МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Задачин В. М.
Конюшенко І. Г.

МОДЕЛЮВАННЯ СИСТЕМ

Конспект лекцій

Харків. Вид. ХНЕУ, 2010

УДК 004.415(042.4)
ББК 32.973я73
З-15

Рецензент – докт. екон. наук, доцент, завідувач кафедри вищої математики і економіко-математичних методів Харківського національного економічного університету *Малярець Л. М.*

Затверджено на засіданні кафедри інформаційних систем
Протокол № 5 від 08.02.2010 р.

Авторський колектив: Задачин В. М., канд. фіз.-мат. наук, доцент – вступ, теми 1 – 8; Конюшенко І. Г., ст. викладач – теми 9 – 16.

Задачин В. М.

З-15 Моделювання систем : конспект лекцій / В. М. Задачин, І. Г. Конюшенко. – Харків : Вид. ХНЕУ, 2010. – 268 с. (Укр. мов.)

Викладено основні підходи і принципи побудови й оцінки якості детермінованих та стохастичних моделей, основи імітаційного моделювання систем масового обслуговування з використанням мови GPSS.

Рекомендовано для студентів напряму підготовки "Комп'ютерні науки" спеціалізацій "Інформаційні управляючі системи та технології" і "Комп'ютерний еколого-економічний моніторинг".

УДК 004.415(042.4)
ББК 32.973я73

Вступ

"Моделювання систем" є нормативною навчальною дисципліною з підготовки студентів за напрямом підготовки "Комп'ютерні науки".

Метою навчальної дисципліни "Моделювання систем" є вивчення студентами основних підходів і принципів побудови моделей та придбання навичок застосування їх для розв'язання задач моделювання, що виникають при розробці чи дослідженні комп'ютерних інформаційних й інших систем.

Теоретичною базою вивчення дисципліни "Моделювання систем" є попередні навчальні дисципліни: "Вища математика", "Дискретна математика", "Теорія ймовірності і математична статистика", "Чисельні методи в інформатиці", "Програмування" та ін.

Знання з навчальної дисципліни "Моделювання систем" застосовуються практично при вивченні всіх навчальних дисциплін, що викладаються пізніше.

Навчальне завдання дисципліни:

- вивчення типових математичних схем моделювання систем;
- вивчення статистичного моделювання систем на ЕОМ;
- вивчення сучасних способів моделювання складних інформаційних систем;
- вивчення практичних підходів до моделювання систем (технології моделювання);
- ознайомлення з основними мовами імітаційного моделювання систем;
- оволодіння методами імітаційного моделювання із застосуванням пакета GPSS World.

Після вивчення навчальної дисципліни "Моделювання систем" студенти повинні отримати професійні компетенції:

- розробки концептуальних та теоретичних моделей розв'язуваних наукових проблем і задач;
- поглибленого аналізу проблем, постановки та обґрунтування задач наукової і проектно-технологічної діяльності;
- розробки та оптимізації бізнес-планів науково-прикладних проектів;

- знання методології та сучасних технологій моделювання;
- знання методів проектування моделей складних систем;
- знання щодо систем масового обслуговування;
- знання та вміння щодо планування та проведення імітаційних експериментів;
- знання принципів побудови засобів імітаційного моделювання;
- вміння будувати імітаційні моделі з використанням математичного пакета MathCad та системи імітаційного моделювання GPSS World;
- вміння оцінювати якість побудованих моделей;
- спроможності приймати рішення за результатами моделювання.

Основою даного конспекту лекцій є матеріал, що був підібраний і використаний у 2005 – 2009 навчальних роках при проведенні теоретичних занять з навчальної дисципліни "Моделювання систем" на факультеті економічної інформатики в Харківському національному економічному університеті для студентів спеціалізацій "Інформаційні управляючі системи та технології" і "Комп'ютерний еколого-економічний моніторинг".

У першому змістовному модулі "Моделювання як наука" містяться основні поняття теорії систем та загальновизнані теоретичні і практичні підходи до моделювання, а другий змістовний модуль – Моделювання як мистецтво – присвячений вивченню спеціалізованої мови для імітаційного моделювання GPSS та оволодінню методами побудови й аналізу якості моделей із застосуванням пакета GPSS World.

Цей конспект лекцій призначено для студентів, які навчаються за напрямом підготовки "Комп'ютерні науки". Він відповідає плану робочої програми з навчальної дисципліни "Моделювання систем" [9].

Модуль 1. Моделювання як наука
Тема 1. Вступ. Предмет дисципліни, її зміст та завдання

1.1. Вступ у теорію моделювання

Моделювання як одну з найважливіших категорій процесу пізнання неможливо відокремити від розвитку людства. З самого дитинства людина пізнає світ, спочатку через іграшки й ігри, відображаючи, тобто моделюючи, дійсність. З роками вона використовує все більш складні моделі, які дають можливість "програвати" різні життєві та виробничі ситуації і тим самим отримувати якнайкращі способи вирішення проблеми [12].

Математичні моделі є одним з основних інструментів пізнання людиною явищ навколишнього світу. Під математичними моделями розуміють основні закономірності і зв'язки, властиві явищу, що вивчається. Це можуть бути формули або рівняння, набори правил або угод, виражені в математичній формі. Так, наприклад, закони Ньютона повністю визначають закономірності руху планет навколо Сонця. Використовуючи основні закони механіки, відносно неважко скласти рівняння, що описують рух космічного апарата, наприклад, від Землі до Місяця. Проте отримати їх розв'язання (чи розв'язок) у вигляді простих формул не є можливим. Тому для розрахунку траєкторій космічних апаратів служать комп'ютери, тобто застосовується комп'ютерне моделювання.

Методи комп'ютерного моделювання широко використовують у всіх сферах діяльності людини – від конструювання моделей технічних, технологічних й організаційних систем до вирішення проблем розвитку людства і навколишнього світу. Класичними об'єктами моделювання є інформаційні, виробничі, транспортні й інші логістичні системи, моделі яких у більшості випадків використовуються для розв'язання задач проектування, реконструкції і довгострокового планування, тобто застосовуються для управління цими системами. Комп'ютерне моделювання повинне застосовуватися завжди, коли потрібно відповісти на запитання: "Що буде, якщо...?", тобто при прийнятті рішення [12].

У розвинених країнах перед інвестуванням коштів у будь-який проект можливості його реалізації перевіряються на імітаційних моделях. Практично всі транснаціональні компанії мають моделі розвитку вироб-

ництва, більш того, вони вкладають значні кошти у дослідження цих моделей. Наприклад, в автомобільній промисловості Німеччини приймають до розгляду технічну документацію тільки за умови її відповідності концепції Digital 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 де монстрація 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 МоскUp тощо [12].

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 й оцінки пропускної спроможності систем. Однак розробка і застосування імітаційних моделей інформаційних систем – це непрості завдання. Етап формулювання абстрактної моделі та етап конструювання моделі часто включають тривалі й дорогі процедури. Абстрактна модель інформаційної системи зазвичай створюється фахівцем із моделювання, який може отримувати знання у потрібній галузі від проектувальників та 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) чи 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 нетечностей не тільки під час побудови моделі, але й під час програмування [12].

Моделювання – складний процес, що потребує багато часу, незважаючи на те, йде мова про окремого фахівця з моделювання чи цілої групи фахівців, упродовж роботи якої потрібні постійний зв'язок і координація. Зазначені причини виправдовують зусилля, докладені для роз-

робки методів, що допомагають прискорити процес моделювання шляхом автоматизації деяких процесів. Сучасні програмні засоби моделювання використовують графічний інтерфейс і дво- або тривимірну анімацію, що значно полегшує сприйняття результатів моделювання несеціалістом.

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 сфери 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 [12]:

– 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 й організації робіт з реалізації та 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 й алгоритмізації;

– вміти вибирати та використовувати методи математичного моделювання при проектуванні та експлуатації складних систем управлін-

ня, 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 у 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-eurgope.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.Ье/~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, Імперіалколеджі, Університеті Варвік, Університеті Ланкастера, Саутамптонському університеті тощо. Американське і європейські товариства моделювання регулярно проводять конференції та публікують їх матеріали [12].

На жаль, активне застосування методів імітаційного моделювання за кордоном не викликало поки що значного його поширення у нашій країні. Пояснити це, 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 [12].

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 конспекту лекцій з навчальної дисципліни "Моделювання систем" – 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.

З усіх видів моделювання – а це перш за все математичне і графічне – 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 [12].

У курсі лекцій обговорюється 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 курсі лекцій основна увага приділяється саме цій мові.

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

1.2. Предмет, завдання та зміст дисципліни

Об'єктом вивчення навчальної дисципліни "Моделювання систем" є різні (технічні, фізичні та ін.) системи (явища, процеси), з якими пов'язана людська діяльність.

Предметом вивчення навчальної дисципліни "Моделювання систем" є загальновизнані методології і сучасні технології моделювання складних систем.

Навчальне завдання дисципліни: вивчення практичних підходів до моделювання систем, оволодіння методами імітаційного моделювання із застосуванням пакета GPSS World.

Після вивчення дисципліни студенти повинні:

знати:

- принципи моделювання та класифікацію способів представлення моделей систем;
- методологію та сучасну технологію моделювання;
- методи проектування моделей складних систем;
- теоретичний матеріал щодо систем масового обслуговування;
- способи планування та проведення імітаційних експериментів з моделями;
- принципи побудови засобів імітаційного моделювання;

вміти:

- застосовувати технологію моделювання;
- представити модель в математичному та алгоритмічному вигляді;
- оцінити якість моделі;
- проводити статистичне моделювання систем;
- моделювати процеси, що протікають в інформаційних системах та мережах;
- будувати імітаційні моделі з використанням математичних пакетів GPSS World та MathCad;

придбати навички:

- побудови імітаційних моделей;
- отримання концептуальних моделей систем;
- побудови моделюючих алгоритмів;
- програмування в системі моделювання GPSS World.

Форми проведення занять: лекції, лабораторні.

Форми контролю:

– поточний контроль – у формі виконання лабораторних робіт;

– модульний контроль – у формі письмової роботи 2 рази за семестр;

– підсумковий контроль – іспит у формі евристичного завдання із використанням ЕОМ;

– підсумкова оцінка складається з результатів іспиту, модульного контролю та поточного контролю.

Теми для вивчення:

Модуль 1. Моделювання як наука

Тема 1. Вступ. Предмет дисципліни, її зміст та завдання.

Тема 2. Моделювання. Основні поняття. Види моделей, їх класифікація. Вимоги до моделей.

Тема 3. Основні види моделювання. Формальні методи побудови моделей.

Тема 4. Ідентифікація параметрів математичної моделі. Адекватність, чутливість, несуперечливість моделі.

Тема 5. Принципи побудови моделей. Технологія моделювання.

Тема 6. Моделі розрахункових процесів і управління. Динамічні моделі, P-, Q-, F-, A-схеми. Мережні моделі.

Тема 7. Імовірнісне моделювання. Моделювання випадкових процесів.

Тема 8. Моделі теорії черг.

Модуль 2. Моделювання як мистецтво

Тема 9. Поняття імітаційного моделювання. Моделі систем масового обслуговування. Принципи роботи GPSS World. Елементи логіки роботи інтерпретатора.

Тема 10. Блоки, що забезпечують побудову моделі типу "СМО з одним пристроєм". Забезпечення пріоритетного обслуговування.

Тема 11. Блоки, що забезпечують побудову моделі типу "багатоканальні СМО". Засоби GPSS World, що використовуються для забезпечення точності результатів імітаційного моделювання.

Тема 12. Стандартні числові й логічні атрибути та їх використання в моделях. Функції в GPSS World. Їх використання в моделях.

Тема 13. Збережувані величини і матриці. Змінні та вирази. Зміна маршрутів транзактів.

Тема 14. Використання таблиць у GPSS World.

Тема 15. Списки користувача та блоки для їх формування. Групи і сімейства транзактів.

Тема 16. Сучасний стан імітаційного моделювання. Основні сфери використання імітаційних моделей.

Тема 2. Моделювання. Основні поняття. Види моделей, їх класифікація. Вимоги до моделей

2.1. Поняття моделювання

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

На сьогоднішній день моделюванню приділяється значна увага. Невипадково один з найпотужніших у світі суперкомп'ютер NEC Vector SX6 (Earth-Simulator), за даними рейтингу Top500 (www://top500.org), встановлений у центрі моделювання Землі в Йокогамі (Японія). Цей комп'ютер призначений 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.

Розглянемо основні поняття загальної теорії систем і моделювання [12].

2.2. Поняття системи

Основними поняттями в теорії і практиці моделювання об'єктів, процесів і явищ є поняття "система" і "модель".

У перекладі з грецької "systema" – це ціле, яке складається з частин; об'єднання. Термін "система" існує вже більш ніж два тисячоліття, проте, різні дослідники визначають його по-різному. На сьогодні існує понад 500 визначень терміна "система". Проте, використовуючи будь-яке з них, насамперед потрібно мати на увазі ті завдання, які ставить перед собою дослідник. Системою може бути і один комп'ютер, і автоматична лінія або технологічний процес, в яких комп'ютер є лише одним з компонентів, і все підприємство або декілька різних підприємств, що функціонують як єдина система в одній галузі промисловості. Те, що один дослідник визначає як систему, для іншого може бути лише компонентом складнішої системи.

Для всіх визначень системи спільним є те, що ***система*** – цілісний комплекс взаємозв'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 елементу 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 зовнішнє середовище відсутнє або не враховується, у зв'язку з поставленими цілями досліджень.

Одне з перших визначень системи (1950 рік) належить американському біологові Л. фон Берталанфі, згідно з яким система складається з деякої кількості взаємозв'язаних елементів. Оскільки між елементами системи існують певні взаємозв'язки, то повинні 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 як може бути системою, так і не бути нею.

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

Визначення поняття системи пов'язані з абстрактною теорією систем, в рамках якої використовуються такі рівні абстрактного опису:

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

Найвищий рівень абстрактного опису систем – ***лінгвістичний***; ґрунтуючись на ньому, можна отримати всі інші рівні. На цьому рівні вводиться поняття предметної області, для опису якої застосовуються моделі алгебри, пов'язані з деякою мовою. Для опису предметної області цією мовою використовуються два рівні формальних мов, за допомогою яких будують логіко-алгебраїчну модель предметної області. На цій моделі підтверджуються дослідницькі прийоми за допомогою нормального апарату, яким можуть бути теорії, побудовані у вигляді дійсних висловлювань з всієї 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, означає будь-яке речення (формулу), побудоване за граматичними правилами цієї мови. Допускають, що таке речення містить змінні, що підбираються, так звані ***конституенти***, які, маючи тільки певні значення, роблять дане висловлювання істинним.

Якщо існує множина висловлювань G, але лише V з них істинні, то вважають, що має місце теорія L щодо множини 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. Усі 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 можливість показати, як, базуючись на лінгвістичному рівні, можна утворити інші рівні абстрактного опису системи.

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

При подальшому викладенні змісту курсу лекцій користуватимемося ***теоретико-множинним*** визначенням системи (А. Холл, Р. Фейджин і Ф. Фейджин), згідно з яким ***система*** – це множина об'єктів, між якими існують певні відношення, а також їх атрибути. Під ***об'єктами*** розуміють компоненти (елементи) 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 же множині об'єктів. Згідно з цим визначенням поняття ***множина*** і ***елемент*** є аксіоматичними.

Таким чином, система S задається парою елементів:

$$S = (X_S, R_S),$$

де X_S, R_S – множини відповідно елементів (об'єктів) системи і відношень між ними. Відношення визначають взаємодію між об'єктами.

У загальному випадку n-відношення R в множинах $X_1, X_2, \ldots, X_n$ є деякою підмножиною декартового добутку[1] $X_1 \times X_2 \times \ldots \times X_n$, який складений з n-вимірних наборів виду $(x_1, x_2, \ldots, x_n)$, де $x_i \in X_i$, $i = 1,2,\ldots,n$.

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

Що стосується атрибутів системи, то вони подібні до функцій, визначених на підмножині об'єктів. Відмінність атрибутів від функцій полягає в тому, що два різних атрибути з точки зору 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 спроможності ділянок автомобільних 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, гальмівний шлях автомобіля змінюється залежно від швидкості руху і погодних умов, а прискорення може бути додатнім (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 одним з елементів більшої системи, тобто може існувати ієрархія систем. Наприклад, двигун є підсистемою автомобіля, який, у свою чергу, є підсистемою транспортного потоку магістралі.

---

[1] Декартовим добутком множин $A \times B$ називається сукупність будь-яких пар виду (a, b), де $a \in A$, $b \in B$, тобто $A \times B = \{(a,b) : a \in A, b \in B\}$

На теоретико-множинному рівні абстрактного опису системи можна отримувати досить таки загальні відомості про реальні системи, а для конкретних цілей потрібні інші моделі, які давали б можливість детальніше 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 даних множин встановлюються за допомогою деяких однозначних функцій, які відображають елементи множини в саму початкову множину, то має місце ***абстрактно-алгебраїчний*** рівень опису систем. У таких випадках вважають, що між елементами множини встановлені нульарні, унарні, бінарні, тернарні й інші відношення.

Якщо ж на даних множинах визначені деякі багатозначні функції, то мають місце ***топологічні абстрактні*** моделі, записані мовою загальної топології або її гілок, які називаються топологією алгебри, гомологічною топологією і т. п.

Вибір потрібного рівня абстрактного опису при вивченні тієї або іншої реальної системи є завжди найбільш відповідальним і найбільш важким кроком у теоретико-системних побудовах. Цей процес майже не піддається формалізації і багато в чому залежить від досвіду і знань дослідника, його професійної підготовки, цілей дослідження і т. п.

Можна показати, як від систем з топологічним рівнем опису перейти до узагальнених динамічних систем. Щоб дати строге математичне визначення поняттю ***динамічна система***, її наділяють властивістю мати "входи" і "виходи", тобто визначають як структурований об'єкт, куди в певні моменти часу можна вводити речовину, енергію, інформацію, а в інші моменти – виводити їх. Динамічні системи можна зобразити і як системи, де процеси відбуваються неперервно, і як системи, в яких всі 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 в кінці 50-х – на початку 60-х років XX століття, пов'язаний з розробкою 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 управління, самоорганізації, навчання і тому подібне, 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 відображувала б всі фактори, взаємозв'язки і реальні ситуації. Займаються цим фахівці з системного аналізу – системні аналітики.

2.3. Поняття моделі

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

Основним видом кількісної аналогії є ***математична схожість***, коли об'єкти описуються за допомогою рівнянь і функцій. Функції і незалежні змінні називаються схожими, якщо вони співпадають з точністю до деяких констант. Окремими видами математичної схожості є ***геометрична схожість***, яка встановлює схожість геометричних образів, і ***часова***, така, що визначає схожість функцій часу, для яких константа часу (масштаб) показує, в яких відношеннях перебувають параметри функцій, такі як період, часова затримка і т. п.

Іншим видом кількісної аналогії є ***фізична схожість***. Критерії фізичної схожості можна отримати, не маючи математичного опису об'єктів, наприклад, на основі значень фізичних параметрів, які характеризують досліджуваний процес у натурі і на моделі. За типами процесу розрізняють види схожості, для якої розроблені відповідні критерії, – гідравлічні, електричні, аеродинамічні й ін.

Вивчення переходу від властивостей реальних об'єктів до властивостей системи є найважливішим завданням теорії систем. У загальній теорії систем визнається об'єктивність існування систем. 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, який формулюється таким чином.

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[2].

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 основою 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 відносно її система. При цьому ізоморфізм (схожість) двох систем стосується і 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 схожості дає можливість встановити відношення еквівалентності (відповідності, схожості) за деякими ознаками між двома системами, що розглядаються. Будь-яка з цих систем може існувати реально або бути абстрактною. Якщо система існує реально, то її можна вивчати, досліджуючи, яким чином зв'язані вхідні впливи з виходами системи. На основі результатів досліджень будується деяка ***абстрактна система***. У ній відношення еквівалентності визначається тільки для тих важливих властивостей і аспектів поведінки, які в початковій та в абстрактній системах повинні бути однаковими. Г. Месарович відзначає, що, базуючись на спостереженнях і дослідженнях однієї системи, можна будувати висновки про властивості й поведінку іншої. Переважно на практиці абстрактна система є ***більш простою***, ніж початкова, якщо не враховувати тих аспектів, які визначають відношення еквівалентності.

---

[2] Дві множини X, Y називаються ***ізоморфними***, якщо між елементами цих множин можна встановити взаємно однозначну відповідність

Таким чином, можна перейти до визначення терміна "модель". У філософській літературі під терміном "модель" розуміють "деяку реально існуючу систему або ту, що представляється в думках, яка, заміщуючи і відображаючи в пізнавальних процесах іншу систему-оригінал, перебуває з нею у відношенні схожості, завдяки чому вивчення моделі дає можливість отримати нову інформацію про оригінал". У цьому визначенні закладений генетичний зв'язок моделювання з теорією схожості, принципом аналогії. Таким чином, моделлю можна називати систему, яку використовують для дослідження іншої системи [12].

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

Підсумовуючи вищесказане, надалі використовуватимемо таке коротке визначення. **Модель** – це реально існуюча або абстрактна система, яка, замінюючи і відображаючи в пізнавальних процесах іншу систему – оригінал, перебуває з нею у відношенні схожості.

У сучасній теорії управління використовуються ***моделі двох основних типів***. Для технологічних об'єктів цей поділ відповідає "***феноменологічним***" і "***дедуктивним***" моделям [12]. Під феноменологічними моделями розуміють переважно емпірично відновлені залежності вихідних даних від вхідних, як правило, з невеликою кількістю входів і виходів. Дедуктивне моделювання передбачає з'ясування і опис основних фізичних закономірностей функціонування всіх компонентів досліджуваного процесу і механізмів їх взаємодії. За допомогою де дуктивних моделей описується процес у цілому, а не окремі його режими.

Перший тип моделей – ***моделі даних***, які не мають потреби, не використовують і не відображають яких-небудь гіпотез про фізичні процеси або системи, з яких ці дані отримані. До моделей даних належать всі моделі математичної статистики. Останнім часом ця сфера моделювання ув'язується з експериментально-статистичними методами і системами, що істотно розширює методологічну базу для прийняття рішень під час розв’язання задач аналізу даних і управління.

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 та ін.). Саме системні моделі є 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 ним можна визначити сукупність властивостей модельованої системи, які повинна мати і модель, тобто від мети моделювання залежить потрібний ступінь деталізації моделі.

2.4. Співвідношення між моделлю та системою

Ураховуючи 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 перебувають у деяких співвідношеннях, від яких залежить ступінь відповідності між ними. На міру відповідності між системою і моделлю вказують поняття ***ізоморфізму*** і ***гомоморфізму***. Система і модель є ізоморфними, якщо існує взаємно однозначна відповідність між ними, завдяки якій можна перетворити одне подання на інше. Строго доведений ізоморфізм для систем різної природи дає можливість переносити знання з однієї області 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 і модель є ізоморфними тільки у разі ***спрощення*** системи, тобто скорочення множини її властивостей (атрибутів) і характеристик поведінки, які впливають на простір станів системи.

Станом динамічної системи (моделі) в деякий момент часу t називається множина значень всіх її параметрів (змінних), виміряних

одночасно у цей момент. При зміні значення хоча б одного параметра системи в наступний момент часу говорять, що стан системи змінився. Стан системи зручно розглядати як точку в багатовимірному просторі. Множина всіх можливих станів системи називається ***простором станів системи***.

Зазвичай модель є більш простою, ніж система. На рис. 2.1 схематично зображена відмінність ізоморфної і гомоморфної залежностей між системою і моделлю для просторових станів системи Z_s і моделі Z_m. Множина станів моделі Z_m визначають, враховуючи мету моделювання і вибраний рівень абстрактного опису.

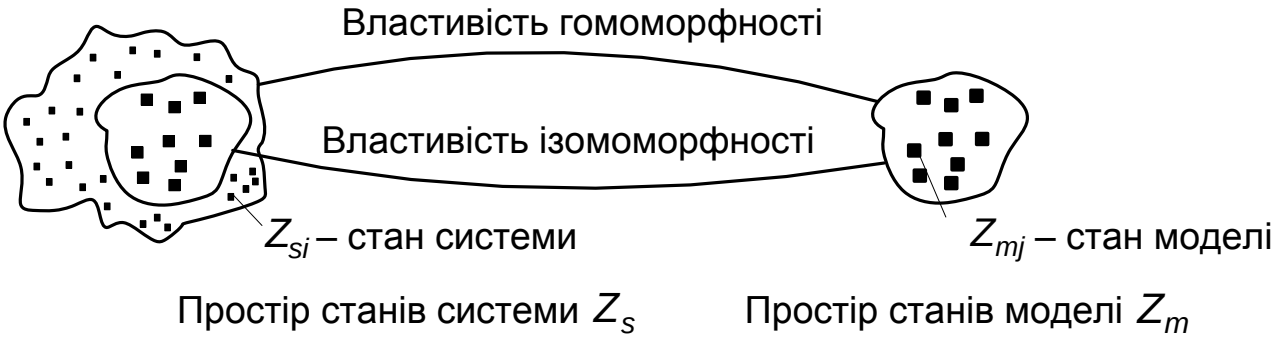


Рис. 2.1. **Схематичне зображення співвідношення між системою і моделлю**

Отже, ***аналогія, абстракція і спрощення*** – це основні поняття, які використовуються при моделюванні систем. Розглянемо відношення між системою і моделлю, враховуючи, що ці відношення відповідають цілям моделювання й обмеженням досліджуваної системи. При використанні 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 і навпаки:

$$P(Z_m = Z_{mj} \mid Z_s = Z_{si}) = P(Z_s = Z_{si} \mid Z_m = Z_{mj}) = 0 \vee 1,$$

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

У цьому випадку 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. Вказані відношення для конкретних станів Z_{si}, Z_{mj} можна записати у такому вигляді:

$$P(Z_m = Z_{mj} \mid Z_s = Z_{si}) = 0 \vee 1,$$

$$P(Z_s = Z_{si} \mid Z_m = Z_{mj}) \le 1,$$

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

3. **Імовірнісні відношення з нескінченною множиною станів**, коли стани системи і моделі визначають стани один одного лише з деякою ймовірністю:

$$P(Z_m = Z_{mj} \mid Z_s = Z_{si}) \le 1,$$

$$P(Z_s = Z_{si} \mid Z_m = Z_{mj}) \le 1.$$

Це так звані стохастичні моделі, до яких, наприклад, належать марківські моделі (ланцюги Маркова) і моделі систем масового обслуговування.

2.5. Види моделей та їх класифікація за різними критеріями

Для того щоб визначити види моделей, перш за все, потрібно вказати ознаки класифікації.

Якщо враховувати, що моделювання – це метод пізнання дійсності, то основною ознакою класифікації можна назвати ***спосіб подання*** моделі. За цією ознакою розрізняють ***абстрактні і реальні*** моделі (рис. 2.2). Під час моделювання можливі різні абстрактні ***конструкції,***

проте, основною є віртуальна (**уявна**) модель, що відображає ідеальне уявлення людини про навколишній світ, який фіксується у свідомості через думки і образи. Віртуальна модель може представлятися у вигляді наглядної ***моделі*** за допомогою графічних образів і зображень.

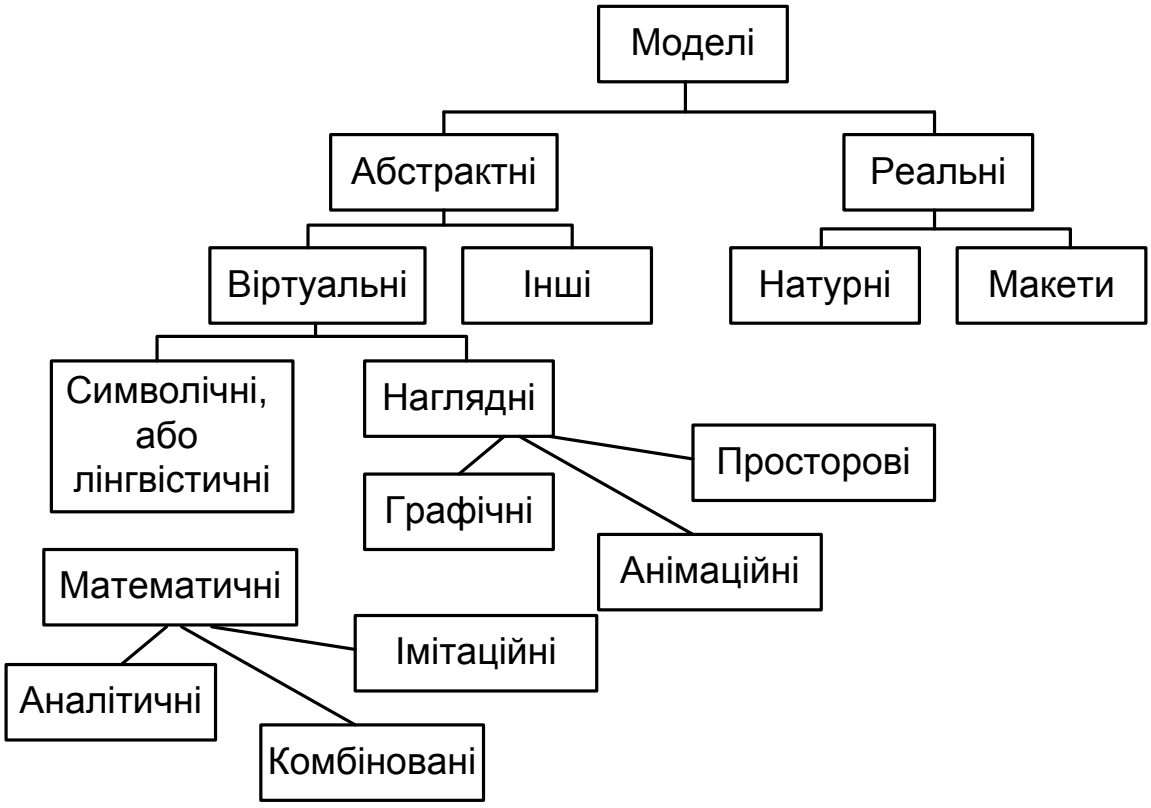


Рис. 2.2. **Основні типи моделей**

Наглядні моделі залежно від способу реалізації можна поділити на дво- або тривимірні графічні, анімаційні і просторові. Графічні й анімаційні моделі широко використовуються для відображення процесів, які відбуваються в модельованій системі. Графічні моделі застосовуються в системах автоматизованого проектування (computer-aided design, CA). Для відтворення тривимірних 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 високому рівню абстрактного опису, як це було вказано вище. На базі її отримують інші рівні опису.

Основним видом абстрактної моделі є ***математична модель***. Її вид залежить як від природи реального об'єкта, так і від задач дослідження об'єкта та необхідної достовірності і точності розв'язку цієї задачі. Будь-яка математична модель, як і всяка інша, описує реальний об'єкт лише з деякою мірою наближення до дійсності. За видом математичні моделі для дослідження характеристик процесу функціонування систем можна розділити на ***аналітичні***, ***імітаційні*** і ***комбіновані***.

Для ***аналітичної моделі*** характерно те, що процеси функціонування елементів системи записуються у вигляді деяких функціональних співвідношень (алгебрі, інтегрально-диференціальних, кінцево-різницевих і т. п.) або логічних умов. Аналітична модель може бути досліджена такими методами:

а) аналітичним, коли прагнуть отримати в загальному вигляді явні залежності для шуканих характеристик;

б) чисельним, коли, не вміючи розв'язувати рівняння в загальному вигляді, прагнуть отримати числові результати при конкретних початкових даних;

в) якісним, коли, не маючи розв'язку в явному вигляді, можна знайти деякі властивості розв'язку (наприклад, оцінити сталість розв'язку).

Якнайповніше дослідження процесу функціонування системи можна провести, якщо відомі явні залежності, що пов'язують шукані характеристики з початковими умовами, параметрами і змінними системи S. Проте такі залежності вдається отримати тільки для порівняно простих систем. При ускладненні систем дослідження їх аналітичним методом наштовхується на значні труднощі, які часто бувають нездоланними. Тому, бажаючи використовувати аналітичний метод, в цьому випадку йдуть на суттєве спрощення початкової моделі, аби мати можливість вивчити хоча б загальні властивості системи. Таке дослідження на спрощеній моделі аналітичним методом допомагає отримати орієнтовні результати для визначення точніших оцінок іншими методами. Чисельний метод дозволяє досліджувати порівняно з аналітичним методом ширший клас систем, але при цьому отримані розв'язки носять приватний характер. Чисельний метод особливо ефективний при використанні комп'ютерів.

В окремих випадках дослідника системи можуть задовольнити і ті висновки, які можна зробити при використанні якісного методу аналізу математичної моделі. Такі якісні методи широко використовуються,

наприклад, в теорії автоматичного управління для оцінки ефективності різних варіантів систем управління.

В ***імітаційній моделі*** відтворюється процес функціонування системи S у часі, причому імітуються елементарні явища, що складають процес, із збереженням їх логічної структури і послідовності протікання в часі, що дозволяє за початковими даними отримати зведення про стани процесу в певні моменти часу, які дають можливість оцінити характеристики системи S.

Основною перевагою використання імітаційних моделей порівняно з аналітичними моделями є можливість розв'язання складніших задач. Імітаційні моделі дозволяють досить просто враховувати такі фактори, як наявність дискретних і безперервних елементів, нелінійні характеристики елементів системи, численні випадкові дії тощо, які часто створюють труднощі при аналітичних дослідженнях. Нині імітаційне моделювання – найбільш ефективний метод дослідження великих систем, а часто і єдиний практично доступний метод отримання інформації про поведінку системи, особливо на етапі її проектування.

Коли результати, отримані при відтворенні на імітаційній моделі процесу функціонування системи S, є реалізаціями випадкових величин і функцій, тоді для знаходження характеристик процесу потрібне його багаторазове відтворення з подальшою статистичною обробкою інформації і доцільно як метод машинної реалізації імітаційної моделі використовувати метод статистичного моделювання. Спочатку був розроблений метод статистичних випробувань, що є чисельним методом, який застосовувався для моделювання випадкових величин і функцій, імовірнісні характеристики яких співпадали з розв'язками аналітичних задач (така процедура отримала назву ***метода Монте-Карло***). Потім цей прийом почали застосовувати і для машинної імітації з метою дослідження характеристик процесів функціонування систем, схильних до випадкових дій, тобто з'явився метод ***статистичного моделювання***.

Таким чином, методом статистичного моделювання надалі називатимемо метод машинної реалізації імітаційної моделі, а методом статистичних випробувань (Монте-Карло) називатимемо чисельний метод розв'язання аналітичних задач.

Метод імітаційного моделювання дозволяє розв’язувати задачі аналізу великих систем S, включаючи задачі оцінки: варіантів структури системи, ефективності різних алгоритмів управління системою, впливу

зміни різних параметрів системи. Імітаційне моделювання може бути покладене також в основу структурного, алгоритмічного і параметричного синтезу великих систем, коли потрібно створити систему із заданими характеристиками при певних обмеженнях, яка є оптимальною за деякими критеріями оцінки ефективності.

Використання ***комбінованих*** (аналітико-імітаційних) ***моделей*** при аналізі і синтезі систем дозволяє об'єднати переваги аналітичних й імітаційних моделей. При побудові комбінованих моделей проводиться попередня декомпозиція процесу функціонування об'єкта на складові підпроцеси, і для тих з них, де це можливо, використовуються аналітичні моделі, а для решти підпроцесів будуються імітаційні моделі. Такий комбінований підхід дозволяє охопити якісно нові класи систем, які не можуть бути досліджені з використанням тільки аналітичного й імітаційного моделювання окремо.

