послідовні циклічні зміни, у результаті чого отримують модель, яка задовольняє вимогам точності та адекватності. Циклічний ітераційний метод проектування потребує розгляду послідовності процедур прийняття рішень у процесі проектування. Крім того, весь хід проектування та остаточний результат значною мірою залежать від вибору початкової імітаційної моделі. Загальну схему такого проектування зображено на рис. 5.1.

Рис. 5.1. Схема циклічного ітераційного проектування

Основна проблема в разі застосування як ітераційного, так і варіантного методу проектування полягає у виборі початкового варіанта моделі. Через те що вже під час формулювання проблеми та в процесі змістовної постановки задачі висуваються вимоги до моделі, визначаються вхідні та вихідні дані, проектувальник повинен вибирати початкову модель, використовуючи метод аналогії, який базується на знанні характеристик компонентів системи, технологічних засобів і прийнятих рішень у подібних умовах. Вибір вихідної імітаційної моделі дуже впливає на результати проектування та може зробити його неможливим або занадто дорогим. Визначення рівнів точності, достовірності й правильності вибраної імітаційної моделі є самостійною проблемою моделювання, яку необхідно вирішувати під час розроблення моделі (див. розділ 5.9).

Методи внесення змін у модель базуються на принципі напрямленого дослідження. Для його застосування можна побудувати в просторі параметрів імітаційної моделі гіперповерхню її показників точності та оптимізувати або хоча б поліпшити ці показники. Сама ж процедура внесення змін у варіант моделі звичайно потребує перевірки гіпотез, які формулюють з огляду на результати проектування попередніх моделей.

Якщо результати порівняння моделі і реальної системи незадовільні, то перш ніж вносити зміни в модель, необхідно сформулювати ряд гіпотез, за допомогою

яких можна визначити причину невідповідності. Гіпотези доцільно формулювати для кількох рівнів представлення імітаційної моделі [60]:

- опису структури;
- алгоритмів поведінки;
- параметрів і вхідних даних.

Вибір рівня, на якому коригуватиметься модель та локалізуватимуться причини невідповідності, є скоріше мистецтвом, ніж наукою, і успішний результат залежить від досвіду, знань та інтуїції проектувальника. Пошук причин невідповідності потрібно починати на рівні вхідних даних, для чого оцінюють чутливість моделі до їхніх змін. Якщо виявилось, що незначна зміна вхідних даних спричиняє значну зміну вихідних, то необхідно уточнити вхідні дані для моделі і (або) локалізувати блоки моделі, на які найбільше впливають ці вхідні дані. Виявлення причин такої сильної залежності може потребувати зміни структури імітаційної моделі шляхом заміни окремих блоків моделі на більш деталізовані, що, у свою чергу, спричинить зміну внутрішніх параметрів моделі та алгоритмів функціонування. Отже, у цьому разі рівні, на яких вносяться зміни в імітаційну модель, є взаємопов'язаними.

Параметричне налагодження імітаційної моделі вимагає пошуку найкращих (оптимальних) параметрів, при яких ступінь невідповідності між моделлю та системою буде мінімальним. Це типове завдання оптимізації параметрів моделі.

Алгоритми поведінки моделі можуть змінюватись локально, для окремих блоків моделі, або для моделі в цілому. Такі зміни вимагають більш детального вивчення поведінки модельованої системи і можуть змінити рівень деталізації в моделі.

Змінити структуру моделі складніше, ніж налагодити параметри моделі, бо це може спричинити зміну алгоритмів поведінки, параметрів і вхідних даних моделі. Таку перебудову моделі можна починати тільки тоді, коли всі інші можливості вичерпано. Перебудова структури моделі може призвести до глобальних змін імітаційної моделі та її заміни новою. Тому перш ніж змінювати структуру моделі, необхідно перевірити всі гіпотези щодо витрат, які потрібні для зміни моделі. Починати перевірку слід з гіпотези, яка вимагає мінімальних витрат, а отже, і мінімальних змін імітаційної моделі.

5.2.3. Ієрархічні методи

Незалежно від того, який метод використовується – варіантний чи ітераційний, – існують два принципово відмінних підходи до проектування імітаційних моделей. Згідно з першим підходом проектування здійснюється за схемою згори вниз (так зване ієрархічне, багаторівневе або *низхідне проектування*), згідно з другим – знизу догори (*висхідне проектування*).

Наявність цих двох підходів пов'язана з формальною теорією структур систем. Перший підхід передбачає розподіл системи на підсистеми з дотриманням принципу цілісності системи та називається декомпозицією. Другий підхід з позиції розгляду структури системи є оберненим до першого і називається композицією. Він передбачає розгляд структури системи з метою створення моделі, який починають з її елементів та підсистем, а потім переходять до системи в цілому.

Низхідне проектування

В основі методів низхідного проектування імітаційних моделей лежить принцип послідовної деталізації, або декомпозиції. Він полягає у поступовому уточнюванні абстрактного опису системи, у процесі якого на кожному етапі побудови моделі задається певний рівень деталізації відображення системи. Отже, в імітаційній моделі один і той самий компонент системи може бути описаний з різним рівнем деталізації. Під час переходу від одного рівня деталізації до іншого потрібно обов'язково перевіряти, чи задовольняє модель функціональним вимогам.

На першому етапі проектування будується найзагальніша однорівнева імітаційна модель системи, за допомогою якої оцінюються лише основні показники її роботи. На наступному етапі деякі блоки моделі описують більш детально. У такий спосіб під час переходу від вищого рівня опису кожного з блоків моделі до нижчого можна досягти більшої точності та адекватності моделі системи в цілому. Даний підхід дозволяє на кожному етапі проектування порівнювати різні варіанти моделі та оцінювати вплив результатів декомпозиції на вихідні параметри системи.

У процесі побудови імітаційної моделі під час переходу з одного рівня опису на інший слід дотримуватись одного з головних принципів декомпозиції ієрархічних систем, який полягає у необхідності ущільнення інформації та зменшення тривалості роботи блоків моделі у разі переходу з одного рівня деталізації на інший. Згідно з цим принципом обсяг інформації, яка передається з рівня більш деталізованого опису моделі на рівень менш деталізованого, має бути меншим. Крім того, час роботи блока на рівні з більшою деталізацією повинен бути меншим, ніж час роботи блока на рівні з меншою деталізацією.

З прагматичного погляду такий перехід на нижчий рівень опису моделі може здійснюватись шляхом заміни блока моделі вищого рівня низкою звернень до підпрограм, функцій або процедур, які докладніше відображають цей блок для нижчого рівня. Щоб побудувати таку програму моделювання, потрібно уніфікувати процес передавання параметрів від одного програмного блока до іншого. Це дає змогу організувати взаємодію блоків моделі, що мають різні рівні деталізації, і легко замінювати один блок на інший, більш детально описаний.

Такий підхід до проектування і програмної реалізації імітаційної моделі передбачає застосування принципів об'єктного та низхідного проектування програм. Для впровадження такого підходу найбільш придатними є об'єктно-орієнтовані мови моделювання і програмування з використанням ієрархії класів об'єктів. В класи об'єднують об'єкти з однаковими характеристиками, діями та поведінкою. Властивості та поведінка, притаманні об'єктам, визначаються в методах. У об'єктно-орієнтованих мовах і пакетах імітаційного моделювання обмін між класами об'єктів різних рівнів здійснюється за допомогою транзакцій або повідомлень, які можуть передавати методи й властивості від одних об'єктів класу до інших об'єктів класів.

Під час переходу від одного рівня деталізації до іншого потрібно обов'язково перевіряти, чи задовольняє модель функціональним вимогам, які пов'язані з принципами проектування ієрархічних систем. Необхідно провести аналіз кожної функції моделі і переконатись у тому, що вона знайшла своє відображення у формальному

описі системи. Аналіз функцій моделі провадиться з врахуванням цілей моделювання і потребує детального описування роботи всіх її елементів на кожному рівні деталізації.

Висхідне проектування

Загальну схему висхідного проектування імітаційних моделей засновано на поступовому відображенні елементів системи в моделі, починаючи з найнижчого рівня системи з наступним переходом до вищого. Такий підхід має істотний недолік, пов'язаний з тим, що, розглядаючи окремі елементи системи та намагаючись відобразити їх якомога детальніше в моделі, проектувальник може не бачити систему в цілому. Це може призвести до того, що під час побудови моделі найвищого рівня імітаційна модель може виявитись функціонально неповною, бо не були враховані взаємозв'язки між різними рівнями системи. Для усунення цього недоліку потрібно повертатись до моделей нижчих рівнів, що може перетворити проектування на малоефективний та довготривалий процес. Більш того, функціонально повна імітаційна модель усієї системи може бути так і не побудована.

Під час як низхідного, так і висхідного проектування з метою зменшення розмірності імітаційної моделі однотипні блоки можуть об'єднуватись у класи, причому кожний блок певного класу може мати свій алгоритм поведінки, відмінний від інших елементів. Найбільшої ефективності під час об'єднання елементів у класи можна досягти, застосовуючи низхідний метод проектування імітаційних моделей.

Отже, за результатами системного аналізу найбільш ефективним методом проектування імітаційних моделей є такий, який поєднує в собі в певній пропорції низхідне та ітераційне проектування. Розроблення інструментальних засобів проектування імітаційних моделей з використанням комп'ютерів привело до утворення автоматизованих систем проектування, в яких використовується метод інтерактивного проектування за участі людини і комп'ютера.

Як було зазначено в розділі 1.10, основна цінність імітаційного моделювання полягає в тому, що воно ґрунтується на методології системного аналізу і дає змогу досліджувати проектовану або аналізовану систему з використанням технології операційного дослідження, яка включає такі взаємопов'язані етапи.

1. Формулювання проблеми і змістовна постановка задачі.
2. Розроблення концептуальної моделі.
3. Розроблення і програмна реалізація імітаційної моделі.
4. Перевірка правильності та достовірності моделі.
5. Організація та планування проведення експериментів.
6. Прийняття рішень за результатами моделювання.

Завдяки цьому імітаційне моделювання можна застосовувати як універсальний підхід до прийняття рішень в умовах невизначеності та до врахування в моделях факторів, які важко формалізуються. Слід мати на увазі, що реалізація імітаційної моделі та проведення експериментів на ній є трудомістким, дорогим процесом. Це вимагає значних витрат комп'ютерного часу, тому імітаційне моделювання слід використовувати тільки тоді, коли розроблення інших видів моде-

лей не дає задовільних результатів. Тому вважають, що імітаційне моделювання є «силовим заходом».

5.3. Формулювання проблеми та змістовна постановка задачі

Типовими об'єктами моделювання є комп'ютерні, виробничі, транспортні, фінансові, складські та інші логістичні системи, які в більшості випадків застосовуються для вирішення задач проектування, модернізації, прогнозування та прийняття рішень. Імітаційне моделювання використовується також під час розроблення забезпечувальних і функціональних підсистем різних комп'ютеризованих систем керування і пов'язане, як звичайно, з методами аналізу та синтезу виробничих процесів, організацією та плануванням виробництва, системами збирання, обробки та відображення інформації. Імітаційне моделювання доцільно провадити, наприклад, для виконання таких завдань:

- вибір та обґрунтування технологічного маршруту виготовлення виробів;
- оцінювання страхових запасів на дільницях комплектації складального виробництва;
- дослідження роботи автоматизованого складу;
- вибір складу технологічного устаткування на дільниці за критерієм мінімізації часу обробки;
- вибір організаційної структури керування цехом підприємства;
- аналіз роботи системи збирання, передавання та обробки інформації;
- аналіз часу доступу користувачів до інформаційної бази;
- вибір комплексу технічних засобів для оброблення інформації в організації або на підприємстві.

З наведеного вище переліку завдань видно, що одна й та сама імітаційна модель має будуватись як багатоцільова, яка дає змогу розв'язувати кілька різних завдань, тобто вирішувати деяку проблему. *Проблема* відрізняється від завдання тим, що точні методи її розв'язування невідомі. Проблема завжди є комплексною і складається з кількох завдань. Тому, формулюючи проблему, для розв'язання якої розробляється модель, потрібно в першу чергу визначити *цілі моделювання*, потім обстежити об'єкт моделювання (систему або процес), визначити межі, в яких провадитиметься дослідження. На цьому етапі моделювання широко залучаються фахівці, що мають досвід роботи з експлуатації системи і виступають експертами з розробки змістовної постановки задачі.

Після завершення цього етапу на змістовному рівні описуються основні характеристики системи, вхідні та вихідні змінні, їх взаємозв'язок, зовнішні впливи на систему, визначаються основні критерії функціонування системи та обмеження. Подальше уточнення та формалізацію моделі здійснюють на етапі створення концептуальної моделі.

5.4. Розроблення концептуальної моделі

Концептуальною називається абстрактна модель, яка виявляє причинно-наслідкові зв'язки, властиві досліджуваному об'єкту в межах, визначених цілями дослідження. Це формальний опис об'єкта моделювання, який відображає концепцію (погляд дослідника на проблему). Ця модель існує в уяві розробника, тобто вона суб'єктивна за своєю природою і відображає загальні властивості та закономірності у світі об'єктів.

Концептом називають деяку структуру, що необхідна для визначення об'єктів. Останні можна визначити через перелік їх атрибутів (властивостей). З позиції аналізу системи, яка моделюється, – це найважливіший етап, бо неправильно сформульована концепція призведе до побудови неправильної моделі.

Є інше визначення концептуальної моделі (термінологічний словник, який розроблено Міністерством оборони США). *Концептуальна модель* – це формулювання змістовного і внутрішнього зображення процесу або системи, яке поєднує концепцію користувача і розробника моделі. Вона включає в явному вигляді логіку, алгоритми, припущення й обмеження стосовно моделі.

Під час розроблення концептуальної моделі необхідно:

- визначити цілі моделювання;
- сформулювати цільові функції (критерії якості) системи, яка моделюється;
- вибрати ступінь деталізації зображення моделі;
- описати вихідні, вхідні змінні і параметри моделі;
- подати функціональні залежності, які описують поведінку змінних і параметрів;
- описати обмеження на можливі зміни величин;
- розробити структурну схему концептуальної моделі та скласти опис її функціонування.

Розроблення концептуальної моделі починається зі складання змістовного опису процесу функціонування об'єкта відповідно до принципів системного аналізу. З урахуванням зазначених цілей та обмежень формулюються критерії функціонування системи або цільові функції, визначаються структура системи, вхідні, вихідні змінні, зовнішні впливи, внутрішні параметри системи, функціональні залежності, які описують змінні і параметри, обмеження на змінні. Якщо формулювати зміст цього етапу з позицій теорії керування, то фактично вирішується завдання структурної та параметричної ідентифікації в широкому розумінні. Клас і структура математичної моделі вибираються на основі результатів теоретичного аналізу об'єкта з урахуванням загальних закономірностей процесів, які відбуваються в системі, або на підставі апріорної інформації. Математичною моделлю може бути і алгоритм.

Під час побудови концептуальної моделі об'єкта суттєвим є виділення та опис станів об'єкта. *Стан системи* визначається неперервними або дискретними значеннями характеристик елементів системи. Важливість поняття стану під час моделювання систем і керування ними полягає у забезпеченні можливості пов'язати

з кожною вхідною змінною єдину вихідну змінну, використовуючи стан системи як параметр. Отже, динамічну поведінку системи задають шляхом описування переходів її з одного стану до іншого в просторі станів.

Стани системи можуть змінюватись безперервно або в дискретні моменти часу, що, у свою чергу, визначає клас неперервних або дискретних математичних моделей, які використовуються для опису поведінки системи. Імітаційне моделювання, по суті, полягає у динамічному відображенні станів системи протягом певного проміжку часу.

5.4.1. Вибір ступеня деталізації опису об'єкта моделювання

Під час імітаційного моделювання важливим є визначення ступеня точності опису реального процесу для отримання ймовірної інформації шляхом моделювання, тобто вибір необхідного рівня деталізації опису процесів функціонування системи. Цей рівень залежить від цілей моделювання, заданих обмежень і можливості отримати вхідні дані із потрібною точністю. Більш детальна модель буде точнішою, однак вона буде складнішою і дорожчою, бо для її побудови, перевірки, документування та використання необхідні більші затрати. Водночас детальніша модель – точніша і тому буде частіше застосовуватись. Отже, доводиться встановлювати рівновагу між вимогами дослідження та вартістю моделі. На рис. 5.2 [67] видно, як змінюються показники моделі залежно від обраної точності.

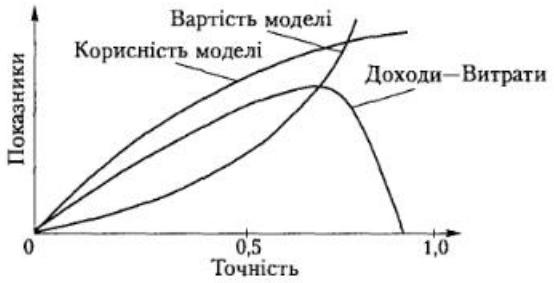


Рис. 5.2. Залежність показників моделі від обраної точності

Для оцінювання рівня деталізації моделі можна використовувати *показники ступеня деталізації*. Одним з них є відношення реального часу до модельного, тобто співвідношення між часом моделювання імітаційної моделі із заданим набором даних і часом роботи реальної системи з тим самим набором даних. Іншим важливішим показником є часова розрізнювальна здатність моделі, яка визначається як найкоротший інтервал часу між послідовними подіями, що фіксуються під час моделювання, тобто найменша ідентифікована порція інформації в моделі.

Рівень деталізації має бути неоднаковим для всієї моделі. Часто він підвищується для тих частин, для яких передбачається велика залежність точності результатів моделювання від вибраної точності опису цих частин.

Від вибраного рівня деталізації моделі залежить і її *стійкість*, тобто здатність точно відтворювати поведінку системи, конфігурація якої або вхідні дані для котрої не було передбачено на етапі побудови моделі. Стійкість, як звичайно, зростає

зі збільшенням ступеня деталізації моделі. Наприклад, під час побудови моделі комп'ютера з метою визначення середнього часу виконання програм, які надходять до нього, можна використати СМО з одним пристроєм. Ця модель буде відтворювати вхідний потік програм із заданим розподілом імовірності часу надходження їх до комп'ютера. Час виконання цих програм буде заданим деякою функцією розподілу ймовірностей. Ця модель буде дуже узагальненою, але вона дасть змогу знайти середній час виконання програм комп'ютером.

Більш детальнішою буде модель, в якій програми, що надходять до комп'ютера, будуть вимагати під час знаходження в комп'ютері деяких ресурсів (оперативної пам'яті, зовнішніх пристроїв, процесорного часу, файлів, програмних модулів тощо). Споживання цих ресурсів програмами теж буде відтворювати процес виконання програм комп'ютером. Ця модель потребує більш детальних даних про роботу цих програм, тобто робочого навантаження на комп'ютер, і буде точніше відтворювати його роботу. Таку модель можна побудувати як мережу СМО, але витрати на її побудову будуть більшими.

Найбільш деталізованою буде модель, якщо програми, які надходять до комп'ютера, подати у вигляді комп'ютерних команд. У цьому разі модель комп'ютера має відтворювати всі його команди, тобто повністю відображати роботу комп'ютера на рівні мікросхем. Ця модель буде найбільш точною та найбільш дорогою, але вона дасть змогу відповісти не тільки на запитання про середній час виконання програм у комп'ютері, а й на ті, які пов'язані з роботою комп'ютера. Наприклад, як зміняться параметри роботи комп'ютера в разі заміни процесора на інший з іншою системою команд.

Отже, ступінь деталізації системи під час побудови моделі має визначатись на основі принципів доцільності, тобто з урахуванням вигоди від використання моделі та затрат на її створення.

5.4.2. Опис змінних моделі

На етапі розроблення концептуальної моделі визначаються вимоги до вхідних даних. Для цього можуть використовуватись різні методи вимірювань, якщо модельована система реально існує. Частину даних можна отримати з технічної та конструкторської документації системи, офіційних звітів, статистичних збірників, довідників. Під час моделювання виробничих систем важливими джерелами вхідних даних, крім фінансової, технічної і технологічної документації, є також анкетування.

Вхідні та внутрішні змінні вибираються відповідно до ступеня деталізації різних частин моделі з урахуванням можливих змін і доступності первинних даних. У разі побудови стохастичної імітаційної моделі доводиться приймати рішення, які далі доцільно використовувати в моделі – емпіричні, теоретико-ймовірнісні або ті, що були отримані на основі результатів статистичного аналізу. Під час вибору способу задання стохастичних змінних потрібно враховувати такі особливості.

1. Використання необроблених емпіричних даних дає змогу імітувати тільки процеси і події, що вже відбулися, тому важливо знати, що даний розподіл буде незмінним у часі.

2. Застосування випадкових розподілів імовірностей визначає закономірності модельованих процесів, а методи їх моделювання ефективніше, ніж використання табличних значень, з погляду на економію ресурсів комп'ютера.

У подальшому під час моделювання надзвичайно важливо провести випробування моделі на чутливість результатів моделювання до зміни видів розподілів та їх параметрів, тобто до вхідних даних.

Щоб отримати вхідні дані, необхідні для моделювання обчислювальних комплексів, які входять до складу технічного забезпечення інформаційних систем, використовують спеціальні програмні засоби вимірювання (тести), які дають змогу визначити продуктивність комп'ютерних систем. Для технічних і виробничих систем використовують зазвичай засоби хронометражу виконання операцій або технологічні карти.

Найскладнішим є збирання статистичних даних про зовнішні впливи на систему. Це пов'язано з великою трудомісткістю та вимагає значних витрат часу, особливо коли деякі події трапляються рідко.

Для новостворюваних систем вихідні дані можуть взагалі не існувати. У цьому разі для отримання даних застосовують експертні оцінки, які одержують шляхом опитування групи експертів і фахівців у даній сфері.

До отримуваних вхідних даних слід підходити з великою відповідальністю, оскільки від них залежать майбутні результати моделювання. Використання неточних даних призводить до неправильних результатів моделювання.

Слід відзначити, що побудова концептуальної моделі – це ітераційний процес, і кількість ітерацій залежить від складності моделі системи. Спочатку, як звичайно, будується узагальнена модель, яка відповідає рівню знань дослідника про систему. Накопичуючи знання щодо системи, розробник будує більш детальну модель доти, доки не буде впевнений, що подальша деталізація моделі недоцільна з погляду поставлених задач або витрат на її створення. Потім концептуальна модель обговорюється з користувачем. Це дозволяє виявити допущені помилки та провести необхідне коригування моделі.

5.4.3. Формалізоване зображення концептуальної моделі

Під формалізацією розуміють такий опис моделі, який передбачає використання математичних методів дослідження, тобто на завершення цього етапу розробляється логіко-математичний опис модельованої системи з урахуванням динаміки її функціонування. Таким чином, *формалізація* – це відображення системи чи процесу в точних поняттях. Частини моделі, які можна описати в математичному вигляді, подають як аналітичні залежності. Інші частини моделі являють собою детальний словесний опис або алгоритм їх функціонування.

Зобразити структуру концептуальної моделі можна за допомогою діаграми (графу) станів системи. Простий приклад такої діаграми, що відображає роботу центрального процесора, наведено на рис. 5.3, *а*. На діаграмі можна зазначити причини зміни станів елементів системи. (Див., наприклад, спрощену діаграму станів верстата, зображену на рис. 5.3, *б*.)

Рис. 5.3. Діаграми станів процесора (*а*) і верстата (*б*)

Причини зміни стану можуть задаватись аналітично як функції часу. Так, час напрацювання верстата на відмову можна задати у вигляді функції розподілу ймовірностей.

Для задання концептуальної структури можуть використовуватись формальні математичні моделі, такі як мережі Петрі, стохастичні мережі СМО, потокові діаграми та ін. Слід зазначити, що вже на цьому етапі на вибір структури може суттєво впливати вибір програмних засобів реалізації імітаційної моделі. Сучасні програмні засоби імітаційного моделювання мають різні візуальні компоненти, за допомогою яких можна побудувати концептуальну модель.

Побудова концептуальної моделі – найвідповідальніший етап моделювання. Неправильна концепція, покладена в основу моделі, неправильні припущення про взаємозв'язки змінних і параметрів призводять до того, що виконання подальших етапів побудови імітаційної моделі виявляється безглуздим і часто спричиняє невиправдані затрати.

Після побудови концептуальної моделі перевіряють її відповідність об'єкту моделювання, використовуючи метод оберненого перетворення. Розглядається побудована модель, здійснюється перехід до прийнятих припущень, апроксимацій та спрощень і повернення до реальних процесів і явищ системи, що моделюється.

Побудована концептуальна модель подається на обговорення експертам, які повинні визначити її відповідність цілям моделювання з погляду вирішення поставленої проблеми. У деяких випадках після створення концептуальної моделі і аналізу її функціонування фахівці можуть отримати достатню інформацію для суттєвого поліпшення роботи системи, яка моделюється, без побудови імітаційної моделі.

5.5. Вибір засобів реалізації імітаційної моделі

На цьому етапі виконуються роботи, пов'язані з підготовкою та реалізацією імітаційної моделі на комп'ютері. Розробляється логічна схема моделі, яка перетворюється потім у програму. Подальша формалізація концептуальної моделі відбувається на етапі розроблення структури імітаційної моделі, але слід мати на увазі, що відображення структури моделі залежить від обраних засобів програмування.

Програмна реалізація імітаційної моделі може бути створена за допомогою:

- ✦ алгоритмічних мов загального призначення;
- ✦ спеціалізованих мов моделювання;
- ✦ пакетів прикладних програм для моделювання;
- ✦ засобів автоматизації програмування імітаційних моделей;
- ✦ діалогових і візуальних систем моделювання;
- ✦ інтелектуальних систем моделювання.

Структуру моделі можна зобразити з використанням елементів програмних засобів створення імітаційної моделі. Тому, перш ніж розпочати розроблення структурної схеми імітаційної моделі, необхідно вибрати мову програмування, якою буде реалізовано модель.

Мову, якою буде здійснено програмну реалізацію імітаційної моделі, слід вибирати залежно від складності імітаційної моделі та типу наявного комп'ютера. Нині існує багато мов моделювання, орієнтованих як на суперкомп'ютери, так і на персональні комп'ютери. Спочатку потрібно визначити, як буде реалізовано імітаційну модель алгоритмічною мовою програмування загального призначення, об'єктно-орієнтованою мовою з візуальним середовищем або з використанням спеціалізованих засобів моделювання.

Щоб надати перевагу одному з вищевказаних засобів, необхідно спочатку визначити затрати на реалізацію моделі, враховуючи такі рекомендації.

Для реалізації простої імітаційної моделі можна використати алгоритмічну мову загального призначення, що дасть змогу отримати більш швидкодіючу модель.

У разі реалізації складної імітаційної моделі, що має велику кількість різних компонентів, перевагу належить віддавати спеціалізованим засобам моделювання. У випадку використання алгоритмічних мов з метою уніфікації програмних модулів і принципів структурного та модульного програмування, необхідно розробити засоби:

- ✦ керування процесом моделювання в модельному часі;
- ✦ збирання статистичних даних про роботу моделі;
- ✦ генерування випадкових величин з різними законами розподілів імовірностей для моделювання стохастичних впливів;
- ✦ діагностики помилок під час виконання програми моделювання.

Ці засоби існують у будь-якій мові моделювання або пакеті прикладних програм, орієнтованому на моделювання.

Перш за все на початковий вибір засобу моделювання впливає наявність його в даному комп'ютері. Під час попереднього вибору засобу необхідно визначити, чи досконало написано настанову та інструкції для користувача, чи забезпечує транслятор достовірну діагностику помилок, чи можливе швидке засвоєння цього засобу.

На наступному етапі потрібно оцінити можливості побудови програмної реалізації імітаційної моделі з допомогою цього засобу та проведення експериментів з моделлю. Для цього необхідно відповісти на такі запитання:

- ✦ Які існують засоби генерування випадкових чисел і змінних?
- ✦ Як здійснюється збирання статистичних даних моделювання?

- Які можливості засобу щодо налагодження моделі?
- Наскільки засіб відповідає можливості відображення структури модельованої системи, яка моделюється?
- Які можливості засобу щодо зміни структури моделі?
- Наскільки просто вносити зміни в програмну реалізацію моделі?
- Чи легко обробляти отриманий статистичний матеріал за результатами прогонів моделі?
- Чи існує інтерфейс з іншими програмними засобами (графічними, анімаційними, статистичними, оптимізаційними пакетами)?
- Наскільки зручний інтерфейс між користувачем і програмою?
- Чи дає засіб можливість формувати звіт про роботу моделі в зручному для користувача вигляді?
- Чи можна використовувати засоби планування проведення експериментів та оптимізації для прийняття рішень?

Із безлічі можливих засобів моделювання перевагу належить віддавати тим засобам, що добре зарекомендували себе як перевірені та використання яких потребує мінімальних витрат на створення моделі та експериментування з нею.

5.6. Розроблення структурної схеми імітаційної моделі та опису її функціонування

На цьому етапі об'єкт моделювання остаточно формалізується у вигляді абстрактної системи. Для цього необхідно описати структуру системи та сукупність математичних описів усіх її елементів, зовнішніх впливів і взаємодій між усіма елементами в процесі функціонування системи.

Структура системи визначається з огляду на побудовану концептуальну модель і вибрані засоби моделювання. У разі реалізації імітаційної моделі алгоритмічною мовою загального призначення на структуру імітаційної моделі істотно впливає вибраний підхід до реалізації імітаційного алгоритму. У разі вибору об'єктно-орієнтованої мови описують класи та підкласи для статичних і динамічних об'єктів, задають інтерфейси, методи класів та властивості об'єктів.

Якщо модель буде реалізовано мовою моделювання GPSS [68], то структура моделі подається у вигляді блок-схеми, що складається з блоків різних типів. Набір блоків у блок-схемі визначає набір операторів мови, які описують структуру системи, що моделюється та логіку її функціонування. У такій схемі блоки відображають виконувані над транзактами операції, а стрілки між блоками — маршрути руху транзактів.

Якщо імітаційною моделлю системи є СМО, то її структуру можна зобразити у вигляді схеми, на якій є генератори вимог, що надходять до системи, буферів або черг, пристроїв для обслуговування, які реалізують затримку вимог у системі. Черги до якого-небудь ресурсу утворюються через його зайнятість. На цій схемі

може бути зображено програмні блоки, які реалізують необхідні функції в реальній системі. Якщо імітаційна модель будується як мережа Петрі, то структура мережі зображується у вигляді орієнтованого графу.

Якщо програмна реалізація імітаційної моделі здійснюється мовою, орієнтованою на процеси, то структуру моделі можна зобразити у вигляді ресурсів і шляхів проходження потоків повідомлень через ці ресурси. Структурну схему можна побудувати детальніше, якщо виділити всі типи подій для потоків повідомлень та дії, які належать до ресурсів. Фрагмент такої схеми зображено на рис. 5.4.

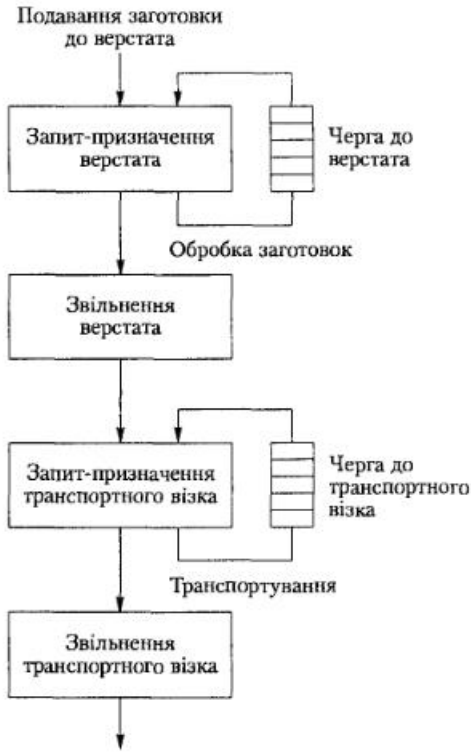


Рис. 5.4. Фрагмент структури імітаційної моделі технологічного обладнання

Стрілками позначено дії, пов'язані з потоком заготовок (повідомлень). Основними подіями в такій схемі є запит, призначення та звільнення ресурсів, таких як верстат і транспортний візок. Якщо ресурс уже зайнятий, до нього утворюється черга заготовок. На цій самій схемі може бути позначено умови зміни потоків повідомлень. Таку структурну схему імітаційної моделі називають діаграмою або картою подій. Для спрощення діаграми подій за значної кількості ресурсів однотипні ресурси доцільно групувати та визначати як узагальнений ресурс або клас.

Побудова діаграми подій – ефективний проміжний етап перетворення концептуальної моделі в програму імітаційної моделі, який дає змогу здебільшого пропускати етап алгоритмізації моделі та переходити безпосередньо до розроблення програми.

Для розробленої схеми імітаційної моделі описують її функціонування, появу повідомлень у моделі та їх рух по моделі з урахуванням прийнятих алгоритмів.

В описі слід відобразити закони або алгоритми появи повідомлень у моделі, правила входження в черги і виходу з них, алгоритми передавання повідомлень з одного блока моделі в інший, закони або алгоритми, за якими оброблюються (затримуються) повідомлення в блоках обслуговування, а також вибрані дисципліни обслуговування повідомлень.

5.7. Програмна реалізація імітаційної моделі

Програмну реалізацію імітаційної моделі рекомендується будувати за модульним принципом. Це дає змогу удосконалювати модель за допомогою ітераційного методу, додаючи до неї модуль за модулем. У процесі налагоджування та експериментування окремі модулі може бути замінено або змінено, що не призведе до істотних змін в усій моделі.

Структура програми моделі має відповідати структурі імітаційної моделі. Така побудова програми робить її наочною та полегшує її налагоджування. Кожний модуль програми має супроводжуватись коментарем. Програмування та налагоджування моделі доцільно провадити поетапно, з наступним збільшенням програмних модулів.

Для оцінювання правильності функціонування програмної реалізації імітаційної моделі провадяться пробні експерименти (тестування моделі), в яких широко використовуються налагоджувальні засоби вибраної системи моделювання. Більшість мов моделювання має засоби, які дають змогу слідкувати за трасами руху повідомлень у моделі, завдяки чому за різних початкових умов можна переконатися в тому, що модель працює так, як було задумано. Ця перевірка дає змогу визначити, чи відповідає відтворюваний програмою процес функціонування математичній моделі об'єкта і прийнятій концепції. У разі невідповідності процесів функціонування моделі та об'єкта програма коригується. Важливо визначити також чутливість моделі до вхідних даних і функцій розподілів випадкових величин. У тому випадку, коли при незначних змінах вхідних даних вихідні дані істотно змінюються, потрібно визначити блоки, на які впливають ці вхідні дані та у разі необхідності замінити ці блоки більш деталізованими.

Якщо в програмі передбачене закінчення моделювання із досягненням заданого значення модельного часу, то за умови, що прогін налагоджений, воно має бути малим. Це необхідно для того, щоб через модель встигло пройти небагато повідомлень. Таке обмеження слід накладати і тоді, коли передбачене закінчення моделювання під час досягнення заданої кількості повідомлень на виході моделі. Обмеження потрібне, тому що, як звичайно, програміст не впевнений у тому, що модель працюватиме правильно, і невідомо, скільки машинного часу може знадобитись для її прогону.

Типова помилка під час налагодження моделі пов'язана з неузгодженістю пропускної здатності окремих елементів системи, тобто повідомлення надходять у деякі елементи моделі частіше, ніж вони встигають обслуговуватись. Через це доцільно

на деяких ділянках моделі, в яких можуть нагромаджуватись повідомлення, задавати обмежувальні умови на довжину черги. У разі виконання цих умов має видаватись повідомлення про те, що черга до певного елемента системи переповнена.

Після закінчення налагоджування функціонування програмної реалізації імітаційної моделі необхідно перевірити її працездатність в усьому діапазоні змін вхідних змінних. Усі значення змінних у моделі має бути зведено до вибраної одиниці модельного часу. Для остаточного тестування моделі на контрольних прикладах необхідно залучати тих людей, які не брали участі в програмуванні моделі, або майбутніх її користувачів. Більш детально процес тестування моделі описується під час розгляду питань про валідацію та верифікацію моделей.

5.8. Автоматизація програмування

Створення імітаційних моделей – складне завдання. Недарма у своїй праці Шенон [67] ставить програмістів, які створюють програмні реалізації імітаційних моделей, вище, ніж системних програмістів. Перші мислять просторово – часовими образами. Для спрощення процесу створення імітаційних моделей розробляються засоби автоматизації, які дають змогу не тільки вилучити проміжну ланку між аналітиком і людиною, що приймає рішення, а й під час створення моделі використовувати терміни предметної галузі, в якій працює аналітик.

5.8.1. Комп'ютерна інженерія

У 1968 році на одній з конференцій НАТО з проблем розроблення програмного забезпечення було вжито термін «Software Engineering», який перекладається як *комп'ютерна інженерія*. Так було названо нову наукову дисципліну, об'єктом дослідження якої є великі комп'ютерні системи і проблеми, що виникають під час їх створення.

У наш час продовжують розвиватись різні методи розробки складного програмного забезпечення. У рамках комп'ютерної інженерії робиться спроба визначити абстрактну систему понять цього процесу. Кожний новий підхід передбачає свою систему, яка схожа на інші, але має деякі нюанси.

У цьому розділі розглядаються відомі об'єктно-орієнтовані методи автоматизації створення програмних систем і розкриваються поняття, які використовуються під час розроблення засобів автоматизації проектування імітаційних моделей.

5.8.2. Мова SDL

Застосовувана в моделюванні мова SDL (Specification and Description Language) – це мова специфікацій та опису. Під специфікацією розуміють точне формальне визначення системи або її частини, під описом – неформальну специфікацію, яка ілюструє той або інший аспект системи. Описи використовують на початкових етапах розробки системи або для документування системи. На етапах детального проектування використовують специфікації, за якими передбачається виконання автоматичного генерування програмного коду. Той факт, що для різних етапів

розробки системи пропонується одна мова, є суттєвою перевагою SDL, оскільки в такому випадку зникає проблема семантичних розривів.

Мову SDL призначено для розробки подійно-орієнтованих розподілених систем. Ця мова почала розвиватись за сприяння міжнародного комітету ITU ще в 1976 році і є однією з найдавніших розробок в комп'ютерній інженерії. Існує два варіанти цієї мови – текстовий (SDL/PR) та графічний (SDL/GR), синтаксис яких здебільшого співпадає. Викладення основ цієї мови можна знайти в літературі [93]. Крім того, є російський варіант стислого викладення мов SDL і MSC (Message Sequence Chart) [17].

Більше десяти компаній в Європі (Telelogic, Verigol та ін.) розробляють CASE-засоби (Computer-Aided Software Engineering – автоматизоване проектування і створення програм) на основі SDL. Ці продукти використовуються багатьма великими європейськими компаніями – виробниками телекомунікаційних систем.

Крім мови SDL комітетом ITU запропоновано цілу низку стандартів на засоби розробки телекомунікаційних систем. Серед них мови високого рівня CHILL, MSC [35] (графічна мова сценаріїв). У Європі щороку проходить велика кількість конференцій, на яких обговорюються різні аспекти цих стандартів.

Мова SDL як засіб аналізу систем широко використовується в європейських телекомунікаційних стандартах. Її основними складовими є структурна модель і розширений скінченний автомат. Вони орієнтовані на здійснення специфікації подійно-орієнтованих систем, хоча мають ширше використання.

Блочний аналіз є основою структурної декомпозиції системи за допомогою мови SDL. Він передбачає зображення системи у вигляді вкладених одна в одну частин (блоків), які не містять виконуваного коду, а містять лише описи. Блоки можуть відповідати великим модулям системи або підзавданням проекту. Виконуваний код у вигляді розширеного скінченного автомату міститься лише в листках дерева цієї декомпозиції, тобто процесах, які, подібно до блоків, можна поставити у відповідність об'єктам. Тому мову SDL було успішно розширено до об'єктно-орієнтованої мови.

Існують графічні нотації, що призначені для використання на початкових етапах розробки системи, а також процес розробки програмного забезпечення на основі мови SDL. Але на сьогоднішній день ці нотації замінено мовою UML, яка потіснила також SDL. У праці [36] визнано, що SDL є мовою програмування, подібною до Java, C++ та ін. Отже, виникає проблема співіснування UML-описів і SDL-специфікацій.

5.8.3. Метод OOSE

Об'єктно-орієнтований програмний інжиніринг – OOSE (Object-Oriented Software Engineering) описано у монографії [77], яка є однією з фундаментальних праць у галузі досліджень об'єктно-орієнтованих методів розробки програмного забезпечення. Основне завдання цього підходу – наблизити комп'ютерну інженерію до типового промислового процесу, яким є, наприклад, будівництво. Основний принцип підходу – об'єктна орієнтованість, як аналізу, проектування, програмування, так і опису процесу розробки програмного забезпечення загалом.

Підхід призначено, у першу чергу, для розроблення великих систем. На рис. 5.5 зображено рівні технології комп'ютерної інженерії згідно з OOSE. На основі OOSE створено метод Objectory, реалізований у продукті компанії Objectory AB. У 1995 році, після злиття цієї компанії з корпорацією Rational Software Corp., цей метод використовувався під час створення раціонального об'єднаного підходу – RUP (Rational Unified Approach).

Рис. 5.5. Структура комп'ютерної інженерії

В OOSE пропонується компактний опис структури комп'ютерної інженерії, основою якої є поняття архітектури. Це поняття включає основні концепції і технології, визначені як об'єктно-орієнтовані, певний набір напівформальних моделей з графічними нотаціями, які надаються для опису розроблюваної системи.

Метод OOSE – лінійна послідовність кроків, тобто процедура для створення ідеальної системи «з нуля». За допомогою цього методу можна визначити, як застосовувати архітектуру для розроблення системи.

Процес розробки є розширенням методу. На відміну від методу він, по-перше, орієнтований на ітеративне розроблення програмного забезпечення (сам метод лінійний) і по-друге – адаптований до практичного застосування (метод – це ідеальна послідовність кроків).

І нарешті, інструментальні засоби – це реалізація архітектури, методу та процесу в конкретному програмному продукті – CASE-засобі, за допомогою якого відбувається розроблення системи.

Аналіз і проектування в OOSE засновані на методі випадків використання (use case approach), за допомогою яких, через побудовані для них сценарії, виділяються об'єкти. Пропонується кілька об'єктних моделей для різних стадій розробки системи і, як у SDL, блочний аналіз.

5.8.4. Метод Буча

Праця Буча [74] є класичною монографією, де висвітлюється об'єктно-орієнтований підхід до аналізу та проектування програмного забезпечення. За допомогою методу описуються об'єктна модель і візуальні засоби, а також сам процес розробки програмного забезпечення з визначенням цілей, видів діяльності та передбачуваних результатів. Крім того, розглядаються організаційні питання створення програмного забезпечення та вплив на них застосування об'єктно-орієнтованого

підходу. Метод засновано на єдиній моделі класів. Її пропонується використовувати на різних етапах розробки отримання єдиної специфікації системи, яка змінюється від однієї фази розробки до іншої.

Як основні поняття у праці [74] використовуються *методологія* і *метод*.

«*Методологія* – це набір методів, які застосовуються протягом всього життєвого циклу створення програмного забезпечення, поєднаних єдиною філософською концепцією». Такою концепцією в Буча є об'єктно-орієнтований погляд на світ.

«*Метод* – це чітко визначений процес створення набору моделей за допомогою специфікованих нотацій; ці моделі описують різні аспекти програмного забезпечення». Буч називає свій підхід методом і ділить його на три частини – *нотації*, *процес*, *прагматика*.

Основа методу – можливість розглядати розроблювану систему з різних поглядів. Результат такого розгляду називається *моделлю* системи. Буч вирізняє такі типи моделей: логічну, фізичну, статичну та динамічну. Під моделлю розуміється як спосіб бачення, так і його результати. У першому випадку використовується також термін *view*, у перекладі з англійської – *вид*.

Нотація – це графічна мова для опису моделей. Ця частина методу є формальною.

Процес – це опис цілей, видів діяльності та результатів для різних фаз об'єктно-орієнтованого аналізу та проектування. Процес не формалізується як набір процедур, а поділяється на частини, для яких описуються інтерфейсні характеристики. Буч підкреслює, що його опис процесу не є набором готових рецептів.

Прагматика в контексті методу Буча – це та специфіка об'єктно-орієнтованого підходу, яка виявляється в організаційних питаннях створення програмного забезпечення: керування проектом, персоналом, ризиками, версіями системи, конкретні програмні засоби підтримки розробки програмного забезпечення та ін. Важливість цих питань зумовлена тим, що проектування та аналіз не є строгою та формально визначеною наукою, для розв'язання значної частини проблем не вдається знайти відповідних формалізацій, і залишається лише обговорити їх на неформальному рівні. Таким чином, ця частина методу є найбільш неформальною.

5.8.5. Мова UML

У класичних працях, присвячених проблемі створення великих програмних систем на основі об'єктно-орієнтованого підходу [74, 77], автори намагалися охопити всі боки життєвого циклу розробки програмного забезпечення, не залишаючи без уваги жодного організаційного питання. Але найбільше практичне втілення мали ті частини цих праць, в яких йдеться про візуальне моделювання як один з основних засобів аналізу та проектування великих програмних систем. Було створено велику кількість спеціалізованих програмних продуктів під загальною назвою CASE- засоби, які реалізують графічні нотації різних об'єктно-орієнтованих методологій. Нарешті, хаосу в цій галузі позбулися завдяки прийняттю стандарту щодо об'єктно-орієнтованих засобів візуальної специфікації – мови UML (Unified Modeling Language). Якщо слідувати структурі методу Буча, то можна сказати,

що стандартизовано лише нотацію, а процес і прагматика в UML не увійшли, тобто було стандартизовано мову, а не способи її застосування.

Мова UML розвивається з 1994 року і є результатом злиття трьох найбільш відомих об'єктно-орієнтованих підходів: методу Буча [74], OMT [85] і OOSE [77]. У 1997 році мову UML було прийнято комітетом OMG як стандарт. Вона практично замінила собою всі інші об'єктно-орієнтовані підходи. Мова UML є грандіозною спробою розробити на основі об'єктно-орієнтованого підходу універсальну мову графічного моделювання для аналізу і проектування складних комп'ютерних систем. Вона об'єднує велику кількість різних графічних нотацій з метою впорядкування хаотичного набору графічних засобів, які використовуються під час створення програмного забезпечення. Стандартизація тут суттєво підвищує рівень розуміння між різними фахівцями, які розроблюють складну систему. Крім того, стандарт полегшує процес перенесення специфікацій, виконаних у різних CASE-пакетах.

В основному документі щодо мови UML [97] описано версію UML 1.1 1997 року. Ця праця ґрунтується на настанові з UML [84].

Нижче коротко розглядаються поняття UML, на яких засновано засоби структурної декомпозиції проекту та розроблюваної системи.

Пакет – це простір імен проекту, який складається з множини сутностей, виражених за допомогою понять і діаграм UML. Пакети можуть включати в себе інші пакети.

Модель – це тип пакету, який є певним закінченим образом системи, що описує її з певного погляду. Наприклад, для розроблюваної системи можна побудувати модель випадків використання, яка буде визначати функціональні вимоги до неї.

Погляд – це певний спосіб бачення системи, з огляду на який будується певна модель системи. Погляд включає в себе набір графічних нотацій та їх семантику. Виділяються такі погляди на систему: статичний, випадків використання, взаємодії, скінченно-автоматний, активностей, фізичний, керований.

Підсистема – це вид пакету, який описує певну частину системи, виділену в єдине ціле за реалізаційними або функціональними міркуваннями. Структура підсистеми ділиться на дві частини – декларативну та реалізаційну. Перша визначає зовнішню поведінку підсистеми і може включати в себе випадки використання, інтерфейси та ін. Реалізаційна частина описує, яким чином реалізується декларативна частина.

Підсистема аналогічна блоку SDL, але загалом система понять UML є більш загальною, ніж у SDL.

5.8.6. Методологія ROOM

Методологія ROOM (Real-Time Object-Oriented Modeling) – це об'єктно-орієнтована розробка систем реального часу. Її розвиток пов'язаний з канадською компанією ObjectTime Limited, яка на основі цієї методології випустила програмний продукт ObjectTime. Методологію було розроблено в 1992 році [98]. У 1994 році вийшла монографія [88], яка містить повний опис ROOM. Модель поведінки ROOM розглянуто в працях [94, 99]. Крім того, продовжує видаватись велика кількість статей, в яких описано різні аспекти використання цієї технології та подальший розвиток. Заслуговують на увагу праці [94, 99], в яких описано, як ROOM

«вкладається» в UML за допомогою механізму розширень UML. Матеріал, викладений у цих працях є базисом для створення мостів між програмними продуктами, які реалізують ROOM і UML.

У методології ROOM передбачено два рівні подання розроблюваної системи:

- рівень схем;
- рівень деталізації.

Виділення цих рівнів спрямоване на автоматичну кодогенерацію. Таким чином, ця методологія суттєво відрізняється від UML, де пропонуються лише погляди на систему (view), застосування яких не зовсім зрозуміле. Для рівня схем методологія ROOM пропонує набір графічних нотацій. Рівень деталізації передбачає використання мови реалізації, оскільки очевидно, що всю систему, якщо вона достатньо складна, неможливо задати специфікацією у вигляді картинок, за якими можна автоматично згенерувати працюючу програму.

Рівень схем складається з графічних нотацій, що дозволяють зобразити структуру системи (класів і об'єктів) та опис моделі її поведінки.

5.8.7. Метод RUP

Мова UML є лише мовою моделювання і способи застосування винесені з її специфікації. Корпорацією IBM Rational Corp. створено надбудову над UML під назвою RUP (Rational Unified Process) [10], яка дає змогу систематизувати процес створення програмного забезпечення на основі UML, пропонуючи використовувати певний набір програмних продуктів (головним чином, компанії Rational Corp.).

Одним із основних понять методу RUP є фаза. Фаза – це етап розробки системи. З нею, як звичайно, пов'язана проміжна або кінцева звітність до проекту. За методом RUP вирізняється така множина фаз.

1. *Початок* – фаза визначення меж проекту, оцінювання реальності його виконання (терміни, плани, кошти, люди, ризики).
2. *Деталізація* – фаза створення архітектурного прототипу системи, визначення вимоги до проекту, його вартості і терміну виконання, складання детального плану роботи.
3. *Конструювання* – фаза реалізації проекту.
4. *Передавання* – фаза передавання системи замовнику.

Цикл розроблення системи завершується після виконання останньої фази, у результаті чого з'являється нова версія системи. Після цього розроблення продовжується вже на новому рівні внаслідок висування нових вимог до системи і завершується випуском чергової версії системи.

Метод RUP передбачає інтерактивність усередині однієї фази. Як звичайно, результатом ітерації є прототип системи. Наводяться такі варіанти кількості ітерацій за фазами: [0,1,1,1], [1,2,2,1], [1,3,3,2]. Таким чином, формулюється загальне правило про кількість ітерацій усередині одного циклу: 6 ± 3.

Поняття фази є вдалою абстракцією для виділення етапів розробки системи. Це поняття узгоджує ітеративний процес розробки програмного забезпечення та лінійний порядок звітності.

Розроблення системи виконується в чотири фази за допомогою робочих процесів (workflow), які є послідовністю дій, пов'язаних загальною специфікою. Робочі процеси розподіляються за різними фазами і можуть протікати паралельно. Метод RUP передбачає виконання таких робочих процесів.

1. Бізнес-моделювання (Business Modeling).
2. Висування вимог (Requirements).
3. Аналіз і проектування (Analysis and Design).
4. Реалізація (Implementation).
5. Тестування (Testing).
6. Впровадження системи (Deployment).

Крім того вирізняють такі допоміжні процеси.

- керування проектом (Project Management);
- створеня конфігурації змін та керування ними (Configuration and Change Management);
- створення конфігурації середовища (Environment).

З наведеного вище огляду засобів CASE-технологій зрозуміло, що всі вони тією або іншою мірою можуть запроваджуватись під час розроблення складних імітаційних систем. Проте існує проблема, яку складно вирішувати за допомогою цих технологій — керування процесом моделювання в модельному часі. Спробою поєднати CASE-технології та імітаційне моделювання є розробка стандарту HLA (High Level Architecture).

5.8.8. Програмні генератори імітаційних моделей

З метою спрощення процесу створення імітаційних моделей останнім часом розробляються діалогові або інтерактивні системи моделювання, які за допомогою інтерфейсу транслюють висловлення на природній мові у програмні реалізації імітаційних моделей. Перші спроби побудови таких систем робились з використанням мови Simula, але засоби автоматичного програмування не були створені через складність реальних модельованих систем. Найбільшого успіху досягнуто під час розробки спеціалізованих систем моделювання для вузьких галузей.

Не зважаючи на те, що засоби генерування моделей орієнтовані на конкретні предметні галузі, бажано, щоб вони не враховували будь-які специфічні особливості певної галузі. Цього можна досягти завдяки уніфікації опису імітаційних моделей, який повинен базуватись на єдиній математичній основі для всіх форм зображення модельованого об'єкта: змістовного і формалізованого описів, програмної реалізації. Однак тоді спостерігається неоднорідність опису, зумовлена проблемою погодження опису елементів модельованого об'єкта у вигляді деяких

математичних співвідношень з чіткими математичними залежностями та певним способами задання зв'язків між цими елементами.

Технологія створення засобів автоматичного генерування імітаційних моделей передбачає [60]:

- вибір ефективної і коректної конструктивної множини елементів моделі;
- розробка засобів специфікації, вибору та налагодження елементів моделі;
- наявність засобів конструювання моделі із обраних елементів.

Коректний вибір конструктивної початкової множини елементів визначає клас моделей, які можуть бути реалізовані за допомогою даних засобів. Щоб максимально розширити клас моделей, потрібно обирати множину елементів, якомога ближчу до мовних засобів імітаційного моделювання. Більш того, мова, яка обирається для моделювання, має бути декларативною, що спростить розроблення інтерфейсу між системою програмування та створюваними генератором імітаційними моделями завдяки уніфікації побудови та використання мовних засобів, закладених у самій мові.

Серед мов, які задовольняють таким вимогам, слід виділити мову імітаційного моделювання GPSS. За допомогою блоків цієї мови можна будувати дискретно-подійні моделі загального виду. Більше того, ця мова перевірена на практиці протягом понад 40 років на комп'ютерах майже всіх типів. Таким чином, створення генератора імітаційних моделей, в якому як проміжний код використовується код мовою GPSS, дає змогу в разі необхідності змінювати або розширяти генеровану програмну реалізацію імітаційної моделі.

Що ж стосується формальної структури генератора імітаційних моделей, то відомо, що найширший клас структур імітаційних моделей покривають стохастичні мережі СМО, які використовуються під час моделювання інформаційних, технологічних, транспортних, ділових процесів, систем обслуговування загального вигляду. Вибір класу структур моделей визначає обмеження на використання об'єктів або систем конкретної предметної галузі, тобто їх необхідно зображувати як дискретні об'єкти класу стохастичних мереж СМО.

Для створення методики автоматизованого синтезу програмних реалізацій імітаційних моделей певного класу потрібно побудувати формалізовану модель системи чи процесу у вигляді теоретико-множинного опису. Широкий клас структур концептуальних і імітаційних моделей можна зобразити як орієнтовані графи у вигляді:

$$G = \langle V, P, U \rangle,$$

де V – кінцева множина вершин; P – кінцева множина дуг; U – функція, яка ставить у відповідність кожному ребру із множини P упорядковану пару вершин із множини V.

Для програмного генератора, що розробляється, вершинам графа відповідає кінцева множина можливих атомарних підмоделей A_1 об'єктів класів K. Характеристика класу K визначається сукупністю атрибутів b, які задають на полі класу

у вигляді пари $b = \langle i, f \rangle$, де i – ім'я атрибуту, що належить даному класу, а f – функція, визначена на полі класу, яка задає порядок розглядання вершин із множини V.

Отже, для стохастичних мереж множина вершин V відображається множиною атомарних підмоделей A_1, яка розбивається на два класи – K_1 і K_2:

$$A_1 = K_1 \cup K_2,$$

де K_1, K_2 – класи об'єктів, модельованих відповідно як одно- та багатоканальні СМО.

Уведемо визначення. Під повідомленням (транзактом) будемо розуміти вимогу на обслуговування СМО, тоді дуги графу стохастичної мережі визначають можливі напрямки передавання повідомлень (транзактів) від однієї підмоделі до іншої.

Множина A_1 визначає створення і використання імітаційних підмоделей, які дають змогу будувати узагальнену імітаційну модель завдяки засобам агрегованого опису класів однотипних підмоделей.

Множина P визначає створення і реорганізацію зв'язків між підмоделями. Тоді пара

$$\langle a_1^i, p^j \rangle \mid a_1 \in A_1, p \in P, \quad i = 1, ..., n; \quad j = 1, ..., m,$$

де n, m – відповідно кількість можливих підмоделей і зв'язків між ними, визначає підмодель СМО і можливі напрямки передавання повідомлень від неї.

Під час проектування СМО виникає необхідність не тільки визначити маршрути рухів повідомлень між підмоделями, але і вказати умови просування транзактів за визначеними маршрутами. Така ситуація можлива, якщо множина вихідних ребер, інцидентних деякому вузлу, містить більше одного елемента. Тому в разі побудови формалізованої моделі програмного генератора потрібно, щоб множина дуг P описувалась деякою сукупністю атрибутів d, значення яких будуть визначати процедуру вибору маршруту руху повідомлень від даної підмоделі.

Нехай F – функція побудови множини пар (вищенаведений вираз для абстрактної імітаційної моделі, причому $F = Q^{-1}$ і $F = \{f\}$.

Використовуючи під час генерування імітаційної моделі функцію F, можна об'єднати атомарний елемент і правило передавання повідомлення від нього, що дасть змогу провадити параметричне настроювання проектованої імітаційної моделі.

Детальний аналіз предметної галузі, тобто мереж СМО (див. розділ 2), показав, що множину підмоделей потрібно доповнити джерелами повідомлень і стоками, тобто клас атомарних підмоделей необхідно розширити до класу

$$A_2 = A_1 \cup B \cup C,$$

де B, C – класи об'єктів, які відповідно породжують повідомлення (генератори) і знищують повідомлення (термінатори).

Для побудови концептуальної моделі достатньо використовувати структуру вигляду

$$M_1 = \langle A_2, P \rangle.$$

У процесі побудови логічної моделі об'єкта ця структура вимагає уточнення. В цьому випадку структура M_1 перетвориться в структуру вигляду

$$M_2 = \langle A_2, P, F \rangle,$$

яка дає змогу додержуватись такої послідовності: атомарні підмоделі – правила вибору – параметричне налагодження.

Якщо для концептуального рівня досить використовувати структуру M_1, задавши її матрицею можливих переходів, то для логічного рівня використання структури M_2 є явно недостатнім і поверхневим. Необхідно провести конкретизацію з урахуванням того, що генератор програм орієнтований на конкретну мову моделювання GPSS, яка у свою чергу орієнтована на процеси. Покажемо, як структуру M_2 може бути перетворено на структуру M_3, що відображає програмну реалізацію імітаційної моделі.

Для цього визначимо специфікацію дискретної системи як структуру

$$M_3 = \langle X, Y, S, W, F_1, F_2 \rangle,$$

де X, Y, S – множини відповідно зовнішніх, внутрішніх і вихідних процесів (алгоритмів передавання повідомлень); F_1 – функції переходів; F_2 – вихідні функції, $W = \langle W_1, W_2 \rangle$ – змінні моделі.

Тоді

$$A_2 = \{\langle x, y \rangle \mid x \in X, y \in Y\},$$

де $x, y \in K_1 \cup K_2 \cup B \cup C$; $F = \langle F_1, F_2 \rangle$, $P \in S$.

Наведемо зв'язок структури M_3 з конкретними блоками мови моделювання GPSS:

- X – множина блоків GENERATE, TERMINATE;
- Y – блоки, пов'язані з визначенням пристроїв, накопичувачів, черг та блоки для збирання статистичних даних про черги;
- S – блоки, які визначають і (або) змінюють маршрут руху транзактів у моделі (TRANSFER, TEST, SELECT, SPLIT та ін.);
- W – визначення змінних у моделі (VARIABLE, SAVEVALUE);
- F_1 – функції, пов'язані з використанням логічних ключів (GATE, LOGICAL) для визначення умов руху транзактів;
- F_2 – функції, які визначають дисципліни надходження, обслуговування, руху і виходу транзактів із моделі.

У цьому разі множину F_2 можна ще інтерпретувати як таке відображення: нехай a_i – поточний процес моделі, $\{a_i\}$ – множина процесів моделі, куди мо-

же бути передано транзакт із процесу a_i (можливе й таке включення $a_2 \in \{a_2\}$), тоді

$$F_2: a_2 \to \{a_2\}.$$

Для програмного рівня формалізації модель визначається синтаксисом мови моделювання GPSS. Цей рівень у процесі автоматизації створення програмної реалізації моделі звичайному користувачеві не доступний, за винятком користувача-програміста, тому немає потреби в його формальному описі.

Особливість використання імітаційних моделей в інтерактивних системах визначається тим, що процедури пов'язані як з діями людини, так і машини. Тому необхідно вирішити проблему забезпечення ефективного діалогу моделі з користувачем, якому не відомі тонкощі математики, імітації, програмування, у рамках наведеного формального зображення процесу проектування імітаційних моделей. Такий діалог здійснюється мовою інтерактивної взаємодії з користувачем, яку реалізують через ієрархічну систему меню, інструментальну візуальну лінійку об'єктів і за допомогою елементів керування.

Отже, формальна модель опису об'єкта для проектування програмної реалізації імітаційної моделі зазнає послідовних змін структур M_1, M_2, M_3, причому, якщо структури M_1, M_2 задає користувач у інтерактивному режимі, відповідаючи на запити системи, то структуру M_3 має створювати програмний генератор, використовуючи лінгвістичний процесор (ЛП) і множину вибору альтернативних варіантів для компонентів атомарних підмоделей і їх зв'язків.

Основне призначення лінгвістичного процесора полягає в тому, що він шляхом інтерактивної взаємодії з користувачем деякою мовою L має побудувати підмножину таких висловлювань D із деякої множини G, що тільки висловлення D буде завжди істинне.

Отже, ЛП задається четвіркою елементів $\langle L, D, G, T\rangle$, де множина T визначає деяку теорію щодо висловлювань G. З прагматичного погляду множина G визначає всі можливі альтернативи вибору з меню, що утворюють мову L. Підмножина D визначає тільки ті вибрані альтернативи, за якими алгоритм роботи ЛП, що визначає теорію T, будує семантично і синтаксично правильні оператори GPSS-програми.

Мова L – це система елементів керування інтерактивного проектування моделей. Множина G відповідає множині альтернативних варіантів підмоделей, а підмножина висловлень D визначає параметричні налагоджені підмоделі для конкретної реалізації імітаційного проекту.

Для формалізації опису роботи ЛП скористаємося методами побудови трансляторів. З прагматичного погляду необхідно перетворити мову L взаємодії користувача з інтерактивною системою проектування в семантично і синтаксично правильно побудовану послідовність блоків GPSS-програми [91]. З урахуванням вищевикладеного визначимо атрибути, які описують елементи класів складових множини атомарних підмоделей A_2, у вигляді, в якому вони обробляються ЛП (з погляду теорії граматик). Для цього введемо спочатку деякі класи функцій аргументу

$h \in H$. Послідовність функції $\{f\}$ використовується для задавання різних імовірнісних законів розподілів часу обслуговування вимог у СМО

$$\{f\} \mid \forall f \in \{f\}: h \Rightarrow \Re,$$

де $h \in H$, а $\Re$ – такий алфавіт:

$$\Re = \langle \text{"Експоненціальний"}, \text{"Рівномірний"}, \text{"Нормальний"}, \text{"Ерланга"}, \text{"Детермінований"}, \text{"Логнормальний"}, \text{"Гамма-розподіл"}, \text{"Бета-розподіл"}, \text{"Вейбула"}.$$

За допомогою послідовності функції $\{\pi\}$ можна задати правила обслуговування вимог у СМО з урахуванням необхідності пріоритетного обслуговування

$$\{\pi\} \mid \forall \pi \in \{\pi\}: h \Rightarrow P^*,$$

де $h \in H$, а P^* – такий алфавіт:

$$P^* = \langle \text{"без пріоритету"}, \text{"відносний"}, \text{"абсолютний"} \rangle.$$

Послідовність функції $\{\rho\}$ дає змогу змінювати пріоритет вимог після обслуговування у визначеної СМО

$$\{\rho\} \mid \forall \rho \in \{\rho\}: h \Rightarrow \langle \text{"не змінюється"}, \text{"після обслуговування"} \rangle;$$

За допомогою послідовності функцій $\{\xi\}$ можна визначати процедури вибору повідомлень на обслуговування з черги до пристрою.

$$\{\xi\}, \ \forall \xi \in \{\xi\}: h \Rightarrow \langle \text{"FIFO"}, \text{"LIFO"}, \text{"за параметром"}, \text{"RAND"} \rangle.$$

Послідовність функції $\{\lambda\}$ служить для задавання певних обмежень для черги до пристою.

$$\{\lambda\} \mid \forall \lambda \in \{\lambda\}: h \Rightarrow \langle \text{"без обмежень"}, \text{"на довжину"}, \text{"на час перебування"} \rangle.$$

Послідовність функції $\{\tau\}$ служить для задавання типів змінних

$$\{\tau\} \mid \forall \tau \in \{\tau\}: h \Rightarrow \langle \text{"цілочисельні"}, \text{"дійсні"} \rangle.$$

Визначимо класи підмоделей одноканальних СМО як

$$K_1: b^1 = \langle \iota^1, f^1, \alpha^1, \beta^1, \pi^1, \rho^1, \delta^1, q^1 \rangle,$$

де ι^1 – ім'я пристрою, функція, значення якої складається з послідовності букв латинського алфавіту та цифр (довжина не більше ніж 8 символів);

f^1 – закон розподілу часу обслуговування вимоги пристроєм, $f^1 \in \{f\}$;

α^1, β^1 – параметри закону розподілу, числові скалярні функції аргументу $h \in H$;

π^1 – пріоритет обслуговування повідомлення в СМО, $\pi^1 \in \{\pi\}$;

ρ^1 – зміна пріоритету, $\rho^1 \in \{\rho\}$;

δ^1 – значення зміни пріоритету (скалярна цілочисельна функція аргументу $h \in H$;

q^1 – характеристика черги повідомлень до пристрою, $q^1 \in Q$.

Черга вимог до пристою описується такою структурою

$$Q = \langle \iota_q, \xi_q, \lambda_q, \delta_q \rangle,$$

де ι_q – ім'я черги, функція, значення якої складається з послідовності букв латинського алфавіту та цифр (довжина не більше ніж 8 символів);

ξ_q – правило вибору повідомлень з черги, $\xi_q \in \{\xi\}$;

λ_q – обмеження, що накладаються на чергу, $\lambda_q \in \{\lambda\}$;

δ_q – обмежувальне значення, скалярна цілочисельна функція аргументу $h \in H$.

Клас підмоделей багатоканальних СМО будемо задавати такою структурою:

$$K_2: b^2 = \langle \iota^2, N^2, n^2, f^2, \alpha^2, \beta^2, \pi^2, \rho^2, \delta^2, q^2 \rangle,$$

де ι^2 – ім'я пристрою, функція, значення якої складається з послідовності букв латинського алфавіту та цифр (довжина не більше ніж 8 символів);

N^2 – кількість каналів багатоканальної СМО, скалярна цілочисельна функція аргументу $h \in$ H;

n^2 – кількість каналів багатоканальної СМО, що займає одночасно одна вимога, скалярна цілочисельна функція аргументу $h \in$ H;

f^2 – закон розподілу часу обслуговування вимоги пристроєм, $f^2 \in \{f\}$;

α^2, β^2 – параметри закону розподілу, числові скалярні функції аргументу $h \in H$;

π^2 – пріоритет обслуговування вимоги в СМО,

$\pi^2 \in \{\pi'\}$, π': $h \Rightarrow P^* \setminus$ ("абсолютний");

ρ^2 – зміна пріоритету, $\rho^2 \in \{\rho\}$;

δ^2 – значення зміни пріоритету, скалярна цілочисельна функція аргументу $h \in H$;

q^2 – характеристика черги повідомлень до пристрою, $q^2 \in Q$.

Клас генераторів вимог задається такою структурою:

$$B: b^3 = \langle f^3, \alpha^3, \beta^3, \Delta_1^3, \Delta_2^3, \Delta_3^3, d_1^3, d_2^3, d_3^3 \rangle,$$

де f^3 – закон розподілу часу надходження вимог до моделі, $f^3 \in \{f\}$;

α^3, β^3 – параметри закону розподілу, числові скалярні функції аргументу $h \in H$;

Δ_1^3, Δ_2^3, Δ_3^3 – відповідно затримка першої вимоги, обмеження на кількість вимог, які генеруються, задавання пріоритету вимогам, булеві функції аргументу $h \in H$, тобто перераховані значення або задаються, або ні;

d_1^3, d_2^3, d_3^3 – значення описаних вище властивостей елементів класу генераторів вимог, числові скалярні функції аргументу $h \in H$.

Клас термінаторів задається структурою

$$C : c^1 = \langle \Omega^1, \omega^1 \rangle,$$

де Ω^1 – підрахування кількості вимог, які залишили модель, величина, що може приймати значення 0 чи 1 (булева);

ω^1 – значення для підрахування кількості вимог, які залишили модель, числові скалярні функції аргументу $h \in H$.

Визначення змінних у моделі задається структурою

$$W : w^1 = \langle \varphi^1, \theta^1, \tau \rangle,$$

де w^1 – характеристика змінної;

φ^1 – ім'я змінної, функція, значення якої складається з послідовності букв латинського алфавіту та цифр (довжина не більше ніж 8 символів);

θ^1 – тіло змінної, функція, значення якої складається з послідовності букв латинського алфавіту та символів #, $, –, +, /, ^, * (довжина не більше ніж 256 символів);

τ – тип змінної, $\tau^1 = \langle$ "з фіксованою крапкою", "з плаваючою крапкою"$\rangle$.

Визначимо алфавіт Λ теорії T, тобто всі можливі послідовності, якими оперує ЛП:

$$\Lambda = \Re \cup P^* \cup \langle \text{"FIFO", "LIFO", "за параметром", "RAND"} \rangle \cup \langle \text{"без обмежень", "на довжину", "на час перебування"} \rangle \cup \langle \text{"з фіксованою точкою", "дійсні"} \rangle \cup A^l \cup A^g \cup Ц,$$

де A^l – латинський алфавіт;

A^g – алфавіт мови GPSS;

Ц – арабські цифри.

Покажемо фрагмент правил виводу ланцюжків D для атомарних підмоделей класу K_1. Попередньо додатково визначимо операцію конкатенації як $a \oplus^* b$, що означає:

- якщо a і b – рядки або символи, то результатом буде проста їхня конкатенація;
- якщо один з операндів є числом, то результатом буде конкатенація його строкового зображення з іншим операндом, наприклад "Windows" $\oplus^*$ 2000 = "Windows 2000".

Операції конкатенації необхідні для перевірки правильності введених користувачем числових значень параметрів, які після перевірки перетворюються у звичайний текстовий формат мови GPSS.

Спочатку покажемо як створюються рядки GPSS-програми вибору функцій розподілу $\{f\}$ для блока ADVANCE:

⟨"Рівномірний"⟩ ⇒ ⟨"ADVANCE "⊕*α¹⊕*","⊕*β¹⟩;
⟨"Детермінований"⟩ ⇒ ⟨"ADVANCE "⊕*α¹⟩;
⟨"Експоненціальний"⟩ ⇒ ⟨"ADVANCE "⊕*α¹⊕*",FN$EXPDIS"⟩;
⟨"Нормальний"⟩ ⇒ ⟨"VNOR VARIABLE "⊕*α¹⊕*"+"⊕*β¹⊕*"#FN$NORM"⟩;
⟨"ADVANCE V$VNOR"⟩;
⟨"Ерланга"⟩|⟨"Логнормальний"⟩|⟨"Гамма-розподіл"⟩|⟨"Бета-розподіл"⟩
|⟨"Вейбула"⟩ ⇒ ⟨"ADVANCE FN$"⊕*$f^*(r)$⊕*"_"⊕*α¹⊕*"_"⊕*β¹⊕*⟩.

де $f^*(r)$: $r \in \Re \Rightarrow$ ⟨"Ерланга", "Логнормальний", "Гамма-розподіл", "Бета-розподіл", "Вейбула"⟩.

Покажемо фрагмент створення рядків GPSS-програми для атомарних підмоделей класу K_1, тобто одноканальних СМО:

⟨"без пріоритету"⟩|⟨"відносний"⟩ ⇒ ⟨"SEIZE "⊕ι²⟩,⟨"RELEASE "⊕ι¹⟩;
⟨"абсолютний"⟩ ⇒ ⟨"PREEMPT "⊕ι¹⟩,⟨"RETURN "⊕ι¹⟩;
⟨"абсолютний"⟩ ⇒ ⟨"SAVEVALUE 1,PR"⟩,⟨"SAVEVALUE 1+,"⊕*δ¹⟩,⟨"PRIORITY X1"⟩;
⟨"FIFO"⟩ ⇒ ⟨"QUEUE "⊕$ι_q$⟩,⟨"DEPART "⊕$ι_q$⟩;
⟨"LIFO"⟩ ⇒ ⟨"QUEUE "⊕$ι_q$⟩,⟨"DEPART "⊕$ι_q$⟩,⟨"LINK SP1, LIFO DEVICE1"⟩,
⟨"UNLINK SP1, DEVICE1"⟩;
⟨"на довжину"⟩ ⇒ ⟨"TEST L Q$"⊕$ι_q$⊕","⊕*$δ_q$⊕*",COMTER"⟩;
⟨"на час перебування"⟩ ⇒ ⟨"MARK T"⊕$ι_q$⟩,⟨"TEST L MP$T"⊕$ι_q$⊕","⊕*$δ_q$⊕*",COMTER"⟩.

Список SP1 пов'язаний з пристроєм з номером 1. Позначка DEVICE1 використовується для блоку SEIZE з номером 1, а позначка COMTER – для вимог, яким відмовлено під час поставлення в чергу (наприклад, вони можуть знищуватись за допомогою блока TERMINATE, що має цю позначку).

Для класу K_2 побудова ланцюжків правил виводу D буде аналогічною за винятком

⟨ι²⊕"STORAGE "⊕*N^2⟩;
⟨"без пріоритету"⟩|⟨"відносний"⟩ ⇒ ⟨"ENTER "⊕ι²⊕","⊕*n^2⟩,
⟨"LEAVE "⊕ι²⊕","⊕*n^2⟩;
⟨"на час перебування"⟩ ⇒ ⟨"MARK T"⊕$ι_q$⟩,⟨"GATE SNF"⊕ι²⟩,
⟨"TEST L MP$T"⊕$ι_q$⊕","⊕*$δ_q$⊕*","COMTER"⟩.

Як було зазначено вище, для більш повного покриття множини різних моделей стохастичних мереж необхідно визначити правила передавання повідомлень від вузла до вузла, коли множина вихідних ребер, інцидентних даній вершині концептуальної схеми моделі, перевищує 1. Набір атрибутів d, за допомогою яких описують елементи множини ребер P, може бути рішенням цієї проблеми, однак зручніше розглядати всю множину вихідних ребер конкретного вузла. Позначимо цю множину як R_{a2}. Уведемо функцію σ як функцію вибору маршруту переміщення транзакту. Множиною значень цієї функції будуть елементи множини

$$\sigma(a_2^i,\theta) \in \{p_j^i\} \mid \forall\, p_j^i \in \{p_j^i\}\ \exists\ \langle a_2^i, p_j^i\rangle,$$

де $a_2^i \in A_2$, $p_j^i \in P$, θ – ознака, за якою описується стан вершини a_2^i, а пари елементів $\langle a_2^i, p_j^i\rangle$ є припустимими щодо функції F.

Тобто функція σ здійснює вибір ребра руху вимоги для вершини a_2^i із множини $R_{a2}{}^i$. Опишемо умови, які накладаються на переміщення вимог між підмоделями. Ці умови буде визначено як атрибути d^r елементів множини $R_{a2}{}^i$:

$$d^r = \{\langle\text{безумовно}\rangle, \langle\text{за імовірністю}\rangle, \{\varphi^r\}\langle\text{обмеження числом}\rangle\{\phi^r\}, \langle\text{в усіх напрямках}\rangle\},$$

де $\{\varphi^r\}$ – множина значень імовірностей переміщення вимог по визначених ребрах, (числові скалярні функції);

$\{\phi^r\}$ – числа, які обмежують кількість транзактів, переданих по визначених ребрах (числові скалярні функції).

Таким чином, для передавання вимог можна побудувати правила висновку ланцюжків D, аналогічні правилам для атомарних підмоделей класу K_1.

На множині вихідних ланцюжків правил висновку теорії T задамо функцію Φ, яка встановлює антирефлексивне транзитивне відношення (строгий порядок). Покажемо в табличному вигляді цю функцію (табл. 5.1) для множини аргументів – ланцюжків D, одержуваних за правилами висновку для атомарних підмоделей класу K_1.

Таблиця 5.1. Впорядкування послідовності блоків згідно з функцією Φ

Аргумент	Значення
⟨"ADVANCE "⟩	8
⟨"PREEMPT "⟩	6
⟨"SEIZE "⟩	6
⟨"RELEASE "⟩	9
⟨"RETURN"⟩	9
⟨"SAVEVALUE 1,PR"⟩	11
⟨"SAVEVALUE 1+,"⊕*δ^1⟩	12
⟨"PRIORITY X1"⟩	13
⟨"QUEUE "⟩	2
⟨"DEPART "⟩	7
⟨"LINK SP1,LIFO"⟩	5
⟨"UNLINK SP1"⟩	10
⟨"TEST L Q$"⊕$\iota_q$⊕","⊕*$\delta_q$⊕*",COMTER"⟩	2
⟨"MARK T"⊕ι_q⟩	1
⟨"TEST L MP$T" ⊕ ι_q ⊕ "," ⊕* δ_q ⊕* ",COMTER"⟩	4
⟨"GATE NU"⟩	3

Таким чином, одержуємо пріоритетно-упорядковану множину операторів програми на мові GPSS.

Розглянемо приклад створення програми реалізації імітаційної моделі СМО. Нехай на вхід лінгвістичного процесора надійшов опис моделі у такому вигляді:

$$M_2 = \langle a, \varnothing, \varnothing \rangle,$$

де $a \in K_1 \mid b_a = \langle$"MACHINE", "Експоненціальний", 50, 0, "абсолютний", "після обслуговування", 10, "QUEUE1", "LIFO", "на довжину", 8$\rangle$;

Застосовуючи правила виводу для класу K_1, отримаємо такі рядки коду мовою GPSS:

```
ADVANCE 50,FN$EXPDIS
PREEMPT MACHINE
RETURN MACHINE
SAVEVALUE 1,PR
SAVEVALUE 1+,10
PRIORITY X1
QUEUE QUEUE1
DEPART QUEUE1
LINK SP1,LIFO,QUEUE1
UNLINK SP1,QUEUE1
TEST L Q$QUEUE1,8,COMTER
```

Використовуючи функцію Φ, отримаємо впорядковану множину рядків:

```
TEST L Q$QUEUE1,8,COMTER
QUEUE QUEUE1
LINK SP1,LIFO,QUEUE1
PREEMPT MACHINE
DEPART QUEUE1
ADVANCE 50,FN$EXPDIS
RETURN MACHINE
UNLINK SP1,QUEUE1
SAVEVALUE 1,PR
SAVEVALUE 1+,10
PRIORITY X1
```

З огляду на вказані вище умови і переходячи до програмної реалізації, можна стверджувати, що в ЛП необхідні такі підпрограми:

- оперування з вершинами концептуальної мережі, які настроюються на тип конкретного вузла, що визначається належністю до заданої підмножини класу атомарних підмоделей;
- генерування програмного коду мовою GPSS для випадку передавання повідомлень;
- роботи з функціями імовірнісних розподілів і функціями, що задає користувач у моделі;
- заповнення текстів заголовків для функцій, змінних, таблиць і накопичувачів;
- збирання статистичних даних про роботу елементів моделі;
- остаточного формування тексту GPSS-програми.

Для зручності програмної реалізації ЛП необхідно задавати вузли не як деяку однорідну множину об'єктів, а розділити їх за типами. Причому якщо ввести типи вузлів аналогічно класам K_1, K_2, B и C, то завдяки уніфікації формальних правил виводу, визначених для кожного з цих класів, і програмних функцій обробки інформації трудомісткість написання коду й ефективність його роботи різко збільшиться.

Згідно з наведеними вище методами проектування імітаційних моделей та формалізованою процедурою проектування програмного генератора, а також з огляду на вимоги, виконання яких передбачає технологія проектування імітаційних моделей досліджуваних об'єктів, визначається організаційна структура інтерактивної системи імітаційного моделювання ISS 2000 [61]. Вона складається із сукупності взаємопов'язаних програмних модулів, які виконують різні процедури: керування, прийняття рішень і перетворення інформації й даних, що містять відомості функціонального та інформаційного характеру.

На концептуальному рівні модель системи задається графом, вершини якого являють собою множину об'єктів моделі, таких як генератори вимог, одно- або багатоканальні пристрої обслуговування, термінатори тощо. На логічному рівні визначаються властивості цих об'єктів і зв'язки між ними. Програмний рівень подання моделі містить готовий код програми моделювання на мові GPSS, який створюється після компіляції проекту моделі.

Такий програмний генератор повністю автоматизує процес створення імітаційної моделі та проведення експериментів з нею, але не означає, що користувач не може змінити або дописати код програми. Користувач може вносити зміни до коду програми моделювання.

5.9. Імітаційна модель персонального комп'ютера

Визначимо цілі моделювання персонального комп'ютера (ПК). Припустимо, що метою моделювання є прогнозування впливів зміни продуктивності устаткування на середню пропускну здатність ПК і середній час виконання програм. Будемо розглядати ПК із загальною шиною, схему якого зображено на рис. 5.6.

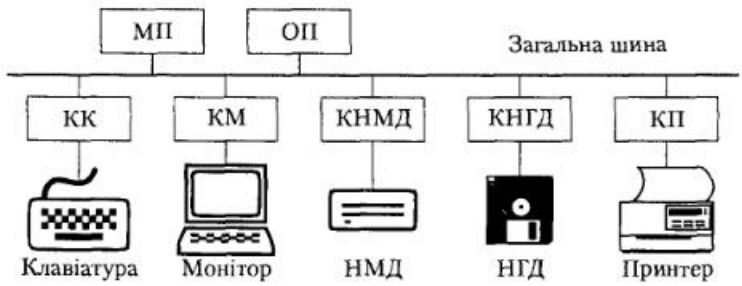


Рис. 5.6. Структура персонального комп'ютера із загальною шиною

З основних пристроїв комп'ютера виділимо ті, які повинні бути відображені в моделі, а саме: мікропроцесор (МП), оперативну пам'ять (ОП), монітор, накопичувач на жорстких магнітних дисках (НМД), накопичувач на гнучких магнітних дисках (НГД), принтер, клавіатуру та контролери цих пристроїв – контролер клавіатури (КК), контролер монітору (КМ), контролер НМД (КНМД), контро лер (КНГД), контролер принтера (КП).

Функціонування комп'ютера можна описати таким чином. Користувачі з клавіатури вводять команди, що складаються з кількох інструкцій, які викликають програмні модулі. Кожну інструкцію будемо вважати одним кроком завдання. Програма та дані знаходяться на магнітних дисках і завантажуються в оперативну пам'ять за окремою командою. Роботу користувача з клавіатурою в моделі можна не враховувати, припускаючи, що набір команд з клавіатури відображається на екрані монітора. Для перенесення програми з магнітних дисків у пам'ять комп'ютера організується вхідний буфер. Якщо обсяг пам'яті достатній для виконання першого кроку, то після завантаження програми в пам'ять і надання їй часу процесора вона може опрацьовуватися ним доти, доки не видасть запит до магнітних дисків або принтера. У той час, коли здійснюється введення-виведення даних на зовнішні пристрої, процесор може виконувати інше завдання. Після виконання операції введення-виведення даних завдання знову чекає в черзі, доки йому не буде надано час процесора. Якщо крок завдання закінчився, то звільняється пам'ять і здійснюється виведення даних на принтер через вихідний буфер.

Вважається, що на жорсткому диску знаходяться програми операційної системи і системні програми, а також бібліотеки користувача. Під час виконання програми вона може запитувати вхідні дані з гнучкого диска. Роботу операційної системи і системних програм до уваги не прийматимемо, припускаючи, що вони в однаковій мірі впливають на програми користувачів.

5.9.1. Концептуальна модель персонального комп'ютера

Всі пристрої, продуктивність яких впливає на час виконання завдання на комп'ютері, будемо розглядати як ресурси системи. Такі пристрої, як монітор, процесор, накопичувачі на магнітних дисках і принтер, не можна експлуатувати в режимі розподілу часу, тобто в конкретний час їх може використовувати тільки одне завдання. У пам'яті ж одночасно можуть знаходитись кілька завдань. Завдання може одночасно використовувати кілька ресурсів.

Враховуючи те, що швидкість роботи контролерів пристроїв уведення-виведення значно перевищує швидкість роботи самих пристроїв, будемо вважати, що швидкість роботи пристроїв визначається швидкістю роботи самих пристроїв, а не їх контролерів.

Тоді структурну схему концептуальної моделі комп'ютера можна відобразити так, як зображено на рис. 5.7. Зазначені на схемі лічильники необхідні для підрахування кількості завдань, які надійшли до комп'ютера і залишили його.

Вхідні і вихідні буфери можна об'єднати відповідно з монітором і пристроєм для друку. Тоді в концептуальній моделі будуть явно задані такі ресурси: МП, ОП, НМД, НГД, МОНІТОР, ПРИНТЕР.

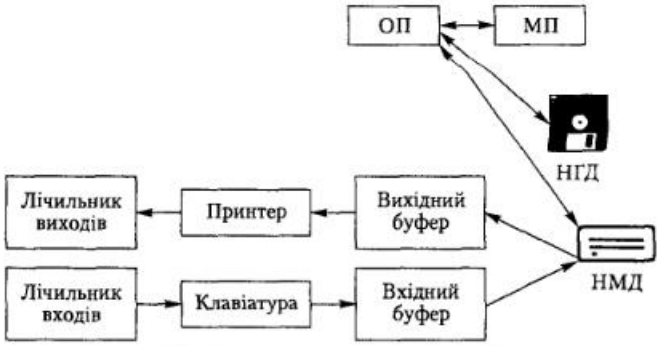


Рис. 5.7. Концептуальна модель комп'ютера

З огляду на мету моделювання визначимо, що на виході моделі необхідно отримати дані про середній час виконання завдань і пропускну здатність комп'ютера (кількість завдань, виконаних за певний час). Вхідні дані визначають закон розподілу надходження завдань у комп'ютер, число кроків у кожному завданні та ємність пам'яті, загальний час виконання кроку завдання процесором, частоту запитів до зовнішніх накопичувачів і пристроїв, розподіл обсягу виведеної на друк інформації, обсяг інформації, яка зчитується або записується на зовнішні накопичувачі, обсяг інформації, що вводиться. Ці дані можна отримати за допомогою програм тестування реального комп'ютера. Крім цих даних необхідно задати параметри для пристроїв комп'ютера, такі як швидкодія процесора, ємність пам'яті, накопичувачів на магнітних дисках, вхідних і вихідних буферів, мінімальний та максимальний час пошуку даних на магнітних дисках, швидкість друку на принтері, швидкість уведення інформації з клавіатури.

5.9.2. Розроблення імітаційної моделі

Для розроблення моделі скористаємось подійним алгоритмом моделювання. Для кожної пари «завдання–ресурс» будемо визначати, як довго деяке завдання J використовуватиме деякий ресурс R, тобто визначатимемо інтервал часу, коли завданню J буде призначено ресурс R і коли воно звільнить цей ресурс.

Однак перед тим, як ресурс R буде призначено завданню J, повинна надійти вимога на його використання. У загальному випадку завдання може чекати в черзі надання ресурсу.

Тоді під час створення імітаційної моделі комп'ютера можна користуватись такими двома способами.

1. Створити та підтримувати протягом моделювання список подій у хронологічному порядку.
2. Створити підпрограми опрацювання подій кожного типу і викликати їх тоді, коли подія даного типу стає першою в списку подій. Всі підпрограми подій можуть змінювати список подій, створюючи нові події та плануючи їхнє виконання в деякому майбутньому модельному часі.

Діаграму подій імітаційної моделі комп'ютера зображено на рис. 5.8.

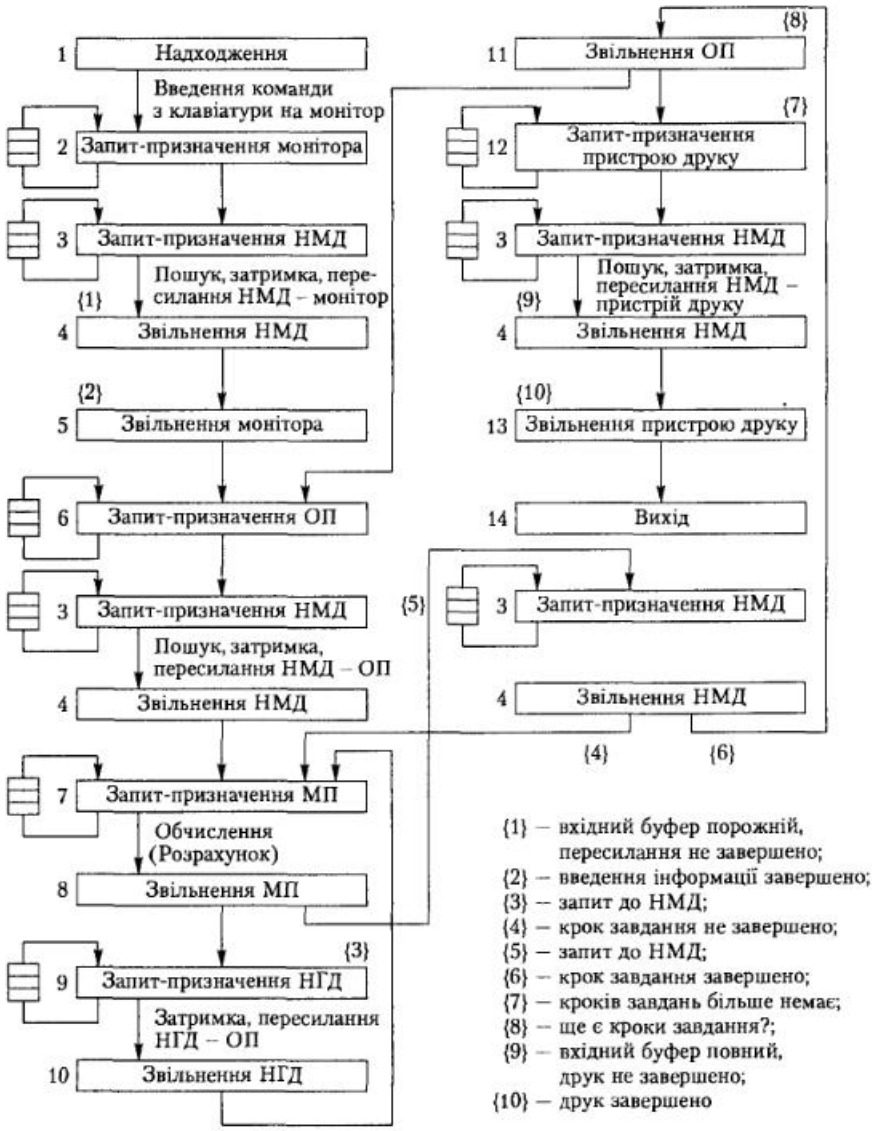


Рис. 5.8. Діаграма подій імітаційної моделі комп'ютера

Тепер опишемо алгоритми роботи цих підпрограм.

Підпрограма Запит-призначення

Якщо ресурс R вільний і його можна негайно виділити завданню J, то він позначається як зайнятий і планується настання наступної події для завдання J. У протилежному випадку запит від завдання J ставиться в чергу до ресурсу R.

Підпрограма Звільнення

Ресурс R позначається як вільний і планується наступна подія для завдання J. Якщо черга до ресурсу R не порожня, планується подія Запит-призначення чергового завдання K для ресурсу R і завдання вибирається з черги до ресурсу R. У протилежному випадку підпрограма не виконує ніяких дій.

Для того щоб простежити шляхи проходження завдань по ресурсам комп'ютера, необхідно визначити ще дві події і відповідні підпрограми: Надходження та Вихід. Між цими подіями не існує явного зв'язку. Якщо уявити персональний комп'ютер як деякий узагальнений ресурс, то підпрограма Надходження завжди призначає цей узагальнений ресурс завданню, яке надійшло.

Підпрограма Надходження

Готує характеристику (параметри) завданню J, тобто фіксує час надходження завдання до ПК. Визначає вимоги завдання до ресурсів. Планує наступну подію для J і прибуття наступного завдання K.

Підпрограма Вихід

Підпрограма Вихід є набагато простішою за підпрограму Звільнення. Вона лише знищує характеристику завдання J.

Структурну схему програми імітаційної моделі комп'ютера наведено на рис. 5.9.

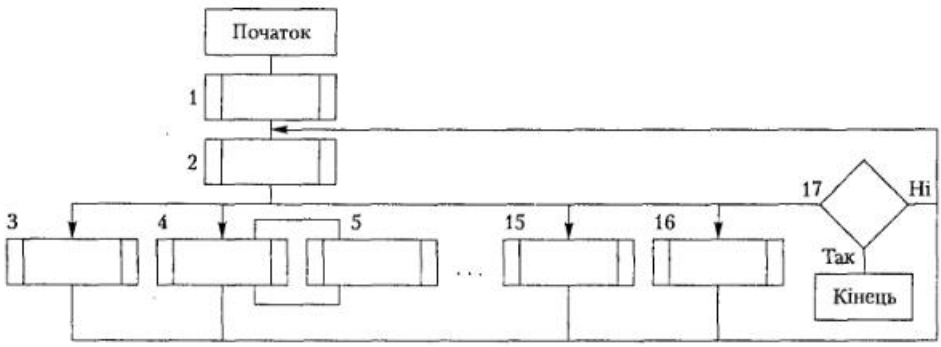


Рис. 5.9. Структурна схема програми імітаційної моделі комп'ютера:
1 — підпрограма ініціалізації; 2 — ПКМ; 3 — підпрограма події 1 (Надходження);
4 — підпрограма події 2 (Запит-призначення МОНІТОРА); 5 — Звільнення МОНІТОРА; ...
15 — підпрограма події 13 (Вихід); 16 — Вибірка; 17 — перевірка завершення моделювання

Програма ініціалізації готує структури даних, необхідні для моделювання, та задає їм початкові значення. Програма керування моделюванням вибирає (викликає) наступну подію, яка після виконання підпрограми повертає керування ПКМ, а потім збільшує модельний час до наступної події. Обидва ці завдання виконуються шляхом перевірки запису в списку подій.

Кожний запис у цьому списку містить:

- тип події, яка використовується для виклику відповідної підпрограми події;
- ім'я або номер завдання, що передається як аргумент усім підпрограмам події;
- тип події, що буде викликатись за даною подією.

Якщо необхідно враховувати пріоритетне планування, алгоритм роботи підпрограми обробки події процесора значно ускладнюється. Програми, які виконуються комп'ютером, можуть мати різний пріоритет, наприклад програми операційної системи і програми користувачів. Якщо поява більш пріоритетного завдання може перервати виконання процесором завдання з меншим пріоритетом, то необхідно дозволити процесору виконувати завдання з більшим пріоритетом, як тільки воно з'явиться в черзі. У цьому випадку подію звільнення процесора від завдання потрібно вивести зі списку подій за допомогою програми Запит-призначення для процесора і процесор призначити завданню з більш високим пріоритетом. Ця ж підпрограма повинна поставити перерване завдання в чергу до процесора для того, щоб надати їй його знову для доопрацювання перерваної програми. Надходження завдання з більшим пріоритетом може також перервати завдання, яке виконує процесор, тобто в цьому випадку може утворитись список перерваних завдань.

На діаграмі подій є події, після яких можливе одночасне настання двох подій. Ці події є взаємовиключними. Порядок звертання до тих або інших подій залежить від логіки та характеристик завдань, виконуваних користувачем.

Наприклад, після звільнення НМД для j-го завдання необхідно запланувати настання однієї з таких подій:

- Запит-призначення НМД;
- Запит-призначення МП;
- Звільнення ОП;
- Звільнення принтера.

Наприклад, Запит-призначення МП буде заплановано в тому випадку, якщо завдання J було тільки що завантажене в пам'ять або запит на НМД був задовільний, а обчислення для поточної команди завдання ще не завершене. Таким чином, інформація про послідовність виконання подій у програмі повинна утримуватись у самій моделі та безперервно опрацьовуватись у процесі моделювання.

Підпрограму Вибірка призначено для збору статистичних даних у процесі моделювання. Наприклад, пропускну здатність комп'ютера можна визначити, якщо підрахувати завдання, які залишили комп'ютер у підпрограмі Вихід, і поділити їх на час моделювання після виходу всіх завдань. Також потрібно збирати статистичні дані про завантаження ресурсів. Для цього необхідно визначити загальний час використання ресурсу та поділити його на загальний час моделювання.

Для визначення середнього часу виконання завдань користувачів у системі позначимо через $\bar{t}_i$ середній час проходження по перших i завданнях, а через t_j – час проходження завдання j. Тоді середній час виконання завдань знайдемо як

$$\bar{t}_i = \frac{1}{i}\sum_{j=1}^{i} t_j. \tag{5.1}$$

Для наступного завдання $j + 1$ формула (5.1) буде мати такий вигляд:

$$\bar{t}_{i+1} = \frac{i}{i+1}\bar{t}_i + \frac{1}{i+1}t_{i+1}. \tag{5.2}$$

Аналогічно запишемо оцінку вибіркової дисперсії $\hat{\sigma}^2_{i+1}$, використовуючи визначення для оцінки дисперсії та знаючи $\hat{\sigma}^2_i$, отримаємо за допомогою кількох алгебричних дій вираз

$$\hat{\sigma}^2_{i+1} = \frac{i}{i+1}\hat{\sigma}^2_i + \frac{i}{(i+1)^2}(t_{i+1} - \bar{t}_i)^2. \qquad (5.3)$$

Формули (5.2) і (5.3) дають змогу рекурентно обчислювати всі послідовні значення відповідно для оцінювання математичного сподівання та дисперсії ($\hat{\sigma}^2_1 = 0$). Потрібно тільки знати значення i, σ_i^2, t_{i+1} і $\bar{t}_i$.

5.9.3. Процесно-орієнтований алгоритм моделювання персонального комп'ютера

Для того щоб урахувати взаємодію процесів у моделі, потрібно вести два списки подій: список майбутніх подій (СМП) і список умовних подій (СУП) (рис. 5.10). Основна увага при розробці моделюючого алгоритму зосереджується на шляхах проходження завдання в моделі і взаємодіях процесів, що виконуються паралельно. Підпрограма процесу описує «життєвий цикл» завдання в системі.

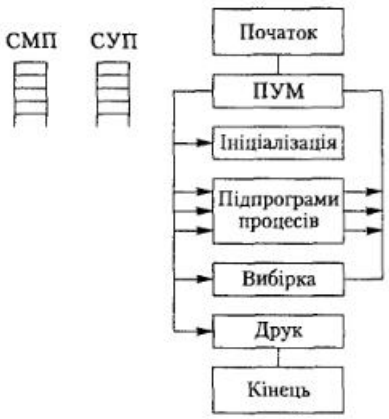


Рис. 5.10. Списки подій для моделі комп'ютера, орієнтованої на процеси

Якщо наступна подія для модельованого завдання повинна відбутись у деякий модельний час, то цю подію можна планувати безумовно, і для неї додається запис у СМП. Якщо це зробити неможливо, наприклад через взаємодію з іншими завданнями, які в даний момент займають необхідні ресурси, то запит подається в СУП. Після цього керування повертається до ПКМ, яка здійснює повернення в підпрограму процесу через точку входу, що є безпосередньо наступною за оператором ЧЕКАТИ ⟨умови⟩ доти, доки завдання не буде вибране для обслуговування цим ресурсом. Таким чином, підпрограми процесів повинні мати ряд точок входу та виходу з процесів.

На рис. 5.11 зображено схему частини процесу для надання завдання МП для обчислювання. У цій схемі ПКМ, а не програма процесу, планує та призначає ресурси. Схему можна модифікувати, щоб це завдання вирішувала програма процесу.

Дійсно, ресурси розглядаються як пасивні об'єкти, і ця схема не моделює в явному вигляді поведінку ресурсу.

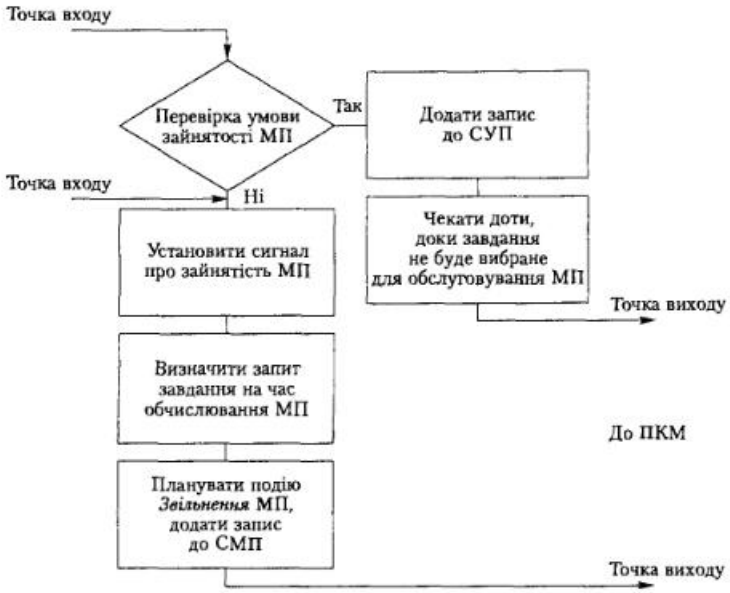


Рис. 5.11. Фрагмент блок-схеми призначення процесора завдання

Фрагмент блок-схеми, яка в явному вигляді моделює поведінку ресурсу, зображено на рис. 5.12.

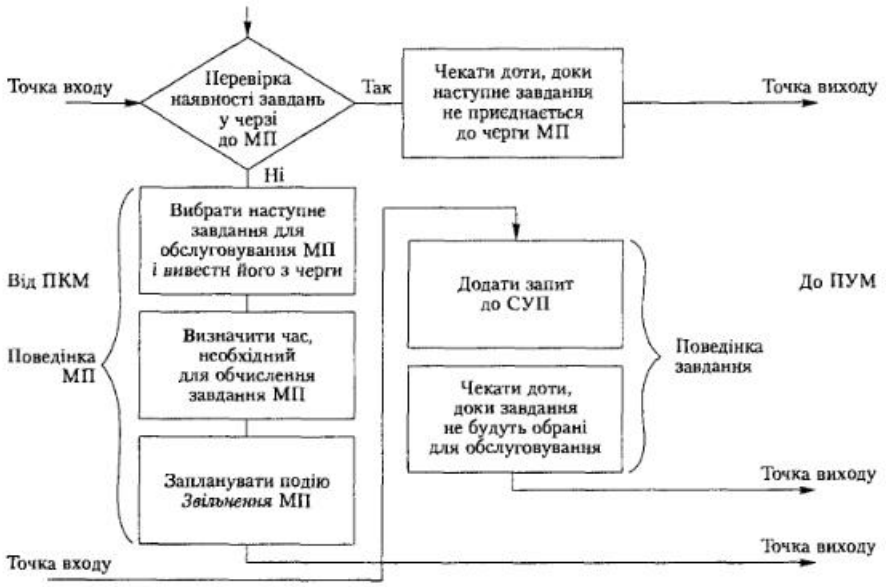


Рис. 5.12. Фрагмент схеми, яка в явному вигляді моделює поведінку ресурсу

5.10. Перевірка достовірності і правильності імітаційних моделей

Технологія імітаційного моделювання охоплює всі етапи життєвого циклу моделі – від її створення до впровадження та супроводження. У розділі 5.3 було визначено основні взаємопов'язані етапи імітаційного моделювання, після кожного з яких необхідно робити перевірки, щоб довести відповідність моделі цілям моделювання. Перш за все під час перевірки моделі потрібно відповісти на два основних запитання:

- Чи правильно побудовано імітаційну модель?
- Чи побудовано правильну імітаційну модель?

Перше питання стосується верифікації (*verification*) – перевірки достовірності моделі, яка доводить, що модель правильно і з достатньою точністю переведено від однієї форми в іншу. Перевірка достовірності моделі пов'язана з правильною побудовою моделі. Точність перетворення формулювання проблеми в модельний опис і точність перетворення модельного зображення із логічної блок-схеми у розроблену комп'ютерну програму оцінюються під час перевірки достовірності моделі.

Друге питання стосується валідації (*validation*), тобто обґрунтованості моделі, відповідності її задуму, або доказу правильності моделі. Модель повинна в межах галузі використання працювати з потрібною точністю, і її використання повинне відповідати цілям моделювання. Це стосується порівняння поведінки моделі з поведінкою системи.

Останнім часом використовують ще один термін – акредитація (*accreditation*), тобто офіційне свідчення того, що імітаційна модель прийнятна для застосування в межах деякої області, обумовлених цілями моделювання.

Існує багато методів для перевірки достовірності та правильності імітаційної моделі й довіри до неї. У праці Болкі [71] розглянуто понад 100 методів для дослідження звичайних і об'єктно-орієнтованих імітаційних моделей та загальні принципи перевірки моделей. Перевірка достовірності і правильності моделі, тестування, адекватність, легалізація і дії щодо оцінювання довіри насамперед пов'язані з вимірюванням і оцінюванням точності під час моделювання та імітації. Обґрунтування адекватності моделі доводить, що модель працює, у межах використання, із задовільною точністю, сумісною з метою імітаційного моделювання.

Ітераційна процедура проектування імітаційної моделі передбачає порівняння результатів моделювання та вихідних змінних реальної системи за умови, що вхідні дані однакові, у результаті чого визначається точність моделі. Модель вважається досить точною, якщо її вихідні змінні відрізняються від вихідних даних модельованої системи менш ніж на задані граничні значення. Отже, точність моделі визначається різницею між вихідними даними системи та моделі. Під час порівняння двох варіантів моделі для визначення кращого необхідно використовувати однакові послідовності випадкових чисел. За показник якості критерію оцінювання точності можна брати похибку, яка визначається як різниця між середніми

значеннями вихідних величин системи та моделі. Якщо для деякої вихідної змінної системи y_i ($i = 1, ..., m$) знайдено n значень, то

$$E_1 = \frac{1}{n}\sum_{i=1}^{n}(y_{ij} - y'_{ij}), \quad i = 1, ..., m; \quad j = 1, ..., n,$$

де y'_{ij} – i-та вихідна змінна моделі.

У разі використання похибки як показника якості критерію E_1 можливий значний розкид конкретних i-х значень для величин y_{ij} та y'_{ij}. Тому такий критерій буде кращим у тому разі, якщо в ньому використовуються функції розподілу для y_{ij} та y'_{ij}, тобто відповідно $F_i(y)$ та $F_i(y')$. Тоді можна визначити похибку як

$$E_2 = \int_0^{+\infty} |F_i(y) - F_i(y')|\, dy.$$

Доцільніше використовувати оцінку E_2, ніж E_1, бо перша дає змогу порівнювати локально, або поточково, ці дві функції і підсумовувати абсолютні значення їх різниць. Завдяки цьому можна уникнути часткової компенсації похибок через різницю знаків. Здобуття результатів моделювання, розподіли яких ідентичні або близькі, є необхідною, але не достатньою умовою для того, щоб модель була точною в межах області розумних значень вхідних змінних. У цьому разі цілком можливо, що поведінка деяких процесів у моделі відрізнятиметься від поведінки цих самих процесів у системі, навіть якщо модель за деяких вхідних умов демонструє такі самі вихідні розподіли, як і система. Причому ефективнішим критерієм оцінювання, ніж E_1 та E_2, буде той, в якому використовується розподіл різниці значень для y_{ij} та y'_{ij}. Як такий критерій можна застосувати середнє арифметичне квадратів різниць значень:

$$E_3 = \frac{1}{n}\sum_{i=1}^{n}(y_{ij} - y'_{ij})^2, \quad i = 1, ..., m; \quad j = 1, ..., n.$$

Отже, ітераційне проектування імітаційної моделі має мінімізувати обрані критерії E_1, E_2, E_3. Критерії E_2, E_3 можуть використовуватись і для детермінованих моделей, хоча вони мають вигляд випадкових і визначають розкид значень вихідної величини.

Складності під час оцінювання точності моделі виникають тоді, коли модельованої системи не існує або вона є недоступною для проведення експериментів. У цьому разі необхідно використовувати ті самі методи, що й під час перевірки правильності моделі з використанням контрольних завдань і залученням експертів.

Порівнюючи два або більше варіантів імітаційної моделі, звичайно не цікавляться абсолютними показниками для кожної моделі, а тільки звертають увагу на характеристики, які порівнюють. Для порівняння потрібно знати діапазони змін основних вхідних змінних і провести якісний аналіз варіантів порівнюваних моделей, як, наприклад, зображено на рис. 5.13. Передбачається, що розглядається два варіанти моделі мережі обробки інформації A та B, для яких основним критерієм

вибору системи є час її реакції на запит користувача $t_{від}$, який залежить від кількості увімкнутих терміналів N. На рис. 5.13, *а* видно, що варіант A в цьому діапазоні зміни навантаження кращий за варіант B. Якщо маємо графіки, зображені на рис. 5.13, *б*, то необхідне уточнення значень параметрів.

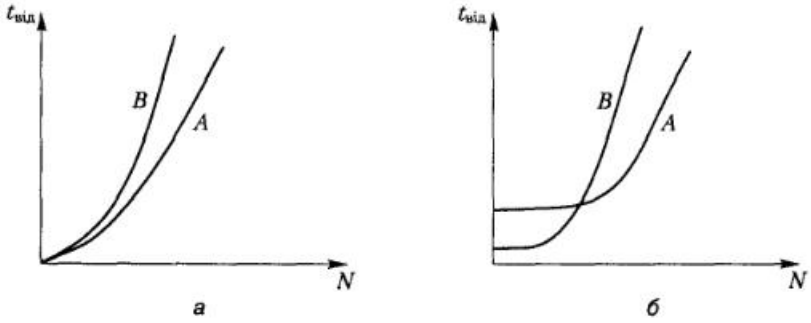


Рис. 5.13. Графіки часу реакції для різних варіантів моделей

Показники якості імітаційних моделей звичайно залежать від кількох параметрів, для визначення яких потрібно будувати складну поверхню в багатовимірному просторі. Тому дуже важливо ретельно планувати проведення експериментів для такої моделі [18].

Важливе значення має тестування моделі, за результатами якого встановлюють, чи існують у моделі похибки або помилки, а також визначають, чи функціонує вона належним чином. Деякі тести призначено для оцінювання точності поведінки (тобто, обґрунтованості) моделі, а деякі – для перевірки точності модельного перетворення з однієї форми в іншу (перевірку достовірності). Наведемо принципи, за якими слід здійснювати тестування моделі.

1. Перевірка повинна провадитись протягом усього життєвого циклу моделювання.
2. Результати перевірки не повинні розглядатись як бінарна змінна, де модель абсолютно правильна або неправильна.
3. Імітаційна модель, побудована відповідно до деякої мети, і рівні довіри до неї також повинні бути оцінені щодо тієї ж мети.
4. Перевірка повинна провадитись без упередженості до розробника.
5. Достовірність моделі можна вимагати тільки за умов, відносно яких модель перевірена, підтверджена і буде використовуватись.
6. Повне тестування імітаційної моделі неможливе.
7. Перевірка і тестування моделі повинні бути заплановані ще під час її розроблення.
8. Помилки повинні бути виявлені якомога раніше в життєвому циклі моделі.
9. Успішне тестування кожної підмоделі (модуля) не означає повної довіри до моделі.
10. Обґрунтованість імітаційної моделі не гарантує довіри до отриманих результатів імітації.

Життєві цикли моделі не повинні інтерпретуватись як строго послідовні. Наявністю неточностей, виявлених під час перевірки моделі, може бути зумовлене повернення до більш раннього етапу і проведення повторних багаторазових дій з метою удосконалення моделі. Технологія моделювання починається з визначення проблеми і закінчується отриманням результатів. Перевірки і тестування моделі дають змогу ідентифікувати і виправляти якісні неточності на усіх технологічних етапах моделювання за допомогою циклічного повторення етапів, де їх було знайдено.

Під час досліджень модель повинна математично і логічно, з визначеним ступенем наближення, відтворювати модельовану систему, процес або явище. Логічні елементи моделей повинні відповідати логічним елементам системи, а математичний апарат має відображати функції, реалізовані в модельованій системі. Вхідні дані повинні бути інформативними, тобто містити всю необхідну інформацію про модельовану систему. Під час оцінювання правильності побудови імітаційної моделі потрібно розглядати весь ланцюжок її перетворення, починаючи з вхідних даних та формального представлення і закінчуючи результатами моделювання. Тому оцінку достовірності моделі зазвичай розглядають на рівнях концепції і реалізації імітаційної моделі. Концептуальна модель розробляється й описується під час неперервного спостереження за об'єктом моделювання і навколишнього середовища, тому для перевірки достовірності концептуальної моделі використовують метод обернених перетворень моделі.

Враховуючи те, що розробка концептуальної моделі починається з постановки задачі і закінчується створенням концептуальної моделі, перевірку починають з аналізу концептуальної моделі, повертаються до прийнятих апроксимацій і спрощень, а потім розглядають реальний процес. Для перевірки концептуальної моделі залучають експертів, які не брали участі в розробленні моделі.

Перевірка достовірності моделі на рівні її реалізації пов'язана з розглядом наступного ланцюжка її перетворень. Починають з логічної схеми (алгоритму роботи моделі), переходять до схеми програми і потім розглядають саму програму моделі. Фактично на цьому етапі потрібно відповісти на такі запитання:

- Чи дає можливість модель виконати поставлене завдання?
- Чи точно відтворена модель у логічній схемі?
- Чи є повною запропонована логічна схема і чи притаманна їй необхідна послідовність?
- Чи правильно використано математичні рівняння?

Далі переходять до перевірки програмної реалізації. Для цього порівнюються функції, виконувані програмою, і логічні схеми. Для перевірки достовірності програми її знову переводять у логічну схему, потім перевіряють окремі модулі програми і всю програму в цілому, використовуючи для цього засоби тестування і налагодження (трасування) зі спеціально підібраними тестами.

На заключному етапі перевірки правильності моделі здійснюються контрольні прогони, а потім робочі розрахунки. Контрольні прогони використовуються для оцінювання чутливості моделі до зміни вхідних даних, а також для перевірки правильності роботи моделі.

Аналіз чутливості тісно пов'язують з точністю результатів порівняння статистичних гіпотез. Для цього можна використати планування проведення експерименту і статистичні критерії для оцінювання відгуків у різних варіантах моделей.

Остаточне налагодження моделі здійснюється на контрольних завданнях, які готуються незалежно від засобів реалізації моделі. Збіг результатів моделювання і контрольних варіантів свідчить про відповідність і коректність моделі до завдань даного типу. Контрольні варіанти можна побудувати зі спрощеними вхідними даними, для яких заздалегідь відомо, як повинна поводитись модельована система і які результати повинні бути на виході.

З цією ж метою імітаційну модель може бути спрощено до такого рівня, який дозволить побудувати аналітичну модель. Тоді можна порівнювати результати імітаційного моделювання й аналітичні розрахунки, але тільки в межах спрощеної моделі.

Для іншого методу перевірки адекватності використовують контрольні завдання, визначені за результатами роботи реальної системи, якщо вона існує. У разі виявлення чутливості моделі до деяких вхідних даних необхідно уточнити ці дані і відкоригувати ті блоки моделей, на які вони впливають більшою мірою, тобто перейти до детальнішого опису цих блоків. Чим детальніше побудовано модель, тим ширша сфера її застосування і тим стійкіша вона до зміни вхідних даних.

Висновки

- Імітаційне моделювання можна застосовувати як універсальний підхід для прийняття рішень в умовах невизначеності та для врахування в моделях факторів, які важко формалізуються.
- Імітаційне моделювання включає такі взаємопов'язані етапи:
  - формулювання проблеми і змістовна постановка задачі;
  - розроблення концептуальної моделі;
  - розроблення і програмна реалізація імітаційної моделі;
  - перевірка правильності та достовірності моделі;
  - організація та планування проведення експериментів;
  - прийняття рішень за результатами моделювання.
- У змістовній постановці задачі формулюються цілі моделювання, описуються основні характеристики системи, вхідні та вихідні змінні, їх взаємозв'язок, зовнішні впливи на систему, а також визначаються основні критерії функціонування системи та обмеження.
- Концептуальною називається абстрактна модель, яка виявляє причинно-наслідкові зв'язки, властиві досліджуваному об'єкту в межах, визначених цілями дослідження. Це формальний опис об'єкта моделювання, який відображає концепцію (погляд дослідника на проблему).
- Під формалізацією розуміють такий опис моделі, який припускає використання математичних методів дослідження, тобто на завершення цього етапу розроб-

ляється логіко-математичний опис модельованої системи з урахуванням динаміки її функціонування.

- Мови моделювання залежно від способу визначення внутрішньосистемного часу традиційно поділяють на три групи: неперервні, дискретні та неперервно-дискретні (комбіновані).
- Терміном *модельний час* називають арифметичну величину, яка має додатні зростаючі значення та під час моделювання відображає плин часу в моделі.
- Побудова діаграми подій – ефективний проміжний етап перетворення концептуальної моделі в програму імітаційної моделі; він дає змогу здебільшого пропускати етап алгоритмізації моделі та переходити безпосередньо до розроблення програми.
- Обґрунтування адекватності моделі доводить, що модель, у межах сфери застосування працює із задовільною точністю, сумісною з метою моделювання.

Контрольні запитання та завдання

1. Які засоби формалізованого зображення можуть використовуватись для концептуальних та імітаційних моделей? Наведіть приклади.
2. Якими параметрами імітаційної моделі можна характеризувати вибраний рівень деталізації? Чому детальніша модель дорожча? Чому вона більш стійка до змін вхідних даних?
3. Перерахуйте основні затрати на розроблення та експлуатацію імітаційної моделі.
4. Як можна відобразити структуру імітаційної моделі? Чи залежить структура моделі від вибраних засобів моделювання?
5. У чому полягають істотні відміни моделювання, орієнтованого на події та процеси? Наведіть приклади подій та процесів для СМО.
6. Які проблеми виникають під час розробки засобів автоматизації побудови моделей? Наведіть види цих засобів та назвіть їх переваги.
7. Побудуйте схему алгоритму і програму моделювання СМО з одним пристроєм. Оцініть трудомісткість її програмування алгоритмічною мовою програмування та мовою GPSS. Які переваги має програмування мовою GPSS?
8. Від чого залежить точність результатів моделювання? Проаналізуйте цю проблему, починаючи від етапу збору вхідних даних для імітаційної моделі. Як пов'язані між собою точність і адекватність моделі?

Розділ 6

Програмне забезпечення імітаційного моделювання

- Принципи побудови мов моделювання
- Історія розвитку засобів імітаційного моделювання
- Класифікація програмних засобів імітаційного моделювання
- Універсальні та об'єктно-орієнтовані системи моделювання
- Методи штучного інтелекту, що застосовують в імітаційному моделюванні

6.1. Принципи побудови мов моделювання

Мови моделювання залежно від способу задання внутрішньосистемного часу традиційно поділяють на три групи: неперервні, дискретні та неперервно-дискретні (комбіновані).

Неперервні мови призначено для моделювання неперервних у часі процесів. У цьому випадку подання об'єкта моделювання зводиться до складання диференціально-різницевих і (або) алгебричних рівнянь, які пов'язують вихідні змінні із вхідними.

Дискретні мови призначено для моделювання дискретних процесів. Відмітною рисою мов даної групи є те, що вони повинні мати засоби, за допомогою яких здійснюється керування процесом моделювання. Для цього використовуються динамічні списки подій, в яких останні впорядковані в модельному часі. Терміном *модельний час* називають арифметичну величину, яка має додатні зростаючі значення та під час моделювання відображає хід часу в моделі. Останній може задаватись шляхом збільшення з постійним кроком (*принцип* Δt) або з випадковим кроком, який дорівнює інтервалу між двома послідовними подіями (*принцип особливих станів* δz). В останньому випадку йдеться лише про ті особливі події, які переводять систему зі стану z_i у стан z_j у просторі станів Z. Наприклад, для СМО такими особливими станами будуть моменти надходження вимог до системи та звільнення пристроїв для обслуговування.

Допоміжні події, на відміну від особливих, не змінюють стан системи та системний час. Вони можуть відбуватись тільки в разі виконання деяких умов. Прикладом такої події в СМО може бути зміна позиції вимог у черзі до пристрою для обслуговування та заняття пристрою в разі його звільнення (див. рис. 2.13).

Принципи побудови дискретних мов моделювання відповідають принципам створення моделей динамічних систем, зображених на рис. 1.8. Традиційно дискретні мови моделювання орієнтовані на планування подій, перегляд (сканування) видів діяльності та процесів. Кожний з цих способів має свій власний механізм відображення модельованих ситуацій, тобто алгоритм моделювання, за яким складається програма керування.

6.1.1. Мови, орієнтовані на події

Згідно з підходом, орієнтованим на події (див. модель (1.8) із розділу 1.8.3), програма керування моделюванням веде хронологічно упорядкований список подій, запланованих на деякий час. Модельний час змінюється від події до події за принципом особливих станів. Для кожної події існує своя підпрограма обробки події, яка викликається ПКМ, коли настає подія даного типу. Ця сама підпрограма породжує одну або кілька нових подій та планує їх появу. За подійним підходом об'єкт моделюється шляхом ідентифікації змін, які відбуваються в ньому в момент звершення подій.

Під час розробки імітаційної моделі завдання розробника полягає в описі подій, які можуть змінити стан системи, та у визначенні логічного взаємозв'язку між ними. У цьому випадку імітація функціонування системи зводиться до впорядкованого в часі виконання підпрограм обробки логічно пов'язаних між собою подій (див. розділ 2.6.4).

Згідно з подійним підходом поведінка модельованого об'єкта відтворюється шляхом здійснення подій протягом модельного часу. Розробка імітаційної моделі зводиться до задання опису подій, які змінюють стан системи, і визначення логічного взаємозв'язку між ними. Наприклад, підпрограма запиту та призначення ресурсу (див. рис. 2.12) не може викликати підпрограму генерування вимог, тому що ці події не зв'язані між собою. Вона може викликати підпрограму організації черги в момент звільнення ресурсу, щоб перевірити, чи є вимоги в черзі.

Структура програми на мові моделювання, орієнтованій на події, часто не має нічого спільного зі структурою реального об'єкта. Причина у тому, що настання події може бути викликано багатьма компонентами об'єкта.