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

Натурні (фізичні) моделі – це існуючі системи (або їх частини), на яких ведуться дослідження. Натурні моделі повністю адекватні реальній системі, що дає можливість отримувати високу точність і достовірність результаттів моделювання. Істотні недоліки натурних моделей – це неможливість моделювання критичних й аварійних режимів їх роботи і висока вартість.

Макетні моделі – це реально існуючі моделі, що відтворюють модельовану систему в певному масштабі. Іноді такі моделі називаються ***масштабними***. Параметри моделі і системи відрізняються між собою. Числове значення цієї відмінності називається масштабом моделювання, або коефіцієнтом схожості. Ці моделі розглядаються в рамках теорії схожості, яка в окремих випадках передбачає геометричну схожість оригінала і моделі для відповідних масштабів параметрів. Прості 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.

Залежно від можливості змінювати в часі свої властивості моделі поділяються на ***статичні*** і ***динамічні***. Статичні моделі, на відміну від динамічних, не змінюють своїх властивостей в часі. Динамічні моделі, як правило, є імітаційними.

Залежно від того, яким чином відтворюються в часі стани моделі, розрізняють ***дискретні***, ***неперервні*** і ***дискретно-неперервні*** (комбіновані) моделі.

Відповідно до співвідношень між станами системи і моделі розрізняють ***детерміновані*** і ***стохастичні*** моделі. Останні, на відміну від детермінованих моделей, враховують імовірнісні явища і процеси, що відбуваються в системі.

Поняття складної системи

Теорія відносності, яка вивчає універсальні фізичні закономірності, що відносяться до всього Всесвіту, і квантова механіка, яка вивчає закони мікросвіту, нелегкі для розуміння, і, тим не менше, вони мають справу з системами, які з погляду сучасного природознавства вважаються простими. **Простими** в тому сенсі, що в них входить невелика кількість змінних, і тому взаємовідношення між ними піддається математичній обробці і виведенню універсальних законів. Однак, крім простих, існують **складні системи**, які складаються з великого числа змінних і, отже, великої кількості різних зв'язків між ними. Чим воно більше, тим важче піддається предмет дослідження досягненню кінцевого результату – виведенню закономірностей функціонування даного об'єкта. Труднощі вивчення даних систем пов'язані і з тією обставиною, що чим складніше система, тим більше у неї так званих **емерджентних** властивостей, тобто властивостей, яких немає у її частин і які є наслідком ефекту цілісності системи.

Такі складні системи вивчає, наприклад, метеорологія – наука про кліматичні процеси. Метеорологія вивчає саме складні системи, оскільки процеси утворення погоди набагато менш відомі, ніж гравітаційні процеси, що, на перший погляд, здається парадоксом. Дійсно, чому ми достатньо точно можемо визначити, в якій точці перебуватиме Земля або яке-небудь інше небесне тіло через мільйони років, але не можемо точно передбачити погоду на завтра? Тому що кліматичні процеси є набагато складнішими системами, що складаються з величезної кількості змінних і взаємодій між ними.

2.6. Вимоги до моделей

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

– ***незалежність результатів*** розв'язання задач від конкретної фізичної інтерпретації елементів моделі;

– ***змістовність***, тобто здатність моделі відображати важливі риси і властивості реального процесу, який вивчається і моделюється;

– ***дедуктивність***, тобто можливість конструктивного використання моделі для отримання результату (управління, прогнозування);

– ***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 впевнений, що не отримає абсурдних результатів, а всі отримані результати відображатимуть необхідні для дослідника характеристики і властивості модельованої системи. Модель повинна дати можливість знайти відповіді на певні питання, наприклад: "що буде, якщо ...", оскільки вони є найбільш доцільними під час глибокого вивчення проблеми. Не слід забувати, що системні аналітики використовують модель для прийняття рішень і пошуку якнайкращих 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 повинна бути автоматизована для замовника до такої міри, щоб він міг працювати з нею в межах своєї предметної області. Таким чином, між моделлю і користувачем має бути реалізований розвинений інтерфейс, який зазвичай створюється за допомогою системи меню, налаштованої на використання моделі в певній області [12].

Ступінь деталізації моделі потрібно вибирати з урахуванням цілей моделювання, можливості отримання необхідних вхідних даних для моделі і враховуючи наявні ресурси для її створення. Відсутність кваліфікованих фахівців може звести роботи зі створення моделі нанівець. З іншого боку, чим детальніше розроблена модель, тим вона стійкіша до вхідних впливів, які не були передбачені під час проектування, і на більшу кількість питань може дати правильні відповіді.

Висновки

1. Система – це цілісний комплекс взаємопов'язаних елементів, який має певну структуру і взаємодіє із зовнішнім середовищем.

2. Модель – це реально існуюча або уявна система, яка, заміщаючи і відображаючи в пізнавальних процесах іншу систему-оригінал, знаходиться з нею у відношенні подібності.

3. Моделювання – це спосіб дослідження будь-яких явищ, процесів або об'єктів шляхом побудови та аналізу їх моделей.

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;

c) станція швидкої допомоги;

d) кафе;

e) 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. Виконайте критичний аналіз різних видів класифікацій моделей. Чому неможлива єдина класифікація? Запропонуйте іншу класифікацію моделей.

Тема 3. Основні види моделювання. Формальні методи побудови моделей

3.1. Основні види моделювання

Єдина класифікація видів моделювання неможлива через багатозначність поняття моделі в науці, техніці, суспільстві. Найбільш широко відомими видами моделювання є ***математичне*** (аналітичне), ***імітa-***

ційне і ***статистичне***. На жаль, різні джерела по-різному трактують ці поняття.

Для ***аналітичного (математичного) моделювання*** характерне те, що процеси функціонування елементів системи записуються у вигляді деяких функціональних співвідношень. При цьому слід зазначити, що під час використання аналітичних моделей багато що залежить від способу подання як моделі, так і результатів моделювання.

Розглянемо простий приклад. Нехай на деякому підприємстві для водопостачання використовується резервуар, об'єм якого W тисяч літрів. Рівень споживання – V_C тисяч літрів на добу, а швидкість заповнення резервуара – V_3 тисяч літрів на добу. Необхідно знайти час T, за який буде заповнений резервуар. Схема цієї системи зображена на рис. 3.1, де резервуар позначений прямокутником, а вхідний і вихідний потоки – стрілками з "вентилями", які регулюють ці потоки. Хмари позначають необмежені потоки. Такі ідеограми широко використовуються під час побудови моделей неперервних процесів.

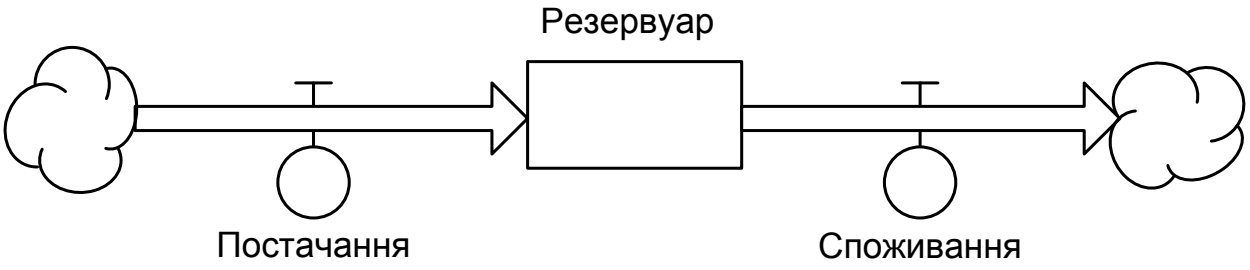


Рис. 3.1. **Схема системи водопостачання**

Знайдемо час заповнення резервуара:

$$T = \frac{W}{V_3 - V_C}. \qquad (3.1)$$

Ця математична модель процесу наповнення резервуару вельми ідеалізується, оскільки всі її параметри вважаються незмінними в часі і зовнішні впливи на систему не враховуються. Завдяки такій ідеалізації маємо дуже просту модель, яка дає можливість розв’язати задачу аналітично. Проте за допомогою такої моделі можна отримати відповідь лише на одне конкретне питання – за який час буде заповнений резервуар.

Якщо задачу про водопостачання наблизити до реальності, то при побудові моделі необхідно враховувати, що потреби підприємства у водопостачанні постійно змінюються, більш того, можливі перебої в роботі насосів під час подачі води.

Розв'язок (3.1) задачі можна записати як

$$W = (V_З - V_C)T .$$

Позначило через $W(t)$ об'єм води в резервуарі в деякий момент часу t, тоді

$$W(t) = (V_З - V_C)t , \qquad (3.2)$$

тобто пошук T зводиться до розв'язання рівняння $W(t) = W$ або $(V_З - V_C)t = W$.

За рахунок неявного запису отримана придатна для дослідження й аналізу реальних процесів математична модель (3.2). Час заповнення резервуара об'ємом W залежить від параметрів моделі $V_З$, V_C. Використання цієї моделі дає можливість вивчити співвідношення між величинами V_C і $V_З$, задаючи різні початкові значення для них, і будуючи графік наповнення резервуара (рис. 3.2).

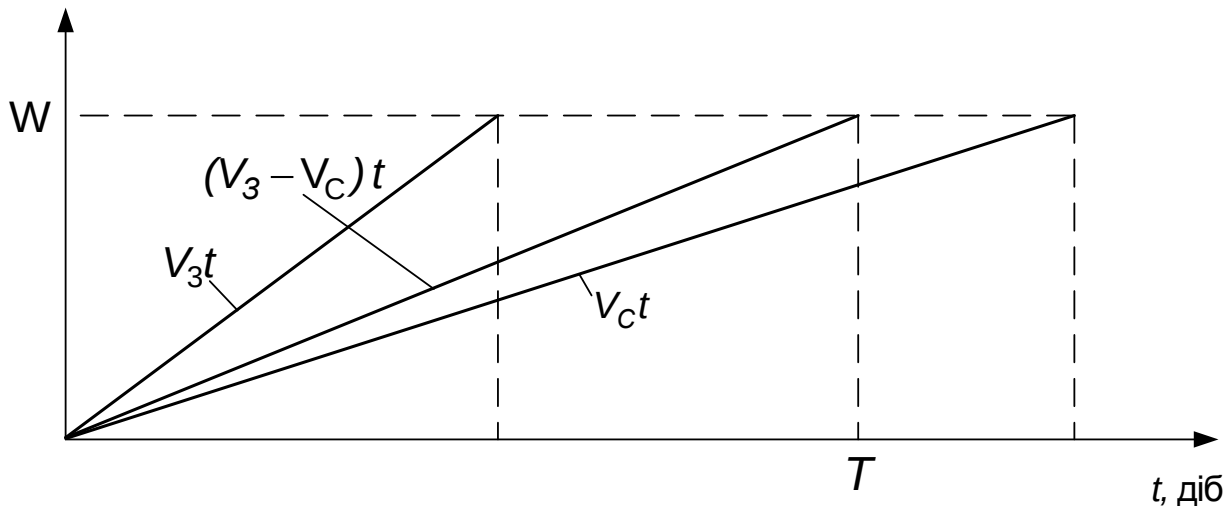


Рис. 3.2. **Графік заповнення резервуара**

Реалізувати цю модель можна і за допомогою чисельних методів. Змінюючи у формулі (3.2) значення t від 0 з деяким кроком Δt до

такого, при якому виконуватиметься рівність $W(t) = W$, отримаємо динамічну характеристику заповнення резервуару. Чим менше крок Δt, тим точніше отримаємо результат, але тим довше розв'язуватиметься задача моделювання.

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

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

Перш за все, слід ввести в модель **структуру** системи, тобто загальний опис елементів і зв'язків між ними, потім визначити засоби відтворення в моделі поведінки системи. Переважно поведінку системи описують за допомогою її станів і моментів переходів між ними. Стан системи у момент часу t визначають як множину значень певних параметрів (змінних) системи в один і той же момент часу t. Будь-яку зміну цих значень можна розглядати як перехід до іншого стану. І, нарешті, імітаційна модель повинна відображати властивості середовища, в якому функціонує досліджувана система. Зовнішнє середовище задають вхідними впливами на модель.

Вся інформація про імітаційну модель взагалі має логіко-математичний характер і подається у вигляді сукупності алгоритмів, які описують процес функціонування системи. Отже, більшою мірою 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 можуть бути задіяні не лише програмні засоби, але і 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. Таким чином, імітаційна модель функціонує так само, як система.

Якщо повернутися до процесу наповнення резервуара (рис. 3.1), то за допомогою імітаційної моделі весь процес можна відтворити з використанням рівняння (3.2). Позначимо через W_i поточний стан резервуара, що відтворюється в певні моменти t_i модельного часу, який змінюється з постійним кроком Δt:

$$W_i = (V_З - V_C)\, t_i, \qquad (3.3)$$

де $t_i = t_{i-1} + \Delta t \ (i = 1, 2, \ldots)$, $t_0 = 0$.

Модель (3.3) є детермінованою. Процес моделювання закінчується, якщо на деякому кроці виконується умова $W_i \geq W$, тобто розв'язок отримуємо за один прогін імітаційної моделі (3.3). Точність результату при цьому залежатиме від значення Δt (рис. 3.3).

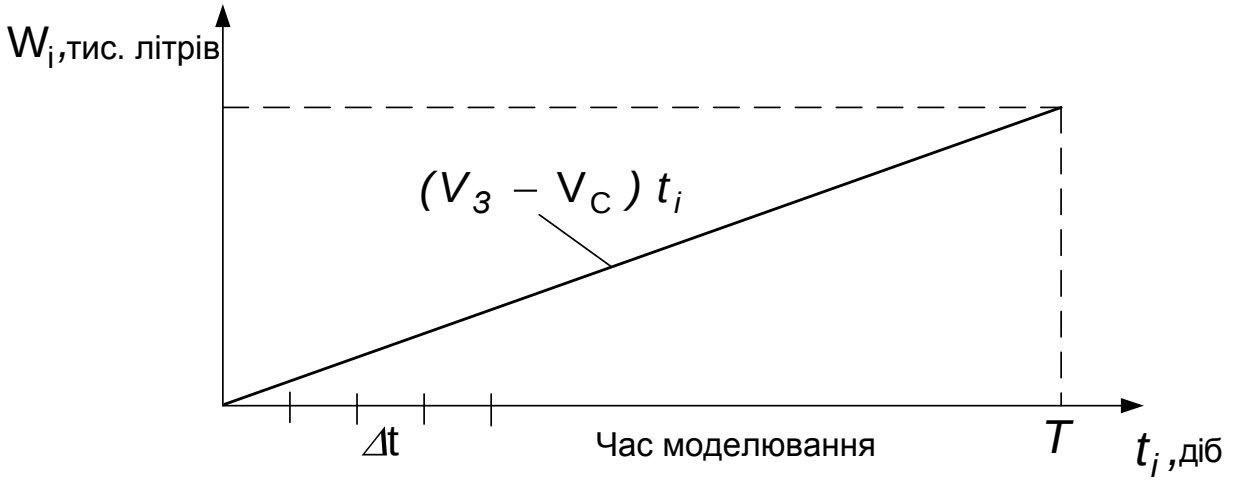


Рис. 3.3. **Динамічна характеристика наповнення резервуара**

За наявності в моделі випадкових факторів виникає необхідність статистичного оцінювання результатів моделювання, яке виконується за допомогою метода ***статистичного моделювання*** (методу Монте-

Карло). Статистичне моделювання є самостійним видом моделювання, яке включається в імітаційне моделювання лише за необхідності моделювання ймовірнісних систем і процесів.

Побудуємо реальнішу модель системи, яка розглядалася вище. Припустимо, що рівень споживання води на підприємстві має імовірнісний характер і змінюється згідно з рівномірним розподілом ймовірності в межах $V_C \pm \Delta V_C$. Модель (3.3) тоді перепишемо у вигляді

$$W_i = (V_3 - V_{Ci})\, t_i, \tag{3.4}$$

де V_{Ci} – рівень споживання води на підприємстві в деякий момент часу t_i. При цьому V_{Ci} буде визначатися як

$$V_{Ci} = V_C - \Delta V_C + 2\Delta V_C r_i, \tag{3.5}$$

де r_i – випадкове число, рівномірно розподілене в інтервалі [0; 1].

Результати роботи імітаційної стохастичної моделі (3.4) – (3.5) наведені на рис. 3.4. У цьому випадку після кожного прогону моделі отримаємо випадкові значення T_j, де j – номер прогону, $j = 1, 2, 3, \ldots$. Відзначимо, що для кожного прогону потрібно генерувати свою послідовність випадкових чисел r_i. Як видно на рис. 3.4, отримані значення T_j будуть відрізнятися від середнього значення T, знайденого за допомого детермінованої моделі (3.1). Таким чином, щоб оцінити час T наповнення резервуара, треба задати точність оцінювання $\varepsilon = \Delta T$ і довірчу ймовірність α. Зазвичай $\alpha = 0.95$, тобто гарантується, що в 95 випадках із 100 середнє значення часу T буде знаходитися в межах $T \pm \Delta T$.

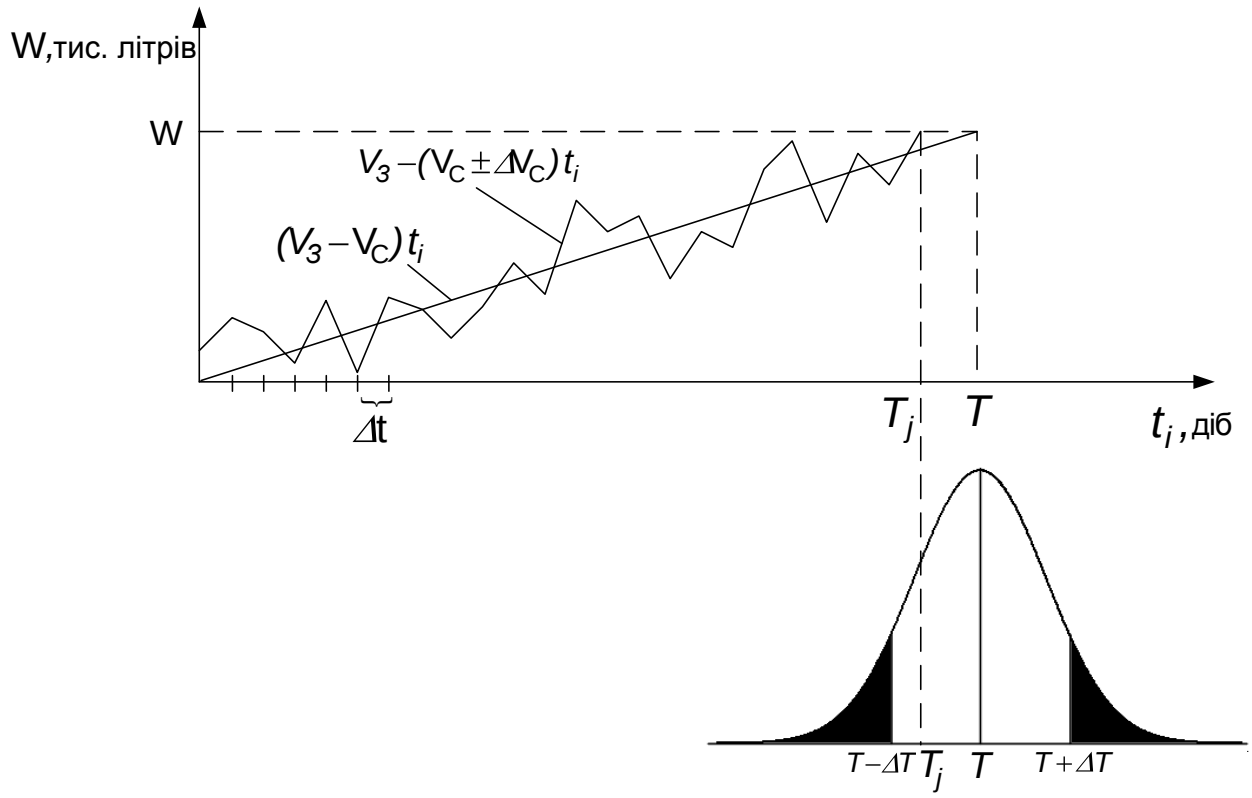


Рис. 3.4. **Графік реалізації стохастичної моделі**

З вищенаведеного прикладу видно, що статистичне моделювання використовується при імітаційному моделюванні лише за необхідності врахування випадкових факторів.

3.2. Декомпозиція систем та простір станів

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

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

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

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

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

де $x_j \in X$, $j = 1, 2, \ldots, n$.

У літературі [12] вказано, що відношення n-го порядку можна розкласти на $n-2$ тривимірних відношень. З погляду дослідження систем найважливішим є наслідок цієї теореми, пов'язаний з введенням поняття стану системи. Розглянемо систему, яка задається відношенням

$$Y \cdot R \cdot X(t). \tag{3.6}$$

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 (3.6) має порядок $n+1$ і не може бути розкладене на відношення нижче третього порядку. Нехай елементи $X(t)$ впорядковані в часі:

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

Тоді відношення (3.6) має вигляд

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

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

$$X^j(t) = [x(t_{j+1}), x(t_{j+2}), \ldots, x(t_n)].$$

Відношення (3.6) буде еквівалентним відношенню

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

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

$$X^{jr}(t) = [x(t_1), x(t_2), \ldots, x(t_j)].$$

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

$$R_1[X^j(t), Z^j] \text{ і } R_2[Z^j, X^{jr}(t)]. \qquad (3.7)$$

Терм Y залежить тільки від проміжного терма Z^j; і не залежить від елементів $x(t_j)$, в яких індекс менше ніж j. Можна стверджувати, що елемент Z^j описує **стан системи**. Якщо система розділена на дві підсистеми відповідно до виразу (3.7), то терм Y залежить тільки від стану системи в момент $\tau = t_j$ і всіх майбутніх елементів X і не залежить від всіх попередніх елементів. Стан системи у момент часу τ називається початковим станом і позначається через $z_0(\tau)$. Наведені міркування справедливі і 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 відтворення в пам'яті. Винятком є необхідність анімаційного або графічного відтворення станів системи в часі і можливість її "програвання" у прямому і зворотному напрямах.

Якщо потрібно зменшити порядок відношення системи шляхом усунення залежності від будь-яких елементів певної підмножини X^r, то нове відношення повинне бути хоча б тривимірним, трьома термами його є входи X, виходи Y і стани Z. Рівняння

$$z(t > \tau) = z(z(\tau); x(\tau, t)) \qquad (3.8)$$

називатимемо рівнянням станів системи, а функцію z – перехідною функцією станів системи. Таким чином, вхідні впливи X перетворюються на виходи системи Y за допомогою рівняння станів (3.6), і саму систему S можна представити у вигляді "чорного ящика", зображеного на рис. 3.5, де зовнішні відношення зв'язують елементи системи із зовнішнім середовищем за допомогою входів системи. При проведенні

досліджень над системою можна впливати на її входи і спостерігати за її виходами. Вхідні змінні, які дослідник може змінювати, проводячи експеримент, називаються управляючими змінними, а ті, які неможливо змінювати, – спостережуваними змінними. Під час моделювання звісно можна змінювати всі вхідні змінні.

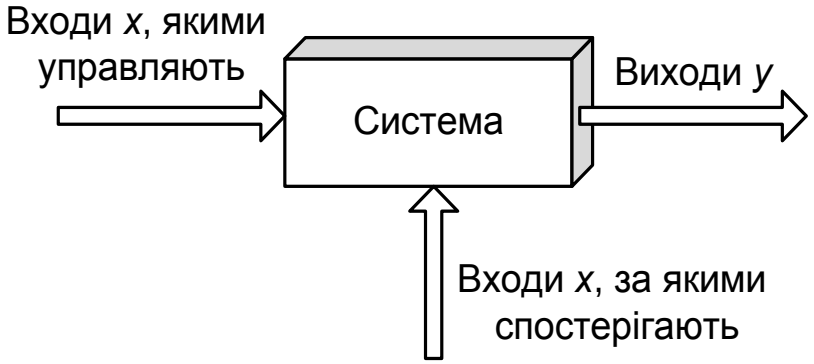


Рис. 3.5. **Кібернетична модель системи**

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 в 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 цілком впорядкованої множини додатних дійсних чисел t, які визначають перебіг часу, пара елементів (t,z), де $t \in T$, $z \in Z$, називають ***станом*** або ***фазою*** системи S, а множина $T \times Z$ – просторо-станом 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, який характеризує об'єкт у поточний або довільний момент часу подібно тому, як координати звичайного простору характе-

ризують просторове розташування. Під фазовим простором розуміється простір, в якому визначені не лише статичні координати точки, координати її положення, але і міститься вся інформація, потрібна для визначення її поведінки в майбутньому.

Важливість поняття стану полягає в можливості, використовуючи його як деякий параметр, пов'язати з кожним значенням вхідних змінних єдине значення вихідних змінних. Якщо зміна станів системи відбувається неперервно в часі, то динамічна система належить до класу неперервних систем. Якщо ж функція (3.8) визначена на дискретній множині моментів часу t, то розглядають клас дискретних динамічних систем. В окремому випадку дискретні моменти часу можуть задаватися 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 (3.8). 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 кожен момент часу t, або шляхом задавання таблиці станів, як це робиться, 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 і початковим станом системи. Тобто, послідовність розташованих у порядку збільшення часу пар $(z(t), x[\tau, t))$ визначає процес і описує поведінку системи.

У разі побудови моделей динамічних систем ці системи описуються у вигляді множини деяких ***реалій*** (рис. 3.6), які можна описувати і моделювати за допомогою властивостей, що змінюють стани системи. Зміна станів системи спричиняє ***події***, яким відповідають певні **умови**. Виникнення певних умов призводить до ***дій***, які утворюють конкретні ***процеси***.

Рис. 3.6. **Схема опису динамічних систем**

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

3.3. Формальні методи побудови моделей

Розглядаючи сфери застосування моделей, можна констатувати, що за допомогою моделі можна досягти ***двох основних цілей: описової***, якщо модель призначена для пояснення і кращого розуміння об'єкта, або ***приписуючої***, коли модель дає можливість передбачити або відтворити характеристики об'єкта чи визначити його поведінку. Таким чином, ***модель є описовою***, якщо вона призначена зображати поведінку (функціонування) або властивості існуючої чи типової системи (наприклад, масштабна модель або письмовий опис, який дає можливість знайомити потенційних покупців з фізичними і робочими характеристиками комп'ютера). Протилежність – ***приписуюча модель***, яка відображає необхідну поведінку або властивості запропонованої системи (наприклад, масштабна модель або письмовий опис, представлений постачальникові комп'ютерів, з фізичними і робочими характеристиками потрібного замовникові комп'ютера).

Приписуюча модель може бути описовою, але не навпаки. Тому існує різний ступінь корисності моделей, які використовуються в технікних і соціальних науках. Це значною мірою залежить від методів і засо-

бів, застосовуваних під час побудови моделей, а також від кінцевої мети. У соціальних науках моделі призначені для пояснення існуючих систем, а в техніці вони є допоміжними засобами для створення нових або досконаліших моделей. Модель, яка придатна для досягнення цілей розробки системи, повинна також пояснювати (тлумачити) її.

При побудові 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 меж відповідних якостей; з цією метою ведеться 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 і т. п.) і його запису у формалізованій формі; це рівень структурних теоретичних схем, таких, як системи масового обслуговування, мережі Петрі, скінченні або імовірнісні автомати, діаграми фонд-потік тощо.

4. Завершення формулювання моделі її "оснащенням" – задавання початкового стану і параметрів об'єкта.

5. Вивчення моделі за допомогою доступних методів (зокрема із застосуванням різних підходів і обчислювальних методів).

У результаті дослідження моделі досягається поставлена мета. У цьому випадку повинна бути встановлена всіма можливими способами (шляхом порівняння з практикою, порівнянням з іншими підходами) її адекватність, тобто відповідність об'єкта сформульованим умовам.

При побудові моделей зазвичай використовують такі формальні підходи: кібернетичний, системна динаміка, теоретико-множинний.

3.3.1. Кібернетичний підхід

Систему можна вивчати й 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 лише за його входами і виходами, оскільки поведінка модельованої системи нічим не відрізняється від поведінки ізоморфних їй систем.

Для побудови моделі використовуються методи **теорії ідентифікації** (див. підрозділ 4.1).

3.3.2. Системна динаміка

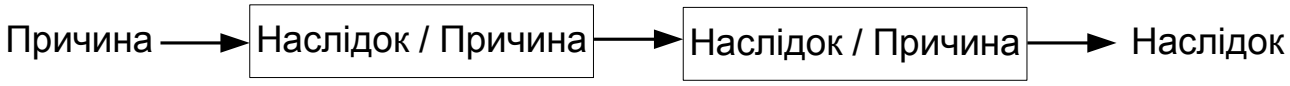
Для формального представлення моделей неперервних систем Дж. Форрестер у 1960 році запропонував підхід, названий **системною динамікою**, який дає можливість будувати моделі динамічних взаємозв'язаних систем за допомогою причинних діаграм циклів і схем виду "фонд-потік". Він же запропонував для чисельного 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 і неперервно-дискретного моделювання базуються на цій мові для побудови моделей. На відміну від математичного розв'язання системи таких рівнянь у замкнутому вигляді використовується чисельне розв'язання з дискретним кроком часу, що дає можливість моделювати на деякому проміжку часу динамічні зміни фондів, пов'язаних з точкою часу, і потоків. Фонди і потоки пов'язані між собою через змінні [12].

Фонд можна трактувати як деяку кількість чого-небудь, що вимірюється в певних одиницях (наприклад, фізичних, грошових та ін.). Фонди можуть акумулювати одиниці фонду. Краще всього їх представляти як резервуари, ресурси або буфера. Фонди поповнюються через вхідні потоки і спорожняються через вихідні. Як буфер фонд може використовуватися для забезпечення балансування швидкості накопичення і витрачання (наприклад, в задачі про водопостачання, яке розглядалася в підрозділі 3.1).

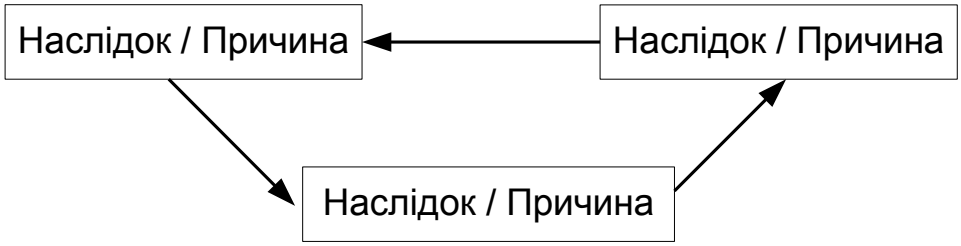
Потік – це процес, що протікає неперервно в часі, оцінити який можна в деяких кількісних одиницях за певний проміжок часу. Залежно від характеристики використання потоки діляться на: обмежені і необмежені, одно- і двонаправлені, конвертовані і неконвертовані. Потік, як правило, обмежується фондом. Потоком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

Існує багато різних способів пов'язувати в динамічних моделях причини і наслідки, не розглядаючи конкретні методи. В їх основі лежить декілька підходів. Розглянемо три з них, наведених на рис. 3.7.

а) ізольоване уявлення



б) лінійний зв'язок



в) причинний цикл, нелінійний зворотний зв'язок

Рис. 3.7. **Три підходи до пов'язування причин і наслідків для побудови моделі**

Перший підхід (**ізольоване уявлення**) полягає в тому, що наслідок виникає з деякої причини і взаємозв'язок між різними причинами відсутній. Такий підхід, наприклад, використовують економісти під час розрахунків. Як правило, для цього застосовують статичні і статистичні моделі.

Другий підхід (**лінійний зв'язок**) передбачає, що між причинами і наслідками існує лінійний зв'язок у вигляді ланцюжка. Такий підхід підтримують інженери і 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 і моделями лінійних рівнянь та вважають, що завжди можна логічно прослідкувати, "що є на вході і що буде на виході".

Згідно з третім підходом (**причинний цикл**) всесвіт розглядається як система з зворотними зв'язками, тобто ланцюжки причин і наслідків циклічно пов'язані між собою. Таке уявлення підтримують кібернетики, прибічники нелінійної динаміки і хаосу. Вони вважають, що всесвіт значною мірою хаотичний, і передбачити майбутнє, враховуючи його минуле, неможливо. Ці системні мислителі використовують циклічні причинні моделі, нелінійні рівняння в кінцевих різницях. Часто поведінка таких моделей далека від реальності й інтуїтивного уявлення і може бути де в чому неочікуваною для дослідника.

На рис. 3.8 зображена проста причинна циклічна модель для деякої популяції, яка має два цикли. Лівий цикл, додатній, свідчить про приріст популяції в разі збільшення народжуваності, яка у свою чергу збільшує народжуваність. Правий цикл, від'ємний, свідчить про зменшення популяції в разі збільшення смертності, яка у свою чергу зменшує смертність. Такі пари причинних циклів можуть використовуватися під час побудови складніших динамічних моделей.

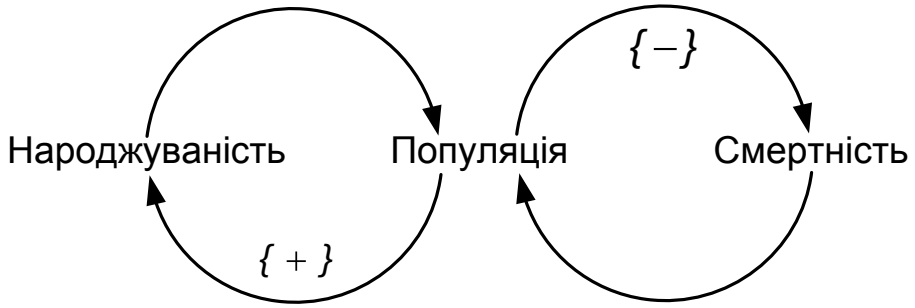


Рис. 3.8. **Найпростіша причинна модель циклу популяції**

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

1. Абстрагування від фізичної структури системи.

2. Концентрація на процесах для визначення траєкторій, за якими система починає і закінчує працювати.

3. Використання простих диференциіально-різницевих рівнянь для опису процесів у системі:

– $\frac{dx}{dt} = kx$ – показникова функція, яка визначає швидкість зміни фонда в часі, де x – фонд (для прикладу з водопостачанням – це швидкість наповнення резервуара);

– $\frac{dx}{dt} = ax - bx^2$ – сигмаїдальна, або логістична, крива, або *S-крива*;

або системи рівнянь:

– $\frac{dx}{dt} = k_1 x - k_2 x^2 - k_3 y$;

– $\frac{dy}{dt} = k_4 y - k_5 y^2 - k_6 x$

(наприклад, x – кількість травоїдних тварин, 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 представити в чисельному вигляді. Для цього використовують прості рівняння рекурсії:

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

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

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

Якщо описати дані рівняння словами, то наступний рівень дорівнюватиме попередньому плюс невелика зміна впродовж короткого проміжку часу. У такий спосіб можна будувати складні динамічні моделі за допомогою створення простих блоків у вигляді відношень і рівнів. У сучасних пакетах моделювання цей процес запису рівнянь автоматизований із застосуванням ідеографічних схем (див. рис. 3.7).Причинні діаграми циклів дають можливість вести ***якісне*** моделювання, а діаграми "фонд-потік" – ***кількісне***. Щоб пояснити явище, необхідно знайти "причини" його виникнення. Припустимо, що така причина визначена і наслідок може спостерігатися кожного разу, коли ця причина присутня. Якщо описують ці концепції системного мислення звичайними словами, то використовують слова або фрази "оскільки", "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 і їх кількістю. Наприклад, вовк може знищити зайців, але кількість вовків не може знищити щось, а може тільки вплинути на кількість зайців.