6.1.2. Мови, орієнтовані на певні види діяльності

Використовуючи підхід, орієнтований на певні види діяльності (див. модель (1.9) із розділу 1.8.3), ПКМ викликає підпрограми видів діяльності, коли виконується умова початку або закінчення даного виду діяльності. На відміну від попереднього підходу, події не плануються, а модельний час постійно збільшується. Програма керування моделюванням має переглянути всі програми видів діяльності протягом модельного часу. Тому такий підхід до організації мови моделювання також називають скануванням, або послідовним переглядом видів діяльності.

Цей підхід доцільний, якщо тривалість дії визначається деякими ситуаціями, виникнення яких залежить від того, наскільки стан системи задовольняє певні умови. Мови моделювання, які базуються на цьому принципі, найефективніше

можуть використовуватись для ситуаційного моделювання (див. модель (1.10) із розділу 1.8.3). Але порівняно з подійним такий підхід менш ефективний, оскільки доводиться сканувати умови для кожної дії. Прискорити моделювання можна, застосувавши метод змінних збільшень часу, тобто комбінацію подійного підходу та підходу зі скануванням видів діяльності. У цьому разі ПКМ переглядає список подій, вибирає найближчий момент часу, на який заплановано подію, та виконує всі підпрограми подій, умови яких виконані. Потім годинник модельного часу переводиться на момент часу наступної події.

6.1.3. Мови, орієнтовані на процеси

Для мов моделювання, орієнтованих на процеси (див. модель (1.7) із розділу 1.8.3), характерна наявність у них деяких динамічних об'єктів (повідомлень, транзактів), які рухаються по моделі за визначеною схемою. *Повідомлення* (*транзакт*) – абстрактна динамічна структура з набором атрибутів, які несуть інформацію про реальний динамічний об'єкт. Наприклад, повідомленнями в СМО можуть бути вимоги, атрибутами для них – пріоритети, число фаз обслуговування та ін. Програми процесів описують шляхи проходження повідомлень у системі.

У міру свого просування повідомлення «споживають» деякі ресурси. Під *ресурсом* розуміємо абстрактну структуру з набором атрибутів, яка характеризує спосіб доступу до неї та її фізичне відображення (у книжці [20] такі об'єкти називаються активностями). Якщо ресурс зайнятий одним повідомленням і потрібний у даний момент часу для іншого повідомлення, то до ресурсу виникає черга. Ресурси можуть розподілятися під час моделювання між деякими процесами. Такі ресурси називаються пам'ятями.

Під *процесом* розуміємо послідовність подій, зв'язок між якими встановлюється за допомогою спеціальних відношень, тобто функцій дій та початкових станів у просторі станів. Кожен стан процесу – це ніби його «моментальний знімок». Основна увага у разі моделювання за допомогою процесів зосереджується на історії проходження повідомлень, тобто на шляху їх руху та на взаємодії паралельних процесів.

Згідно з прагматичним підходом *процес – це програмний блок*, складений з ряду процедур (підпрограм), які описують поведінку повідомлень. Шлях руху повідомлення задається порядком звернення до процедур.

Поява повідомлення в програмі задається виконанням деякої процедури, яка називається генеруванням повідомлень. Під час генерування для кожного повідомлення (транзакта) створюється своя структура даних – запис. Цей запис зазвичай містить такі поля:

- назва або номер повідомлення;
- час виникнення повідомлення;
- назва або номер процедури (програмного блока), де знаходиться повідомлення;
- назва або номер наступної процедури (програмного блока), куди має перемістити повідомлення;

- ✦ запланований модельний час переміщення повідомлення;
- ✦ пріоритет повідомлення;
- ✦ параметри, які задають атрибути повідомлення.

У процесі моделювання повідомлення може створювати нові повідомлення або розділятися на кілька повідомлень, тоді в запис повідомлення додається поле з назвою або номером родини, або ансамблю, повідомлень. Після закінчення історії повідомлення воно знищується (знищується запис про повідомлення), тобто виконується останній оператор у програмному блоці. Атрибути повідомлення задаються параметрами (полями в запису) під час їх породження.

Структура програм реалізації імітаційних моделей, написаних мовами моделювання, орієнтованими на процеси, як правило, співпадає зі структурою об'єкта моделювання. Це суттєво спрощує аналіз та інтерпретацію результатів моделювання.

У мові GPSS [68] використовується динамічний об'єкт, який називається транзактом. Для його генерування служить блок GENERATE, а для знищення – блок TERMINATE. Частина програми, розміщена між цими двома блоками, називається програмним сегментом. У ньому описується історія руху транзактів. Блок SPLIT дає змогу розділити транзакти. Параметри транзактів задаються блоками ASSIGN.

Аналогами пристроїв для обслуговування реальних систем у мові GPSS є об'єкти типу *ресурси*. До об'єктів цього типу відносять пристрої (блоки SEIZE– RELEASE), багатоканальні пристрої (блоки LEAVE– ENTER), логічні перемикачі (блок LOGIC).

За допомогою об'єктно-орієнтованих мов моделювання можна описувати класи однотипних моделей. Моделі в цих мовах створюються динамічно, тому моделі можуть бути процесами. Моделі можуть також породжувати підмоделі. Обов'язковою умовою є наявність зовнішньої моделі. Приклад ієрархії моделей обчислювальної мережі наведено на рис. 6.1. У квадратних дужках зазначено кількість екземплярів моделей. Моделі та повідомлення є самостійними процесами, тоді як методи приєднуються до того самостійного процесу, яким вони були породжені.

Рис. 6.1. Ієрархія моделей обладнання обчислювальної мережі

6.2. Квазіпаралельна робота програм у модельному часі

У різні моменти модельного часу повідомлення можуть виконувати різні оператори програми моделювання. Досягається це за рахунок квазіпаралельної роботи програми в системному часі, тобто годинник модельного часу не змінить значення модельного часу доти, доки не буде оброблено всі події для даного моменту модельного часу. Кільком подіям у моделі може відповідати один момент модельного часу. Це дає змогу відобразити одночасні події, які відбуваються в модельованій системі. Для того щоб у цьому випадку не виникали конфлікти під час визначення черговості виконання подій, установлюють деяке правило обробки одночасних подій. Здебільшого ці події обробляються в порядку їх планування, тобто в такому порядку, в якому програми для цих подій призначили моменти модельного часу. Змінюють порядок обробки подій за допомогою пріоритетів, які є атрибутами повідомлень. Найбільше конфліктів виникає в мовах моделювання з цілочисловими значеннями модельного часу, оскільки в цьому разі через округлення значень часу зростає ймовірність настання двох або більше подій в один і той самий момент модельного часу.

6.3. Стани процесів

Під час виконання програми моделювання процеси можуть перебувати в чотирьох станах, відображених на рис. 6.2: активному, зупиненому, пасивному та завершеному.

Рис. 6.2. Стани процесів програми моделювання:
1 — виконання умов або переведення годинника; *2* — планування майбутніх подій;
3 — виконання дії; *4* — припинення виконання процесу; *5* — завершення виконання процесу

Процес є *активним*, якщо повідомлення в даний момент часу виконує процедуру, яка описує його поведінку в моделі. Таке повідомлення називається поточним. Водночас у системі може бути тільки одне активне повідомлення, тому що центральний процесор може виконувати в кожний момент часу тільки одну команду програми (йдеться тільки про однопроцесорні комп'ютери, на яких реалізується програма моделювання). Під час виконання процедури процес можна *зупинити* (перевести до стану чекання) до визначеного моменту часу або до виконання умови його активізації. Процес перебуває в *пасивному* стані, якщо невідомі час або умова його активізації, але він ще не *завершений*, тобто не закінчив виконання своєї процедури і не знищений, а існує в системі тільки як структура даних.

6.4. Організація керування процесом моделювання

Під час моделювання систем послідовності відіграють значну роль і використовуються для організації черг до ресурсів, а також, наприклад, для послідовностей записів, які відображають списки подій. Взагалі будь-який динамічний об'єкт (повідомлення або транзакт) в імітаційній моделі задається як *запис*, в полях якого вказуються *ідентифікатор* об'єкта та його *атрибути*. Таким чином, під час моделювання виникає необхідність маніпулювати послідовностями, і перш ніж розглянути організацію керування процесом моделювання, слід згадати, яким чином здійснюється оперування послідовностями в комп'ютері.

Для зберігання в комп'ютері скінченних послідовностей використовують зазвичай масиви, визначивши розмір масиву як кількість членів послідовності. У пам'яті комп'ютера такий масив зберігається як неперервна послідовність адрес комірок пам'яті, в яких зберігаються значення або записи даних послідовності. Для того щоб звернутись до елемента масиву, досить зазначити тільки його *індекс* у масиві, який можна зобразити як номер рядка в матриці.

Масиви вигідні для пошуку будь-якого *i*-го елемента масиву. Масиви незручні, коли нові елементи додають у середину списку, або коли список необхідно перебудувати. Крім того, масиви звичайно мають постійний розмір, визначений під час компіляції або після початкового розподілу пам'яті. В процесі моделювання максимальну кількість записів для будь-якого списку буває важко або неможливо визначити заздалегідь, тому що поточне число елементів у списку може значно змінюватись протягом виконання імітації. Найгірше те, що більшість систем моделювання вимагає великої кількості списків, що призводить до надмірних витрат комп'ютерної пам'яті.

У тому випадку, коли кількість елементів послідовності невідома, необхідно мати *динамічні масиви*, або *списки*, для яких пам'ять у комп'ютері виділяється динамічно з різних (не обов'язково неперервних) адрес комірок пам'яті, як це виконується в сучасних засобах імітаційного моделювання. Для того щоб пов'язати між собою окремі комірки, необхідно мати змінні типу посилання, значенням яких є адреса оперативної пам'яті. Для цього використовується *покажчик*, який може посилатись на інший покажчик (рис. 6.3). Таким чином організуються списки. *Список* – це впорядкований набір або впорядковані записи, які мають *вершину*, або *голову* (перший елемент у списку) і для яких характерний деякий спосіб обходу для пошуку інших елементів. Крім того, необхідно позначити *кінцевий елемент* (*кінець*) списку. *Головний покажчик* – змінна, що показує на запис вершини списку. Деякі списки мають *покажчик кінця* списку, який вказує на останній елемент. Кожний запис у списку має поле, що містить посилання на наступний елемент списку. Деякі списки мають ще одне поле, в якому є посилання на попередній елемент списку, що дає змогу обходити список у зворотному напрямку, від кінця до вершини.

Така організація списків дає змогу виконувати прогін моделі як у прямому, так і зворотному модельному часі.

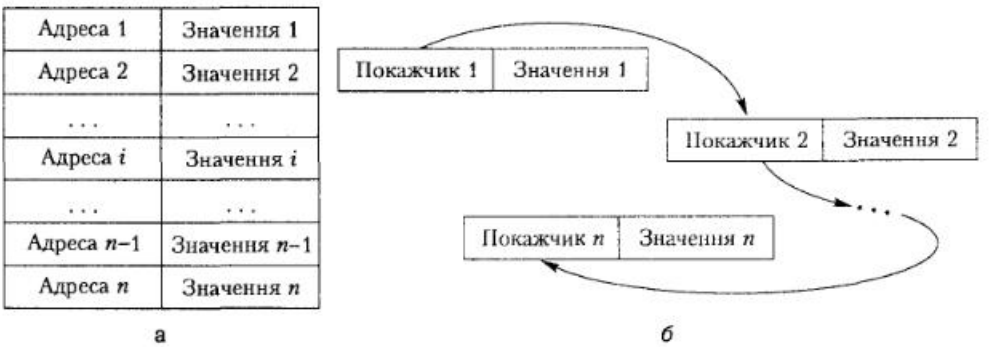


Рис. 6.3. Організація списків різних типів: масив комірок у пам'яті (а); лінійний список (*б*)

Для будь-якого типу списку головними діями під час його обробки є внесення запису до нього і вилучення запису із нього. Головні операції з елементами списку такі:

- вилучення запису з вершини списку;
- вилучення запису з будь-якого місця в списку;
- внесення запису в кінець списку;
- внесення запису в довільну позицію списку, визначену відповідно до правила.

Першу і третю операції можна виконати за мінімальний час, регулюючи два покажчики запису: голову або покажчик кінця, інші дві операції вимагають принаймні неповного пошуку по списку. Створення цих двох ефективних операцій – ціль методів оброблення списків.

Згідно з підходом до планування подій, коли модельний час повинен бути переведеним на майбутню подію і підпрограма події повинна бути виконаною, спочатку має місце операція з вилучення події із вершини СМП. Якщо довільна подія скасовується або об'єкт вилучається зі списку, заснованого на деякому з його атрибутів (наприклад, пріоритеті або терміні виконання) до початку дії, то виконується друга операція з вилучення. Коли виконується приєднання об'єкта до кінця черги, утвореної як список за правилом FIFO, тоді згідно з третьою операцією об'єкт вноситься в кінець списку. Нарешті, якщо побудова черги визначається за правилом «самий ранній термін – перший», то після прибуття в чергу об'єкт має бути внесеним до списку не у вершину або кінець, а в позицію, обумовлену терміном для правила упорядкування.

Під час моделювання функції керування процесами та їх синхронізації виконує ПКМ, яка веде хронологічно впорядкований список подій. Зазвичай це циклічний двонапрямлений список (рис. 6.4). Для позначення початку та кінця списку використовується елемент голови списку (ГС). Кожний елемент списку містить інформацію про час, на який заплановано подію, посилання на попередній і наступний елементи та на процес (адресу оператора процедури або підпрограми), який має бути переведений у активний стан. Перший елемент списку вказує на *активний* у даний момент часу процес, який обробляє поточне повідомлення. У момент створення списку (нульовий момент модельного часу) елемент ГС є одночасно і попереднім елементом і спадкоємцем самого себе. Далі в цей список вбудовуються елементи,

які відображують заплановані (намічені), але ще не виконані до даного моменту модельного часу події. Отже, поточний модельний час (t = 1,6) зазначено в першому елементі списку.

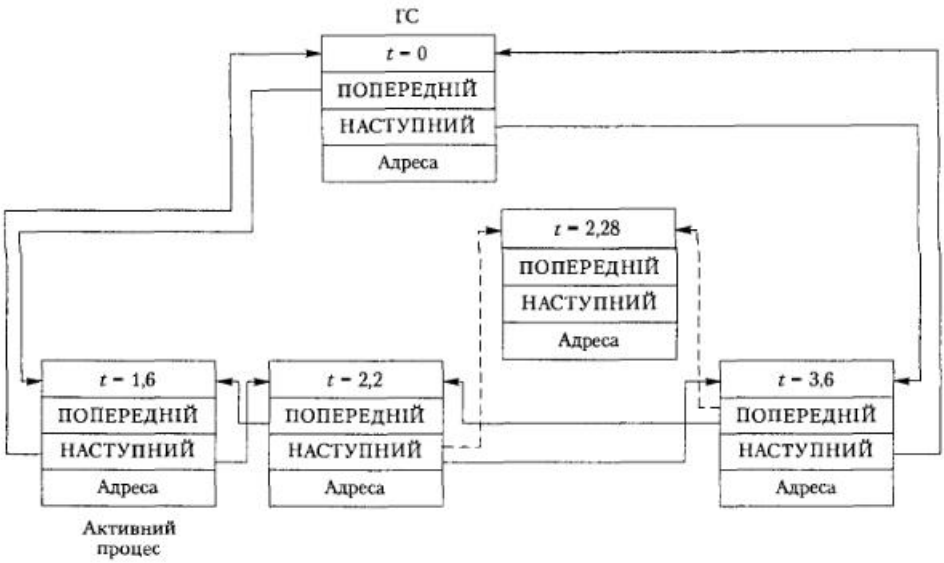


Рис. 6.4. Організація циклічного двонапрямленого списку

На рис. 6.4 зображене повідомлення з модельним часом t = 2,8, який вміщується у список активних повідомлень. Для цього повідомлення час події (t = 2,8) запланував активний процес, який виконується.

Коли процес перебуває в активному стані, він може запланувати події для інших процесів або наступний активний стан для самого себе. Наприклад, виконуючи процедуру генерування повідомлень у даний момент модельного часу, процес планує собі новий активний стан — наступний момент генерування нового повідомлення. Крім того, процес може скасувати деякі із запланованих раніше подій, які ще не відбулись. Під час планування нових подій до списку заносяться нові елементи або змінюються позиції вже наявних елементів. Відміна події відповідає вилученню елемента зі списку, тобто знищенню запису про повідомлення і звільненню динамічної пам'яті.

Існують і складніші структури організації списків [33, 86], коли списки поділяються на підсписки, якими керує ПКМ (інтерпретатор), як у мові GPSS [61] (рис. 6.5). Щоб зменшити витрати машинного часу на перегляд списків, система GPSS веде два основних списки подій. Перший — це *список поточних подій* (СПП), куди входять усі події, заплановані на поточний момент модельного часу незалежно від того, умовні вони чи безумовні. Програма керування моделюванням у першу чергу переглядає цей список і намагається перемістити по моделі ті транзакти, для яких виконано умови. Якщо жодну з умов не виконано, то ПКМ звертається до другого списку — *списку майбутніх подій* (СМП). Вона переносить усі події, заплановані на найближчий модельний час, з цього списку в СПП і поновлює

його перегляд. Таке перенесення здійснюється також у разі збігу поточного часу моделювання із часом першої події в СМП.

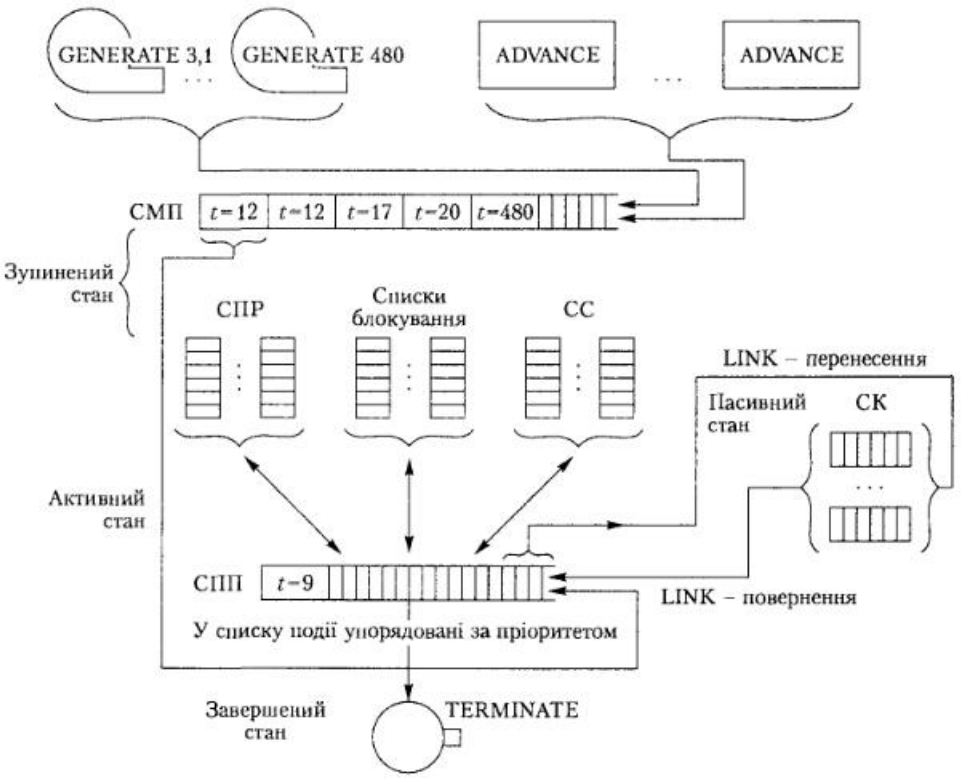


Рис. 6.5. Організація списків у мові GPSS

У СПП транзакти розміщені в порядку зменшення пріоритету (тобто транзакти, що мають більш високий пріоритет, розміщені ближче до початку списку). Транзакти з однаковими пріоритетами розміщуються у порядку надходження до списку. У СПП кожний транзакт може знаходитись або в *активному стані* (тобто переглядатись у даний модельний час), або в стані затримки.

Спочатку (під час виконання команди START, яка починає фазу інтерпретації GPSS-моделі) ПКМ звертається до всіх блоків GENERATE моделі. Кожний із цих блоків планує момент появи транзактів і заносить їх у СМП, після чого ПКМ звертається до СПП. Оскільки в списку поки що немає транзактів, ПКМ переглядає СМП, вибирає з нього всі транзакти, заплановані на найближчий час, і переносить їх у СПП, після чого намагається просунути перший транзакт цього списку по блоках моделі. Якщо переміщення транзакту було затримане з якоїсь причини, не пов'язаної з блоком ADVANCE, то він залишається в СПП і ПКМ намагається переміщувати такий транзакт зі списку далі по блоках. Якщо транзакт увійшов у блок ADVANCE, то планується його вихід з цього блока, і транзакт переноситься в СМП.

Списки поточних і майбутніх подій можна побачити на екрані монітора, якщо ввести команду EVENTS для GPSS/PC, або у вікні списків для GPSS World.

Щоб досягти високої ефективності перегляду, важливо також переглянути транзакти, рух яких заблоковано (наприклад, через зайнятість деякого ресурсу). Найпростішим рішенням є перегляд усіх заблокованих транзактів для кожного нового значення модельного часу та вибір тих, для яких знято умови блокування. Якщо модельовану систему перевантажено, то цей спосіб через значні витрати машинного часу дуже невигідний, бо кожний транзакт переглядається багаторазово до того, як вийде зі стану блокування.

Коли причиною переведення транзактів у стан блокування є стан деякого ресурсу системи в даний час, то набагато ефективнішим буде спосіб обробки, за яким заблоковані з цієї причини транзакти взагалі не переглядатимуться доти, доки не зміниться відповідний стан ресурсу. Такий спосіб можна реалізувати, наприклад, шляхом реєстрації для кожної одиниці ресурсу транзактів, рух яких заблоковано через стан саме цього ресурсу в даний час. У момент, коли змінюється стан цього ресурсу, необхідно відразу ж переглянути транзакти, які очікують на цю зміну, і відновити їх обробку.

Якщо транзакти знаходяться в активному стані, процедури перегляду намагаються перемістити їх до наступних блоків. Якщо переміщення транзакту блокується якимось ресурсом через його зайнятість, то входження до наступного блока неможливе, і транзакт переводиться в стан затримки. Такі транзакти не переглядаються і розміщуються у відповідному списку затримки.

У випадку коли під час обслуговування потокового активного транзакту сталася зміна стану ресурсу, перегляд починається спочатку і заново обслуговуються всі транзакти в СПП, які знаходяться в активному стані. Якщо зміна списків ресурсів не відбулась, то імітатор знову звертається до СПП і перевіряє, чи залишились у ньому транзакти, які необхідно обробляти.

Якщо в СПП не можна продвинути жодний транзакт по моделі, то ПКМ звертається до СМП і транзакти, заплановані на найближчий час, переносяться в кінець СПП, після чого відновлюється перегляд СПП.

Моделювання закінчується, коли лічильник завершення в команді START буде переведеним на нуль або в СПП і СМП не буде жодного транзакту.

Список блокувань – це список транзактів, які очікують на зміну стану ресурсу. Існує шість видів таких списків, пов'язаних із пристроями, сім – з багатоканальними пристроями (БКП) і два – з логічними ключами. З пристроями використовуються списки для зайнятого, незайнятого, доступного, недоступного пристроїв, а також для пристроїв, що працюють без переривання і з перериванням. З БКП використовуються списки для заповненого, незаповненого, порожнього, непорожнього, доступного, недоступного БКП і транзактів, які можуть увійти в БКП. З логічними ключами пов'язані списки для увімкнених і вимкнених ключів.

Крім двох основних списків, СПП і СМП, існує *список переривань* (СПР), який містить перервані під час обслуговування транзакти, а також транзакти, які викликали переривання. Список переривань використовується для організації обслуговування одноканальних пристроїв за абсолютними пріоритетами. У ньому знаходяться транзакти, перервані під час обслуговування. Це дає змогу організовувати пріоритетні дисципліни обслуговування транзактів у пристроях.

Список синхронізації (СС) містить транзакти, які в даний час порівнюються. Цей список працює з транзактами, створеними за допомогою блока SPLIT, який створює копії транзактів, що належать до однієї сім'ї або ансамблю. Три блоки синхронізації працюють тільки з транзактами, отриманими за допомогою блока SPLIT. Синхронізацію руху транзактів однієї сім'ї виконують такі блоки: MATCH (синхронізує рух транзактів з іншим блоком MATCH), ASSEMBLE (збирає всі копії транзактів та видає один транзакт-предок), GATHER (збирає задану кількість транзактів і затримує їх копії доти, доки не буде зібрано необхідної кількості копій транзактів).

Блок SPLIT можна використовувати багаторазово, створюючи тим самим ієрархію копій транзактів.

Зупинені процеси знаходяться в СМП, СС і списках блокувань.

Список користувача (СК) містить транзакти, виведені користувачем зі СПП за допомогою блока LINK і вміщені в СК як тимчасово неактивні (переведені користувачем у *пасивний стан*). Під час роботи ПКМ вони їй недоступні доти, доки вони не будуть повернуті користувачем у СПП за допомогою блока UNLINK (ПОВЕРНУТИ).

Моделювання закінчується тоді, коли лічильник завершення в команді START буде переведено на нуль або коли в СПП і СМП не буде жодного транзакту.

6.5. Системи планування в мовах моделювання

У мовах, орієнтованих на процеси та дії, використовуються інтерогативні системи планування на відміну від мов моделювання, орієнтованих на події, де застосовується система імперативного планування [20]. Імперативне планування передбачає наявність у мові конструкцій типу ЧЕКАТИ ВИКОНАННЯ УМОВИ A, де A – предикат. Під час імперативного планування можна застосовувати конструкції тільки типу ЧЕКАТИ ЧАС T, де T – заданий момент часу. У цьому разі предикат істинний за умови, якщо модельний час дорівнює часу, запланованому за допомогою оператора ЧЕКАТИ ЧАС T. Системи імперативного планування мають можливості планування інтерогативних систем, однак обернене твердження несправедливе. Отже, у мовах моделювання, орієнтованих на процеси, можна застосовувати обидві наведені вище конструкції.

У системах з інтерогативним плануванням існує необхідність перевіряти умови, задані предикатами. Це призводить до значного зниження швидкості моделювання. Така перевірка може здійснюватись з кроком Δt або при кожній зміні модельного часу. Останній спосіб більш швидкий, але і в цьому разі при кожній зміні модельного часу необхідно перевіряти виконання умов для всіх процесів моделі.

Для інтерогативного планування доводиться звичайно вести два системних списки: хронологічно впорядкований безумовних подій і умовних подій. Програма керування моделюванням встановлює значення модельного часу таким, що дорівнює часу першої запланованої події, та викликає виконання всіх підпрограм процесів, запланованих на цей момент часу. Потім вона переглядає список умовних подій та активізує всі процеси, умови виконання яких задовольняються. Такий

перегляд триває доти, доки не залишаться тільки умовні події, які не можна було б виконати (настання одних умовних подій може вплинути на настання інших). Після цього годинник модельного часу переводиться на наступну безумовну подію.

У мові GPSS для затримання транзактів використовується блок ADVANCE (оператор імперативного планування). Оператором інтерогативного планування є блок GATE, який затримує транзакт до виконання умови.

Слід зауважити, що зазначені особливості організації керування та синхронізації процесів є внутрішніми засобами мов моделювання і що вони приховані від розробника моделей. Розробник має тільки описати чергування подій для кожного процесу засобами мови моделювання, але важливо усвідомлювати, яким чином ці події обробляються мовою моделювання [86].

Отже, мовам, орієнтованим на процеси, властиві найбільш розвинуті засоби конструювання імітаційних моделей. Тому їх доцільно використовувати під час розробка складних імітаційних моделей або комплексів моделей, орієнтованих на вирішення деяких проблем.

Основним критерієм під час вибору засобів моделювання має бути економічний, який мінімізує витрати на розробки імітаційної моделі та проведення експериментів з нею.

Для створення імітаційних моделей з використанням алгоритмічних мов необхідні уніфіковані програмні модулі, які забезпечили б проходження модельного часу, генерування випадкових чисел, керування процесами, збирання та обробку статистичних даних тощо. Ці засоби вбудовано в мови та пакети імітаційного моделювання, історія розвитку і перспективи використання яких розглядаються нижче.

6.6. Історія розвитку засобів імітаційного моделювання

Визначальними чинниками в історії розвитку програмного забезпечення імітаційного моделювання були покоління мов моделювання. Протягом більш як сорока років змінювалися концепції, парадигми програмування, комп'ютерні платформи, що істотно вплинуло на особливості відповідних етапів розвитку.

Відомі фахівці в галузі імітаційного моделювання, такі як Р. Ненсі і Ф. Ківіат, у своїх працях визначали кілька етапів розвитку засобів імітаційного моделювання.

Етап 1 (1955–1960 роки)

Імітаційні моделі розроблялись на основі таких загальновідомих універсальних мов програмування, як FORTRAN і ALGOL. У зв'язку з переходом від програм, написаних у кодах комп'ютерів, до мов програмування загального призначення значно розширилося коло користувачів і коло завдань, які можна було розв'язувати за допомогою засобів імітаційного моделювання. Мова FORTRAN застосовувалась для розроблення великої кількості алгоритмів моделювання, бібліотеки яких і сьогодні є актуальними в науковому світі. У мові ALGOL вперше було введене поняття процедури, що в подальшому внесло корінні зміни в концепцію програмування і відкрило шлях до об'єктного програмування.

Етап 2 (1961–1965 роки)

З'явились перші мови моделювання, а саме: GPSS, SIMSCRIPT, SIMULA, CSL, SOL. Розроблено так звану концепцію світогляду (world view) дискретних мов моделювання, згідно з якою ці мови визначались як орієнтовані на події, дії та процеси. Поява спеціалізованих мов моделювання значно спростила процес створення і дослідження імітаційних моделей. Користувачам не потрібно було самотужки створювати окремі програми для керування процесом моделювання, відслідковуючи події та організовувати збір статистичних даних за результатами моделювання. Ці проблеми вирішувались за допомогою засобів мови моделювання. Використання мов дало змогу також уніфікувати ряд об'єктів, необхідних для побудови імітаційних моделей. У першу чергу такими об'єктами стали накопичувачі або пристрої пам'яті та обслуговування, черги тощо. Крім того, мови мали засоби генерування динамічних об'єктів, вбудовані генератори випадкових чисел і величин. Всі ці засоби загалом дали змогу значно скоротити час створення та налагодження програм імітаційного моделювання.

Етап 3 (1966–1970 роки)

З'явилось друге покоління мов моделювання – GPSS V, SIMSCRIPT II.5 [79], SIMULA 67 [4], орієнтованих на комп'ютери третього покоління. Слід відзначити появу мови SIMULA 67 – мови загального призначення, а не моделювання. У ній уперше було введено поняття об'єкта та дії, що виконується об'єктом, тобто послідовності операторів (сценаріїв), які описують поведінку або функціонування об'єкта. Також було введено поняття класів, як множин споріднених об'єктів, та ієрархії класів. Об'єкти імітаційної моделі могли функціонувати квазіпаралельно в часі. Ієрархія квазіпаралельних систем стала якісно новим ступенем структурованого програмування. Ці особливості мови SIMULA 67 заклали підґрунтя об'єктно-орієнтованого програмування. Крім того, ця мова мала системні класи, які забезпечували роботу зі списками та сукупністю даних і моделювання дискретних систем. Таким чином, мова SIMULA 67 випередила свій час більш ніж на 20 років.

Етап 4 (1971–1978 роки)

Етап розвитку та вдосконалення вже розроблених мов і засобів моделювання з метою підвищення ефективності процесів моделювання і перетворення його в більш простий і швидкий метод дослідження складних систем. З'явились неперервно-дискретні мови моделювання, такі як ACSL, що дало змогу використовувати в одній імітаційній моделі як неперервні, так і дискретні компоненти.

Для розв'язання проблеми таксономії імітаційних моделей велике значення мали роботи Зейглера (Zeigler) і Ерена (Oren) – вони ввели метаконцепції моделі та схеми експерименту.

Етап 5 (1979–1984 роки)

Перехід від програмування моделей до розвитку технологій моделювання. Основний акцент було перенесено на створення інтегрованих засобів імітаційного моделювання. З появою імітаційних моделей змінилас концепція моделювання; воно стало розглядатись як єдиний процес побудови і дослідження моделей, що

має програмну підтримку. Наріжним каменем стає формальне поняття моделі, яке не лише пояснює динаміку системи, але може бути предметом математичних досліджень. Стає можливим аналіз достовірності багатьох практично важливих властивостей моделі (стаціонарних розподілів, малих імовірностей, чутливості, надійності і достовірності результатів моделювання). Ці властивості відіграють важливу роль під час дослідження великомасштабних систем, де ціна помилки особливо висока.

Етап 6 (1985–1994 роки)

Програмне забезпечення імітаційного моделювання перенесене на персональні комп'ютери, що використовують засоби графічного інтерфейсу (для візуалізації та анімації процесів моделювання).

Етап 7 (1995–1998 роки)

Розроблено засоби технологічної підтримки процесів розподіленого імітаційного моделювання на мультипроцесорних обчислювальних системах і в мережах.

Існують й інші підходи до визначення етапів розвитку програмних засобів імітаційного моделювання. Наприклад, у праці [72] виділено такі етапи:

- пошук (1955–1960 роки);
- звершення (1961–1965 роки);
- формування (1966–1970 роки);
- розповсюдження (1971–1978 роки);
- об'єднання та регенерація (1979–1986 роки);
- сучасний розвиток (з 1987 року).

Б. Шмідт (B. Schmidt) пропонує вирізняти п'ять поколінь програмних засобів моделювання:

- перше (50-ті роки XX сторіччя; мови FORTRAN, ALGOL) – програмування моделей мовою високого рівня без будь-якої спеціальної підтримки завдань моделювання;
- друге (60-ті роки XX сторіччя, мови GPSS, SIMULA, SIMSCRIPT) – спеціалізована підтримка моделювання у вигляді відповідних виразів мови, генераторів випадкових чисел, засобів подання результатів;
- третє (70-ті роки XX сторіччя; мова ACSL) – можливості проведення неперервно-дискретного моделювання в одній системі;
- четверте (80-ті роки XX сторіччя; мови SIMFACTORY, XCELL) – орієнтація на конкретні галузі, можливість анімації;
- п'яте (90-ті роки XX сторіччя; мови SIMPLEX II, SIMPLE++) – графічний інтерфейс, інтегроване середовище для створення і редагування моделей, планування проведення експериментів, керування моделюванням та аналіз результатів.

Запропоновані класифікації етапів мало чим відрізняються одна від одної. У двох останніх зроблено акцент на конкретних особливостях створених засобів моделювання, у той час як перша дає узагальнене уявлення про мови моделювання.

6.7. Розвиток технологій імітаційного моделювання в Україні

Наведений нижче огляд етапів розвитку засобів імітаційного моделювання в Україні зроблено на базі публікації [57].

Значний вклад у розвиток засобів імітаційного моделювання зробили фахівці Інституту кібернетики Академії наук України. Протягом 1966–1968 років під керівництвом Т. П. Мар'яновича провадились роботи зі створення мови та системи моделювання з дискретними подіями СЛЭНГ (автор мови СЛЭНГ – Л. А. Калениченко). Цими розробками покладено початок розвитку засобів імітаційного моделювання в Україні. Як прототип було обрано мову SOL.

Створена в Інституті кібернетики система СЛЭНГ використовувалась більш ніж у 20 різних організаціях і була впроваджена на таких моделях комп'ютерів, як М-20, М-220, БЭСМ-3М, БЭСМ-4М. Ця система застосовувалась під час розробки компонентів обчислювальних машин і систем, призначених для виконання завдань планування виробництва, для оцінювання показників надійності складних систем тощо.

У 1973 році в Інституті кібернетики під керівництвом д-ра техн. наук В. В. Литвинова було завершено роботи зі створення і реалізації на комп'ютері БЭСМ-6 системи АЛСИМ-БЭСМ [32]. Система призначалась для дослідження обчислювальних систем і мереж. У системі було виділено три рівні мови: опису моделей, керування моделюванням, керування завданнями. Ця система застосовувалась під час виконання завдань радіолокації, протиповітряної оборони, аналізу і розподілу ресурсів.

Протягом 1973–1975 років у Інституті кібернетики виконувались роботи зі створення мови моделювання НЕДИС [49] і відповідної імітаційної системи, призначеної для моделювання широкого класу дискретних, неперервних і неперервно-дискретних систем (керівники розробки – Т. П. Мар'янович і В. У. Гусєв; В. В. Гусєв – автор мови НЕДИС). У процесі розробки системи НЕДИС широко використовувався досвід, набутий під час упровадження системи СЛЭНГ.

За своїми можливостями система НЕДИС подібна до систем, побудованих на базі таких мов, як SIMULA-67 і GASP-IV. Система використовувалась для проектування обчислювальних машин і систем, передавання даних, моделювання надзвичайних ситуацій у вугільних шахтах і процесів на залізничному транспорті, під час планування ремонтних і профілактичних робіт у авіабудуванні, проектуванні засобів і систем космічної техніки, оптимізації технологічних процесів у суднобудуванні.

У 1979–1980 роках під керівництвом В. В. Литвинова було створено моделювальний комплекс АЛСИМ-2, реалізований пізніше на комп'ютерах серії ЄС ЕОМ. Математичне забезпечення комплексу АЛСИМ-2 давало змогу виконувати різноманітні завдання проектування обчислювальних систем і мереж. Комплекс моделювання АЛСИМ-2 включав дві підсистеми. До першої належали засоби планування проектування та керування ним і засоби генерування документальної частини

проекту. Друга підсистема забезпечувала вирішення задач проектування. До складу комплексу АЛСИМ-2 було включено систему керування інформацією за допомогою мови визначення даних (для опису схем структур даних, збережених у базі даних) і мову маніпулювання даними (для обміну з базами даних, коригування баз даних, аналізу, синтезу і перетворення даних). Комплекс АЛСИМ-2 широко використовувався для дослідження процесів функціонування військово-морських баз Тихоокеанського узбережжя та під час розробка проекту їх автоматизації.

Протягом 1991–1993 років у Інституті кібернетики провадились роботи зі створення системи НЕДИС-90 і реалізації її на персональному комп'ютері IBM PC AT/386. (Автор мови НЕДИС-90 – В. В. Гусєв.) Система призначалась для проектування в реальному масштабі часу проблемно-орієнтованих мов, що мали широке коло застосуваня – створення дискретних, неперервних і гібридних моделей інформаційних, економічних, біологічних та інших систем, планування й обробка результатів експериментів, логічного моделювання, синтезу описів нижнього рівня (схем). Користувачі системи отримали можливість розробляти власні функціональні еквіваленти мов SIMULA, GASP-IV, VHDL і їх спеціалізовані діалекти без побудови нових компіляторів. Розроблені проблемно-орієнтовані мови могли бути як імперативними, так і декларативними. Створена технологія розробки нових мов моделювання для різних додатків базувалась на використанні механізму контекстних модулів.

Систему НЕДИС-90 побудовано на основі компілятора з базової мови об'єктно-орієнтованого програмування, що дає змогу створити нові визначення на основі існуючої системи позначень. Систему реалізовано в 1994 році як компілятор мови С для комп'ютерів, сумісних з IBM PC.

Значні роботи зі створення проблемноорієнтованих пакетів на основі мови НЕДИС було виконано під керівництвом Т. П. Мар'яновича і д-ра техн. наук А. І. Нікітіна. Це такі пакети, як СИМПО, САУККС, ПАРК, МЕРЕЖА, КОМПЛЕКС.

Чималий внесок у становлення імітаційного моделювання зробили науковці кафедри «Обчислювальні машини і програмування на електронно-обчислювальних машинах» Одеського інституту народного господарства. У 1975–1980 роках було розроблено кілька версій пакета прикладних програм ДИСМ, орієнтованого на моделювання систем з дискретними подіями. Базовою мовою ДИСМ стала мова PL/1, реалізована на ЄС ЕОМ. Керівники розробки – В. І. Мановицький, Є. М. Сурков.

У 1980 році було створено програмний комплекс ПОСИМЕЯ-ФОСИМ для моделювання систем з дискретними подіями. Керівник роботи – Є. М. Сурков. Розробники комплексу – А. І. Ахламов, Ю. Б. Пигарев. Комплекс забезпечував автоматизацію проектування імітаційних моделей у діалоговому режимі та використовувався під час дослідження технологічних процесів у морському порту, керування судами морського флоту, систем обробки даних у реальному масштабі часу, керування пасажирським міським транспортом, планування робіт служби мережі Міністерства зв'язку СРСР, завдань керування і статистичної обробки в інформаційно-обчислювальному центрі ЦСУ Молдавської РСР.

У 1994 році в Національному технічному університеті України «Київський політехнічний інститут» на кафедрі «Автоматизовані системи обробки інформації та управління» було розроблено інтерактивну систему імітаційного моделювання ICIM [60], яка має генератор імітаційних GPSS-програм. Перша версія системи ICIM працювала під керівництвом DOS для персональних комп'ютерів (автор – В. М. Томашевський). У 1998 році було створено другу версію – ICIM'95 [62], з графічними засобами побудови імітаційних моделей для Windows. Основне призначення ICIM – моделювання дискретних систем, які можна зобразити за допомогою мереж СМО загального вигляду. Для цієї системи вперше у вітчизняній практиці було розроблено лінгвістичний процесор, за допомогою якого користувач у діалоговому режимі автоматично створював GPSS-програму й запускав її на виконання. Остання версія ICIM – ISS 2000 [61] – має генератор формул і більш ергономічний інтерфейс (http://www.simulation.org.ua/soft.php).

У 1996–1998 роках в Інституті бізнесу, економіки та інформаційних технологій Одеського державного політехнічного університету було розроблено другу версію діалогової автоматизованої системи імітаційного моделювання (ДАСІМ), що призначалася для моделювання дискретних мережних систем. (Керівник проекту – О. Ю. Семишин.) Реалізацію ДАСІМ (http://www.ospu.odessa.ua/adss) виконано для ОС Unix і DOS. Основне призначення ДАСІМ – моделювання обчислювальних систем і виробничих процесів, моделі яких базуються на мережах СМО. Система має власну діалогову мову для створення імітаційної моделі.

Серед інших робіт зі створення проблемно-орієнтованих пакетів моделювання слід відзначити розробки Вінницького політехнічного інституту (засоби автоматизації ерготичних систем – В. Я. Діденко), Львівського обчислювального центру Інституту прикладних проблем механіки і математики (інструментальні засоби моделювання систем з мікропроцесорним керуванням – В. І. Власенко); Київського науково-дослідного інституту автоматизованих систем планування і управління у будівництві (засоби імітаційного моделювання для дослідження технологічних процесів на будівельних об'єктах Київського політехнічного інституту – С. Д. Бушуєв).

Останнім часом фахівці Інституту кібернетики НАН України спрямували свою діяльність на розробки засобів паралельного імітаційного моделювання.

6.8. Сучасний етап розвитку імітаційного моделювання

На сучасному етапі розвитку технологій імітаційного моделювання слід відзначити період кінця 90-х років XX сторіччя і початку XXI сторіччя, який характеризується інтенсивним розвитком методів і засобів моделювання для паралельних і мультипроцесорних обчислювальних систем, розвитком веб-технології в моделюванні, створенням систем моделювання реального часу з розробкою стандарту HLA, побудовою моделюючих систем з використанням інтелектуальних агентів і методів штучного інтелекту.

6.8.1. Засоби паралельного моделювання

Із розвитком високопродуктивних обчислювальних систем розширились можливості імітаційного моделювання великомасштабних моделей. В огляді мов і бібліотек паралельного дискретно-подійного моделювання, представленому в праці [89], відзначені дві основні переваги використання методів і засобів паралельного імітаційного моделювання. Перша – підвищення швидкодії імітаційних програм, а друга – збільшення об'єму доступної пам'яті, що дає змогу реалізовувати значно складніші імітаційні моделі порівняно з однопроцесорними системами.

Для паралельного виконання програм необхідно, щоб мови і пакети моделювання забезпечували функціонування кількох фрагментів програми на різних процесорах. Сучасні операційні системи (наприклад, Unix, Windows) мають засоби розпізнавання незалежних фрагментів програм і планування їх роботи на різних процесорах. Однак неефективне планування може призвести до нерівномірного завантаження процесорів, коли до деяких процесорів будуть створюватись великі черги. Складності під час планування мають місце, якщо вихідні дані одних частин програм є вхідними для інших.

Створення алгоритмів паралельного моделювання потребує від розробників моделі обізнаності з основними технологіями паралельних обчислень, пов'язаних із протоколами синхронізації. З цієї причини було б доцільно знайти такий поступовий спосіб переходу від послідовного моделювання до паралельного дискретно-подійного, за якого розробник концентрував би свої зусилля на процесі моделювання і не відволікався на проблеми обміну і синхронізації. Весь тягар організації паралельного виконання імітаційних програм повинні взяти на себе розробники мов і бібліотек паралельного дискретно-подійного моделювання. Це сприятиме розповсюдженню засобів паралельного імітаційного моделювання.

У більшості мов моделювання, орієнтованих на процеси, реалізоване квазіпаралельне виконання процесів, що дає змогу паралельно запускати їх на різних процесорах. Але в разі такого розпаралелювання процесів виникає ціла низка проблем, пов'язаних з роботою ПКМ. Якщо ПКМ буде функціонувати тільки на одному процесорі, то необхідно забезпечити його обмін повідомленнями з підпрограмами процесів, що виконуються на інших процесорах. У цьому випадку слід ураховувати, що більшість часу імітаційні програми витрачають на роботу саме ПКМ, яка маніпулює списками подій. Таким чином, для прискорення роботи імітаційних моделей бажано виконувати ПКМ паралельно на різних процесорах.

Для організації паралельного дискретно-подійного моделювання застосовуються протоколи обміну повідомленнями, які використовують консервативні або оптимістичні алгоритми, які базуються на концепції *логічного процесу*. Кожний логічний процес для взаємодії з іншими обмінюється повідомленнями з часовими помітками для подій. Щоб зберегти хронологічну послідовність подій під час моделювання, він має обробляти всі вхідні повідомлення про події, не порушуючи порядок перебігу модельного часу. А для цього потрібно обмежувати причинні зв'язки. Такий підхід характерний для консервативного алгоритму, коли кожний логічний процес обробляє вхідне повідомлення від події тільки тоді, коли ніякі інші повідомлення від події з меншою часовою поміткою не будуть надіслані до системи іншими логічними процесами. Таке обмеження може призвести до тупикової

ситуації, для ліквідації якої було запропоновано різні алгоритми. Звичайно використовується алгоритм, який посилає нульові повідомлення з помітками модельного часу, щоб гарантувати неперервний перебіг модельного часу для всіх пов'язаних між собою логічних процесів. Основний недолік цього алгоритму – потенційно велика кількість нульових повідомлень, що передаються, порівняно з кількістю фактичних повідомлень від подій. Щоб генерувати нульові повідомлення, система має бути здатною прогнозувати наступні події в імітаційній моделі. Однак якість прогнозування залежить від обізнаності користувача з імітаційною моделлю і характеристиками подій під час виконання програми моделювання.

Оптимістичний алгоритм дає змогу обробляти події без перевірки того, чи призведе обробка повідомлення від події до будь-якого критичного порушення. Кожний логічний процес реєструє часовий перехід у випадку обробки кожної вхідної події, що відбувається. Якщо повідомлення надходить від події, що має помітку з меншим часом, ніж подія, яку було оброблено, то логічний процес виправляє цю помилку, скасовуючи оброблені раніше події. Такий процес відомий як відкочування (*rollback*), тобто повернення до пройденого моменту часу. Для реалізації механізму відкочування зазвичай використовуються антиповідомлення, у результаті обробки яких локальні або віддалені логічні процеси повертаються до попереднього несуперечливого стану. Взагалі паралельні алгоритми імітації, що використовують оптимістичний алгоритм, потребують більшого об'єму пам'яті, ніж ті, що використовують консервативний алгоритм, через те що для оптимістичного алгоритму необхідно зберігати інформацію про події. Періодично обчислюючи глобальний віртуальний час, система може звільняти пам'ять від даних про застарілі стани і логічні процеси як оброблених повідомлень, так і подій.

Прикладом паралельної об'єктно-орієнтованої мови моделювання, яка використовує протокол з оптимістичним алгоритмом, є мова APOSTLE. Мова MAISIE, що базується на мові C, підтримує багато різних протоколів моделювання, включаючи оптимістичний і консервативний. На базі мови MAISIE створено кілька середовищ об'єктно-орієнтованого імітаційного моделювання, таких як Moose і Parsec. Ще однією об'єктно-орієнтованою мовою моделювання, яка використовує оптимістичний алгоритм, є мова ModSim, заснована на мові Modula-2, і розроблена компанії CACI.

6.8.2. Засоби, орієнтовані на веб-технології

Розвиток сіткових інформаційних технологій торкнувся і засобів імітаційного моделювання. З'явилось багато пакетів, які використовують можливості збільшення продуктивності засобів імітаційного моделювання шляхом паралельного виконання програм у мережі.

Використання технології клієнт-сервер відкриває можливості одночасної роботи над великими імітаційними проектами цілим групам користувачів. Застосування веб-технологій типу CORBA, RMI дає змогу підтримувати на належному рівні технології проектування і розробки розподілених імітаційних моделей.

Для роботи в мережі та організації розподіленого імітаційного моделювання потрібно керуватись стандартом, яким є архітектура високого рівня.

6.8.3. Архітектура високого рівня

Архітектура високого рівня HLA (High Level Architecture) – стандарт, який визначає загальну архітектуру систем розподіленого моделювання і підтримує моделювання систем, що складаються з різних компонентів, які можна реалізувати за допомогою різних мов моделювання. Необхідність розробка стандартів моделювання для складних імітаційних моделей виникла ще в кінці 90-х років XX сторіччя. Ці стандарти, визначені Міністерством оборони США, регламентують процес проектування і реалізації складних комплексних моделей. На теперішній час існує велика кількість імітаційних моделей, які застосовуються в різних галузях, але вони реалізовані за допомогою різних засобів імітаційного моделювання. За необхідності реалізації нового складного імітаційного проекту можна було б використати накопичений досвід у вигляді готових частин моделей. На жаль, доводиться часто виконувати значний обсяг роботи, щоб пристосувати існуючі моделі до нового проекту. У деяких випадках простіше написати нову програму для реалізації моделі, ніж модифікувати існуючу. Іншими словами, традиційним імітаційним моделям не завжди вистачає властивостей *багаторазового використання* і *здатності до взаємодії*.

Властивість багаторазового використання передбачає, що складові імітаційної моделі можуть застосовуватись у різних сценаріях моделювання та додатках. Подібна до цієї властивості здатність до взаємодії означає, що складові моделей багаторазового використання можуть бути поєднаними з іншими компонентами без додаткового коду. Більш того, ця властивість передбачає можливість об'єднання складових частин моделей для проведення моделювання на різних розподілених обчислювальних платформах, досить часто – в реальному масштабі часу. Отже, необхідно мати деякий стандарт, який гарантував би можливість взаємодії різних компонентів моделі, реалізованих на різних комп'ютерах і різних платформах. Таким стандартом стала архітектура високого рівня.

Складну комплексну модель можна розглядати як ієрархію компонентів, об'єднаних між собою. На найнижчому рівні знаходяться моделі елементів системи. Це можуть бути математичні моделі, моделі СМО тощо. Стандарт HLA [90] дає змогу під час моделювання приєднувати до існуючого об'єднання інші компоненти. *Об'єднання* – це більш ніж програмна модель, оскільки вона може включати інтерфейс з оператором, реальне апаратне забезпечення і загальне програмне забезпечення, яке виконує функції збирання та обробки даних, їх аналізу тощо.

Архітектура високого рівня включає три елементи:

- правила об'єднання;
- технічні вимоги до інтерфейсу;
- об'єктну модель шаблону (ОМШ).

На найвищому рівні HLA складається з правил, яким повинні відповідати умови об'єднання в термінах HLA. Таких правил десять: п'ять правил об'єднання і п'ять правил об'єднань.

Правила об'єднання визначають вимоги до документації, об'єктне зображення, обмін даними, вимоги до інтерфейсу та властивості, які використовуються монопольно. Правила об'єднань описують окремі об'єднання й охоплюють документацію,

передавання потрібних об'єктних властивостей та керування ними (три правила), а також керування часом.

Технічні вимоги до інтерфейсу визначають стандарт інфраструктури під час прогону моделі (Runtime Infrastructure, RTI). Програмний модуль RTI необхідний для виконання та підтримки моделювання. Він може керувати виконанням одного або кількох процесів на різних комп'ютерах – це один модуль, який одночасно може підтримувати кілька об'єднань. На рис 6.6 зображено програмні компоненти HLA.

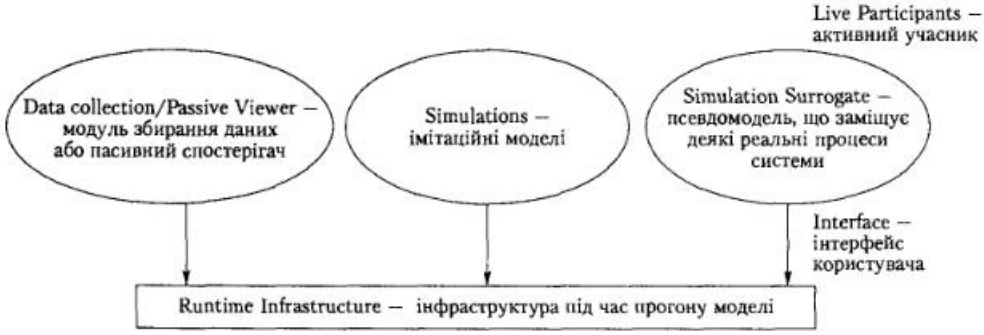


Рис. 6.6. Програмні компоненти HLA

Приклад використання HLA для системи імітаційного моделювання MODSIM III описано у праці [78].

6.9. Системи імітаційного моделювання

Крім спеціалізованих засобів для розробки програм імітаційного моделювання можна використовувати алгоритмічні мови загального призначення та системи візуального програмування. Створені у такий спосіб програми будуть мати більшу швидкодію, ніж програми, побудовані за допомогою спеціалізованих засобів, але для їх розроблення, програмування та налагодження потрібно більше часу. Скоротити тривалість розроблення імітаційних моделей можна за допомогою засобів і мов імітаційного моделювання. За даними Е. Кіндлера [20], кількість таких засобів вже сягнула за 500 і з кожним роком все збільшується.

Засоби і мови імітаційного моделювання традиційно розділяють на дискретні, неперервні й комбіновані. У цьому розділі розглядаються тільки найбільш поширені з них. Деякі з мов моделювання, такі як GPSS, SIMSCRIPT і SIMULA, мають більш ніж сорокарічну історію, але і досі є актуальними. Що стосується комбінованих (дискретно-неперервних або неперервно-дискретних) засобів імітаційного моделювання, то більшість з них дають змогу провадити моделювання систем обох типів або мають додаткові модулі для моделювання систем певного типу. Наприклад, у GPSS World (дискретна мова) є блок INTEGRATE для імітаційного моделювання неперервних процесів, а пакети Stella і iThink (неперервні системи моделювання) мають вбудовані елементи для моделювання дискретних систем.

6.10. GPSS

Однією з перших мов моделювання, які полегшують процес написання програм, була мова GPSS, створена у компанії IBM Джеффрі Гордоном. Свого часу ця мова була реалізована практично на всіх типах комп'ютерів і належала до першого десятка кращих мов програмування, випереджаючи навіть мову АЛГОЛ. На теперішній час існує кілька її версій для персональних комп'ютерів: для ОС DOS – GPSS/PC, для OS2 і DOS – GPSS/H, для Windows – GPSS World [95].

Мова моделювання GPSS (General Purpose Simulating System) – система моделювання загальноцільового призначення) використовується для побудови подійних дискретних імітаційних моделей і проведення експериментів з ними. У середовищі GPSS модель системи задається у вигляді блок-схеми або послідовності операторів програми, еквівалентній блок-схемі. Блок-схема – це набір геометричних фігур із характерними контурами блоків мови GPSS, з'єднаних між собою лініями. Блоки – це окремі підпрограми, реалізовані засобами макроасемблера або мови С. У різних версіях мови кількість блоків для створення імітаційних програм різна і сягає кількох десятків.

Мова моделювання GPSS містить спеціальні засоби опису динамічної поведінки ймовірнісних систем. Цей опис задається як зміна станів системи в дискретні моменти часу, тобто час моделювання змінюється випадково від події до події.

Мова GPSS – це мова моделювання дискретних систем декларативного типу, орієнтована на процеси і побудована за принципами об'єктної мови. Основними елементами цієї мови є *транзакти* і *блоки*, які відображають відповідно динамічні і статичні об'єкти системи.

Як кожна мова програмування, GPSS містить словник і граматику, за допомогою яких може бути розроблено програми моделювання систем певного типу. Транслятор мови працює в дві фази. На першій фазі компіляції перевіряється синтаксис і семантика GPSS-програми, а на другій – здійснюється просування транзактів по моделі від блока до блока. Транзакти в GPSS є деякими динамічними об'єктами моделювання.

Об'єкти системи, які моделюються, призначено для різних цілей. Вибір об'єктів для конкретної моделі залежить від характеристик модельованої системи. Кожний об'єкт має певну кількість властивостей, названих у GPSS *стандартними числовими атрибутами* [68], які призначені для збирання статистичної інформації про об'єкти.

Кожна GPSS-модель складається з блоків і транзактів. Послідовність блоків GPSS-моделі відображає напрямки, у яких переміщаються транзакти. У GPSS концепція передавання керування від блока до блока має особливості. Як і всі мови моделювання, GPSS має внутрішній механізм передавання керування, який реалізується в модельному часі, що дає змогу відображати динамічні процеси в реальних системах. Керування від блока до блока передається за допомогою руху транзактів у модельному часі; звернення до підпрограм блоків відбувається через рух транзактів. Змістовне значення транзактів визначає розробник моделі. Саме він установлює аналогію між транзактами і реальними елементами системи, яка

моделюється. Така аналогія ніколи не вказується інтерпретатору GPSS, вона залишається в уяві розробника моделей.

Система GPSS World [50, 61] має розширені можливості порівняно зі стандартною мовою GPSS за рахунок вбудованої алгоритмічної мови PLUS, що дало змогу включити в систему засоби планування проведення експериментів. Крім того, наявність блока INTEGRATE дає можливість будувати неперервні або дискретно-неперервні імітаційні моделі.

6.11. SIMSCRIPT

Мова SIMSCRIPT уперше була запропонована у 1962 році співробітником компанії RAND Corporation Г. Марковичем [82] і зареєстрована компанією CACI Products Company (http://www.caciasl.com/products/simscript.cfm). Перша версія мови була подійно-орієнтованою, в ній існували динамічні об'єкти з атрибутами та статичні об'єкти – активності. Під час взаємодії між цими елементами створювались черги об'єктів, над якими можна виконувати певні дії. Дії визначаються підпрограмами, що викликають підпрограми подій. Події, які бувають зовнішніми та внутрішніми, заносяться до списку подій. Програмний код виглядає як звичайна англійська мова.

З типової подійно-орієнтованої мови, в перших версіях, SIMSCRIPT розвинулась у інтегрований засіб розробки імітаційних моделей, у версії SIMSCRIPT II.5 [79], керування яким виконується за допомогою системи SIMLAB, що включає універсальну мову моделювання SIMSCRIPT II.5 з упровадженими конструкціями для процесноорієнтованого моделювання і утиліти. Версія SIMSCRIPT II.5 дає змогу створювати дискретно-подійні, неперервні та комбіновані моделі. Система SIMSCRIPT II.5 складається з таких компонентів:

- SIMLAB – середовище розробки та програмування, яке може застосовуватись для платформ Windows та UNIX;
- SIMDRAW – графічний редактор;
- COMPILER – транслятор програми Simscript II.5 у код C (це гарантує можливість використання коду для різних систем);
- LIBSIM – модуль динамічної підтримки бібліотеки під час виконання моделювання;
- LIBSIMG – графічна бібліотека, що містить графіку та анімацію;
- SIMDEBUG – діалоговий символічний налагоджувач;
- DATAGRAPH – автономний пакет, розроблений для пристосування різних функцій розподілів імовірностей для даних модельованої системи або імітаційної моделі;
- SIMVIDEO – утиліта для фіксації і програвання отриманих результатів моделювання за допомогою анімації.

Система SIMSCRIPT широко застосовується в багатьох країнах як засіб моделювання систем передача даних, комп'ютерних мереж, транспортних і виробничих систем, військових операцій і планування логістичних процесів. Вона включає

засоби опису предметної області, до якої належать пристрої, атрибути, набори (тобто черги), за допомогою яких користувач може співвідносити реальні об'єкти з моделлю. Це подійно-орієнтована мова, яка може бути перетворена в мову, схожу за формою на мову моделювання процесів. Для забезпечення графічного інтерфейсу з імітаційною моделлю SIMSCRIPT на персональному комп'ютері додатково встановлюється пакет SIMGRAPHICS.

6.12. Taylor II і Taylor ED

Велика кількість розробників імітаційних моделей використовують систему Taylor II Simulation – програмний пакет для імітаційного моделювання та аналізу процесів виробництва. Ця система дає змогу оцінювати продуктивність виробничих систем, ідентифікувати критичні параметри, вимірювати показники використання й час зайнятості ресурсів. Вона також може використовуватись для підтримки інвестиційних рішень або перевірки різних стратегій керування виробництвом.