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

3.3.3. Теоретико-множинний підхід

Згідно з теоретико-множинним підходом формальна модель динамічної системи має такий вигляд [12]:

$$M = \langle T, X, Y, Z, z(t), P \rangle, \tag{3.9}$$

де T – модельний час; X, Y – множина значень 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 $p = \langle x, z, t \rangle$, де $t \in T$, а τ – початковий момент модельного часу для процесу $p \in P$. Таке визначення задає модель системи у вигляді схеми процесів, в якій множина процесів може 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 ***подієву*** схему моделі:

$$M = \langle T, X, Y, Z, z(t), C \rangle, \tag{3.10}$$

де C – множина подій, яка визначається як множина впорядкованих у часі пар елементів $c = \langle t, d[\tau, t_j) \rangle$, 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

ції схемі процес моделювання описується як послідовність подій, які відбуваються в моделі.

Припустимо, що завдяки виконанню деякої умови u з множини U почне виконуватися 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***:

$$M = (T, X, Y, Z, z(t), D). \tag{3.11}$$

У цій схемі процес моделювання описується як перевірка всіх умов у разі кожної зміни модельного часу, щоб знайти умову, яка розпочне певну дію з множини D. Зміна часу t може відбуватися з постійним або змінюваним від події до події кроком.

Схеми моделей (3.9) – (3.11) широко застосовуються під час побудови алгоритмів моделювання і мов дискретного імітаційного моделювання.

Якщо припустити, що виконання деякої множини процесів P може привести до зміни станів $z \in Z$ і виникнення нових процесів, що послужить причиною появи деякої множини ситуацій L, тобто $z(t) : P_z \to L$, то отримаємо **ситуаційну** або **причинно-наслідкову** схему:

$$M = (T, X, Y, Z, z(t), L). \tag{3.12}$$

У ній потрібно описати множину ситуацій і множину правил (алгоритмів), за якими визначають виконуваний процес. 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 тих, що існують, то вона перетворюється на модель зі штучним інтелектом.

Під час ситуаційного моделювання, як правило, повний опис всіх можливих ситуацій замінюється деякою множиною узагальнених ситуацій, кожна з яких з певною мірою ймовірності відтворює один з можливих станів системи. Для кожної ситуації існує набір правил дії. Вибір того або іншого правила може здійснюватися за деяким критерієм або за допо-

могою таблиць прийняття рішень, а в простіших випадках – згідно з заданою ймовірністю. Моделювання виконується шляхом програвання різних ситуацій за певним сценарієм, яким в окремому випадку може бути алгоритм моделювання. Таким чином, створюють різні ігри, наприклад ділові, військові, економічні, розважальні. ***Гра*** – це спрощене відтворення реального процесу, яке переважно 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 і соціальні системи. Під час вивчення таких систем дослідник, базуючись на системному аналізі, вивчає й описує впливи однієї системи на іншу.

Висновки

1. Імітаційне моделювання – це метод конструювання моделі системи та проведення експериментів над моделлю.

2. Статистичне моделювання використовується при імітаційному моделюванні якщо є потреба врахування випадкових факторів.

Контрольні запитання та завдання

1. Порівняйте числовий метод розв'язання задачі про водопостачання з методом імітаційного моделювання. Що є між ними спільного?

2. Покажіть, яким чином можна провести декомпозицію для не скінченних множин (див. підрозділ 3.2).

3. 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.

4. Дайте ситуаційний опис переходу пішоходом дороги. Розгляньте всі можливі ситуації.

Тема 4. Ідентифікація параметрів математичної моделі. Адекватність, чутливість, несуперечливість моделі

4.1. Постановка задачі ідентифікації моделей

У загальному випадку ***задача ідентифікації*** формулюється так: на основі результатів спостереження за вхідними і вихідними змінними системи потрібно побудувати оптимальну в деякому розумінні математичну модель.

Основними етапами ідентифікації є такі [12]:

1. Вибір класу і структури моделі і мови її опису.
2. Вибір класу і типів вхідних впливів X.
3. Обґрунтування критеріїв схожості системи і моделі.
4. Вибір методу ідентифікації і розробка відповідних алгоритмів оцінювання параметрів моделі.
5. Перевірка адекватності отриманої в результаті ідентифікації моделі.

Залежно від обсягу апріорної інформації про клас і структуру системи відрізняють задачі ідентифікації в широкому і вузькому розумінні.

Задача ідентифікації в 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. Вони розроблені для тих ситуацій, що досить часто виникають на практиці, коли дослідник нічого не знає про параметри досліджуваної системи (звідси і назва методів – ***непараметричні***).

Задача ідентифікації у вузькому розумінні полягає в оцінюванні параметрів і станів системи, якщо відома структура моделі ("сірий ящик"). Задачею ідентифікації є кількісне оцінювання певних параметрів моделі. Для цього використовується параметрична ідентифікація мате-

матичної моделі. Прикладами таких моделей можуть бути диференціальні і різницеві рівняння, моделі типу "вхід – стан – вихід".

На рис. 4.1 зображена загальна схема ідентифікації моделі (оцінювання параметрів моделі). Вхідні впливи X на систему і модель однакові, виходи системи Y_S і моделі Y_M в загальному випадку відрізняються. Для їх порівняння потрібно сформувати критерій схожості і мінімізувати його, тобто настроїти модель.

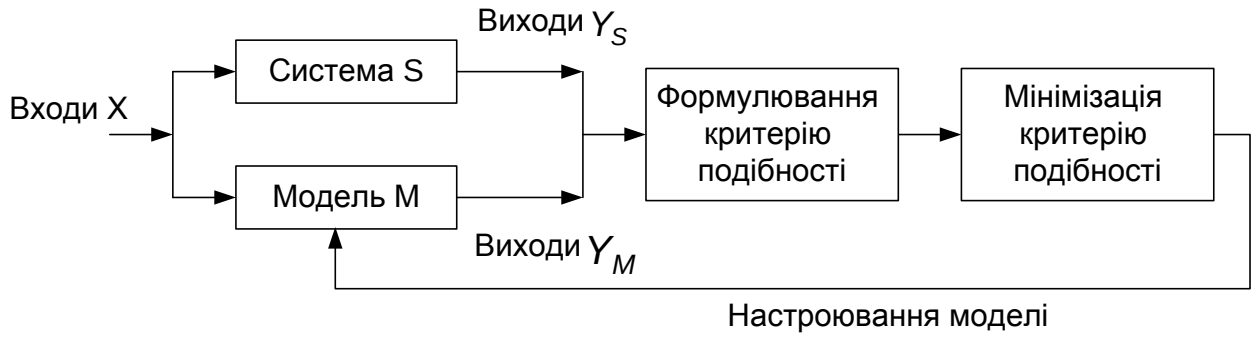


Рис. 4.1. **Загальна схема ідентифікації моделі**

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

4.2. Основні етапи розв'язання задачі ідентифікації та їх взаємозв’язок

Взаємозв’язок основних етапів розв’язання задачі ідентифікації можна проілюструвати такою схемою (рис. 4.2).

Метод найменших квадратів для ідентифікації параметрів моделі

Найбільш відомим та досить ефективним методом розв'язання задачі ідентифікації параметрів моделі є ***метод найменших квадратів***.

Задача ідентифікації параметрів моделі типу "вхід – вихід" в загальному вигляді формулюється таким чином.

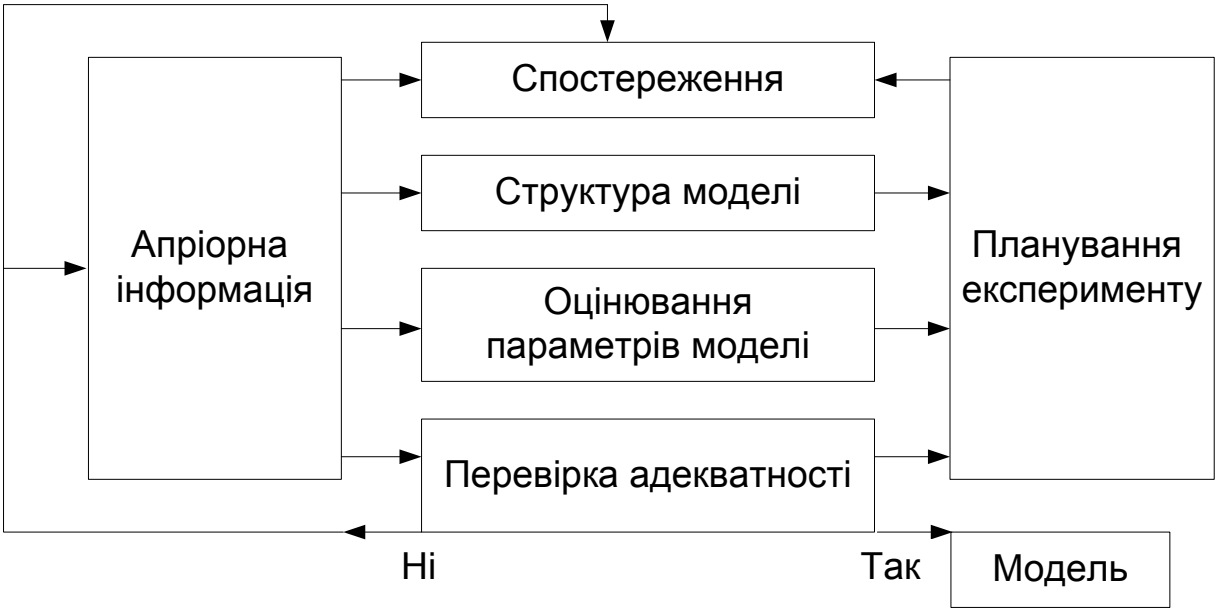


Рис. 4.2. **Схема взаємозв'язку основних етапів розв'язання задачі ідентифікації**

Нехай деяка система описується вхідними x і вихідними y змінними (тобто відповідає схемі, зображеній на рис. 3.5) і яким-небудь чином обрана структура моделі (тобто вид залежності y від x):

$$y = G(x; a) + e, \tag{4.1}$$

де $a \in R^k$, $a = (a_1, a_2, \ldots a_k)^T$ – деякі параметри моделі; e – помилка моделі (враховуючи й випадкові помилки в даних експерименту).

Необхідно на основі результатів спостереження за вхідними й вихідними змінними системи (даних експерименту) знайти оцінку параметрів моделі, тобто побудувати оптимальну в деякому розумінні математичну модель.

Метод найменших квадратів для розв'язання цієї задачі полягає в наступному. Розглянемо випадок, коли вхідних змінних x може бути декілька ($x \in R^m$), а вихідна змінна y одна ($y \in R^1$). Нехай є дані n експериментів $(x^i, y_i), i = \overline{1,n}$, причому значення y_i містять випадкову помилку. Уводиться функція (від параметрів a) виду:

$$\Phi(a) \equiv \sum_{i=1}^{n} \left[y_i - G(x^i; a) \right]^2, \tag{4.2}$$

яку можна розглядати як міру відхилення моделі $G(x;a)$ від даних експерименту $y_1, y_2, \ldots, y_n$. Тоді оцінку параметрів a моделі $G(x;a)$ можна визначити з умови найменшого відхилення $G(x;a)$ від даних експерименту, тобто оцінка параметрів a знаходиться як точка, у якій функція $\Phi(a)$ досягає по $a \in R^k$ мінімального значення (точка мінімуму).

4.3. Поняття адекватності, сталості та чутливості моделі, формальні способи їх перевірки

Оцінка якості моделі є завершальним етапом її розробки й переслідує дві цілі [10]:

1) перевірити відповідність моделі її призначенню (цілям дослідження);

2) оцінити ймовірність і статистичні характеристики результатів, отриманих при проведенні експериментів з моделлю.

При аналітичному моделюванні ймовірність результатів визначається двома основними факторами:

1) конкретним вибором математичного апарату, використовуваного для опису досліджуваної системи;

2) методичною помилкою, властивою даному математичному методу.

При імітаційному моделюванні на ймовірність результатів впливає цілий ряд додаткових факторів, основними з яких є:

моделювання випадкових факторів, засноване на використанні датчиків випадкових чисел, які можуть вносити "перекручування" у поводження моделі;

наявність нестаціонарного режиму роботи моделі;

використання декількох різнотипних математичних методів у рамках однієї моделі;

залежність результатів моделювання від плану експериментів;

необхідність синхронізації роботи окремих компонентів моделі.

Придатність імітаційної моделі для рішення завдань дослідження характеризується тим, у якій мірі вона має так звані ***цільові властивості***. Основними з них є:

адекватність;

сталість;

чутливість.

Нижче розглянуті деякі способи проведення оцінки якості моделі за кожним з них.

Оцінка адекватності моделі. У загальному випадку під адекватністю розуміють ступінь відповідності моделі тому реальному явищу або об'єкту, для опису якого вона будується.

Разом з тим, створювана модель орієнтована, як правило, на дослідження певної підмножини властивостей цього об'єкта. Тому можна вважати, що адекватність моделі визначається ступенем її відповідності не стільки реальному об'єкту, скільки цілям дослідження. Найбільшою мірою це твердження справедливо щодо моделей проектованих систем (тобто в ситуаціях, коли реальна система взагалі не існує).

Проте в багатьох випадках корисно мати формальне підтвердьження (або обґрунтування) адекватності розробленої моделі. Один з найпоширеніших способів такого обґрунтування – використання методів математичної статистики. Суть цих методів полягає в перевірці висунутих гіпотез (у цьому випадку – про адекватність моделі) на основі деяких статистичних критеріїв.

Процедура оцінки адекватності моделі заснована на порівнянні вимірів на реальній системі й результатів експериментів на моделі й може проводитися різними способами. Найпоширеніші з них:

за середніми значеннями відгуків (виходів) моделі й системи;

за дисперсіями відхилень відгуків моделі від середнього значення відгуків системи;

за максимальним значенням відносних відхилень відгуків моделі від відгуків системи.

Названі способи оцінки досить близькі між собою по суті, тому обмежимося розглядом першого з них.

При цьому способі перевіряється гіпотеза про близькість середнього значення спостережуваної змінної моделі y середньому значенню відгуку реальної системи y^*.

У результаті N_0 експериментів на реальній системі одержують множину значень (вибірку) вихідної змінної y^*. Виконавши N_M експе-

риментів на моделі, також одержують множину значень спостережуваної змінної y.

Потім обчислюються оцінки математичного очікування й дисперсії відгуків моделі й системи, після чого висувається гіпотеза про близькість середніх значень y^* й y (у статистичному сенсі). Основою для перевірки гіпотези є t-статистика (розподіл Стьюдента). Її значення, обчислене за результатами випробувань, порівнюється із критичним значенням $t_{кр}$, узятим з довідкової таблиці. Якщо виконується нерівність $t \le t_{кр}$, то гіпотеза приймається.

Оцінка сталості моделі. При оцінці адекватності моделі як існуючої, так і проектованої системи реально може бути використана лише обмежена підмножина всіх можливих значень вхідних параметрів (робочого навантаження й зовнішнього середовища). У зв'язку із цим для обґрунтування вірогідності одержуваних результатів моделювання велике значення має перевірка сталості моделі. У теорії моделювання це поняття трактується в такий спосіб.

Сталість моделі – це її здатність зберігати адекватність при дослідженні ефективності системи на всьому можливому діапазоні робочого навантаження, а також при внесенні змін у конфігурацію системи.

Яким чином може бути оцінена сталість моделі? Універсальної процедури перевірки сталості моделі не існує. Розроблювач змушений вдаватися до методів "для даного випадку", частковим тестам і здоровому глузду. Часто буває корисна апостеріорна перевірка. Вона полягає в порівнянні результатів моделювання й результатів вимірів на системі після внесення в неї змін. Якщо результати моделювання прийнятні, упевненість у стійкості моделі зростає.

У загальному випадку можна стверджувати, що чим ближче структура моделі структурі системи й чим вище ступінь деталізації, тим більша сталість моделі.

Сталість результатів моделювання може бути також оцінена методами математичної статистики. Для перевірки гіпотези про сталість результатів може бути використаний критерій Уїлкоксона.

Критерій Уїлкоксона служить для перевірки того, чи відносяться дві вибірки до однієї й тієї ж генеральної сукупності (тобто чи володіють вони тією самою статистичною ознакою).

При статистичній оцінці стійкості моделі відповідна гіпотеза може бути сформульована в такий спосіб: при зміні вхідного (робочого) навантаження або структури імітаційної моделі закон розподілу результатів моделювання залишається незмінним.

Перевірку зазначеної гіпотези H проводять при таких вихідних даних: є дві вибірки $X=(x_1,\ldots,x_n)$ і $Y=(y_1,\ldots,y_m)$, отримані для різних значень робочого навантаження; щодо законів розподілу X і Y ніяких припущень не робиться.

Значення обох вибірок упорядковуються разом за зростанням. Потім аналізується взаємне розташування x_i й y_j. У випадку $y_j < x_i$ говорять, що пари значень (x_i, y_j) утворять інверсію.

Наприклад, нехай для $n=m=3$ після упорядкування вийшла така послідовність значень: $y_1, x_2, y_3, x_1, y_2, x_3$; тоді маємо інверсії: (x_1, y_1), (x_2, y_1), (x_3, y_1), (x_1, y_3), (x_3, y_3), (x_3, y_2).

Підраховують повне число інверсій U. Якщо гіпотеза вірна, то U не повинне сильно відхилятися від свого математичного очікування M:

$$M=\frac{n\cdot m}{2}.$$

Від гіпотези відмовляються, якщо $|U-M|>\Delta U_{кр}$, де $\Delta U_{кр}$ визначають за таблицею для заданого рівня значущості.

Оцінка чутливості імітаційної моделі. Очевидно, що сталість є позитивною властивістю моделі. Однак якщо зміна вхідних впливів або параметрів моделі (у деякому заданому діапазоні) не відбивається на значеннях вихідних параметрів, то користь від такої моделі невелика. У зв'язку з цим виникає завдання оцінювання чутливості моделі до зміни параметрів робочого навантаження й внутрішніх параметрів самої системи.

Таку оцінку проводять за кожним параметром (змінною) x_k окремо. Заснована вона на тому, що зазвичай діапазон можливих змін пара-

метра відомий. Одна з найпростіших і розповсюджених процедур оцінювання полягає в такому:

1. Обчислюється величина відносного середнього збільшення параметра x_k:

$$\Delta x_k = \frac{2(x_{k\,max} - x_{k\,min})}{x_{k\,max} + x_{k\,min}} \cdot 100\%.$$

2. Проводиться пара модельних експериментів при значеннях $x_k = x_{k\,max}$ й $x_k = x_{k\,min}$ і середніх фіксованих значеннях інших параметрів. Визначаються значення відгуку моделі $y_1 = f(x_{k\,max})$ і $y_2 = f(x_{k\,min})$.

3. Обчислюється відносне збільшення спостережуваної змінної y:

$$\Delta y_k = \frac{2(y_1 - y_2)}{y_1 + y_2} \cdot 100\%.$$

У результаті для k-го параметра моделі мають пари значень (Δx_k, Δy_k), що характеризують чутливість моделі за цим параметром.

Аналогічно формуються пари для інших параметрів моделі, які утворять множину $\{\Delta x_k, \Delta y_k\}$.

Дані, отримані при оцінці чутливості моделі, можуть бути використані, зокрема, при плануванні експериментів. А саме, більша увага має приділятися тим параметрам, за якими модель є більш чутливою.

4.4. Поняття несуперечливості моделі

Несуперечливість – властивість, що полягає в тому, що не кожна формула цієї системи доказова в ній. Формальні системи, що мають цю властивість, називаються несуперечливими, або ***формально несуперечливими***. Інакше формальна система називається ***суперечливою***, або ***несумісною***. Для широкого класу формальних систем, мова яких містить знак заперечення "$\neg$" несуперечність еквівалентна властивості: "не існує такої формули ϕ, що ϕ і $\neg\phi$ обидві доказові". Клас формул даної формальної системи називається несуперечливим, якщо не всяка

формула цієї системи виводиться з даного класу. Формальна система називається ***змістовно несуперечливою***, якщо існує модель, в якій істинні всі теореми цієї системи. Оскільки модель це теж система, то поняття суперечливості і несуперечливості застосовне і до неї.

Суперечливі визначення об'єктів і суперечливі моделі іноді виникають у результаті абсолютизації локальних властивостей реально існуючих об'єктів. Інша можлива причина появи суперечливих моделей – наявність різних неузгоджених джерел інформації, яка служить основою моделювання.

Висновки

1. Задача ідентифікації в широкому розумінні виконується в умовах апріорної невизначеності структури моделі системи ("чорний ящик").
2. Задача ідентифікації у вузькому розумінні полягає в оцінюванні параметрів і станів системи, якщо відома структура моделі ("сірий ящик").
3. Найбільш відомим та досить ефективним методом розв'язання задачі ідентифікації параметрів моделі є метод найменших квадратів.

Контрольні запитання та завдання

1. Сформулюйте завдання ідентифікації в широкому та вузькому розумінні для задачі про водопостачання.
2. У чому полягає метод найменших квадратів для розв'язання задачі ідентифікації параметрів моделі?
3. Назвіть основні цільові властивості імітаційної моделі.

Тема 5. Принципи побудови моделей. Технологія моделювання

5.1. Основні принципи побудови моделей

Розглянемо коротко основні принципи моделювання, які відображають достатньо багатий досвід, накопичений на даний час в сфері розробки і використання моделей [12].

Принцип інформаційної достатності. При повній 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), досягши якого можна побудувати її адекватну модель.

Принцип доцільності. Модель створюється для досягнення деяких цілей, які визначають на первинному етапі формулювання проблеми моделювання.

Так, цілями моделювання можуть бути:

– осмислення дійсності;

– постановка над моделлю експериментів з подальшою інтерпретацією їх результатів стосовно модельованої системи;

– прогнозування майбутньої поведінки системи;

– спілкування з іншими особами, громадськими організаціями, пристроями обробки інформації;

– навчання і тренування фахівців.

Принцип здійсненності. Створювана модель повинна забезпечувати досягнення мети дослідження з урахуванням граничних ресурсів з вірогідністю (ймовірністю), істотно відмінною від нуля, і за скінченний час. Зазвичай задають деяке граничне значення P (ступінь ризику) вірогідності (ймовірності) досягнення мети моделювання $P(t)$, а також сам граничний термін t досягнення мети. Модель вважають здійсненною, якщо $P(t) > P$.

Принцип множинності моделей. Модель, яка створюється, повинна відображати насамперед 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 можливість відобразити певний процес з різних боків і з різним ступенем детальності.

Принцип агрегації. У більшості випадків складну систему можна представити як таку, що складається з агрегатів (підсистем), для адекватного формального опису яких придатними є деякі стандартні математичні схеми. Принцип агрегації дає можливість досить гнучко перебудовувати модель залежно від завдань (задач) дослідження.

Принцип параметризації. У ряді випадків модельована система має у своєму складі деякі відносно ізольовані підсистеми, які характеризуються певними параметрами, зокрема векторними. Такі підсистеми можна замінювати в моделі відповідними числовими величинами, а не описувати процес їх функціонування. У разі потреби залежність значень цих величин від ситуації може задаватися у вигляді таблиць, графіків або аналітичних виразів (формул), наприклад за допомогою регресійного аналізу. Принцип параметризації дає можливість 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 систему. 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 повинне відобразитися в моделі. Таким чином, в моделі можуть одночасно перебувати блоки з різним ступенем деталізації, що моделюють одні і ті ж компоненти проектованої системи. Іншими словами, під час побудови моделі потрібно застосовувати методологію ***ітераційного багаторівневого*** моделювання.

Розробку моделі доцільно починати зі створення простої початкової моделі, яку у процесі уточнення вхідних даних і характеристик системи ускладнюють і корегують, тобто ***адаптують*** до нових умов. Разом з тим модель повинна залишатися досить ***наглядною***, тобто її структура повинна відповідати структурі модельованої системи, а рівень деталізації моделі повинен вибиратися з урахуванням цілей моделювання, ресурсних обмежень (наприклад, час, кваліфіковані людські ресурси і засоби, виділені на проектування) та можливості отримання початкових даних.

Таким чином, модель повинна бути багаторівневою, адаптивною, наглядною, цільовою, розвиватися ітераційним способом, ускладнюватися і корегуватися у процесі створення, що можливо тільки за умови побудови її блочним (модульним) способом. Програмування і відладку моделі доцільно вести поетапно, з послідовним збільшенням програмних модулів.

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

Розглянутий підхід можна реалізувати також у вигляді спеціалізованої мови або пакета моделювання, який дає можливість створювати узагальнені моделі шляхом введення засобів розмноження підмоделей, реорганізації зв'язків між ними і їх параметричної настройки. Цей спосіб орієнтований на фахівців, добре знайомих з мовою моделювання. Інший спосіб реалізації цього підходу полягає в розробці діалогових, інтелектуальних систем моделювання з використанням банку моделей і бази знань, які користувач може настроювати на конкретну реалізацію [12]. У цьому випадку етап програмування можна цілком виключити під час програмної реалізації імітаційної моделі завдяки використанню ефективних методів взаємодії з базами даних і застосуванню засобів генерації моделей.

5.2. Технологія моделювання: основні етапи, їх взаємозв'язок та характеристики

Основою моделювання є методологія системного аналізу. Це дає можливість досліджувати систему, яка проектується або аналізується, за технологією операційного дослідження. Комп'ютерне моделювання (а зазвичай застосовується саме комп'ютерна модель) включає такі взаємопов'язані етапи [12]:

1. Формулювання проблеми і змістовна постановка задачі.
2. Розробка концептуальної моделі.
3. Розробка програмної реалізації моделі, яка включає:

а) вибір засобів програмування, за допомогою яких буде реалізована модель;

б) розробка структурної схеми моделі і складання опису її функціонування;

в) програмна реалізація моделі.

Перевірка адекватності моделі.

4. Організація і планування проведення експериментів, що включає оцінювання точності результатів моделювання.

5. Інтерпретація результатів моделювання і прийняття рішень.

6. Оформлення результатів дослідження.

На першому етапі замовник формулює проблему. Організовуються зустрічі керівника проекту з замовником, аналітиками з моделювання і експертами з проблеми, що вивчається. Визначаються цілі дослідження і спеціальні питання, відповіді на які будуть отримані в результаті 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 моделюваної системи, а також необхідне програмне забезпечення.

На цьому ж етапі ведеться цілеспрямоване дослідження моделюваної системи, притягуються експерти з вирішуваної проблеми, які володіють достовірною інформацією. Збирається інформація про конфігурацію системи і способи її експлуатації для визначення параметрів моделі і вхідних розподілів ймовірностей.

На другому етапі розробляється ***концептуальна модель*** – абстрактна модель, яка дає можливість виявити причинно-наслідкові зв'язки, властиві досліджуваному об'єкту в межах, визначених цілями дослідження. По суті, це формальний опис об'єкта моделювання, яке 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 початкові дані для моделювання.

Рівень деталізації моделі залежить від таких факторів: мети проекту; критеріїв оцінювання показників роботи; доступності даних; достовірності результатів; комп'ютерних можливостей; думки експертів з вирішуваної проблеми; обмежень, пов'язаних з часом і фінансуванням. Ведеться структурний аналіз концептуальної моделі, пропонується опис допущень, які обговорюються з замовником, керівником проекту, аналітиками й експертами з вирішуваної проблеми.

Розробляються моделі початкових даних, ведеться їх статистичний аналіз, за результатами якого визначають розподіли ймовірності, регресійні, кореляційні й інші залежності. На цьому етапі для попереднього аналізу даних широко застосовують різні статистичні пакети (наприклад, STATISTICA).

Для динамічних систем ведеться післяопераційний аналіз функціонування модельованої системи з детальним описом роботи елементів системи. За результатами такого аналізу можна з'ясувати, чи можна вирішити проблему без застосування засобів моделювання. Детально оброблена концептуальна модель дає можливість замовникові з іншого боку поглянути на роботу системи і, наприклад, визначити її вузькі місця, які сприяють зниженню її пропускної спроможності.

Одна зі складних проблем, з якою має справу аналітик моделювання, полягає у визначенні, адекватності моделі системі. Якщо імітаційна модель "адекватна", її можна використовувати для прийняття рішень щодо системи, яку вона представляє, тобто нібито вони приймалися на основі результатів проведення експериментів з реальною системою. Модель складної системи може тільки приблизно відповідати оригіналу, незалежно від того, скільки зусиль було витрачено на її розробку, оскільки абсолютно адекватних моделей не існує.

Оскільки модель завжди повинна розроблятися для певного набору цілей, то модель, яка є адекватною для однієї мети, може не бути такою для дослідження іншої мети. Слід зазначити, що адекватна модель не обов'язково є достовірною, і навпаки. Модель може бути достовірною, але, в цьому випадку, не використовуватися для прийняття рішень. Наприклад, достовірна модель може не бути адекватною з політичних або економічних причин.

При програмній реалізації моделі визначаються засоби програмування, тобто мови програмування або пакети. Наприклад, можуть використовуватися мови програмування загального призначення, такі, як C або Pascal, чи 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 розробку моделі з їх допомогою витрачається менше часу. Разом з тим використання спеціалізованого програмного забезпечення для моделювання призводить до зменшення тривалості процесу програмування і вартості всього проекту.

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

Здійснюється програмування моделі та її відлагоджування, виконуються тестові прогони моделі на основі контрольних даних, ведеться аналіз чутливості, щоб визначити, які фактори в моделі суттєво впливають на робочі характеристики системи і повинні моделюватися точніше.

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***. Під час перевірки достовірності потрібно відповісти на питання про відповідність моделі модельованій системі, тобто визначити, наскільки ізоморфні система і модель. Як правило, в разі моделювання вимога ізоморфізму об'єкта і моделі надмірна, оскільки в цьому випадку складність моделі повинна відповідати складності об'єкта. Тому будують гомоморфні моделі, в яких виконується вимога однозначної відповідності моделі об'єкту.

На етапі ***верифікації*** розглядають, чи правильно реалізована концептуальна модель (модельні припущення) в комп'ютерну програму, тобто виконують налагоджування програми моделювання. Це складне завдання, оскільки може існувати багато логічних шляхів.

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

Валідація – це процес, що дає можливість встановити, чи є модель (а не комп'ютерна програма) достатньо точним відображенням системи ***для конкретних цілей дослідження***.

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 проводять такі експерименти:

– порівнюють середні значення і дисперсії різних альтернатив;

– визначають важливість врахування впливу змінних і обмежень, які накладаються на ці змінні;

– визначають оптимальні значення з деякої множини можливих значень змінних.

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

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

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

Вищенаведені етапи моделювання взаємозв'язані, а сама процедура створення моделі є ітераційною. Це пояснюється тим, що після виконання кожного етапу перевіряється правильність і достовірність моделі і в разі невідповідності моделі об'єкту здійснюється повернення до попередніх етапів з метою корегування і настройки моделі. Залежно від характеру внесених змін повертаються безпосередньо до попереднього етапу або до попередніх етапів. Детальніше технологія моделювання розглядається в наступних розділах.

На останньому етапі моделювання документально оформляють всі результати дослідження і готують програмну документацію для використання їх під час розробки поточних і майбутніх проектів.

Висновки

1. При створенні моделі потрібно: дотримуватися принципів інформаційної достатності, доцільності, здійсненності, множинності моделей, агрегації, параметризації; застосовувати методологію ітераційного багаторівневого моделювання.
2. Комп'ютерне моделювання розбивається на декілька взаємопов'язаних етапів, а сама процедура створення моделі є ітераційною.

Контрольні запитання та завдання

1. Відсортуйте етапи створення комп'ютерної моделі за їх ступенем важливості.
2. Чим відрізняються поняття адекватності і достовірності моделі?

Тема 6. Моделі розрахункових процесів і управління. Динамічні моделі, P-, Q-, F-, A-схеми. Мережні моделі

6.1. Поняття типової математичної схеми моделі

Початковою інформацією при побудові математичних моделей процесів функціонування систем служать дані про призначення і умови роботи досліджуваної (проектованої) системи S. Ця інформація визначає основну мету моделювання системи S і дозволяє сформулювати вимоги до математичної моделі M, що розробляється. Причому рівень абстрагування залежить від кола тих питань, на які дослідник системи хоче отримати відповідь за допомогою моделі, і якоюсь мірою визначає вибір математичної схеми [11, 12].

Введення поняття математична схема дозволяє розглядати математикові не як метод розрахунку, а як метод мислення, як засіб формулювання понять, що є найбільш важливим при переході від словесного опису системи до формального подання процесу її функціонування у вигляді деякої математичної моделі (аналітичної або імітаційної). При

користуванні математичною схемою насамперед дослідника системи S має цікавити питання про адекватність відображення у вигляді конкретних схем реальних процесів у досліджуваній системі, а не можливість отримання відповіді (результату розв'язання) на конкретне питання дослідження. Наприклад, подання процесу функціонування інформаційно-обчислювальної системи колективного користування у вигляді мережі схем масового обслуговування дає можливість добре описати процеси, що відбуваються в системі, але при складних законах вхідних потоків і потоків обслуговування не дає можливості отримання результатів в явному вигляді [11].

Математичну схему можна визначити як ланку (концептуальну модель) при переході від змістовного до формального опису процесу функціонування системи з урахуванням дії зовнішнього середовища, тобто має місце ланцюжок "описова модель – математична схема – математична (аналітична або (і) імітаційна) модель".

6.2. Загальний вид математичної моделі системи

Кожна конкретна система S характеризується набором властивостей, під якими розуміються величини, що відображають поведінку модельованого об'єкта (реальної системи) і враховують умови її функціонування у взаємодії із зовнішнім середовищем (системою) E. При побудові математичної моделі системи необхідно вирішити питання про її повноту. Повнота моделі регулюється, в основному, вибором межі "система S – середовище E". Також має бути вирішене завдання спрощення моделі, яка допомагає виділити основні властивості системи, відкинувши другорядні. Причому віднесення властивостей системи до основних або другорядних суттєво залежить від мети моделювання системи (наприклад, аналіз імовірнісно-часових характеристик процесу функціонування системи, синтез структури системи і т. д.). Модель об'єкта моделювання, тобто системи, можна представити у вигляді множини величин, що описують процес функціонування реальної системи і створюють у загальному випадку такі підмножини [11]:сукупність ***вхідних дій*** на систему (якими, як правило, управляють)

$$x \in X \subset R^{n_X},$$

сукупність ***дій зовнішнього середовища*** (за якими спостерігають)

$$v \in V \subset R^{n_V},$$

сукупність ***внутрішніх (власних) параметрів*** системи

$$h \in H \subset R^{n_H},$$

сукупність ***вихідних характеристик*** системи

$$y \in Y \subset R^{n_Y}.$$

Причому в перелічених підмножинах можна виділити керовані і некеровані змінні. У загальному випадку x, v, h, y є елементами непересічних підмножин і містять як детерміновані, так і стохастичні складові.

При моделюванні системи S вхідні дії, дії зовнішнього середовища E і внутрішні параметри системи є ***незалежними (екзогенними) змінними***, які у векторній формі мають відповідно вигляд: $x(t) = (x_1(t), x_2(t), \ldots, x_{n_X}(t))^T$; $v(t) = (v_1(t), v_2(t), \ldots, v_{n_V}(t))^T$; $h(t) = (h_1(t), h_2(t), \ldots, h_{n_H}(t))^T$, а вихідні характеристики системи є ***залежними (ендогенними) змінними*** і у векторній формі мають вигляд $y(t) = (y_1(t), y_2(t), \ldots, y_{n_Y}(t))^T$.

Процес функціонування системи S описується в часі оператором F_S, який у загальному випадку перетворить незалежні змінні в залежні відповідно до співвідношень виду

$$y(t) = F_S(x, v, h, t). \qquad (6.1)$$

Сукупність залежностей вихідних характеристик системи від часу $y_i(t)$ для всіх $i = \overline{1, n_Y}$ називається ***вихідною траєкторією*** $y(t)$. Залежність (6.1) називається ***законом функціонування системи*** S і позначається F_S. У загальному випадку закон функціонування системи F_S може бути заданий у вигляді: функції, функціонала, логічних умов, в

алгоритмічній і табличній формах або у вигляді словесного правила відповідності.