Основні особливості цієї системи такі:

- наявність великого набору модельних конструкцій і законів розподілу випадкових величин;
- можливості роботи з чергами об'єктів різних типів;
- можливості моделювання в різних режимах (використання оптимізаційного алгоритму моделювання, покрокове моделювання, установлення фіксованого часу моделювання, переривання процесу моделювання за станом і т. ін.);
- наявність анімаційних засобів (схематична, дво- і тривимірна).

Остання версія системи Taylor Enterprise Dynamics (Taylor ED) реалізована завдяки об'єктно-орієнтованому підходу (останнім часом змінено і назву системи на Flexsim ED www.flexsim.com). Базовим поняттям у новій версії є «атом». *Атом* – об'єкт з чотирма вимірами (4D). Кожний атом характеризується місцем у просторі і швидкістю, тобто координатами (x, y, z) і динамічною поведінкою. Атоми можуть успадковувати свою поведінку від інших атомів і містити інші атоми. Їх можна створити і знищити, вони можуть переміщатись у різних напрямках. Атоми взаємодіють між собою через наскрізні канали. Всі компоненти моделі, чи то безпосередньо ресурс, результат, об'єкт моделі, таблиця, звіт, бібліотека або навіть додаток, у системі Taylor ED формуються з атомів. Користувачі формують моделі, використовуючи готові атоми, що зберігаються в бібліотеці, або за допомогою власних атомів зі специфічними властивостями, які створюються за допомогою мови 4DScript. Під час прогону моделі система забезпечує керування поведінкою атомів, інтерфейсом користувача і засобами візуалізації.

Процес створення і запуску моделей здійснюється автономно і не потребує взаємодії користувача із системою Taylor ED. Застосування системи Taylor ED істотно спрощує процес моделювання і дає змогу користувачам пристосовувати його до власних потреб. Система використовується переважно для моделювання потокових систем у виробництві, діловодстві, транспортних перевезеннях, інформаційних системах.

6.13. Об'єктно-орієнтоване візуальне моделювання

Основні принципи та концепції візуального моделювання розглянемо на прикладі системи VSE (Visual Simulation Environment – середовище візуального моделювання), яка використовує об'єктно-орієнтований підхід до розробки і виконання імітаційних моделей з дискретними подіями. Це універсальне середовище може використовуватись для вирішення складних проблем у галузях керування системами транспорту, ділових і виробничих процесів, комп'ютерних мереж, охорони здоров'я, а також керування системами та мережами забезпечення і перевезень. Розробник – корпорація Orca Computer, Inc. (www.orcacomputer.com).

Програмне забезпечення VSE є об'єктно-орієнтованим і вимагає від аналітика розуміння об'єктно-орієнтованого підходу (ООП). Розглянемо базові концепції ООП у термінах VSE.

Об'єкт – елемент модельованої системи. Кожен об'єкт має деякі характеристики і виконує деякі дії. Об'єкти є стандартними блоками VSE-моделей.

Клас – угруповання об'єктів з однаковими характеристиками, діями і поведінкою. Класи верхнього рівня називаються *суперкласами*. Клас може мати *підкласи*, які успадковують усі характеристики, дії і поведінку батьківського класу. Клас верхнього рівня називається *кореневим класом*.

Вбудована в VSE бібліотека класів називається VSLibrary. В ній містяться всі класи та об'єкти VSE-моделей.

Створення об'єкта, який належить до класу, називається *реалізацією об'єкта*. Новий об'єкт успадковує всі властивості (змінні екземпляра) і поведінку (методи екземпляра), визначені в батьківському класі.

Властивості та поведінка, притаманні об'єктам, визначаються в *методах*. Існують методи класу і методи екземпляра. *Методи класу* використовуються для забезпечення властивостей, специфічних для класу, а *методи екземпляра* – для визначення властивостей і поведінки кожного об'єкта з цього класу. Код методу створюється тільки один раз у класі і не переноситься в кожну реалізацію об'єкта з даного класу. Це значно спрощує проектування моделі, тому що потенційні зміни обмежені рамками лише одного класу. Будь-який метод VSE містить оголошення вхідних параметрів, локальних змінних методу, типів значень, які повертаються методом, а також логіку методу в термінах VSE.

Усі екземпляри об'єктів мають унікальні адреси і пов'язуються один з одним за допомогою повідомлень. Адреси об'єктів називаються об'єктними посиланнями і використовуються для того, щоб визначити об'єкт, який отримує повідомлення. Відправка повідомлення до об'єкта – це виклик одного з його методів.

Те, як об'єкт виконує свої функції, повністю приховане від зовнішнього середовища. Ця властивість називається *інкапсуляцією*. У системі VSE реалізована ще одна властивість ООП – *поліморфізм*, тобто здатність об'єктів різних класів виявляти різну поведінку у відповідь на одне й те саме повідомлення. Щоб пов'язати об'єкт системи з моделлю, використовують реквізит – *асоціацію*. Принцип захисту асоціації полягає в тому, що кожний об'єкт або компонент реальної системи повинен мати відповідний об'єкт моделі.

Наведемо основні принципи візуального моделювання, реалізовані в системі VSE [92, 100]:

1. «Що бачите на екрані, те й отримаєте під час моделювання» (концепція WYSIWYG, тобто від «what you see is what you'll get»).

   Поєднання логічного та об'єктного моделювання з метою збереження концептуальної структури програмного продукту значно ускладнює саму модель і перевірку її правильності, погіршує зручність експлуатації, зменшує можливість багаторазового використання і надійність. Структура концептуальної моделі повинна враховувати пряме та природне зображення моделі. У системі VSE це забезпечується парадигмою моделювання – у середовищі моделювання ви бачите структуру системи, яка моделюється, і це дає змогу уникнути складної логіки в зображенні моделі.

2. Принципи концептуальної структури.

   Ці принципи визначають вимоги, яким повинна задовольняти концептуальна модель системи. Серед них найважливішими є такі:

   - *графічний* – модель повинна бути графічно-структурною;
   - *ієрархічний* – статична (компоненти) і динамічна (динамічні об'єкти) моделі будь-якої структури можуть бути розкладені на більш прості складові;
   - *об'єктно-орієнтований* – модель створюється згідно з принципами об'єктно-орієнтованого програмування;
   - *компонентний* – модель може бути спроектована шляхом багаторазового використання компонентів моделі, що зберігаються в бібліотеці компонентів;
   - *візуальний* – для створення анімаційних ефектів можна використовувати будь-яке зображення, що зберігається в пам'яті комп'ютера, і обробляти його редактором VSE; система моделювання автоматично відтворює всі анімаційні ефекти;
   - *багатофункціональний перегляд* – під час створення складних візуальних імітаційних моделей можна використовувати машинно-орієнтований, матеріально-орієнтований перегляд, або перегляд, що складається із комбінації двох режимів перегляду.

3. Спрощення логічної структури моделі.

   Всі моделі в середовищі VSE будують за три етапи. На першому етапі за допомогою графічного методу створюють ієрархічні структури статичної й динамічної моделі, на другому – визначають нові класи та об'єкти. І тільки на третьому етапі з використанням об'єктноорієнтованої мови сценаріїв VSE задають логіку виконання всіх методів моделі. Локалізація логічної специфікації до конкретного методу значно зменшує складність моделі, спрощує її розроблення і збільшує зручність використання.

4. Ієрархічна архітектура моделі.

   Графічна та ієрархічна декомпозиції імітаційної моделі відбуваються зверху вниз, що дає змогу виділити в статичній та динамічній моделях окремі компоненти – об'єкти. Динамічний об'єкт – це об'єкт, який під час моделювання фізично або логічно переміщається в моделі від однієї точки до іншої. Наприклад, автобус, літак, пасажир, завдання, яке виконує комп'ютер, можна подати як динамічні об'єкти. Динамічний об'єкт, як і статичний, можна розділити на

окремі складові, тобто зобразити у вигляді ієрархічної структури компонентів. Наприклад, автобус – це об'єкт, що перевозить пасажирів, тобто інші динамічні об'єкти.

5. Система VSE базується на графічних зображеннях.

   Модель системи у VSE задається у вигляді графічної ієрархічної структури. Редактор VSE дає змогу виконувати обробку графічних зображень у різних форматах. Зображенням, наприклад, може бути план розміщення компонентів моделі або його частина.

6. Одночасний перегляд кількох анімацій.

   VSE має багатовіконний графічний інтерфейс, який дає можливість спостерігати за анімаційними ефектами для різних компонентів моделі. Кількість відкритих одночасно вікон обмежується тільки розмірами екрана. Система моделювання дає змогу приєднувати до персонального комп'ютера кілька моніторів, що збільшує екранний простір.

7. Автоматизація створення програмного забезпечення.

   Під час створення складних візуальних імітаційних моделей застосовується сучасна парадигма автоматизованого проектування програмного забезпечення. Модель генерується автоматично на основі об'єктно-орієнтованої графічної структури та специфікації моделі. Помилки під час виконання програми відображаються на специфікації моделі, а не для мови програмування.

8. Планування проведення експериментів.

   Механізм передавання повідомлень об'єктно-орієнтованої парадигми дає змогу застосовувати будь-який статистичний пакет для організації взаємодії з імітаційною моделлю.

9. Статистичний аналіз вихідної інформації.

   Під час планування проведення експериментів з моделлю для побудови довірчих інтервалів оцінок параметрів можна обирати будь-яку методику, що застосовується для збирання даних, наприклад метод повторних експериментів, метод підвибірок і т. ін. Результати моделювання можна зберігати у файлах вихідних даних. За допомогою VSE Output Analyzer можна відкривати ці файли і провадити статистичний аналіз результатів моделювання.

   Імітатор VSE Simulator створює середовище для проведення експериментів. Це автономний додаток, розроблений для програмної реалізації імітаційних моделей. Застосовуючи VSE Simulator, користувач може змінювати структуру моделі, значення параметрів об'єктів і провадити різні експерименти з моделлю. Модель можна передати також іншим користувачам, які мають у своєму розпорядженні VSE Simulator. Це дає їм змогу провадити експерименти та отримувати необхідні результати, змінюючи значення параметрів моделі (змінних екземпляра).

   Аналізатор вихідних даних VSE Output Analyzer використовується для статистичного аналізу вихідних даних і побудови довірчих інтервалів оцінок параметрів моделі.

   Застосування технології VSE підвищує продуктивність праці дослідників і рівень автоматизації процесу проектування імітаційної моделі.

6.14. Об'єктно-орієнтований пакет SIMPLE++

Пакет SIMPLE++ (http://www.eM-Plant.de) призначено для моделювання складних виробничих процесів. Розробник цього пакета – компанія Tecnomatix Technologies. Програми пакета написано на мові C++, яка підтримує технології об'єктно-орієнтованого програмування. Таким чином, одна з основних переваг пакета в тому, що програми моделювання можна переносити на різні комп'ютерні платформи.

Пакет SIMPLE++ має інтуїтивно зрозумілий графічний інтерфейс користувача, який дає змогу працювати як з програмами моделювання, так і з анімаційними додатками. Користувач має можливість налагодити параметри інтерфейсу відповідно до свого стилю роботи.

У пакеті реалізована функція «incremental operation» (операція збільшення). Це означає, що за потреби можна в будь-який час змінити рівень деталізації зображення структури моделі. Крім того, для кожного об'єкта моделі користувач може побудувати власну маску, в якій можна зазначити тільки ті параметри моделювання, які цікавлять користувача.

Пакет SIMPLE++ може функціонувати в середовищі різних операційних систем і має засоби взаємодії з базами даних SQL. Всі об'єкти, необхідні для створення моделі, доступні на панелі об'єктів. Матеріальні та інформаційні потоки моделі формуються за допомогою базових об'єктів. Велика кількість базових об'єктів, функцій, типів даних і операторів, а також наявність мови керування SimTALK забезпечують статистичну обробку вихідної інформації.

Об'єднуючи базові об'єкти в інтерактивному режимі, користувач може створювати власні об'єкти. Існує можливість розмістити їх на палітрі базових об'єктів, що забезпечить модульність проектування та ієрархічність моделей.

Пакет SIMPLE ++ широко застосовується для потреб автомобільної промисловості, тому більшість його основних бібліотек також орієнтовані на роботу в цій галузі (наприклад, PaintShop, збиральний конвеєр і JobShop).

6.15. Інтерактивний пакет для моделювання Simulink

Пакет Simulink [11] використовується для проектування систем керування, моделювання комунікаційних систем, цифрової обробки сигналів тощо. Імітаційні моделі створюються за допомогою структурних компонентів – блок-діаграм, які дають змогу моделювати динамічні системи, оцінювати їх характеристики, модифікувати проект. Пакет містить бібліотеку з великою кількістю блоків, які використовуються для проектування моделей систем різних типів: лінійних, нелінійних, неперервних, дискретних і гібридних. Моделі можуть утворювати складні ієрархії, тим самим забезпечуючи спрощене зображення компонентів і підсистем.

Сам процес моделювання може бути інтерактивним або виконуватись з командного рядка MATLAB. Тісна інтеграція з пакетом MATLAB дає змогу використовувати в Simulink усі засоби проектування й аналізу систем, які є в MATLAB. За допомогою їх можна отримати достовірну інформацію про роботу системи.

У поєднанні з такими програмними продуктами, як Real-Time Workshop і Stateflow, пакет Simulink утворює сімейство програм цифрової обробки сигналів, розроблення комунікаційних систем, систем керування і моделювання. Можливості середовища моделювання можна розширити завдяки застосуванню спеціалізованих додатків, таких як DSP Blockset, Fixed-Point Blockset, Power System Blockset і Communications Toolbox. Програмний продукт Real-Time Workshop доповнює пакети Simulink і Stateflow Coder, забезпечуючи автоматичне генерування коду C або Ada95 з моделей Simulink, більш того Real-Time Workshop підтримує роботу динамічних моделей на різних комп'ютерних платформах, включаючи системи реального часу. За допомогою Real-Time Workshop можна легко генерувати код для дискретних, неперервних і гібридних систем.

Програмний продукт Dials & Gauges Blockset дає можливість легко додавати графічні інструменти і панелі керування до моделей Simulink, завдяки чому можна створити додаток, який має реалістичний вигляд модельованої системи.

За допомогою Simulink Report Generator можна готувати і редагувати звіти з оцінками показників роботи моделей Simulink і Stateflow у різних форматах, таких як HTML, RTF, XML і SGML.

6.16. Системи візуального моделювання неперервних процесів

Теоретичною базою для розробки засобів моделювання неперервних процесів можна вважати цикл робіт із системної динаміки та імітаційного моделювання, опублікованих на початку 60-х років XX сторіччя Дж. Форрестером у рамках проекту мови DYNAMO в Массачусетському технологічному інституті. Суттєві зміни в теорії і практиці системного динамічного підходу відбулись у кінці 80-х років з появою персональних комп'ютерів і застосуванням модельно-орієнтованих програмних продуктів, в яких структуру моделей було представлено у вигляді ідеограм, тобто схематичних зображень (наприклад, DYNAMO, Stella, і iThink, Powersim, Vensim, Modus та ін.).

Програмні комплекси Stella і iThink (виробництва компанії High Performance Systems, Inc., www.hps-ltd.com) призначено для перетворення моделей прийняття рішень у імітаційні моделі. Основну увагу сфокусовано на розвиток у людини вміння приймати рішення, необхідні для дослідження систем із складними взаємозалежними зв'язками між частинами. Вказані програми дають змогу швидко та просто формулювати і перевіряти гіпотези та моделювати наслідки їх втілення.

Для побудови моделі необхідно не лише виділити її складові, визначити зв'язки між ними та отримати кількісну оцінку параметрів, потрібно також уміти правильно побудувати *уявну модель*, точно вибрати пріоритети і врешті-решт правильно *сформулювати* свою думку. Концептуально побудувати модель прийняття рішень можна за допомогою причинно-наслідкових діаграм зі зворотними зв'язками і схем виду «фонд–потік», які розглядались у розділі 1.8.2.

Під час моделювання процес формування первинних зображень предметної області можна звести до процесу створення найпростіших ідеограм, які за зовнішньою

примітивністю приховують дуже змістовну інформацію. Недарма програмні інструменти, що використовують ідеографічні нотації, рекламуються сьогодні як *інструменти візуального мислення*. Подальше уточнення, удосконалення та ускладнення моделей зводиться практично до зображення зворотних зв'язків і циклів, а також до об'єднання таких ідеограм у більш розгорнуті графічні ансамблі.

Ідеограми потокових моделей Stella і iThink будуються з таких елементів, як *фонди*, *потоки*, *конвертори* і *конектори*.

Фонд – це кількість будь-чого, що існує в певний момент часу і визначається в деяких одиницях (наприклад, грошових або фізичних). Фонди зображуються у вигляді прямокутника, який здатний накопичувати одиниці фонду. Фонди поповнюються через вхідні потоки і розтрачуються через вихідні. Вони найбільш схожі на довгострокову пам'ять, буфер, ресурс, бункер або резервуар. Як буфер фонд може використовуватись для забезпечення балансування швидкості нагромадження і вичерпування. Така функція фонду – основа концепції динамічної поведінки потокових моделей (рис. 6.7).

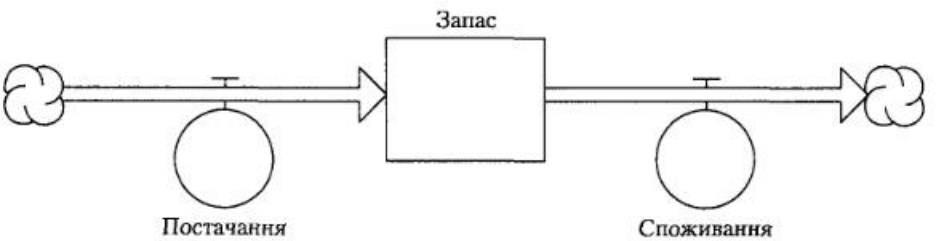


Рис. 6.7. Потокова модель

У реальному житті практично ніколи не буває так, щоб постачальник був готовий поставити споживачу стільки одиниць продукції, скільки необхідно на певний час. На практиці доводиться стикатися з різного роду ресурсами, які використовуються в режимах наповнення або вичерпування. Позики, доходи від продажів, випуск акцій – це ресурси, що поповнюються. Поточні витрати, поточні активи, фіксовані активи, скорочення пасивів і виплати власникам – ресурси, які вичерпуються. Усі ці ресурси доцільно інтерпретувати за допомогою ідеограм фондів.

Фонд як резервуар – багатоцільовий модельний механізм, який підтримує практично всі можливі варіанти інтерпретації динаміки потокових процесів. У найпростішому випадку – це все, що наповнює резервуар і перемішується з його вмістом, нічим від нього не відрізняючись.

Фонди поповнюються або зменшуються за допомогою потоків, які за характером використання розділяються на обмежені і необмежені, односпрямовані і двоспрямовані, конвертовані і неконвертовані. Потік – це процес, неперервний у часі. Оцінити його можна у фізичних або грошових одиницях, співвіднесених із заданим проміжком часу (грошові одиниці на місяць, літри за годину, вартість акції на час закриття біржі у визначений день і т. ін.). Потік зображується фігурою, яка складається зі шляхопроводу, вентиля, регулятора потоку і покажчика напрямку (рис. 6.8).

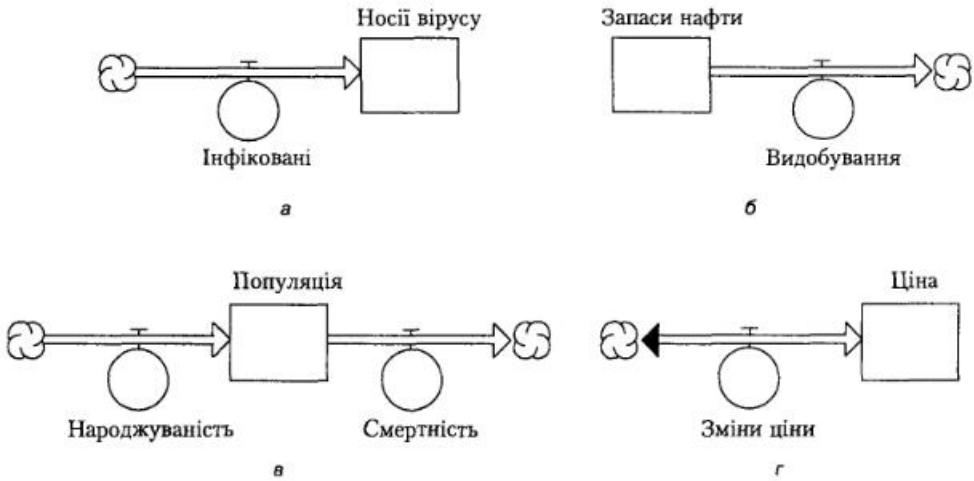


Рис. 6.8. Основні елементи потокових моделей: вірусне інфікування (*а*); видобуток нафти (*б*); цикл популяції (*в*); встановлення ціни (*г*)

Інтенсивність потоку звичайно обмежується фондом. Однак нерідкі випадки, коли модельна ситуація вимагає використання необмежених потоків, і тоді відповідне джерело або споживач позначають піктограмою необмеженості (у вигляді хмарки). Дійсно, як можна обмежити потік замовлень за цінами, які стихійно складаються на ринку споживчих товарів? Якими рамками можна обмежити ступінь довіри до тієї або іншої фінансової компанії? Як можна обмежити народжуваність у деякій популяції?

Для деталізації й уточнення поведінки потокових схем як перетворювачі модельних одиниць використовуються конвертори. Вони застосовуються як змінні або алгебричні вирази для визначення, наприклад, доходу, ціни, рейтингу. Зображуються конвертори у вигляді кіл, з'єднаних з іншими елементами стрілками – конекторами. Конвертори можуть бути механізмами уточнення й еквівалентної заміни фондів. На рис. 6.9 конвертори забезпечення і вимога на відвантаження пов'язані співвідношенням

```
забезпечення = запас / вимога на відвантаження.
```

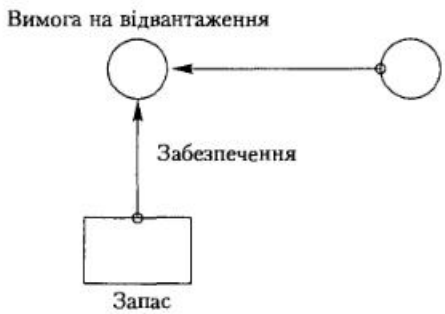


Рис. 6.9. Зв'язки конверторів для забезпечення виробництва

Конвертори можуть також бути механізмами уточнення й еквівалентної заміни потоків. На рис. 6.10 прибуток представлено не як потік, а як конвертор, що обчислює значення різниці між доходами і витратами.

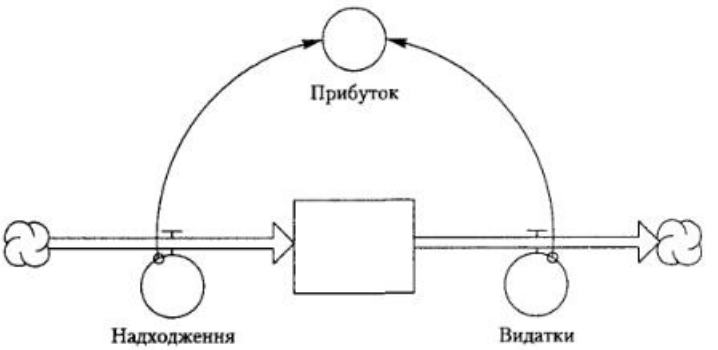


Рис. 6.10. Визначення прибутку за допомогою конвертора

Найчастіше конвертори використовуються як звичайні проміжні змінні і служать для керування процесом взаємодії елементів системи, регулюючи інтенсивність потоків. Кожен конвертор характеризується власним значенням. Значення конверторів можуть задаватись повноцінними алгебричними виразами з використанням представницького набору вбудованих функцій. Треба пам'ятати, що конвертори – не акумулятори, вони не створюють затримки в часі, й останній конвертор у ланцюжку конверторів спрацьовує одночасно з першим.

Нарешті, конвертори застосовуються як зовнішні генератори, наприклад джерела потоків заданої інтенсивності, а також як графічні функціональні елементи, які часто використовуються для задання параметрів і режимів керування активністю потокових моделей.

У системах Stella і iThink визначено такі три способи використання графічних функціональних елементів:

- для графічного відображення потоку за допомогою конвертора або регулятора (зокрема, дискретні значення для регулятора потоку можуть задаватись у певні моменти імітаційного часу через виклик убудованої функції TIME);
- для графічного відображення за допомогою конвертора або регулятора потоку дискретних значень ємності фонду;
- для графічного відображення за допомогою конвертора або регулятора потоку ефектів впливу неформалізованих факторів, наприклад впливу людського фактора на виробництві.

Конектори використовуються для логічного зв'язку елементів потокових діаграм. На рис. 6.11 зображено всі можливі випадки застосування конекторів. Конектор може з'єднувати в логічні пари конвертор із потоком, фонд із потоком, фонд із конвертором, потік із потоком, потік із конвертором і конвертор з конвертором. Слід відзначити, що стрілка конектора ніколи не вказує на фонд, тому що керування фондами завжди здійснюється через вхідні і вихідні потоки.

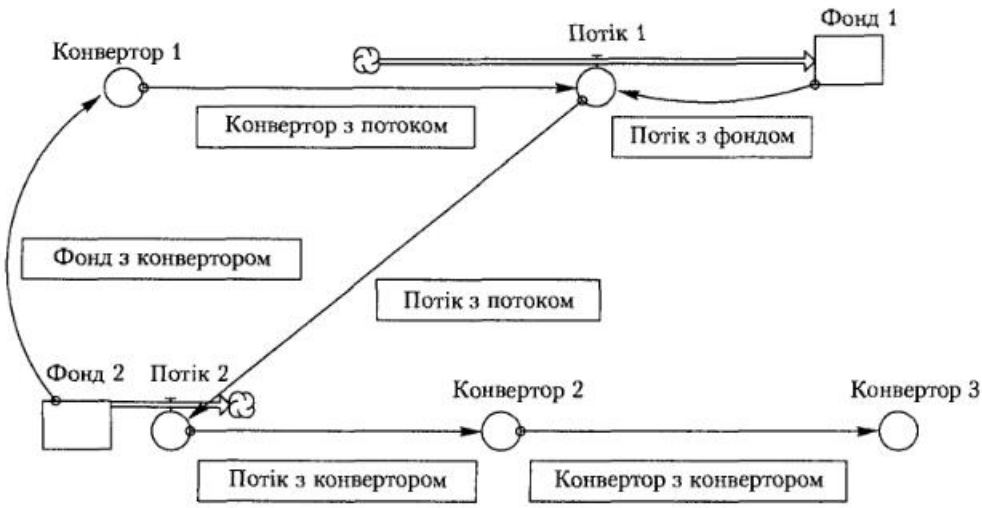


Рис. 6.11. Приклади застосування конекторів

Конектори, як і потоки, можуть застосовуватись для конструювання блок-схем, які в системах Stella і iThink використовуються як зовнішній рівень зображення моделей.

Динаміка процесів і об'єктів, якими б складними вони не були, у системах Stella і iThink виражається за допомогою всього п'яти типів базових параметрів:

- збільшення фондів;
- вичерпування фондів;
- робочий процес;
- сполучення потоків;
- адаптація фондів.

Процес *збільшення фондів* використовується в ситуаціях, коли потрібно змінити інтенсивність вхідного потоку фонду. Процеси такого типу широко розповсюджені в найрізноманітніших сферах діяльності. На рис. 6.12 зображено схему регулювання вхідного потоку згідно з виразом

```
вхідний потік = фонд * інтенсивність збільшення.
```

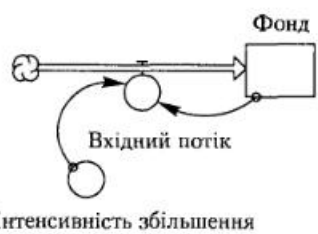


Рис. 6.12. Схема регулювання зростання фонду інтенсивністю його збільшення

Процес *вичерпування фондів* використовується в ситуаціях, коли на моделі потрібно регулювати процеси занепаду, старіння і спрацювання. На рис. 6.13 наведено приклад процесу вичерпування фонду згідно з виразом

```
вихідний потік = фонд * інтенсивність вичерпування.
```

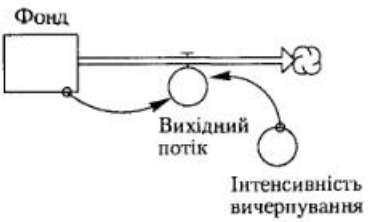


Рис. 6.13. Приклад процесу вичерпування фонду

Робочий процес найчастіше використовується в потокових структурах, що відтворюють семантику виробничої і ділової активності. Він застосовується в разі моделювання процесів зростання і зменшення вмісту фондів, коли режим функціонування визначається його заданим базовим ресурсом та інтенсивністю (продуктивністю).

На рис. 6.14 зображено кілька робочих процесів, інтенсивність яких обчислюється за формулою

```
робочий процес = ресурс * продуктивність.
```

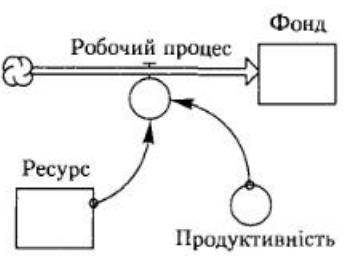


Рис. 6.14. Приклад робочого процесу

Процес *сполучення потоків* використовується за необхідності виразити факт взаємодії процесу з деяким іншим, керуючим процесом, який протікає паралельно в часі. Інтенсивність і характер впливу керуючого процесу визначається значеннями конверсійного коефіцієнта. На рис. 6.15 наведено приклади, які демонструють застосування базового процесу сполучення потоків у різноманітних контекстах і сферах ділових процесів, де параметри керування визначаються за формулою

```
керування = керуючий потік * конверсійний коефіцієнт.
```

Вважається, що взаємодіючі процеси протікають паралельно (робота на підприємстві (рис. 6.15, *а*) і навчання у вищому навчальному закладі (рис. 6.15, *б*), генерування об'єктного коду програми і виникнення помилок у текстах програм (рис. 6.15, *в*), збільшення кількості робітників і збільшення обсягів випуску

продукції (рис. 6.15, *г*)), однак один з них найчастіше є керуючим, від якого тією або іншою мірою залежить характер поведінки процесу керованого. Саме подібні несиметричні взаємодії визначають семантику сполучення потоків у рамках спеціальної конструкції базового процесу і можуть допомогти в дослідженні нетривіальних ситуацій.

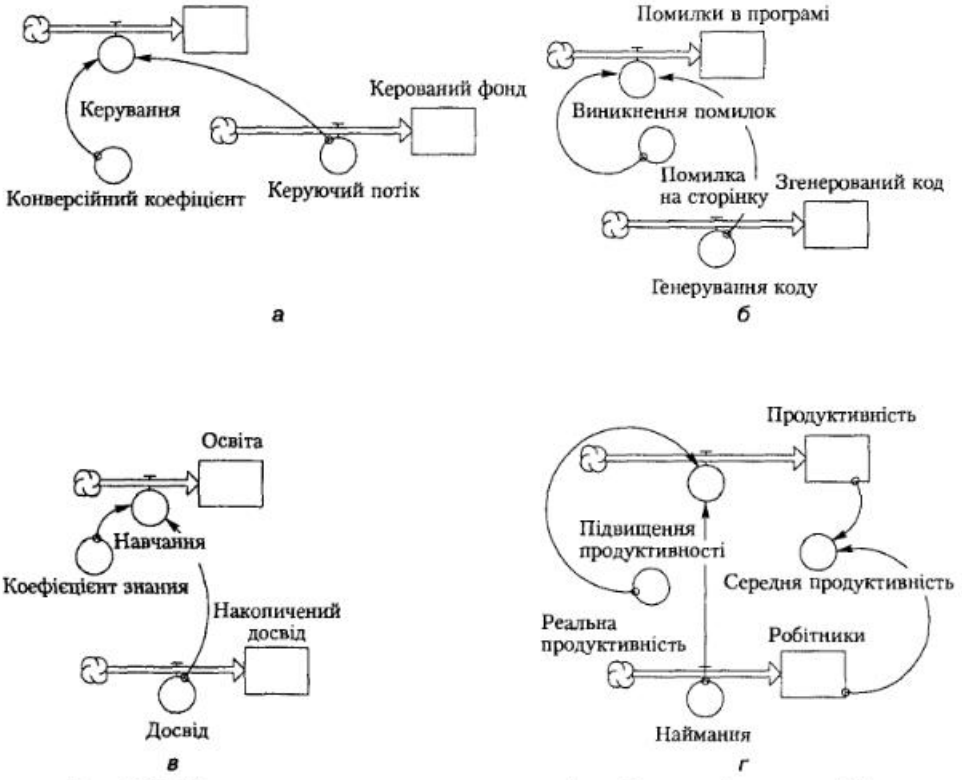


Рис. 6.15. Приклади процесу сполучення потоків: робота на підприємстві (*а*); генерування об'єктного коду програми і виникнення помилок у текстах програм (*б*); навчання у вузі (*в*); збільшення кількості робітників і обсягів випуску продукції (*г*)

Адаптація фондів – це потокова конструкція, яка застосовується за необхідності відобразити процеси, в яких уміст фонду адаптується до мінливих умов і приводиться в деякий стан рівноваги. Природно, що така інтерпретація можлива лише під час використання двоспрямованого потоку. Ця конструкція становить безсумнівний інтерес у проектуванні складних поведінкових моделей і вимагає до себе підвищеної уваги. Аналогії подібних модельних ситуацій можна зустріти в найрізноманітніших сферах. Так, процес адаптації фонду можна побачити під час спостереження за фізичними процесами нівелювання температури в замкнутому об'ємі газового потоку, у разі дослідження ефекту вирівнювання обсягу популяції в умовах установлення екологічної рівноваги.

На рис. 6.16 наведено приклади процесів адаптації фондів у періоди, які задаються значенням *t* адаптації, де змінна потік обчислюється за формулою

```
потік = (плановані значення вмісту фонду-фонд)/ t адаптації.
```

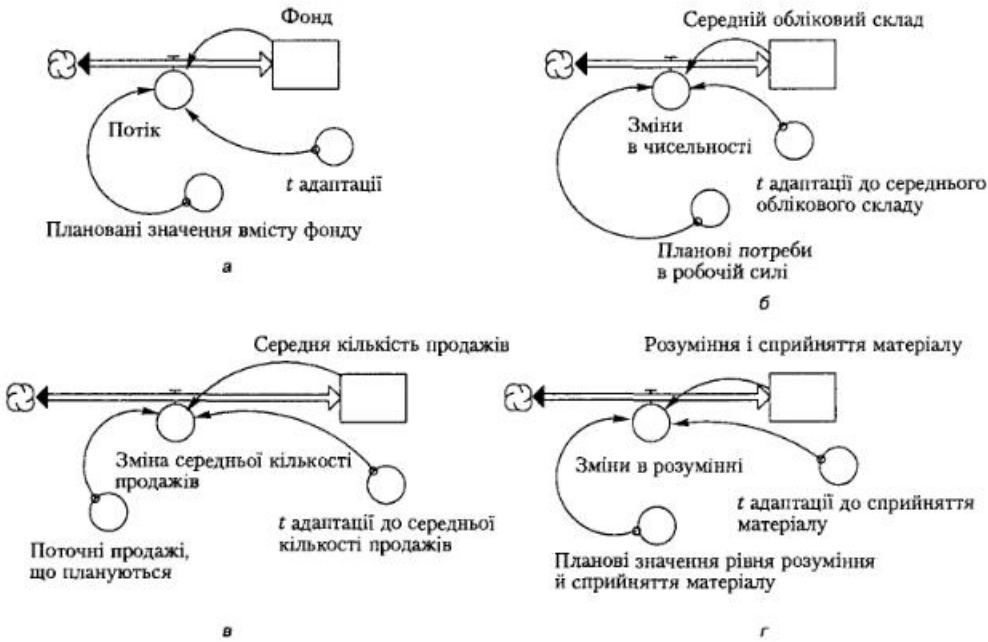


Рис. 6.16. Приклади процесів типу адаптації фондів: вірусне інфікування (*а*); видобування нафти (*б*); цикл популяції (*в*); встановлення ціни (*г*)

На рис. 6.16, *а* наведена загальна схема процесу адаптації. Дійсно на рис. 6.16. *б*, рівень фонду «Середній обліковий склад» стабілізується на певному рівні в міру збільшення часу адаптації, незважаючи на можливі різкі коливання планованих (щомісячних або сезонних) потреб у робочій силі. Середній обсяг продажу товару (рис. 6.16, *в*) за тривалий проміжок часу також буде плавно вирівнюватись, незважаючи на істотні коливання в планованих поточних продажах. Рівень фонду «Розуміння і сприйняття матеріалу» (рис. 6.16, *г*) у випадку запланованих коливань рівня освоєння навчального матеріалу також буде збігатись до деякого значення.

Моделі в системах Stella і iThink представляються трьома ієрархічними рівнями:

- блок-схемами;
- базовими потоковими схемами;
- формальними специфікаціями, які автоматично генеруються до вигляду, за якого вони готові до проведення імітаційних експериментів.

У блок-схемі моделі відображаються всі підмоделі, які взаємодіють між собою через потокові і конекторні зв'язки. Якщо дотримуватися методології організації низхідного проектування, кожну підмодель можна буде зобразити за допомогою базової потокової схеми.

6.17. Методи штучного інтелекту в імітаційному моделюванні

Важливість моделювання в дослідженнях методів штучного інтелекту безперечна, наприклад, у системах автоматизованого планування виробництва для складання плану використовується певний набір правил, які регламентують можливі дії для досягнення однієї або більше цілей. Запропонований план необхідно перевіряти на повноту, ефективність і реальність.

Отримати більш чітке уявлення про алгоритми функціонування інтелектуальної системи можна шляхом моделювання складних або критичних ситуацій (наприклад, виробничих планів, аварійних ситуацій і т. ін.). Однак багато розроблюваних систем штучного інтелекту не можна перевірити в реальних умовах, оскільки для цього часто потрібно створювати іншу систему або розробляти нове обладнання.

У свою чергу, методи штучного інтелекту також широко використовуються для імітаційного моделювання. Інтелектуальні системи можуть знадобитись для запускання програм імітації, визначення послідовності етапів прийняття рішень, моментів доступу до імітаційної моделі, а також для прийняття рішень щодо самих систем.

Одне з основних обмежень традиційних методів моделювання – це нездатність відтворити інтелектуальну (розумну) поведінку людини. Тому на практиці в разі моделювання поведінки людей часто використовуються спрощені моделі прийняття рішень (наприклад, задається якась традиційна стратегія, а не моделюються всі етапи прийняття рішень). Цей підхід дає змогу проаналізувати ті аспекти системи, які часто ігноруються (вважаються незначущими) або які занадто складно моделювати.

Іноді доводиться імітувати адаптивну поведінку системи, коли її параметри залежать від поточного стану. У цьому випадку прості правила прийняття рішень (наприклад, завжди ставати в найкоротшу чергу) часто неадекватні. Ще важливішим фактором є те, що визначальний елемент багатьох систем – це людина, яка приймає рішення і значною мірою контролює процеси в системі, наприклад хід виробництва. Щоб імітувати таку систему, потрібно під час моделювання мати зв'язок з особою, яка приймає рішення, або ж у моделі врахувати поведінку. Останній підхід було дуже вдало використано в системі візуального інтерактивного моделювання – VIS.

Розглянемо деякі підходи до моделювання, запропоновані спеціалістами в галузі штучного інтелекту, а також спеціальні прикладні програми, написані під час проведення цих досліджень.

Порівняно новим і прогресивним підходом до моделювання з позицій штучного інтелекту є створення інтелектуальних оболонок для існуючих імітаційних пакетів. Інтелектуальна оболонка займає проміжне місце між пакетом і користувачем, генерує необхідні інструкції щодо користування пакетом, консультує користувача, інтерпретує та пояснює результати моделювання.

Кількість програмних засобів моделювання різко зросла з того часу, як почали застосовувати методи штучного інтелекту. Найкращими прикладами таких засобів ймовірно є системи ROSS, KBS, SimKit і T-Prolog [73] (існують й інші систе-

ми, наприклад HIRES, SIMYON, BLOBS, але вони або є експериментальними або важкодоступними).

Існують системи ROSS, KBS і SimKit з деяким об'єктно-орієнтованим розширенням. Вважається, що першу об'єктно-орієнтовану мову SIMULA [4] було розроблено як мову моделювання дискретних систем. Проте жодна з вищенаведених систем з їх декларативною структурою і визначеними засобами передача повідомлень не схожа на пакет SIMULA.

Система ROSS, розроблена Rand Corporation AI Group, є не тільки першим засобом моделювання, який базується на технологіях штучного інтелекту, але і найбільш повно розробленим. Це високо інтерактивний засіб, написаний на мові Lisp, тому що ROSS-програму можна зупинити і дослідити в будь-який момент часу. Користувач, як і будь-який інший об'єкт, може обмінюватись повідомленнями з моделлю. Крім того, код програми досить зрозумілий і добре структурований, що дозволяє проектувати великі імітаційні моделі.

Система KBS, розроблена компанією Reddy and Fox in Carnegie-Mellon, застосовує мову у вигляді схем (це мова представлення знань, яка використовує фрейми і транслятор з мови Lisp) для того, щоб зобразити процес моделювання і включити в нього елементи об'єктного програмування. На відміну від системи ROSS, яка в основному є дискретно-подійною, KBS базується на концепції динамічних систем. Особливий наголос зроблено на методах автоматичного аналізу моделей.

Система SimKit – це продукт корпорації IntelliCorp, написаний на платформі KEE (Knowledge Engineering Environment – середовище побудови знань) і мови Lisp. Фрейми, які зображують об'єкти, використовуються для побудови імітаційних моделей. У системі SimKit є можливість розробки спадкоємної імітації, в якій специфічні моделі можна сконструювати шляхом маніпулювання вікнами.

Система T-Prolog забезпечує логічну основу для моделювання. Її розроблено фахівцями Інституту координації комп'ютерних технологій (Institute for Coordination of Computer Techniques) Угорщини. Система являє собою розширення мови Prolog. Вже є дискретно-неперервна версія, що має назву TC-Prolog. Програма моделювання конструюється за допомогою операторів мови Prolog з використанням додаткових можливостей T-Prolog, які включають базові дискретно-подійні засоби (наприклад, «чекати деякий час», «зайняти пристрій» і т. ін.). Прогін моделі можна простежити в прямому і зворотному напрямках, випробовуючи різні варіанти моделі.

Розробка системи PROSS (Prolog Simulation System) – це спроба створити систему, яка допускає моделювання процесів прийняття рішень у системах штучного інтелекту. Цей підхід базується на впровадженні в Prolog елементів мови GPSS. У такий спосіб методи штучного інтелекту спочатку можна реалізувати за допомогою операторів мови Prolog, а потім інтегрувати їх з імітаційною моделлю. Мова GPSS забезпечує добре відому схему моделювання і зменшує перешкоди у використанні системи PROSS.

Систему PROSS написано на базі мови HC-Prolog, яка за синтаксисом подібна до мови Lisp. HC-Prolog досить суттєво відрізняється від версії стандартної мови Edinburgh Prolog. Програми генерування випадкових величин і статистичного

аналізу інформації впроваджено шляхом приєднання до HC-Prolog засобів із бібліотеки Pascal-SIM.

Інтелектуальне середовище імітаційного моделювання РДО (Ресурси–Дії–Операції) [12] створено в Московському державному технічному університеті (МДТУ ім. М. Е. Баумана) на кафедрі «Комп'ютерні системи автоматизації виробництва». Поштовхом до розроблення середовища РДО стали вимоги до універсальності імітаційного моделювання щодо класів модельованих систем і процесів, простоти модифікації моделей, моделювання складних систем керування паралельно з об'єктом керування (включаючи імітаційне моделювання в реальному масштабі часу) і ряд інших. Ці вимоги було сформульовано розробникам під час виконання досліджень із системного аналізу і керування складними технічними та організаційними системами різної природи.

Основні сфери застосування системи РДО – моделювання складних систем керування, кількісний аналіз і оптимізація дискретних процесів, оптимізація прийняття рішень, оцінювання альтернативних варіантів проектів, прогнозування економічних процесів, навчальні системи, тренажери та ігри тощо.

Під час розв'язування завдань у цих сферах необхідно враховувати фактори невизначеності, динамічну взаємну обумовленість поточних рішень і прогнозованих подій, комплексну взаємозалежність між параметрами системи, а часто й чітко визначені послідовності дискретних проміжків часу.

Для опису функціонування складних систем у середовищі РДО використовуються модифіковані продукційні правила [13], які дають змогу описувати знання, необхідні для прийняття рішень. Продукційні правила мають такий вигляд:

```
Якщо <умова> то <подія 1>,
чекати Δ t = φ (t, стан системи), то <подія 2>.
```

Протягом певного періоду часу подія може включати кілька дій, які змінюють стани модельованих процесів, об'єктів, систем. Опис модельованого об'єкта мовою інтелектуального середовища РДО здійснюється в такій послідовності.

1. Визначаються типи ресурсів, необхідних для моделювання.
2. Встановлюються ресурси відповідно до визначених типів і задаються початкові значення параметрів ресурсів.
3. Окреслюється множина операцій, які буде виконувати система.
4. Описуються передумови виконання кожної операції та правила зміни станів ресурсів під час виконання операції.
5. Обираються показники функціонування системи, необхідні для статистичного аналізу результатів моделювання, зміст файлу трасирування ресурсів і станів системи.

Інтелектуальне середовище РДО дає змогу описувати дії, визначені користувачем, константи і послідовності випадкових величин, показники роботи моделі, а також файли анімації, що відображаються під час прогону моделі.

Особливістю системи моделювання є наявність двох режимів моделювання – режиму прогону моделі і режиму експерта. У деяких ситуаціях, коли необхідне

втручання користувача у функціонування системи, вона переходить у режим експерта, користувач приймає рішення, а потім знову переводить систему в режим прогону для продовження моделювання.

Спрощене зображення знань мовою інтелектуального середовища РДО дає змогу розробнику моделей зосередитися на предметній області, а не на програмуванні.

Висновки

Розглядаючи засоби імітаційного моделювання, можна зробити висновок, що їх розвиток нерозривно пов'язаний з удосконаленням комп'ютерів і засобів програмного забезпечення. Назвемо основні напрями розвитку систем імітаційного моделювання:

- вдосконалення засобів паралельного моделювання та використання веб-технологій;
- втілення засобів штучного інтелекту не тільки під час створення моделей, але й у разі обробки результатів моделювання та прийняття рішень;
- інтеграція засобів моделювання із засобами оптимізації за результатами моделювання;
- розширення можливостей автоматизації всіх етапів моделювання, починаючи з формулювання проблеми моделювання і закінчуючи прийняттям рішень за результатами моделювання.

Контрольні запитання та завдання

1. Охарактеризуйте етапи розвитку засобів імітаційного моделювання. Чи пов'язані ці етапи з розвитком комп'ютерної техніки?
2. Який внесок українських фахівців у розвиток засобів імітаційного моделювання?
3. Які переваги надає застосування об'єктно-орієнтованого підходу під час побудови засобів імітаційного моделювання?
4. Наведіть приклади застосування потокових моделей. Використовуючи потокові моделі, побудуйте модель бюджету своєї родини на один рік. Врахуйте витрати на відпустку.
5. Які проблеми виникають під час розроблення паралельних систем імітаційного моделювання та як вони вирішуються?
6. Назвіть переваги використання архітектури високого рівня під час розроблення складної імітаційної моделі (наприклад, моделі літака).
7. Наведіть приклади застосовування елементів штучного інтелекту в системах імітаційного моделювання.

Розділ 7

Планування та проведення експериментів з моделями

- Оцінювання точності результатів моделювання
- Методи зниження дисперсії
- Повний і дробовий факторні експерименти
- Пошук оптимумів на поверхні відгуку
- Методи прискорення імітаційного моделювання

Усяке наукове дослідження, що проводиться за допомогою моделі, передбачає планування та проведення експериментів, у ході яких збираються необхідні дані. Ці дані підлягають аналізу, і на його основі визначають, чи можна під час моделювання досягти поставлених цілей. Якщо експеримент не сплановано належним чином, існує небезпека того, що не буде отримано бажаних результатів або результати виявляться помилковими. Це може призвести до прийняття неправильних рішень чи необхідності повторного проведення експерименту (найбільш трудомістка операція) і навіть до перебудови моделі. Разом з тим повинна існувати можливість повторення іншими дослідниками експерименту, результати якого вважаються значущими.

7.1. Проблеми планування імітаційних експериментів

Імітаційні моделі відтворюють процеси, що протікають у реальній системі. У тому випадку, коли в моделі враховуються випадкові фактори, під час моделювання необхідно провести велику кількість прогонів моделі, як із різними вхідними даними, так і з різними послідовностями випадкових чисел. Для моделей детермінованих систем, у яких не враховуються випадкові фактори, зазвичай досить лише одного прогону моделі для всіх допустимих комбінацій вхідних даних і параметрів моделі. Однак на практиці такі моделі майже не зустрічаються.

У ході експерименту з моделлю одержують безліч даних, які мають бути структуровані та інтерпретовані для використання під час прийняття рішень стосовно структури та параметрів системи чи моделі. Для того щоб правильно інтерпретувати отримані вихідні дані, необхідно планувати проведення експериментів з моделлю.

Планування експерименту – це розроблення такого плану експерименту, який дає можливість за мінімальної кількості прогонів моделі і за мінімальних затрат ресурсів зробити статистично значимі висновки або знайти оптимальні рішення щодо функціонування системи. Під час планування експериментів, як правило, визначають:

- вхідні дані для кожного експерименту і кількість прогонів імітаційної моделі;
- тривалість одного прогону моделі і перехідного процесу моделювання;
- стратегію збирання даних під час кожного прогону моделі;
- методи оцінювання точності вихідних даних і побудови довірчих інтервалів;
- чутливість моделі до вхідних даних, різних видів розподілів випадкових величин, сценаріїв поведінки модельованої системи;
- умови і сценарії проведення експерименту;
- умови генерування потоків випадкових чисел у системі моделювання та імовірнісних вхідних даних;
- стратегію досягнення мети експерименту (наприклад, порівняння альтернативних варіантів системи або оптимізація цільової функції).

Кінцева мета проведення експериментів – це одержання статистичної інформації, достатньої для прийняття рішень відповідно до результатів моделювання. Моделювання здебільшого провадиться з метою визначення деяких екстремальних значень характеристик модельованої системи (*оптимізуючий* експеримент) або для виявлення важливих факторів, які впливають на модельовану систему (*відсівний* експеримент). Під час експериментів обох типів використовують факторні плани й будують поліноми різного порядку, які апроксимують поверхню відгуку. Для пошуку екстремальних значень застосовуються числові методи оптимізації. Під час таких експериментів визначається функціональна залежність вихідної змінної (функції відгуку, чи просто відгуку) від вхідних змінних, або факторів; ця залежність відображає критерій ефективності модельованої системи. Таким чином, пошук найкращого рішення характеризується числовим значенням цього критерію, і для знаходження екстремальних значень необхідно досліджувати поверхні відгуку (провадити експерименти) у різних точках факторного простору. Ефективність проведення експериментів багато в чому залежить від початкової точки у факторному просторі.

Один із найважливіших видів експериментів, проваджуваних з моделлю, – це *структурна оптимізація* [63], під якою будемо розуміти пошук найкращої структури модельованої системи. У цьому випадку аналізується кілька моделей, причому вони можуть відрізнятися структурою, параметрами і алгоритмами функціонування. Для таких експериментів не існує єдиного числового критерію оптимізації, що ускладнює використання класичних методів. Однак під час такого дослідження кількість моделей, як правило, невелика, тому для структурної оптимізації можна скористатися методом висування гіпотез або простим перебиранням варіантів.

Оптимізація єдиного варіанта модельованої системи провадиться за допомогою пошуку вузьких місць і їх усунення, тобто балансування модельованої системи. Вузькі місця визначають пропускну здатність усієї системи (див. розділ 2.4), і пошук найкращого рішення здійснюється шляхом порівняння розглянутих варіантів.

Перелічимо основні проблеми, які виникають під час проведення експериментів з імітаційними моделями.

1. Визначення початкових умов проведення експерименту.

   Зазвичай експеримент починають, коли модель перебуває в стані «пусто і вільно», тобто в моделі немає динамічних об'єктів або транзактів і всі пристрої та ресурси вільні. Якщо розглядається досить тривалий період моделювання, то можна задати так званий період «розігріву» чи «розгону» моделі, або *перехідний процес*, після якого модель переходить у сталий (стаціонарний) режим роботи. Урахування даних перехідного процесу для оцінювання вихідних змінних моделі спричинює зміщення статистичних оцінок параметрів.

   Щоб зменшити вплив вихідних даних перехідного процесу на кінцеві результати, моделювання слід починати з використання модальних (найбільш імовірних) або середніх значень сталого режиму. Такий спосіб запуску моделі забезпечує зменшення тривалості перехідного процесу моделі, але застосування даного способу ефективне лише в тому випадку, коли завантаження пристроїв обслуговування в моделі невелике. У разі наближення коефіцієнтів завантаження пристроїв до одиниці на виході моделі можна спостерігати стаціонарний процес, під час якого неможливо чітко визначити дані перехідного процесу (рис. 7.1).

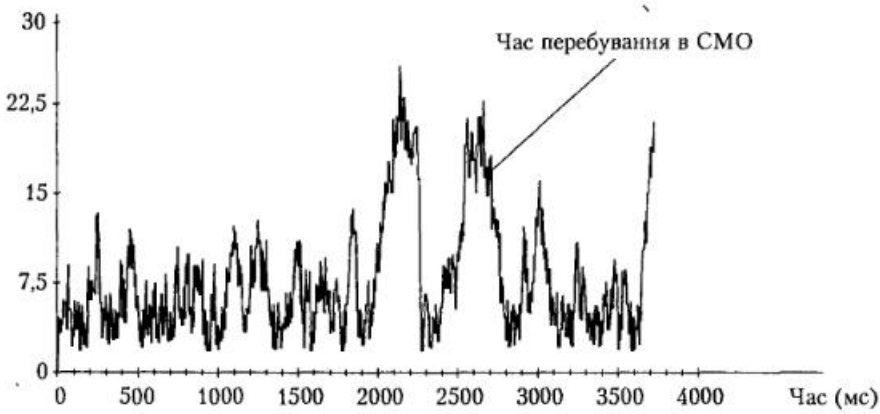


Рис. 7.1. Стаціонарний процес виходу із СМО з близьким до одиниці коефіцієнтом завантаження

У разі оцінювання статистичних параметрів вихідних величин рекомендується не враховувати дані перехідного процесу, оскільки вони можуть викликати істотне зміщення шуканих оцінок. Усунення зміщення досягається шляхом відкидання даних перехідного процесу (в мові GPSS це можна зробити, використавши команду RESET). Найскладнішим є встановлення моменту

досягнення сталого стану системи. Досі не існує цілком надійних методів визначення цього моменту [67]. Однак дана проблема може бути вирішена за допомогою діалогових та інтелектуальних систем моделювання, які дають змогу контролювати і графічно відображати хід моделювання. Найефективніший спосіб визначення початку сталого режиму – це спостереження за графіками зміни вихідного процесу в часі (рис. 7.2).

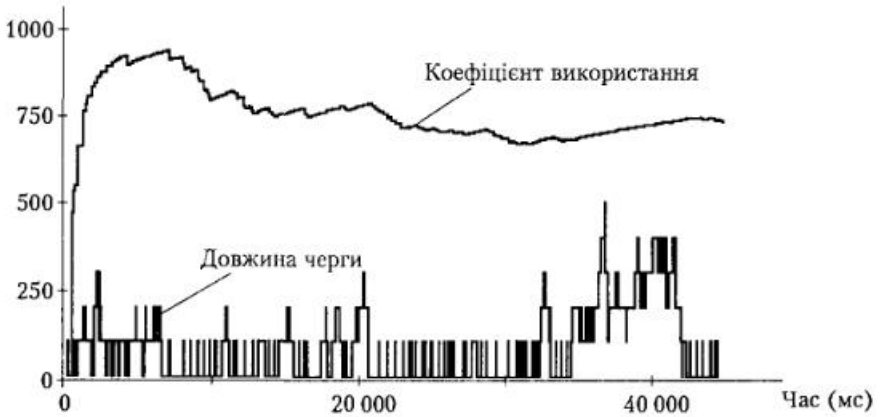


Рис. 7.2. Графіки вихідних даних моделі СМО

2. Зупинення процесу моделювання.

Правила зупинення процесу моделювання дозволяють визначити тривалість прогону імітаційної моделі – а від цього залежить точність результатів. Наприклад, якщо потрібно провести моделювання роботи виробничої дільниці протягом робочого тижня, то час прогону моделі можна визначити саме цим терміном. Дослідник сам приймає рішення, чи буде досягнуто за вказаний період моделювання сталий режим роботи моделі. Він також обирає метод збирання вихідних даних і оцінювання точності результатів моделювання. Точність оцінювання параметрів системи в цьому випадку визначається за одним досить тривалим прогоном моделі.

Визначають два типи імітаційного моделювання – *скінченне* і *нескінченне*. Для першого процес моделювання закінчується, коли відбувається «природна» подія, яка дає змогу визначити тривалість прогону моделі. Така подія часто відбувається в той момент, коли система звільняється від вимог, чи в момент, після якого вже не можна одержати корисної інформації, або моделювання закінчується за показниками таймера. Така подія визначається до виконання прогонів моделі, її настання має бути детермінованою (згідно з таймером) або випадковою величиною. Оскільки початкові умови скінченного моделювання, як правило, впливають на критерії оцінювання, вони мають представляти умови, характерні для роботи реальної системи.

Під час виконання нескінченного імітаційного моделювання не існує події, настання якої давало б змогу визначати тривалість прогону моделі. Моделювання цього типу використовується у разі дослідження поведінки системи за умов

сталого режиму її роботи протягом тривалого часу. Параметри, що оцінюються в цьому випадку, вважаються сталими, якщо вони залежать від характеристик сталих законів розподілів імовірностей деякого вихідного процесу.

3. Стан моделі в момент припинення прогону.
   Під час моделювання завжди виникає питання щодо доцільності використання динамічних компонентів, або транзактів, які залишилися в моделі після закінчення її роботи. Урахування характеристик цих компонентів може призвести до збільшення зміщення статистичних оцінок параметрів моделі. Наприклад, під час моделювання роботи дільниці цеху було зроблено припущення, що найменш тривалі роботи виконуються в першу чергу. Тоді на момент закінчення моделювання може виникнути ситуація, коли в моделі залишаться тільки роботи, термін виконання яких великий. Якщо їх не враховувати, оцінка середньої тривалості виконання робіт у цеху буде заниженою.
4. Визначення тривалості прогону моделі за наявності в ній процесів з різними швидкостями протікання.
   Для оцінювання точності результатів моделювання здебільшого використовують параметри найповільнішого процесу в моделі. У цьому випадку оцінки для більш швидких процесів будуть набагато кращими і ефективнішими, ніж для повільних, тобто довірчі інтервали для останніх будуть більшими. Під час розроблення імітаційної моделі обирають такий ступінь її деталізації, щоб швидкості процесів, які протікають у моделі, не відрізнялися більш ніж на два порядки. У разі необхідності моделювання рідких подій або повільних процесів, наприклад відмов устаткування, потрібно укрупнювати стани для швидких процесів. Для цього використовують аналітико-імітаційні моделі.

7.2. Оцінювання точності результатів моделювання

Оцінити точність результатів моделювання можна шляхом побудови довірчих інтервалів для вихідних змінних (відгуків) моделі. Точність результатів залежить насамперед від кількості реалізацій (прогонів моделі) і тривалості прогону для кожної реалізації моделі. Якщо модель детермінована, то для отримання точних результатів моделювання достатньо одного прогону. У загальному випадку дані спостереження одного прогону моделі представляють одиничну вибірку або часовий ряд. *Часовий ряд* – це реалізація випадкового процесу. В результаті кожного прогону моделі утворюються часові ряди для кожного значення відгуку моделі досліджуваних випадкових процесів.

Під час моделювання стохастичних систем потрібно розглядати два режими роботи моделей: *перехідний* і *стаціонарний*. Стаціонарний режим визначається сталим процесом на виході моделі. Слід зазначити, що для більшості реальних систем характеристики стохастичних процесів, у тому числі й закони розподілу, з часом змінюються. За наявності нової інформації відносно параметрів системи потрібно повторно проаналізувати сталі параметри моделі.

7.2.1. Перехідний режим роботи моделі

Для більшості виробничих систем неможливо гарантувати, що вони працюватимуть у стаціонарному режимі. Винятком є системи роботизованого автоматичного виробництва. Якщо модель працює в перехідному режимі, то необхідну кількість прогонів моделі можна розрахувати за тими ж формулам, що і для методу статистичних випробувань; при цьому під час кожного прогону моделі з використанням однакових вхідних даних і параметрів формується своя послідовність випадкових чисел. Оскільки випадкові величини незалежні, то незалежними мають бути й отримані вихідні дані для кожного прогону моделі. Це дає змогу побудувати довірчий інтервал, скориставшись центральною граничною теоремою. Кількість прогонів моделі визначається за формулами (4.23) і (4.26). Необхідну точність ε можна задати, наприклад, такою, що дорівнює ± 5 % середнього значення величини, для якої будується довірчий інтервал, якщо $\alpha = 0{,}05$.

Після проведення прогонів моделі розраховуються оцінки загального середнього значення вихідної змінної та середньоквадратичного відхилення і будується довірчий інтервал для середнього значення. Більшість програмних засобів імітаційного моделювання забезпечують автоматичне проведення таких розрахунків. Наприклад, якщо модель реалізовано мовою GPSS World, то після останнього прогону достатньо викликати процедуру ANOVA, яка і побудує довірчі інтервали для вихідних змінних.

Якщо кількість прогонів невелика (менше ніж 30), то для побудови довірчого інтервалу використовують розподіл Стьюдента (t-розподіл). За наявності більшого числа прогонів для визначення значення t_α можна використовувати нормальний розподіл.

Процедура повторних прогонів (реплікацій) має важливу властивість – незалежність вибірок. Ця властивість є універсальною і може застосовуватись для моделювання як перехідного, так і стаціонарного режиму роботи моделі. Слід зауважити, що в разі моделювання стаціонарного режиму ця процедура дає змогу зібрати дані й перехідного періоду, які не використовуються під час аналізу стаціонарного режиму.

7.2.2. Стаціонарний режим роботи моделі

Під час моделювання деяких систем, наприклад комп'ютерних та комунікаційних, виникає потреба аналізувати їх роботу в сталому, стаціонарному режимі. Існування такого режиму в системі дає змогу побудувати довірчий інтервал для оцінок параметрів за результатами не багатьох прогонів моделі, а лише одного, досить тривалого.

Основна проблема, пов'язана з побудовою довірчого інтервалу, обумовлена тим, що вихідні дані імітаційної моделі є корельованими. Крім того, наявність перехідних процесів у моделі призводить до зміщення статистичних оцінок. На жаль, не існує надійних методів виявлення моменту завершення перехідного періоду роботи моделі. Якщо критерієм оцінювання є вартісна характеристика (прибуток, витрати та ін.), яка визначається для стаціонарного режиму роботи моделі, тривалість прогону може бути визначена на основі результатів спостереження за зміною

величини, що дорівнює відношенню оцінюваного показника за весь період моделювання до тривалості моделювання (наприклад, витрати за одиницю часу), для якої будують графік зміни в модельному часі. Оскільки тривалість перехідного періоду може змінюватись в залежності від комбінацій вхідних змінних моделі і послідовностей випадкових чисел, потрібно визначити найдовший перехідний період. Статистичні дані перехідного періоду роботи моделі не повинні враховуватись під час розрахунків статистичних оцінок для вихідних змінних.

Необхідно, щоб тривалість прогону відповідала сталому режиму функціонування моделі (рис. 7.3). На практиці діють таким чином. За графіком вихідної змінної моделі визначають час моделювання, коли закінчується перехідний період $t^{\text{пер}}$. Статистичні дані, зібрані за цей період, не враховуються під час статистичного аналізу. Наприклад, у мові GPSS це робиться за допомогою команди RESET, яка видаляє накопичені статистичні дані, але залишає транзакти в моделі, фіксує поточні значення довжин черг і максимальний вміст блоків STORAGE. Після цього повторно викликається команда START, що задає тривалість прогону моделі не меншою ніж $100t^{\text{пер}}$. Кількість таких прогонів, необхідну для оцінювання вихідних змінних, визначають за формулою (4.26).

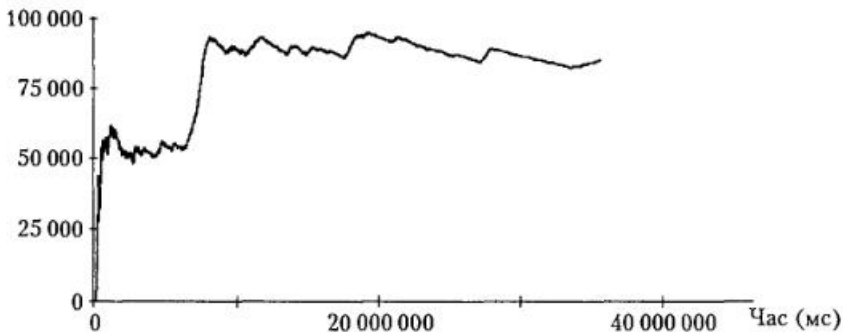


Рис. 7.3. Визначення сталого режиму роботи моделі

Розглянутий вище спосіб є наближеним, але він широко використовується на практиці. Існують інші методи, застосування яких дає змогу отримати більш надійні оцінки для характеристик стаціонарних процесів. У книзі А. Лоу і В. Келтона [18] описується шість методів для обчислення оцінки стаціонарного середнього випадкового процесу, серед яких основну увагу приділено методу реплікації і вилучення.

7.2.3. Метод реплікації і вилучення

Метод реплікації і вилучення, за умов правильного його використання, дає змогу отримати досить хороші статистичні оцінки; його можна легко реалізувати. Аналіз випадкових процесів за цим методом подібний до того, який застосовують під час скінченного моделювання, однак для отримання оцінок використовуються ті спостереження, які не належать до перехідного періоду (тобто отримані за період $t^{\text{пер}}$) кожного прогону імітаційної моделі.

Припустимо, що під час моделювання виконано n повторних прогонів імітаційної моделі, тривалість кожного прогону дорівнює m' спостереженням, де m' істотно перевищує перехідний період роботи $n_{\text{пер}}$, визначений графічним методом. Нехай Y_{ji} – це дані i-го спостереження в ході j-го повторного прогону імітаційної моделі ($j = 1, ..., n$; $i = 1, ..., m$). Позначимо через X_j випадкову величину, яка обчислюється за результатами j-го прогону за формулою

$$X_j = \frac{\sum_{i=n_{\text{пер}}+1}^{m'} Y_{ji}}{m' - n_{\text{пер}}}, \quad j = 1, ..., n.$$

Для оцінювання X_j використовуються тільки ті спостереження з j-го прогону моделі, які проводились під час сталого режиму роботи, а саме, $Y_{j,l+1}$, $Y_{j,l+2}$, ..., $Y_{j,m}$. Тоді X_j є незалежною і однаково розподіленою випадковою величиною, для якої $M[X_j] \approx \mu$, а $\overline{X}(n')$ – це наближена незміщена точкова оцінка для μ. Наближений $100(1-\alpha)$ %-й довірчий інтервал для μ визначається за формулою

$$\overline{X}(n') \pm t_{n'-1,\,1-\alpha/2} \sqrt{\frac{S^2(n')}{n'}},$$

де $\overline{X}(n')$ і $S^2(n)$ відповідно – середнє значення та вибіркова дисперсія випадкової величини.

Цей метод передбачає використання n пробних прогонів (реплікацій) для визначення тривалості перехідного процесу $n_{\text{пер}}$ і тільки $m' - n_{\text{пер}}$ спостережень з інших n робочих прогонів (реплікацій) – для статистичного аналізу.

У деяких випадках для того, щоб визначити величину $n_{\text{пер}}$ і побудувати довірчий інтервал, потрібно використовувати n початкових пробних прогонів моделі тривалістю m' спостережень. Зокрема, якщо m' значно більше, ніж вибрана тривалість перехідного періоду $n_{\text{пер}}$, можна використати кілька початкових прогонів моделі для розв'язання обох завдань. Оскільки графічний метод є наближеним, невелика кількість спостережень, що не належать до перехідного періоду, може спричинити суттєвий зсув оцінок параметра μ. Проте, якщо значення m' набагато більше $n_{\text{пер}}$, ці спостереження відчутно не впливатимуть на загальну якість оцінки (тобто значення зміщення) величини X_j (отриманої на основі $m' - n_{\text{пер}}$ спостережень) або $\overline{X}(n)$. Таким чином, статистично більш точні результати можна отримати за методом реплікації і вилучення з двома незалежними множинами прогонів моделі.

Величина половини довжини довірчого інтервалу, визначеного за допомогою методу реплікації і вилучення, залежить від значення дисперсії $D[X_j]$ випадкової величини X_j, яке є невідомим під час перших n прогонів. Отже, якщо виконується задане число повторних прогонів імітаційної моделі, отримане значення половини довжини довірчого інтервалу може бути недостатньо малим для досягнення визначеної мети. Однак відомо, що довжина довірчого інтервалу має квадратичну залежність від кількості повторних прогонів. Таким чином, можна розрахувати необхідну кількість повторних прогонів.

7.2.4. Ергодичні процеси

Перевірити, чи є вихідний процес моделі стаціонарним, можна також, спостерігаючи за поведінкою його автоковаріаційної функції. Наявність тенденції до її затухання у більшості випадків свідчить про те, що процес є стаціонарним. Оскільки аналітичний вираз для автоковаріаційної функції випадкового процесу зазвичай невідомий, оцінити можна лише її значення. Враховуючи те, що отримані під час імітаційного моделювання часові ряди мають достатню довжину, автоковаріаційну функцію оцінюють за формулою [46]

$$\mathrm{cov}_h = \frac{1}{N}\sum_{i=1}^{N-h}(X_i X_{i+h}) - \frac{1}{N-h}\left(\sum_{i=1}^{N-h} X_i\right)\left(\sum_{i=1}^{N-h} X_{i+h}\right),$$

де N – кількість точок у часовому ряді; h – значення зсуву по ряду; X_i та X_{i+h} – i-те та $i+h$-те значення змінної.

Під час імітаційного моделювання важливе значення мають стаціонарні ергодичні процеси, характеристики яких можуть бути оцінені за результатами лише одного часового ряду.

Послідовність вибіркових середніх значень $\{\overline{X}_N\}$, $N = 1, 2, \ldots$ є *ергодичною*, якщо дисперсія величини $\overline{X}_N$ прямує до нуля в разі нескінченного збільшення N (рис. 7.4, *б*). Таким чином, вибіркове середнє асимптотично прямує до математичного сподівання, якщо дисперсія вибіркового середнього прямує до нуля. У цьому випадку стаціонарний режим роботи моделі не буде залежати від початкових умов моделювання.

Щоб перевірити на практиці, чи є процес ергодичним, потрібно суттєво змінити початкові умови імітаційного експерименту. Наприклад, якщо під час запуску першого прогону моделі черги були порожніми, а прилади обслуговування – вільними, то другий прогін моделі необхідно провадити за наявності великої кількості вимог або транзактів у чергах і пристроях. Якщо під час моделювання в обох випадках буде отримано близькі результати, то це, як правило, свідчитиме про ергодичність процесів моделі.

У разі використання мови GPSS змінити умови моделювання можна, задавши кілька блоків GENERATE і TRANSFER:

```
GENERATE ...,N
TRANSFER , ПОМІТКА
```

Параметр ПОМІТКА позначає вхід пристрою, а N – кількість транзактів, які передаються у пристрій.

Більш точно перевірити, чи є будь-який випадковий процес ергодичним і стаціонарним, можна, розрахувавши його автокореляційну функцію, яка в цьому випадку повинна бути згасаючою (рис. 7.4, *в*). Врахуйте, що дані перехідного періоду мають бути вилучені.

Для оцінювання точності характеристик випадкових стаціонарних процесів (рис. 7.4, *а*) можна скористатись однією достатньо довгою реалізацією, де довжина прогону з m спостереженнями визначається за формулою

$$m = Nm' - n_{\text{пер}}.$$

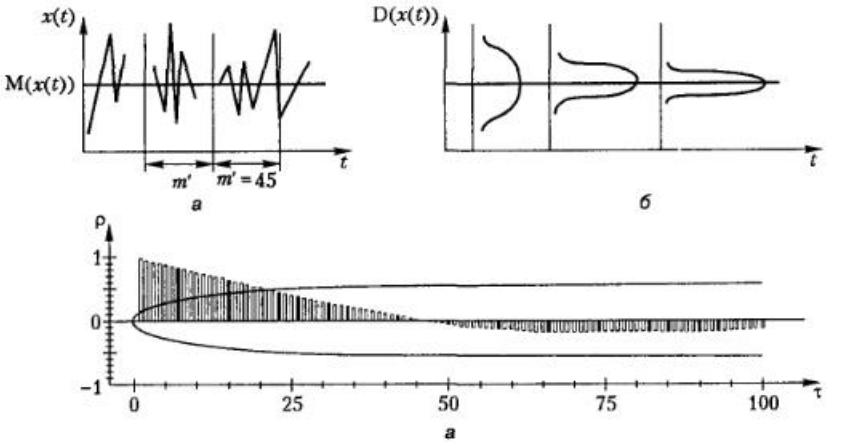


Рис. 7.4. Графіки ергодичного процесу (*а*), його дисперсії (*б*) та автокореляційної функції (*в*)

Цей метод також передбачає використання n реплікацій (пробних прогонів) для оцінювання тривалості перехідного періоду $n_{пер}$. Кількість елементів часового ряду, які належать до інтервалу m', обчислюють, застосовуючи автокореляційну функцію ρ за умови, що значення $\rho \to 0$ (на рис. 7.4, *в* $m' = 45$). Отже, всі значення часового ряду, крім $n_{пер}$ початкових значень, об'єднуються у N блоків фіксованої довжини m'. Кількість блоків N також задається.

Далі діють за таким правилом: середнє за часом дорівнює середньому по множині. Кількість елементів, які входять у кожну послідовність m', однакова, тобто отримана вибірка розбивається на підвибірки (блоки) фіксованого розміру. Середні значення в кожному блоці будуть незалежними однаково розподіленими випадковими величинами, тому величину N можна розрахувати так само, як і для методу статистичних випробувань, за формулою (4.26).

7.2.5. Регенеративні процеси

Розглянемо модель регенеративного процесу СМО з одним пристроєм [28] типу $D/G/1$, зображену на рис. 7.5, де D – детермінований вхідний потік, G – довільний закон обслуговування.

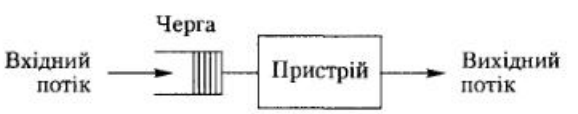


Рис. 7.5. Модель системи масового обслуговування

Нехай інтенсивність детермінованого потоку $\lambda = 60$ с, тривалість обслуговування має рівномірний розподіл в інтервалі 10...60 с. Мета моделювання – оцінювання значення математичного сподівання часу перебування вимоги в черзі $M\{W\}$ з деяким ступенем надійності оцінки. Теорія масового обслуговування не дає

змоги розв'язати аналітично цю задачу, коли час надходження вимог у систему є детермінованим. Тому проведемо дослідження за допомогою імітаційної моделі.

Нехай за період моделювання надійшло N вимог, тоді середній час перебування в черзі можна визначити як

$$W = \frac{\sum_{i=1}^{N} W_i}{N},$$

де W_i – час чекання i-ї вимоги, $i = 1, ..., N$.

Ця оцінка за умови, що $N \to \infty$ з імовірністю 1 буде збігатися до математичного сподівання M{W}. Проте ця вибіркова оцінка буде зміщеною.

На основі результатів моделювання потрібно побудувати 95 %-й довірчий інтервал для істинного значення M(W). Причому в разі будь-якого незалежного повторення прогонів моделі ймовірність того, що довірчий інтервал містить істинне значення M{W}, дорівнюватиме 0,95.

У даному випадку скористатись стандартними статистичними методами не можна, тому що значення W_k і W_{k-1} є сильно корельованими. Таким чином, наявність кореляційного зв'язку і зміщенності – серйозні перешкоди, які виникають під час оцінювання вибіркового середнього. Це ставить під сумнів значимість результатів проведення імітаційних експериментів.

Припустимо, що моделювання розпочато зі стану, коли в системі не було жодної вимоги, тобто $W_1 = 0$. Проаналізуємо часовий ряд на виході моделі (рис. 7.6).

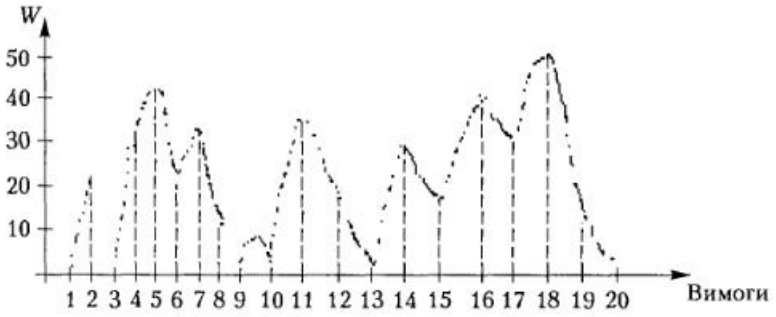


Рис. 7.6. Часовий ряд на виході моделі

Час чекання для вимог 1, 3, 9, 10, 13, 20 дорівнює нулю, тобто $W_k = 0$, $k = 1, 3, 9, 10, 13, 20$.

Назвемо *циклом* проміжок часу, який складається з періоду зайнятості й наступного періоду простою пристрою. На рис. 7.5 показано п'ять циклів, в які система начебто відновлюється (регенерує). Позначимо через Y_k суму значень тривалості чекання на k-му циклі, а через α_k – число вимог у k-му циклі. Тоді пари (y_1,α_1), (y_2,α_2), ..., (y_5,α_5) будуть незалежними випадковими величинами, що мають однаковий розподіл. У нашому випадку маємо такі значення для цих пар:

$$(y_1,\alpha_1) = (10;2); \quad (y_2,\alpha_2) = (130;6); \quad (y_3,\alpha_3) = (0;1);$$
$$(y_4,\alpha_4) = (50;3); \quad (y_5,\alpha_5) = (180;7).$$

Зрозуміло, що найкращою оцінкою за всіх умов буде середнє арифметичне, тобто

$$\frac{W_1 + \ldots + W_N}{N} = \frac{Y_1 + Y_2 + \ldots + Y_n}{\alpha_1 + \alpha_2 + \ldots + \alpha_n} = \frac{Y_1 + Y_2 + \ldots + Y_n/n}{\alpha_1 + \alpha_2 + \ldots + \alpha_n/n},$$

де N – загальна кількість вимог, n – кількість циклів.

За законом великих чисел, чисельник з імовірністю 1 за умови $n \to \infty$ прямує до значення математичного сподівання будь-якої величини, наприклад $M\{Y_1\}$, а знаменник – до $M\{\alpha_1\}$. Якщо $N \geq n$ і $N \to \infty$, то і $n \to \infty$. Тоді ліва частина виразу буде збігатись до математичного сподівання часу перебування вимоги в черзі, а права – до значення $M\{Y_1\}/M\{\alpha_1\}$, тобто

$$M\{W\} = M\{Y_1\}/M\{\alpha_1\}.$$

З наведеного вище прикладу видно, яким чином потрібно розділяти часову послідовність на статистично незалежні підпослідовності. Перевага такого підходу полягає в тому, що немає необхідності визначати момент закінчення перехідного процесу. Усі отримані під час моделювання дані є інформативними. Вже не виникає питання й про момент закінчення прогону моделі, тобто в цьому випадку він має співпадати з кінцем останнього циклу в моделі.

Тепер дамо визначення регенеративного процесу [28].

Послідовність $\{\bar{X}_n, n \geq 1\}$ випадкових векторів розмірністю k є *регенеративним процесом*, якщо існує така зростаюча послідовність випадкових дискретних моментів часу $1 \leq \beta_1 < \beta_2 < \ldots$, що називаються моментами регенерації, за якої протікання процесу, починаючи з кожного з цих моментів часу, визначається тими ж імовірнісними характеристиками, що і в момент β_1.

Це означає, що частини процесу $\{\bar{X}_n, \beta_j \leq n < \beta_{j+1}\}$ між будь-якими двома послідовними моментами регенерації β_j, β_{j+1} є ідентичними. Проте для частини процесу $1 \leq \beta_j$, незалежної від інших, допускається розходження за розподілом. Частина процесу $\{\bar{X}_n, \beta_j \leq n < \beta_{j+1}\}$ буде називатись j-м циклом регенерації. Взагалі клас регенеративних моделей досить широкий – СМО, мережі СМО, моделі керування запасами.

Позначимо через $\alpha_j = \beta_{j+1} - \beta_j$ тривалість j-го циклу регенерації і припустимо, що тривалості періодів регенерації $\{\alpha_j, j \geq 1\}$ – незалежні й однаково розподілені випадкові величини. Вважаємо, що $M\{\alpha_j\} < \infty$. Під час моделювання це припущення не є сильним обмеженням. Воно виконується для будь-якої СМО, а також для будь-якого позитивного поворотного незвідного ланцюга Маркова[1].

Будь-який регенеративний процес має в деякому сенсі стаціонарний розподіл, тобто існує такий k-вимірний випадковий вектор $\bar{X}$, для якого розподіл величини $\bar{X}_n$ збігається з розподілом $\bar{X}$, якщо $n \to \infty$:

$$\lim P\{\bar{X}_n \leq \bar{X}\} = P\{\bar{X} \leq \bar{x}\},$$

або $\bar{X}_n \to \bar{X}$, якщо $n \to \infty$.

---

[1] Ланцюг Маркова незвідний, якщо будь-який стан може бути досягнуто з будь-якого іншого. Ланцюг Маркова поворотний, якщо можливість повернутися в той же стан після відвідування його, дорівнює 1.

Нехай f – деяка вимірювана функція, яка залежить від K аргументів і приймає тільки дійсні значення (у прикладі, що розглядався вище, $f = M\{W\}$). Припустимо, що головна мета моделювання полягає в оцінюванні значення $r = M\{f(\vec{X})\}$.

Розглянемо результати застосування методів регенерації, які будемо використовувати під час побудови довірчого інтервалу I для r.

Нехай

$$Y_j = \sum_{i=\beta_j}^{\beta_{j+1}} f(\vec{X}_i).$$

Аналогічно: $\alpha_j = (\beta_{j+1} - \beta)$ – довжина j-го циклу регенерації. Тоді основну властивість регенеративного процесу можна сформулювати так:

Послідовність

$$\{(Y_j, \alpha_j), j \geq 1\} \tag{7.1}$$

складається з незалежних і однаково розподілених випадкових векторів. Якщо математичне сподівання $r = M\{|f(\vec{X})|\} < \infty$, то

$$r \equiv M\{f(\vec{X})\} = \frac{M\{Y_1\}}{M\{\alpha_1\}}. \tag{7.2}$$

Вираз (7.1) справедливий, тому що вектори Y_j,$\{(Y_j, \alpha_j), j \geq 1\}$ характеризують послідовні цикли регенерації, а ці цикли незалежні й мають однакові статистичні властивості. Стосовно виразу (7.2) відомо, що за загальних умов з імовірністю 1, якщо $N \to \infty$,

$$\frac{f(\vec{X}_1) + f(\vec{X}_2) + \ldots + f(\vec{X}_N)}{N} \to M\{f(X)\}. \tag{7.3}$$

Припустимо, що $\beta_1 = 1$, тобто цикли регенерації починаються з цього моменту моделювання. Якщо n-й цикл закінчується в момент N, то вираз (7.3) можна записати як

$$\frac{(Y_1 + Y_2 + \ldots + Y_n)/n}{(\alpha_1 + \alpha_2 + \ldots + \alpha_n)/n}, \tag{7.4}$$

враховуючи те, що Y_j – це сума $f(\vec{X}_i)$ за j-й цикл, а α_j – довжина даного циклу. Але відповідно до закону великих чисел відношення (7.4) з імовірністю 1 збігається з $M\{Y_1\}/M\{\alpha_2\}$ за умови, що кількість циклів прямує до нескінченності ($n \to \infty$).

У випадку, коли $n \to \infty$, якщо $N \to \infty$, одержуємо основне співвідношення, визначене у виразі (7.2).

Якщо перший цикл регенерації починається не відразу, то слід урахувати уявний перехідний період Y_0:

$$\frac{(Y_0 + Y_1 + \ldots + Y_n)/n}{(\alpha_0 + \alpha_1 + \ldots + \alpha_n)/n}, \tag{7.5}$$

де Y_0, α_0 відповідають перехідному періоду і Y_0 розраховується як

$$Y_0 = f(\vec{X}_0) + \ldots + f(\vec{X}_{\beta_1 - 1}).$$

У загальному випадку функція розподілу випадкової величини α_0 може відрізнятись від функції розподілу α_j, так само як функція розподілу величини Y_0 відрізняється від функції розподілу Y_j. Проте за наявності великого числа циклів n значення Y_0 буде малим порівняно із сумою інших значень Y_j, і α_0 – порівняно із сумою α_j, і відношення (7.5) усе ж буде наближатись до значення $M\{Y_1\}/M\{\alpha_1\}$.

Враховуючи формули (7.1) і (7.2), задачу статистичного оцінювання можна сформулювати таким чином: за наявності незалежних і однаково розподілених спостережень $\{(Y_j,\alpha_j), j \geq 1\}$ необхідно обчислити значення $r = M\{Y_1\}/M\{\alpha_1\}$. Зважаючи на незалежність і рівність розподілів величин для визначення оцінки $\hat{r}$, можна скористатись класичними методами математичної статистики.

Тепер сформулюємо задачу побудови довірчого інтервалу: за наявності незалежних однаково розподілених пар значень (Y_1, α_1), (Y_2, α_2), ..., (Y_n, α_n) необхідно побудувати 100 $(1 - \delta)$ %-й довірчий інтервал для $M\{Y_1\}/M\{\alpha_1\}$, коли значення n велике.

Скористаємось центральною граничною теоремою. Введемо позначення

$$V_j = Y_j - r\alpha_j,$$

де V_j – незалежна й рівномірно розподілена випадкова величина.

З урахуванням виразів (7.1) і (7.2) маємо $M(V_j) = M(Y_j) - r\,M(\alpha_j) = 0$.

Нехай величини $\overline{Y}$, $\overline{\alpha}$, $\overline{V}$ позначають вибіркові середні:

$$\overline{Y} = \frac{1}{n}\sum_{j=1}^{n} Y_j, \quad \overline{\alpha} = \frac{1}{n}\sum_{j=1}^{n} \alpha_j, \quad \overline{V} = \frac{1}{n}\sum_{j=1}^{n} V_j,$$

а

$$\overline{V} = \overline{Y} - r\overline{\alpha}.$$

Дисперсію випадкової величини V_j можна визначити як $\sigma^2 = M\{V_j^2\}$. Припускаючи, що $0 < \sigma^2 < \infty$, згідно з центральною граничною теоремою для кожного дійсного x маємо

$$\lim_{n\to\infty} P\left\{\frac{\overline{V}\sqrt{n}}{\sigma} \leq x\right\} = \phi(x), \tag{7.6}$$

де $\phi(x)$ – стандартизована нормальна функція. Для більшості задач моделювання припущення $0 < \sigma^2 < \infty$ не є строгим обмеженням. З виразу (7.6) маємо, що

$$\lim_{n\to\infty} P\left\{\frac{(\overline{Y} - r\alpha)\sqrt{n}}{\sigma} \leq x\right\} = \phi(x),$$

або

$$\lim_{n\to\infty} P\left\{\frac{(\hat{r} - r)\sqrt{n}}{\sigma/\overline{\alpha}} \leq x\right\} = \phi(x), \tag{7.7}$$

де $\hat{r} = \overline{Y}/\overline{\alpha}$. Безпосередньо з (7.7) обчислити довжину довірчого інтервалу для r неможливо, тому що середньоквадратичне відхилення σ невідоме, але його можна оцінити як

$$\hat{\sigma}^2 = M\{(Y_1 - r\alpha_1)^2\} = D\{Y_1\} - 2\hat{r}\,\mathrm{cov}\{Y_1, \alpha_1\} + \hat{r}^2 D\{\alpha_1\}.$$

Нехай S_{11}, S_{22}, S_{12} – відповідно вибіркові дисперсії значень Y_j, α_j, а (Y_j, α_j) – вибірковий змішаний момент, тобто

$$S_{11} = \frac{1}{n-1}\sum_{j=1}^{n}(Y_j - \overline{Y})^2 = \frac{1}{n-1}\sum_{j=1}^{n}Y_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n}Y_j\right)^2,$$

$$S_{22} = \frac{1}{n-1}\sum_{j=1}^{n}(\alpha_j - \overline{\alpha})^2 = \frac{1}{n-1}\sum_{j=1}^{n}\alpha_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n}\alpha_j\right)^2,$$

$$S_{12} = \frac{1}{n-1}\sum_{j=1}^{n}(Y_j - \overline{Y})(\alpha_j - \overline{\alpha}) = \frac{1}{n-1}\sum_{j=1}^{n}Y_j\alpha_j - \frac{1}{n(n-1)}\sum_{j=1}^{n}Y_j\sum_{j=1}^{n}\alpha_j.$$

Тоді $S^2 = S_{11} - 2\hat{r}S_{12} + \hat{r}^2S_{22}$.

Легко показати, що з імовірністю 1 значення $S^2 \to \sigma^2$, якщо $n \to \infty$.

Замінивши у виразі (7.7) σ на S, отримаємо:

$$\lim_{n\to\infty} P\left\{\frac{(\hat{r}-r)\sqrt{n}}{S/\overline{\alpha}} \le x\right\} = \phi(x). \tag{7.8}$$

Будемо вважати, що Z_σ^* – це обернена функція від $\phi^{-1}(1-\sigma/2)$, тоді

$$\phi(Z_\sigma^*) = 1 - \frac{\sigma}{2}.$$

З виразу (7.8) для великих значень n маємо

$$P\left\{-Z_\sigma^* \le \frac{(\hat{r}-r)\sqrt{n}}{S/\overline{\alpha}} \le Z_\sigma^*\right\} \cong 1-\sigma.$$

Тоді можна записати довірчий інтервал як

$$\hat{I} = \left[\hat{r} - \frac{Z_\sigma^* S}{\overline{\alpha}\sqrt{n}};\ \hat{r} + \frac{Z_\sigma^* S}{\overline{\alpha}\sqrt{n}}\right].$$

Звідси довжину довірчого інтервалу для оцінки $\hat{I}$ за великих значень n можна отримати у вигляді

$$\hat{J} \cong 2\frac{Z_\sigma^* S}{\overline{\alpha}\sqrt{n}}.$$

Перш ніж побудувати алгоритм для оцінювання довірчого інтервалу, потрібно мати спосіб визначення циклів регенерації в імітаційній моделі. Для розімкнутих мереж СМО початок циклу регенерації можна визначити, порівнюючи кількість вимог, що залишили систему, і кількість вимог, що надійшли до системи. Якщо процес моделювання починався зі стану, коли в мережі СМО не було жодної вимоги, то моментом регенерації можна вважати стан, коли в мережі знову не буде жодної вимоги. Цей момент і буде початком циклу регенерації. Різниця між кількостями вимог на вході і виході мережі буде дорівнювати нулю (у загальному випадку вона може дорівнювати будь-якому числу).

Нехай у замкнутій мережі СМО в момент початку моделювання перебуває k транзактів, що займають M пристроїв ($k > M$) (див. рис. 2.6). Початком циклу регенерації є будь-який момент часу, коли в мережі знову буде зайнято M цих же пристроїв.

Для систем керування запасами початок циклу регенерації можна визначити, наприклад, за допомогою моменту досягання рівня запасів, при якому приймається рішення про їх поповнення.

Тепер опишемо загальний алгоритм побудови довірчого інтервалу:

1. Провести моделювання системи протягом n циклів регенерації.
2. Для кожного j-го циклу регенерації обчислити значення (Y_j, α_j).
3. На основі результатів моделювання обчислити вибіркові статистики $\overline{Y}, \overline{\alpha}, \hat{r}, S_{11}, S_{22}, S_{12}, S^2$.
4. Довжину довірчого інтервалу обчислити за формулою

$$\hat{r} \pm \frac{Z_\sigma^* S}{\overline{\alpha}\sqrt{n}},$$

де Z_σ^* визначається за таблицею для стандартизованого нормального розподілу; наприклад, для рівня $\alpha = 0{,}9$ значення $Z_\sigma^* = 1{,}645$.

Вище було зазначено, що коли перший цикл не починається разом із початком моделювання, отримані дані, які відносяться до першого циклу, необхідно відкинути.

Як правило, моделювання закінчується після досягнення заданої кількості циклів регенерації, наприклад 1000. Зазначимо, що в цьому випадку границі довірчого інтервалу буде розраховано саме за цими 1000 циклами. Якщо довжина отриманого довірчого інтервалу для точного оцінювання недостатня, необхідно розрахувати кількість циклів з огляду на довжину отриманого інтервалу (залежність довжини довірчого інтервалу від кількості циклів – квадратична). Такий підхід зазвичай називають пробною вибіркою, за результатами якої виконують остаточні розрахунки. Реалізація цього методу мовою GPSS наведена в книзі [61].

Якщо вважати, що метою є визначення моменту закінчення моделювання, коли довжина довірчого інтервалу досягне заданого значення, то бажано мати метод, за допомогою якого можна було б визначити кінець моделювання за умови, що довжина довірчого інтервалу досягла заданого значення. Враховуючи те, що довжина інтервалу під час моделювання з вибіркою фіксованого обсягу у загальному випадку буде випадковою, обсяг вибірки, потрібний для отримання довірчого інтервалу фіксованої довжини, також буде випадковим. У цьому разі критерій зупинення моделювання називають випадковим правилом зупинення.

Більш доцільним є підхід, який дає змогу закінчувати процес моделювання, коли довжина інтервалу залежить від самого показника, що оцінюється. Цей критерій можна назвати критерієм зупинення моделювання за відносною довжиною довірчого інтервалу. У цьому випадку можна розробити процедуру автоматичного зупинення моделювання.

Для того щоб автоматично зупиняти процес моделювання, потрібно задати ширину довірчого інтервалу відносно середнього значення оцінюваної величини.

За умови, що ширина довірчого інтервалу не перевищує ± 5 % середнього значення, моделювання із заданою довірчою ймовірністю необхідно закінчити тоді, коли довірчий інтервал не буде перевищувати ± 5 % оцінюваного розміру. Для того щоб переконатися, що таке влучення не випадкове, бажано провести ще один цикл регенерації та упевнитися, що в цьому разі довірчий інтервал не збільшився. Такий підхід передбачає обчислення довірчого інтервалу після виконання кожного циклу регенерації та зменшує швидкість роботи моделі.

7.3. Методи зниження дисперсії

Виконання імітаційних експериментів з моделями складних виробничих систем потребує багато часу і ресурсів комп'ютера, тому дослідник повинен використовувати різні засоби і можливості для підвищення ефективності моделювання. Ефективність моделювання передбачає не тільки економне програмування моделі і скорочення часу моделювання, а й зменшення витрат на статистичний аналіз. Критерієм ефективності статистичного аналізу може слугувати дисперсія оцінок вихідних даних моделі.

Методи зниження дисперсії (МЗД) були розроблені ще на початкових етапах використання комп'ютерних технологій, коли вони застосовувались під час моделювання систем методом Монте-Карло. Виділяють дві групи МЗД – методи, що використовуються після збирання вибіркових даних, і методи, що застосовуються безпосередньо під час моделювання і впливають певним чином на саму вибіркову процедуру [21].

Статистична надійність оцінювання відгуку модельованих систем залежить від обсягу вибірки, який можна визначити на основі оцінки стандартного відхилення. Припустимо, що середнє значення відгуку μ визначається за результатами моделювання, під час якого отримано n незалежних спостережень за середніми значеннями $x_1, x_2, ..., x_n$.

Відомо, що вибіркове стандартне відхилення величини $\bar{x}$ для n незалежних спостережень можна визначити як $\sigma/\sqrt{n-1}$, де σ – стандартне відхилення одиничних спостережень x_i, $i = \overline{1,n}$. Мірою надійності оцінки μ є довірчий інтервал

$$\bar{x} \pm a\frac{\sigma}{\sqrt{n-1}},$$

де a – деяка константа. Надійність оцінки $\bar{x}$ можна збільшити (тобто зменшити довірчий інтервал і вплив випадкових відхилень), якщо аналізувати більшу за обсягом вибірку. Використання МЗД дає змогу при заданому обсязі вибірки збільшити точність оцінювання відгуку, або при заданій точності скоротити обсяг вибірки. У результаті матиме місце оцінка з тим самим математичним сподіванням, але з меншою дисперсією.

Число спостережень n, необхідне для досягнення заданої надійності, запишемо як функцію дисперсії

$$n = b\sigma^2, \quad b = const.$$

Отже, під час застосування МЗД для фіксованого обсягу вибірки доцільно визначати зменшення стандартного відхилення, яке пропорційне довжині довірчого інтервалу. Якщо ж припустити, що бажану надійність оцінки задано заздалегідь, то слід використовувати дисперсію, тому що її зменшення пропорційне числу спостережень, необхідних для заданої надійності. Застосовуючи МЗД, можна зменшити дисперсію шуканих оцінок, але немає сенсу домагатися зменшення дисперсії, якщо необхідні для цього зусилля надто великі.

Організація проведення експериментів з імітаційними моделями передбачає застосування МЗД, які вбудовуються в програму моделювання і дають змогу керувати процесом збирання вибіркових значень для відгуків. До таких методів належать методи доповнювальних величин [21], загальних випадкових чисел та російської рулетки [2]. Для моделювання рідких подій [24] використовують такі МЗД, як значуща вибірка [21, 67]. Цей метод передбачає заміну початкового розподілу вхідних змінних новим розподілом. Значенням вхідних змінних, при яких отримують найбільш важливі значення відгуку, присвоюється більша ймовірність, а потім коригується перетворений відгук.

Методи МЗД зазвичай застосовуються для оцінювання середнього арифметичного і математичного сподівання. Вважається, що це і є метою імітаційного моделювання. Дуже часто метою може бути оцінювання ймовірності того, що відгук не перевищить деяке фіксоване значення, тобто процентіль. Таке оцінювання ймовірності можна звести до оцінювання середнього, якщо ввести нову змінну Y з математичним сподіванням p, де p – ймовірність того, що шукана змінна X не перевищить деяке фіксоване значення.

7.3.1. Метод доповнювальних величин

Метод доповнювальних величин застосовується для створення негативної кореляції між двома дослідами. Цей метод використовується, щоб знизити дисперсію тільки під час імітаційного моделювання однієї системи. Спочатку на основі послідовності R генерують потік випадкових чисел для одного випробування, а на основі додаткових величин $(1 - R)$ – для іншого. Потім провадять два прогони моделі. Під час першого використовують послідовність випадкових чисел $r_1, r_2, ..., r_n$, а під час другого – послідовність $(1 - r_1), (1 - r_2), ..., (1 - r_n)$.

За допомогою цієї процедури вводиться негативний кореляційний зв'язок між дослідами, що приводить до зниження дисперсії оцінки відгуку. Якщо метою моделювання є визначення середнього арифметичного відгуку системи μ, то оцінку можна записати як

$$\bar{x} = \frac{1}{2}(y_1 + y_2),$$

де y_1 та y_2 – оцінки відгуку, що обчислюються за результатами першого та другого прогонів моделі. Тоді дисперсія середнього обчислюється за формулою

$$D(\bar{x}) = \frac{1}{4}\left(D(y_1) + D(y_2) + 2\,\mathrm{cov}\,(y_1, y_2)\right),$$

тобто дисперсія зменшиться на величину $0{,}5\,\mathrm{cov}\,(y_1, y_2)$.

Покажемо, що послідовності рівномірно розподілених випадкових чисел R і $(1 - R)$ зумовлюють негативну кореляцію між дослідами. Припустимо, що відгук y залежить лише від однієї випадкової змінної x (на практиці, як правило, він залежить від послідовності випадкових змінних), і що y — монотонна функція від x, тобто $g_1(x_1) > g_1(x_2)$, якщо $x_1 > x_2$. Відомо, що x генерується за допомогою випадкових чисел R, наприклад методом оберненої функції, тобто $g_2(r)$. Тоді, звичайно, виконується умова $g_1(x_1) > g_1(x_2)$, $r_1 > r_2$. Нехай x — монотонна функція $g_3(r)$. Таким чином, більшому значенню r відповідає більше значення y. Але більше значення r призводить до зменшення величини $(1 - r)$, яка є доповненням до одиниці, що у свою чергу призводить до зменшення значень y. Таким чином, більшим значенням $y = g_3(r)$ (більшим, ніж математичне сподівання μ), відповідають менші значення (менші, ніж μ) відгуку доповнювальної величини $y_g = g_3(1 - r)$. Це означає, що між y і y_g існує негативний кореляційний зв'язок.

Обидва значення відгуку є реалізаціями однієї випадкової величини, тому що їх згенеровано за допомогою випадкових чисел r_i, рівномірно розподілених в інтервалі $[0,1]$.

Дійсно,

$$F(y) = \int_{-\infty}^{+\infty} g_3(r) f_r(r)\, dr = \int_0^1 g_3(r)\, dr.$$

Для доповнювальної величини $z = (1 - r)$ отримаємо

$$F(y) = \int_{-\infty}^{+\infty} g_3(1-r)\, f_r(1-r)\, d(1-r) = \int_0^1 g_3(z)\, dz,$$

де z має ту ж функцію розподілу, що і r.

Для немонотонних залежностей, зображених на рис. 7.7, знайдемо випадкові доповнювальні величини як

$$R_2 = C - R, \quad 0 \le r \le C.$$
$$R_2 = 1 + C - R, \quad C < r \le 1.$$

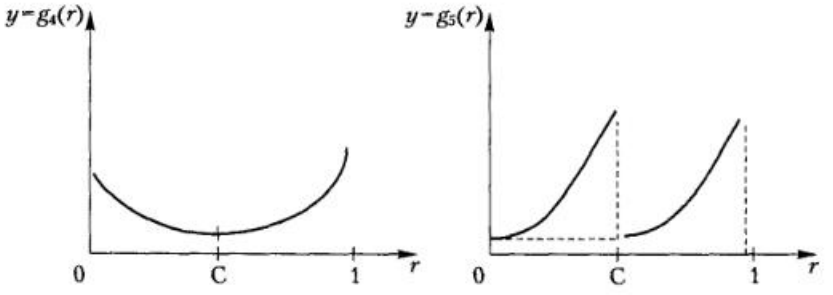


Рис. 7.7. Функції відгуків для різних вхідних значень

Можна показати, що випадкові числа R_2 зумовлюють негативну кореляцію $y = g_4(r)$ та $y = g_5(r)$.

7.3.2. «Російська рулетка» і розбивання вибірки

Метод російської рулетки розглянемо на прикладі моделі телефонної мережі, яка складається з п'яти вузлів [2]. Спочатку сформулюємо задачу моделювання: в деякий випадковий момент часу необхідно знайти імовірність з'єднання першого і п'ятого абонентів за умови, що існує деяка відмінна від нуля ймовірність того, що будь-яка з ліній, зображених на рис. 7.8, буде зайнятою. Ймовірності p_{ij} (i = = 1,2, ..., 5; j = 1,2, ..., 5) того, що лінії вільні, відомі.

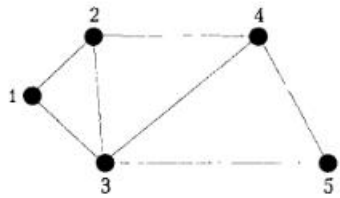


Рис. 7.8. *Схема з'єднань телефонної мережі*

Скористаємося методом статичних випробувань і розглянемо 100 варіантів з'єднання між абонентами 1 і 5. Для моделювання необхідні 700 випадкових чисел. Під час моделювання слід враховувати особливості мережі передачі. А саме, якщо лінії 1–2 і 1–3 зайняті (аналогічно для ліній 4–5, 3–5), то зв'язок між цими абонентами неможливий, тому немає сенсу розігрувати імовірності для інших ліній, тобто дане випробування можна завершити. Отже, якщо у таких випадках припинити випробування, то середня кількість випадкових чисел, необхідних для проведення одного випробування, зменшиться.

З іншого боку, використовуючи таку ж кількість випадкових чисел, можна збільшити число випробувань. У цьому разі дисперсія оцінки ймовірності зменшується. Цей метод припинення випробувань називається «російською рулеткою».

7.3.3. Загальні випадкові числа

На практиці імітаційне моделювання найчастіше використовується для порівняння характеристик кількох систем або варіантів однієї системи. Зрозуміло, що у процесі моделювання потрібно порівнювати моделі систем, які функціонують в однакових умовах. Тому моделювання бажано проводити за однакових початкових умов для моделі і однакових вхідних змінних, а також якщо є можливість використовувати одні й ті ж послідовності випадкових чисел. Застосування спільних випадкових чисел зумовлює кореляційний зв'язок між відгуками. Відомо, що дисперсія різниці між оцінками відгуку X і Y відповідно першої і другої систем обчислюється за формулою

$$D(X - Y) = D(X) + D(Y) - 2\,\mathrm{cov}\,(X, Y).$$

Отже, дисперсія оцінки зменшиться, якщо між послідовностями випадкових чисел існує позитивний кореляційний зв'язок. Це можливо, якщо реакція обох систем на вхідні змінні є однаковою. Наприклад, дві СМО відрізняються кількістю пристроїв, правилами організації черг, часом обслуговування, але у разі

збільшення інтенсивності надходження вимог і тривалості часу обслуговування в обох системах збільшується час перебування вимог у черзі.

У разі застосування описуваного методу необхідно, щоб кожна випадкова вхідна змінна обчислювалась на основі власної послідовності випадкових чисел. Якщо під час моделювання використовується мова GPSS, то для всіх генераторів випадкових чисел потрібно встановлювати однакові початкові значення за допомогою команди RMULT.

Даний метод є одним із найпоширеніших методів зниження дисперсії. Причина його успіху полягає у тому, що більшість програмних засобів імітаційного моделювання використовують одні й ті самі потоки випадкових чисел.

7.4. Факторний план

Методи планування експериментів з моделями, що розглядаються далі, особливо ефективні на ранніх стадіях розробки моделей, коли ще не повністю ясно, які змінні і параметри мають найбільший вплив на вихідні характеристики системи.

Як правило, для кожної моделі зазначають вхідні і вихідні змінні. Вхідні змінні називаються *факторами*, вихідні – *відгуками*. Під час проведення експерименту кожний фактор може приймати одне або кілька значень, які називаються *рівнями фактора*. За допомогою множини факторів визначають один із можливих станів модельованої системи і задають умови проведення одного з можливих експериментів. Існує певний зв'язок між рівнями факторів і відгуками системи, характер якого заздалегідь невідомий. Цей зв'язок можна визначити як

$$y_l = \psi_l(x_1, x_2, \ldots, x_i, \ldots, x_m), \quad l = \overline{1,n},$$

де y_l – l-й відгук; n– число відгуків, що аналізуються; x_i – i-й фактор; m– число факторів.

Функція ψ називається *функцією відгуку* або реакцією. Її геометричний образ – поверхня відгуку. Враховуючи те, що функція ψ заздалегідь невідома, використовують іншу наближену функцію

$$\hat{y}_l = \varphi_l(x_1, x_2, \ldots, x_i, \ldots, x_m), \quad l = \overline{1,n}.$$

Функцію φ_l визначають за результатами проведення експериментів і записують у вигляді ступеневого полінома першого, другого або, рідше, третього порядку (зазвичай перші члени ряду Тейлора). Після проведення експериментів апроксимуючі поліноми заміняють рівняннями регресії і за допомогою методу найменших квадратів (МНК) обчислюють статистичні оцінки їх невідомих коефіцієнтів.

Для того щоб побудувати функцію φ_i за результатами проведення експериментів, необхідно визначити коефіцієнти рівняння регресії. Завдання визначення коефіцієнтів регресії є типовим для регресійного аналізу. Попередньо потрібно забезпечити такі умови.

1. Результати спостережень $y_1, y_2, \ldots, y_m$ у m точках факторного простору мають бути реалізаціями нормально розподіленої випадкової величини.

2. Дисперсії реалізацій $D(y_i)$ $(i = 1, 2, ..., m)$ мають бути рівними між собою, тобто дисперсії величин не залежать від їх абсолютних значень.
3. Фактори $X_1, X_2, ..., X_N$ мають бути незалежними величинами, які вимірюються з настільки малою помилкою, що нею можна знехтувати (на відміну від помилки визначення величини y).

Третя умова виконується завжди, якщо регресійний аналіз використовується в імітаційному моделюванні, а інші умови потрібно перевіряти за допомогою спеціальних тестів.

Вирізняють фактори *керовані* (їх рівні цілеспрямовано вибираються дослідником) і такі, що *спостерігаються* (їх значення тільки спостерігаються і реєструються). Фактор може бути кількісним або якісним. Значенням кількісного фактора є число або числовий вираз.

Фактори можуть бути фіксованими (детермінованими) і випадковими. Значення випадкового фактора вибирається із сукупності можливих значень. Під час планування експериментів одночасно змінюються кілька факторів, які мають бути сумісними і незалежними. Сумісність визначає те, що всі комбінації факторів можна здійснити, а незалежність – це можливість установлення факторів на будь-якому рівні, незалежно один від одного.

У результаті проведення експерименту фактори можуть відсіюватись, тобто з усієї множини факторів визначаються ті, які істотно впливають на відгуки моделі. На відгук моделі зазвичай впливає тільки частина вхідних факторів.

Ще один вид факторного експерименту використовується для визначення екстремальних значень на поверхні відгуку. В цьому випадку проведення серії експериментів планується таким чином, щоб досягти екстремуму на поверхні відгуку.

Дамо визначення факторного експерименту.

Факторний експеримент – це план, згідно з яким всі рівні кожного фактора зустрічаються в сполученні з усіма рівнями всіх інших факторів. Рівні визначають кількісні значення факторів. Різні рівні деякого фактора можуть відповідати якісним значенням (наприклад, різні дисципліни обслуговування вимог пристроєм) або кількісним значенням (наприклад, число пристроїв для обслуговування). Якщо деякий фактор f, $(f = 1, ..., k)$ має L_f рівнів, то загальне число комбінацій рівнів визначається добутком:

$$L_1, ..., L_k = \prod_{f=1}^{k} L_f. \tag{7.9}$$

Якщо число рівнів для кожного з факторів однакове, то загальним числом комбінацій буде L^k.

Ліва частина виразу (7.9) використовується для позначення факторного плану.

Застосування факторного плану замість класичної схеми, відповідно до якої щораз змінюється тільки один фактор, має такі переваги.

1. Стає більш повною картина впливу кожного з факторів, оскільки вони аналізуються у різних умовах.

2. Велика кількість комбінацій факторів, що використовуються під час проведення експерименту, полегшує прогнозування результатів, які можна отримати за наявності певної комбінації умов.
3. Якщо ефекти, зумовлені дією окремих факторів, статистично незалежні, то про кожний фактор можна отримати не менше інформації, ніж у тому випадку, коли під час проведення експериментів змінюється лише один фактор, а інші є фіксованими.
4. Якщо різні фактори не є незалежними (як це часто буває), а зумовлюють ефекти взаємодії (кореляцію), то тільки за допомогою факторного експерименту можна отримати інформацію щодо характеру цих взаємодій. За наявності кількох взаємозалежних істотних факторів неможливо обійтись без проведення факторного експерименту. Для низки задач, які часто зустрічаються, спеціально розроблено велику кількість стандартних факторних планів.

Якщо у моделі є тільки один фактор, планування експерименту буде досить простим: необхідно виконати моделювання для різних рівнів фактора і побудувати довірчий інтервал на кожному з його рівнів.

У разі моделювання різних систем аналізуються як мінімум два фактори. Розглянемо двофакторний експеримент із двома факторами на двох рівнях і двома спостереженнями в кожному досліді, тобто план 2^2. Фактори прийнято позначати буквами латинського алфавіту A, B, C, ...

Результати проведення експериментів зведено в табл. 7.1.

Таблиця 7.1. Матриця повного факторного експерименту 2^2

Фактор *A*	**Фактор *B***	
	Рівень 1	**Рівень 2**
Рівень 1	y_{111}	y_{121}
	y_{112}	y_{122}
Рівень 2	y_{211}	y_{221}
	y_{212}	y_{222}

У цій таблиці елемент y_{ijg} позначає g-ті спостереження ($g = 1, 2$); комбінація рівнів i та j факторів A та B. Кількість спостережень (прогонів моделі) g визначається згідно з необхідною точністю оцінювання відгуків моделі. У загальному випадку в разі проведення двофакторного експерименту число рівнів факторів A і B відповідно може бути різним. Позначимо математичне сподівання як $M(y_{ijg}) = \eta_{ij}$, тоді під час планування проведення експерименту припустимо, що правильною буде така модель:

$$y_{ijg} = \eta_{ij} + e_{ijg}, \quad i = \overline{1, I}; \quad j = \overline{1, J}; \quad g = 1, 2, 3, ...,$$

де e_{ijg} – похибка досліду. Вважається, що ці помилки є незалежними нормально розподіленими випадковими величинами з математичним сподіванням 0 і дисперсією σ^2. Під час моделювання помилки дослідів можна зробити незалежними, якщо для прогонів моделі застосовувати різні послідовності випадкових чисел.

7.5. Дисперсійний аналіз ANOVA

Для того щоб визначити, чи є фактор *значущим*, використовується дисперсійний аналіз ANOVA (analysis of variance), який можна застосовувати тільки для кількісних факторів. За допомогою цього аналізу визначаються кількісні відхилення спостережень від середнього значення. Фактор, який не впливає на відгук, є *незначущим*. З іншого боку, якщо фактор впливає на відгук, то кількісне значення першого порівнюють з оцінкою мінливості спостереження, тобто стандартною помилкою. Цю операцію застосовують для виключення ефектів, що є не чим іншим як випадковими флуктуаціями.

Неявно в ANOVA використовується адитивна математична модель, яка дає змогу визначити компоненти змін під час спостережень. Її називають статистичною моделлю. Найпростіша статистична модель має такий вигляд:

$$y_{ig} = \mu + e_{ig},$$

тобто кожне i-те спостереження є сумою загального середнього, яке обчислюється за результатами спостережень у всіх дослідах μ, і випадкової похибки e_{ig}. У цій моделі загальне середнє не змінюється від досліду до досліду, на відміну від похибки.

Статистична модель для аналізу даних експериментів з одним фактором A має такий вигляд:

$$y_{ig} = \mu + \alpha_i^A + e_{ig},$$

де α_i^A – головний ефект фактора A на рівні i. Усі спостереження на даному рівні аналізуються з використанням тих же самих значень α_i^A. Зважаючи на те, що в цьому експерименті є тільки один фактор, число комбінацій визначається кількістю рівнів цього фактора.

Для двох факторів загальна модель факторного плану така:

$$y_{ijg} = \mu + \alpha_i^A + \alpha_j^B + \alpha_{ij}^{AB} + e_{ijg}, \tag{7.10}$$

де α_j^B – головний ефект фактора B на рівні j; α_{ij}^{AB} – ефект, що визначає взаємодію фактора A на рівні i і фактора B на рівні j. Сума ефектів двох факторів не дорівнює сумі їх окремих ефектів, тому що фактори взаємодіють між собою. *Головний ефект* фактора визначає частку впливу фактора на значення функції відгуку за рахунок зміни його значення.

Дисперсійний аналіз, заснований на статистичній моделі (7.10), закінчується побудовою таблиці ANOVA, в якій аналізується вплив факторів A, B, ефекту взаємодії між цими факторами і випадкових похибок спостережень.

За допомогою ANOVA перевіряється гіпотеза про відсутність впливу фактора. Якщо ця гіпотеза справедлива, то вважається, що всі спостереження отримано з однієї генеральної сукупності. Для перевірки гіпотези використовується F-розподіл Фішера. Критерій Фішера визначає відношення двох вибіркових дисперсій. Якщо фактор суттєво впливає на відгук, то величина F-розподілу істотно зростає і F-статистика стає значущою. Таким чином, наявність великих значень F приводить до того, що гіпотеза про відсутність впливу фактора відкидається, тобто фактор є значущим.

7.6. Особливості планування експериментів

Опишемо послідовність дій, які необхідно виконувати під час планування експериментів.

1. Визначення відгуків (вихідних змінних) системи.
2. Визначення факторів, які впливають на відгук системи. Більшість систем підпорядковуються принципу Парето — з огляду на характеристики системи істотними є лише деякі з множини факторів. У більшості систем 20 % факторів визначають 80 % властивостей системи.
3. Визначення рівнів факторів. Мінімальна кількість рівнів для кожного фактора два — нижня і верхня межі значення фактора. У разі використання цього числа рівнів можна визначити тільки лінійні ефекти. Для врахування квадратичних ефектів необхідно використовувати три рівні, для кубічних ефектів — чотири і т. д. Аналіз значно спрощується, якщо брати тільки рівновіддалені одне від одного значення рівнів. У цьому випадку маємо так зване ортогональне планування, або ортогональний експеримент.

Для множинних експериментів з числом факторів більше одного дисперсійний аналіз передбачає використання для заключного аналізу ортогонального експерименту. Це означає, що оцінки відгуків у межах аналізу мають бути некорельованими. На практиці ортогональність гарантує використання тих самих випадкових послідовностей чисел під час виконання експериментів у межах кожної комбінації рівнів обробки.

7.7. Повний факторний експеримент

Експеримент, в якому реалізуються всі можливі сполучення рівнів факторів, називається *повним факторним експериментом*. Розглянемо простий двофакторний експеримент з одним фактором на двох рівнях, одним фактором на трьох рівнях і з двома спостереженнями в кожному досліді, тобто план 3×2. Запишемо в табл. 7.2 матрицю експерименту.

Таблиця 7.2. Матриця двофакторного експерименту

Фактор A	Фактор B	
	Рівень 1	Рівень 2
Рівень 1	y_{111}	y_{121}
	y_{112}	y_{122}
Рівень 2	y_{211}	y_{221}
	y_{212}	y_{222}
Рівень 3	y_{311}	y_{321}
	y_{312}	y_{322}

У загальному випадку: значення фактора y_{ijg}, де g — номер спостереження, i та j — номери рівнів факторів A та B відповідно. Нехай математичне сподівання

вихідної змінної $M(y_{ijg}) = \eta_{ij}$. Тоді очікувану функцію відгуку можна записати у такому вигляді:

$$y_{ijg} = \eta_{ij} + e_{ijg}, \quad i = \overline{1, I}; \quad j = \overline{1, J}; \quad g = 1, 2, 3, ..., \tag{7.11}$$

де e_{ijg} – похибка досліду (або шум), яка вважається незалежною нормально розподіленою випадковою величиною з математичним сподіванням нуль і дисперсією σ^2, або

$$e_{ijg} : \text{ННР}(0, \sigma^2). \tag{7.12}$$

Покажемо, що моделі для планування експериментів є окремими випадками моделей лінійної регресії [21]. Знайдемо середнє за всіма дослідами:

$$\mu = \frac{\sum_{i \in I} \sum_{j \in J} \eta_{ij}}{IJ} = \eta_{\bullet\bullet}, \tag{7.13}$$

де крапка означає усереднення по всіх значеннях відповідного індексу.

Якщо знайти середнє значення відгуку для фактора A на рівні i з усіма рівнями фактора B, то

$$A_i = \frac{\sum_{j \in J} \eta_{ij}}{J} = \eta_{i\bullet}. \tag{7.14}$$

Тоді α^A_i – головний ефект фактора A на рівні i визначається як різниця між його середнім і загальним середнім:

$$\alpha_i^A = A_i - \mu = \eta_j - \eta_\bullet. \tag{7.15}$$

З виразів (7.13)–(7.15) видно, що середнє головного ефекту дорівнює нулю, тому що

$$\sum_{i=1}^{I} \alpha_i^A = \frac{1}{J} \sum_i \sum_j \eta_{ij} - \sum_i \mu = I\mu - I\mu = 0. \tag{7.16}$$

Головний ефект фактора B на рівні j визначаємо як

$$\alpha_j^B = B_j - \mu = \frac{1}{I} \sum_i \eta_{ij} - \mu = \eta_\bullet - \eta_{\bullet\bullet}. \tag{7.17}$$

Аналогічно

$$\sum_{j=1}^{J} \alpha_j^\beta = 0. \tag{7.18}$$

Якщо припустити, що фактори не взаємодіють між собою, то одержимо таку модель для планування проведення експерименту:

$$M(y_{ijg}) = \eta_{ij} = \mu + \alpha_i^A + \alpha_j^B. \tag{7.19}$$

З виразу (7.19) маємо

$$\eta_{i1} - \eta_{i2} = \alpha_1^B - \alpha_2^B. \tag{7.20}$$

Вираз (7.20) є вірним для всіх рівнів i фактора A.

Відобразивши графічно, як фактор A впливає на рівень i фактора B, одержимо паралельні криві відгуку (рис. 7.9). Якщо є взаємодія між факторами A і B, то зміна фактора A викликає різноманітні зміни відгуку на різних рівнях фактора B. Таку взаємодію між рівнями i та j факторів A, B відповідно визначаємо як

$$\alpha_{ij}^{AB} = \eta_{ij} - A_i - B_j + \mu = \eta_{ij} - \eta_{i\bullet} - \eta_{\bullet j} + \eta_{\bullet\bullet}. \tag{7.21}$$

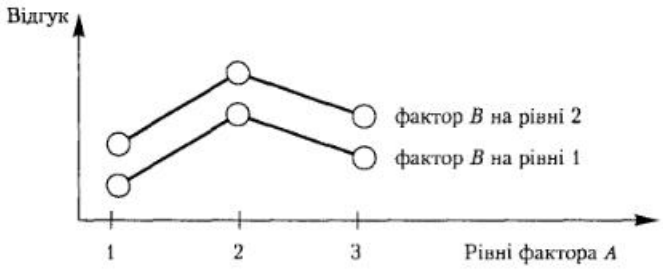


Рис. 7.9. Графік впливів факторів

Аналогічно, як було у виразах (7.16) і (7.18), маємо:

$$\alpha_{\bullet j}^{AB} = \alpha_{i\bullet}^{AB}.$$

Тоді загальна модель з урахуванням взаємодії двох факторів буде такою:

$$M(y_{ijg}) = \eta_{ij} = \mu + \alpha_i^A + \alpha_j^B + \alpha_{ij}^{AB}. \tag{7.22}$$

Верхні індекси позначають фактори, що взаємодіють між собою, а нижні — рівні, для яких визначається ефект.