Дуже важливим для опису і дослідження системи S є поняття ***алгоритму функціонування*** A_S, під яким розуміється метод отримання вихідних характеристик $y(t)$ з урахуванням вхідних дій $x(t)$, дій зовнішнього середовища $v(t)$ і власних параметрів системи $h(t)$. Очевидно, що один і той же закон функціонування F_S системи S може бути реалізований різними способами, тобто за допомогою множини різних алгоритмів функціонування A_S.

Співвідношення (6.1) є математичним описом поведінки об'єкта (системи) моделювання в часі t, тобто відображають його динамічні властивості. Тому математичні моделі такого виду прийнято називати динамічними ***моделями*** (системами) [11].

Для ***статичних моделей*** математична модель (6.1) є від обра-женням між двома підмножинами властивостей модельованого об'єкта Y і X,V,H, що у векторній формі може бути записано як

$$y = f(x,v,h). \qquad (6.2)$$

Співвідношення (6.1) і (6.2) можуть бути задані різними способами: аналітично (за допомогою формул), графічно, таблично і т. д. Такі співвідношення у ряді випадків можуть бути отримані через властивості системи S у конкретні моменти часу, які називаються станами. Стан системи S характеризується векторами

$$z = z(t) = (z_1(t), z_2(t), \ldots, z_{n_z}(t))^T .$$

Якщо розглядати процес функціонування системи S як послідовну зміну станів $z^1(t), z^2(t), \ldots, z^k(t)$ (де $z^j(t) \in R^{n_z}\ \forall j = \overline{1,k}$), то вони можуть бути інтерпретовані як координати точки в n_Z-вимірному фазовому просторі. Причому кожній реалізації процесу відповідатиме деяка фазова траєкторія. Сукупність всіх можливих значень станів $\{z^k\}$ називається ***простором станів*** об'єкта моделювання Z, тобто $z^k \in Z, Z \subset R^{n_z}$ [11, 12].

Стани системи S у момент часу t^*, $t_0 < t^* \le T$ (t_0 – початковий момент часу, T – кінцевий момент часу), повністю визначаються початковими умовами $z^0 = z(t_0) \in Z$, вхідними впливами $x(t)$, внутрішніми параметрами $h(t)$ і впливами зовнішнього середовища $v(t)$, які мали місце за проміжок часу $t^* - t_0$, за допомогою двох векторних рівнянь:

$$z(t) = \Phi(z^0, x, v, h, t), \tag{6.3}$$

$$y(t) = F(z, t). \tag{6.4}$$

Перше рівняння по початковому стану z^0 і незалежним змінним x, ν, h визначає вектор-функцію, а друге по отриманому значенню станів $z(t)$ – залежні змінні на виході системи $y(t)$. Таким чином, ланцюжок рівнянь об'єкта "вхід – стани – вихід" дозволяє визначити характеристики системи

$$y(t) = F[\Phi(z^0, x, v, h, t)]. \tag{6.5}$$

У загальному випадку час у моделі системи S може розглядатися на інтервалі моделювання $(0, T)$ (тобто $t_0 = 0$) як неперервний, так і дискретний, тобто квантований на відрізки довжиною Δt часових одиниць кожен, коли $T = m\Delta t$, де m – кількість інтервалів дискретизації.

Таким чином, під ***математичною моделлю об'єкта*** (системи) розуміють скінченну підмножину змінних $x(t), v(t), h(t)$ разом з математичними зв'язками між ними і характеристиками $y(t)$ [11, 12].

Якщо математичний опис об'єкта моделювання не містить елементів випадковості або вони не враховуються, тобто якщо можна вважати, що в цьому випадку стохастичні дії зовнішнього середовища $v(t)$ і стохастичні внутрішні параметри $h(t)$ відсутні, то модель називається детермінованою в тому сенсі, що характеристики однозначно визначаються детермінованими вхідними впливами:

$$y(t) = f(x, t). \tag{6.6}$$

Очевидно, що детермінована модель є окремим випадком стохастичної моделі.

Наведені математичні співвідношеннями є математичними схемами загального виду і дозволяють описати широкий клас систем. Проте у практиці моделювання об'єктів у галузі системотехніки і системного аналізу на початкових етапах дослідження системи раціональніше використовувати ***типові математичні схеми***: диференціальні рівняння, скінченні й імовірнісні автомати, системи масового обслуговування і т. д. [11, 12].

Не маючи такого ступеня спільності, як розглянуті моделі, типові математичні схеми мають переваги простоти і наглядності, але при суттєвому звуженні можливостей застосування. Як детерміновані моделі, коли при дослідженні випадкові фактори не враховуються, для представлення систем, що функціонують у неперервному часі, використовуються диференціальні, інтегральні, інтегродиференціальні й інші рівняння, а для представлення систем, що функціонують у дискретному часі, – скінченні автомати і кінцево-різницеві схеми. Як стохастичні моделі (при врахуванні випадкових факторів) для представлення систем з дискретним часом використовуються імовірнісні автомати, а для представлення систем з неперервним часом – системи масового обслуговування і т. д.

Перелічені типові математичні схеми, природно, не можуть претендувати на можливість опису на їх базі всіх процесів, що відбуваються у великих інформаційно-управляючих системах, до яких належать АСУ. Для таких систем у ряді випадків перспективнішим є застосування агрегативних моделей [11].

Агрегативні моделі (системи) дозволяють описати широке коло об'єктів дослідження з відображенням системного характеру цих об'єктів. Саме при агрегативном описі складний об'єкт (система) розчленовується на скінченне число частин (підсистем), зберігаючи при цьому зв'язки, що забезпечують взаємодію частин.

Таким чином, при побудові математичних моделей процесів функціонування систем можна виділити такі основні підходи: неперервно-детермінований (наприклад, диференціальні рівняння); дискетно-детермінований (скінченні автомати); дискретно-стохастичний (імовірнісні автомати); неперервно-стохастичний (системи масового обслуговування); узагальнений або універсальний (агрегативні системи).

Математичні схеми, що розглядаються в наступних розділах, повинні допомогти оперувати різними підходами у практичній роботі при моделюванні конкретних систем.

6.3. Неперервно-детерміновані моделі (D-схеми)

Розглянемо особливості неперервно-детермінованого підходу на прикладі використання як математичних моделей диференціальних рівнянь [11]. ***Диференціальними рівняннями*** називаються такі рівняння, в яких невідомими будуть функції однієї або декількох змінних, причому в рівняння входять не тільки самі функції, але і їх похідні різних порядків. Якщо невідомі – функції багатьох змінних, то рівняння називаються ***рівняннями в частинних похідних***, інакше при розгляді функцій тільки однієї незалежної змінної рівняння називаються ***звичайними диференціальними рівняннями*** [1].

Зазвичай у таких математичних моделях незалежною змінною, від якої залежать невідомі шукані функції, служить час t. Тоді математичне співвідношення для детермінованих систем (6.6) в загальному вигляді буде

$$y' = f(y,t),\ y(t_0) = y_0, \qquad (6.7)$$

де $y' = \frac{\partial y}{\partial t}$, $y(t) = (y_1(t), y_2(t), \ldots, y_n(t))^T$ і $f = (f_1, f_2, \ldots, f_{n_V})^T$ – n-вимірні вектори; $f(y,t)$ – вектор-функція, яка визначена на деякій $(n+1)$-вимірній ($y \in R^n, t \in R^1$) множині і є неперервною.

Оскільки математичні схеми такого виду відображають динаміку системи, що вивчається, тобто її поведінка в часі, то вони називаються ***D-схемами*** (англ. dynamic).

Найбільш важливе для системотехніки застосування *D-схем* як математичного апарату в теорії автоматичного управління [11]. Для ілюстрації особливостей побудови і застосування *D-схем* розглянемо простий приклад формалізації процесу функціонування двох елементарних систем різної фізичної природи: механічної S_M (коливання маятника, рис. 6.1, а) і електричної S_K (коливальний контур рис. 6.1, б) [11].

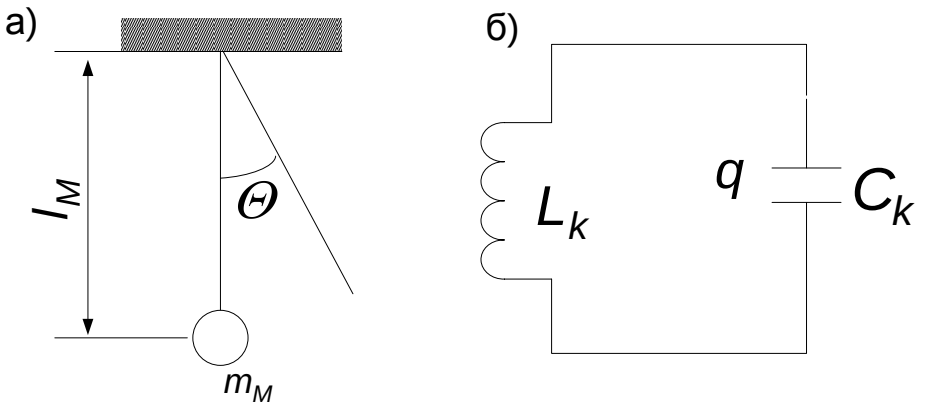


Рис. 6.1. **Елементарні схеми**

Процес малих коливань маятника описується звичайним диференціальним рівнянням

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

де m_M, l_M – маса і довжина підвісу маятника; g – прискорення вільного падіння; $\Theta(t)$ – кут відхилення маятника у момент часу t.

З цього рівняння вільного коливання маятника можна знайти оцінки характеристик, що цікавлять дослідника. Наприклад, період коливання маятника:

$$T_M = 2\pi\sqrt{\frac{l_M}{g}}.$$

Аналогічно, процеси в електричному коливальному контурі описуються звичайним диференціальним рівнянням:

$$L_K \frac{d^2q(t)}{dt^2} + \frac{q(t)}{C_K} = 0,$$

де L_K, C_K – індуктивність і ємкість конденсатора; $q(t)$ – заряд конденсатора в момент часу t.

З цього рівняння можна отримати різні оцінки характеристик процесу в коливальному контурі. Наприклад, період електричних коливань:

$$\mathrm{T_K} = 2\pi\sqrt{L_K C_K}.$$

Очевидно, що ввівши позначення $h_0 = m_M l_M^2 = L_K$, $h_1 = 0$, $h_2 = m_M g l_M = \frac{l}{C_K}$, $\Theta(t) = q(t) = z(t)$, отримаємо звичайне диференціальне рівняння другого порядку, яке описує поведінку цієї замкнутої системи:

$$h_0 \frac{d^2 z(t)}{dt^2} + h_1 \frac{dz(t)}{dt} + h_2 z(t) = 0, \qquad (6.8)$$

де h_0, h_1, h_2 – параметри системи; $z(t)$ – стан системи у момент часу t.

Таким чином, поведінка цих двох об'єктів може бути досліджена на основі загальної математичної моделі (6.8). Крім того, необхідно відзначити, що поведінка однієї з систем може бути проаналізована за допомогою іншої. Наприклад, поведінка маятника (системи S_M) може бути вивчена за допомогою електричного коливального контура (системи S_K).

Якщо система S (тобто маятник або контур), що вивчається, взаємодіє з зовнішнім середовищем E, то з'являється вхідна дія $x(t)$ (зовнішня сила для маятника і джерело енергії для контура) і неперервно-детермінована модель такої системи матиме вигляд:

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

З точки зору загальної схеми математичної моделі (див. підрозділ 6.2) $x(t)$ є вхідним (управляючим) впливом, а стан системи S у даному випадку можна розглядати як вихідну характеристику $y(t)$, тобто вважати, що вихідна змінна $y(t)$ співпадає зі станом системи $z(t)$ в кожен момент часу, тобто $y(t) = z(t)$.

При розв'язанні задач системотехніки важливе значення мають проблеми управління великими системами. Слід звернути увагу на системи автоматичного управління – окремий випадок динамічних

систем, описуваних *D-схемами* і виділених в окремий клас моделей через їх практичну специфіку [11].

Описуючи процеси автоматичного управління, дотримуються зазвичай представлення реального об'єкта у вигляді двох систем: управляючої і управляємої (об'єкта управління). Структура багатовимірної системи автоматичного управління загального виду представлена на рис. 6.2, де позначені **незалежні** (ендогенні) **змінні**: $x(t)$ – вектор вхідних (задаваних) впливів; $\nu(t)$ – вектор збуджуючих впливів; $h'(t)$ – вектор сигналів помилки; $h''(t)$ – вектор управляючих впливів; **залежні** (екзогенні) **змінні**: $z(t)$ – вектор станів системи S; $y(t)$ – вектор вихідних змінних (зазвичай $y(t) = z(t)$).

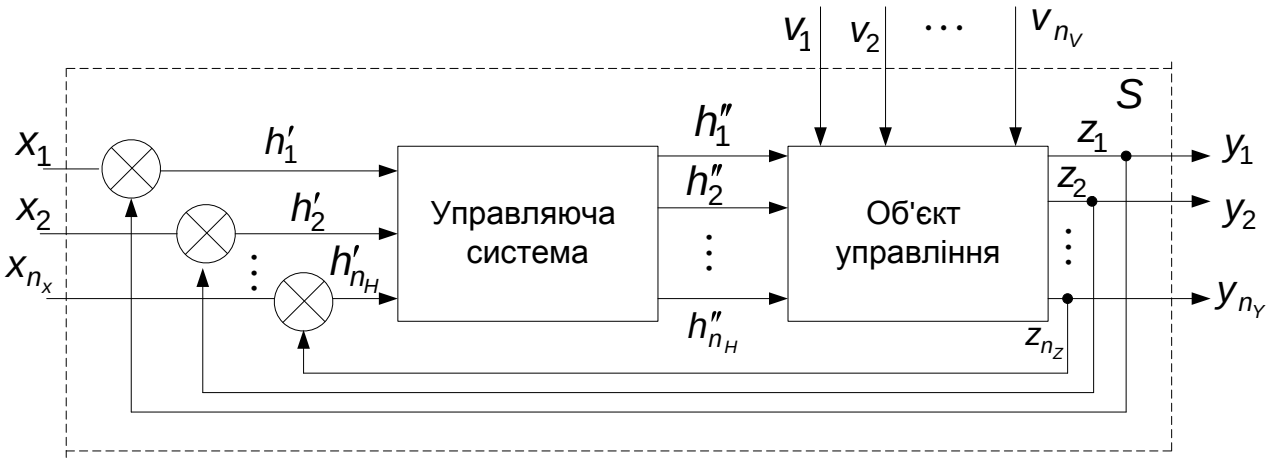


Рис. 6.2. **Структура системи автоматичного управління**

Сучасна управляюча система – це сукупність програмно-технічних засобів, що забезпечують досягнення об'єктом управління певної мети. Наскільки точно об'єкт управління досягає заданої мети, можна судити для одновимірної системи за координатою стану $y(t)$. Різниця між заданим $y_{зад}(\mathrm{t})$ і дійсним $y(t)$ законами зміни управляємої величини є помилка управління $h'(t) = y_{зад}(t) - y(t)$.

Системи, для яких помилки управління $h'(t) = 0$ у всі моменти часу, називаються ідеальними. На практиці реалізація ідеальних систем неможлива. Таким чином, помилка $h'(t)$ – необхідна складова автомАтичного управління, заснованого на принципі заперечного зворотного

зв'язку, оскільки для приведення у відповідність вихідної змінної $y(t)$ її заданому значенню використовується інформація про відхилення між ними. Завданням системи автоматичного управління є зміна змінної $y(t)$ згідно із заданим законом з певною точністю (з допустимою помилкою). При проектуванні й експлуатації систем автоматичного управління необхідно вибрати такі параметри системи S, які забезпечили б необхідну точність управління, а також стійкість системи в перехідному процесі [11].

Якщо система стала, то представляє практичний інтерес поведінка системи в часі, максимальне відхилення регульованої змінної $y(t)$ в перехідному процесі, час перехідного процесу і т. п. Висновки про властивості систем автоматичного управління різних класів можна зробити за видом диференціальних рівнянь, що приблизно описують процеси в системах. Порядок диференціального рівняння і значення його коефіцієнтів цілком визначаються статичними і динамічними параметрами системи S.

Таким чином, використання ***D-схем*** дозволяє формалізувати процес функціонування неперервно-детермінованих систем S й оцінити їх основні характеристики, застосовуючи аналітичний або імітаційний підхід, реалізований у вигляді відповідної мови для моделювання неперервних систем, або такий, що використовує аналогові і гібридні засоби обчислювальної техніки.

6.4. Дискретно-детерміновані моделі (F-схеми)

Особливості дискретно-детермінованого підходу на етапі формАлізації процесу функціонування систем розглянемо на прикладі використання як математичного апарата теорії автоматів. ***Теорія автоматів*** – це розділ теоретичної кібернетики, в якому вивчаються математичні моделі – автомати. На основі цієї теорії система представляється у вигляді автомата, що переробляє дискретну інформацію і змінює свої внутрішні стани лише в допустимі моменти часу. Поняття автомат варіюється залежно від характеру систем, що конкретно вивчаються, від прийнятого рівня абстракції і доцільного ступеня спільності [7].

Автомат можна представити як деякий пристрій (чорний ящик), на який подаються вхідні сигнали і знімаються вихідні і який може мати деякі внутрішні стани. ***Скінченним автоматом*** називається автомат, у

якого множина внутрішніх станів і вхідних сигналів (а, отже, і множина вихідних сигналів) є скінченними множинами.

Абстрактно скінченний автомат можна представити як математичну схему (***F-схему***), що характеризується шістьма елементами:

– скінченною множиною X вхідних сигналів (вхідним алфавітом);

– скінченною множиною Y вихідних сигналів (вихідним алфавітом);

– скінченною множиною Z внутрішніх станів (внутрішнім алфавітом або алфавітом станів);

– початковим станом $z_0 \in Z$;

– функцією переходів $\phi(z,x)$;

– функцією виходів $\varphi(z,x)$.

Автомат, що задається ***F-схемою***: $F = \langle Z, X, Y, \phi, \varphi, z_0 \rangle$, функціонує в дискретному автоматному часі, моментами якого є такти, тобто рівні інтервали часу, що примикають один до одного, кожному з яких відповідають постійні значення вхідного і вихідного сигналів та внутрішні стани. Позначимо стан, а також вхідний і вихідний сигнали, що відповідають t-му такту при $t = 0,1,2,\ldots$, через $z(t)$, $x(t)$, $y(t)$. При цьому, за умовою, $z(0) = z_0$, а $z(t) \in Z$, $x(t) \in X$, $y(t) \in Y$.

Абстрактний скінченний автомат має один вхідний ($X \subset R^1$) й один вихідний канали ($Y \subset R^1$). У кожен момент $t = 0,1,2,\ldots$ дискретного часу *F-автомат* перебуває в певному стані $z(t)$ з множини Z станів автомата, причому в початковий момент часу $t = 0$ він завжди перебуває в початковому стані $z(0) = z_0$. У момент t, будучи в стані $z(t)$, автомат здатний сприйняти на вхідному каналі сигнал $x(t) \in X$ і видати на вихідному каналі сигнал $y(t) = \psi[z(t), x(t)]$, переходячи в стан $z(t+1) = \varphi[z(t), x(t)]$, $z(t) \in Z$, $y(t) \in Y$. Абстрактний скінченний автомат реалізує деяке відображення множини слів вхідного алфавіту X на множину слів вихідного алфавіту Y. Іншими словами, якщо на вхід скінченного автомата, встановленого в початковий стан z_0, подавати в деякій послідовності літери вхідного алфавіту $x(0)$, $x(1)$, $x(2)$,..., тобто

вхідне слово, то на виході автомата будуть послідовно з'являтися літери вихідного алфавіту $y(0)$, $y(1)$, $y(2)$,..., утворюючи вихідне слово.

Таким чином, робота скінченного автомата відбувається за такою схемою: у кожному t-му такті на вхід автомата, що перебуває в стані $z(t)$, подається деякий сигнал $x(t)$, на який він реагує переходом в $(t+1)$-му такті в новий стан $z(t+1)$ і видачею деякого вихідного сигналу. Сказане вище можна описати такими рівняннями: для *F-автомата* першого роду, що називається також автоматом Мілі,

$$z(t+1) = \varphi[z(t), x(t)], \quad t = 0,1,2,\ldots, \tag{6.9}$$

$$y(t) = \psi[z(t), x(t)], \quad t = 0,1,2,\ldots, \tag{6.10}$$

для *F-автомата* другого роду

$$z(t+1) = \varphi[z(t), x(t)], \quad t = 0,1,2,\ldots, \tag{6.11}$$

$$y(t) = \psi[z(t), x(t-1)], \quad t = 1,2,3,\ldots. \tag{6.12}$$

Автомат другого роду, для якого

$$y(t) = \psi[z(t)], \quad t = 0,1,2,\ldots, \tag{6.13}$$

тобто функція виходів не залежить від вхідної змінної $x(t)$, називається автоматом Мура.

Таким чином, рівняння (6.9) – (6.13), що повністю задають ***F-автомат***, є окремим випадком рівнянь (6.3) і (6.4), коли система S детермінована і на її єдиний вхід надходить дискретний сигнал X.

За кількістю станів розрізняють скінченні автомати з пам'яттю і без пам'яті. Автомати з пам'яттю мають більше одного стану, а автомати без пам'яті (комбінаційні або логічні схеми) мають лише один стан. При цьому, згідно з (6.10), робота комбінаційної схеми полягає в тому, що вона ставить у відповідність кожному вхідному сигналу $x(t)$ певний вихідний сигнал $y(t)$, тобто реалізує логічну функцію виду:

$$y(t) = \psi[x(t)], \quad t = 0,1,2,\ldots.$$

Ця функція називається булевою, якщо алфавіти X і Y, яким належать значення сигналів x і y, складаються з двох літер.

За характером відліку дискретного часу скінченні автомати поділяються на ***синхронні*** і ***асинхронні***. У синхронних *F-автоматах* моменти часу, в які автомат "зчитує" вхідні сигнали, визначаються примусово синхронізуючими сигналами. Після чергового синхронізуючого сигналу з урахуванням "зчитаного" і відповідно до рівнянь (6.9) – (6.13) відбувається перехід у новий стан і видача сигналу на виході, після чого автомат може сприймати наступне значення вхідного сигналу. Таким чином, реакція автомата на кожне значення вхідного сигналу закінчується за один такт, тривалість якого визначається інтервалом між сусідніми синхронізуючими сигналами. Асинхронний *F-автомат* зчитує вхідний сигнал неперервно і тому, реагуючи на достатньо довгий вхідний сигнал постійної величини x, він може, як випливає з (6.9) – (6.13), кілька разів змінювати стан, видаючи відповідне число вихідних сигналів, поки не перейде у сталий, який вже не може бути змінений даним вхідним.

Щоб задати скінченний *F-автомат*, необхідно описати всі елементи множини $F = \langle Z, X, Y, \phi, \varphi, z_0 \rangle$, тобто вхідний, внутрішній і вихідний алфавіти, а також функції переходів і виходів. Причому серед множини станів необхідно виділити стан z_0, в якому автомат перебував у момент часу $t = 0$. Існують декілька способів задавання роботи F-автоматів, але найчастіше використовуються табличний, графічний і матричний.

Найпростіший табличний спосіб задавання скінченного автомата заснований на використанні таблиць переходів і виходів, рядки яких відповідають вхідним сигналам автомата, а стовпці – його станам. При цьому зазвичай перший зліва стовпець відповідає початковому стану z_0. На перетині i-го рядка і k-го стовпця таблиці переходів поміщається відповідне значення $\varphi(z_k, x_i)$ функції переходів, а в таблиці виходів – відповідне значення $\psi(z_k, x_i)$ функції виходів. Для *F-автомата* Мура обидві таблиці можна поєднати, отримавши так звану відмічену таблицю переходів, в якій над кожним станом z_k автомата, що позначає стовпець таблиці, стоїть відповідний цьому стану, згідно з (6.10), вихідний сигнал $\psi(z_i)$.

Опис роботи *F-автомата* Мілі таблицями переходів ψ і виходів φ ілюструється табл. 6.1, а опис *F-автомата* Мура – таблицею переходів (табл. 6.2).

Таблиця 6.1

Таблиця переходів F-автомата Мілі

x_i	z_k			
	z_0	z_1	…	z_K
Переходи				
x_1	$\varphi(z_0, x_1)$	$\varphi(z_1, x_1)$	…	$\varphi(z_K, x_1)$
x_2	$\varphi(z_0, x_2)$	$\varphi(z_1, x_2)$	…	$\varphi(z_K, x_2)$
…	…	…	…	…
x_i	$\varphi(z_0, x_i)$	$\varphi(z_1, x_i)$	…	$\varphi(z_K, x_i)$
Виходи				
x_1	$\psi(z_0, x_1)$	$\psi(z_1, x_1)$	…	$\psi(z_K, x_1)$
x_2	$\psi(z_0, x_2)$	$\psi(z_1, x_2)$	…	$\psi(z_K, x_2)$
…	…	…	…	…
x_i	$\psi(z_0, x_i)$	$\psi(z_1, x_i)$	…	$\psi(z_K, x_i)$

Таблиця 6.2

Таблиця переходів F-автомата Мура

x_i	$\psi(z_k)$			
	$\psi(z_0)$	$\psi(z_1)$	…	$\psi(z_K)$
	z_0	z_1	…	z_K
x_1	$\varphi(z_0, x_1)$	$\varphi(z_1, x_1)$	…	$\varphi(z_K, x_1)$
x_2	$\varphi(z_0, x_2)$	$\varphi(z_1, x_2)$	…	$\varphi(z_K, x_2)$
…	…	…	…	…
x_i	$\varphi(z_0, x_i)$	$\varphi(z_1, x_i)$	…	$\varphi(z_K, x_i)$

При другому способі задавання скінченного автомата використовується поняття направленого графа. Граф автомата є набором вершин, що відповідають різним станам автомата, і дуг, що з'єднують вершини графа і відповідають тим чи іншим переходам автомата. Якщо вхідний сигнал x_k спричиняє перехід із стану z_i у стан z_j, то на графі автомата дуга, що з'єднує вершину z_i з вершиною z_j, позначається x_k. Для того щоб задати функцію виходів, дуги графа необхідно позначити відповідними вихідними сигналами. Для автоматів Мілі ця розмітка виконується так: якщо вхідний сигнал x_k діє на стан z_i, то, згідно зі сказаним, отримується дуга, що виходить з z_i і позначена x_k; цю дугу додатково позначають вихідним сигналом $y = \psi(z_i, x_k)$. Для автомата Мура аналогічна розмітка графа така: якщо вхідний сигнал x_k, діючи на деякий стан автомата, спричиняє перехід у стан z_j, то дугу, направлену в z_j і позначену x_k, додатково позначають вихідним сигналом $y = \psi(z_j, x_k)$.

На рис. 6.3 а, б, наведені задані раніше таблицями *F-автомати* Мілі $F1$ і Мура $F2$ відповідно.

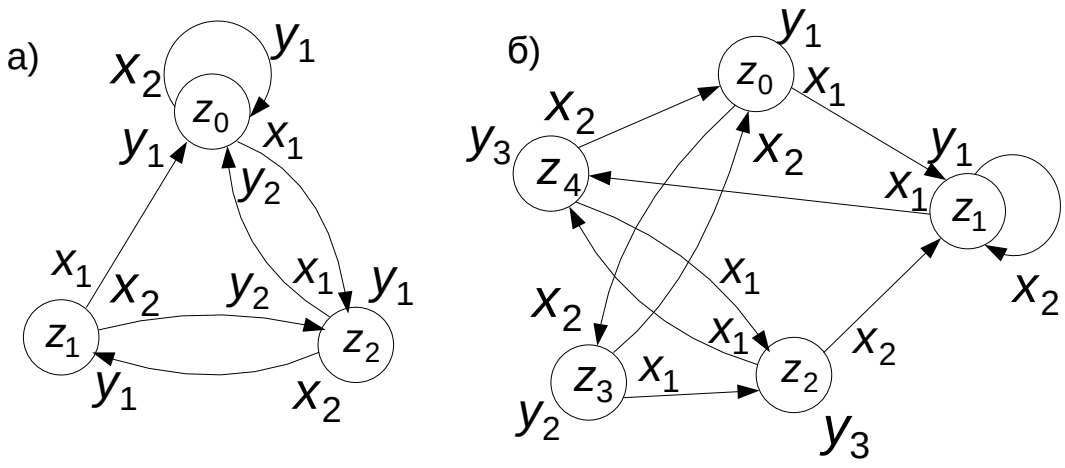


Рис. 6.3. **Графи автоматів Мілі (а) і Мура (б)**

При розв'язанні задач моделювання систем часто зручнішою формою є матричне задавання скінченного автомата.

Таким чином, поняття *F-автомата* в дискретно-детермінованому підході до дослідження на моделях властивостей об'єктів є марема-

тичною абстракцією, зручною для опису широкого класу процесів функціонування реальних об'єктів в автоматизованих системах управління. В якості таких об'єктів в першу чергу слід назвати елементи і вузли ЕОМ, пристрої контролю, регулювання й управління, системи часової і просторової комутації в техніці обміну інформацією тощо. Для всіх перерахованих об'єктів характерна наявність дискретних станів і дискретний характер роботи в часі, тобто їх опис за допомогою ***F-схем*** є ефективним. Але широта їх застосування не означає універсальності цих математичних схем. Наприклад, цей підхід не придатний для опису процесів прийняття рішень, процесів у динамічних системах з наявністю перехідних процесів і стохастичних елементів.

6.5. Дискретно-стохастичні моделі (P-схеми)

Розглянемо особливості побудови математичних схем при дискетно-стохастичному підході до формалізації процесу функціонування досліджуваної системи S. Оскільки сутність дискретизації часу при цьому підході залишається аналогічною розглянутим у підрозділі 6.4 скінченним автоматам, то вплив фактора стохастичності прослідкуємо також на різновиді таких автоматів, а саме на імовірнісних (стохастичних) автоматах [11].

У загальному вигляді ***імовірнісний автомат*** можна визначити як дискретний потактовий перетворювач інформації з пам'яттю, функціонування якого в кожному такті залежить тільки від стану пам'яті в ньому і може бути описане статистично.

Застосування схем імовірнісних автоматів (***P-схем***) має важливе значення для розробки методів проектування дискретних систем, що проявляють статистично закономірну випадкову поведінку, для з'ясування алгоритмічних можливостей таких систем і обґрунтування меж доцільності їх використання, а також для розв’язання задач синтезу за вибраним критерієм дискретних стохастичних систем, що задовольняють заданим обмеженням.

Введемо математичне поняття ***P-автомата***, використовуючи поняття, введені для ***F-автомата***. Розглянемо множину G, елементами якої є всілякі пари (x_i, z_s), де x_i і z_s – елементи вхідної підмножини X і підмножини станів Z відповідно. Якщо існують дві такі функції φ і ψ, що з їх допомогою здійснюються відображення $G \to Z$ і $G \to Y$,

то говорять, що $F = \langle Z, X, Y, \phi, \varphi \rangle$ визначає автомат детермінованого типу.

Введемо в розгляд більш загальну математичну схему. Нехай Φ – множина всіляких пар виду (z_k, y_i), де y_i – елемент вихідної підмножини Y. Зажадаємо, щоб будь-який елемент множини G індукував на множині Φ деякий закон розподілу такого вигляду:

Елементи з F (x_i, z_s)	(z_1, y_1)	(z_1, y_2)	…	(z_K, y_{J-1})	(z_K, y_J)
	b_{11}	b_{12}	…	$b_{K(J-1)}$	b_{KJ}

При цьому

$$\sum_{k=1}^{K} \sum_{j=1}^{J} b_{kj} = 1,$$

де b_{kj} – ймовірність переходу автомата в стан z_k і появи на виході сигналу y_i, якщо він був у стані z_s і на його вхід у цей момент часу надійшов сигнал x_i. Кількість таких розподілів, представлених у вигляді таблиць, дорівнює кількості елементів множини G. Позначимо множину цих таблиць через B. Тоді четвірка елементів $P = \langle Z, X, Y, B \rangle$ називається ймовірнісним автоматом (***P-автоматом***).

6.6. Неперервно-стохастичні моделі (Q-схеми)

Особливості неперервно-стохастичного підходу розглянемо на прикладі використання в якості типових математичних схем ***систем масового обслуговування (СМО)***, які називатимемо ***Q-схемами***. Системи масового обслуговування є класом математичних схем, розроблених у теорії масового обслуговування і різних застосуваннях для формалізації процесів функціонування систем, які за своєю суттю є процесами обслуговування.

Як процес обслуговування можуть бути представлені різні по своїй фізичній природі процеси функціонування економічних, виробничих, технічних й інших систем, наприклад: потоки постачань продукції дея-

кому підприємству, потоки деталей і комплектуючих виробів на складальному конвеєрі цеху, заявки на обробку інформації ЕОМ від віддалених терміналів і т. д. При цьому характерною для роботи таких об'єктів є випадкова поява заявок (вимог) на обслуговування і завершення обслуговування у випадкові моменти часу, тобто стохастичний характер процесу їх функціонування.

Таким чином, 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.

Зупинимося на основних поняттях масового обслуговування, необхідних для використання ***Q-схем*** як при аналітичному, так і при імітаційному підходах.

У будь-якому елементарному акті обслуговування можна виділити дві основні складові: очікування обслуговування заявкою і саме обслуговування заявки. Це можна зобразити у вигляді деякого i-го приладу обслуговування $П_i$ (рис. 6.4), що складається з накопичувача заявок H_i, в якому може одночасно перебувати $l_i = 0,\ L_i^H$ заявок, де L_i^H – місткість l-го накопичувача і канала обслуговування заявок (або просто канала) K_i На кожен елемент приладу обслуговування $П_i$ надходять потоки подій, в накопичувач H_i – потік заявок w_i, на канал K_i – потік обслуговувань $u_{i\,i}$.

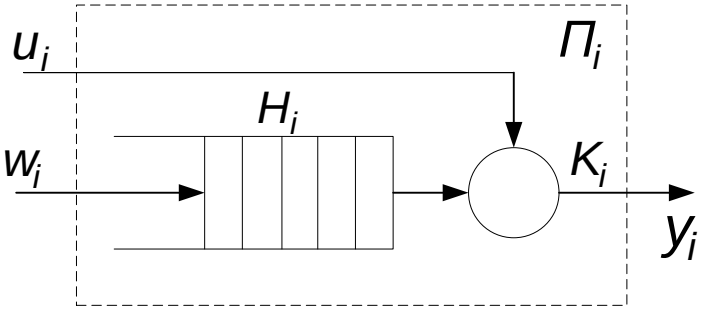


Рис. 6.4. **Прилад обслуговування заявок**

Потоком подій називається послідовність подій, що відбуваються одна за одною в якісь випадкові моменти часу. Розрізняють потоки

однорідних і неоднорідних подій. ***Потік подій*** називається ***однорідним***, якщо він характеризується тільки моментами надходження цих подій (збуджуючими моментами) і задається послідовністю $\{t_n\} = \{0 \le t_1 \le t_2 \ldots \le t_n\}$, де t_n – момент надходження n-ї події (невід'ємне дійсне число). Однорідний потік подій також може бути заданий у вигляді послідовності проміжків часу між n-ю і $(n-1)$-ю подіями $\{\tau_n\}$, яка однозначно пов'язана з послідовністю збуджуючих моментів $\{t_n\}$, де $\tau_n = t_n - t_{n-1}$, $n \ge 1$, $t_0 = 0$, тобто $\tau_1 = t_1$.