Покажемо, що модель факторного експерименту є окремим випадком рівняння регресії. Для простоти будемо вважати, що немає взаємодії між факторами і повторень дослідів. Використовуючи вирази (7.11) і (7.19), отримаємо систему рівнянь

$$\begin{aligned} y_{11} &= \mu + \alpha_1^A + \alpha_1^B + e_{11}; \\ y_{12} &= \mu + \alpha_1^A + \alpha_2^B + e_{12}; \\ &\dots \quad \dots \quad \dots \\ y_{32} &= \mu + \alpha_3^A + \alpha_3^B + e_{32}, \end{aligned} \tag{7.23}$$

яку в матричному вигляді можна записати так:

$$\vec{Y} = X\vec{\beta} + \vec{e}, \tag{7.24}$$

де

$$\vec{Y}^T = [y_{11}, y_{12}, \dots, y_{32}], \tag{7.25}$$

X– матриця причинних або незалежних (фіктивних) факторів:

$$X = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}, \tag{7.26}$$

де перший стовпчик – це значення μ, другий, третій і четвертий – α_i^A, п'ятий і шостий – α_i^B, $i = 1, 2, 3$; $j = 1, 2$; $\vec{\beta}$ – вектор ефектів або параметрів.

Транспонований вектор $\vec{\beta}$:

$$\vec{\beta}^T = [\mu, \alpha_1^A, \alpha_2^A, \alpha_3^A, \alpha_1^B, \alpha_2^B]. \tag{7.27}$$

Вектор помилок:

$$\vec{e}^T = [e_{11}, e_{12}, \ldots, e_{32}]. \tag{7.28}$$

На основі виразів (7.16) і (7.18) отримаємо двосторонні умови:

$$\alpha_1^A + \alpha_2^A + \alpha_3^A = 0; \tag{7.29}$$

$$\alpha_1^B + \alpha_2^B = 0. \tag{7.30}$$

Обмеження (7.29) і (7.30) разом із так званими нормальними рівняннями вигляду

$$X^T\vec{Y} = X^T X\vec{\beta} \tag{7.31}$$

дають лише одні оцінки МНК. З регресійного аналізу відомо, що у разі справедливості виразу (7.21) ці оцінки одночасно будуть і оцінками максимальної правдоподібності, а також лінійними незміщеними оцінками з мінімальними значеннями дисперсії.

Таким чином, моделі факторних планів – це окремий випадок загальної лінійної регресійної моделі. Вектор параметрів $\vec{\beta}$ містить сумарне середнє, головні ефекти і взаємодії; матриця незалежних змінних X складається лише з двох значень – 0 і 1 (використовують також позначення +1 та –1, або просто символи «+» і «–»). Отже, планування експерименту означає, що X вибирається таким чином, щоб оцінки мали деякі бажані властивості.

7.7.1. Дворівневий факторний план

Повний факторний експеримент передбачає реалізацію всіх можливих комбінацій рівнів факторів. У найпростішому випадку значення факторів задають на двох рівнях. За наявності k факторів, загальна кількість комбінацій буде 2^k.

Розглянемо графічну інтерпретацію факторного експерименту (рис. 7.10). Вважатимемо, що нижньому рівню фактора відповідає значення −1, верхньому — +1, а основному — 0. Виконати подібне перетворення можна так:

$$\tilde{x}_i = \frac{(x_i - x_{i0})}{\Delta x}, \quad i = \overline{1,k}.$$

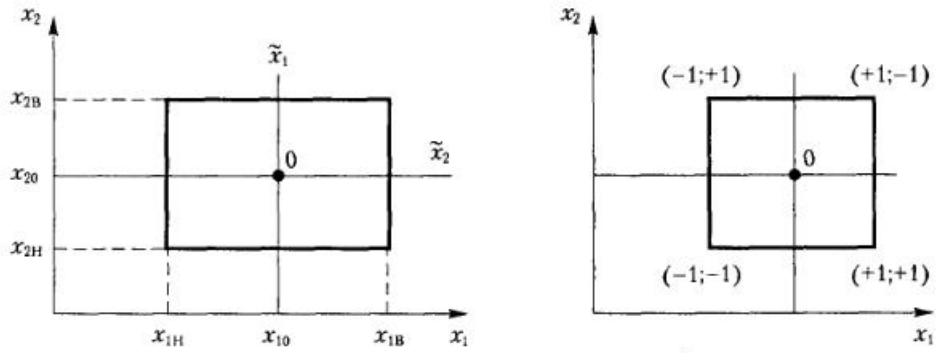


Рис. 7.10. Графічне зображення плану 2^2

Розглянемо результати проведення експериментів, зведені в табл. 7.3.

Таблиця 7.3. План дворівневого факторного експерименту

Фактор *A*	Фактор *B*	
	Рівень1	**Рівень 2**
Рівень1	y_{11}	y_{12}
Рівень 2	y_{21}	y_{22}

На основі даних табл. 7.3 можна записати таку систему рівнянь:

$$\begin{aligned} y_{11} &= \mu + \alpha_1^A + \alpha_1^B + \alpha_{11}^{AB} + e_{11}; \\ y_{12} &= \mu + \alpha_1^A + \alpha_2^B + \alpha_{12}^{AB} + e_{12}; \\ y_{21} &= \mu + \alpha_2^A + \alpha_1^B + \alpha_{21}^{AB} + e_{21}; \\ y_{22} &= \mu + \alpha_2^A + \alpha_2^B + \alpha_{22}^{AB} + e_{22}. \end{aligned} \tag{7.32}$$

Оцінки параметрів моделі (7.32) за МНК можна знайти з урахуванням додаткових умов, які випливають із виразів (7.16), (7.18) і (7.21). Тоді отримаємо:

$$\alpha_1^A = \alpha_2^A; \tag{7.33}$$

$$\alpha_1^B = \alpha_2^B; \tag{7.34}$$

$$\alpha_{21}^{AB} = \alpha_{11}^{AB}; \tag{7.35}$$

$$\alpha_{21}^{AB} = \alpha_{11}^{AB}; \tag{7.36}$$

$$\alpha_{22}^{AB} = -\alpha_{21}^{AB} = \alpha_{11}^{AB}. \tag{7.37}$$

Підставивши вирази (7.33)–(7.37) у вираз (7.32), отримаємо систему рівнянь:

$$\begin{aligned} y_{11} &= \mu - \alpha_2^A - \alpha_2^B + \alpha_{11}^{AB} + e_{11}; \\ y_{12} &= \mu - \alpha_2^A + \alpha_2^B - \alpha_{11}^{AB} + e_{12}; \\ y_{21} &= \mu + \alpha_2^A - \alpha_2^B - \alpha_{11}^{AB} + e_{21}; \\ y_{22} &= \mu + \alpha_2^A + \alpha_2^B + \alpha_{11}^{AB} + e_{22}. \end{aligned} \tag{7.38}$$

Запишемо систему рівнянь (7.38) у матричному вигляді

$$\vec{Y} = X\vec{\beta} + \vec{e}, \tag{7.39}$$

де

$$\vec{Y}^{\mathrm{T}} = (y_{11}, y_{12}, y_{21}, y_{22}), \tag{7.40}$$

$$X = \begin{bmatrix} +1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix}, \tag{7.41}$$

$$\vec{\beta}^{\mathrm{T}} = (\mu, \alpha_2^A, \alpha_2^B, \alpha_{11}^{AB}), \tag{7.42}$$

$$\vec{e}^{\mathrm{T}} = (e_{11}, e_{12}, e_{21}, e_{22}). \tag{7.43}$$

Зауважимо, що стовпчики матриці X – ортогональні, тобто

$$\vec{x}_i^{\mathrm{T}}\vec{x}_j = 0, \quad (i \neq j), \tag{7.44}$$

де $\vec{x}_i$ і $\vec{x}_j$ – будь-які два стовпчики матриці X. Очевидно, що X – невироджена матриця. Отже, оцінки МНК вектора $\vec{\beta}$ такі:

$$\vec{b} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\vec{Y}. \tag{7.45}$$

З виразу (7.44) і за умови, що

$$\vec{x}_i^{\mathrm{T}}\vec{x}_j = N, \tag{7.46}$$

де N – число дослідів (у нашому випадку $N = 4$), отримаємо

$$(X^{\mathrm{T}}X) = NI, \tag{7.47}$$

де I – одинична матриця.

Тоді деякий h-й елемент $X^{\mathrm{T}}\vec{y}$ визначається як

$$\sum_{g=1}^{N} x_{gh}y_g, \quad (h = \overline{1,H}), \tag{7.48}$$

де x_{gh} – g-й елемент вектора $\vec{x_h}$; H – загальне число параметрів (у даному випадку чотири). Підставимо вирази (7.47) і (7.48) у вираз (7.45). Тоді

$$b_n = \frac{1}{N}\sum_{g=1}^{N} x_{gh}y_g. \tag{7.49}$$

Звідси

$$b_1 = \hat{\mu} = \frac{1}{4}(y_{11} + y_{12} + y_{21} + y_{22}) = y_{\bullet\bullet}. \tag{7.51}$$

Порівняємо вираз (7.51) з визначенням головного ефекту α_2^A:

$$\alpha_2^A = \eta_{\bullet} - \eta_{\bullet\bullet}. \tag{7.52}$$

Як бачимо, оцінка головного ефекту співпадає зі значенням самого ефекту. Таким самим способом можна показати, що оцінки за МНК головного ефекту α_2^B і ефекту взаємодії α_{11}^{AB} утворюються просто за аналогією з їхніми визначеннями (7.17) і (7.21).

Зверніть увагу, в матриці X перший стовпчик стосується тільки сумарного середнього μ і містить лише одиниці зі знаком плюс. Другий та третій стовпчики відповідають головним ефектам α_2^A і α_2^B факторів A і B відповідно. Елемент g ($g = \overline{1,N}$) цих стовпчиків приймає значення -1, якщо фактор знаходиться на нижньому рівні, та $+1$ – на верхньому рівні. Для якісних факторів нижній і верхній рівні є лише мнемонічними символами.

Четвертий стовпчик матриці X показує результат взаємодії двох факторів α_{11}^{AB}. Елементи цього стовпчика – добуток елементів другого і третього стовпчиків. Тоді регресійну модель можна записати як

$$y_g = \beta_0 + \sum_{s=1}^{2} d_{gs}\beta_s + (d_{g1}d_{g2})\beta_{12} + e_g, \quad g = \overline{1,N}, \tag{7.53}$$

де $d_{gs} = -1$, якщо фактор S в g-му досліді приймає значення нижнього рівня і $d_{gs} = +1$ – у протилежному випадку; β_0 – загальне середнє μ; β_s – головний ефект S-го фактора (наприклад, $\beta_1 = \alpha_2^A = -\alpha_1^A$, $\beta_2 = \alpha_2^B = -\alpha_1^B$); β_{12} – ефект взаємодії двох факторів ($\beta_{12} = \alpha_{11}^{AB} = -\alpha_{12}^{AB} = -\alpha_{21}^{AB} = \alpha_{22}^{AB}$).

Рівняння (7.53) – це повний поліном другого степеня без квадратичних членів (немає членів $d_{g1}^2\beta_{11}$, $d_{g2}^2\beta_{22}$).

7.7.2. Факторний план 2^k

Розглянемо факторний план для випадку, коли $k = 3$ (табл. 7.4).

Таблиця 7.4. Матриця повного факторного експерименту 2^k

Комбінації факторів	Фактори			Відгук
	A	*B*	*C*	
1	−1	−1	−1	1
2	+1	−1	−1	*a*
3	−1	+1	−1	*b*
4	+1	+1	−1	*ab*
5	−1	−1	+1	*c*
6	+1	−1	+1	*ac*
7	−1	+1	+1	*bc*
8	+1	+1	+1	*abc*

Для k факторів стовпчик s-го фактора ($s = \overline{1,k}$) містить спочатку 2^{s-1} значень -1, потім 2^{s-1} значень $+1$, 2^{s-1} значень -1 і т. д.

Відгук системи визначається згідно з наступним правилом: якщо в досліді фактор A приймає значення верхнього рівня, то у відгуку символ a присутній, якщо нижнього рівня — відсутній. Аналогічно обчислюється відгук для всіх інших факторів. Значення $+1$ у таблиці показує, що в даному досліді фактор приймає значення верхнього рівня, а -1 — нижнього. Загальне число дослідів $N = 2^k$.

З матриці плану очевидно, що в одній половині дослідів фактор A приймає значення верхнього рівня, а в іншій — нижнього. Оцінка головного ефекту фактора A обчислюється за формулою

$$\hat{\alpha}^A = \frac{\sum\limits_i y_i}{N/2} - \frac{\sum\limits_j y_j}{N/2}. \tag{7.54}$$

У цьому виразі індекс i відповідає відгукам для тих комбінацій факторів, при яких фактор A приймає значення на верхньому рівні, а j — відповідно на нижньому. Тому вираз (7.54) еквівалентний виразу

$$\hat{\alpha}^A = \frac{2}{N}\left\{\sum_i (+1) y_i + \sum_j (-1) y_j\right\} = \frac{2}{N}\sum_{g=1}^{N} x_{g1} y_g,$$

де x_{g1} — g-й елемент стовпчика 1-го фактора. У загальному випадку оцінка головного ефекту фактора s має такий вигляд:

$$\hat{\alpha}^s = \frac{2}{N}\sum_{g=1}^{N} x_{gs} y_g, \quad (s = \overline{1,k}). \tag{7.55}$$

Можна показати, що аналогічно виразам (7.32)–(7.52) оцінка у виразі (7.55) — це оцінка за методом найменших квадратів головного ефекту $\hat{\alpha}^s$ фактора s. Можна довести, що оцінки за методом найменших квадратів для ефекту взаємодії факторів j, m, r визначаються як

$$\hat{\alpha}^{j,m,\ldots,r} = \frac{2}{N}\sum_{g=1}^{N} (x_{gj}\, x_{gm} \ldots x_{gr}) y_g. \tag{7.56}$$

Оцінки загального середнього $\hat{\mu}$ за методом найменших квадратів обчислюються за формулою

$$\hat{\mu} = \bar{y} = \frac{1}{N}\sum_{g=1}^{N} x_{g0}\, y_g, \tag{7.57}$$

де

$$x_{g0} = 1, \quad g = \overline{1,N}.$$

Факторний експеримент 2^k містить 2^k комбінацій факторів або точок експерименту в k-вимірному просторі з координатами ± 1, як зображено на рис. 7.11.

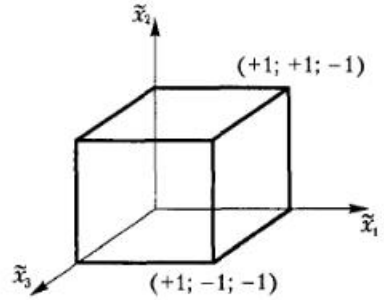


Рис. 7.11. Графічне зображення плану 2^3

Якщо позначити число дослідів через N, то можна визначити *матрицю плану*:

$$D = \{d_{ij}\}, \quad (i = \overline{1,N};\ j = \overline{1,k}),$$

де $d_{ij} = -1$, якщо j-й фактор приймає значення на нижньому рівні в i-й комбінації.

Після додавання стовпчика з одних одиниць і всіх стовпчиків добутків шуканих факторів одержимо з матриці D матрицю незалежних змінних X.

Наведемо матриці D і X (табл. 7.5) для випадку, коли $k = 3$, в яких опущено одиниці.

Таблиця 7.5. Матриці плану і незалежних змінних

Матриця плану D			Матриця незалежних змінних X							
$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{I}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{12}$	$\overline{13}$	$\overline{23}$	$\overline{123}$
−	−	−	+	−	−	−	+	+	+	−
+	−	−	+	+	−	−	−	−	+	+
−	+	−	+	−	+	−	−	+	−	+
+	+	−	+	+	+	−	+	−	−	−
−	−	+	+	−	−	+	+	−	−	+
+	−	+	+	+	−	+	−	−	−	−
−	+	+	+	−	+	+	−	+	+	−
+	+	+	+	+	+	+	+	+	+	+

Загальне середнє, головні ефекти та всі ефекти взаємодії можна оцінити, якщо помножити відповідний стовпчик матриці X на стовпчик спостереження Y.

Рівняння регресії з k факторами на двох рівнях тоді записується так:

$$y_i = \sum_{j=1}^{J} x_{ij}\gamma_j + e_i = \beta_0 + \sum_{s=1}^{k} d_{is}\beta_s + \sum_{s=1}^{k-1}\sum_{z=s+1}^{k}(d_{is}d_{iz})\beta_{sz} + \sum_{s=1}^{k-2}\sum_{z=s+1}^{k-1}\sum_{v=z+1}^{K}(d_{is}d_{iz}d_{iv})\beta_{szv} + \ldots +$$
$$+ (d_{i1}d_{i2}\ldots d_{ik})\beta_{123\ldots k} + e_i,$$

де x_{ij} і d_{is} – елементи матриць X, D відповідно; $J = 2^k$ – число параметрів регресії γ_j. Ці параметри позначають загальне середнє β_0, головний ефект β_s, ефекти двофакторної взаємодії β_{sz}, ..., ефекти взаємодії k факторів $\beta_{12\ldots k}$.

7.8. Дробовий дворівневий факторний експеримент

Планування експерименту звичайно застосовується для визначення важливих факторів, що істотно впливають на відгук (відсівний експеримент). Враховуючи те, що із зростанням числа факторів кількість комбінацій факторів швидко збільшується, необхідно виділити найбільш важливі фактори, тобто попередньо відсіяти незначущі фактори. Для цього використовуються плани порядку 2^{k-p}, коли ефекти взаємодії більш високого порядку приймаються рівними нулю (вважається, що поліном низького порядку дасть адекватне регресійне рівняння).

Кількість дослідів у повному факторному експерименті значно перевищує кількість обумовлених коефіцієнтів лінійної моделі головного експерименту, тобто повний факторний експеримент є надмірним. Якщо припустити, що деякі ефекти в цих планах є нульовими, то для побудови моделі знадобиться менше ніж 2^k дослідів. Щоб зробити такий вибір, необхідно знайти, до яких наслідків призведе відкидання деяких дослідів Розглянемо приклад повного факторного експерименту 2^3 (табл. 7.6).

Таблиця 7.6. Матриця повного факторного експерименту 2^3

№ Досліду	Матриця незалежних змінних X								
	$\bar{I}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\overline{12}$	$\overline{13}$	$\overline{23}$	$\overline{123}$	$M(\bar{y})$
1	+	–	–	–	+	+	+	–	1
2	+	+	–	–	–	–	+	+	a
3	+	–	+	–	–	+	–	+	b
4	+	+	+	–	+	–	–	–	ab
5	+	–	–	+	+	–	–	+	c
6	+	+	–	+	–	+	–	–	ac
7	+	–	+	+	–	–	+	–	bc
8	+	+	+	+	+	+	+	+	abc

Припустимо, що проведено лише чотири досліди, для яких

$$x_1x_2x_3 = +1.$$

Тоді викреслимо із плану 1-й, 4-й, 6-й, 7-й рядки (отримаємо табл. 7.7) і покажемо, як обчислити оцінки ефектів парної взаємодії із неповного факторного експерименту для чотирьох дослідів, що залишились. Наприклад, для стовпчика $\bar{1}$ отримаємо оцінку головного ефекту

$$\hat{\alpha}^A = \frac{2}{N}(y_2 - y_3 - y_5 + y_8), \tag{7.58}$$

де число дослідів $N = 4$.

Таблиця 7.7. Неповний факторний експеримент ($x_1\, x_2\, x_3 = +1$)

№ Досліду	Матриця незалежних змінних $\bar{X}$								$M(\bar{y})$
	$\bar{I}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\overline{12}$	$\overline{13}$	$\overline{23}$	$\overline{123}$	
2	+	+	−	−	−	−	+	+	a
3	+	−	+	−	−	+	−	+	b
5	+	−	−	+	+	−	−	+	c
8	+	+	+	+	+	+	+	+	abc

З формули (7.58) видно, що фактор $\bar{1}$ знаходиться на верхньому рівні в дослідах 2 і 8, а на нижньому рівні – в дослідах 3 і 5. Звідси ефект фактора A:

$$\frac{a+abc}{2}-\frac{b+c}{2}=\frac{1}{2}(a+b-c+abc).$$

Розглянемо тепер стовпчик $\overline{23}$, з якого одержуємо:

$$\hat{\alpha}^{BC}=\frac{2}{N}(y_2-y_3-y_5+y_8). \tag{7.59}$$

Взаємодія між двома факторами, що мають два рівні, визначиться таким чином. Якщо фактор C приймає значення верхнього рівня, то ефект фактора B визначається як

$$\eta_{22}-\eta_{12}, \tag{7.60}$$

а якщо фактор C приймає значення нижнього рівня, то ефект фактора B відповідно буде

$$\eta_{21}-\eta_{11}. \tag{7.61}$$

Взаємодія між факторами B і C матиме місце тільки у випадку, якщо значення виразів (7.60) і (7.61) будуть різні. Тоді взаємодія визначатиметься як «середня» різниця між (7.60) і (7.61), а саме:

$$\alpha^{BC}=\frac{1}{2}[(\eta_{22}-\eta_{12})-(\eta_{21}-\eta_{11})],$$

тобто ефект взаємодії між факторами B і C – це середнє арифметичне різниці значень ефектів B і C на їх верхніх і нижніх рівнях відповідно.

Ефект взаємодії між факторами B і C, за умови що фактори B і C знаходяться на верхньому та нижньому рівнях відповідно, можна визначити як $abc - c$. Якщо фактор C приймає значення на нижньому рівні, ефект B можна оцінити як $b - a$. Половина різниці між цими ефектами становить

$$\frac{1}{2}(abc-c)-(b-a)=\frac{1}{2}(abc-c-b+a). \tag{7.62}$$

Порівняємо вирази (7.59) і (7.58). Маємо такі ж оцінки для $\hat{\alpha}^A$ і $\hat{\alpha}^{BC}$. Або іншим шляхом це можна показати, використовуючи останній стовпчик табл. 7.7:

$$M(y_2 - y_3 - y_5 + y_8) = a - b - c + abc. \tag{7.63}$$

Запишемо праву частину виразу (7.63) як

$$a - b - c + abc = \frac{2}{N}(-1 + a - b + ab - c + ac - bc + abc) +$$

$$+ \frac{2}{N}(+1 + a - b - ab - c - ac + bc + abc)$$

при $N = 4$, або, враховуючи вирази (7.55) і (7.56), як

$$a - b - c + abc = \alpha^A + \alpha^{BC}. \tag{7.64}$$

Об'єднавши вирази (7.63) і (7.64), отримуємо

$$M(y_2 - y_3 - y_5 + y_8) = \alpha^A + \alpha^{BC}.$$

Із дробового факторного експерименту в цьому прикладі випливає, що мають місце однакові значення для головного ефекту фактора A та ефекту взаємодії факторів B і C. Це так звані змішані ефекти або ефекти, що оцінюються спільно. Якщо ефект взаємодії дорівнює нулю, значення виразу $(y_2 - y_3 - y_5 + y_8)$ буде незміщеною оцінкою α^A головного ефекту фактора A.

Таким чином, для побудови плану 2^k відкидаємо ті рядки з повного факторного експерименту, що мають значення +1 для деякого ефекту. Це так звані *напівреплiки*, тобто тут використовується половина повного факторного експерименту. Аналогічно, для другої напівреплiки необхідно відкинути ті рядки, які мають значення −1 для деякого ефекту.

При великій кількості факторів k навіть напівреплiки (тобто плани 2^{k-1}) можуть виявитись занадто громіздкими. У цих планах деякі ефекти взаємодії високого порядку можна прирівняти до нуля, та взяти меншу частину від повного факторного експерименту. Репліки, що становлять $(1/2)^p$ частину повного факторного плану з k факторами, називають планом типу 2^{k-p}.

Плани можна застосовувати послідовно, тобто спочатку одержати спостереження для одних комбінацій рівнів факторів, потім для інших і після аналізу цих спостережень вирішити, для якої комбінації (старої або нової) слід провести додаткові спостереження. Нові спостереження знову аналізуються (звичайно разом з попередніми) для ухвалення рішення про подальші спостереження і т. д. У планах 2^{k-p} можна спочатку провести частину експерименту, проаналізувати спостереження, і якщо цей аналіз покаже, що дана частина експерименту занадто мала для оцінки всіх можливих ефектів, експеримент розширюють таким чином, щоб він дав змогу оцінити вплив усіх факторів.

7.9. Пошук екстремальних значень на поверхні відгуку

Під час планування проведення експериментів для пошуку екстремальних значень відгуку застосовуються плани типу 2^{k-p} і додаткові комбінації факторів, які дають змогу оцінити відгук як функцію (поліном першого або другого порядку) від незалежних змінних. У цьому випадку всі k факторів мають бути кількісними.

Якщо метою аналізу поверхні відгуку є пошук оптимальної комбінації рівнів k кількісних факторів, то в цьому випадку виконують такі дії.

1. Планують експеримент, тобто вибір комбінації рівнів факторів. При цьому використовуються такі плани експериментів: повні 2^k, дробові типу 2^{k-p} і деякі спеціальні.
2. Проводять експеримент і будують за їх результатами рівняння регресії поверхні відгуку.
3. Здійснюють підйом по поверхні відгуку до вершини. Для визначення напрямку збільшення значення функції відгуку звичайно використовують метод якнайшвидшого підйому (крутого сходження). У напрямку, в якому очікується збільшення функції відгуку, етапи 1, 2 повторюють доти, доки не буде досягнуто області максимуму.
4. Проводять канонічний аналіз в області максимуму функції поверхні відгуку. В цьому разі визначають, чи мають місце один максимум, кілька максимумів, сідлова точка або гребінь.

Запишемо рівняння поверхні відгуку в такому вигляді:

$$y = f(x_1, ..., x_k),$$

де $x_1, ..., x_k$ – незалежні змінні, k – число факторів. У багатьох випадках мета імітаційного моделювання полягає в пошуку таких величин або рівнів незалежних змінних, при яких функція відгуку досягає екстремального значення. Для визначення напрямку руху до екстремальної точки у випадку використання кількісних, неперервних і вимірюваних величин застосовують ряд невеликих, повних і неповних факторних експериментів. Враховуючи те, що поверхня відгуку невідома, її апроксимують деякою гладкою функцією, яка є поліном першого порядку

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_kx_k,$$

або другого порядку

$$y = a_0 + a_1x_1 + a_2x_2 + ... + a_kx_k + a_{12}x_1x_2 + a_{13}x_1x_3 + ... + a_{k-1,k}x_{k-1}x_k + \\ + a_{11}x^2_1 + a_{22}x^2_2 + ... + a_{kk}x^2_k.$$

Параметри $a_0, a_1, ..., a_k, ...$ оцінюють за результатами проведення факторного експерименту.

Для пошуку екстремуму широко використовують метод якнайшвидшого підйому. Він заснований на лінійній апроксимації поверхні відгуку в околі точки.

За побудованою лінійною функцією визначається напрямок якнайшвидшого підйому $\vec{Q}$ до точки оптимуму (рис. 7.12). У напрямку вектора $\vec{Q}$ виконується невеликий крок, після чого описана процедура повторюється знову. Метод не передбачає визначення довжини кроку, однак дає можливість побачити напрямок руху до оптимуму.

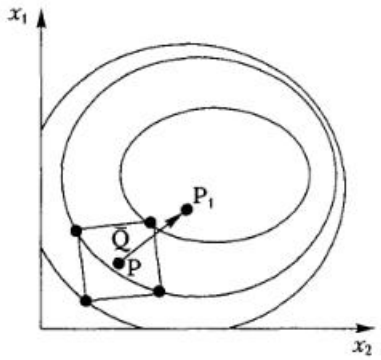


Рис. 7.12. Визначення напрямку руху до точки оптимуму

Припустимо, що дослідник провів у точці P експеримент із 2^k комбінаціями факторів і додатково два спостереження в центрі плану. За допомогою цього експерименту можна визначити коефіцієнти моделі a_0, a_1, a_2 (для випадку $k = 2$), які показують нахил площини апроксимації. Напрямок якнайшвидшого підйому показує відносні величини зміни факторів, що забезпечують максимальне збільшення значення відгуку. Піднявшись за цим напрямком до деякої точки P_1, необхідно повторити всю процедуру пошуку. Такий ітераційний процес дає змогу досягати найкращих значень відгуку. Однак ця процедура неефективна поблизу точки екстремуму, оскільки коефіцієнти a_1 та a_2, які визначають нахил апроксимуючої площини, стають невеликими за абсолютною величиною і точність їх оцінювання низька. Це означає, що поблизу точки екстремуму лінійна апроксимація поверхні відгуку є недостатньою, і виникає потреба переходу до апроксимації поліномом більш високого ступеня.

У випадку наявності двох факторів для оцінювання коефіцієнтів a_0, a_1, a_2 достатньо провести експеримент типу 2^k. Однак два додаткових спостереження в геометричному центрі плану P дають змогу не тільки уточнити коефіцієнти рівняння регресії, але й отримати кілька додаткових ступенів вільності для перевірки статистичної значущості оцінок параметрів. Те саме можна зробити за допомогою повторного експерименту. Поблизу екстремуму поверхні відгуку бажано апроксимувати поверхні щонайменше поліномом другого порядку. Для цього використовують наближення

$$y = a_0 + a_1x_1 + a_2x_2 + a_{12}x_1x_2 + a_{11}x^2_1 + a_{22}x^2_2.$$

Для оцінювання коефіцієнтів регресії цієї моделі потрібно виміряти кожний фактор принаймні на трьох рівнях, тобто провести факторний експеримент 3^k. Проте цей експеримент дає досить низьку точність оцінок коефіцієнтів регресії.

Тому спеціально для квадратичних поліномів використовують інші плани експерименту. З них найбільш корисними є центральний композиційний або ротатабельний план.

Ротатабельність (від латинської – «обертаюсь») плану означає, що стандартна помилка функції відгуку є однаковою для рівновіддалених від центра експерименту області точок плану. Такі плани утворюються шляхом додавання додаткових точок до даних, які отримують у результаті проведення факторних експериментів з 2^k комбінаціями для будь-якого числа факторів. Ці точки лежать на поверхні сфери, центром якої є точка основного рівня. Таким чином утворюються правильні геометричні фігури з центральними точками.

Наприклад, для трьох факторів маємо «куб плюс зірка плюс центральна точка». Зірка утворюється шляхом обертання кубу навколо центральної точки.

7.10. Прискорення процесу імітаційного моделювання

Досвід роботи з імітаційними моделями показує, що 60–70 % витрат на проведення експериментів пов'язані з комп'ютерним часом. Особливо зростають ці витрати у разі моделювання стохастичних систем у стаціонарному режимі, якщо потрібно отримати оцінки вихідних змінних моделі з наперед заданою точністю. У цьому випадку тривалість одного прогону моделі може становити десятки хвилин або навіть кілька годин.

Тривалість прогону моделі залежить від її складності та визначається перш за все числом подій, які фіксуються в моделі за деякий проміжок часу, а число подій у моделі залежить від її роздільної здатності, тобто вибраного дослідником ступеня деталізації моделі (див. розділ 5.4), та від швидкості протікання у ній процесів.

Справедливість останнього твердження легко довести на прикладі моделювання найпростішої одноканальної стохастичної СМО при змінюванні навантаження на пристрій для обслуговування, яке досягається зміною інтенсивності вхідного потоку. У цьому випадку основними подіями під час моделювання будуть моменти часу надходження вимог, початку та кінця їх обслуговування. Число цих подій зростатиме зі збільшенням завантаження приладу. Щоб досягти необхідної точності результатів моделювання, для таких систем доцільно застосовувати регенеративний метод аналізу для незалежних циклів (див. розділ 7.2.5). Із зростанням завантаження пристрою цикли регенерації збільшуватимуться, що спричинить збільшення тривалості прогону моделі.

Досягти істотного скорочення витрат на моделювання можна лише за рахунок цілеспрямованих дій на всіх етапах моделювання. Під час переходу до формалізації концептуальної моделі слід максимально застосовувати аналітичні залежності, вибирати такий ступінь деталізації імітаційної моделі, щоб швидкості процесів, які протікають у ній, відрізнялися не більш ніж на один-два порядки. Якщо ж потрібно моделювати рідкі події, то застосовують імітаційно-аналітичні моделі, основою яких є імітація повільних процесів та згортання швидких завдяки укрупненням станів системи та усередненням її характеристик.

Існує багато праць, присвячених моделюванню рідких подій, серед яких можна виділити [9, 15, 24, 25]. У більшості з них розглядаються аналітико-імітаційні або аналітико-статистичні методи для розрахунку надійності високонадійних систем. Так, в одному з підходів до вирішення даної проблеми застосовуються аналітико-статистичні методи розрахунку оцінок близькості шуканих показників надійності систем з характеристиками, які мають незначні відмінності [25]. Суть методу полягає в тому, що спочатку для різниці шуканих характеристик двох систем будують аналітичну формулу, яка містить деякі параметри, обчислювані в неявному вигляді. Потім за допомогою методу статистичного моделювання будують незміщені оцінки даних параметрів і підставляють їх в аналітичну формулу. Другий, дуже поширений, підхід полягає у використанні різних методів зниження дисперсії статистичних оцінок [9, 15, 24], що дає змогу скоротити вибірку або підвищити точність оцінок.

Третій підхід передбачає застосування імітаційно-аналітичних методів, в яких поєднуються аналітичні розрахунки та імітація деяких подій. Для цього використовується метод усереднення, який полягає в апроксимації швидко протікаючих процесів у імітаційній моделі на заданих відрізках часу усередненим процесом. Аналітична модель звичайно будується на основі методів аналізу СМО або за допомогою регресійного аналізу. Імітаційно-аналітичне моделювання здійснюється в два етапи. На першому етапі із застосуванням імітаційної моделі визначається момент появи найближчої події повільного процесу. Для цього використовуються розраховані аналітично усереднені характеристики швидких процесів, потім установлюють початкові значення для моделі другого етапу. На другому етапі за допомогою аналітичної моделі та отриманих на першому етапі початкових умов обчислюються усереднені характеристики швидких процесів.

Витрати під час дослідів з імітаційною моделлю можна скоротити завдяки зменшенню числа експериментів та тривалості прогону моделі.

З огляду на те, що дані одного прогону імітаційної моделі утворюють часову послідовність, або часовий ряд, доцільно спробувати застосувати методи аналізу часових рядів у імітаційному моделюванні. Такі спроби вже робилися, однак їх метою було зниження дисперсії середнього [46], а не прискорення процесу моделювання.

Аналіз часових рядів спрямований на виявлення тренду, періодичної та випадкової складових. Вихід стохастичної імітаційної моделі характеризується лише випадковою складовою, в якій виділяють перехідний та стаціонарний режими. Дані перехідного режиму мають випадковий тренд, який не дає змоги апроксимувати його гладкою кривою. Період коливань вихідних даних також має випадковий характер. Тому, будуючи параметричні моделі, для вихідних даних імітаційної моделі доцільно обмежитись тільки моделями авторегресії. Авторегресійний процес порядку k з постійними коефіцієнтами визначається рівнянням

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_k y_{t-k} + e_t, \qquad (7.63)$$

тобто значення ряду в момент часу t виражається через попередні значення та випадкове збурення e в момент t.

На практиці звичайно використовуються авторегресійні моделі першого та другого порядків, що відповідає випадковим процесам Маркова та Юла. Параметри $a_1, a_2, ..., a_k$ оцінюються через коефіцієнти автокореляції, а параметр a_0 – за середнім значенням ряду μ. Для марківського процесу $a_0 = \mu$, $a_1 = r_1$. Для процесу Юла значення обчислюються за формулами

$$a_0 = \mu; \quad a_1 = \frac{(1-r_2)r_1}{1-r_1^2}; \quad a_2 = \frac{r_2 - r_1^2}{1-r_1^2},$$

де μ – середнє по ряду; r_1, r_2 – значення автокореляційної функції для зсуву 1 та 2.

Розраховані коефіцієнти a_0, a_1, a_2 використовуються на першій ітерації підбору коефіцієнтів утворення параметричної моделі, яка описується рівнянням (7.63). Після цього будується згладжений ряд і за методом найменших квадратів мінімізується відхилення від початкового ряду. У загальному випадку порядок авторегресійної моделі визначається числом значущих коефіцієнтів частинної автокореляційної функції.

В основу авторегресійної моделі покладено гіпотезу про стаціонарність досліджуваного явища, і відносно випадкової компоненти e_t висувається гіпотеза, що вона є випадковим процесом у широкому розумінні з нормальною функцією розподілу та математичним сподіванням, що дорівнює нулю. У разі невиконання цих умов переходять від вихідного ряду до його різниць першого або другого порядку.

Дослідження вихідних даних різних імітаційних моделей підтвердили правильність викладених вище положень. Для утворення параметричних моделей використано методику Бокса–Дженкінса [6, 19]. Аналіз результатів показав [60], що авторегресійна модель може застосовуватись тільки після вилучення даних перехідного режиму. В усіх випадках не доводилось використовувати авторегресійні моделі більш ніж другого порядку. Мінімальне число точок вихідного ряду має бути порядку 150, причому бажано застосовувати дані, які безпосередньо йдуть за перехідним періодом. Оскільки більшість імітаційних експериментів провадиться з метою оцінювання математичного сподівання деякої вихідної величини із заданою точністю, то використання авторегресійної моделі дає змогу визначити тенденцію зміни досліджуваної вихідної величини в часі та прогнозувати її подальшу зміну в майбутньому. Причому якщо вихідні дані одного прогону імітаційної моделі є реалізацією деякого випадкового процесу, то завдяки використанню параметричної моделі можна передбачати середню оцінку для прогнозованого інтервалу часу.

В момент часу $t + l$ модель авторегресії може бути представлена як

$$y_{t+l} - a_1 y_{t+l-1} + ... + a_k y_{t+l-k} = e_t.$$

Якщо взяти з обох боків умовне математичне сподівання при фіксованих до моменту часу t попередніх значеннях отримуємо

$$y_{t+l} - a_1 y_{t+l-1} + ... + a_k y_{t+l-k} = 0.$$

Різницеве рівняння має розв'язок

$$y_{t+l} - b_0^{(t)} f_0[t+l] + f_1[t+l] + \ldots + b_k^{(t)} f_k[t+l], \quad l > 0,$$

де $f_0[t+l], f_1[t+l], \ldots, f_k[t+l]$ – функція усереднення до часу l.

Для заданого часу t коефіцієнти $b_j^{(t)}$ $(j = 0, \ldots, k)$ постійні для усіх l. Зокрема, якщо $k = 1$, то для моделі авторегресії першого порядку маємо

$$\tilde{y}_{t+1} = a\tilde{y}_t, \quad b_0^{(t)} = \tilde{y}_t, \quad \tilde{y}_{t+1} = a\tilde{y}_t a^l; \quad (l > 0),$$

де $\tilde{y}_t = y_t - \mu$.

Для стаціонарного процесу необхідне виконання умови $-1 < a < 1$. Якщо $l \to \infty$, то $\tilde{y}_{t+l}$ буде прямувати до μ, тобто $\tilde{y}_{t+l} - \mu \approx 0$.

Таким чином, при великих значеннях випередження l з мінімальною середньоквадратичною похибкою значення ряду має експоненціально прямувати до його математичного сподівання. Те ж саме справедливе для процесу авторегресії другого порядку.

Звичайно, дуже складним є визначення інтервалу прогнозування l [34, 58], оскільки для цього не існує точних методів. У разі використання авторегресійної моделі прогноз збігається до деякого сталого значення, якщо наступне значення y_t дорівнює попередньому y_{t-1}, або для моделі першого порядку

$$y_t = y_{t-1} = a_0/(1 - a_1). \tag{7.64}$$

Звідси випливає, що число точок прогнозу від моменту збіжності до досягнення сталого значення залежатиме від різниці між останнім значенням рівня вихідного ряду та величиною, розрахованою за формулою (7.64). Причому значення прогнозованої величини буде зменшуватись, якщо останнє значення вихідного ряду більше обчисленої величини, та збільшуватись, якщо воно менше, ніж розрахована величина. Саме ж значення розрахованої величини буде тим більшим, чим ближче значення a_1 до одиниці. Причому при значенні $1 - a_1$, близькому до нуля, спостерігається сильна залежність середнього значення, знайденого за авторегресійною моделлю (7.63), від вихідного значення ряду через втрату точності обчислень. Тому, якщо значення коефіцієнта a_1 близьке до одиниці, слід відкидати дані перехідного процесу так, щоб перше значення ряду було великим і більшим за середнє значення вихідного ряду [60].

Оскільки початкове значення $a_1 = r_1$, тобто дорівнює величині оцінки коефіцієнта автокореляції вихідного ряду з лагом один, число точок передбачення до моменту, коли $y_t = y_{t-1}$, буде великим, якщо значення часового ряду дуже корельовані. У цьому разі необхідно керуватися правилом: число точок прогнозу має бути не більшим за довжину вихідного ряду. У решті випадків прогноз здійснюється доти, доки не буде виконано умову (7.64).

Для моделювання процесу авторегресії другого порядку умова (7.64) матиме вигляд

$$y_{t-2} = y_{t-1} = y_t = a_0/(1 - a_1 - a_2). \tag{7.65}$$

Параметр a_0 обчислюється через середнє значення, знайдене згідно з авторегресійною моделлю (7.63), з урахуванням коефіцієнтів поправки a_1, a_2, ..., a_k. Якщо значення ряду некорельовані, тобто незалежні, то $a_0 = \mu$, і згідно з умовами (7.65) всі наступні прогнозовані значення дорівнюватимуть μ, тобто ці значення будуть збігатися до середнього. У цьому разі втрачається сенс створення параметричної моделі виду (7.63) для вихідних змінних імітаційної моделі.

Дослідження моделей одноканальних СМО [60] показали, що використання авторегресійної моделі для оцінювання часу перебування заявки в системі дає змогу скоротити обсяг вибірки у 2–100 і більше разів залежно від ступеня корельованості даних. Чим більш корельовані дані, тим більшого скорочення вибірки можна досягти. Однак отримані в результаті моделювання оцінки дисперсії середнього для прогнозованих значень можуть істотно перевищувати оцінки, одержані за результатами моделювання. Причому вони зростають із збільшенням корельованості та періоду прогнозу. Це підтверджує той факт, що у разі сильної кореляції між членами ряду даних дисперсія y_t буде набагато більшою за дисперсію e_t і ряд досить малих збурень породжуватиме значні коливання [19]. Перехідний процес у даному випадку буде уповільнюватись.

Тому бажано створити параметричну модель, яка не лише передбачала б тенденцію прагнення оцінки середнього до математичного сподівання стаціонарного ряду, а й давала б точнішу оцінку дисперсії, ніж імітаційна модель.

Ураховуючи те, що авторегресійна модель часового ряду містить інформацію про взаємозв'язок значень змінної у часі, а функція розподілу – інформацію про ймовірність набуття випадковою величиною визначеного значення, бажано було б відтворити можливі значення для прогнозованих рівнів вихідної змінної імітаційної моделі. Це можна зробити для стаціонарних у вузькому розумінні слова процесів, в яких функція розподілу не змінюється в часі, за допомогою методу статичного моделювання.

Тоді алгоритм використання параметричної моделі буде таким.

1. Побудова прогнозованих значень за допомогою авторегресійної моделі (7.63) за методикою, описаною вище.
2. Моделювання випадкової величини з функцією розподілу, отриманої за значеннями вихідного ряду, для кожного значення прогнозу з урахуванням зміщення середнього значення вихідного ряду та здобутого значення для точки прогнозу. Оскільки врахування зміщення середнього може привести до того, що будуть отримані значення, які не можна здобути на імітаційній моделі, то в разі перевищення максимально можливої величини треба запам'ятати максимальне значення. Аналогічно, якщо значення менше за мінімально можливе, треба запам'ятати мінімальне значення.
3. Додавання даних, отриманих під час моделювання, до вихідного ряду. Оскільки дані, які додають, не корельовані, то значення автокореляційної функції зменшаться тим більше, чим більше прогнозованих даних буде додано до вихідного часового ряду.
4. Оцінювання точності результатів. Через те, що процедура утворення авторегресійної моделі вимагає проведення розрахунків автокореляційної функції,

то для оцінювання точності результатів моделювання доцільно застосовувати метод розкладання всієї вибірки, що включає вихідні значення та значення, які прогнозують, на підвибірки. Обсяг підвибірки обчислюється за значеннями автокореляційної функції, коли коефіцієнти автокореляції стають незначущими (див. розд. 7.2.4). Середні значення підвибірок будуть незалежними, і їх можна використати для утворення довірчого інтервалу. Якщо отримана точність виявиться недостатньою, слід збільшити обсяг початкової вибірки для утворення параметричної моделі і повторити процедуру передбачення.

Така методика дає змогу побудувати адаптивний алгоритм для класу регенеруючих моделей. У цьому разі збирання даних, отриманих на виході імітаційної моделі, закінчується із припиненням циклу регенерації моделі. Потім будується параметрична модель, здійснюється прогнозування та оцінюється точність вихідної змінної. Якщо одержана точність виявляється недостатньою, то імітаційний експеримент поновлюється з нового циклу регенерації. Здобуті дані додають до вже існуючих і знову оцінюють точність. У разі потреби повертаються на перший етап побудови авторегресійної моделі.

На рис. 7.13 наведено часовий ряд тривалості перебування вимог T_{pr} в одноканальній СМО виду $M/M/1$, який одержано під час прогону імітаційної моделі з 120 циклами регенерації та коефіцієнтом завантаження пристрою 0,7. Оцінка середнього значення часу перебування вимог у системі дорівнює 2031,61 + 717,2153 з довірчою ймовірністю 0,9.

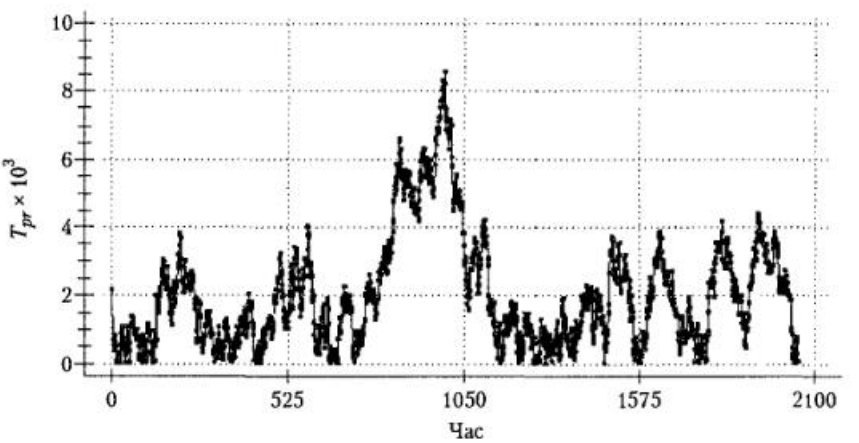


Рис. 7.13. Часовий ряд тривалості перебування вимог T_{pr} у системі

Автокореляційну функцію для цього часового ряду зображено на рис. 7.14, а частинну автокореляційну функцію – на рис. 7.15, на якому видно, що параметрична модель буде першого порядку. Прогнозування щодо цього ряду здійснювалось після відкидання 1800 значень для перехідного процесу з використанням авторегресійної моделі, яка описується рівнянням

$$y_t = 14{,}9618 + 0{,}98615\, y_{t-1}. \qquad (7.66)$$

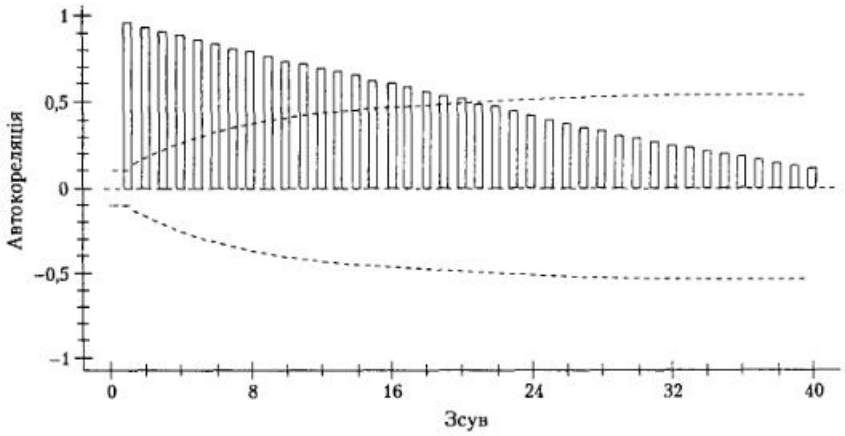


Рис. 7.14. Автокореляційна функція часового ряду

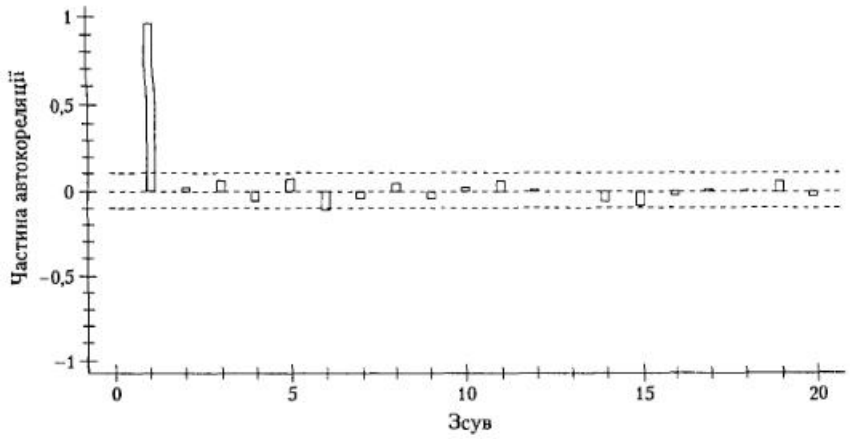


Рис. 7.15. Частинна автокореляційна функція

Про якість параметричної моделі (7.66) можна скласти уявлення за графіком помилки e_t, яка повинна мати нормальну функцію розподілу з математичним сподіванням, що дорівнює нулю. Функцію розподілу помилки e_t на нормованому графіку зображено на рис. 7.16.

Графік передбачення 250 точок з 95 %-м рівнем довіри наведено на рис. 7.17. Коефіцієнт автокореляції даних з одиничним зсувом дорівнює 0,9734, оцінка середнього значення часу перебування, отримана на основі згладженого ряду, – 2206,455.

На рис. 7.18 наведено відновлений ряд для функції розподілу, що отримана на основі вихідного ряду. Оцінка середнього значення часу перебування вимог у СМО для цього ряду становить 1821,92. Аналітичні розрахунки для такої СМО дають значення середнього часу перебування заявки 1800. Щоб дістати середнє значення з 5 %-м відхиленням від середнього та довірчою ймовірністю 0,9 на імітаційній моделі, слід провести моделювання близько 12 000 циклів регенерації. Тоді початковий обсяг вибірки зросте більш ніж у 100 разів.

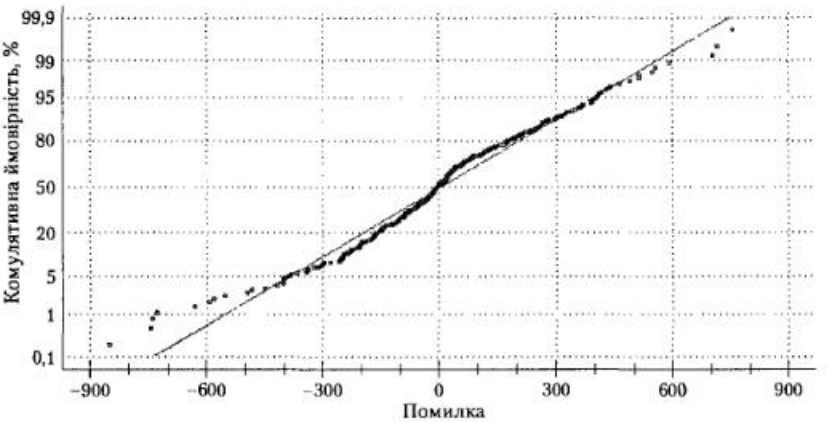


Рис. 7.16. Функція розподілу помилки моделювання

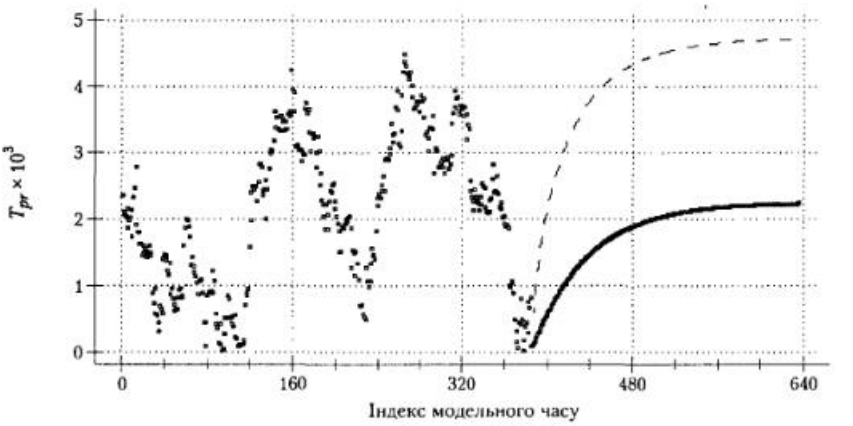


Рис. 7.17. Графік прогнозованих значень

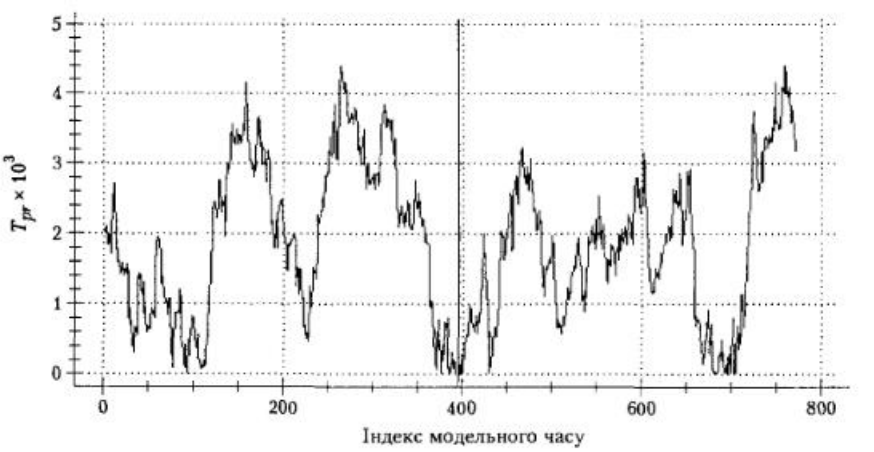


Рис. 7.18. Відновлений ряд

Отже, використання цієї методики дає змогу значно скоротити обсяг вибірки під час моделювання процесів з сильною кореляційною залежністю.

Висновки

- Для будь-якого експерименту з моделлю має існувати можливість його повторного проведення іншими дослідниками.
- Вихідні дані імітаційних експериментів потрібно структурувати та інтерпретувати таким чином, щоб їх можна було використовувати для прийняття рішень стосовно структури і параметрів системи або моделі.
- Планування експерименту – це розробка такого плану проведення експерименту, який дає можливість за мінімальну кількість прогонів моделі і за мінімальних затрат ресурсів зробити статистично значущі висновки або знайти найкращі рішення щодо функціонування системи.
- Імітаційне моделювання провадиться, як правило, з метою визначення деяких екстремальних значень характеристик модельованої системи (оптимізуючий експеримент) або для виявлення важливих факторів, що впливають на модельовану систему (відсівний експеримент).
- Оцінювання точності результатів моделювання пов'язане з побудовою довірчих інтервалів для вихідних змінних (відгуків) моделі.
- Застосування методів зниження дисперсії дає змогу при заданому обсязі вибірки збільшити точність оцінювання відгуку або при заданій точності скоротити обсяг вибірки.
- Під час моделювання рідких подій застосовують імітаційно-аналітичні моделі, основою яких є імітація повільних процесів та згортання швидких процесів завдяки укрупненням станів системи та усередненням її характеристик.

Контрольні запитання та завдання

1. Напишіть програму моделювання СМО виду $M/M/1$ з коефіцієнтом завантаження 0,9. Використовуючи графічні можливості середовища GPSS, побудуйте графік спостереження за часом чекання в черзі в процесі моделювання. Змініть початкові умови моделювання так, щоб під час старту кількість вимог у черзі змінювалась від 0 до 5 з кроком в одну вимогу та визначіть тривалість перехідного процесу при різній кількості вимог. При якій кількості вимог тривалість перехідного процесу буде мінімальною? Поясніть чому.
2. Як під час скінченного імітаційного моделювання розрахувати кількість прогонів моделі, необхідну для того, щоб отримати результат з заданою точністю?
3. Назвіть проблеми, що виникають під час планування імітаційних експериментів. Як вирішуються ці проблеми для стаціонарних процесів?

4. У чому полягають особливості проведення експериментів з використанням регенеруючих процесів? Поясніть свою відповідь на прикладах.
5. В яких випадках під час імітаційного моделювання доцільно застосовувати методи зниження дисперсії?
6. Назвіть початкові умови, необхідні для планування експериментів. Яким чином їх можна виконати під час імітаційного моделювання?
7. Як можна скоротити кількість експериментів у повному факторному експерименті? Перерахуйте необхідні припущення щодо взаємодії факторів.
8. Поясніть, чому під час розв'язання задач оптимізації систем для проведення імітаційних експериментів обираються ротатабельні композиційні плани.
9. Дослідіть роботу алгоритму пошуку екстремуму функції з двома змінними $z = \sqrt{9 - x^2 - y^2}$, яка є верхньою частиною сфери з радіусом 3, за допомогою оптимального експерименту в GPSS World. Скористайтеся програмою, наведеною в прикладі 7.5 у книжці [61]. Чому під час пошуку екстремуму функції відбувається перехід від лінійної регресійної моделі до нелінійної?
10. Які проблеми виникають під час моделювання рідких подій? Як вони вирішуються?

Розділ 8

Прийняття рішень за результатами моделювання

- Подання результатів моделювання
- Методи пошуку оптимальних значень параметрів
- Прийняття рішень щодо удосконалення систем
- Порівняння альтернативних конфігурацій системи
- Приклади прийняття рішень

Заключний етап експериментів з імітаційними моделями – це прийняття рішень щодо визначення оптимальних параметрів системи або удосконалення її структури, тому результати моделювання важливо подавати у зручному для аналізу і зрозумілому для замовника чи аналітика вигляді. Вибір методів прийняття рішень за результатами імітаційного моделювання суттєво залежить від мети досліджень. Це можуть бути як складні методи оптимізації, так і методи простого перебору чи вибору одного із кількох варіантів.

8.1. Подання результатів моделювання

Імітаційне моделювання завжди використовується для прийняття рішень щодо модельованої системи, але перш ніж представляти результати моделювання аналітику, їх потрібно перетворити таким чином, щоб вони мали наочний та інформативний вигляд. Більшість програмних засобів імітаційного моделювання відображають результати моделювання у формі стандартного звіту, в якому зібрані статистичні дані, що стосуються характеристик системи, наприклад завантаженості ресурсів, довжини черг, часу перебування вимог у чергах і пристроях, а також гістограми розподілів вихідних величин та ін. Крім стандартних статистичних даних користувач може включити у звіт необхідні йому дані й надрукувати їх у потрібному для нього вигляді.

Враховуючи те, що кількість прогонів імітаційної моделі зазвичай дуже велика, досягти високої наочності даних можна, якщо спочатку накопичити їх в окремій базі даних, а потім надати користувачу засоби для доступу до даних та перетворення їх у зручний для виконання аналізу вигляд. Такий спосіб дає можливість використовувати всі існуючі програмні засоби статистичної обробки інформації.

Результати моделювання подаються звичайно у вигляді таблиць або графіків, що подібні до тих, які використовуються для зображення даних у комерційних статистичних пакетах. Дослідження функцій розподілу випадкових величин, одержаних під час імітаційного моделювання (часових рядів), здійснюється за допомогою таблиць частот і гістограм. Для цього, наприклад у мові GPSS, використовується блок TABULATE, який дає змогу побудувати таблицю розподілу, що містить інформацію про середнє значення, середньоквадратичне відхилення досліджуваної величини, ліву та праву межі довірчих інтервалів оцінок параметрів, відсоткову частість потрапляння транзактів у інтервал групування. Цю інформацію можна зобразити графічно, наприклад за допомогою математичних пакетів, таких як STATISTICA, MathCAD, Maple, і використати для перевірки гіпотез відносно функцій розподілу випадкових величин. Приклад гістограми часу перебування вимог у СМО та апроксимацію її за допомогою експоненціального закону розподілу наведено на рис. 8.1.

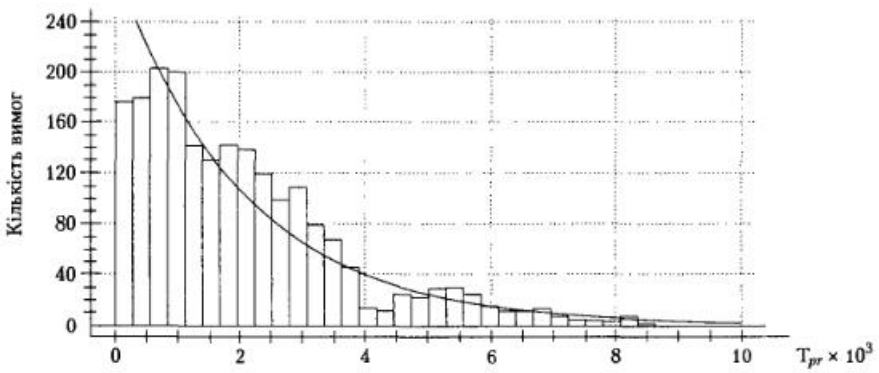


Рис. 8.1. Гістограма часу перебування вимог у СМО

Ще один параметр, який часто використовується для подання результатів моделювання та обчислюється за даними таблиць розподілів випадкових величин, — кумулятивна відсоткова частка (процентіль). Цей параметр показує, що досліджувана випадкова величина, наприклад у 95 випадках із 100, не перевищить деякого значення. Найчастіше процентіль застосовується у моделюванні діалогових систем або систем передавання даних, для яких час реакції системи або проходження інформації по мережі не повинен перевищувати наперед задане значення. Таким чином, процентіль показує рівень надійності отриманої граничної оцінки.

Візуалізація даних також є важливим етапом статистичного аналізу результатів моделювання. Важливо підібрати тип графіка, який найбільше відповідає даним, що аналізуються. У прикладній статистиці навіть існує термін «графічний аналіз даних».

З усіх статистичних пакетів, за допомогою яких можна обробити отримані під час моделювання дані, найбільш придатним є пакет STATISTICA [7]. Цей пакет має потужні засоби графічного зображення статистичної інформації та нараховує понад 100 типів різних графіків, серед яких є дво- та тривимірні, матричні та тернарні. Двовимірні графіки дають змогу будувати двовимірні гістограми та діаграми розсіяння, діаграми розсіяння з гістограмами, графіком значень змінної, графік

на ймовірнісному папері, графіки квантіль–квантіль, ймовірність–ймовірність, графіки діапазону значень, двовимірні графіки значень, наукові двовимірні графіки, кумулятивні графіки, кругові діаграми, графіки пропущених значень і графіки функцій, аналітично заданих користувачем.

За допомогою статистичних просторових графіків можна побудувати точкові графіки (об'ємна діаграма розсіяння), графік поверхні, карту ліній рівнів, графік трасування, різні типи графіків за категоріями, тернарні графіки за категоріями та графіки функцій, аналітично заданих користувачем. Статистичні матричні графіки призначено для візуалізації зв'язків між змінними моделі.

Слід мати на увазі, що неправильний вибір типу графіка, який відображає результати імітаційного моделювання, може не лише не полегшити, але й, навпаки, ускладнити їх інтерпретацію. Одна із серйозних проблем пов'язана з тим, що часто на графіку неможливо відобразити велику кількістю значень; це ускладнює роботу з двовимірними таблицями і статистичними графіками. Іноді наявність взаємозв'язків, структур і трендів у даних важко помітити, доки результати моделювання не будуть упорядковані або зображені відповідним способом.

Крім статистичних графіків, наведених вище, для подання результатів моделювання в графічному вигляді широко використовуються діаграми Ганта та графіки Ківіата.

Діаграми Ганта призначені для зображення спільних характеристик (профілів роботи), рівнів завантаження компонентів системи або для відображення графіків (розкладу) їх роботи. На рис. 8.2 наведено приклад діаграми, що відображає результати моделювання виробничої дільниці, на якій розміщені фрезерний, токарний та стругальний верстати.

Рис. 8.2. Приклад діаграми Ганта з результатами моделювання виробничої дільниці

На діаграмі видно, що завантаженість всіх верстатів досить висока, але проміжок часу одночасної роботи всіх верстатів на дільниці незначний і становить дуже малий відсоток від загального часу роботи всіх верстатів. Це вказує на існування проблеми нерівномірності завантаження ресурсів, яку важко було б виявити, розглядаючи, наприклад, табличні дані про завантаженість верстатів. Профіль використання обладнання дозволяє аналітику зробити висновки щодо станів активних елементів (верстатів) і часу їх спільної роботи. Про такі діаграми мова йшла в розділ 2, де розглядався метод графічного моделювання СМО.

Дані моделювання можна подати і у вигляді кругового графіка (рис. 8.3), радіуси якого використовуються як осі для зображення характеристик роботи системи. Така стисла форма зображення даних називається графіком Ківіата, який теж дає змогу аналітику виявити наявність деяких проблем. Звичайно на радіусах даного графіка відкладають по черзі так звані хороші та погані значення параметрів. Чим більша різниця між цими значеннями, тим більше графік Ківіата буде схожий на зірку. Приклад такого графіка для зображеної на рис. 8.2 діаграми наведено на рис. 8.3. Величини Y_1, Y_3, Y_5 відповідають коефіцієнтам завантаження верстатів, а величини Y_2, Y_4, Y_6 – коефіцієнтам простою токарного, фрезерного та стругального верстатів.

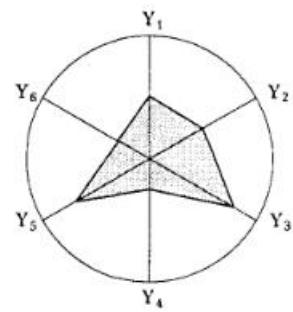


Рис. 8.3. Приклад графіка Ківіата

Крім графічних засобів візуалізації даних, отриманих в результаті експериментів з моделлю, сучасні пакети для імітаційного моделювання дискретних систем у більшості випадків мають засоби для дво- або тривимірної анімації, які дають змогу наочно відтворити динаміку роботи модельованої системи. Засоби анімації, вбудовані у пакети, і анімаційні ефекти можуть відтворюватись під час роботи імітаційної моделі. Існують і окремі пакети для створення анімації, наприклад Proof Animation компанії Wolverine software, які підключають до імітаційної моделі за допомогою стандартного інтерфейсу. Для анімаційного відтворення роботи імітаційної моделі необхідно зберегти в окремому файлі траси переміщення всіх динамічних об'єктів і задати напрямки їх руху в кожній ситуації (сценарії). Окремо задається положення статичних об'єктів на екрані монітора. Приклад такої анімації під час моделювання роботи госпіталю наведено на рис. 8.4.

Анімація моделі доцільна насамперед під час демонстрації імітаційного проекту замовнику, тому що дає можливість, не вникаючи в деталі побудови моделі, показати, як буде працювати система, і швидко оцінити достовірність результатів моделювання.

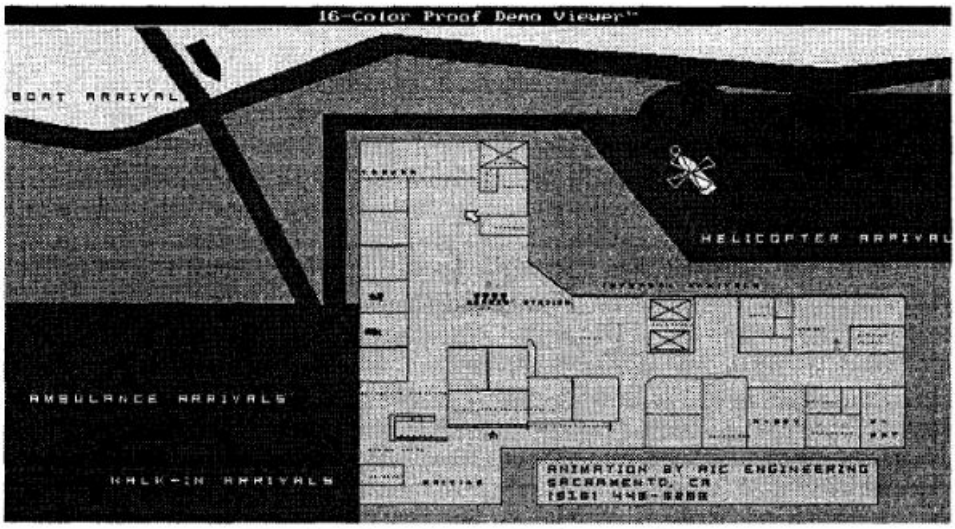


Рис. 8.4. Анімація моделі роботи госпіталю

Анімація допомагає і під час прийняття рішень, оскільки, спостерігаючи за роботою модельованої системи, можна приймати чи відкидати гіпотези щодо її удосконалення, які було б важко сформулювати, вивчаючи лише числові результати моделювання. Прикладом використання анімації може бути імітаційна модель руління літаків на стоянку для диспетчера аеропорту.

8.2. Методи прийняття рішень

Імітаційне моделювання у більшості випадків використовується для прийняття рішень під час проектування структури складної системи або для пошуку оптимальних значень її параметрів. Можна визначити кілька основних напрямків прийняття рішень за результатами моделювання:

- пошук найкращих стосовно деякого критерію ефективності значень параметрів складних систем управління;
- пошук оптимального значення критерію ефективності системи;
- порівняння альтернативних варіантів структури системи та визначення найкращого з них;
- моделювання аварійних ситуацій за сценарієм типу «що буде, якщо...».

У разі оптимізації управлінських рішень в умовах невизначеності щодо модельованої системи зазвичай існує єдиний, як правило, економічний, критерій, що підлягає оптимізації. Проте отримане за допомогою математичної моделі рішення рідко є найкращим з будь-якого погляду і потребує коригування для узгодження з реальною ситуацією. Для цього необхідно вносити зміни у структуру моделі й поновлювати пошук нових найкращих варіантів, тобто сама процедура пошуку оптимальних рішень є ітераційною і включає ітераційні методи настроювання

моделі. Для такої процедури пошуку критерій не може бути виражений явною функцією від вхідних змінних і параметрів системи. Скоріше, вона передбачає наявність критерію якості роботи системи, значення якого можна знайти лише під час прогонів імітаційної моделі. Загальну схему імітаційного моделювання за підтримки методів оптимізації наведено на рис. 8.5.

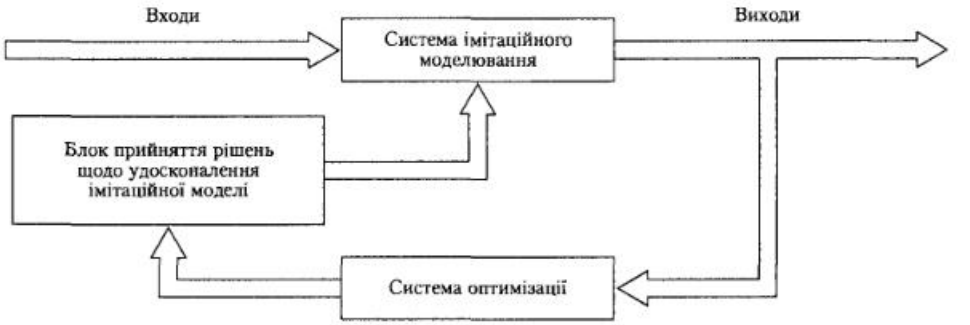


Рис. 8.5. Схема імітаційного моделювання за підтримки методів оптимізації

Реалізувати наведену схему можна різними методами, залежно від програмних засобів, що використовуються під час моделювання. Якщо до складу системи імітаційного моделювання входить підсистема оптимізації, то наведена схема реалізується в рамках однієї програмної системи. Але в більшості випадків під час реалізації цієї схеми застосовуються окремі пакети для імітаційного моделювання та оптимізації. Що стосується блока, призначеного для прийняття рішень відносно удосконалення імітаційної моделі, то ці процедури важко формалізувати, тому їх доцільно доручити аналітику.

Слід відзначити, що непросто відповісти на питання стосовно можливості отримання за допомогою імітаційної моделі оптимального значення критерію ефективності системи. Оптимальне значення критерію можна знайти, якщо чітко вказані змінювані параметри системи та діапазони їх змін. Наприклад, коли обчислюється максимальна пропускна здатність системи, а змінюваним параметром є інтенсивність замовлень, які надходять до системи, то неодмінно існує значення оптимальної пропускної здатності системи. Однак на практиці такі прості ситуації трапляються дуже рідко. У більшості випадків пропускна здатність системи неявно залежить від безлічі параметрів, діапазони зміни яких визначити важко. Часто витрати на удосконалення системи залежать від наявних коштів і затрат, спричинених можливими змінами у системі. Границі деяких параметрів також важко оцінити кількісно. Із цих причин більшість практичних задач зовсім не схожі на математичні задачі оптимізації.

Виникають ситуації, коли оптимальний розв'язок задачі існує, але його дуже важко знайти, або зусилля на його пошук невиправдані, наприклад, через значну вартість робіт, пов'язаних із цим пошуком, або обмеження у часі (термін виконання більшості імітаційних проектів становить кілька місяців). Тому зазвичай здійснюється пошук рішень, які істотно покращують значення критерію ефективності системи.

Проблема ще більше ускладнюється, коли один імітаційний проект є частиною загального проекту, що передбачає створення нової виробничої або технічної системи, в якій імітаційна модель використовується для перевірки можливих конфігурацій системи або оптимізації режимів її роботи. Прикладами таких проектів можуть бути проектування і оптимізація технологічних процесів виготовлення нових моделей автомобілів у концернах Тойота та BMW.

Таким чином, у разі пошуку найкращих рішень доцільніше використовувати термін «удосконалення», замість терміна «оптимізація». Методи оптимізації застосовуються на кожній ітерації пошуку найкращих рішень лише тоді, коли чітко визначена мета (цільова функція) оптимізації. Процедури удосконалення використовуються, якщо є можливість поліпшити показники модельованої системи і пошук цих показників виправданий. У більшості випадків такі рішення приймаються за результатами статистичного аналізу вихідних даних імітаційної моделі. У разі аналізу СМО найважливішими показниками, отримуваними під час статистичного аналізу, є коефіцієнти завантаження ресурсів системи та довжини черг. Наприклад, якщо для деякого варіанта системи знайдене оптимальне рішення, але коефіцієнт завантаження одного з ресурсів надто малий, то необхідно перевірити можливість заміни цього ресурсу на менш продуктивний та більш дешевий. Для визначення можливості поліпшення показників системи зазвичай висувають гіпотези стосовно параметрів системи, а потім за допомогою імітаційної моделі перевіряють, чи є їх значення оптимальними. Як правило, кількість можливих гіпотез невелика, що дає змогу знайти методом їх простого перебору найефективніший варіант системи.

8.3. Методи оптимізації

Для пошуку оптимальних значень параметрів системи можна застосовувати традиційні методи лінійного, нелінійного та дискретного програмування. Однак під час моделювання складних виробничих систем часто виникають проблеми, зумовлені статистичним характером процесів, що протікають у системі, та необхідністю виконання процедур статистичної обробки результатів моделювання. При застосуванні методів оптимізації під час моделювання необхідно враховувати як фактор випадковості, який впливає на значення оцінок критеріїв якості роботи, так і обмеження на значення параметрів систем. Методи оптимізації, що розглядаються далі, легко інтегруються з імітаційними моделями, проте назвати метод, завжди ефективний у будь-якому імітаційному експерименті для будь-якої імітаційної моделі, неможливо. У більшості випадків можна лише оцінити точність результатів та визначити кількість ітерацій, потрібних для отримання належного результату. Витрати, необхідні для обчислення незсунених оцінок параметрів з мінімальною дисперсією, визначаються відповідно до тривалості комп'ютерних обчислень.

Найпростіший метод оптимізації – випадковий відбір деяких точок у просторі пошуку оптимальних значень параметрів системи. Точки з найбільшим (або найменшим) значенням критерію вважаються оптимальними. Цей метод зручно застосовувати у комбінації з іншими методами пошуку множини початкових точок у за-

дачі глобальної оптимізації параметрів. Найбільш ефективним методом подолання локальної оптимальності для дискретної оптимізації є метод пошуку з обмеженнями.

Ймовірнісні пошукові методи мають деякі переваги порівняно з градієнтними методами оптимізації. Під час використання імовірнісних методів пошуку цільова функція може бути розривною. Ці методи доцільно застосовувати для пошуку глобального оптимуму багатомодальних функцій.

У разі вибору методу Хука і Джівса [5] необхідно враховувати обмеження на число параметрів оптимізації (їх не має бути більше 20). Генетичні методи дають оптимальний розв'язок і при значно більшій кількості змінних.

Методи пошуку за зразком найкраще підходять для розв'язання задач оптимізації при невеликій кількості параметрів оптимізації без обмежень і порівняно з генетичними методами потребують менше обчислювальних затрат. Найбільш перспективними є методи стохастичної апроксимації, під час здійснення яких використовують комбінацію градієнтних методів і методів пошуку на поверхні відгуку.

Методи стохастичної апроксимації дають змогу оптимізувати параметри імітаційної моделі протягом однократного її прогону. Під час прогону моделі існує можливість коригувати оцінки параметрів оптимізації на основі спостережень за вибірковими значеннями цільової функції. Отже, відпадає потреба в повторних прогонах моделі. Така послідовність багаторазових спостережень і модифікацій значень цільової функції дозволяє отримати оптимальну оцінку відразу після імітаційного прогону моделі. Крім того, дані методи оптимізації потребують менше комп'ютерного часу, їх також зручно використовувати для оптимізації в реальному часі та в системах керування із застосуванням імітаційної моделі.

Метод пошуку на поверхні з використанням градієнта має ті ж переваги, що й методи пошуку за зразком та градієнтний метод, а також декілька переваг, які характерні саме для нього:

- існує можливість одноразового прогону моделі, на відміну від методів пошуку на поверхні відгуку;
- на кожній ітерації використовується не лише значення локального градієнту, а й інформація про усі попередні точки;
- фіксуються глобальні параметри точок на поверхні відгуку та параметри градієнтів у цих точках, що дає змогу швидко наблизитись до точки оптимуму, проте поблизу оптимуму необхідно виконати велику кількість ітерацій, які дозволять досягти його.

Залежно від типу імітаційної моделі (неперервна або дискретна) використовують методи оптимізації цільової функції відносно множини неперервних або дискретних параметрів, які мають певні обмеження. Майже в усіх моделях очікуваний результат можна виразити через експлуатаційні параметри системи.

Розглянемо систему, критерій ефективності якої $Q(\bar{X})$ залежить від неперервного параметра $v \in V$, де V – область можливих значень, і визначається як

$$Q(v) = \mathrm{M}_{Y|v}[G(\bar{X})],$$

де $\mathrm{M}_{Y|v}$ – математичне сподівання за умови, що випадковий вектор $\bar{X}$ з відомою функцією щільності імовірності $f(x; v)$ залежить від значення параметра v і критерію якості роботи.

У дискретних системах для оцінювання значення $Q(v)$ необхідно припустити, що $v = v_0$. Тоді за законом великих чисел

$$Q(v_0) = \frac{1}{n}\sum_{i=1}^{n} G(x_i)$$

сходиться до дійсного значення, де x_i – незалежні, однаково розподілені випадкові реалізації вектора $\bar{X}$ з відомою функцією $f(x; v_0)$; i – номер незалежного відгуку, $i = 1, 2, ..., n$.

Метою дослідження є оптимізація критерію $Q(v)$ відносно параметра v.

Вибір того або іншого з класичних методів оптимізації і можливість його застосування під час імітаційного моделювання залежить від постановки задачі моделювання.

8.4. Використання методів оптимізації під час проектування

Широке впровадження технологій імітаційного моделювання у процеси проектування складних систем неможливе без використання методів оптимізації. Методи імітаційного моделювання дозволяють побудувати імітаційну модель системи з будь-яким ступенем деталізації, але пошук оптимальних значень параметрів системи потребує застосування процедур оптимізації. Більш того, як було описано вище, існують засоби автоматизованого проектування імітаційних моделей, що значно полегшує їх створення. Використання таких засобів звільняє проектувальника від роботи, пов'язаної з побудовою імітаційної моделі, й націлює його на пошук оптимальної комбінації параметрів. У цьому разі проектувальник має змогу розв'язати надзвичайно складні задачі проектування. Однак результати, отримані за допомогою математичного моделювання і методів оптимізації, у більшості випадків негнучкі, і їх важко впровадити у реальному житті. Проблеми виникають переважно внаслідок того, що під час постановки та вирішення задач оптимізації за допомогою традиційного (детермінованого) підходу, як правило, не враховуються різні невизначеності, що впливає на ефективність розробленої системи в реальних умовах.

Значення екстремуму функції ефективності, отримане шляхом вирішення завдань оптимізації з детермінованими параметрами, є досяжним максимальним значенням, і його можна розглядати як звичайний оптимум з погляду його практичного використання. Цей оптимум може значно відрізнятися від тих значень, що отримані у реальних умовах, через помилки і випадковості, які не враховуються під час розв'язання задач оптимізації з детермінованими параметрами. Альтернатива детермінованому підходу – стохастична оптимізація.

Перед тим як вирішувати те або інше практичне завдання оптимізації, аналітик перш за все повинен з'ясувати:

- з яким типом невизначеності йому доведеться зіткнутися та яким чином це може вплинути на вибір оптимального рішення;
- чи можливо в рамках прийнятої моделі урахувати стохастичний характер досліджуваної ситуації.

Без правильної математичної формалізації проблеми проекту неможливо створити ефективно функціонуючу складну технічну систему. Під час створення системи проектувальники повинні сформувати вектор значень її ефективності $\vec{Y} = (y_1, y_2, ..., y_m)$, який має бути максимізованим або мінімізованим. Цей вектор залежить від вектора змінних параметрів $\vec{X} = (x_1, x_2, ..., x_n)$, що зумовлюють зміну значень ефективності. Крім того, необхідно врахувати вектор зовнішніх умов $\vec{Z} = (z_1, z_2, ..., z_k)$. Взаємозв'язок цих векторів $\vec{Y} = f(\vec{X}, \vec{Z})$ визначає математичну модель, яка використовується під час дослідження. Існування математичної моделі дає змогу формулювати проблему проектування як завдання оптимізації пошуку одного або кількох векторів $\vec{X}^* \in D$, що гарантують найвищу ефективність. У цьому разі

$$D = \{\vec{X} \in R^n | x_{j-} \le x_j \le x_{j+};\ \ j = \overline{1,N};\ \ g_i(\vec{X}, \vec{Z}),\ \ i = \overline{1,w}\},$$

де $\vec{X}$ – область пошуку, $g_i(\vec{X}, \vec{Z})$ – значення умовної ефективності. Таке «ідеальне» формулювання проблеми проектування розцінювалось донедавна як необхідна і достатня умова для того, щоб створити оптимальний проект. У разі використання такого підходу під час вирішення задач реального життя виникають серйозні проблеми, пов'язані з неможливістю пошуку оптимальних проектних рішень. Основна причина цього – велика кількість невизначеностей, які не можна врахувати під час моделювання системи, формулювання проблеми оптимізації та її вирішення.

Для включення невизначеності до математичної моделі необхідно задавати вектори: $\vec{X} = x(\bar{x}, \xi_x)$; $\vec{Z} = z(\bar{z}, \xi_z)$; $f(X, Z) = \Psi(\bar{f}(\vec{X}, \vec{Z}), \xi_f(\vec{X}, \vec{Z}))$, де $\bar{x}$, $\bar{z}$, $\bar{f}$ – ідеальні вектори відповідно параметрів, зовнішніх умов і математичної моделі; $\vec{\xi} = (\xi_x, \xi_z, \xi_f)$ – вектори випадкових значень, які включають невизначеності відповідно параметрів, умов і математичної моделі. Тобто, щоб вирішити проблему оптимізації, необхідно визначити системне значення ефективності $\vec{Y} = (\vec{X}, \vec{Z})$ для заданих значень $\bar{x}$, $\bar{z}$ і мати інформацію щодо законів розподілів компонент вектора $\vec{\xi}$, а також функціональної залежності $\Psi(\bar{f}, \xi_f)$.

Якщо під час моделювання виробничих процесів необхідно знайти максимум середнього значення ефективності або мінімум відхилення ефективності від середнього, то перед дослідником постає задача оптимізації параметрів. У цьому випадку для визначення законів розподілу компонент вектора ξ_x аналізують технологічний процес і роблять припущення щодо виду функцій розподілу або використовують статистичні дані, накопичені під час спостережень аналогічних технологічних процесів. Проте у більшості випадків приймається гіпотеза, що розподіл компонент вектора ξ_x нормальний або близький до нормального.

Проблема включення невизначеності умов середовища звичайно формулюється як задача адаптації моделі, коли функціонування системи, що проектується, коригується відповідно до змін у зовнішньому середовищі. У цьому випадку вирішується задача оптимізації системи, щоб мінімізувати вплив, зумовлений невизначеністю умов зовнішнього середовища. У даній ситуації потрібно знати закони розподілів компонент ξ_x, які здебільшого є результатом інтегрування експериментальних даних. Слід зауважити, що для розв'язання задачі, яка передбачає невизначеність параметрів і умов зовнішнього середовища, можна застосовувати детерміновану математичну модель $\bar{Y} = \bar{f}(\bar{x}, \bar{e}, \xi_x, \xi_e)$.

8.5. Прийняття рішень щодо удосконалення системи

Під час одного прогону імітаційної моделі неможливо визначити оптимальні значення параметрів системи або прийняти рішення відносно оптимізації її структури. Процедура пошуку оптимальних рішень щодо удосконалення модельованої системи завжди є ітераційною та циклічною і реалізується на різних етапах імітаційного моделювання. Роботи, що виконуються в рамках основного циклу, можна розбити на два етапи: діагностика і коригування моделі.

На етапі діагностики визначається, чи має система недолік та чи можна його локалізувати або усунути. Типова діагностична процедура включає формулювання гіпотези, проведення попереднього аналізу та перевірку гіпотези. Якщо результати аналізу на раціональність і результати наступних перевірок виявляться негативними, то варто сформулювати іншу гіпотезу і повторювати цю процедуру аж поки не буде отримане позитивне рішення або не буде перевірено всі гіпотези. В останньому випадку можна зробити висновок, що систему вдосконалити неможливо. Алгоритм вибору найкращого рішення, який використовує методи перевірки гіпотез, наведено на рис. 8.6.

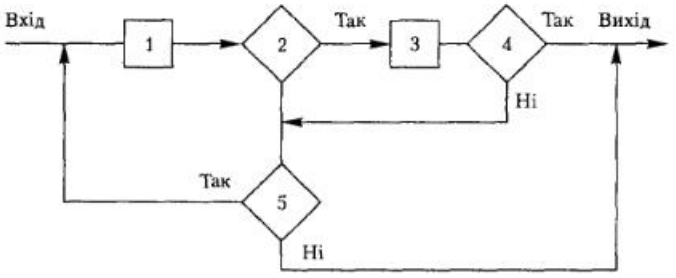


Рис. 8.6. Алгоритм вибору найкращого рішення
1 — формулювання гіпотези; 2 — перевірка, чи припускає гіпотеза раціональні зміни; 3 — перевірка гіпотези; 4 — перевірка, чи є задовільними результати перевірки гіпотези; 5 — перевірка на наявність іншої гіпотези

У своїй праці Д. Феррарі [64] пише: «...формулювання гіпотези... — це мистецтво, й основою успіху є міцний фундамент знань, розуміння та досвід, а також інтуїція». Перед тим як гіпотезу буде сформульовано, потрібно отримати необхідну інформацію про систему, визначити, які показники ефективності системи не такі, що очікувались, і спробувати визначити причини цієї невідповідності. Слід встановити наявність певної проблеми, локалізувати її та виявити причини її виникнення. Для цього формулюють кілька гіпотез, які потім необхідно перевірити. Єдиного підходу до формулювання гіпотез не існує, оскільки ця процедура суттєво залежить від типу моделі та складності проблеми, яка повинна бути вирішена за допомогою цієї моделі. Тому можна лише запропонувати деякі загальні методи формулювання гіпотез [64].

1. Ідентифікація схожих ситуацій. Використовується існуючий досвід проведення подібних робіт і формулюються гіпотези, подібні до відомих.

2. Виявлення значень, які істотно відрізняються від інших. Такі значення часто відповідають правильним гіпотезам. Їх можна ігнорувати тільки після ретельного вивчення.
3. Виявлення закономірностей. У моделі визначають цікаві закономірності в часі, такі як цикли або тенденції. Для цього доцільно застосовувати графічні методи – побудова діаграми станів системи, часових рядів та діаграм Ганта.
4. Виявлення кореляційних зв'язків. Наявність кореляції між параметрами та показниками критерію ефективності системи може сприяти висуненню правильних гіпотез.
5. Виявлення та аналіз невідповідностей. Існування очевидних взаємозв'язків між даними, взятими з різних джерел, між даними, пов'язаними із системою, а також між даними, отриманими під час моделювання, та тими, що очікуються. Виявлені невідповідності з великою ймовірністю дають змогу висунути справедливі гіпотези.

Перш ніж перевіряти гіпотези на моделі, необхідно перевірити їх на раціональність і можливість їх реалізації. Якщо втілення гіпотези економічно не виправдане або гіпотезу технічно не можна реалізувати, то її необхідно відкинути ще на попередньому етапі перевірки. Гіпотези, які залишились, перевіряють за допомогою імітаційної моделі.

Якщо гіпотеза, що перевіряється, виявляється справедливою, переходять до етапу коригування моделі. Серед усіх гіпотез вибирають одну. Методи коригування можна розглядати на різноманітних рівнях подання моделі, починаючи від рівня вхідних даних, алгоритмів поведінки і закінчуючи рівнем зміни конфігурації системи. Коригування необхідно починати з найпростіших змін у моделі, переходячи до більш складних. Важливо визначити чутливість моделі до вхідних даних і виявити блоки, на які впливають ці дані. Такі блоки слід описати більш детально. Лише після цього можна коригувати алгоритми поведінки та структуру моделі. Після внесення змін може виникнути потреба у нових вхідних даних. Крім того, необхідно обчислити економічну доцільність внесення змін у модель.

Задача аналізу системи за допомогою імітаційної моделі зводиться здебільшого до виявлення у ній вузьких місць. Нехай показник ефективності $Q(v_1, v_2, ..., v_i, ..., v_n)$ системи S залежить від n параметрів $v_1, v_2, ..., v_i, ..., v_n$, які можна змінювати під час моделювання. Величини v_i – це параметри навколишнього середовища та внутрішні параметри моделі. Припустимо, що збільшення значень параметрів приводить до зростання ефективності функціонування системи. Функція $Q(v_1, ..., v_n)$, $i = 1, ..., n$ здебільшого характеризується явно вираженими нелінійностями, що стають вузькими місцями.

Вузьким місцем системи S є значення $Q(v_1, ..., v_n)$ у певній області D простору параметрів, якщо у даній області це значення істотно зростає внаслідок зміни одного або кількох параметрів. При цьому навіть великі коливання значень інших параметрів не приводять до відчутної зміни значення $Q(v_1, ..., v_n)$, якщо система не виходить за межі області D. У такому випадку аналіз поведінки системи в областях навколо глобального або локальних максимумів функції $Q(v_1, ..., v_n)$ не проводиться. Вузькі місця виникають тоді, коли до деяких ресурсів системи створюються великі черги через порушення балансу між потоком запитів до цього ресурсу та його можливостями задовольняти їх.

Ознакою наявності вузького місця в системі може бути велика різниця між коефіцієнтами використання компонентів системи, особливо коли один із них прямує до одиниці. Іншою важливою ознакою наявності вузького місця є з'ясування того факту, що продуктивність системи явно нижча за очікувану або можливу. Під час формулювання і перевірки гіпотез про наявність вузького місця доцільно використовувати графічні методи подання результатів моделювання за допомогою діаграм Ганта та графіків Ківіата. Наприклад, діаграми Ганта добре ілюструють часові співвідношення між інтервалами часу, коли компоненти системи зайняті і коли вони вільні. Таку інформацію важко одержати лише на основі коефіцієнтів використання окремих компонентів.

Один з підходів до поліпшення показників ефективності значень $Q(v_1, ..., v_n)$ систем пов'язаний з послідовним усуненням вузьких місць. У загальному випадку зміна одного або кількох параметрів v_i моделі не завжди приводить до балансування системи, оскільки можуть виникнути нові вузькі місця, які до цього були приховані щойно усуненими. Крім того, зміни, внесені в модель, можуть спричинити перехід системи S в іншу область простору параметрів, в якій значення критерію $Q(v_1, ..., v_n)$ буде обмежене через наявність іншого вузького місця. У цьому разі важко безпосередньо використовувати градієнтні методи пошуку максимуму функції $Q(v_1, ..., v_n)$, оскільки у наборі параметрів можуть бути присутні також якісні фактори, наприклад стратегія керування ресурсом. Якщо показник ефективності $Q(v_1, ..., v_n)$ не залежить від якісних факторів, застосовують методи екстремального планування експериментів.

У тому випадку, коли модельована система відображається мережею СМО, для попереднього аналізу її роботи і пошуку вузьких місць використовують операційний аналіз (див. розділ 2.7). Покрокове усунення вузьких місць у системі за результатами операційного аналізу і балансування коефіцієнтів завантаження вузлів у мережі дає змогу отримати систему з мінімальним часом перебування в ній вимог, тобто систему з максимальною пропускною здатністю. У багатьох випадках це дає можливість знайти початкове наближення для визначення необхідної кількості пристроїв у вузлах мережі, яке потім уточнюється під час імітаційного моделювання. Таким чином вдається оптимізувати структуру системи. Слід зауважити, що досягти повної збалансованості системи не завжди вдається, до того ж це може бути економічно недоцільним. Найкраще за економічним критерієм рішення можна отримати завдяки використанню методів оптимізації або перебору варіантів.

Виявлення вузьких місць у системі та їх локалізація – найпростіші задачі, які можна розв'язати шляхом перевірки гіпотез. Зазвичай для ідентифікації причин появи вузького місця необхідно висувати та перевіряти більш складні гіпотези. Припустимо, що було виявлено деякий ресурс системи, який є її вузьким місцем. Це означає, що існує невідповідність між інтенсивністю запитів до цього ресурсу і продуктивністю обслуговування ним запитів. Однак це не є свідченням того, що продуктивність даного ресурсу недостатня. Збільшення продуктивності може бути економічно недоцільним. У цьому випадку необхідно розглянути гіпотезу про встановлення буферу (накопичувача) перед ресурсом, що дозволить усунути нерівномірність запитів до цього ресурсу. Технічна реалізація такої гіпотези може бути економічно найбільш виправданою.

Серед інших гіпотез, що допомагають вирішити проблеми появи вузьких місць – перенаправлення запитів до іншого аналогічного ресурсу, зміна алгоритму планування обслуговування ресурсу або режимів його роботи.

8.6. Порівняння альтернативних варіантів системи

Однією із задач, які доводиться розв'язувати під час прийняття рішень є порівняння альтернативних варіантів системи, режимів її роботи, стратегій розвитку тощо. Подібна задача розглядалась під час порівняння результатів, отриманих в процесі моделювання імітаційних моделей та реальних систем (див. розділ 5.10). У даному розділі ми з'ясуємо, наскільки надійний і однозначний результат можна отримати під час порівняння альтернативних варіантів системи.

Розглянемо два альтернативних варіанти системи – A і B. Метою дослідження є вибір кращого варіанту (відповідно до заданого критерію) і створення можливості подання результатів вибору в чисельному вигляді (наприклад, варіант A кращий за варіант B на 30 %). Припустимо, що вибрана множина параметрів V системи, які адекватно характеризують обидва варіанти. Будемо вважати також, що від багатьох параметрів V залежить значення деякого критерію F (наприклад, економічного), що показує, який із двох варіантів системи кращий і наскільки (якщо множина V містить лише один параметр, його можна прийняти за критерій). Таким чином, за ідеальних умов моделювання на основі порівняння критеріїв F_A і F_B одержимо розв'язок задачі.

Під час порівняння двох варіантів системи у процесі моделювання слід дотримуватись певних вимог, невиконання яких може негативно вплинути на точність і надійність оцінювання значень критеріїв F_A і F_B. По-перше, порівнювати альтернативні варіанти потрібно за однакових зовнішніх умов, тобто за наявності однакових вхідних змінних і послідовностей потоків випадкових чисел. По-друге, всі фактори, від яких залежить значення критерію F (крім тих, що характеризують обидва варіанти), необхідно змінювати так, щоб їх вплив на різницю $(F_A - F_B)$ був мінімальний; порівняння варіантів системи має провадитись для всієї області можливих значень параметрів V. Інші проблеми, які можуть виникнути в цьому випадку, було описано в розділі 5.10.

Ще одним джерелом помилок, що можуть виникнути під час моделювання, є статистична природа більшості параметрів V системи, від яких залежить критерій F. Аналітика звичайно цікавить числове значення статистичних характеристик випадкових величин, наприклад середнє значення показників, але під час моделювання отримати можна лише оцінки цих значень. Неправильне рішення може також бути прийняте через недостовірність і неефективність статистичних оцінок. Таким чином, під час порівняння двох альтернативних варіантів системи існує ймовірність прийняття неправильних рішень.

Припустимо, що F – це показник ефективності системи, який характеризує середній час $\bar{t}$ виконання робіт системою. Будемо вважати, що ефективність системи A вища, ніж ефективність системи B, якщо $\bar{t}_A < \bar{t}_B$. При цьому оцінки серед-

нього часу виконання робіт обчислюються для однакових вхідних даних і різних послідовностей випадкових чисел. При порівнянні цих оцінок можна отримати такі результати:

- однакові якісні результати: якщо $\bar{t}_A > \bar{t}_B$, то $\hat{\bar{t}}_A > \hat{\bar{t}}_B$; якщо $\bar{t}_A < \bar{t}_B$, то $\hat{\bar{t}}_A < \hat{\bar{t}}_B$; якщо $\bar{t}_A \approx \bar{t}_B$; то $\hat{\bar{t}}_A \approx \hat{\bar{t}}_B$;
- близькі кількісні результати: оцінка приросту ефективності однієї системи у відсотках стосовно іншої не повинна істотно відрізнятися від дійсного підвищення.

Необхідно зазначити, що у даному випадку, порівнюючи дві системи, ми фактично оцінюємо різницю $d_{AB} = \bar{t}_A - \bar{t}_B$ за різницею оцінок $\hat{d}_{AB} = \hat{\bar{t}}_A - \hat{\bar{t}}_B$, яка обчислюється на основі вибіркових середніх. Тому задачу порівняння двох варіантів системи можна сформулювати як задачу математичної статистики таким чином: обчислити довірчий інтервал для величини d_{AB} при заданому рівні довіри α, або, іншими словами, обчислити імовірність потрапляння значення d_{AB} за межі заданого довірчого інтервалу. За результатами моделювання можна визначити розподіл оцінок $\hat{\bar{t}}_A$ і $\hat{\bar{t}}_B$. Якщо $\hat{\bar{t}}_A$ і $\hat{\bar{t}}_B$ незалежні та нормально розподілені величини, їх різниця також має нормальний розподіл. Отже, величина довірчого інтервалу для d_{AB} задається виразом

$$\hat{d}_{AB} - z_\alpha \sigma_{AB} \le d_{AB} \le \hat{d}_{AB} + z_\alpha \sigma_{AB}, \qquad (8.1)$$

де z визначається за таблицями стандартного нормального розподілу з довірчою ймовірністю α, а σ_{AB} – стандартне відхилення $\hat{d}_{AB}$ відносно d_{AB}. Враховуючи те, що дисперсії σ_A^2 і σ_B^2 невідомі, їх необхідно обчислити за результатами n прогонів імітаційної моделі (див. розділ 4.8).

Припустимо, що кількість випробувань $n_A = n_B = n$ і $\hat{\sigma}_{AB}^2 = \hat{\sigma}_A^2 + \hat{\sigma}_B^2$. Оскільки випадкова величина $(\hat{d}_{AB} - d_{AB})/\hat{\sigma}_{AB}$ має розподіл Стьюдента (t-розподіл), то ширина довірчого інтервалу (8.1) обчислюється за формулою:

$$\hat{d}_{AB} - t_\alpha \sigma_{AB} \le d_{AB} \le \hat{d}_{AB} + t_\alpha \sigma_{AB}, \qquad (8.2)$$

де t_α знаходять за таблицями t-розподілу з $n - 1$ ступенями вільності.

У даному випадку зроблено суттєві припущення. Якщо $n_A \ne n_B$, використовувати t-розподіл Стьюдента неможливо, тому що умова $\sigma_A = \sigma_B$ не задовольняється. Припущення щодо нормальності розподілу відхилити важче, але якщо вважати, що $\hat{d}_{AB}$ має скінченне середнє та скінченну дисперсію, то застосування нерівності Чебишева дає ширину довірчого інтервалу

$$\hat{d}_{AB} - \sigma_{AB}/\sqrt{1-\alpha} \le d_{AB} \le \hat{d}_{AB} + z_\alpha\, \sigma_{AB}/\sqrt{1-\alpha}.$$

Якщо припущення щодо нормальності розподілу малоймовірне, можна використовувати непараметричні методи, наприклад рангові тести [18]. Слід зауважити, розподіл оцінки $\hat{d}_{AB}$ ближче до нормального, ніж розподіли оцінок $\hat{\bar{t}}_A$ і $\hat{\bar{t}}_B$, тоді, коли розподіли $\hat{\bar{t}}_A$ і $\hat{\bar{t}}_B$ мають асиметрію одного знаку.

Якщо ширину довірчого інтервалу для d_{AB} оцінено, порівняння його ширини зі значенням $\hat{d}_{AB}$ дає корисну інформацію про можливий знак d_{AB} і про його значущість. У тому випадку, коли довірчий інтервал для відповідного рівня α містить значення нуль, d_{AB} може знаходитись з будь-якого боку початку координат з деякою ймовірністю; у цьому інтервалі обидві системи мають однакову продуктивність. Таким чином, відхилити гіпотезу, що $d_{AB} = 0$ за даного рівня значущості $1 - \alpha$ не можна, і різниця між значеннями продуктивності двох систем статистично не значуща. У разі зниження рівня значущості оцінка стає більш точною, а тести — більш чутливими, але для їх використання необхідні вибірки значно більшого розміру.

Якщо під час порівняння альтернативних варіантів двох імітаційних моделей застосовуються однакові вхідні дані та однакові послідовності потоків випадкових величин (тобто застосовуються загальні випадкові числа (див. розділ 7.3.3)), то пари значень t_{Ai} і t_{Bi}, що характеризують тривалість деякої роботи i, є залежними. У цьому випадку дисперсія $\hat{d}_{AB}$ вже не дорівнює сумі дисперсій $\hat{\bar{t}}_A$ і $\hat{\bar{t}}_B$, як це було за умови їх незалежності, тобто оцінки дисперсії необхідно розраховувати за формулою для повної дисперсії

$$\hat{\sigma}^2_{AB} = \hat{\sigma}^2_A + \hat{\sigma}^2_B - 2\,\mathrm{cov}\left(\hat{\bar{t}}_A, \hat{\bar{t}}_B\right).$$

Об'єднання в пари значень t_{Ai} і t_{Bi} робить $\mathrm{cov}(\hat{\bar{t}}_A, \hat{\bar{t}}_B)$ позитивною, що зменшує σ_{AB} і збільшує точність порівняння вибірок однакового розміру на відміну від незалежних спостережень. Якщо припустити, що різниця $d_{ABi} - d_{AB}$, де $d_{AB} = t_{Ai} - t_{Bi}$, незалежні та розподілені нормально величини з нульовим середнім і з дисперсією σ^2_d, то значення $\hat{d}_{AB}$ розподілені нормально біля d_{AB} з дисперсією σ^2_d/n, а σ^2_d можна оцінити за формулою

$$\hat{\sigma}^2_d = \frac{1}{n-1}\left(\sum_{i=1}^{n} d^2_{ABi} - \frac{1}{n(n-1)}\left(\sum_{i=1}^{n} d_{AB}\right)^2\right).$$

Оскільки змінна $(\hat{d}_{AB} - d_{AB})/(\hat{\sigma}_d/\sqrt{n})$ має розподіл Стьюдента, то довірчий інтервал для d_{AB} при довірчому рівні α задається формулою (8.2), де оцінку для $\hat{\sigma}_{AB}$ необхідно замінити на $\hat{\sigma}_d\sqrt{n}$.

У випадку порівняння альтернативних варіантів системи необхідно мати на увазі, що причиною прийняття помилкового рішення може бути зміщення статистичних оцінок, внесене в результат через похибки методів оцінювання, що виникають внаслідок неточності імітаційних моделей. Однак вплив будь-якої систематичної похибки на значення показника критерію ефективності обох варіантів системи, як правило, є однаковим і результат порівняння від нього не залежить. Таким чином, під час порівняння альтернативних варіантів кількох систем більше уваги слід приділяти однорідності вхідної інформації, ніж методам підвищення точності оцінювання критерію F для кожної системи.

Крім розглянутого в цьому розділі сукупного внеску джерел дисперсії в $\hat{d}_{AB}$, внесок кожного із джерел дисперсії показників можна обчислити за методами дисперсійного аналізу ANOVA у процесі планування експериментів.

8.7. Приклади прийняття рішень за допомогою імітаційного моделювання

У цьому розділі розглядаються дві моделі виробничих систем і приклади процедур прийняття рішень за результатами моделювання з використанням методів операційного аналізу і перевірки гіпотез.

8.7.1. Моделювання технологічного процесу ремонту та заміни обладнання

Мета дослідження

Необхідно визначити найкращий варіант технологічного процесу ремонту та заміни обладнання для забезпечення мінімальної собівартості виробництва за H робочих днів. Зверніть увагу на те, що оцінки параметрів необхідно обчислити для стаціонарного режиму.

Постановка задачі

Розглянемо технологічний процес ремонту та заміни обладнання. Припустимо, що деяка виробнича дільниця має L верстатів, які працюють цілодобово (24 години на добу). Всього в технологічному процесі задіяно M верстатів, що більше або дорівнює L (причому L – власні верстати, а решту орендують для резерву). Будь-який з верстатів може вийти з ладу в будь-який час. Якщо верстат вийшов з ладу, його замінюють іншим, резервним, а зламаний направляють в майстерню для ремонту. Відремонтований верстат повертається вже як резервний.

Для ремонту верстатів у майстерні є три спеціалізовані дільниці. Технологічний цикл ремонту починається на дільниці діагностики, де визначаються причина виходу з ладу обладнання та необхідний вид ремонту. Ремонт виконується на механічних і електронних дільницях. Статистичні дані аналізу виходу верстатів з ладу свідчать, що у 75 % випадків ремонту потребує електронне обладнання верстатів, а у 25 % – механічне. Діагностикою зайнято m_1 робітників, ремонтом механічного обладнання – m_2, а ремонтом електронного – m_3 робітників.

Заробітна плата робітників у ремонтній майстерні – W гривень за годину, плата за орендовані верстати – S гривень за добу. Погодинний збиток під час використання L верстатів у виробництві становить Q гривень за верстат. Збитки виникають внаслідок зменшення обсягів виробництва.

Практичний досвід експлуатації показує, що тривалість діагностики становить $A_1 \pm B_1$ годин (закон розподілу тривалості діагностики – рівномірний), тривалість ремонту електронного обладнання верстата – $A_2 \pm B_2$ годин (розподіл рівномірний), а механічного – $A_3 \pm B_3$ годин (розподіл також рівномірний). Якщо верстат використовується у виробництві, час напрацювання на відмову має експоненціальний розподіл з параметром T годин. Час, витрачений на перевезення верстатів із цеха в майстерню та у зворотному напрямку, незначний, і його не враховують. Додатковою умовою, яка спрощує постановку задачі, є те, що всі робітники в майстерні, як і верстати, взаємозамінні.

Дані для моделювання наведені в табл. 8.1.

Таблиця 8.1. Дані для моделювання технологічного процесу ремонту та заміни обладнання

L	T	$A_1 \pm B_2$	$A_2 \pm B_2$	$A_3 \pm B_3$	H	W	S	Q
50	160	2±1	30±10	45±5	360	7,75	650	120

Плата за оренду верстатів не залежить від того, використовують їх чи ні. Керівнику потрібно визначити, скільки робітників має працювати в майстерні і скільки верстатів має бути орендовано, тобто скільки верстатів треба мати в резерві для заміни тих, що вийшли з ладу.

Повний опис моделі та принципи її побудови наведено в літературі [61, 62].

Пошук найкращого рішення

Мета дослідження – мінімізація вартості виробництва, що потребує визначення оптимального варіанту технологічного процесу ремонту та заміни обладнання. Для його пошуку необхідно виконати дії у такій послідовності.

1. Розрахувати середній час ремонту верстатів R за допомогою методів операційного аналізу мереж СМО (див. розділ 2.7.2).
2. Сформулювати гіпотезу щодо потенційного вузького місця системи і визначити його.
3. Описати стратегію пошуку розв'язку завдання, визначити необхідну кількість орендованих верстатів і число ремонтників для проведення моделювання.
4. Розробити програму проведення експериментів, попередньо визначивши кількість прогонів моделі для кожної комбінації «кількість робітників–кількість верстатів» із записом вартості витрат у файл результату.
5. Використовуючи дисперсійний аналіз ANOVA, провести аналіз результатів моделювання і зробити висновки щодо найкращого варіанта технологічного процесу ремонту та заміни обладнання.

Для пошуку найкращих рішень скористаємось методом структурної оптимізації [62, 63]. Модель використовується для оцінки комбінації «кількість робітників–кількість орендованих верстатів», які мінімізували б щоденні середні витрати на виробництво.

Якщо кількість робітників фіксована, середні денні витрати будуть змінюватись залежно від кількості орендованих верстатів. Залежність витрат має вигляд увігнутої донизу кривої. Аналогічно, при заданій кількості орендованих верстатів крива впливу кількості найнятих робітників на денні витрати має той же вигляд. Якщо зобразити графік у тримірному просторі «кількість орендованих верстатів–кількість робітників–денні витрати», можна припустити, що цей графік буде також увігнутим донизу і матиме одну точку мінімуму, яку можна знайти, перебираючи комбінації «кількість найнятих робітників–кількість орендованих верстатів».

Розглянемо «ідеальну» систему, в якій верстат, що вийшов з ладу, відразу потрапляє на ремонтну дільницю. Знайдемо нижню оцінку кількості ремонтників, необхідних для проведення ремонту. Функціонування кожного верстату складається з двох фаз: фази роботи із середньою тривалістю T годин і фази ремонту із середньою тривалістю R годин.

Мінімальний середній час ремонту верстатів можна визначити за допомогою методів операційного аналізу, використовуючи коефіцієнти відвідування вузлів вимогами V_j, яка є моделлю майстерні (рис. 8.7), та середню тривалість R ремонту верстата на кожному робочому місці.

$$R = \sum_j V_j R_j, \quad j = 1, 2, 3.$$

Згідно з формулою (2.7) $V_1 = 1$; $V_2 = q_{12} = 0{,}25$; $V_3 = 0{,}75$.

Рис. 8.7. Мережа СМО для виробничої дільниці

Пропускна здатність майстерні визначається вузьким місцем, тобто дільницею, коефіцієнт завантаження якої U_j наближається до одиниці. Потенційне вузьке місце визначають за формулою

$$V_d R_d = \max\{V_1 R_1, V_2 R_2, V_3 R_3\}.$$

Задача полягає у збалансуванні системи, тобто досягненні такого завантаження, при якому середній час ремонту буде приблизно однаковий, оскільки тривалість ремонту верстатів повністю визначається вузьким місцем. Це можна зробити шляхом збільшення кількості ремонтників на тих робочих місцях, для яких $i \neq i_0$, де i_0 – номер найменш завантаженого робочого місця, що визначається за формулою

$$V_{i_0} R_{i_0} = \min\{V_1 R_1, V_2 R_2, V_3 R_3\}.$$

З умови приблизної рівності середніх значень тривалості ремонту випливає приблизна рівність

$$\frac{V_1 R_1}{m_1} \approx \frac{V_2 R_2}{m_2} \approx \frac{V_3 R_3}{m_3}, \tag{8.3}$$

де $m_{i0} = 1$, тобто кількість робітників на найменш завантаженому робочому місці приймається за одиницю.

Таким чином, пропускна здатність майстерні буде збалансованою, якщо коефіцієнти завантаження всіх робочих місць майстерні будуть приблизно однаковими. Тоді коефіцієнт завантаження всіх верстатів такої системи визначатиметься як

$$U_0 = \frac{T}{T+R}.$$

Щоб виключити витрати внаслідок зниження обсягу виробництва, загальна кількість верстатів, яка обертається в системі, повинна бути

$$M = \left\lfloor \frac{L}{U_0} \right\rfloor,$$

де $\lfloor x \rfloor$ означає операцію округлення x до найближчого цілого з надлишком.

Припустимо, що кількість резервних верстатів у системі

$$L_r = M - L. \tag{8.4}$$

Однак, беручи до уваги те, що час відмови і тривалість ремонту є випадковими величинами, деякі верстати, що вийшли з ладу, можуть простоювати у черзі, чекаючи ремонту. Таким чином, враховуючи заданий витратний критерій, необхідно орендувати більш ніж L_r верстатів.

Найкращу комбінацію «кількість орендованих верстатів–кількість робітників» можна знайти таким чином.

1. Вважаємо, що елементи комбінації «кількість орендованих верстатів–кількість робітників» визначаються за допомогою виразів (8.3) і (8.4).
2. Проводимо серію експериментів для всіх допустимих комбінацій «кількість орендованих верстатів–кількість робітників». Після кожного прогону моделі зберігаємо значення коефіцієнтів завантаження U_j ($j = 0, 1, 2, 3$) і значення витрат на виробництво в матриці результатів.
3. Результати моделювання, отримані в серії прогонів після виконання команд ANOVA, заносимо в табл. 8.2. Близькість значення коефіцієнта завантаження верстатів U_0 до одиниці свідчить про те, що верстати швидко повертаються на виробничу дільницю, тобто не простоюють. Це дозволяє зробити висновок, що кількість відремонтованих і (або) орендованих верстатів є достатньою.

Таблиця 8.2. Результати моделювання виробничої системи

Комбінації	Кількість ремонтників			Кількість верстатів L_r	Середні значення коефіцієнтів завантаження				Середнє значення вартості виробництва
	m_1	m_2	m_3		U_0	U_1	U_2	U_3	

4. За результатами моделювання визначаємо вузьке місце системи. Висуваємо гіпотезу про збільшення кількості ремонтників у цьому вузлі (збільшення може перевищувати одиницю). Якщо ж вузьких місць немає, збільшується кількість

верстатів L_r і для перевірки нової комбінації «кількість орендованих верстатів–кількість робітників» повернемося до пункту 2 цього алгоритму.

5. Моделювання припиняємо, якщо при зміні значень L_r, m_1, m_2, m_3 не зменшується середня вартість виробництва. Комбінація «кількість орендованих верстатів–кількість робітників», якій відповідає найменше середнє значення витрат на виробництво, є розв'язком задачі.

За цим алгоритмом не завжди доцільно повністю збалансовувати завантаження робочих місць, тому що заробітна плата ремонтників під час їх простою може бути незначною порівняно зі збитками виробництва.

За результатами дисперсійного аналізу ANOVA робимо висновок про значущість розходжень отриманих результатів, наводимо значення критерію Фішера для найкращого розв'язку, аналізуємо побудований довірчий інтервал.

Якщо отримано кілька значень, близьких до оптимального значення функції витрат на виробництво, і ця функція є спадною, необхідно збільшити кількість прогонів у серіях відповідних комбінацій «кількість орендованих верстатів–кількість робітників», а потім для них провести дисперсійний аналіз ANOVA.

Наведений алгоритм можна реалізувати за допомогою системи GPSS World.

8.7.2. Моделювання виробничої дільниці

Мета дослідження

Метою моделювання роботи виробничої дільниці є визначення найкращих керуючих дій, запроваджуваних з метою вдосконалення технологічної дільниці за критерієм збільшення доходу, отриманого в результаті виконання робіт.

Постановка задачі

Процедуру прийняття рішень розглянемо також на прикладі моделі виробничої дільниці [60] із заданими маршрутами руху деталей (рис. 8.8). Цей приклад аналогічний прикладу, описаному в праці Т. Шрайбера [68], за винятком процедур прийняття рішень і включення в технологічний процес конвеєра для передавання деталей між верстатами.

Виробнича дільниця обладнана чотирма верстатами: токарним, свердлильним, шліфувальним та фрезерним. На дільниці обробляються деталі чотирьох типів. Деталі кожного типу потребують обробки на верстатах відповідного типу в певній послідовності, яка задається маршрутною картою.

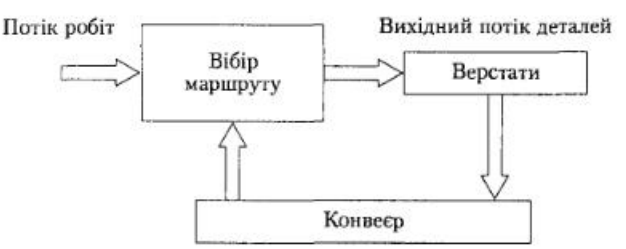


Рис. 8.8. Структурна схема концептуальної моделі

Кількість етапів обробки, послідовність проходження деталей по технологічному маршруту та середній час обробки деталей усіх типів наведено в маршрутній карті проходження деталей по дільниці (табл. 8.3). Верстати в маршрутній карті вказані в порядку виконання технологічних операцій.

Час надходження деталей на дільницю має розподіл Пуассона із середнім значенням 24 деталі за 8 годин роботи дільниці. Поява будь-якого типу роботи рівноймовірна та не залежить від робіт інших типів.

Характеристики технологічного процесу виготовлення однієї деталі кожного типу і отриманий прибуток наведено в табл. 8.4.

Таблиця 8.3. Технологічна карта процесу виготовлення деталей

Тип деталі	Кількість етапів обробки	Послідовність проходження деталей через верстати	Час обробки, хв
1	6	Токарний	8,8
		Фрезерний	12,0
		Свердлильний	12,0
		Шліфувальний	13,0
		Свердлильний	10,5
		Токарний	11,5
2	4	Фрезерний	20,0
		Шліфувальний	14,0
		Фрезерний	14,5
		Свердлильний	16,0
3	5	Токарний	17,6
		Свердлильний	19,0
		Фрезерний	14,0
		Токарний	11,6
		Шліфувальний	30,0
4	4	Свердлильний	19,0
		Токарний	16,8
		Фрезерний	13,0
		Шліфувальний	19,0

Таблиця 8.4. Характеристики виготовлення деталей

Тип деталі	Прибуток, грн.	Собівартість, грн.	Допустимий час виготовлення, хв	Штраф за затримку виготовлення понад допустимий термін, грн.
1	1550	350	1890	80
2	1850	420	1600	120
3	1350	280	2300	160
4	1450	315	1400	100

Повний опис імітаційної моделі та її програмну реалізацію мовою GPSS наведено в літературі [61, 62].

Пошук найкращого рішення

Процедура визначення найкращих рішень щодо керування технологічною дільницею та її удосконалення є ітераційною і полягає у внесенні змін у технологію обробки деталей на дільниці. З цією метою всі роботи можна поділити на такі етапи:

- виявлення причин зниження продуктивності роботи дільниці та зменшення прибутку від виконаних робіт;
- висунення гіпотез і попередній аналіз їх правильності;
- перевірка гіпотез і порівняння отриманих результатів;
- видача рекомендацій щодо вдосконалення технологічної дільниці.

Порядок роботи з імітаційною моделлю такий:

- здійснення пробного прогону моделі та усунення помилок, якщо вони є (під час пробного прогону бажано зменшити час моделювання);
- виконання повного прогону моделі;
- аналіз результатів прогону та висунення гіпотези щодо вдосконалення технологічної дільниці.

Пропонуються такі робочі гіпотези.

Гіпотеза А. Перехід на більш продуктивні режими роботи устаткування (зменшення тривалості виконання робіт на верстатах). Такі зміни швидкісних режимів можуть призвести до зниження якості продукції, що в свою чергу зменшить прибуток, але він може зрости у разі збільшення загальної продуктивності роботи дільниці та скорочення незавершеного виробництва на кінець робочого дня. Крім того, можуть зменшитися штрафи, що накладаються через недотримання допустимих термінів виготовлення деталей, а завдяки переходу верстатів на швидкісний режим швидкість їх роботи можна збільшити на 20 %. Вихідні дані для перевірки гіпотези *А* – можливі зміни швидкостей роботи верстатів та величини прибутку – наведено в табл. 8.5.

Таблиця 8.5. Дані для перевірки гіпотези *А*

Збільшення швидкості обробки при застосуванні різної кількості верстатів	Зменшення вартості деталей по типах, %			
	Тип 1	Тип 2	Тип 3	Тип 4
1	1,5	1,2	2,8	2,0
2	2,5	1,5	3,0	2,8
3	3,0	2,0	3,5	3,2
4	3,5	2,8	3,8	3,6

Гіпотеза В. Збільшення кількості однотипних верстатів на дільниці. Такі зміни приводять до тих самих наслідків, що й для гіпотези *А*, однак у цьому разі якість виготовлених деталей не погіршиться. Водночас зменшиться прибуток через амортизаційні відрахування на нові верстати (табл. 8.6).

Таблиця 8.6. Дані для перевірки гіпотези *B*

Кількість однотипних верстатів	Збільшення собівартості деталі, виготовленої на верстаті певного типу, %			
	Свердлильний	Токарний	Фрезерний	Шліфувальний
2	5	8	7	10
3	10	16	14	20
4	15	24	21	30

Гіпотеза С. Зменшення сумарного штрафу. Якщо впорядкувати роботи за зменшенням відношення штрафу, що накладається через затримку виготовлення деталей понад допустимий термін, до часу обробки, то зменшиться сумарний штраф на дільниці. Для реалізації цієї гіпотези необхідно задати пріоритети обробки деталей відповідно до наведеного відношення.

Гіпотези *A*, *B*, *C* можна використовувати одночасно.

За допомогою імітаційної моделі необхідно перевірити гіпотези, вибрати найкращий варіант удосконалення роботи технологічної дільниці, описавши стратегію вибору цього варіанта, та обчислити прибуток.

За один прогін моделі неможливо визначити оптимальну структуру виробничої дільниці. Ця процедура неминуче виявляється ітеративною і вимагає утворення множини гіпотез та їхньої перевірки. Для кожної гіпотези слід провести кілька прогонів моделі, щоб одержати результати з потрібною точністю.

Перед проведенням експериментів множину гіпотез упорядковують за значенням збільшення матеріальних витрат на впровадження гіпотези. Для даного прикладу упорядкований список гіпотез такий: визначення початкової структури дільниці; введення нового режиму роботи устаткування; встановлення пріоритетів у чергах до верстатів усіх типів; встановлення пріоритетів і нового режиму роботи; введення нових додаткових верстатів. Найкращу гіпотезу слід вибирати з урахуванням того, що завантаженість устаткування не повинна перевищувати критичне значення — 87 %.