Потоком неоднорідних подій називається послідовність $\{t_n, f_n\}$, де $\{t_n\}$ – збуджуючі моменти; f_n – набір ознак події. Наприклад, стосовно процесу обслуговування для неоднорідного потоку заявок може бути задана приналежність до того або іншого джерела заявок, наявність пріоритету, можливість обслуговування тим або іншим типом каналу і т. п. [11].

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. Зазвичай 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 шпиталю та ін.

Найчастіше ефективність функціонування будь-якої СМО визначається за такими показниками [12]:

– 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.

Більш детально СМО та їх характеристики будуть розглянуті в підрозділі 9.3.

6.7. Узагальнені моделі (А-схеми)

Найбільш відомим загальним підходом до формального опису процесів функціонування систем є підхід, запропонований Н. П. Бусленко. Цей підхід дозволяє описувати поведінку неперервних і дискретних, детермінованих і стохастичних систем, тобто порівняно з розглянутими раніше є узагальненим (універсальним) і базується на понятті ***агре-***

гативної системи (англ. aggregate system), що є формальною схемою загального вигляду, яку називатимемо ***А-схемою*** [11].

Аналіз існуючих засобів моделювання систем і завдань, що вирішуються за допомогою метода моделювання на ЕОМ, неминуче призводить до висновку, що комплексне вирішення проблем, які виникають у процесі створення і машинної реалізації моделі, можливо лише у випадку, якщо моделюючі системи мають у своїй основі єдину нормальну математичну схему, тобто ***А-схему***. Така схема повинна одночасно виконувати декілька функцій: бути адекватним математичним описом об'єкта моделювання, тобто системи S, служити основою для побудови алгоритмів і програм при комп'ютерній реалізації моделі M, дозволяти у спрощеному варіанті (для окремих випадків) проводити аналітичні дослідження.

Наведені вимоги певною мірою суперечливі. Проте у рамках узагальненого підходу на основі ***А-схем*** вдається знайти між ними деякий компроміс.

За традицією, що встановилася в математиці взагалі і у прикладній математиці зокрема, при агрегативному підході спочатку дається формальне визначення об'єкта моделювання – агрегативної системи, яка є математичною схемою, що відображає системний характер об'єктів, які вивчаються. При агрегативном описі складний об'єкт (система) розбивається на скінченну кількість частин (підсистем), зберігаючи при цьому зв'язки, що забезпечують їх взаємодію. Якщо деякі з отриманих підсистем виявляються у свою чергу ще досить складними, то процес їх розбиття триває до тих пір, поки не утворяться підсистеми, які в умовах даного завдання моделювання можуть вважатися зручними для математичного опису. Унаслідок такої декомпозиції складна система представляється у вигляді багаторівневої конструкції з взаємозв'язаних елементів, об'єднаних у підсистеми різних рівнів [11].

Як елемент ***А-схеми*** виступає агрегат, а зв'язок між агрегатами (усередині системи S і з зовнішнім середовищем E) здійснюється за допомогою оператора спряження R. Очевидно, що агрегат сам може розглядатися як ***А-схема***, тобто може розбиватися на елементи (агрегати) наступного рівня.

Будь-який агрегат характеризується такими множинами: моментів часу T, вхідних X і вихідних Y сигналів, станів Z в кожен момент часу

t. Стан агрегата у момент часу $t \in T$ позначається як $z(t) \in Z$, а вхідні і вихідні сигнали, як $x(t) \in X$ і $y(t) \in Y$ відповідно [11].

Вважатимемо, що перехід агрегата із стану $z(t_1)$ в стан $z(t_2) \neq z(t_1)$ відбувається за малий інтервал часу, тобто має місце стрибок δz. Переходи агрегата з стану $z(t_1)$ в $z(t_2)$ визначаються власними (внутрішніми) параметрами самого агрегата $h(t) \in H$ і вхідними сигналами $x(t) \in X$.

У початковий момент часу t_0 стани z мають значення, що дорівнюють z^0, тобто $z^0 = z(t_0)$, що задаються законом розподілу процеса $z(t)$ в момент часу t_0, а саме $L[z(t_0)]$. Припустимо, що процес функціонування агрегата у випадку впливу вхідного сигналу x_n описується випадковим оператором V. Тоді в момент $t_n \in T$ надходження в агрегат вхідного сигналу x_n можна визначити стан

$$z(t_n + 0) = V[t_n, z(t_n), x_n].$$

Позначимо напівінтервал часу $t_1 < t \leq t_2$ як $(t_1, t_2]$, а напівінтервал $t_1 \leq t < t_2$ як $[t_1, t_2)$. Якщо інтервал часу (t_n, t_{n+1}) не містить жодного моменту надходження сигналів, то для $t \in (t_n, t_{n+1})$ стан агрегата визначається випадковим оператором U відповідно до співвідношення:

$$z(t) = U[t, t_n, z(t_n + 0)].$$

Сукупність випадкових операторів V і U розглядається як оператор переходів агрегата в нові стани. При цьому процес функціонування агрегата складається зі стрибків станів δz в моменти надходження вхідних сигналів x (оператор V) і змін станів між цими моментами t_n і t_{n+1} (оператор U). На оператор U не накладається ніяких обмежень, тому допустимі стрибки станів δz в моменти часу, які не є моментами надходження вхідних сигналів x. У подальшому моменти стрибків δz будемо називати особливими моментами часу t_δ, а стани $z(t_\delta)$ – особливими станами ***А-схеми***. Для опису стрибків станів

δz в особливі моменти часу t_δ будемо використовувати випадковий оператор W, що є частковим випадком оператора U, тобто

$$z(t_\delta + 0) = W[t_\delta, z(t_\delta)].$$

У множині станів Z виділяється така підмножина $Z^{(Y)}$, що якщо $z(t_\delta)$ досягає $Z^{(Y)}$, то цей стан є моментом видачі вихідного сигналу, що визначається оператором виходів:

$$y = G[t_\delta, z(t_\delta)].$$

Таким чином, під ***агрегатом*** будемо розуміти будь-який об'єкт, що визначається впорядкованою сукупністю розглянутих множин T, X, Y, Z, $Z^{(Y)}$, H і випадкових операторів V, U, W, G.

Послідовність вхідних сигналів, розташованих у порядку їх надходження в ***А-схему***, називатимемо ***вхідним повідомленням*** або x***-повідомленням***. Послідовність вихідних сигналів, впорядковану відносно часу видачі, назвемо ***вихідним повідомленням*** або y***-повідомленням***.

Існує клас великих систем, які, зважаючи на їх складність, не можуть бути формалізовані у вигляді математичних схем одиночних агрегатів, тому їх формалізують деякою конструкцією з окремих агрегатів A_k, $k = \overline{1, N_A}$, яку назвемо агрегативною системою або ***А-схемою***. Для опису деякої реальної системи S у вигляді *А-схеми* необхідно мати опис як окремих агрегатів A_k, так і зв'язків між ними.

Приклад. Розглянемо *А-схему*, структура якої наведена на рис. 6.5 [11].

Функціонування *А-схеми* пов'язане з переробкою інформації, передача останньої на схемі показана стрілками. Уся інформація, що циркулює в *А-схемі*, поділяється на зовнішню і внутрішню. Зовнішня інформація надходить від зовнішніх об'єктів, що не є елементами схеми, яка розглядається, а внутрішня інформація виробляється агрегатами самої *А-схеми*. Обмін інформацією між *А-схемою* і зовнішнім середовищем E відбувається через агрегати A_k, які називаються полюсами *А-схеми*. При цьому розрізняють вхідні полюси *А-схеми*, що є агре-

гатами, на які надходять x-повідомлення (агрегати A_1, A_2, A_3), і вихідні полюси *А-схеми*, вихідна інформація яких є y-повідомленнями (агрегати A_1, A_3, A_4, A_5, A_6). Агрегати, що не є полюсами, називаються внутрішніми.

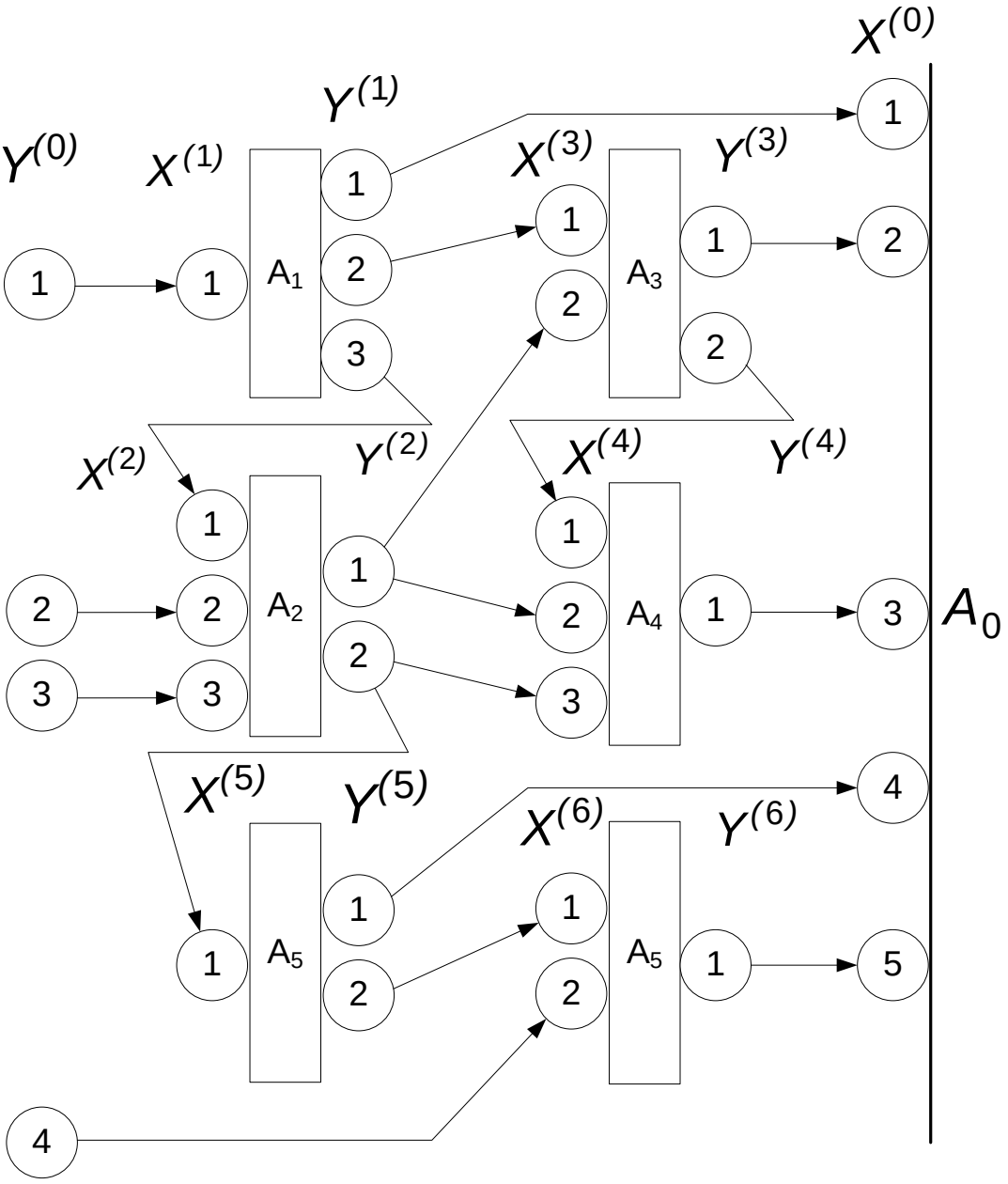


Рис. 6.5. **Структура агрегативної системи**

Кожний k-й агрегат *А-схеми* A_k має вхідні контакти, на які надходить сукупність елементарних сигналів $X_i^{(k)}(t), i = \overline{1, I_k}$, що одночасно виникають на вході елемента, і вихідні контакти, з яких знімається

сукупність елементарних сигналів $Y_j^{(k)}(t),\ j=\overline{1,J_k}$. Таким чином, кожен агрегат *А-схеми* A_k має I_k вхідних і J_k вихідних контактів.

Опис окремого агрегата вже розглянуто, тому для побудови формального поняття *А-схеми* залишається вибрати достатньо зручні способи математичного опису взаємодії між агрегатами. Для цього введемо ряд припущень про закономірності функціонування *А-схем*, що добре узгоджуються з досвідом дослідження реальних складних систем [11]:

1) взаємодія між *А-схемою* і зовнішнім середовищем E, а також між окремими агрегатами всередині системи S здійснюється при передачі сигналів, причому взаємні впливи, що мають місце поза механізмом обміну сигналами, не враховуються;

2) для опису сигналу достатньо деякого скінченного набору характеристик;

3) елементарні сигнали миттєво передаються в *А-схемі* незалежно один від одного за елементарними каналами;

4) до вхідного контакту будь-якого елемента *А-схеми* підключається не більше одного елементарного каналу, до вихідного контакту – будь-яке скінченне число елементарних каналів за умови, що до входу одного і того ж елемента А-схеми направляється не більше ніж один із згаданих елементарних каналів.

Взаємодія *А-схеми* із зовнішнім середовищем E розглядається як обмін сигналами між зовнішнім середовищем E й елементами *А-схеми*. Відповідно до цього зовнішнє середовище E можна представити у вигляді фіктивного елемента системи A_0, вхід якого містить I_0 вхідних контактів $X_i^0(t),\ i=\overline{1,I_0}$, а вихід – J_0 вихідних контактів $Y_j^0(t),\ j=\overline{1,J_0}$. Сигнал, що видається *А-схемою* в зовнішнє середовище E, приймається елементом A_0 як вхідний сигнал, що складається з елементарних сигналів $X_i^0(t),\ i=\overline{1,I_0}$. Сигнал, що надходить в *А-схему* із зовнішнього середовища E, є вихідним сигналом елемента A_0 і складається з елементарних сигналів $Y_j^0(t),\ j=\overline{1,J_0}$.

Таким чином, подальше використання узагальненої типової математичної схеми моделювання, тобто *А-схеми*, у принципі не

відрізняється від розглянутих раніше *D-, F-, P- і Q-схем*. Для окремого випадку, а саме для кусочно-лінійних агрегатів, результати можуть бути отримані аналітичним методом. У складніших випадках, коли застосування аналітичних методів неефективне або неможливе, удаються до імітаційного методу. Причому, представлення об'єкта моделювання у вигляді *A-схеми* може бути тим фундаментом, на якому базується побудова імітаційної системи і її зовнішнього та внутрішнього математичного забезпечення. Стандартна форма представлення досліджуваного об'єкта у вигляді *A-схеми* приводить до уніфікації не тільки алгоритмів імітації, але і до можливості застосовувати стандартні методи обробки й аналізу результатів моделювання системи S [11].

Висновки

1. Поняття математичної схеми дозволяє розглядати математику не як метод розрахунку, а як метод мислення.

2. Математична схема – це ланка при переході від змістовного до формального опису процесу функціонування системи з урахуванням дії зовнішнього середовища.

3. Під математичною моделлю системи розуміють скінченну підмножину незалежних і залежних змінних разом з математичними зв'язками між ними.

4. У практиці моделювання об'єктів у галузі системотехніки і системного аналізу на початкових етапах дослідження системи раціонально використовувати типові математичні схеми.

5. Агрегативні моделі дозволяють описати широке коло об'єктів дослідження з відображенням системного характеру цих об'єктів.

Контрольні запитання та завдання

1. Що називається математичною схемою?
2. Які змінні в моделі системи є незалежними і залежними?
3. Що називається законом функціонування системи?
4. Що розуміється під алгоритмом функціонування системи?
5. Які умови й особливості використання при розробці моделей систем типових математичних схем?

Тема 7. Імовірнісне моделювання. Моделювання випадкових процесів

При дослідженні систем методом імітаційного моделювання сама випадковість безпосередньо включається у процес моделювання і складає його суттєвий елемент. Кожного разу, коли на хід модельованого процесу впливає випадковий фактор, його вплив імітується за допомогою спеціального організованого розіграшу (жеребу). Таким чином, будується одна реалізація випадкового явища, що є як би результатом одного досліду. У процесі моделювання для отримання точнішої оцінки результату формується велике число реалізації (прогонів моделі).

Формування (розігрування) реалізацій випадкових процесів (подій, величин і функцій) із заданими характеристиками називають ***моделюванням випадкових процесів***.

Проте програми вироблення (генератори) випадкових чисел з необхідним законом розподілу можуть виявитися дуже громіздкими. Тому випадкові числа з необхідним законом розподілу отримують не безпосередньо, а шляхом перетворення випадкових чисел, що мають деякий початковий розподіл. До початкового розподілу висувають такі вимоги: простота отримання чисел на ЕОМ; зручність перетворення випадкових чисел у розподіл із заданим законом. Встановлено, що рівномірний закон розподілу достатньою мірою задовольняє цим вимогам. Нижче буде показано, що моделювання випадкових процесів може бути побудоване на використанні датчика випадкових чисел, рівномірно розподілених в інтервалі (0; 1). Оскільки отримані таким чином лічильники випадкових чисел є в принципі наближеними для відповідних законів розподілу, то їх ще називають ***генераторами псевдовипадкових чисел***.

7.1. Моделювання випадкових процесів

Моделювання простих подій. Нехай ймовірність події A задана: $P(A) = p$. Виберемо за допомогою датчика випадкових чисел деяке число r і вважатимемо, що якщо воно менше або дорівнює p, то подія A відбулася, якщо більше p, то не відбулася, тобто умовою здійснення події є виконання нерівності $r \le p$ (рис. 7.1).

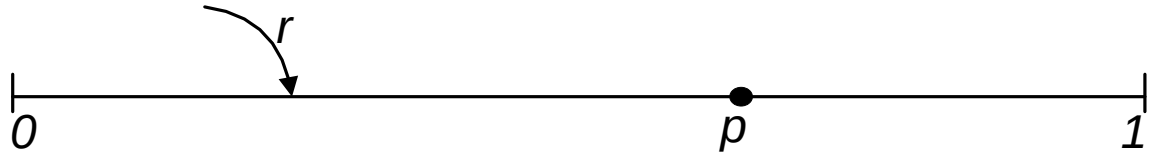


Рис. 7.1. **Моделювання простої події**

Дійсно, якщо ξ – випадкова величина, рівномірно розподілена в інтервалі (0,1), то

$$P(\xi < p) = \int_0^p p_\xi(x)dx = p,$$

де $p_\xi(x)$ – функція щільності розподілу ймовірностей.

Моделювання повної групи подій. Нехай є повна група n подій A_i з ймовірностями p_i .· Оскільки події утворюють повну групу, то $\sum_{i=1}^{n} p_i = 1$.

Розділимо весь інтервал (0,1) на n відрізків, довжини яких складають $p_1, p_2, \ldots, p_n$. Виберемо за допомогою датчика випадкових чисел деяке число r і вважатимемо, що якщо воно потрапило, наприклад, на ділянку p_k, то це означає, що відбулася подія A_k, тобто

$$l_{k-1} < r \le l_k$$

де

$$l_{k-1} = \sum_{i=1}^{k-1} p_i, \quad l_k = \sum_{i=1}^{k} p_i = l_{k-1} + p_k$$

Дійсно

$$P(l_{k-1} < \xi \le l_k) = \int_{l_{k-1}}^{l_k} p_\xi(x)dx = p_k.$$

Процедура моделювання в цьому випадку полягає в послідовному порівнянні випадкових чисел r з величинами l_k (рис. 7.2).

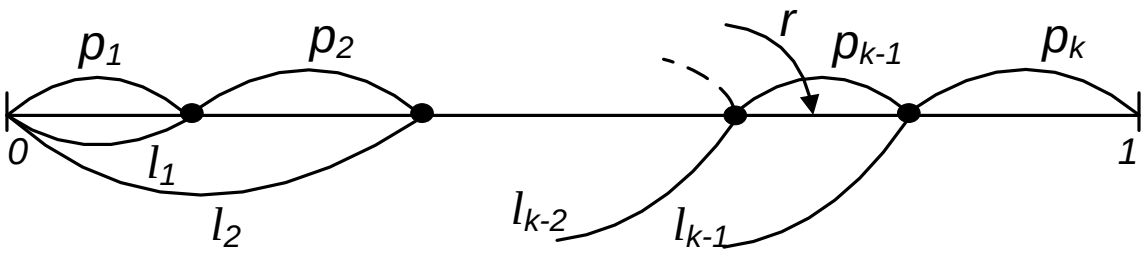


Рис. 7.2. **Моделювання повної групи подій**

Моделювання складних незалежних подій. Часто буває необхідно здійснити моделювання складних подій, що складаються з двох або декількох простих подій. Нехай, наприклад, є дві незалежні події A і B з ймовірностями $P(A)=p_A$ і $P(B)=p_B$. У цьому випадку можливі такі результати сумісних випробувань: $AB, A\overline{B}, \overline{A}B, \overline{A}\overline{B}$, і ймовірності їх відповідно дорівнюють p_Ap_B, $p_A(1-p_B)$, $(1-p_A)p_B$, $(1-p_A)(1-p_B)$. При цьому вони утворюють повну групу подій, оскільки

$$p_Ap_B+p_A(1-p_B)+(1-p_A)p_B+(1-p_A)(1-p_B)=1.$$

Тоді можна змоделювати складні події двома способами.

У першому способі вибираємо за допомогою датчика випадкових чисел два числа r_i, r_{i+1} (рис. 7.3) і моделюємо складні події за схемою:

$$AB \quad при \quad \begin{cases} r_i \le p_A, \\ r_{i+1} \le p_B; \end{cases} \qquad \overline{A}B \quad при \quad \begin{cases} r_i > p_A, \\ r_{i+1} \le p_B; \end{cases}$$

$$A\overline{B} \quad при \quad \begin{cases} r_i \le p_A, \\ r_{i+1} > p_B; \end{cases} \qquad \overline{A}\overline{B} \quad при \quad \begin{cases} r_i > p_A, \\ r_{i+1} > p_B. \end{cases}$$

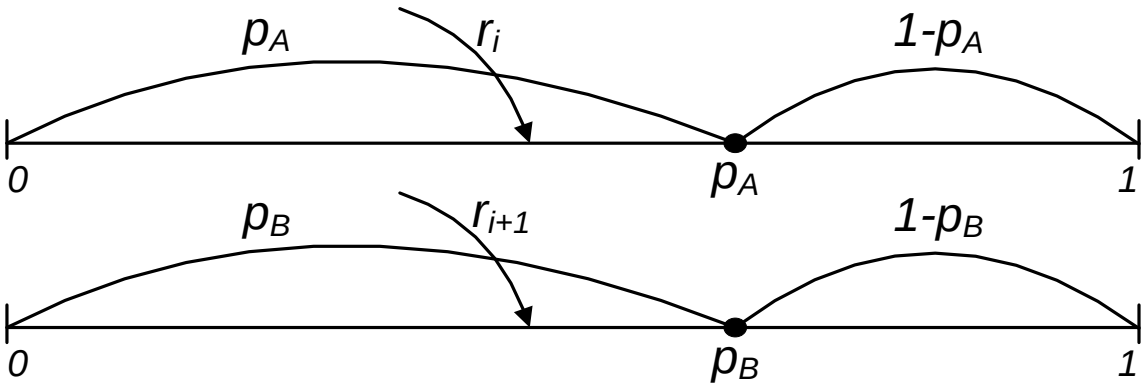


Рис. 7.3. **Моделювання складних незалежних подій (вар. 1)**

У другому випадку, скористаємося схемою моделювання повної групи подій. Для цього досить мати одне число (один жереб) r_i, але при цьому число порівнянь зростає (рис. 7.4).

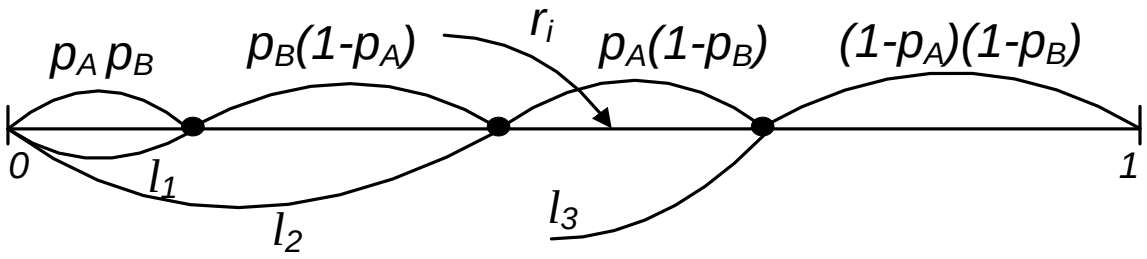


Рис. 7.4. **Моделювання складних незалежний подій (вар. 2)**

Моделювання складних залежних подій. Розглянемо тепер випадок, коли події A і B є залежними. Тоді можливі такі результати сумісних випробувань: $AB, A\overline{B}, \overline{A}B, \overline{A}\,\overline{B}$. Нехай ймовірності подій A і B складають p_A, p_B.Окрім того, задана умовна ймовірність $p_{B/A}$ події B за умови, що подія A відбулася.

Скористаємося схемою моделювання простих подій. Із сукупності $\{r_i\}$ витягуємо число r_i і перевіряємо умову

$$r_i \le p_A. \qquad (7.1)$$

Якщо нерівність справедлива, то має місце подія A. Тепер для випробування, пов'язаного з подією B, використовуємо ймовірність $p_{B/A}$. Із сукупності $\{r_i\}$ беремо чергове число r_{i+1} і перевіряємо умову $r_{i+1} \le p_{B/A}$. Якщо ця нерівність справедлива, то має місце результат AB; якщо ж немає, то результат $A\overline{B}$.

Якщо нерівність (7.1) не виконується, то має місце подія $\overline{A}$. Тому для випробування, пов'язаного з подією B, необхідно використовувати умовну ймовірність $p_{B/\overline{A}} = P(B|\overline{A})$.

Відмітимо, що A і $\overline{A}$ утворюють повну групу подій, тобто $p_A + p_{\overline{A}} = 1$. Тоді, використовуючи формулу повної ймовірності

$$P(B) = P(A)P(B|A) + P(\overline{A})P(B|\overline{A}),$$

за відомими значеннями $P(A)=p_A$, $P(B)=p_B$ і $P(B|A)=p_{B/A}$ знаходимо

$$P(B|\overline{A})=\frac{P(B)-P(A)P(B|A)}{P(\overline{A})}=\frac{P(B)-P(A)P(B|A)}{1-P(A)}.$$

Тепер перевіряємо умову $r_{i+1}\le p_{B/\overline{A}}$. Залежно від виконання або невиконання цієї нерівності за умови, що нерівність (7.1) не має місця, отримуємо результати $\overline{A}B$ або $\overline{A}\overline{B}$. Схема моделювання зображена на рис. 7.5.

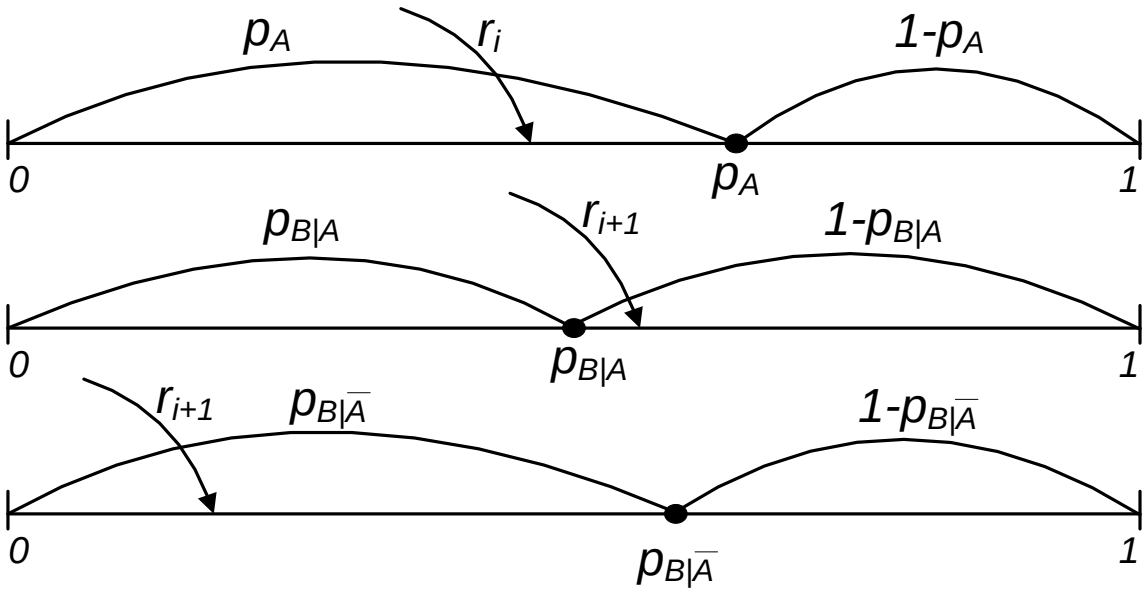


Рис. 7.5. **Моделювання складних залежних подій**

Таким чином моделюються складні залежні події.

7.2. Генератори псевдовипадкових чисел

Розглянемо основні методи моделювання неперервних і дискретних випадкових величин.

Метод зворотної функції. Згідно з визначенням функції розподілу $F_\xi(x)$ випадкової величини ξ, вона приймає значення на відрізку $[0,1]$. Тому для отримання реалізації x_i неперервної випадкової величини ξ можна спочатку згенерувати за допомогою датчика випадкових чисел деяке число r, а потім знайти x_i, при якому $r_i=F_\xi(x_i)$ або

$x_i = F_\xi^{-1}(r_i)$, де F_ξ^{-1} – функція, зворотна відносно функції розподілу $F_\xi(x)$ (рис. 7.6).

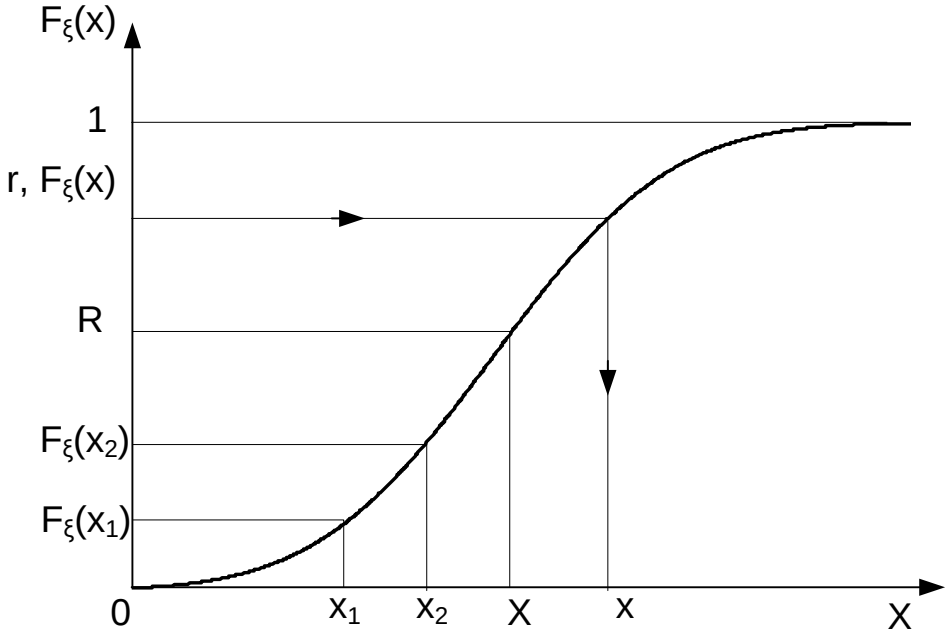


Рис. 7.6. **Графік функції розподілу** $F_\xi(x)$

Тому розглянутий метод моделювання неперервної випадкової величини носить назву ***методу зворотної функції***.

Розглянемо моделювання випадкових величин, розподілених за різними законами, з використанням методу зворотної функції, граничних теорем теорії ймовірностей і за допомогою кускової апроксимації законів розподілу.

Моделювання випадкової величини, розподіленої за нормальним законом. Нормальний розподіл є видом розподілу, що найчастіше зустрічається. З ним доводиться стикатися при аналізі виробничих похибок, контролі технологічних процесів і режимів, при аналізі і прогнозуванні різних явищ. Цей закон є граничним, до якого наближаються інші закони розподілу.

Щільність розподілу нормального закону виражається формулою

$$p(x) = \frac{1}{\sigma_\xi \sqrt{2\pi}} exp\left(-\frac{(x - m_\xi)^2}{2\sigma_\xi{}^2} \right),$$

де m_ξ – математичне очікування, σ_ξ – стандартне середньоквадратичне відхилення.

Пронормуємо випадкову величину ξ, тобто розглянемо випадкову величину $\eta = (\xi - m_\xi) / \sigma_\xi$. Математичне очікування і середньоквадратичне відхилення випадкової величини η відповідно дорівнюють $m_\eta = 0$, $\sigma_\eta = 1$.

Для того щоб розіграти значення x_i випадкової величини ξ, потрібно спочатку розіграти значення y_i випадкової величини η і від нього перейти до x_i за формулою

$$x_i = \sigma_\xi y_i + m_\xi .$$

У разі розіграшу на ЕОМ застосовують спосіб, заснований на центральній граничній теоремі теорії ймовірностей [3].

Згідно з цією теоремою, при додаванні достатньо великого числа незалежних випадкових величин, порівнянних за своїми дисперсіями, отримується випадкова величина, розподілена приблизно за нормальним законом, причому цей закон тим ближче до нормального, чим більше випадкових величин додається.

Як показали дослідження, при додаванні 12 випадкових величин з рівномірним розподілом в інтервалі $(0,1)$ отримується випадкова величина, яка з точністю, достатньою для більшості практичних задач, може вважатися нормальною.

Таким чином, процедура побудови нормально розподіленої випадкової величини ξ полягає в наступному:

1. Додають 12 випадкових величин R_i, рівномірно розподілених в інтервалі $(0,1)$, тобто складають суму

$$\mu = \sum_{i=1}^{12} R_i .$$

Випадкова величина μ має такі числові характеристики: математичне очікування

$$M(\mu) = m_\mu = \sum_{i=1}^{12} m_{R_i} = 12 \times (1/2) = 6 ,$$

дисперсію

$$D(\mu) = \sum_{i=1}^{12} D(R_i) = 12 \times (1/12) = 1$$

і середньоквадратичне відхилення $\sigma_\mu = \sqrt{D(\mu)} = 1$.

2. Нормують величину μ, тобто переходять до величини $\eta = (\mu - m_\mu) / \sigma_\mu$.

3. Від величини η переходять до величини ξ за формулою $\xi = \eta \sigma_\xi + m_\xi$.

На ЕОМ описана процедура реалізується алгоритмом:

1. За допомогою датчика випадкових чисел генерують 12 чисел r_i.

2. Обчислюють x_i за формулою

$$x_i = \sigma_\xi (\sum_{i=1}^{12} r_i - 6)_i + m_\xi .$$

Моделювання випадкової величини, розподіленої за показниковим законом. З показниковим законом розподілу доводиться часто стикатися при визначенні показників надійності систем, дослідженні систем масового обслуговування й ін.

Щільність розподілу випадкової величини ξ, розподіленої за показниковим законом, виражається формулою:

$$p_\xi(x) = \lambda e^{-\lambda x} \ (x > 0),$$

а функція розподілу – формулою

$$F_\xi(x) = 1 - e^{-\lambda x} \ (x > 0),$$

де λ – параметр показникового закону.

Нехай параметр λ показникового закону заданий і потрібно провести розіграш x_i випадкової величини ξ. Також можна скористатися методом зворотної функції:

1. За допомогою датчика випадкових чисел генерують число r_i.