Якщо коефіцієнти завантаження верстатів перевищують критичне значення, потрібно встановити додаткові верстати цього типу. Завантаження введених верстатів на 60–70 % є умовою швидкої окупності.

Результат виконання даного завдання свідчить, що найкращою є гіпотеза переходу на новий режим роботи устаткування виробничої дільниці та визначення пріоритетів під час обробки деталей на верстатах. На рис. 8.9 зображено графіки зміни прибутку для початкової структури дільниці та для поліпшеної структури в разі моделювання роботи дільниці протягом 11 днів.

При порівнянні альтернативних варіантів побудови модельованої системи виникає питання щодо надійності та однозначності отриманих результатів. Установлено, що в цьому випадку слід дотримуватись таких вимог:

- порівнювати альтернативні варіанти потрібно за однакових зовнішніх умов, тобто в разі наявності однакових вхідних змінних і послідовностей потоків випадкових чисел;

- всі фактори, що впливають на критерій F, крім тих, які характеризують обидва варіанти (1 і 2), необхідно змінювати так, щоб їх вплив на різницю $(F_1 - F_2)$ був мінімальний.

Рис. 8.9. Графіки зміни прибутку

Висновки

- Основне призначення моделювання – підготовка результатів експериментів для прийняття рішень.
- Перш ніж результати моделювання буде представлено аналітику, їм потрібно надати наочного та інформативного вигляду.
- Методи оптимізації використовуються на кожній ітерації пошуку найкращих рішень тільки тоді, коли чітко визначена мета (цільова функція) оптимізації.
- Процедури удосконалення системи застосовуються, якщо є можливість поліпшення показників модельованої системи та якщо пошук цих показників виправданий.
- Більшість методів прийняття рішень пов'язані з висуванням гіпотез, їх перевіркою на моделях та визначенням найкращої гіпотези.

Контрольні запитання та завдання

1. Наведіть приклади, коли дані щодо завантаження обладнання модельованої виробничої дільниці, отримані зі стандартного звіту результатів моделювання (скажімо, мовою GPSS), будуть неінформативними. В яких випадках необхідно застосовувати діаграми Ганта та графіки Ківіата?
2. Що дає анімація імітаційної моделі для замовника та розробника. Чи завжди анімація необхідна?

3. Чому класичні методи оптимізації не можна застосовувати під час імітаційного моделювання? Які переваги надає метод висування та перевірки гіпотез для прийняття рішень?
4. Проведіть загальний огляд методів оптимізації та проаналізуйте доцільність їх використання для прийняття рішень за результатами моделювання.
5. Які проблеми виникають під час порівняння альтернативних варіантів реалізації системи за результатами моделювання. Як вирішують ці проблеми?
6. Скористайтеся програмою моделювання технологічного процесу ремонту та заміни обладнання (розділ 8.7.1), наведеною в книжці [61], і варіантами завдань для пошуку найкращих рішень.
7. Скористайтеся програмою моделювання виробничої дільниці (розділ 8.7.2), наведеною в книжці [61], і варіантами завдань для пошуку найкращих рішень. Під час виконання самостійної роботи з програмою виробничої дільниці дайте відповідь на такі запитання:
   Як треба змінити модель виробничої дільниці, щоб використати деталі інших типів з іншими заданими маршрутами руху?

   Як треба змінити модель виробничої дільниці, якщо використати верстат іншого типу, наприклад стругальний?

Розділ 9

Імітаційне моделювання виробничих та комп'ютерних систем

- Моделі виробничих систем
- Моделі процесів обслуговування вимог та розподілу ресурсів
- Моделі процесів управління проектами
- Моделі комп'ютерних систем та мереж

Більшість імітаційних моделей, в яких виникає потреба на практиці, – це моделі процесів різних видів: технологічних, логістичних, інформаційних, виробничих та ін. Процесний підхід є також одним із головних під час моделювання і проектування бізнес-процесів. Він дає змогу не тільки застосувати для створення і реалізації моделі сучасні програмні засоби, але і перевірити ефективність процесів; для цього використовуються імітаційні пакети, такі як Arena для BPWin.

За допомогою моделей інформаційних систем, як правило, досліджуються процеси перетворення, обробки та передачі інформації, які реалізуються комп'ютерними системами та системами передавання даних. У цьому розділі розглядаються особливості моделювання процесів, що протікають у виробничих та комп'ютерних системах різних типів.

9.1. Виробничі процеси

Виробничі процеси – це процеси, результатом яких є випуск різноманітної продукції. Слід розрізняти два основних типи виробничих процесів – неперервні та дискретні. Імітаційні моделі неперервних процесів подаються здебільшого у вигляді систем лінійних, нелінійних, диференціальних та інтегро-диференціальних рівнянь, для розв'язання яких використовуються числові методи. Для дискретних виробничих процесів типовими операціями є складання, розбирання, монтаж, контроль якості виробів, усунення браку, а також розподіл виробів і обладнання по групах та об'єднання таких груп. Під час створення моделей таких процесів слід враховувати правила побудови черг, послідовність виконання операцій, які задаються маршрутними технологічними картами, а також можливі простої

обладнання. Необхідно також зважати на графіки роботи технічного персоналу і параметри надійності обладнання. Імітаційна модель таких процесів повинна відслідковувати інформацію щодо окремих об'єктів потоку виробів та їх атрибутів. Прикладами дискретних виробничих процесів можуть бути процеси виконання замовлень, розрахунку потрібних місць складування запасів та керування ними, визначення технологічного маршруту збирання обладнання, формування рахунків до сплати або обробка вимог тощо.

Головним завданням імітаційного моделювання виробничих процесів є пошук оптимальних параметрів та режимів процесу, що вже діє, або проектування нових технологічних процесів, які повинні забезпечити досягнення певної мети. Використовуючи модель процесу, можна уточнювати та визначати ймовірнісні характеристики параметрів, виявляти конкретні зв'язки між ними (наприклад, між рівнем завантаження виробничої дільниці, середнім часом пролежування заготівок і середнім часом, необхідним для виконання технологічної операції та ін.).

Будь-який технологічний процес, незалежно від його типу, являє собою сукупність об'єктів, що взаємодіють: виробів, одиниць технологічного обладнання, транспортних засобів, складів тощо. Вироби, рухаючись від одного об'єкту до іншого під час виконання технологічної операції, споживають певні ресурси. Тому виробничий процес можна зобразити як деякий матеріальний потік, що об'єднує всі без винятку об'єкти, наявні в системі. Всі операції, які виконуються над об'єктами матеріального потоку, мають певну тривалість у часі незалежно від того, чи змінюється стан об'єкту протягом виконання операції. У ході більшості технологічних операцій, таких як монтаж або демонтаж, обробка на верстатах, зварювання, стан об'єктів змінюється. Типовими операціями, що не впливають на стан об'єктів, є транспортування або випробування.

У мові GPSS для моделювання ресурсів виробничих процесів звичайно застосовують блоки SEIZE, RELEASE (для окремих ресурсів, наприклад пристроїв обслуговування), або блоки ENTER, LEAVE (для кількох однотипних ресурсів, наприклад складських приміщень або бункерів). Операції складання і демонтажу відтворюють за допомогою блоків MATCH, GATHER, ASSEMBLE. Технологічні маршрути у мові GPSS задають за допомогою функцій [61] або матриць (блок MSAVEVALUE).

Об'єкт у потоці може бути окремою одиницею (наприклад, заготівка) або групою деталей (наприклад, бункер з кількома заготівками, що рухається по конвеєру). Під час моделювання групового матеріального потоку окремі вимоги об'єднуються в ансамблі вимог. Для керування такими вимогами використовуються блоки GATE M і GATE NM.

Проектування процесів, що протікають у виробничих системах, передбачає створення стійкої технологічної схеми, оскільки послідовність операцій під час випуску продукції повторюється. Важливою процедурною концепцією моделювання таких процесів і аналізу їх ефективності є визначення періоду нестійкої роботи системи і усунення статистичних даних, зібраних за цей період.

Вимоги, які пред'являються до програмного забезпечення, яке застосовується для моделювання виробничих систем, подібні до тих, що пред'являються до програм моделювання будь-яких систем. Єдина різниця у них пов'язана з моделюванням виробничих систем з вантажно-розвантажувальним обладнанням.

9.2. Процеси розподілу ресурсів

Процеси розподілу – це процеси, до складу яких входять операції транспортування і постачання, що забезпечують переміщення продукції або людей між різними пунктами в мережі розподілення. Характерною особливістю процесу транспортування, на відміну від постачання, є те, що об'єкти потоку під час виконання даної операції – це люди, а не товари або вироби. Типові процеси транспортування можна зустріти у системах міського транспорту. Прикладами процесів постачання є збут продукції, доставка пошти і товарів.

Під час моделювання процесів розподілу необхідно спостерігати за такими характеристиками, як місце призначення, обсяг або витрати, а також контролювати властивості об'єктів потоку. Іноді при моделюванні процесів транспортування транспортні ресурси доцільніше зображати як об'єкти потоку. Транспортні засоби можуть характеризуватись кількістю місць, швидкістю та маршрутом руху. Для синхронізації об'єктів потоку бажано застосовувати списки користувачів (у мові GPSS це блоки LINK і UNLINK), що дозволяє моделювати сумісний рух транспортного засобу та деталей або пасажирів.

Для більшості процесів розподілу характерний перехідний режим. Тому тривалість моделювання має бути достатньою, щоб охопити весь цикл процесу, в тому числі й період, коли процес протікає в стаціонарному режимі. Крім того, для аналізу показників ефективності системи прогін моделі необхідно виконати кілька разів.

9.3. Процеси обслуговування

Однією з найважливіших сфер, де застосовується імітаційне моделювання, є процеси обслуговування клієнтів, оскільки в окремих системах сумарний час очікування клієнта у черзі може досягати 90 % загального часу його перебування у системі.

Імітаційне моделювання процесів обслуговування вважається складним завданням, тому що в одній моделі люди можуть бути як об'єктами потоку, так і ресурсами. Людям притаманна складна і непередбачувана поведінка, тому моделювання її в різних ситуаціях вимагає неабиякої гнучкості програмування. Моделі процесів виготовлення продукції, підготовки документів, функціонування устаткування або пересування транспортних засобів, побудовані без урахування людського фактора, набагато простіші.

У більшості імітаційних моделей час обслуговування і моменти появи вимог є випадковими величинами. Отже, для моделювання таких систем необхідно використовувати випадкові величини із заданими законами розподілу ймовірностей, які також треба визначити перед початком моделювання.

Оскільки надходження вимог до системи носить циклічний і випадковий характер, системи обслуговування рідко функціонують у стійкому режимі. Тому під час моделювання таких систем треба обов'язково враховувати наявність перехідних процесів у моделі та можливість зміни інтенсивності потоків у часі.

9.4. Процеси керування розробленням проектів

Усі проекти, незважаючи на велику кількість їх типів і різноманітність масштабів, мають ряд спільних характеристик, а саме:

- спрямованість на досягнення конкретних цілей;
- необхідність координування взаємопов'язаних дій;
- обмеженість у часі й унікальність кожного проекту.

Виконання проектів націлено на отримання певних результатів. Для того щоб досягти конкретних цілей, необхідне планування виконання робіт. У комплексі взаємопов'язаних цілей проекту, як правило, є головна. Наприклад, головною метою проекту, пов'язаного з розробкою програмного забезпечення, може бути створення автоматизованої системи керування підприємством. Проміжні цілі (підцілі) – це розробка технічного, інформаційного та програмного забезпечення, що передбачає деталізацію підцілей нижчого рівня, пов'язаних зі створенням бази даних, математичного і програмного забезпечення, тестуванням системи. Під час розробки бази даних, у свою чергу, також може бути виділено цілі більш низького рівня – розроблення логічної структури бази даних, реалізація бази даних за допомогою системи керування нею, завантаження даних тощо.

Великі проекти включають значну кількість взаємопов'язаних дій та етапів. У одних випадках ці взаємозв'язки досить очевидні (наприклад, наявність технологічної або логічної залежності), в інших – приховані. Через це деякі проміжні цілі не можуть бути реалізовані, доки не буде завершено виконання інших завдань. Якщо на різних етапах реалізації проекту порушується логічний зв'язок і синхронізація дій, весь проект може опинитися під загрозою зриву. Таким чином, проект – це динамічна система, яка потребує особливих підходів до керування нею.

Проміжок часу, відведений для виконання будь-якого проекту, завжди обмежений. Проект має чітко виражений момент початку і закінчення. Проект закінчується, коли досягнута його головна мета. Значну частину зусиль під час роботи з проектом спрямовано саме на забезпечення того, щоб проект було завершено в намічений термін. Для цього будуються графіки виконання робіт проекту.

Виконання проектів – здебільшого заходи неповторні й одноразові. Разом з тим за ступенем унікальності проекти можуть значно відрізнятись один від одного. Якщо розробляється оригінальна технологія або система, то, безумовно, маємо справу із завданням унікальним, яке ніколи раніше не виконувалось. І оскільки під час виконання проекту не завжди можна використати досвід розробки попередніх проектів, він повний ризику і невизначеності.

Керування ризиками – це фундамент процесів керування проектами. Ризик можна визначити як вплив негативних подій і їх наслідків. Для мінімізації ризиків необхідно ідентифікувати область потенційного ризику, визначити ймовірність його виникнення і потенційні наслідки. Для керування ризиками менеджеру проекту слід проаналізувати великий обсяг інформації різних видів і оцінити ситуацію з урахуванням особистого досвіду. Часто така оцінка суб'єктивна і не враховує ступінь невизначеності фактора ризику. В результаті створюється неповна, а часом і неточна, картина.

Необхідно визначати, в яких ситуаціях може виникнути ризик, зрозуміти його природу та оцінити його ймовірність. Для полегшення аналізу керування ризиками, дію ризику прийнято оцінювати з позиції його впливу на вартість, якість і тривалість виконання робіт. Наприклад, ризик втрати висококваліфікованого програміста в проекті, пов'язаному з розробкою програмного забезпечення, впливає на вартість, якість і тривалість виконання робіт проекту. Керування ризиками визначається як систематичний процес, пов'язаний з ідентифікацією, аналізом ризиків і прийняттям рішень, що забезпечують мінімізацію негативних наслідків настання ризикових подій і максимізацію ймовірності наслідків настання позитивних подій.

Аналіз ризиків можна виконати за допомогою моделі. Для цього реалізують такі заходи:

- для встановлення тривалості окремих (або всіх) робіт проекту створюють процедури введення випадкових величин із заданими законами розподілу ймовірностей;
- для визначення впливу ймовірностей на терміни виконання проекту проводять аналіз ризиків за допомогою методу статистичного моделювання;
- для здійснення аналізу впливу невизначеностей на тривалість виконання проекту формують звіти про виконання основних робіт.

Під час аналізу ризиків необхідно враховувати рівень ризику прийняття певного рішення в реальних умовах, що характеризуються неповнотою, невизначеністю та неточністю інформації. Тому поряд з традиційними методами імітаційного моделювання необхідно застосовувати методи ситуаційного моделювання і теорії ризиків. Слід також ураховувати, що аналіз та оцінка ризиків часто мають суб'єктивний характер.

Таким чином, головним завданням під час керування проектами є забезпечення максимальної ефективності виконання робіт у заданий термін, у рамках виділених коштів або ресурсів, відповідно до технічного завдання. Для того щоб вписатися з обмеженнями за часом, використовуються методи побудови календарних графіків робіт і контролю їх виконання, для керування грошовими обмеженнями – методи формування фінансового плану (бюджету) проекту і контролю за його дотриманням під час виконання робіт. Для ресурсного забезпечення існують спеціальні методи керування людськими і матеріальними ресурсами (наприклад, матриця відповідальності, діаграми завантаження ресурсів). Звичайно в аналізі подібних процесів задіяно програмний інструментарій керування проектами, такий, як Microsoft Office Project 2003 та ін. Проте оцінки часу, необхідного для реалізації повного циклу процесу, і вимог до ресурсів, які одержують у результаті аналізу, виконаного з використанням методик імітаційного моделювання, є більш точними, оскільки часові параметри здійснення проекту дуже нестабільні, а спільне використання ресурсів призводить до появи безлічі взаємозв'язків.

Приклад реалізації керування проектами для визначення впливу чисельності робочої сили на терміни виконання проекту можна знайти в праці Т. Шрайбера [68] (приклад 7 D).

9.5. Комп'ютерні системи та мережі

У більшості випадків імітаційні моделі комп'ютерних систем і мереж використовують на етапі їх проектування для оцінювання прогнозованих характеристик. Слід мати на увазі, що на попередньому етапі проектування для таких досліджень можна застосовувати і методи операційного аналізу мереж СМО (див. розділ 2.7.2). Такі мережі добре відображають роботу багатопроцесорних і багатомашинних комп'ютерних мереж та систем [31, 35]. В мережах СМО вузли моделюють роботу окремих пристроїв або вузлів.

Однією з найважливіших задач, які необхідно розв'язувати під час моделювання комп'ютерних систем, є дослідження різних режимів роботи і функцій операційної системи. Операційну систему можна розглядати як сукупність функціональних компонентів, кожен з яких відповідає за реалізацію певної функції системи; серед них основними є функції планування і виконання програм. Для успішного виконання програми потрібні певні ресурси. До них належать:

- ресурси, необхідні для послідовного виконання програм (передусім процесорний час);
- ресурси, що дають можливість зберігати інформацію, яка забезпечує виконання програм (регістри процесора, оперативна пам'ять тощо).

Операційна система виступає в ролі менеджера цих ресурсів і надає їх прикладним програмам на вимогу. При розподілі ресурсів операційна система розв'язує можливі конфлікти, запобігає несанкціонованому доступу програм до тих ресурсів, на які вони не мають прав, забезпечує ефективну роботу комп'ютерної системи.

У разі моделювання комп'ютерних систем у першу чергу слід враховувати принцип паралелізму, який передбачає одночасне (з погляду прикладного програміста) виконання дій різними фрагментами коду програми або різними пристроями обчислювальної системи. Цей принцип реалізується на одному процесорі шляхом перемикання задач (випадок псевдопаралелізму) або може базуватися на паралельному виконанні коду на декількох процесорах (випадок справжнього паралелізму).

Можна виділити такі основні види паралелізму: багатопроцесорних систем, операцій введення-виведення, взаємодії з користувачем і розподілених систем. Паралелізм багатопроцесорних систем є справжнім паралелізмом, тому що в таких системах інструкції виконують декілька процесорів одночасно. Якщо програма підтримує паралелізм введення-виведення, вона може виконуватись, не чекаючи на завершення операцій введення-виведення. Паралелізм взаємодії з користувачем необхідний, щоб виділити окремі потоки для безпосередньої взаємодії з користувачем. У ході інтерактивного сеансу роботи користувач може виконувати різні дії (і очікувати негайної реакції на них) до завершення обробки попередніх дій. Паралелізм розподілених систем дає можливість організувати паралельне обслуговування запитів, коли основний процес приймає запити, відразу передає їх для виконання іншим процесам і продовжує очікувати нові.

Моделювання мультипрограмного режиму побудоване на принципі розподілу часу роботи процесора між кількома програмами. Для кожної програми виділяється свій проміжок часу (квант часу), впродовж якого вона займає процесор. Якщо за цей час програма не завершується, надходить сигнал переривання, і процесор переходить до виконання іншої програми. Звичайно програма, виконання якої було перервано, стає в чергу згідно з правилами, що існують у системі. Через деякий час програма знову надходить до процесора. Таким чином, у процесі виконання кожна програма займає процесор кілька разів. Подібні черги створюються і до інших пристроїв системи, наприклад до пристроїв введення-виведення, до накопичувачів на магнітних дисках та ін. Здебільшого такі системи моделюються як одноканальні СМО, на відміну від багатоканальних, що моделюють роботу багатопроцесорних систем. Головними завданнями під час моделювання комп'ютерної системи у режимі розподілу часу є оцінювання величини квантів часу, які виділяються для програм певних груп користувачів, дослідження стратегій організації черг. Ще одне завдання, що виникає під час моделювання обчислювальних систем, – оцінювання розмірів основної та зовнішньої пам'яті і кількості процесорів, необхідних для виконання основних функцій системи.

Під час моделювання звичайно досліджують тривалість перебування програми користувача в комп'ютерній системі (від введення програми або запиту на її виконання до отримання результатів її роботи або відповіді) або оцінюють кількість програм, виконаних системою за одиницю часу. В таких випадках у системі моделювання для кожної програми чи запиту користувача фіксують такі дані: моменти надходження до системи і завершення роботи, обсяг потрібної пам'яті, кількість і тривалість запитів до пристроїв введення-виведення, обсяг даних, що виводяться на пристрої введення-виведення тощо. Стратегія використання ресурсів визначається алгоритмами планування та виконання програм в операційній системі. А окремі ресурси моделюються одно- або багатоканальними пристроями обслуговування.

Моделювання комп'ютерних мереж – це більш складне завдання, ніж моделювання систем, оскільки у цьому разі необхідно враховувати набагато більше число факторів і забезпечити виконання багатьох суперечливих вимог. Під час створення імітаційної моделі мережі головним завданням проектувальника є ретельний аналіз роботи мережі, що складається з безлічі окремих компонентів: комп'ютерів, комутаторів, маршрутизаторів, пристроїв введення-виведення, ліній зв'язку та інших, і прийняття рішення стосовно вибору типу моделі та методів дослідження. Попереднє вивчення мереж виконують за допомогою методів операційного аналізу, орієнтованих на обчислювальні системи [76], а більш детальне дослідження проводять за допомогою імітаційних моделей на основі мереж СМО.

Метою моделювання комп'ютерних мереж може бути:

- визначення типів і параметрів активного і пасивного обладнання мережі, яке забезпечить певні мінімальні потреби передавання, обробки та збереження інформації;
- оцінювання необхідного запасу продуктивності основних пристроїв мережі, яка забезпечить прогнозований ріст виробничих потужностей інформаційної системи протягом найближчих років;

- вибір одного чи кількох варіантів архітектури мережі на базі критерію вартості обладнання;
- перевірка роботи обчислювальної мережі із заданою архітектурою.

Типовим завданням дослідження мережі під час моделювання є оцінювання середнього часу затримки передачі файлів великих розмірів внаслідок збільшення кількості комп'ютерів у мережі. Метою досліджень може бути і оптимізація роботи компонентів мережі, наприклад комутаторів або маршрутизаторів. У цьому випадку імітаційну модель мережі слід будувати як комплекс мереж СМО, який, можливо, містить сотні або навіть тисячі СМО.

Проектувальник мережі повинен отримати відповіді перш за все на такі запитання: як керувати перевантаженням у мережі; яку ємність повинен мати буфер кожного вузла; які пріоритети мають призначатися різним лініям зв'язку, що використовуються у мережі та ін. Проектувальники мають також дослідити, яким є середній час відповіді на запит у мережі та як зміниться час реакції мережі на запит залежно від алгоритмів розподілу пакетів. Це вимагає визначення періоду чекання, коефіцієнтів використання компонентів, тривалості перемикання комутаторів і періоду затримки відповідей на запити, спричинені перемиканням тощо.

Під час передачі пакетів доцільно аналізувати такі показники роботи мережі, як продуктивність, час відповіді та обробка пакетів. Слід зазначити, що від виду досліджень залежить спосіб визначення часу затримки. При спрощеному визначенні фіксується проміжок часу від моменту введення запиту користувачем до появи відповіді системи; тривалість введення запиту та виведення відповіді не враховується. У разі моделювання процесів взаємодії користувачів у комп'ютерних системах з розподілом часу затримка відповіді визначається від моменту закінчення введення даних до моменту закінчення виведення, а у пакетних системах – від початку введення запиту до завершення виведення.

Дещо інші запитання постають у разі дослідження процесів, що пов'язані з роботою адміністраторів вузлів мережі, наприклад, компанії або міста, а саме: яка топологія мережі є оптимальною; які смуги частот мають бути виділені для індивідуальних користувачів та інші.

Два важливих показники, які часто використовуються як характеристики роботи мережі або системи в цілому, – час чекання і продуктивність. Час чекання означає затримку, пов'язану з виконанням певної операції; наприклад, це може бути час від моменту передачі у мережу першого біта повідомлення до моменту появи останнього біта у кінцевого адресата. Точне визначення цих проміжків часу залежить від цілей вивчення і моделювання системи.

За час чекання часто приймають час відповіді, особливо коли аналізується робота системи. Просте визначення часу відповіді – фіксація проміжку часу від моменту, коли користувач направляє запит на виконання операції, до моменту видачі системою відповіді.

Продуктивність здебільшого визначається як швидкість проходження запитів або як проміжок часу, за який запит обслуговується системою. Інтерактивну продуктивність вимірюють у запитах за секунду, тоді як пакетну – у завданнях за секунду. Продуктивність процесора часто вказують у мільйонах команд за секунду

(Million Instructions Per Second, MIPS) або мільйонах операцій з плаваючою комою за секунду (Million Floating Point Operations Per Second, MFLOPS), продуктивність системи діалогової обробки запитів – у транзакціях за секунду (Transactions Per Second, TPS).

Продуктивність визначається як кількість запитів користувачів, що можуть бути виконані за одиницю часу. В мережах зв'язку це відноситься до числа бітів інформації (загального або для кожного користувача окремо), яку може бути передано мережею за секунду. Під час дослідження роботи системи продуктивність обчислюється як кількість завдань, виконуваних за одиницю часу.

Слід мати на увазі, що тривалий час чекання не обов'язково означає, що система має низьку продуктивність. Наприклад, мережі супутникового зв'язку мають широку смугу частот і високу продуктивність, але тривалий час чекання.

Важливим показником ефективності роботи системи є також коефіцієнт використання ресурсу, що показує, яка частина ресурсу витрачається на обслуговування запитів. Коефіцієнт розраховується у процентах або у долях одиниці від обсягу всього ресурсу пристрою чи обладнання. Для процесорів коефіцієнт використання можна визначити як відношення тривалості простою до тривалості роботи, для пам'яті – як відношення об'єму пам'яті, що зайнята, до загального об'єму пам'яті (використання визначається як середнє арифметичне за фіксований проміжок часу).

Зрозуміло, що значення коефіцієнту використання ресурсу лежить у проміжку між нулем (повністю не використовується) та одиницею (завжди зайнятий). За цим показником можна визначати критичні параметри в системі або її граничні можливості. Деякі ресурси, наприклад пам'ять комп'ютера, завжди завантажені лише частково. У цьому випадку коефіцієнт використання визначається як середня частина зайнятого ресурсу за інтервал часу.

Інші показники функціонування комп'ютерних систем, що часто розраховуються під час моделювання, пов'язані з надійністю обладнання і характеризують середній проміжок часу між перебоями в його роботі, або з доступністю обладнання, яка визначає середній проміжок часу між невдалими спробами доступу до нього. Імітаційне моделювання часто застосовується для оцінювання надійності системи, тобто середнього проміжку часу між відмовами компонента системи або системи в цілому.

Висновки

- Моделювання виробничих процесів націлене або на пошук оптимальних параметрів і режимів, або на проектування процесів, які мають забезпечити досягнення певної мети.
- Під час моделювання процесів розподілу необхідно спостерігати за такими характеристиками, як місце призначення, обсяг або витрати, а також контролювати властивості об'єктів потоку.
- При моделюванні процесів сучасних інформаційних технологій досліджуються процеси перетворення, обробки та передавання інформації, які виконуються комп'ютерними системами та системами передачі даних.

Контрольні запитання та завдання

1. Перерахуйте основні виробничі операції, що виконуються під час моделювання виробничих дискретних процесів складання. Яким чином можна моделювати ці операції мовою GPSS?
2. Виконайте завдання 2, «Моделювання процесу складання комп'ютерів», з розділу 7, що наведено в книжці [61].
3. Виконайте завдання 23, «Моделювання роботи маршрутних таксі», з того ж розділу.
4. Визначіть імовірнісні характеристики часу передачі повідомлень між абонентами регіональної мережі обчислювальних машин (завдання 26 з того ж розділу).

Термінологічний словник

Абстрактна мова. Сукупність застосовуваних у мові символів і правил користування ними. Вислів даною абстрактною мовою означає, що є якесь речення (формула), побудоване за граматичними правилами цієї мови. Ця формула може містити варійовані змінні, так званні конституєнти, які тільки за певних значень роблять даний вислів істинним.

Аналітична модель. Модель, що складається з системи рівнянь, які можна розв'язати (наприклад, система рівнянь, що представляють закони попиту та пропозиції на світовому ринку).

Аналогія. Схожість об'єктів за деякими ознаками. Аналогія між об'єктами може встановлюватися за якісними і (або) кількісними ознаками.

Апроксимація. Наближене відтворення одних математичних об'єктів за допомогою інших (наприклад, наближене відображення складної функції за допомогою однієї або декількох більш простих).

Архітектура. Структура компонентів у програмі або системі, взаємозв'язок компонентів і принципи керування ними, що обумовлені наявністю певних конструкцій і тенденцій у розвитку протягом деякого часу.

Атрибути. Властивості чи характеристики об'єктів.

Неперервна модель. Математична або обчислювальна модель, вихідні змінні якої є неперервними у часі. Протилежність — дискретна модель.

Гіпотези. Наукові припущення, що зроблено для пояснення певних явищ дійсності; звичайно визначають невідомі закономірності у системі або постановку задачі. За відсутністю інформації висувають гіпотези щодо можливих результатів, які потім перевіряють експериментально.

Гомоморфізм. Вказує на однозначну відповідність між двома системами чи об'єктами лише в одному напрямку, тобто від однієї системи або об'єкта до іншого. Одна із систем або один з об'єктів може бути моделлю.

Головний ефект фактора. Внесок фактора в значення функції відгуку під час переходу його від нижнього рівня до верхнього.

Гра. Спрощене відтворення реального процесу; звичайно використовується для навчання, прийняття рішень, досліджень або розваг.

Детермінована модель. Модель, результати функціонування якої визначені через відомі відношення для станів і подій. Один і той же заданий вхід завжди спричиняє появу одного і того ж результату (наприклад, модель, що відтворює відому хімічну реакцію). Протилежність — стохастична модель.

Детермінований алгоритм. Процес обчислень, результатом якого є унікальний і передбачуваний результат для заданих вхідних даних.

Детермінований. Має відношення до процесу (моделі, імітації або змінної), результат якого (або значення) не залежить від випадку. Протилежність — стохастичний, чи ймовірнісний.

Динамічна модель. Модель системи, у якій відбуваються зміни через виникнення подій у часі або рух об'єктів у просторі (наприклад, модель моста, який має навантаження, що змінюється, для визначення характеристики моста).

Дискретна імітація. Імітація, що використовує дискретну модель.

Дискретна модель. Математична або обчислювальна модель, вихідні змінні якої приймають тільки дискретні значення (наприклад, модель, що прогнозує рівні запасів організації, ґрунтуючись на відвантаженнях, які змінюються, і платежах). Протилежність — неперервна модель.

Дискретна система. Система, для якої стани змінних змінюються миттєво в часі.

Дисперсія. Числова характеристика розподілу ймовірностей випадкової величини X, яка характеризує міру розсіювання випадкової величини відносно її, математичного сподівання і визначається формулою $D(X) = \sigma_x^2 = M(x - M(X))^2$.

Діаграма подій. Відображає послідовність подій у часі. Використовуючи її можна відтворити роботу системи, тобто провести імітаційне моделювання графічними методами.

Довіра до моделювання (моделі) й імітації (M&I). Офіційне свідчення того, що модель або імітація прийнятна для використання у визначених цілях.

Згладжування. Інтерполяція попереднього стану об'єкта (розміщення, швидкість і т. ін.) до поточного стану зі створенням вирівняного переходу між двома послідовними модифікованими станами об'єктів.

Ідентифікація системи. Побудова математичної моделі процесу чи об'єкта шляхом спостереження за його вхідними та вихідними змінними.

Ізоморфізм. Взаємна однозначність між системами або об'єктами, що розглядаються. Одна з систем або один із об'єктів може бути моделлю. Строго доведений ізоморфізм для систем різної природи дозволяє переносити знання з однієї галузі в іншу.

Імітаційне моделювання. Метод конструювання моделі системи та проведення експериментів на моделі. Суттєвими особливостями цього виду моделювання є *опис структури* модельованої системи, застосування засобів *відтворення функціонування* (*поведінки*) системи на моделі, *відображення властивостей середовища*, в якому функціонує досліджувана система.

Керування імітацією. Механізм, що забезпечує централізоване керування процесом імітації. Функції керування імітацією включають: початковий запуск, повторний запуск, підтримку, відключення, використання, збір і розподіли деяких типів даних. Зазвичай ці функції виконує програма-інтерпретатор.

Комп'ютерне моделювання. Реалізація процесу моделювання за допомогою комп'ютера. Важливою особливістю комп'ютерного моделювання є його *інтерактивність* — наявність можливості втручання користувача в процес моделювання та впливу на його результати. Вона забезпечується завдяки узгодженості дій користувача та моделі, яка відтворює об'єкти реального середовища або гіпотетичні події та процеси.

Концептуальна модель. Абстрактна модель, яка виявляє причинно-наслідкові зв'язки, властиві досліджуваному об'єкту в межах, визначених цілями дослідження.

По суті, це формальний опис об'єкта моделювання, який відображає концепцію (погляд дослідника на проблему).

Або

змістовний опис об'єкта чи процесу, в якому відображені концепції користувача і розробника моделі. Він включає в явному вигляді логіку, алгоритми, припущення й обмеження.

Лінгвістична модель. Модель, описувана деякою абстрактною мовою, в окремому випадку це може бути природна мова. Відміна даної моделі від формальної полягає в тому, що слова можуть мати декілька значень, тобто модель може бути неоднозначною. Природною мовою можна зробити модель у вигляді опису (наприклад, твір або сценарій). (Див. також *приписуюча модель.*)

Макетна модель. Реально існуюча модель, що відтворює модельовану систему у деякому масштабі.

Масштабна модель. Фізична модель, модель даної системи в зміненому масштабі, (наприклад, точна копія літака, розмір якої складає одну десяту розміру оригіналу).

Математична модель. Абстрактна модель, відображена у вигляді математичних виразів і відношень. Якщо відношення задаються аналітично, то їх можна розв'язати в *замкненому вигляді* (явно) відносно шуканих змінних як функції від параметрів моделі, або в *частково замкненому вигляді* (неявно), коли шукані змінні залежать від одного або багатьох параметрів моделі. До моделей цього класу належать диференціальні, інтегральні, різничні рівняння, імовірнісні моделі, моделі математичного програмування та ін. Якщо не можна здобути точний розв'язок математичної моделі, використовуються *числові методи.*

Математичне сподівання. Числова характеристика розподілу ймовірностей випадкової величини X, яка має щільність розподілу $f(x)$. Математичне сподівання

$$M(X) = \int_{-\infty}^{\infty} xf(x)\,dx.$$

Якщо X набуває значення $x_1, x_2, ..., x_n$ з імовірностями $p_1, p_2, ..., p_n$, то

$$M(X) = \sum_{i=1}^{n} x_i p_i.$$

Мережа Петрі. Орієнтований дводольний граф з маркерами (помічений орієнтований граф), який має дві групи вершин: вузли та переходи. Вузли можуть бути пустими або поміченими та визначають стан мережі. Переходи визначають дії. Орієнтовані ребра графу задають зв'язки між вузлами та переходами.

Метод аналогій. Дає змогу встановити відношення еквівалентності (відповідності, схожості) між двома системами, що розглядаються, за деякими ознаками. Будь-яка з цих систем може реально існувати або бути абстрактною.

Метод статистичних випробувань, або метод Монте-Карло. Дає приблизне вирішення різноманітних математичних проблем за допомогою виконання стати-

стичних вибіркових експериментів на комп'ютері. Моделювання методом Монте-Карло у широкому розумінні визначається як будь-який спосіб вирішення моделі, де використовуються випадкові числа або псевдовипадкові числа.

Модель «прозорого ящика». Модель, внутрішнє наповнення якої відоме і цілком видиме. Протилежність – *модель «чорного ящика»*.

Модель «чорного ящика». Модель, входи і виходи та функціональні експлуатаційні характеристики якої відомі, а внутрішнє зображення невідоме або невизначене. Протилежність – *модель «прозорого ящика»*.

Модель наочна (описова). Модель, призначена для зображення поведінки або властивостей існуючої системи або типової системи (наприклад, масштабна модель або письмовий опис, що дозволяє знайомити потенційних покупців з фізичними і робочими характеристиками комп'ютера). Протилежність – *приписуюча модель*.

Модель символічна. Модель, властивості якої виражені за допомогою символів. Це може бути графічна, математична, оповідальна, програмна і таблична модель.

Модель статична. Модель системи, в якій не відбувається ніяких змін під час спостережень, (наприклад, масштабна модель моста, використовувана для вивчення його типу, замість його експлуатаційних показників при зміні навантажень).

Модель. Фізичне, математичне або інше логічне зображення системи, об'єкта, явища або процесу.

Модельний час (див. також час імітаційний). Арифметична величина, яка має додатні зростаючі значення та при моделюванні відображає плин часу в моделі. Останній може задаватися шляхом змінювання часу протікання процесу з постійним кроком (*принцип* Δt) або з випадковим кроком, що дорівнює інтервалу між двома послідовними подіями (*принцип особливих станів* δz). В останньому випадку йдеться тільки про ті «особливі» події, які переводять систему з одного стану z_i в інший стан z_j у просторі станів Z. Можлива й комбінація цих принципів.

Моделювання й імітація (M&I). Використання статичних або динамічних моделей, включаючи емулятори, дослідні зразки, які моделюють пристрої і стимулятори для формування даних як підстави для створення організаційних або технічних рішень. Терміни «моделювання» та «імітація» часто взаємозамінні.

Моделювання. Застосування моделі. Строго структурована методологія створення і підтвердження фізичного, математичного або логічного зображення системи, об'єкта, явища або процесу.

Моделюючий пристрій (імітатор). Пристрій, комп'ютерна програма або система, що реалізує імітаційну модель; може використовуватись під час досліджень або для навчання і дублює істотні особливості обстановки і забезпечує безпосередню участь людини.

Натурні моделі. Самі існуючі системи або їх частини, на яких проводяться дослідження.

Перевірка достовірності (верифікація). Процес визначення того, що модель або виконувана імітація точно відтворює детальний концептуальний опис, прийнятий розробником. Перевірка достовірності також оцінює ступінь відповідності

моделі або імітації змісту і проводиться з використанням прийнятих методів програмування.

Перевірка правильності (валідація). Процес визначення ступеня точності, з яким модель або імітація відображає реальний або створюваний світ.

Перевірка правильності (кимось). Процес визначення, чи здається модель або імітація розумною людям, які добре інформовані щодо системи завдяки вивченню експлуатаційних показників моделі. У цьому процесі не розглядається програмний код або логіка, а, скоріше, вивчаються входи і виходи з метою визначення їх реалістичності або наочності.

Повідомлення (траизакт). Абстрактна динамічна структура з набором атрибутів, які несуть інформацію про реальний динамічний об'єкт.

Подія. Зміна властивостей об'єкта, взаємодія об'єктів, утворення нового об'єкта або знищення існуючого. Ця зміна пов'язана з фіксованою точкою на осі часу. Кожна подія містить оцінку часу, що вказує, коли вона відбувається.

Прикладна підтримка моделювання й імітації (M&I). Сукупність технічних і організаційних засобів, здійснюваних організацією, яка використовує результат або продукт певного призначення.

Приписуюча модель. Модель, що відображає необхідну поведінку чи властивості запропонованої системи (наприклад, масштабна модель або письмовий опис з фізичними і робочими характеристиками певної моделі, представлений постачальнику комп'ютерів). Протилежність — *описова (наочна) модель.*

Прогнозуюча модель. Модель, у якій певні показники майбутніх станів можуть бути передбаченими або ймовірними (наприклад, модель, яка прогнозує характер погоди, що ґрунтується на поточному значенні температури, вологості, швидкості повітряних потоків тощо у різних місцевостях).

Процес. Описує поведінку системи і визначається послідовністю станів, зв'язок між якими задається функцією дії та початковим станом системи. Процес також можна уявити як «життєвий цикл» динамічного об'єкта. Згідно з прагматичним підходом процес — це програмний блок, складений з ряду процедур (підпрограм), які описують поведінку повідомлень. Шлях руху повідомлення задається порядком звернення до процедур.

Реальна модель. Модель, у якій принаймні один представлений компонент є матеріальним об'єктом (наприклад, побудована точна фізична копія).

Ресурс (активність). Абстрактна структура з набором атрибутів, яка характеризує спосіб доступу до неї та її фізичне відображення.

Система. Набір компонентів, організованих для виконання визначених або установлених функцій.

Список. Множина (постійна або тимчасова) пов'язаних об'єктів, упорядкованих деяким логічним способом. Для позначення початку списку використовується елемент, який називається головою списку. Якщо список циклічний двоспрямований, то голова списку є також кінцевим елементом. Зазвичай списки використовуються для керування процесом імітації або організації черг.

Спрощення системи. Скорочення множини властивостей (атрибутів) та аспектів поведінки системи, які визначають простір станів системи.

Стан системи. Визначається неперервними або дискретними значеннями характеристик елементів системи. Основне значення стану в моделюванні полягає у забезпеченні можливості зв'язати з кожною вхідною змінною єдину вихідну змінну, використовуючи стан системи як параметр.

Стохастична модель. Модель, у якій для визначення результату використовують одну або більше випадкових величин для врахування невизначеності процесу, або в якій вхідні дані представлені відповідно за допомогою деякого статистичного розподілу. Наприклад, модель, що оцінює витрачені гроші в кожному відділі універсаму, заснована на імовірнісних значеннях кількості клієнтів і покупок кожного клієнта.

Стохастичний процес. Будь-який процес, що протікає в часі і може бути аналітично описаний лише у термінах теорії імовірностей.

Стохастичний. Має відношення до процесу (або моделі чи змінної), результат якого (або значення якої) залежить від випадку. Протилежність – *детермінований.*

Структура моделі. Зображення фізичної чи логічної структури системи (наприклад, зображення мережі комп'ютерів як набору блоків, об'єднаних лініями зв'язку). (Див. *модель процесу.*)

Теорія подібності. Наукова основа моделювання як методу пізнання і дослідження різних об'єктів, в якій головну роль відіграє метод аналогій, тобто подібності об'єктів за деякими ознаками.

Терми. Правильно побудовані вирази, значеннями яких є об'єкти, наприклад назви предметів, члени речення тощо.

Уявна (віртуальна) модель. Модель, що є відображенням ідеального уявлення людей про навколишній світ, який фіксується у свідомості за допомогою думок і образів.

Функтори. Певні відношення між термами.

Функція дії (англ. термін «activity»). Функція, що переводить систему з одного стану в інший, тобто зв'язує одну подію з іншою (наприклад, для СМО функціями дії є надходження, чекання та обслуговування вимог). Використовуючи опис подій та функцій дій, можна побудувати алгоритм, що імітує функціонування системи через її стани. Функція дії може бути задана в явному вигляді (наприклад, за допомогою диференціального рівняння), або у вигляді моделюючого алгоритму, який визначає стан системи в кожний момент часу.

Час імітаційний. Це арифметична величина, яка має додатні зростаючі значення та при моделюванні відображає плин часу в моделі. Імітаційний час може протікати швидше, повільніше або в тому ж самому темпі, що й астрономічний час.

Або

початкова точка відліку часу в межах імітаційного процесу (наприклад, універсальний скоординований час), що встановлюється функцією керування імітацією перед початком імітації (при паралельному моделюванні зазвичай використовується для всіх індивідуальних учасників).

Часовий ряд. Реалізація випадкового процесу, яка є результатом спостереження за випадковим процесом на вході або виході стохастичної імітаційної моделі й фіксується за допомогою вимірів.

Література та посилання

1. *Айвазян С. А.* Прикладная статистика: Основы моделирования и первичная обработка данных / *С. А. Айвазян, И. С. Енюков, Л. Д. Мешалкин.* — М.: Финансы и статистика, 1983. — 471 с.
2. *Акоф Р., Сасиени М.* Основы исследования операций. — М.: Мир, 1971. — 534 с.
3. *Альянах И. Н.* Моделирование вычислительных систем. — Л.: Машиностроение. Ленингр. отд-ние, 1988. — 223 с.
4. *Андрианов А. И., Бычков С. В., Хорошилов А. И.* Программирование на языке СИМУЛА-67. — М.: Наука, 1985. — 288 с.
5. *Банди Б.* Методы оптимизации: Вводный курс. — М.: Радио и связь, 1988. — 128 с.
6. *Бокс Дж., Дженкинс Г.* Анализ временных рядов: прогноз и управление. — Вып.1. — М.: Мир, 1974. — 406 с.
7. *Боровиков В. П., Боровиков И. П.* STATISTICA. Статистический анализ и обработка данных в среде Windows. — М.: Информ.-издат. дом «Филинъ», 1998. — 608 с.
8. *Брукс Ф. П.* (мл.) Как проектируются и создаются программные комплексы. Мифический человеко-месяц. — СПб.: Символ-Плюс, 1999. — 150 с.
9. *Васильев Д. В., Сабинин О. Ю.* Ускоренное статистическое моделирование систем управления. — Л.: Энергоатомиздат, 1987. — 136 с.
10. *Вендров А. М.* CASE-технологии: современные методы и средства проектирования информационных систем. — М.: Финансы и статистика, 1998. — 175 с.
11. *Гультяев А.* Визуальное моделирование в среде MATLAB: Учебный курс. — СПб.: Питер, 2000. — 432 с.
12. *Емельянов В. В., Ясиновский С. И.* Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. — М.: АНВИК, 1998. — 427с.
13. *Емельянов В. В., Попов Д. С.* ЭС планирования производства и ее реализация на языке интеллектуального РДО-имитатора // Перспективные информационные технологии и интеллектуальные системы. — 2000. — № 3. — С. 4–12.
14. Енциклопедія кібернетики. — К.: АН УРСР; Голов. ред. УРЕ. — Т. 2, 1973. — 573 с.
15. *Ермаков С. М., Михайлов Г. А.* Статистическое моделирование. — М.: Наука, 1982. — 296 с.
16. *Заболотский В. П., Оводенко А. А., Степанов А. Г.* Математические модели в управлении: Учеб. пособие. — СПб.: СПбГУАП, 2001. — 196 с.
17. *Карабегов А. В., Тер-Микаэлян Т. М.* Введение в язык SDL. — М.: Радио и связь, 1993. — 184 с.
18. *Кельтон В., Лоу А.* Имитационное моделирование. — 3-е изд. — СПб.: Питер; К.: Издат. группа BHV, 2004. — 847 с.
19. *Кендел М.* Временные ряды. — М.: Финансы и статистика, 1981. — 199 с.
20. *Киндлер Е.* Языки моделирования. — М.: Энергоатомиздат, 1985. — 288 с.
21. *Клейнен Дж.* Статистические методы в имитационном моделировании. — Вып.1, 2. — М.: Статистика, 1978.

22. *Клейнрок Л.* Теория массового обслуживания. — М.: Машиностроение, 1979. — 432 с.

23. *Кнут Д. Э.* Искусство программирования. — Т. 2: Получисленные алгоритмы. — 3-е изд. — М.: Издат. дом «Вильямс», 2001. — 832 с.

24. *Коваленко И. Н.* Анализ редких событий при оценке эффективности и надежности систем. — М.: Сов. радио, 1980. — 208 с.

25. *Коваленко И. Н., Кузнецов Н. Ю.* Методы расчета высоконадежных систем. — М.: Сов. радио, 1980. — 176 с.

26. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. — М.: Наука, 1974. — 832 с.

27. *Котов В. Е.* Сети Петри. — М.: Наука, 1984. — 196 с.

28. *Крейн М., Лемуан О.* Введение в регенеративный метод анализа моделей. — М.: Наука, 1982. — 104 с.

29. *Кузин Л. Т.* Основы кибернетики.— Т. 2: Основы кибернетических моделей. — М.: Энергия, 1979. — 584 с.

30. *Купрашвили А. Ю., Савустьяненко Э. И., Томашевский В. Н.* Организация интерактивной системы моделирования // Электронное моделирование. — 1987. — № 1. — С. 16–19.

31. *Литвин В. Г., Аладышев В. П., Винниченко А. И.* Анализ производительности мультипрограммных ЭВМ. — М.: Финансы и статистика, 1984. — 159 с.

32. *Литвинов В. В.* Математическое обеспечение проектирования вычислительных систем и сетей. — К.: Техніка, 1982. — 216 с.

33. *Литвинов В. В., Марьянович Т. П.* Методы построения имитационных систем. — К.: Наук. думка, 1991. — 120 с.

34. *Лукашин Ю. Г.* Адаптивные методы краткосрочного прогнозирования. — М.: Статистика, 1979. — 254 с.

35. *Максимей И. В.* Имитационное моделирование на ЭВМ. — М.: Радио и связь, 1988. — 232 с.

36. *Мансуров Н. Н., Майлингова О. Л.* Методы формальной спецификации программ: языки MSC и SDL. — М.: Изд-во АО «Диалог-МГУ», 1998. — 125 с.

37. *Марк Д. А., Мак-Гоуен К.* .ADT — методология структурного анализа и проектирования. — М.: Метатехнология, 1993.

38. *Мартин Ф.* Моделирование на вычислительных машинах. — М.: Сов. радио, 1972. — 288 с.

39. *Морозов К. Е.* Математическое моделирование в научном познании. — М.: Мысль, 1969. — 215 с.

40. *Нейлор Т.* Машинные имитационные эксперименты с моделями экономических систем. — М.: Мир, 1975. — 500 с.

41. *Новиков О. А., Петухов С. И.* Прикладные вопросы теории массового обслуживания. — М.: Сов. радио, 1969. — 400 с.

42. Общая теория систем: Сб. статей / Пер. с англ. — М.: Мир, 1966.

43. Основы моделирования сложных систем: Учеб. пособие / Под. общ. ред. д-ра техн. наук *И. В. Кузьменко* — К.: Вища шк., 1981. — 360 с.

44. Основы системного анализа и проектирования АСУ: Учеб. пособие / *А. А. Павлов, С. Н. Гриша, В. Н. Томашевский* и др.; Под общ. ред. *А. А. Павлова*. — К.: Вища шк., 1991. — 367 с.

45. *Питерсон Дж.* Теория сетей Петри и моделирование систем. — М.: Мир, 1984. — 264 с.

46. *Прицкер А.* Введение в имитационное моделирование и язык СЛАМ II. — М.: Мир, 1987. — 646 с.

47. *Поляк Ю. Г.* Вероятностное моделирование на электронных вычислительных машинах. — М.: Сов. радио, 1971. — 400 с.

48. Программное обеспечение персональных ЭВМ / *А. А. Стогний, С. А. Ананьевский, Я. И. Барсук* и др.; Под ред. А. А. Стогния — К.: Наук. думка, 1989. — 368 с.

49. Программные средства моделирования непрерывно-дискретных систем / *В. М. Глушков, В. В. Гусев, Т. П. Марьянович, М. А. Сахнюк.*— К.: Наук. думка, 1975. — 167 с.

50. *Рыжиков Ю. И.* Имитационное моделирование. Теория и технология. — СПб.: КОРОНА принт; М.: Альтекс-А, 2004. — 384 с.

51. *Рузавин Г. И.* Философские проблемы оснований математики. — М.: Наука, 1983. — 304 с.

52. Самарский А. А., Михайлов А. П. Математическое моделирование. Идеи. Методы. Примеры. — 2-е изд., испр. — М.: Физматлит, 2001. — 316 с.

53. *Ситник В. Ф., Орленко Н. С.* Імітаційне моделювання: Навч. посібник. — К.: КНЕУ, 1998. — 208 с.

54. *Ситник В. Ф., Орленко Н. С.* Імітаційне моделювання: Навч.-метод. посібник для самост. вивч. дисц. — К.: КНЕУ, 1999. — 208 с.

55. *Советов Б. Я., Яковлев С. А.* Моделирование систем: Учебник для вузов. — М.: Высш. шк., 1998. — 320 с.

56. *Советов Б. Я., Яковлев С. А.* Моделирование систем. Практикум: Учеб. пособие для вузов. — М.: Высш. шк., 1999. — 224 с.

57. Становление и развитие имитационного моделирования в Украине / *В. В. Бигдан, В. В. Гусев, Т. П. Марьянович, М. А. Сахнюк* // Пр. міжнар. симп. «Комп'ютери у Європі. Минуле, сучасне та майбутнє». — К., 1998. — С. 182–193.

58. Статистическое моделирование и прогнозирование: Учеб. пособие / *Г. М. Гамбаров, Н. М. Журавель* и др.; Под ред. А. Г. Гранберга. — М.: Финансы и статистика, 1990. — 383 с.

59. *Тимофеев Н. К.* Алгоритмы и программы на Паскале. — К.: Вища шк., 1992.

60. *Томашевський В. М.* Імітаційне моделювання систем і процесів. — К.: ІСДО, 1994. — 124 с.

61. *Томашевский В., Жданова Е.* Имитационное моделирование в среде GPSS. — М.: Бестселлер, 2003. — 416 с. — (Факультет).

62. *Томашевський В. М., Жданова О. Г., Жолдаков О. О.* Вирішення практичних завдань методами комп'ютерного моделювання. — К.: Корнійчук, 2001. — 267 с.

63. *Томашевський В. М., Жданова О. Г.* Метод структурної оптимізації з використанням імітаційної моделі // Міжнародна конференція з індуктивного моделювання. — Т. 2. — Львів: Державний НДІ інформаційної інфраструктури, 2002. — С. 224–227.

64. *Феррари Д.* Оценка производительности вычислительных систем. — М.: Мир, 1981. — 576 с.

65. *Форрестер Дж.* Промышленная динамика. — М.: Прогресс, 1971.

66. *Цвиркун А. Д., Ахинфиев В. К., Филиппов В. А.* Имитационное моделирование в задачах синтеза структуры сложных систем: Оптимизац.-имитац. подход / Отв. ред. В. Н. Бурков. — М.: Наука, 1985. — 173 с.

67. *Шеннон Р.* Имитационное моделирование систем — искусство и наука. — М.: Мир, 1978. — 418 с.

68. *Шрайбер Т. Дж.* Моделирование на GPSS. — М.: Машиностроение, 1980. — 593 с.

69. Язык спецификаций SDL/PLUS и его применение / *Я. М. Бардзинь, А. А. Калкиньш, Ю. Ф. Стродс, В. А. Сыцко.* — Рига, 1988. — 313 с.

70. *Arsham H.* Techniques for Monte Carlo Optimizing // Journal Monte Carlo Methods and Applications. — 1998. — Vol. 4, N 3. — P. 181–230.

71. *Balci O.* Verification, Validation And Accreditatuon Of Simulation Models // Proceedings of the 29th conference on Winter simulation. — N.Y.: ACM Press, 1997. — P. 135–141.

72. *Banks J., Carson J. S., Nelson B. L.* Discrete-Event System Simulation. — 3d ed. — S. l.: Prentice Hall, 2001. — 594 p.

73. *Beasley J. E., Whitchurch G.* O. R. education — a survey of young O. R. workers // Operational Research Society. — 1984. — N 35. —P. 281–288.

74. *Booch G.* Object-Oriented Analysis And Design With Application. — 2d ed. — New Jersey: Prentice Hall, 1994. — 589 p.

75. *Braek F., Haugen Th.* Engineering Real Time Systems. — S. l.: Prentice Hall International, 1993. — 398 p.

76. *Denning P. J., Buzen J. P.* Operational analysis of queueing network models // Computing Serveys. — 1978. — Vol. 10, N 3. — P. 225–261.

77. *Jacobson I.* Object-Oriented Software Engineering. — S. l.: ASM press., 1992. — 528 p.

78. *Johnson G. D.* Networked simulation with HLA and MODSIM III CACI Products Company // Proceedings of the 31st conference on Winter simulation: Simulation — a bridge to the future. — Vol. 2. Phoenix, 1999. — P. 106–1070.

79. *Kiviat P. J., Markowitz H. M., Villaneva R.*, SIMSCRIPT II.5 Programming Language / Ed. E. C. Russell. — Los Angeles: CACI. — 1997.

80. *Kleijnen J., Rubinstein R. Y.* Sensitivity Analysis by the Score Function Method // European Journal of Operations Research. — 1996. — Vol. 88. — P. 413–427.

81. *Kruchten P.* The Rational Unified Process: An Introduction. — S. l.: Addison-Wesley, 1998. — 255 p.

82. *Markowitz H. M., Bernard H., Karr H. W.* SIMSCRIPT, A Simulation Programming Language. — Englewood Cliffs: Prentice Hall, 1963.

83. *L'Ecuyer P.* Uniform random number generator // Proceedings of the 30th conference on Winter simulation. — Washington, 1998. — P. 97–104.

84. *Rumbaugh J., Blaha M., Premerlani W.* et al. Object-oriented modeling and design. — New Jersey: Prentice Hall. 1991. — 500 p.

85. *Rumbaugh J., Jacobson I., Booch G.* The Unified Modeling Language Reference Manual. — S. l.: Addison-Wesley, 1999. — 550 p.

86. *Schriber T. J., Brunner D. T.* Inside discrete-event simulation software: how it works and why it matters // Proceedings of the 1999 Winter Simulation Conference. — S. l., 1999. — P. 72–80.

87. *Selic B.* An Efficient Object-Oriented Variation of Statecharts Formalism for Distributed Real-Time System // CHDL'93: IFIP Conference on Hardware Description Languages and Their Applications, April 26–28. — Ottawa. — 1993.

88. *Selic B., Gullekson G., Ward P. T.* Real-Time Object-Oriented Modeling. — S. l.: John Wiley & Sons, 1994. — 525 p.

89. Survey of Languages and Runtime Libraries for Parallel Discrete-Event Simulation / *Yoke-Hean Low, Chu-Cheow Lim, Wentong Cai* et al. // SIMULATION. — 1999. — Vol. 72, N 3. — P. 170–186.
90. The DoD High Level Architecture: An Update / *J. S. Dahmann, R. M. Fujimoto, R. M. Weatherley* // Proceedings of the 30th conference on Winter simulation. — Washington, 1998. — P. 797–804.
91. *Tomashevskiy V.* Automatic generating of GPSS/PC programs for queueing network // Proceedings 15th European Simulation Multiconference. — Prague, 2001. — P. 203–208.
92. Visual Simulation Environment technology transfer / *O. Balci, A. I. Bertelrud, C. M. Esterbrook, R. E. Nance* // Proceedings of the 1998 Winter Simulation Conference. — N.Y.: ACM Press, 1997. — P. 1323–1329.
93. *Flodin A.* Full power with SDL and UML // Telelogic Signals. — 1998. — N 2. http://www.telelogic.com.
94. http://www.microTOOL.de/case.e.
95. Miniteman Software, GPSS WORLD REFERENCE MANUAL, Holly Springs, NC, U.S.A. http://www.minitemansoftware.com/reference, 2000.
96. OMG Unified modeling language specification (draft). Version 1.3R.
97. OMG Unified modeling language specification. Version 1.1. 1997 ().
98. *Selic B., Gullekson G., McGee J., Engelberg I.* ROOM: An Object-Oriented Methodology for Developing Real-Tie Systems. CASE'92 Fifth International Workshop on Computer-Aided Software Engineering, July 6-10, 1992, http://www.objectime.on.ca/. 11p.
99. *Selic B., Rumbaugh J.* Using UML for Modeling Complex Real-Time Systems. ObjecTime. http://www.objectime.on.ca/. 1998. 22 p.
100. www.orcacomputer.com\VSE, 1997.

Корисні посилання в Інтернет

http://www.scs.org/ — Міжнародне товариство комп'ютерного моделювання

http://eurosim.cashburn.at/ — Федерація Європейських товариств з моделювання

www.albany.edu/cpr/sds/ — Товариство системної динаміки

http://www.simulation.org.ua/ — український портал з імітаційного моделювання

http://www.gpss.ru/ — російський портал з імітаційного моделювання

http://www.efg2.com/Lab/Library/SimulationAndModeling.htm — найбільш загальні ресурси з моделювання

http://www.simulationinformation.com/ — національний центр США з моделювання

www.tbm.tudelft.nl/webstaf/edwinv/SimulationSoftware/Review_Simple.htm — загальна характеристика пакетів моделювання

http://www.statsoft.ru/home/textbook/default.htm — електронний підручник зі статистики

http://glspro.narod.ru/teach/index.html — підручник з імітаційного моделювання економічних процесів

http://ermak.cs.nstu.ru/~shalag/enter.html — підручник з моделювання

http://yevgeny.nm.ru/institut/model.html — конспект лекцій з дисципліни «Моделювання»

http://carbon.cudenver.edu/~hgreenbe/glossary/index.php — глосарій з математичного програмування

http://pespmc1.vub.ac.be/ASC/indexASC.html — сітьовий словник з кібернетики та системам

http://www.ntcnvg.ru/up.htm — навчальний посібник з прикладної математики

http://www.systems-thinking.org/index.htm — мнемонічні моделі роздумів

Алфавітний покажчик

К

М

Н

О

П

Р

С

Т

У

Ф

Ц

Ч

Ш

Навчальне видання

Томашевський Валентин Миколайович
МОДЕЛЮВАННЯ СИСТЕМ

Підручник

Керівник проекту В. П. Пасько
Редактор С. Г. Єзерницька
Коректор Н. М. Тонконог
Комп'ютерна верстка Д. С. Трішенкова

ТОВ «Видавнича група BHV»
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія ДК №175 від 13.09.2000 р.

Підписано до друку 09.08.05. Формат 70×100 1/16.
Папір офсетний. Гарнітура Petersburg. Друк офсетний.
Ум. друк. арк. 28,38. Обл.-вид. арк. 26,61.
Наклад 3000 прим. Зам. №30142.

Виготовлено в ТОВ «Освітня книга»,
м. Київ, вул. Орловська, 2/7, оф. 6.
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія ДК № 2245 від 26.07.2005 р.