2. Знаходять x_i з умови $r_i = F_\xi(x_i) = 1 - e^{-\lambda x_i}$. Тоді $-\lambda x_i = ln(1 - r_i)$, тобто

$$x_i = -\frac{ln(1 - r_i)}{\lambda}.$$

Оскільки $r_i \in (0, 1)$, то можна скористатися і формулою

$$x_i = -\frac{ln(r_i)}{\lambda}.$$

Моделювання випадкової величини, розподіленої за рівномірним законом. Рівномірний розподіл використовується при вивченні помилок округлення й ін.

Нехай випадкова величина ξ рівномірно розподілена в інтервалі (a, b). Розглянемо випадкову величину $\eta = (\xi - a)/(b - a)$. Тоді η рівномірно розподілена в інтервалі $(0, 1)$. При цьому $\xi = a + \eta(b - a)$.

Таким чином, отримання реалізації x_i випадкової величини ξ проводять в два етапи:

1. За допомогою датчика випадкових чисел генерують число r_i.

2. Знаходять x_i за формулою

$$x_i = a + r_i(b - a).$$

Моделювання випадкової величини, розподіленої за законом Пуассона. Закон Пуассона описує число подій, що відбуваються за однакові проміжки часу, за умови, що ці події відбуваються незалежно одна від одної. Розподілом Пуассона добре описуються число викликів на телефонну станцію за певний час доби, вихід негабаритів після вибуху гірських порід, число самородків при розробці родовищ золота й ін. Закон Пуассона називають ***законом появи рідкісних подій***.

Якщо випадкова величина ξ, що характеризує число настання деякої події, приймає цілочисельні значення $k = 0,1,2,\ldots$ з ймовірностями

$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

(де k – число подій, λ – параметр розподілу), то вона розподілена за законом Пуассона.

Ймовірність попадання випадкової величини ξ в заданий інтервал, наприклад в інтервал $[m,n]$, дорівнює

$$P(m \le \xi \le n) = p(m) + p(m+1) + \ldots + p(n) = e^{-\lambda} \sum_{k=m}^{n} \frac{\lambda^k}{k!}.$$

Розіграш випадкової величини, розподіленої за законом Пуассона, можна проводити відповідно до процедури розіграшу дискретної випадкової величини (див. підрозділ 7.1, моделювання повної групи подій). Відзначимо, що для розіграшу випадкової величини ξ потрібно мати датчик випадкових чисел, які мають рівномірний розподіл в інтервалі (0,1), і після розіграшу r_i перевіряти справедливість нерівності

$$l_{k-1} < r_i \le l_k,\ k = 0,1,2,\ldots$$

де $l_n = e^{-\lambda} \sum_{k=0}^{n} \frac{\lambda^k}{k!}$ (рис. 7.7).

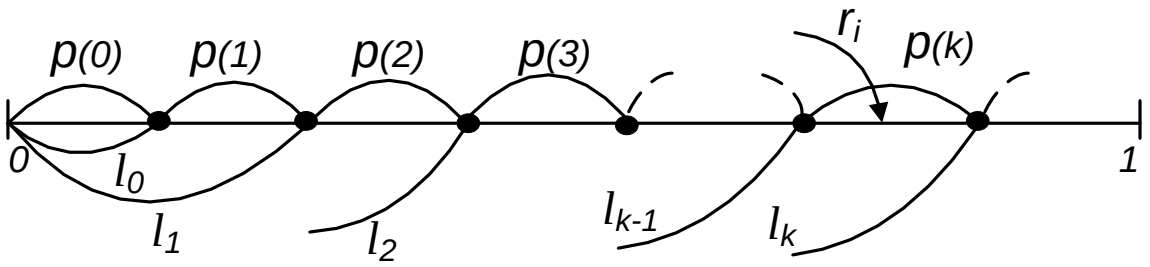


Рис. 7.7. **Розіграш випадкової величини, розподіленої за законом Пуассона**

Проте такий підхід виявляється занадто трудомістким. Можна вчинити інакше.

З курсу теорії ймовірностей відомо [3], що закон Пуассона є граничним для біноміального розподілу, який має вигляд

$$p(k,n) = C_n^k p^k (1-p)^{n-k},$$

де $p(k,n)$ – ймовірність настання події k разів при n випробуваннях, якщо ймовірність настання події в кожному випробуванні дорівнює p; C_n^k – число поєднань з n елементів по k.

Таким чином,

$$\lim_{\substack{n\to\infty \\ p\to 0 \\ a=np}} C_n^k p^k (1-p)^{n-k} = \frac{\lambda^k}{k!} e^{-\lambda}.$$

Це гранична властивість біноміального розподілу часто знаходить застосування на практиці. Допустимо, що проводиться велика кількість n незалежних дослідів, в кожному з яких подія A має дуже малу ймовірність p. Тоді для обчислення ймовірності того, що подія A відбудеться рівно k разів, можна скористатися формулою

$$P(k,n) \approx \frac{\lambda^k}{k!} e^{-\lambda},$$

де $\lambda = np$ – параметр закона Пуассона.

На підставі вищевикладеного процедура моделювання випадкової величини ξ, розподіленої за законом Пуассона, полягає у проведенні n випробувань, де ймовірність появи кожної події є малою величиною: $p \le 0.1$ (рідкісна подія):

1. Визначають число випробувань $n = \frac{\lambda}{p}$, де $p \le 0.1$. "Лічильнику числа подій" k привласнюється значення 0.

2. З сукупності випадкових чисел $\{r_i\}$ з рівномірним розподілом в інтервалі (0,1) вибирають число r_i і перевіряють умову $r_i \le p$. Якщо ця умова виконується, то k збільшується на одиницю, якщо не виконується – k не змінюється.

3. Після проведення n таких випробувань вміст лічильника числа подій k зчитується і використовується як випадкове число із законом розподілу Пуассона.

Схема проведення випробувань зображена на рис. 7.8.

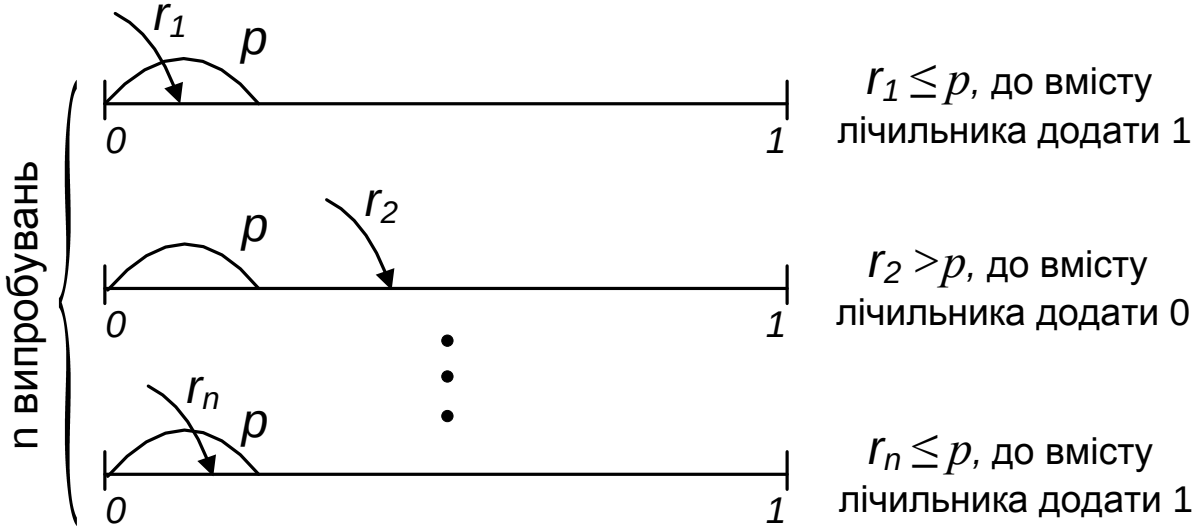


Рис. 7.8. **Розіграш випадкової величини, розподіленої за законом Пуассона**

Таким чином моделюється випадкова величина, розподілена за законом Пуассона.

7.3. Метод Монте-Карло

Під методом Монте-Карло розуміють чисельний метод розв'язання аналітичних задач.

Сутність методу Монте-Карло полягає в наступному: потрібно знайти значення a деякої величини, що вивчається (розв'язок деякої аналітичної задачі). Для цього вибирають таку випадкову величину ξ, математичне очікування якої дорівнює a, тобто $M(\xi) = a$. Практично ж чинять таким чином:

1. Проводять n випробувань, у результаті яких отримують n можливих значень x_i випадкової величини ξ (вибірку).

2. Обчислюють середнє арифметичне $\overline{x} = \sum_{i=1}^{n} x_i$ і приймають його як оцінку (наближеного значення) a^* шуканого значення a.

Оскільки метод Монте-Карло вимагає проведення великого числа випробувань, його часто називають ***методом статистичних випро-***

бувань. Теорія цього методу вказує [4] як найдоцільніше вибрати випадкову величину ξ, як знайти її можливі значення. Зокрема, розробляються способи зменшення дисперсії використовуваних випадкових величин, внаслідок чого зменшується помилка, що допускається при заміні шуканого математичного очікування a його оцінкою a^*.

Ідея методу Монте-Карло пізніше стала застосовуватися і для машинної імітації з метою дослідження характеристик процесів функціонування систем, схильних до випадкових дій, таким чином з'явився ***метод статистичного моделювання*** (див. підрозділ 3.1).

Висновки

1. Датчик випадкових чисел, рівномірно розподілених в інтервалі (0; 1), є базовим для побудови генераторів псевдовипадкових чисел з іншими законами розподілу.

2. Метод Монте-Карло – це чисельний метод розв'язання аналітичних задач, заснований на статистичних випробуваннях.

3. Метод статистичного моделювання – це метод машинної реалізації імітаційної моделі

Контрольні запитання та завдання

1. Дайте визначення випадкової величини, рівномірно розподіленої в інтервалі $[a,b]$.

2. Якими законами розподілу добре описуються різні явища і процеси, що виникають у практичних задачах.

3. Чи можна розглядати процес виборів президента країни шляхом всенародного голосування як застосування методу Монте-Карло?

Тема 8. Моделі теорії черг

8.1. Мережі Петрі

Мережі Петрі – це апарат для моделювання динамічних дискретних систем (переважно асинхронних паралельних процесів) [12]. Мережа Петрі визначається як четвірка $\langle P,T,I,O\rangle$, де P і T – скінченні множини позицій і переходів, I і O – множини вхідних і вихідних функцій. Іншими словами, мережею Петрі є дводольний орієнтований граф, в якому

позиціям відповідають вершини, що зображуються кружечками, а ***переходам*** – вершини, що зображуються потовщеними рисками; функціям I відповідають дуги, направлені від позицій до переходів, а функціям O – дуги, направлені від переходів до позицій (рис. 8.1).

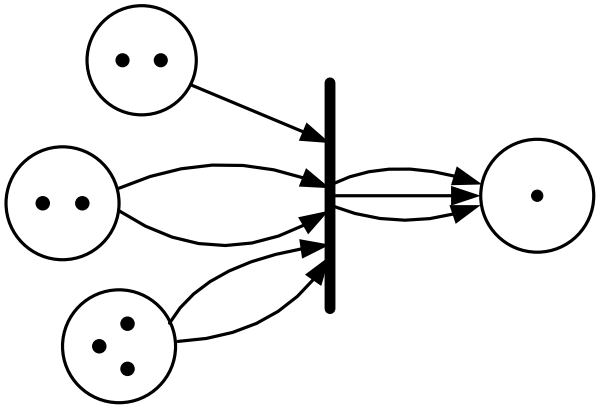


Рис. 8.1. **Фрагмент мережі Петрі**

Як і в системах масового обслуговування, в мережах Петрі вводяться об'єкти двох типів: динамічні, які зображуються ***мітками (маркерами)*** всередині позицій, і статичні, яким відповідають вершини мережі Петрі.

Розподіл маркерів за позиціями називають ***маркуванням***. Маркери можуть переміщатися в мережі. Кожну зміну маркування називають ***подією***, причому кожна подія пов'язана з певним переходом. Вважається, що події відбуваються миттєво і різночасно при виконанні деяких умов.

Кожній умові в мережі Петрі відповідає певна позиція. Здійсненню події відповідає ***спрацьовування*** (збудження або запуск) переходу, при якому маркери з вхідних позицій цього переходу переміщаються у вихідні позиції. Послідовність подій утворює модельований процес.

Правила спрацьовування переходів (див. рис. 8.1) конкретизують таким чином: перехід спрацьовує, якщо для кожної з його вхідних позицій виконується умова $N_i > K_i$, де N_i – число маркерів в i-й вхідній позиції, K_i – число дуг, що йдуть від i-ї позиції до переходу; при спрацьовуванні переходу число маркерів в i-й вхідній позиції зменшується на K_i, а в j-й вихідній позиції збільшується на M_j, де M_j – число дуг, що пов'язують перехід з j-ю позицією.

На рис. 8.1 показаний приклад розподілу маркерів за позиціями перед спрацьовуванням, це маркування записують у вигляді (2,2,3,1). Після спрацьовування переходу маркування приймає вигляд (1,0,1,4).

Можна вводити ряд додаткових правил і умов в алгоритми моделювання, отримуючи той або інший різновид мереж Петрі. Так, перш за все, корисно ввести модельний час, щоб моделювати не тільки послідовність подій, але і їх прив'язку до часу. Це здійснюється наданням переходам ваги – тривалості (затримки) спрацьовування, яку можна визначати, використовуючи алгоритм, що задається при цьому. Отриману модель називають ***часовою мережею Петрі.***

Якщо затримки є випадковими величинами, то мережу називають ***стохастичною***. У стохастичних мережах можливе введення ймовірності спрацьовування збуджених переходів. Так, на рис. 8.2 наведено фрагмент мережі Петрі, що ілюструє конфліктну ситуацію, – маркер у позиції p може запустити або перехід t_1 або перехід t_2. У стохастичній мережі передбачається імовірнісний вибір переходу, що спрацьовує, в таких ситуаціях.

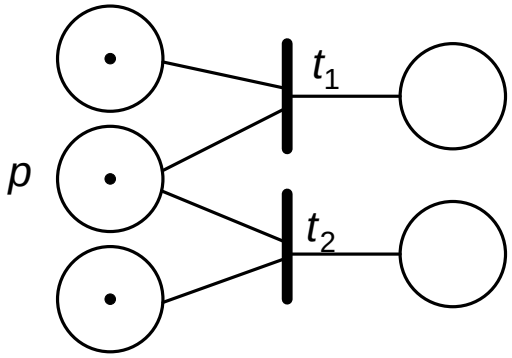


Рис. 8.2. **Конфліктна ситуація**

Якщо затримки визначаються як функції деяких аргументів, якими можуть бути кількість маркерів в яких-небудь позиціях, стани деяких переходів і тому подібне, то мережу називають ***функціональною***.

У багатьох задачах динамічні об'єкти можуть бути декількох типів, і для кожного типу потрібно вводити (чи запроваджувати) свої алгоритми поведінки в мережі. У цьому випадку кожен маркер повинен мати хоча б один параметр, що позначає тип маркера. Такий параметр зазвичай

називають кольором; колір можна використовувати як аргумент у функціональних мережах. Мережу Петрі при цьому називають ***кольоровою***.

Серед інших різновидів мереж Петрі слід згадати ***інгібіторні*** мережі, що характеризуються тим, що в них можливі заборонні (інгібіторні) дуги. Наявність маркера у вхідній позиції, пов'язаній з переходом інгібіторною дугою, означає заборона спрацьовування переходу.

Введені поняття пояснимо на таких простих прикладах.

Приклад 1. Необхідно описати з допомогою мережі Петрі функціонування системи з підприємств A, B і C. Підприємства A і B поставляють вузли $X1$ і $X2$ відповідно, а на підприємстві C відбувається збирання, у кожен складальний вузол входить один вузол $X1$ і два вузли $X2$. На рис. 8.3 підприємствам A, B і C відповідають переходи t_1, t_2 і t_3.

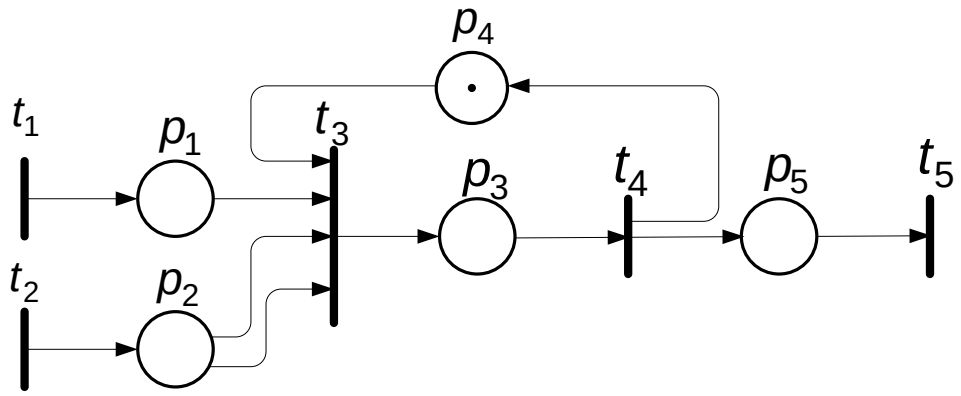


Рис. 8.3. **Мережа Петрі для приклада 1**

Спрацювання переходу t_3 відбувається тільки в тому випадку, якщо, по-перше, в позиції p_1 є мітка, а в позиції p_2 – не менше двох міток, що означає надходження від підприємств A і B відповідних комплектуючих, і, по-друге, є мітка в позиції p_4, що означає, що підприємство C закінчило збирання попереднього виробу і готове приступити до збирання наступного. Поки черговий виріб не буде зібраний, мітки в p_4 не буде, отже, запити, що надійшли у вхідні позиції p_1 і p_2, змушені чекати спрацювання переходу t_4. Переходам t_1, t_2 і t_3 постав-

лені у відповідність процедури обчислення затримок спрацювання. Затримки в перших двох переходах дорівнюють інтервалам часу між появами готових вузлів, затримка в t_3 дорівнює часу збирання виробу.

Приклад 2. Потрібно описати за допомогою мережі Петрі процеси виникнення і усунення несправностей в деякій технічній системі, що складається з M однотипних блоків; в запасі є один справний блок; відомі статистичні дані про інтенсивності виникнення відмов і тривалість таких операцій, як пошук несправностей, заміна і ремонт блоку, який відмовив. На рис. 8.4 представлена відповідна мережа Петрі. Відзначимо, що при числі міток у позиції, що дорівнює M, можна в ній не ставити M точок, а записати в позиції значення M.

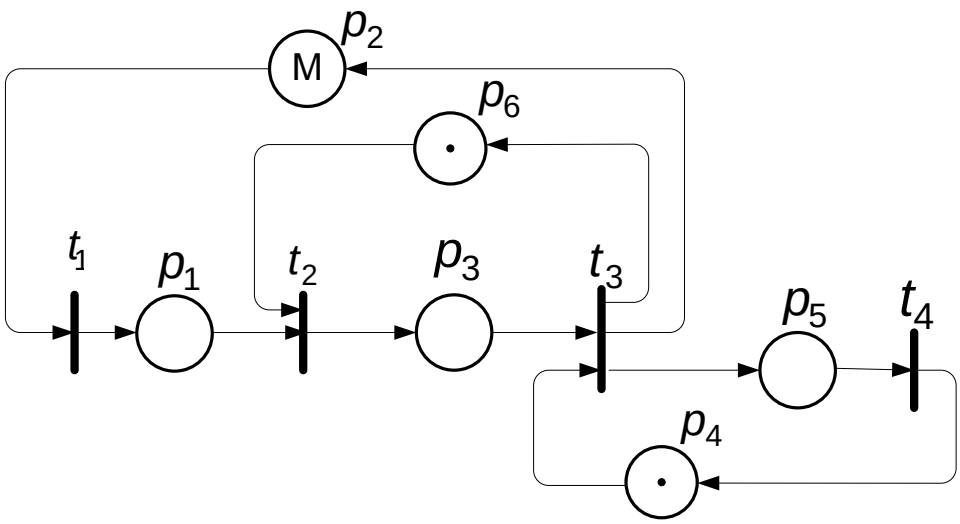


Рис. 8.4. **Мережа Петрі для приклада 2**

У нашому прикладі значення M в позиції p_2 відповідає числу наявних у системі блоків. Переходи відображають такі події: t_1 – відмова блоку, t_2 – пошук несправного блоку, t_3 – його заміна, t_4 – закінчення ремонту.

Очевидно, що при непустій позиції p_2 перехід t_1 спрацьовує, але із затримкою, що дорівнює обчисленому випадковому значенню моделюваного відрізку часу між відмовами. Після виходу маркера з t_1 він потрапляє через p_1 в t_2, якщо є мітка в позиції p_6, це означає, що обслуговуюча систему бригада фахівців вільна і може приступити до

пошуку несправності, що виникла. У переході t_2 мітка затримується на час, що дорівнює випадковому значенню тривалості пошуку несправності. Далі маркер опиняється в p_3 і якщо є запасний блок (маркер у p_4), то запускається перехід t_3, з якого маркери вийдуть в p_2, p_5 і p_6 через відрізок часу, необхідний для заміни блока. Після цього в t_4 імітується відновлення несправного блока.

Дана модель описує функціонування системи в умовах, коли відмови можуть виникати і в робочому, і в несправному станах системи. Тому не виключені ситуації, при яких більш ніж один маркер опиниться в позиції p_1.

8.2. Ланцюги Маркова

Марківські випадкові процеси названі за ім'ям видатного російського математика А. А. Маркова (1856 – 1922), який вперше почав вивчення імовірнісного зв'язку випадкових величин і створив теорію, яку можна назвати "динамікою ймовірності". Надалі основи цієї теорії стали початковою базою загальної теорії випадкових процесів, а також таких важливих прикладних наук, як теорія дифузійних процесів, теорія надійності, теорія масового обслуговування (теорія черг) і т. д. Нині теорія марківських процесів і її застосування широко застосовуються в найрізноманітніших сферах [12].

Завдяки порівняній простоті і наглядності математичного апарату, високій достовірності і точності отримуваних розв'язків, особливої уваги марківські процеси набули у фахівців, які займаються дослідженням операцій і теорією прийняття оптимальних рішень [12].

Перш ніж дати опис загальної схеми ланцюгів Маркова, розглянемо простий приклад.

Приклад 3. Припустимо, що мова йде про послідовні кидання монети при грі "в орлянку"; монета кидається в умовні моменти часу $t = 0,1,2\ldots$ і на кожному кроці гравець може виграти ± 1 з однаковою ймовірністю $\frac{1}{2}$. Таким чином, в момент t його сумарний виграш є випадковою величиною $\xi(t)$ з можливими значеннями $i = 0,\pm 1,\pm 2\ldots$. За умови, що $\xi(t) = k$, на наступному кроці виграш буде вже дорівнювати

$\xi(t+1)=k\pm1$, приймаючи вказані значення $i=k\pm1$ з однаковою ймовірністю $\frac{1}{2}$. Умовно можна сказати, що тут з відповідною ймовірністю відбувається перехід зі стану $\xi(t)=k$ у стан $\xi(t+1)=k\pm1$.

Узагальнюючи цей приклад, можна уявити собі систему із зліченим числом можливих "фазових" станів, яка з плином дискретного часу $t=0,1,2\ldots$ випадково переходить зі стану у стан. Нехай $\xi(t)$ є її стан у момент t в результаті ланцюжка випадкових переходів:

$$\xi(0)\to\xi(1)\to\xi(2)\to\ldots\to\xi(t)\to\ldots \qquad (8.1)$$

Формально позначимо всі можливі стани через ξ_i. Припустимо, що при відомому стані $\xi(t)=\xi_k$ на наступному кроці система переходить у стан $\xi(t+1)=\xi_i$ з умовною ймовірністю

$$p_{ki}=P(\xi(t+1)=\xi_i\,|\,\xi(t)=\xi_k) \qquad (8.2)$$

незалежно від її поведінки у минулому, точніше, незалежно від ланцюжка переходів (8.1) до моменту t:

$$P(\xi(t+1)=\xi_i\,|\,\xi(0)=\xi_m,\ldots,\xi(t)=\xi_k)=P(\xi(t+1)=\xi_i\,|\,\xi(t)=\xi_k)\ \forall t,k,i\,. \qquad (8.3)$$

Умова (8.3) називається ***марківською властивістю***.

Таку імовірнісну схему називають ***однорідним ланцюгом Маркова зі зліченим числом станів***. Її однорідність полягає в тому, що визначені в (8.2) ***перехідні ймовірності*** p_{ki} $\left(\sum_i p_{ki}=1,\ \forall k\right)$, не залежать від часу. При цьому матриця $P=\{p_{ki}\}$ називається ***матрицею ймовірності переходу*** за один крок і не залежить від номера кроку. Ясно, що $P=\{p_{ki}\}$ – квадратна матриця з невід'ємними елементами й одиничними сумами по рядках. Така матриця (скінченна або нескінченна) називається ***стохастичною матрицею.*** Будь-яка стохастична матриця може служити матрицею перехідних ймовірностей.

Ланцюгом Маркова першого порядку називається одна з форм марківських процесів, для якої кожний конкретний стан залежить тільки від безпосередньо попереднього (описаний вище). Ланцюгом Маркова

другого і вищих порядків називається процес, в якому поточний стан залежить від двох і більше попередніх.

Математичні моделі, що використовують ланцюги Маркова, є перехідними між детермінованими і чисто випадковими моделями.

Багато природних процесів можна вважати марківськими. Припустимо, що є серія щотижневих спостережень за рівнем води в річці, яка потрапляє в одну з трьох градацій – низький, нормальний, високий. За цими даними складена табл. 8.1 частот переходу від одного стану до іншого.

Таблиця 8.1

Від стану	До стану			Сума по рядку
	низький	нормальний	високий	
низький	12	6	0	18
нормальний	5	80	15	100
високий	0	14	16	30

Якщо поділити кожне число на суму по відповідному рядку, отримаємо ймовірність переходу від одного стану до іншого. Це, безумовно, буде не дійсне значення ймовірності, а її статистична оцінка. Ці оцінки наведені в табл. 8.2

Таблиця 8.2

Від стану	До стану			Сума по рядку
	низький	нормальний	високий	
низький	0,67	0,33	0,00	1,00
нормальний	0,05	0,80	0,15	1,00
високий	0,00	0,47	0,53	1,00

Тоді матриця ймовірностей переходу має вигляд:

$$P = \begin{pmatrix} 0.67 & 0.33 & 0.00 \\ 0.05 & 0.80 & 0.15 \\ 0.00 & 0.47 & 0.53 \end{pmatrix}.$$

Відзначимо, що якщо побудована матриця ймовірності переходу, то тим самим побудована модель випадкового процесу (системи), а значить можна зробити прогноз про майбутній стан системи.

Такий метод прогнозування (побудований на основі ланцюгів Маркова) може бути використаний для прогнозу значення множини показників, які змінюються в часі одночасно, але безпосередньо функціональні зв'язки між ними не встановлені зважаючи на відсутність інформації або крайньої складності цих зв'язків. Прикладом може служити прогноз потреб галузей народного господарства в ресурсах. При реалізації такого прогнозу встановлюється на перспективу сама структура споживання ресурсів різними галузями [2].

Так, нехай $a = \{a_i\}$ – набір n прогнозованих показників, T – інтервал спостережень за значеннями показників, $A_T = \{a_{it}\}$ – матриця розмірності $n \times T$ значень i-го показника у момент часу t $(t = \overline{1,T})$. Тоді, якщо відома матриця переходів $P = \{p_{kj}\}$ (розмірності $n \times n$), то прогноз обчислюється як

$$A_{T+1} = P\,A_T;\ \ A_{T+2} = P^2\,A_T;\ \ \ldots\ A_{T+k} = P^k\,A_T.$$

Висновки

1. Мережі Петрі і ланцюги Маркова можуть використовуватися як для моделювання систем масового обслуговування, так й інших систем.

2. Для розширення сфери застосування мереж Петрі при моделюванні систем вводяться різноманітні різновиди цих мереж.

Контрольні запитання та завдання

1. Дайте визначення мережі Петрі.

2. Як можна зробити прогноз про майбутній стан складної стохастичної системи на основі ланцюгів Маркова?

Модуль 2. Моделювання як мистецтво

Тема 9. Поняття імітаційного моделювання. Моделі систем масового обслуговування. Принципи роботи GPSS World. Елементи логіки роботи інтерпретатора

9.1. Поняття імітаційного моделювання та імітаційної моделі

Різноманітні економіко-математичні методи і моделі – потужний, теоретично і практично розроблений апарат економіко-математичного аналізу. Проте цей арсенал прийомів і математичних методів не дозволяє охопити всі задачі планування й управління, які представляють практичний інтерес і розв'язання яких може бути засноване на аналізі кількісних показників. Мова в даному випадку йде про характер задач, аналіз яких за допомогою згаданих методів або їх складніших модифікацій дійсно виявляється ефективним. Багато дуже важливих практичних задач, у тому числі й оптимізаційних, не можуть бути розв'язані з використанням економіко-математичних методів, або ж отримані з їх допомогою розв'язки виявляються недостатньо ефективними.

Серед основних об'єктивних причин цього явища можна відзначити обмежені роздільні здатності різних економіко-математичних моделей за такими основними аспектами, як рівень деталізації модельованих систем і процесів, а також забезпеченість ефективними обчислювальними методами аналітичного розв'язання. Як правило, математичні моделі, що використовуються для дослідження економічних систем, володіють досить високим рівнем абстракції, що й обумовлює їх універсальність.

З розвитком автоматизованих систем управління, розширенням сфер застосування засобів обчислювальної техніки значно різноманітніше стає коло економічних і управлінських завдань, які необхідно вирішувати. Практика вимагає постановки і вирішення все більш складних (комплексних) завдань. У цих умовах побудова адекватних моделей завдань і розробка методів їх рішення стають все більш насущними проблемами. Особливо це стосується таких завдань, в яких необхідно одночасно враховувати фактори невизначеності, динамічну взаємну

обумовленість поточних рішень, наступних подій, комплексну взаємозалежність між досліджуваними факторами. Як правило, такі практичні завдання мають велику розмірність, мають велику кількість внутрішніх взаємозв'язків, тому їх не вдається звести до відомих моделей типу математичного програмування або застосувати для їх вирішення інші традиційні методи математичного моделювання. Для вирішення таких завдань розробляється, а останніми роками отримав особливо широкий розвиток метод імітаційного моделювання на ЕОМ. Щоб з'ясувати, що ж є методом імітаційного моделювання, розглянемо деякі принципові особливості цього економіко-математичного аналізу.

За одним із визначень, **імітація – це чисельний метод проведення на ЕОМ експериментів з математичними моделями, що описують поведінку складних систем протягом тривалих періодів часу**. Таке уявлення про метод імітації в економіко-математичному аналізі засноване на використанні властивості наслідування, тобто відтворення процесів, що протікають у досліджуваній складній системі, штучними засобами за допомогою математичних моделей, що реалізовуються на ЕОМ.

Математичні моделі, які використовуються в імітації, можуть суттєво відрізнятися від традиційних.

Позначимо ці відмінності детальніше. При цьому виходитимемо з того, що дослідження реальної системи за допомогою математичних методів є реалізацією низки послідовних етапів і пов'язується, як правило, з досягненням певної мети досліджень: вивчення діючої реальної системи, аналізу гіпотетичної системи або проектування досконалішої системи.

Можна виділити декілька основних етапів моделювання:

1. Постановка задачі дослідження, вивчення модельованої системи, збирання емпіричної інформації, виділення основних проблем моделювання.

2. Формування математичної моделі, вибір структури і принципів опису моделі та її підмоделей, допустимих спрощень, вимірюваних параметрів і критеріїв оцінки якості моделі.

3. Розробка програмного забезпечення розв'язання моделі або імітаційного алгоритму, генерація чи складання машинних програм.

4. Оцінка адекватності математичної моделі і перевірка достовірності та придатності моделюючого алгоритму за ступенем погодженості і допустимості результатів контрольних експериментів з вхідними даними.

5. Планування багатоваріантних експериментів, вибір функціональних характеристик системи, що вивчається, для дослідження, визначення методів обробки результатів експериментів.

6. Робота з моделлю, проведення розрахунків і експериментів.

7. Аналіз результатів, формулювання висновків за даними моделювання і практичне використання результатів.

Поряд з аналітичними і чисельними методами розв'язування, орієнтованими на традиційні математичні моделі, для використання імітаційних моделей застосовується спосіб моделювання процесів на ЕОМ. Такий підхід передбачає використання в дослідженні специфічного різновиду математичних моделей – імітаційних моделей. Важливою особливістю імітаційного моделювання процесів на ЕОМ є те, що математичну модель, яка є вираженими в тій або іншій формі взаємозв'язками між параметрами і змінними досліджуваної системи, не обов'язково спеціально перетворювати до заздалегідь визначеного вигляду.

Для імітаційного моделювання характерне відтворення явищ, що описуються моделлю, із збереженням їх логічної структури, послідовності чергування в часі, а інколи і фізичного змісту. Таке відтворення явищ виконується за допомогою спеціальних моделюючих установок (апаратурне моделювання) або засобів обчислювальної техніки. В останньому випадку забезпечується високий ступінь подібності між математичними (кількісними) характеристиками реальних процесів і їх модельними відображеннями в ЕОМ. Причому, на противагу аналітичному і чисельному методам зміст операцій, що виконуються при роботі з імітаційною моделлю, майже не залежить від того, які величини вибрані як шукані.

Методи імітаційного моделювання мають деякі принципові відмінності і в частині способів використання математичних моделей, які зближують імітаційне моделювання з методами фізичного моделювання і натурних експериментів на реальних системах. Основну відмінність можна представити наочніше, якщо взяти до уваги призначення мате-

матичних моделей, що реалізовуються способами аналітичного і чисельного дослідження процесів, й імітаційних моделей.

Аналітичні моделі призначені, головним чином, для отримання рішення, що визначає в готовому вигляді значення шуканих змінних на основі закладеної в моделі інформації про досліджувану систему.

На відміну від них імітаційні моделі призначені для отримання інформації про модельовану систему і вироблення в подальшому відповідних оцінок, застосовних для формування рішень. Вироблення рішень у цьому випадку проводиться, як правило, поза імітаційною моделлю.

Імітаційні моделі будуються передусім для того, щоб на підставі інформації про досліджувану систему і процеси, що протікають в ній, узагальнити та деталізувати наявні дані до такого ступеня, при якому вони могли б стати придатними для вироблення рішень у рамках деяких завдань дослідження системи. Причому, як правило, методи вирішення цих завдань безпосередньо в імітаційну модель не включаються.

Найважливішою особливістю методу імітаційного моделювання є те, що імітаційні моделі можуть застосовуватися для опису і дослідження досить складних процесів практично на межі можливої формалізації. Вони використовуються і тоді, коли частина процесів досліджуваної системи взагалі не формалізується. Останнє характерне для таких процесів, які реалізуються в досліджуваній системі за участю людини, що ухвалює рішення.

Застосування методів імітації для дослідження системи, формування імітаційної моделі ґрунтуються на використанні максимального обсягу доступної інформації про систему як таку, яка може бути представлена в деякому формальному вигляді за допомогою математичних співвідношень і залежностей, так і таку, яка може бути виражена у вигляді функції розподілу ймовірності випадкової величини й інших прийомів. У цьому випадку в аналіз залучається і та частина даних про систему, яка не може бути отримана через те, що у розпорядженні дослідника може не виявитися відомостей про поведінку системи у всіх принципово можливих режимах її функціонування або в допустимих межах зміни параметрів процесів, що вивчаються.

Для моделювання досліджуваного процесу на ЕОМ необхідно, щоб математична модель цього процесу була представлена у формі спеціального моделюючого алгоритму. Відповідно до цього алгоритму в

ЕОМ буде вироблена інформація, що описує елементарні явища досліджуваного процесу з урахуванням їх зв'язків і взаємних впливів. Природно, що явища досліджуваного процесу і явища, які відбуваються в ЕОМ, що реалізовує алгоритм моделювання, за своїм фізичним змістом суттєво відрізняються. Але вони мають бути близькі з погляду складу і характеру інформації, що описує поведінку реальної системи, та інформації, оброблюваної ЕОМ у процесі імітації.

Реалізація імітаційного алгоритму в ЕОМ є модельним відтворенням кількісних характеристик елементарних явищ, характерних для досліджуваного реального процесу. У даному випадку немає не обхідності не тільки перетворювати початкову математичну модель досліджуваної системи у форму, що допускає аналітичне або чисельне розв'язання, але і підбирати для її вирішення деякий дуже далекий за своєю структурою від самої моделі аналітичний або чисельний метод.

Розробка програмного забезпечення вирішення моделі або імітаційного алгоритму.

Побудова імітаційного алгоритму в корені відрізняється від програмування методів аналітичного і чисельного вирішення математичних моделей на ЕОМ. Процедури алгоритмізації в цьому випадку найбільше визначаються змістом математичної моделі, а не ґрунтуються на чіткому виділенні сукупності шуканих величин.

Якщо для вирішення звичайних математичних моделей можуть бути використані різні пакети прикладних програм, що реалізовують на ЕОМ ті або інші алгоритми аналітичного і чисельного вирішення завдань, то програмування імітаційних моделей виявляється значно складнішим. Це обумовлюється особливостями, властивими методу імітаційного моделювання. Перш за все, основною діяльністю дослідника при роботі з імітаційною моделлю є спостереження, реєстрація і вимірювання контрольованих параметрів процесів, що вивчаються. Тому в моделюючий алгоритм повинні бути включені не тільки процедури, які відтворюють кількісні характеристики досліджуваних реальних процесів і їх взаємозв'язки, але і спеціальні програми, що дозволяють накопичувати дані вимірювань у процесі імітації.

Крім того, моделюючий алгоритм повинен дозволяти у процесі експерименту імітувати дію на процес випадкових і неконтрольованих параметрів. Усе це забезпечує відносну незалежність імітаційної моделі від прийнятого розділення даних, що фігурують у ній, на початкові умови,

параметри системи і шукані величини. Така незалежність є принциповою особливістю методу імітаційного моделювання, оскільки при дослідженні процесів звичайними аналітичними або чисельними методами зміна сукупності шуканих величин, як правило, вимагає інших форм представлення математичної моделі або ж використання різних алгоритмів для вирішення загалом однієї й тієї ж задачі.

Суттєво і те, що імітаційне моделювання виявляється зручним апаратом для дослідження випадкових процесів – це одне з основних застосувань даного методу.

Для реалізації на ЕОМ імітаційний алгоритм повинен бути представлений у вигляді набору машинних програм, що реалізовують процедури імітації, збирання даних за спостереженнями і вимірами, введення та виведення початкових даних і результатів експериментів тощо. Програмування імітаційних алгоритмів здійснюється з використанням універсальних мов програмування або ж за допомогою спеціалізованих проблемно-орієнтованих мов імітаційного моделювання.

9.2. Основні поняття теорії масового обслуговування

Теорія масового обслуговування (ТМО) – галузь прикладної математики, що використовує методи теорії випадкових процесів. Поняття "система масового обслуговування" пов'язане з явищем очікування. Стимулом до розвитку теорії масового обслуговування послужили спроби передбачити потреби, які випадково змінюються, за результатами спостережень і на основі цього організувати обслуговування, що характеризується прийнятним часом очікування. Теорія масового обслуговування дозволяє розкрити природу черг, що забезпечує можливість кращого управління процесом.

Теорія масового обслуговування, як і взагалі моделювання, безпосередньо не пов'язана з оптимізацією. Вона скоріш намагається розробити, вивчити і порівняти різні ситуації, що характеризуються створенням черги, і, таким чином, побічно досягти наближеної оптимізації.

Першою у теорії масового обслуговування була робота Іохансеса "Час очікування і число викликів" (1907 р.). Однак вона пройшла непоміченою. Початок цьому науковому напряму поклала робота А. К. Ерланга "Теорія ймовірності і телефонні переговори" (1909 р.) та інші його роботи, в яких він вивчав проблеми теорії скупчення. Практичну спрямо-

ваність в 20 – 30-х роках мали роботи й інших авторів, наприклад, Мошина, Фрайя. Узагальнення методів і розробка загальної теорії масового обслуговування були початі дещо пізніше. Істотний внесок до її створення і розвитку зробили радянські вчені Хинчин А. Я., Гнеденко Б. В., Колмогоров А. Н., Бусленко Н. П., Вентцель Е. С., Севостьянов Б. А. та інші, а також закордонні вчені К. Пальм, Д. Кендалл, Ф. Поллачек, Сааті Т. Л. та ін.

У теорії масового обслуговування розглядаються задачі планування й управління, пов'язані з визначенням ефективності виконання низки робіт (послуг) над певними об'єктами (обслуговуваними одиницями). Модель конкретної системи масового обслуговування має практичне значення і допускає можливість розв'язання в тому випадку, якщо обслуговувані одиниці (об'єкти, заявки) надходять в обслуговуючу систему в масовому порядку, через випадкові проміжки часу, час виконання робіт (обслуговування об'єктів) також носить випадковий характер. Так, у магазині касир обслуговує велику кількість покупців, їх надходження в магазин, а також час обслуговування наперед неможливо точно визначити.

Типовими виробничими задачами, вирішуваними методами теорії масового обслуговування, є задачі організації телефонного і телеграфного зв'язку, обслуговування в торгівлі, ремонт і наладка обладнання, відвантаження готової продукції на багатьох підприємствах та ін. Теорія масового обслуговування може застосовуватися при розробці оптимальної структури управління народним господарством і окремими його ланками, а також в економіці, де об'єктом виступає потреба в продукції певного виду, а обслуговуючою системою є заводи, що випускають цю продукцію. Теорію масового обслуговування успішно можна використати і при розробці інформаційних систем, проектуванні компьтерних мереж тощо.

Як правило, системи масового обслуговування є дуже складними. Методами теорії масового обслуговування проводиться аналіз окремих систем, потім на його основі вирішується проблема оптимізації параметрів системи.

Теорія масового обслуговування вивчає процеси, в яких, з одного боку, постійно виникають запити на виконання яких-небудь послуг, а, з іншого, відбувається задоволення цих запитів, тобто виконання послуг.

Та частина процесу, в якій виникають запити, є обслуговуваною системою, а та частина, яка приймає запити і задовольняє їх, називається обслуговуючою системою. Сукупність обслуговуваної і обслуговуючої систем називається системою масового обслуговування (СМО).

Основними поняттями теорії масового обслуговування є:

вимога – кожен окремий запит на виконання якої-небудь роботи або послуг;

джерело вимог – частина обслуговуваної системи, яка у будь-який момент часу може надіслати лише одну вимогу;

обслуговування – задоволення запиту, що надійшов в обслуговуючу систему, на виконання послуг;

обслуговуючий апарат – частина обслуговуючої системи, яка здатна в будь-який заданий момент часу задовольняти лише одну вимогу (обслуговуюча система – це сукупність однорідних обслуговуючих апаратів, причому під однорідністю розуміється здатність задовольняти однакові вимоги);

потік вимог – послідовність появи вимог у часі;

час обслуговування – час, протягом якого задовольняється запит, тобто період від початку обслуговування (а не від моменту надходження вимоги в систему) і до його завершення.

9.3. Системи масового обслуговування, їх класифікація та основні характеристики

Із системами масового обслуговування (СМО) ми зустрічаємось повсякчас. Кожному з нас доводилось чекати обслуговування в черзі (у магазині, на автозаправці, в бібліотеці, кав'ярні тощо). Аналогічні ситуації виникають, коли треба скористатися телефонним зв'язком або виконати свою програму на комп'ютері. Будь-яке виробництво теж можна представити як послідовність систем обслуговування. До типових систем обслуговування належать також ремонтні і медичні служби, транспортні системи, аеропорти, вокзали тощо.

Особливого значення набули такі системи у процесах інформатики. Це передусім комп'ютерні системи, мережі передавання інформації, операційні системи, бази і банки даних. Системи обслуговування відіграють значну роль у повсякденному житті. Досвід моделювання різних

типів дискретних систем свідчить про те, що приблизно 80 % цих моделей ґрунтуються на СМО.

Систему масового обслуговування загалом можна представити як сукупність послідовно пов'язаних між собою вхідних потоків вимог на обслуговування (потоків замовлень), черг, каналів обслуговування і потоків обслужених замовлень. Будь-який пристрій, який безпосередньо обслуговує замовлення, називають *каналом обслуговування*.

Системи масового обслуговування можна класифікувати, базуючись на наявності тієї чи іншої ознаки.

1. **За характером надходження замовлень у систему**: системи з регулярним і випадковим потоками замовлень. Якщо кількість замовлень, які надходять у систему за одиницю часу (інтенсивність потоку), стала або є заданою функцією часу, то маємо систему з регулярним потоком замовлень, в іншому разі – з випадковим. Випадковий потік замовлень може бути стаціонарним або нестаціонарним. Якщо параметри потоку замовлень не залежать від розташування інтервалу часу, який розглядають, на осі часу, то маємо стаціонарний потік замовлень, в противному разі – нестаціонарний. Наприклад, якщо кількість покупців, які приходять до магазину, не залежить від часу доби, то потік замовлень (покупців) – стаціонарний.

2. **За кількістю замовлень, які надходять за одиницю часу**: системи з ординарним і неординарним потоками замовлень. Якщо ймовірність надходження двох або більше замовлень в один момент часу дорівнює нулю або настільки мала, що нею можна знехтувати, то маємо систему з ординарним потоком замовлень. Наприклад, потік літаків, які прибувають на злітну смугу аеродрому, можна вважати ординарним, оскільки ймовірність надходження двох і більше літаків до каналу обслуговування (злітної смуги) в один і той самий момент часу дуже мала.

3. **За зв'язком між замовленнями**: системи без післядії від замовлень, які надійшли, і з післядією. Якщо ймовірність надходження замовлень у систему в деякий момент часу не залежить від того, скільки вимог уже надійшло до системи, тобто не залежить від передісторії процесу, який вивчають, то маємо задачу без післядії, у противному разі – з післядією. Прикладом задачі з післядією може слугувати потік студентів на складання заліку викладачу.

3. **За характером поведінки замовлень у системі**: системи з відмовами, з обмеженим очікуванням і з очікуванням без обмеження:

– якщо нове замовлення, яке прибуло на обслуговування, застає усі канали обслуговування уже зайнятими і покидає систему, то маємо систему з відмовами. Замовлення може покинути систему і тоді, коли черга досягла певних розмірів. Якщо ракета супротивника з'являється в час, коли всі протиракетні пристрої обслуговують інші ракети, то вона без проблем залишає зону обслуговування;

– якщо нове замовлення, яке прибуло на обслуговування, застає усі канали обслуговування зайнятими і стає у чергу, але перебуває у ній обмежений час і, не дочекавшись обслуговування, покидає систему, то маємо систему з обмеженим очікуванням. Прикладом такого "нетерплячого" замовлення може бути самоскид із цементним розчином. Якщо час очікування великий, то щоб запобігти затвердненню розчину, він може бути розвантажений в іншому місці;

– якщо нове замовлення, яке прибуло на обслуговування, заставши усі канали обслуговування зайнятими, змушене очікувати своєї черги до того часу, поки не буде обслужене, то маємо систему з очікуванням без обмеження. Приклад: літак, який перебуває на аеродромі до того часу, поки не звільниться злітна смуга.

5. **За способом вибору замовлень на обслуговування**: з пріоритетом, за часом надходження, випадково, останнього обслуговують першим. Іноді в такому випадку кажуть про дисципліну обслуговування:

– якщо система масового обслуговування охоплює кілька категорій замовлень і з певних міркувань необхідно дотримуватись різного підходу до їхнього відбору, то маємо систему з пріоритетом. Зокрема, під час надходження виробів на будмайданчик, перш за все монтують ті, які необхідні у цей момент;

– якщо канал, який звільнився, обслуговує замовлення, яке раніше за інших надійшло до системи, то маємо систему з обслуговуванням замовлень за часом надходження. Це найпоширеніший клас систем. Наприклад, покупця, який підійшов до продавця першим, обслуговують раніше за інших. Цей спосіб вибору замовлень на обслуговування застосовують там, де внаслідок технічних, технологічних або організаційних умов замовлення не можуть випереджати одне одного;

– якщо замовлення з черги надходять до каналу обслуговування у випадковому порядку, то маємо систему з випадковим вибором замовлень на обслуговування. Приклад: вибір слюсарем-сантехніком одного з декількох замовлень на усунення несправностей, які надійшли від мешканців. Вибір тут, зазвичай, визначають місцезнаходженням самого слюсаря: він надасть перевагу замовленню мешканця, який перебуває від нього найближче, якщо інші чинники не визначають вибору;

– останнього обслуговують першим. Цей спосіб вибору вимог на обслуговування використовують у тих випадках, коли зручніше й економніше брати на обслуговування замовлення, яке найпізніше надійшло до системи. Зокрема, якщо будівельні вироби складені один на одному, то зручніше спочатку брати виріб, який надійшов останнім.

6. **За характером обслуговування замовлень**: на системи з детермінованим і випадковим часом обслуговування. Якщо інтервал часу між моментами надходження замовлення до каналу обслуговування і моментом виходу замовлення з цього каналу є сталим, то йдеться про систему з детермінованим часом обслуговування, в іншому разі – з випадковим.

7. **За кількістю каналів обслуговування**: на одноканальні і багатоканальні системи. Наприклад, для зведення будинку можна використати один будівельний кран (один канал обслуговування) або декілька (багато каналів) для обслуговування виробів, які прибувають на будівництво.

8. **За кількістю етапів обслуговування**: на однофазні і багатофазні системи. Якщо канали обслуговування розташовані послідовно, і вони неоднорідні, оскільки виконують різні операції обслуговування, то йдеться про багатофазну систему масового обслуговування. Прикладом такої системи може бути обслуговування автомобілів на станції технікного обслуговування (миття, діагностування тощо).

9. **За однорідністю замовлень, які надходять на обслуговування**: на системи з однорідними і неоднорідними потоками замовлень. Наприклад, якщо для розвантаження прибувають фургони однакової вантажомісткості, то такі замовлення називають однорідними, якщо різної – то неоднорідними.

10. **За обмеженістю потоку замовлень**: на замкнені і розімкнені системи. Якщо потік замовлень обмежений і замовлення, які покинули

систему, через деякий час до неї повертаються, то маємо замкнену систему, у противному разі – розімкнену. Прикладом замкненої ситеми може слугувати бригада робітників, які налагоджують станки у ткацькому цеху.

З метою скорочення запису для позначення будь-якої однофазної СМО використовують систему кодування A/B/C/D/E, де на місці латинської літери ставлять відповідні характеристики системи:

A – закон розподілу інтервалів між надходженнями замовлень. Найчастіше використовують такі закони розподілу: показниковий (M), ерлангівський (E), гіперекспоненціальний (H), ґамма-розподіл (Г), детермінований (D). Для позначення довільного характеру розподілу використовують символ G;

B – закон розподілу часу обслуговування в каналах СМО. Тут використовують такі самі позначення, як і для розподілу інтервалів між надходженнями замовлень;

C – кількість каналів обслуговування. Тут використовують такі позначення: для одноканальних систем записують 1, для багатоканальних l (кількість каналів);

D – кількість місць у черзі. Якщо кількість місць у черзі необмежена, то це позначення можна не використовувати. Для скінченної кількості місць у черзі в загальному випадку приймають позначення r або n (кількість місць);

E – дисципліна обслуговування. Найчастіше використовують такі варіанти системи обслуговування: FIFO (першим прийшов – першим вийшов), LIFO (останнім прийшов – першим вийшов), RANDOM (випадковий порядок обслуговування). За дисципліни FIFO це позначення можна не використовувати.

Приклади позначень:

M/M/1 – СМО з одним каналом обслуговування, нескінченною чергою, показниковими законами розподілу інтервалів часу між надходьженнями замовлень і часу обслуговування та дисципліною обслуговування FIFO;

E/H/l/r/LIFO – СМО з кількома каналами обслуговування, скінченною чергою, ерлангівським законом розподілу інтервалів часу між над-

ходженнями замовлень, гіперекспоненціальним розподілом часу обслуговування та дисципліною обслуговування LIFO;

G/G/l – СМО з кількома каналами обслуговування, нескінченною чергою, довільними закономи розподілу інтервалів часу між надходженнями замовлень і часу обслуговування та дисципліною обслуговування FIFO.

Вивчення або задання потоку замовлень, механізму (кількості каналів, тривалості обслуговування тощо) та дисципліни обслуговування дає підстави для побудови моделі системи.

Суть досліджень реальних процесів за допомогою теорії масового обслуговування – кількісний опис потоку вимог, що надходять у систему, і часу обслуговування, визначення через їх характеристики показників якості функціонування системи як при існуючому варіанті її організації, так і при інших можливих варіантах й отримання висновків про поліпшення роботи системи масового обслуговування шляхом зміни її організації. Проте для аналітичного вирішення завдання недостатньо мати лише кількісний опис потоку вимог і часу обслуговування. Для правильного вибору виведених в аналітичній теорії масового обслуговування розрахункових формул, за якими визначаються показники якості функціонування системи, необхідно знати також тип системи обслуговування.

Показники якості функціонування систем масового обслуговування залежать не тільки від величини параметрів потоку вимог і часу обслуговування, але і від різних ознак, – форми системи і її внутрішньої організації, порядку обслуговування й інших, – залежно від яких всі задачі масового обслуговування діляться на декілька типів. Для кожного з них є свій набір показників якості функціонування системи і свої формули їх розрахунку залежно від величини параметрів потоку вимог і часу обслуговування.

На практиці найчастіше зустрічаються системи, в яких потік вимог близький до простого, а час обслуговування є показниковим. Для таких систем характерним є очікування, скінченне число обслуговуючих апаратів, обмежений потік вимог і неврегульоване обслуговування.

У теорії масового обслуговування розглядаються також системи з урахуванням можливості виходу з ладу обслуговуючих пристроїв. При розрахунку цих систем застосовуються методи теорій надійності і заміни обладнання. На основі теорії ймовірності і математичної статистики

теорія надійності дозволяє встановити закономірності виникнення поломок обладнання, розробити методи контролю надійності виробів, оптимізувати їх надійність з економічної точки зору.

Завданням теорії масового обслуговування є відшукання залежностей величини, що характеризує якість функціонування обслуговуючої системи або її ефективність, від способів організації системи в цілому. Під ефективністю обслуговуючої системи розуміється характеристика рівня виконання цією системою її функцій. Показники ефективності визначаються трьома групами факторів: характеристиками якості і надійності системи обслуговування, економічними показниками й особивостями функціонування системи.

Під якістю функціонування системи масового обслуговування розуміють не власне якість виконання тієї або іншої роботи, запит на яку надійшов, а ступінь задоволення потреби в обслуговуванні. При цьому поняття "якість" функціонування системи масового обслуговування у кожному окремому випадку матиме свій конкретний зміст і буде виражатися різними кількісними показниками, наприклад, величина черги на обслуговування; середній час обслуговування, очікування обслуговування або перебування вимоги в обслуговуючій системі; час простою обслуговуючих апаратів; упевненість, що всі вимоги, які надійшли в систему, будуть обслужені і т. д.

Кількісні показники якості функціонування систем масового обслуговування залежать від виду системи, а також від величин, що характеризують основні її параметри. Тому **метою теорії масового обслуговування є розробка математичних методів для відшукання основних показників процесів масового обслуговування, що характеризують якість функціонування системи масового обслуговування при різних варіантах її організації**.

При вирішенні завдань масового обслуговування виявляються функціональні залежності між показниками якості функціонування системи масового обслуговування і характеристиками потоку вимог, часу обслуговування, способу організації обслуговування. Завдання вважається вирішеним, якщо вдається вибрати для даного типу системи масового обслуговування кількісні показники якості її функціонування і виразити їх через параметри, що характеризують вхідний потік вимог та час їх обслуговування.

Таким чином, **предметом дослідження теорії масового обслуговування** є кількісна сторона процесів масового обслуговування.

Показниками якості й ефективності функціонування системи, що найчастіше зустрічаються, є:

1) ймовірність того, що обслуговуванням зайнято n пристроїв;

2) ймовірність втрати об'єкта, що дорівнює ймовірності того, що всі пристрої зайняті обслуговуванням;

3) середня кількість зайнятих і вільних пристроїв;

4) коефіцієнти простою і завантаження пристроїв;

5) середній час перебування об'єкта в черзі і системі;

6) середня величина черги;

7) ймовірність того, що час перебування об'єкта в черзі триватиме не більше певної величини;

8) ймовірність того, що кількість елементів у черзі більше деякого числа m;

9) кількість необслужених (втрачених) об'єктів;

10) витрати і втрати при функціонуванні системи.

Оцінка ефективності дозволяє оптимізувати системи масового обслуговування, тобто оптимально організувати систему обслуговуючих пристроїв (їх кількість і склад) або потік об'єктів, що надходить. Для цього при оцінці ефективності застосовуються вартісні показники:

вартість обслуговування об'єкта;

вартість втрат, пов'язаних з очікуванням обслуговування в одиницю часу;

вартість збитків, пов'язаних з втратою об'єктів;

вартість експлуатації обслуговуючих пристроїв в одиницю часу;

вартість одиниці часу простою пристрою та ін.

9.4. Принципи роботи GPSS World

Система GPSS World призначена для імітаційного моделювання систем із дискретними та неперервними процесами. У ній мова моделювання GPSS покращена вбудованою мовою програмування низького рівня PLUS. Мова GPSS ґрунтується на припущенні, що модель складної системи можна зобразити за допомогою сукупності елементів і логічних правил їхньої взаємодії у процесі функціонування системи. Також припускається, що у процесі моделювання системи можна виокремити

невеликий набір абстрактних елементів, які називають **об’єктами**. Набір логічних правил обмежений і його можна описати невеликою кількістю стандартних операцій. Комплекс програм, які описують функціонування об’єктів і виконують логічні операції, є основою для створення програмної моделі системи цього класу.

У складі системи GPSS World є також спеціальна **програма-планувальник**, яка виконує такі функції:

забезпечення просування за заданими розробником маршрутами динамічних об’єктів, які називають транзактами;

планування подій, які відбуваються в моделі, шляхом реєстрації часу появи кожної події і виконання їх у наростаючій часовій послідовності;

реєстрація статистичної інформації про функціонування моделі;

просування модельного часу у процесі моделювання системи.

Для забезпечення правильної послідовності обробки подій у часі призначений системний годинник, який зберігає значення абсолютного модельного часу. Усі проміжки часу описують додатними числами.

Об'єкти в GPSS. Модель в GPSS будується з окремих елементів, що називаються об'єктами. Об’єкти в GPSS класифікують на сім категорій і 15 типів:

1. Динамічна – транзакти.
2. Операційна – блоки.
3. Апаратна (статична) – одноканальні пристрої, багатоканальні пристрої, логічні ключі.
4. Обчислювальна – змінні, функції, генератори випадкових чисел.
5. Статистична – черги, таблиці.
6. Запам’ятовуюча – комірки, матриці комірок.
7. Групуюча – числові групи, групи транзактів, списки.

Стан моделі в будь-який момент часу визначається сукупністю станів усіх об'єктів, що складають модель; зміна станів моделі припускає зміну стану хоча б одного об'єкта. Основна особливість GPSS як системи моделювання полягає в тому, що стан моделі змінюється лише тоді, коли динамічний об'єкт – транзакт – проходить через операційний блок. Саме транзакт, рухаючись по моделі, є ініціатором зміни станів пристроїв, статистичних об'єктів й інших транзактів.

Змістовне значення транзактів визначає розробник моделі. Він встановлює аналогію між транзактами і реальними елементами системи,

яку моделює. Таку аналогію ніколи не вказує планувальник GPSS. Вона присутня лише в пам'яті розробника моделі.

Динамічні об'єкти (транзакти) створюються у певних точках моделі, просуваються планувальником через блоки, а потім знищуються. З кожним транзактом може бути пов'язана певна кількість параметрів, які містять необхідну інформацію про нього. Параметри нумерують або їм дають імена. Номери параметрів та імена використовують для посилань на значення, присвоєні параметрам. Транзакти можуть мати різні пріоритети. Пріоритет визначає перевагу, яку отримує транзакт, коли він та інші транзакти претендують на один і той самий ресурс.

Операційні об'єкти (блоки) задають логіку функціонування моделі системи і визначають шлях руху транзактів між об'єктами апаратної категорії. Модель системи можна зобразити сукупністю блоків, об'єднаних відповідно до логіки роботи реальної системи у так звану блок-схему. Блок-схему моделі можна зобразити графічно, наочно показуючи взаємодію блоків у процесі моделювання.

У блоках можуть відбуватися події чотирьох головних типів:

1) створення або знищення транзактів;

2) зміна числового атрибута об'єкта;

3) затримка транзакта на певний період часу;

4) зміна маршруту руху транзакта в моделі.

Версія GPSS, реалізована в системі GPSS World, містить 53 типи блоків. За призначенням блоки класифікують на кілька груп.

1. Блоки, які здійснюють модифікацію атрибутів транзактів:

а) генерування і знищення транзактів (GENERATE, SPLIT, TERMINATE, ASSEMBLE);

б) затримка у часі (ADVANCE);

в) синхронізація руху двох (MATCH) і кількох (GATHER) транзактів;

г) зміна параметрів транзактів (ASSIGN, INDEX, MARK, PLUS);

д) зміна пріоритету транзакта (PRIORITY).

2. Блоки, які змінюють послідовність руху транзактів (DISPLACE, TRANSFER, LOOP, TEST, GATE).

3. Блоки, пов'язані з групуючою категорією (ADOPT, ALTER, EXAMINE, JOIN, REMOVE, SCAN).

4. Блоки, які описують об'єкти апаратної категорії:

а) одноканальні пристрої (SEIZE, RELEASE, PREEMT, RETURN, FUNAVAIL, FAVAIL);

б) багатоканальні пристрої (ENTER, LEAVE, SAVAIL, SUNAVAIL);

в) ключі (логічні перемикачі) (LOGIC).

5. Блоки, які зберігають необхідні значення для того, щоб використати їх пізніше (SAVEVALUE, MSAVEVALUE).

6. Блоки, які забезпечують отримання статистичних результатів:

а) черги (QUEUE, DEPART);

б) таблиці (TABULATE).

7. Блоки для організації списку користувача (LINK, UNLINK).

8. Блоки для організації введення-виведення:

а) відкриття/закриття файла (OPEN/CLOSE);

б) зчитування/запис у файл (READ/WRITE);

г) встановлення позиції поточного рядка (SEEK).

9. Спеціальні блоки (BUFFER, COUNT, EXECUTE, INTEGRATION, SELECT, TRACE, UNTRACE).

Об’єкти апаратної категорії – це абстрактні елементи, на які можна здійснити декомпозицію (поділ) устаткування реальної системи. Діючи на ці об’єкти, транзакти можуть змінювати їхній стан і впливати на рух інших транзактів. До об’єктів цього типу належать одноканальні пристрої, багатоканальні пристрої та логічні ключі.

Одноканальні пристрої (пристрої) – це устаткування, яке в будь-який момент часу може займати лише один транзакт. Наприклад, один канал зв’язку, каса продажу квитків з одним віконцем, один телефон-автомат тощо.

Багатоканальні пристрої (БКП) призначені для імітації устаткування, яке здійснює паралельне обслуговування, тобто може одночасно обслуговувати кілька транзактів. БКП можна використовувати як аналог, наприклад, багатоканальної ремонтної майстерні, квиткових кас з кількома віконцями, телефонного переговорного пункту з кількома кабінами тощо.

Інколи події, які відбувалися в системі раніше, можуть заблокувати, змінити рух транзактів і настання найближчих подій. Наприклад, якщо один канал зв’язку вийшов з ладу, то всі наступні замовлення на передавання повідомлень потрібно перенаправити на інші канали зв’язку. Для моделювання таких ситуацій використовують *логічні ключі*. Транзакт може встановлювати ці ключі в позиції "увімкнено" або "вимкнено". Пізніше стан ключів можуть перевіряти інші транзакти для визначення шляху їхнього руху.

Обчислювальна категорія призначена для опису таких ситуацій у процесі моделювання, коли зв'язки між компонентами системи найпростіше і компактно виражаються у вигляді математичних (аналітичних і логічних) співвідношень. З цією метою об'єктами обчислювальної категорії введені арифметичні та булеві змінні і функції.

Змінні – це складні вирази, які містять константи, стандартні числові атрибути (СЧА), бібліотечні арифметичні функції, арифметичні та логічні операції.

Вирази можна застосовувати у змінних і в операторах GPSS. У першому випадку вирази визначають командами GPSS, у другому їх визначають як частину мови PLUS.

Кожному об'єкту відповідають атрибути, які описують його стан у певний момент часу. До них є доступ для використання протягом всього процесу моделювання; їх *називають стандартними (системними) числовими атрибутами* (СЧА). Усього в GPSS World є понад 50 СЧА.

Булеві змінні дають змогу користувачеві перевіряти в одному блоці GPSS одночасно кілька умов, враховуючи стан або значення цих умов та їхніх атрибутів, тобто в даному блоці здійснюється звертання до булевої змінної, вираз якої містить у собі перевірку кількох умов. Булеві змінні можуть мати вигляд комбінації стандартних числових атрибутів, пов'язаних між собою за допомогою булевих операторів, охоплюючи й інші змінні. Булеві змінні визначають так само, як і арифметичні, але замість арифметичних операцій перевіряються різноманітні логічні умови.

За допомогою *функцій* користувач може здійснювати обчислення неперервних або дискретних функціональних залежностей між аргументом функції (незалежна величина) і залежним значенням функції. Усі функції в GPSS задають табличним способом за допомогою команд опису функцій. Як і змінні, функції не зв'язані з певними блоками.

Крім бібліотечних арифметичних функцій, GPSS World має 24 вмонтованих *генератори випадкових чисел*.

Об'єкти запам'ятовуючої категорії забезпечують звертання до збережених значень. Комірки збережених величин і матриці комірок збережених величин використовують для зберігання деякої числової інформації. Наприклад, значення, занесене в комірку, може бути коефіцієнтом використання пристрою у певний момент часу. Будь-який активний транзакт здійснює запис інформації у ці об'єкти. Пізніше запи-

сану в ці об’єкти інформацію зчитує будь-який транзакт. Матриці мають до шести вимірів.

До **статистичних об’єктів** належать черги і таблиці. У будь-якій системі рух потоку транзактів може бути зупинений через недоступність пристроїв. Наприклад, потрібний пристрій може бути вже зайнятим або БКП, до якого треба увійти, вже заповнений. У цьому випадку затримані транзакти можуть бути поставлені в чергу – ще один тип об’єктів GPSS. Облік цих черг – одна з головних функцій планувальника. Планувальник автоматично накопичує певну статистику стосовно пристроїв і черг. Крім того, користувач може збирати додаткову статистичну інформацію, вказавши спеціальні точки в моделі.

Для полегшення табулювання статистичної інформації в GPSS передбачено спеціальний об’єкт – таблицю. Таблиці використовують для отримання вибіркових розподілів деяких випадкових величин. Таблицю складають з частотних класів (діапазонів значень), куди записують число, яке дорівнює кількості потраплянь конкретного числового атрибута до кожного, того чи іншого, частотного класу. Для кожної таблиці обчислюють також математичне очікування і середнє квадратичне відхилення. Ця статистика є стандартною для всіх таблиць.

До **групуючої категорії** належать три типи об’єктів: числова група, група транзактів і списки.

У деяких моделях транзакти є об’єктами, що характеризуються спільними СЧА, які зазвичай зображають як параметри транзакта і змінюються під час проходження транзактів через блок ASSIGN. Однак це обмежує доступ до атрибутів, а інколи доводиться одночасно змінювати деякі атрибути всіх транзактів цієї множини. Об’єкти такого типу дають змогу користувачеві звертатися до атрибутів транзактів даної групи. Вони працюють в одному з режимів: у режимі транзакта або числовому. Режим визначають першим звертанням до певної групи. У режимі транзакта величини, які надходять до групи, зображають номерами за чергою транзактів даної групи. У другому режимі величини, які надходять, зображають групою числових значень.

Списки. У GPSS-моделі кожен транзакт розглядається як елемент одного або декількох списків. У GPSS існують списки: поточних подій; майбутніх подій; користувача; переривання; парності. Є тільки один список поточних й один список майбутніх подій. У загальному випадку

може існувати більш ніж один список користувача, список переривання й список парності.

Список поточних подій (СПП). У цьому списку перебувають транзакти, що відповідають подіям, час настання яких менше або дорівнює поточному часу. Транзакти розміщаються в порядку зменшення їхнього пріоритету; у середині класу транзактів з однаковим пріоритетом вони розміщаються в порядку надходження їх у список. Кожний транзакт може перебувати або в активному стані (переглядатися планувальником в поточний момент часу), або у стані затримки. Якщо транзакт перебуває в активному стані, то планувальник, здійснюючи перегляд, робить спробу просунути цей транзакт у наступний блок. Якщо рух транзакта блокується якимсь пристроєм, то вхід у наступні блоки виконати неможливо, транзакт переводиться у стан затримки і поміщається у відповідний список затримки.

Списки затримки – це списки транзактів, що чекають зміни стану пристрою. Як тільки пристрій звільняється, всі транзакти списку затримки для даного пристрою переводяться в активний стан.

Список майбутніх подій (СМП). Містить транзакти, що відповідають подіям, час настання яких перевищує поточний час. Транзакти розміщені строго в порядку збільшення часу початку руху. Пріоритети транзактів не впливають на порядок розміщення в цьому списку.

Списки користувача. Містять тимчасово неактивні транзакти, виведені користувачем зі списку поточних подій. Списки користувача використовують для організації черг з усіма дисциплінами обслуговування, крім дисципліни "першим прийшов – першим обслуговований".

Списки переривань. Містять транзакти, обслуговування яких перервано на одному або більше пристроях. Транзакти у списках переривань розміщуються в порядку надходження. Усі транзакти, що зайняли пристрої, і перервані іншими транзактами, поміщаються в ці списки доти, поки не будуть зняті всі умови переривання. Якщо ці умови не знімаються, то транзакт може залишатися в списку переривань необмежено довго.

Списки парності. Містять транзакти, що очікують у блоках ASSEMBLE або GATHER надходження заданого числа транзактів з тієї ж самої родини й перебувають у блоках MATCH, чекаючи транзакт з тієї ж

самої родини на сполучений блок MATCH. Ці списки є піднаборами набору транзактів системи.

Пристрій має:

список відкладених переривань – список транзактів, які є у черзі до пристрою за пріоритетом;

список переривань – список транзактів, обслуговування яких цим пристроєм було перерване;

список затримки – список транзактів, які перебувають у черзі до пристрою в порядку пріоритету;

список повторних спроб – список транзактів, які очікують зміни стану пристрою.

Багатоканальний пристрій має:

список затримки – список транзактів у порядку пріоритету, які очікують на можливість зайняти звільнені канали БКП;

список повторних спроб – список транзактів, що очікують зміни стану БКП.

До списку повторних спроб належать транзакти, для яких не виконані умови входження в наступний блок. Ці умови перевіряються під час спроби входження транзакта в блоки GATE, TEST, TRANSFER ALL і TRANSFER BOTH. Транзакти зі списку повторних спроб чекають на зміну СЧА. Після цієї зміни транзакт активується і перевіряються умови входження його в наступний блок. Якщо умови виконуються, транзакт входить в наступний блок й автоматично вилучається зі списку повторних спроб.

Модельний час. Кожне просування транзакта в моделі є подією, яка повинна відбутися в певний момент модельного часу. Для того щоб підтримувати правильну послідовність подій у часі, планувальник має таймер модельного часу, який автоматично корегується відповідно до логіки, визначеної моделлю. Таймер GPSS має такі особливості:

реєструються лише додатні дійсні значення часу;

одиницю модельного часу (масштаб) визначає розробник моделі, який задає всі часові інтервали в одних і тих самих одиницях;

планувальник не аналізує стан моделі в кожний наступний момент модельного часу (який відрізняється від поточного на одиницю модельного часу), а просуває таймер до моменту часу, коли повинна відбутися найближча наступна подія.

9.5. Елементи логіки роботи інтерпретатора

Координація процесів у моделі забезпечується інтерпретатором GPSS – планувальником транзактів. Головним завданням, яке виконує планувальник (інтерпретатор), є визначення того, який транзакт потрібно вибрати наступним для просування в моделі, коли попередній транзакт припинив свій рух. З цією метою планувальник розглядає кожен транзакт як елемент деякого списку. Планувальник GPSS вводить транзакти залежно від умов у моделі до того чи іншого списку, проглядає списки, вибирає наступний транзакт для опрацювання, корегує таймер модельного часу після опрацювання всіх транзактів у списку поточних подій.

В основному логіка роботи планувальника стає зрозумілою з розгляду механізму відстежування шляху транзактів, що рухаються в моделі. Планувальник розглядає кожен транзакт як елемент одного або декількох списків. Кожен транзакт може бути представлений як ланка в списку. Списки є відкритими, а не замкнутими, отже, вони мають два кінці, початковий і кінцевий. Як елемент списку транзакт займає певне положення відносно початку списку. Положення транзакта у списку тісно пов'язане з тим, як скоро транзакт повинен знову повернутися в модель для продовження руху. Послідовність обробки, у свою чергу, тісно пов'язана з часом виникнення подій при виконанні моделювання.

Події ділять на три фази: зміна модельного часу, перегляд списку поточних подій, просування транзактів.

Фаза 1. Зміна значення модельного часу

Припустимо, що планувальник обслужив всі транзакти, які мають відбутися в даний момент часу. Далі планувальник повинен збільшити значення модельного часу так, щоб воно дорівнювало часу настання найближчої події в майбутньому. Оскільки у СМП транзакти розміщені в порядку зростання часу надходження, нове значення модельного часу дорівнює часу надходження першої події з СМП (кожному транзакту зі списку подій відповідає свій час настання події). Таким чином, планувальник встановлює значення модельного часу рівним часу настання події для першого транзакта з СМП. У цей момент фактично і змінюється модельний час. Проте планувальник повинен перевірити, чи немає ще інших подій, які мають відбутися в той же самий момент модельного часу. Для цього він перевіряє час настання події для наступного транзакта з СМП. Якщо час настання події для наступного транзакта

дорівнює новому значенню поточного часу, то цей транзакт також переводиться на обробку в СПП, а планувальник переходить до аналізу часу настання події для наступного транзакта з СМП. Цей процес триває доти, поки у СМП не з'явиться транзакт, для якого час настання події перевищує поточне значення модельного часу, або поки в СМП не залишиться жодного транзакта. Закінчення перегляду транзактів з СМП показує, що всі транзакти, які необхідно обробити в даний момент модельного часу, переведені у СПП.

Алгоритм фази 1 наведено на рис. 9.1.

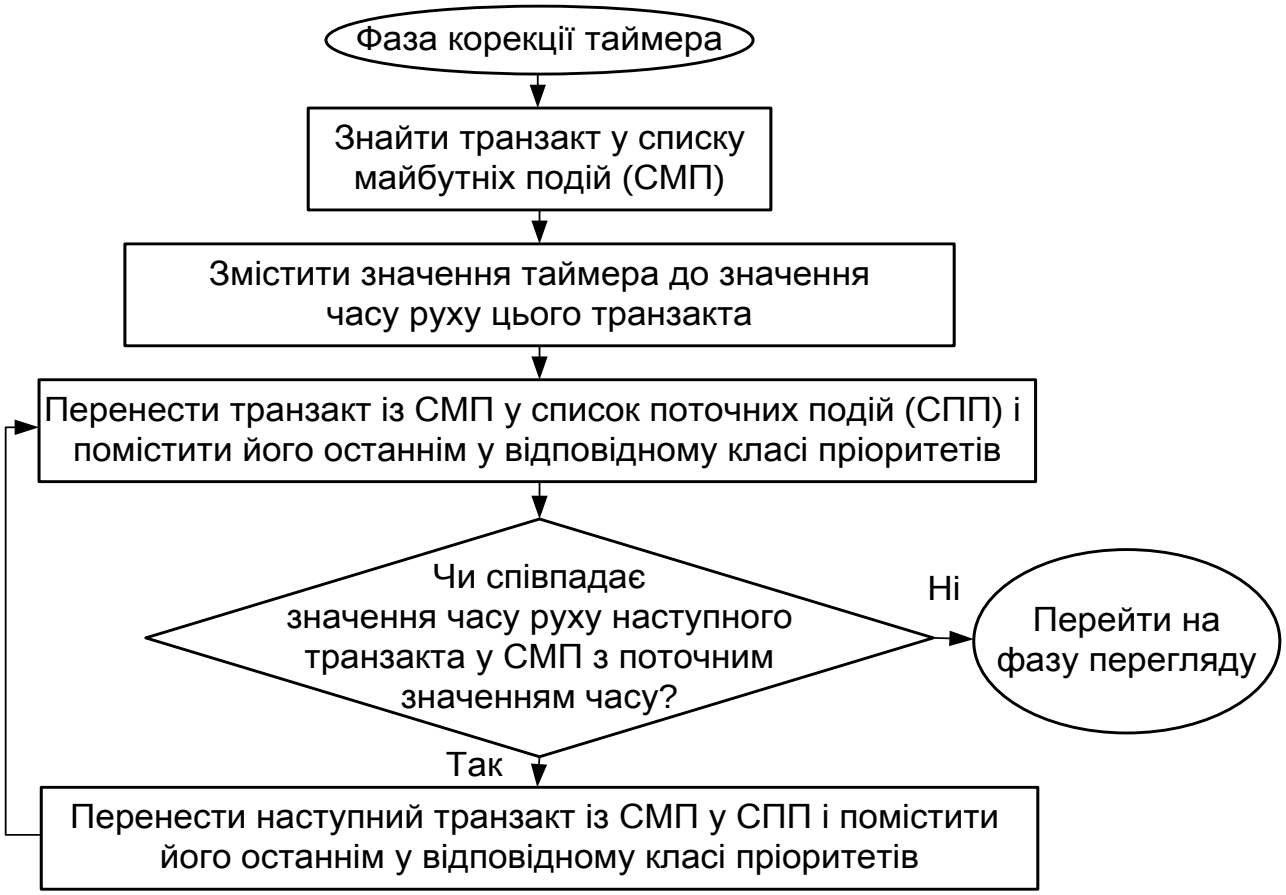


Рис. 9.1. **Фаза 1. Зміна значення модельного часу**

Фаза 2. Перегляд списку поточних подій

Вибравши всі потрібні транзакти з СМП, планувальник починає переглядати СПП. При цьому він намагається просунути всі транзакти, що перебувають в активному стані в СПП, і відповідним чином змінити стан моделі. Перш за все встановлюється в нуль "прапорець зміни стану", який показує, змінився чи не змінився стан якого-небудь пристрою в даний момент часу. Оскільки на початку обробки транзактів, що належать до нового значення модельного часу, ніякі зміни в стані пристрою відбутися ще не могли, прапорець встановлюється в нуль.

Далі в результаті перегляду вибирають перші транзакти з СПП і перевіряють, чи встановлений в нуль індикатор перегляду даного транзакта. Індикатор перегляду показує, в якому стані перебуває даний транзакт: в активному чи у стані затримки. Якщо транзакт перебуває в активному стані, то планувальник робить спробу просунути його на стільки блоків, на скільки можливо, поки не зустрінуться блоки, що містять блокуючі умови або явно задану затримку в часі. Якщо ж транзакт перебуває у стані затримки через те, що рух транзакта заблокований станом пристрою (тобто транзакт перебуває у списку затримки) або якщо планувальник закінчив просування транзакта, обслуговується наступний транзакт з СПП.

Планувальник продовжує працювати таким чином до тих пір, поки не перегляне весь список. До цього час всі транзакти, що належать до даного моменту модельного часу, вже оброблені, і настає фаза зміни значення модельного часу.

Алгоритм фази 2 наведено на рис. 9.2.

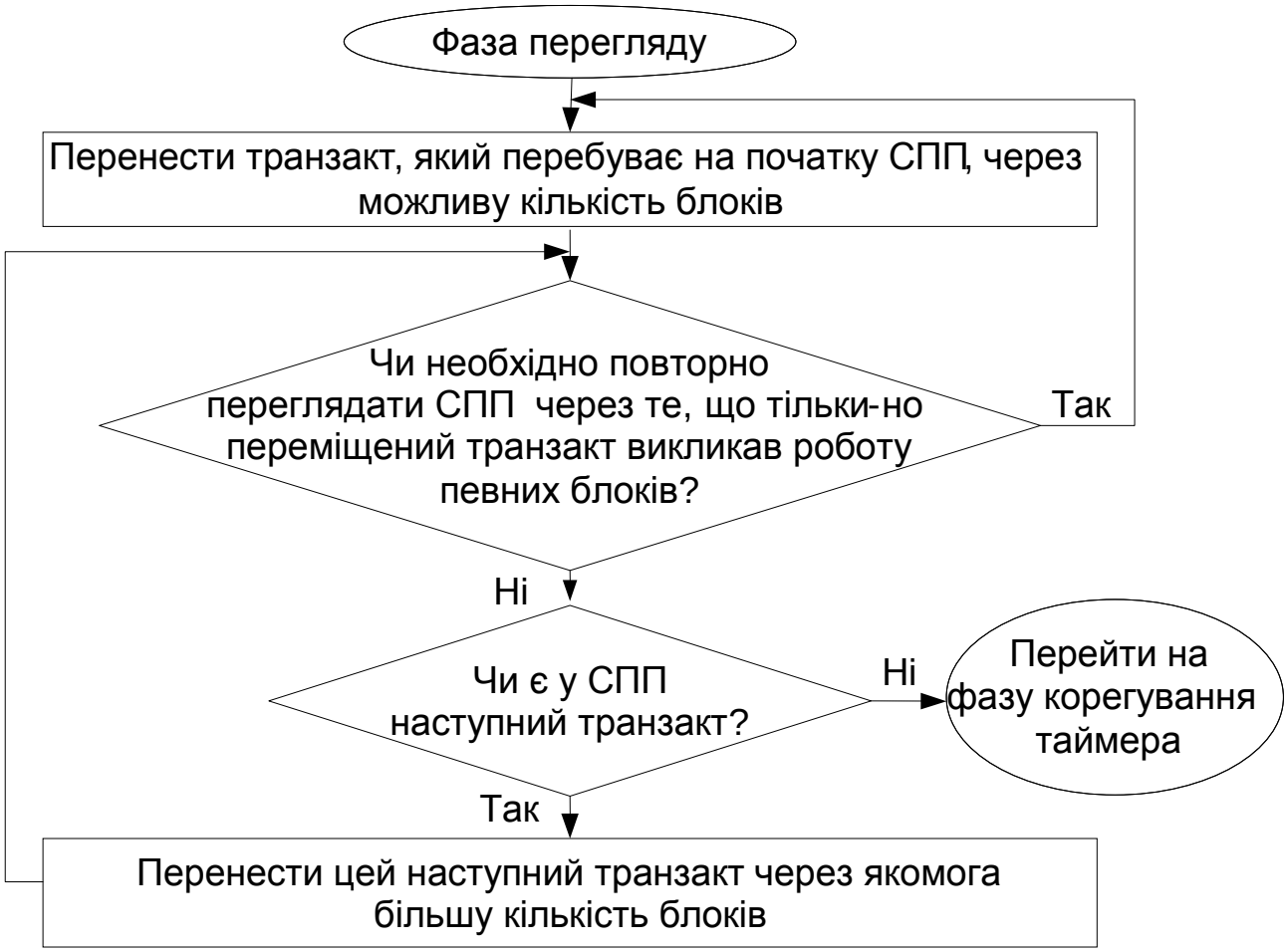


Рис. 9.2.**Фаза 2. Перегляд списку поточних подій**

Фаза 3. Просування транзактів

Як наголошувалося, якщо транзакт у СПП перебуває в активному стані, планувальник робить спробу просунути його. Якщо транзакт може рухатися, планувальник його рухає і виконує операції, відповідні блоку, який проходить транзакт. Після цього планувальник визначає, чи належить блок, до якого увійшов транзакт, до блоку типу BUFFER. Якщо належить, то планувальник негайно припиняє обробку транзакта і починає перегляд СПП.

Якщо блок, до якого увійшов транзакт, не є блоком типу BUFFER, планувальник перевіряє, чи не змінився стан пристрою у процесі виконання попереднього блоку. Якщо змінився, то планувальник встановлює в одиницю "прапорець зміни стану", а в нуль – індикатори перегляду для всіх транзактів, що перебувають у списку затримки, пов'язаному з відповідним пристроєм.

Якщо у блоці, що виконується, явно не задана затримка транзакта, планувальник відразу ж робить спробу просунути транзакт через наступний блок. Цей процес продовжується до тих пір, поки транзакт не отримує відмову при спробі увійти в блок, не зустрічає блок ADVANCE з явно заданою затримкою в часі або не знищується блоком TERMINATE чи ASSEMBLE.

Якщо рух транзакта блокується через стан пристрою, то індикатор перегляду для цього транзакта встановлюється в одиницю. Це означає, що транзакт перейшов у стан затримки. Транзакт поміщається у відповідний список затримки, і потім планувальник визначає, чи змінився стан пристрою у процесі обробки цього транзакта. Якщо рух транзакта припинений через задану для нього затримку в часі, він поміщається в СМП, і потім інтерпретатор також перевіряє, чи змінився стан пристрою.

Якщо під час обслуговування поточної активної події стан пристрою змінився, то перегляд починається спочатку, і знову обслуговуються всі транзакти СПП, що перебувають в активному стані. Якщо стан пристрою не змінився, то планувальник знову звертається до СПП і перевіряє, чи залишилися в ній транзакти, що вимагають обробки.

Описана послідовність виконується для кожного нового моменту модельного часу доти, поки лічильник завершення моделювання не буде дорівнювати нулю. Після виконання цієї умови моделювання припиняється.

Висновки

1. Імітаційні моделі описують об’єкт дослідження деякою мовою, імітуючи елементарні явища, з яких складається функціонування системи, зі збереженням їхньої логічної структури, послідовності протікання у часі, особливостей і складу інформації про стан процесу.

2. Застосування методів імітації для дослідження системи, формування імітаційної моделі ґрунтуються на використанні максимального обсягу доступної інформації про систему.

3. Метою теорії масового обслуговування є розробка математичних методів для відшукання основних показників процесів масового обслуговування, що характеризують якість функціонування системи масового обслуговування при різних варіантах її організації.

4. Система GPSS World призначена для імітаційного моделювання систем з дискретними та неперервними процесами. Проблемною областю GPSS є системи масового обслуговування.

Контрольні запитання та завдання

1. Дайте визначення імітаційного моделювання. Назвіть основні принципи побудови імітаційних моделей.
2. Дайте визначення системи масового обслуговування. Назвіть її основні характеристики.
3. Класифікація СМО.
4. Показники якості функціонування СМО.
5. Мови моделювання дискретних процесів.
6. Особливості загальноцільової системи моделювання GPSS World.
7. Основні принципи роботи GPSS World.
8. Дайте визначення модельного часу.
9. Назвіть об'єкти в GPSS World. Дайте їх характеристику.
10. Поняття списків транзактів.
11. Елементи логіки роботи планувальника: зміна значення модельного часу, перегляд списку поточних подій, просування транзактів.

Тема 10. Блоки, що забезпечують побудову моделі типу "СМО з одним пристроєм". Забезпечення пріоритетного обслуговування

10.1. Блоки, що забезпечують побудову моделі типу "СМО з одним пристроєм"

Потоки, які існують у реальних системах, у моделях імітують за допомогою транзактів. Тому, передусім, розглянемо введення транзактів у модель і виведення з неї.

Введення транзактів у модель

GENERATE (генерувати) – це блок, через який транзакти входять у модель. На кількість різних блоків GENERATE в одній моделі обмежень не існує. Інтервал часу між послідовними появами транзактів блоку GENERATE називають *інтервалом надходження*.

Коли транзакт входить у модель через блок GENERATE, планувальник планує час надходження наступного транзакта шляхом розіграшу випадкового числа відповідно до розподілу інтервалів часу надходження з подальшим додаванням розіграного значення до поточного значення таймера. При досягненні цього значення часу наступний транзакт вводиться в модель через блок GENERATE і т. д.

Інформація про ймовірнісний розподіл інтервалу надходження задається за допомогою операндів A і B. Усі можливі розподіли інтервалів часу надходження в GPSS класифікують на рівномірно розподілені й усі інші розподіли (детальніше про розподіли у підрозділі 12.3).

Блок GENERATE має такий формат:

GENERATE A,B,C,D,E

Операнд A задає середній інтервал часу між послідовними надходженнями транзактів у модель.

Операнд B задає половину поля допуску для значення операнда A. Операнд B називають модифікатором, він змінює значення інтервалу генерування транзактів порівняно з інтервалом, зазначеним операндом A. Є два типи модифікаторів: модифікатор-інтервал і модифікатор-функція.

За допомогою модифікатора-інтервала задають рівномірний закон розподілу часу між створенням транзактів.

Операнди A і B можуть бути іменем, додатним числом, виразом у дужках або безпосередньо СЧА.

Різниця A–B значень, заданих операндами A і B, дає нижню межу інтервалу, а сума A+B – його верхню межу. Після генерування чергового транзакта вибирається число з отриманого інтервалу. Це буде значенням часу, через який наступний транзакт вийде з блоку GENERATE.

Наприклад, блок

GENERATE 15,5

створює транзакти через випадкові інтервали часу, рівномірно розподілені на відрізку [10; 20].

Операнди A і B задавати не обов’язково, їх значення за замовчуванням – "нуль".

Модифікатор-функцію використовують, якщо закон розподілу інтервалу надходження відрізняється від рівномірного. У цьому випадку у полі B записують посилання на функцію (її СЧА), яка описує потрібний закон розподілу. Під дією модифікатора-функції значення операнда A множиться на значення функції, заданої операндом B.

Задаючи операнди A і B слід пам’ятати, що за будь-якого способу обчислення інтервалу часу значення операнда B не повинне перевищувати значення операнда A.

Операнд C задає зміщення інтервалів (початкову затримку), тобто момент часу появи першого транзакта у блоці GENERATE. Після появи першого транзакта всі наступні появи транзактів виникають згідно з розподілом інтервалів часу, заданих операндами A і B. Операнд C можна використовувати як для прискорення, так і для уповільнення приходу першого транзакта або для призначення приходу в потрібний момент часу. Початкова затримка може бути меншою, такою самою або більшою порівняно із середнім часом, заданим операндом A. Коли операнд C не використовують, то інтервали генерування визначають операнди A і B (вони не впливають на затримку). Операнд C може бути іменем, додатним числом, виразом у дужках або безпосередньо СЧА.

Операнд D задає граничне значення загальної кількості транзактів, які можуть увійти в модель через цей блок GENERATE протягом часу моделювання. Коли ця кількість досягнута, даний блок GENERATE

перестає бути активним. Якщо не визначене граничне значення (операнд D не використовується), то блок GENERATE залишається активним до завершення моделювання.

Операнд E встановлює клас пріоритету кожного з транзактів, які потрапляють у модель через блок GENERATE. Для задавання пріоритетів з метою підвищення ефективності роботи GPSS World рекомендується використовувати послідовність цілих чисел 0, 1, 2, 3,... Чим більше число, тим вищий пріоритет. Якщо операнд E не використовується, то за замовчуванням пріоритет створених даним блоком GENERATE транзактів дорівнює нулю.

Операнди D та E можуть бути іменем, додатним числом, виразом у дужках або безпосередньо СЧА, але вони можуть мати значення лише цілих додатних і цілих чисел відповідно.

У будь-якому блоці GENERATE має бути заданий хоча б один з операндів A і D. ***Не можна використовувати на полі операнда параметри транзактів.***

У початковий момент часу в кожному блоці GENERATE здійснюється підготовка до виходу одного транзакта. На цій стадії модель ще цілковито не ініціалізована до виконання. З цієї причини всі СЧА, використані у блоці GENERATE, повинні бути вже визначені раніше. Тому в моделі перед блоком GENERATE мають стояти команди визначення EQU, INITIAL, FUNCTION, VARIABLE, FVARIABLE.

Приклади запису блоків GENERATE:

```
GENERATE    10
GENERATE    25,15
GENERATE    5,,30
GENERATE    6,3,20,100
GENERATE    15,,20,,2
```

У першому прикладі середній час появи транзактів у моделі складає 10 одиниць модельного часу. У другому прикладі час появи транзактів у моделі рівномірно розподілений на інтервалі [10; 40]. У третьому прикладі перший транзакт з’явиться в моделі в момент часу, рівний 30, надалі транзакти з’являтимуться в моделі кожні 5 одиниць модельного часу. У четвертому прикладі блоком GENERATE буде згенеровано 100 транзактів, причому перший з’явиться в моделі в момент часу рівний 20, а інтервали появи наступних транзактів рівномірно розподілені на інтервалі [3; 9]. У п’ятому прикладі починаючи з 20-ї одиниці модельного

часу кожні 15 одиниць часу в моделі з'являтимуться транзакти з пріоритетом, рівним 2.

Вилучення транзактів з моделі та завершення моделювання

Транзакти вилучаються з моделі, потрапляючи до блока TERMINATE (завершити). Блок TERMINATE завжди дає дозвіл на входження всім транзактам, які намагаються це зробити. У моделі може бути будь-яка кількість блоків TERMINATE. Блок має такий формат запису:

TERMINATE A

Значенням операнда А є кількість одиниць, на яку блок TERMINATE зменшує значення ***лічильника завершень*** під час надходження транзакта до цього блока. Операнд А може бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. За замовчуванням значення операнда А дорівнює нулю. У цьому випадку транзакт знищується, а значення лічильника завершень не змінюється.

Лічильник завершень є коміркою пам'яті, яка зберігає додатне ціле число, записане на початку моделювання командою START. Поточне значення лічильника завершень доступне для користувача через системний СЧА TG1.

У процесі моделювання транзакти потрапляють до блока TERMINATE і віднімають число, яке дорівнює значенню операнда А, від лічильника завершень. У момент, коли значення лічильника досягає нуля, моделювання завершується.

Необхідно звернути увагу на той факт, що хоча в моделі може бути багато блоків TERMINATE, лічильник завершень тільки один.

Моделювання починається, коли зустрічається команда START (почати). Для визначення початкового значення лічильника завершення використовується операнд А команди START.

Розглянемо приклад, в якому блок TERMINATE і команда START використовуються для управління процесом моделювання.

Припустимо, що одиницею часу обрана 1 хвилина і необхідно промоделювати поведінку системи протягом 8 годин (480 хв.), потім моделювання має закінчитися. Це можна зробити включивши в модель сегмент з двох блоків:

```
GENERATE    480
TERMINATE   1
```

Цей сегмент з двох блоків забезпечує завершення моделювання у момент модельного часу, рівний 480.

У всіх інших блоках TERMINATE в моделі мається на увазі використання операнда А за замовчуванням (нуль). Це означає, що завершення моделювання, визначене лічильником завершень, не залежатиме від інших блоків TERMINATE.

У команді START як операнд А повинна бути використана одиниця:

```
START     1
```

Одиничне значення операнда А команди START викликає запис у лічильник завершень одиниці як початкове значення. У процесі моделювання завершення рухів транзактів, які відбуваються час від часу в інших блоках TERMINATE, не впливають на лічильник завершень. Отже, у момент модельного часу, що дорівнює 480, транзакт потрапить у блок GENERATE. Він відразу ж перейде в наступний блок TERMINATE. Оскільки операнд цього блоку є одиниця, то з лічильника завершень віднімається одиниця. Це зменшує значення лічильника від одиниці до нуля. У результаті інтерпретатор припиняє моделювання.

Тепер припустимо, що для виконання дій, описаних у попередньому прикладі, розробник вставляє в модель сегмент з двох блоків:

```
GENERATE     1
TERMINATE    1
```

(Інший вид сегменту з двох блоків, що спричиняє завершення моделювання у момент часу, рівний 480).

У всіх інших блоках TERMINATE моделі мається на увазі використання операнда А за умовчанням; як операнд А команди START використовується число 480:

```
START 480
```

Відмітимо, що у блоці GENERATE як інтервал часу задана одиниця. Інакше кажучи, транзакти з'являються тут в моменти часу 1, 2, 3, 4, ..., 478, 479 і 480. Кожен з цих транзактів потрапляєв в наступний блок TERMINATE, де з лічильника завершень віднімається одиниця при видаленні транзакта. Коли у блок TERMINATE увійде 480-й транзакт, значення лічильника завершень вже буде дорівнювати одиниці. 480-й транзакт викликає зменшення його від одиниці до нуля і моделювання завершується.

Метод завершення моделювання, представлений у першому прикладі, є переважнішим, хоча в логічному сенсі вони рівнозначні, оскільки

останній приклад вимагає 480 виконань обробки блоків GENERATE і TERMINATE. Оскільки обробка кожного блоку вимагає часу EOM, то останній приклад принаймні в 480 разів дорожче попереднього. При другому способі завершення моделювання інтерпретатор GPSS пройде всі фіксовані моменти часу, оскільки таймер повинен приймати значення 1, 2, 3, 4 ..., 478, 479 і 480 через те, що транзакти входять у модель через блок GENERATE. Це значною мірою збільшує число приростів значень часу до таймера у процесі моделювання, що сприяє ще більшому збільшенню часу моделювання.

Користувач задає час моделювання, зазначаючи в команді START значення лічильника завершень.

Команду START використовують для запуску процесу моделювання. Вона має такий формат запису:

START A,B,C,D

Операнди:

A – задає значення лічильника завершень, яке визначає момент завершення прогону моделі, і може бути лише цілим додатним числом;

B – операнд виведення статистики. Він може бути NP (немає виведення даних) або відсутнім. У випадку задання NP стандартний звіт не виводиться. За умовчанням виводиться стандартний звіт;

C – не використовується і збережений для сумісності з описами попередніх версій GPSS. Його використовують як лічильник "кадрів" для періодичного виведення звіту. Ця функція доступна у випадку використання в моделі кількох команд START і REPORT;

D – визначає необхідність виведення вмісту списків подій. Якщо у полі операнда D зазначити будь-яке додатне ціле число, то списки поточних і майбутніх подій вводяться до стандартного звіту. Якщо поле операнда D порожнє, то за умовчанням списки у стандартному звіті не виводяться.

Команду START можна записувати в кінці програми моделі (тоді після трансляції моделі відразу починається моделювання) або можна вводити у програму моделі в інтерактивному режимі.

Інколи може виникнути необхідність завершити моделювання не через певний час, а після проходження через модель певної кількості транзактів, які імітують, наприклад, виготовлені деталі, передані каналом зв’язку повідомлення тощо. У цьому випадку сегмент задання часу моделювання не потрібен. Для організації такого способу завершення моде-

лювання потрібно у блоках TERMINATE, які виводять з моделі транзакти, кількість яких визначає момент завершення моделювання, зазначити число, на яке зменшується лічильник завершень. У команді START також вказують число, ділення якого на записане у блоці TERMINATE число дасть потрібну кількість транзактів, які мають пройти через модель. Наприклад, якщо потрібно завершити моделювання після проходження через модель 1000 транзактів, то вона має містити такий сегмент

```
GENERATE      100,40
……
TERMINATE     1
START     1000
```

Блоків TERMINATE, які виводять з моделі транзакти, сумарна кількість яких, як у наведеному прикладі, дорівнює 1000, може бути кілька. Усі вони повинні мати 1 у полі операнда A. В інших блоках TERMINATE, якщо вони є в моделі, поле операнда A має бути порожнім.

Моделювання пристоїв

У GPSS елементами, які потребують обслуговування, є транзакти. Вони обслуговуються деякими пристроями: запити обробляються ЕОМ, робочі ремонтують обладнання, деталі обробляються верстатами тощо. Незалежно від того, люди це або предмети, обслуговуючі елементи називають пристроями чи приладами.

Пристрої характеризуються двома основними властивостями:

кожен пристрій у будь-який момент часу може обслуговувати лише один транзакт. Якщо у процесі обслуговування з’являється новий транзакт, то він повинен або почекати в черзі на своє обслуговування, або залишити систему необслуженим, або перервати обслуговування, яке відбувається в даний момент;

коли на пристрій надходить транзакт, у моделі потрібно передбачити час, необхідний для його обслуговування.

Оскільки пристроїв у моделі може бути багато, то для їхньої ідентифікації їм присвоюють імена – числові або символьні.

У загальному випадку у процесі моделювання використання пристрою відбуваються такі події:

1) очікування своєї черги;

2) займання пристрою;

3) обслуговування пристроєм;

4) вивільнення пристрою.

Під час моделювання можливі такі режими організації функціонування пристроїв:

займання пристрою і його звільнення;

переривання обслуговування пристроєм;

недоступність пристрою і відновлення доступності.

Зайняття та звільнення пристрою

Транзакти рухаються в моделі від блока до блока. Якщо в певний момент активності транзакт має намір зайняти пристрій, то для цього він заходить (або намагається увійти) до відповідного блока, який описує цей пристрій.

Блок, який описує пристрій, повинен мати такі властивості:

якщо пристрій вже використовують, то транзакт не може увійти до блока і мусить чекати у черзі або має бути вилучений з моделі;

якщо пристрій не використовують, то транзакт може увійти до блока. Унаслідок цього статус пристрою змінюється на "зайнятий".

Блок, що має такі властивості, це блок SEIZE (зайняти). Потраплянння транзакта у блок SEIZE моделює займання пристрою.

Після завершення обслуговування потрапляння того самого транзакта до блока RELEASE (звільнити) моделює звільнення пристрою і призводить до зміни стану пристрою із "зайнятий" на "незайнятий".

Формати блоків:

SEIZE A

RELEASE A

В обох блоках операнд А – це ім’я або номер відповідного пристрою, який займає (звільняє) транзакт. Операнд може бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. У той час як транзакти перебувають у моделі тимчасово, пристрої, які використовують у моделі, існують постійно протягом всього процесу моделювання.

Якщо транзакт заходить у блок SEIZE, то пристрій, зазначений у полі операнда А, перейшовши у стан "зайнятий", залишається в цьому стані аж до того часу, поки той самий транзакт не пройде через відповідний блок RELEASE, звільняючи пристрій.

Перед тим, як звільнити пристрій, транзакт може пройти через необмежену кількість блоків. Наприклад:

SEIZE Prystriy

RELEASE Prystriy

Якщо пристрій на ім’я Prystriy не зайнятий, то активний транзакт займає його. Якщо пристрій зайнятий, то транзакт переводиться до списку затримки цього пристрою позаду транзактів з тим самим пріоритетом і не заходить до блока SEIZE. Транзакту також не дозволяють увійти до блока SEIZE, якщо пристрій на ім’я Prystriy перебуває в недоступному стані.

Пристрій, як вже було зазначено, може мати ім’я або номер. У блоках SEIZE і RELEASE дозволяється записувати в полі операнда A номер безпосередньо без попереднього присвоєння його імені командою EQU. Наприклад:

SEIZE 4

RELEASE 4

Блоки SEIZE і RELEASE у разі виникнення необхідності можна перевизначати. Для цього вони повинні мати мітки.

Переривання обслуговування пристроєм

Переривання початого обслуговування можливе тільки для пристроїв і практикується лише при винятковій важливості транзакта, який щойно прийшов. Типовими прикладами є аварійні ситуації, потреба в екстренній медичній або іншій допомозі. Допущення переривань посилює ефект обслуговування з відносним пріоритетом.

Переривання і подальше поновлення початого обслуговування організовується блоками PREEMPT і RETURN. Заявка (транзакт), яка є причиною переривання, сама не може бути перерваною. Ім'я пристрою вказується в полі A блоку PREEMPT. Перерваний транзакт із записаним залишком часу обслуговування поміщається у список переривань і повертається у список майбутніх подій, коли переривання закінчиться.

Блок RETURN знімає переривання із вказаного в полі A пристрою. Якщо у блоці PREEMPT будуть транзакти вищого пріоритету, ніж перерваний, пристрій займе один з них.

PREEMPT A,B,C,D,E

Операнди:

A – ім'я або номер пристрою, що переривається;

B – режим зайняття пристрою. Операнд (необов'язковий) має бути PR або Null;

C – блок нового призначення транзакта, який у цей час володіє пристроєм;

D – номер параметра перерваного транзакта, який запам'ятовує недоданий час обслуговування, якщо перерваний транзакт видаляється зі списку майбутніх подій;

E – при значенні RE виключає перерваний транзакт з боротьби за пристрій (тоді операнд C необов'язковий).

Якщо пристрій не зайнятий, то блок PREEMPT поводиться подібно SEIZE. Інакше залежно від операнда B він діє в режимі пріоритету або в режимі переривання. У режимі пріоритету тільки транзакт з вищим пріоритетом може витіснити інший транзакт з пристрою. У режимі переривання (порожнє поле B), якщо пристрій вже зайнятий по пріоритету, активний транзакт поміщається у список завислих транзактів. При наданні пристрою транзактам із цього списку віддається більша перевага, ніж тим, що йдуть по пріоритету або зі списку затримки.

Операнди C – E визначають, що потрібно зробити з перерваним транзактом. Таким транзактам не дозволяється бути у списку поточних подій. Проте зняті з пристрою транзакти, перебування яких у блоці ADVANCE не завершене, поміщаються у список поточних подій автоматично, якщо у блоці PREEMPT опущені операнди C і E.

Транзакту не дозволяється входити у блок PREEMPT в режимі переривання, якщо пристрій у поточний момент зайнятий по пріоритету. Таким чином, GPSS не забезпечує імітацію вкладених переривань.

Блок RETURN A звільняє пристрій A або видаляє перерваний транзакт з боротьби за пристрій.

Тут можуть бути два випадки. Якщо транзакт у поточний момент володіє пристроєм, то він звільняє пристрій і намагається увійти до наступного блоку. Якщо є потреба вирішити питання про передачу пристрою наступному транзакту претендент послідовно розшукується у списках завислих, перерваних і, нарешті, затриманих транзактів. У другому випадку транзакт відстороняється від пристрою по пріоритету. Тут транзакт видаляється зі списку перерваних транзактів для цього пристрою, причому його право на володіння пристроєм не зачіпається.

Окремої розмови заслуговує питання про дообслуговування перерваної заявки. Для реалізації цієї дисципліни використовується операнд D блоку PREEMPT – обчислюється час, що залишився, і заноситься в параметр перерваного транзакта. При повторному перегляді списку поточних подій перерваний транзакт входить у блок ASSIGN, що працює в режимі приросту. Там до нього додається додаткова затримка –

фіксована або випадкова. При відновленні в блоці ADVANCE вміст згаданого параметра визначить затримку, що фіксується у списку майбутніх подій.

Недоступність пристрою і відновлення доступності

Пристрої, окрім ступеня зайнятості, характеризуються поточною готовністю. Режим готовності пристрою вводиться блоком FAVAIL, режим неготовності – блоком FUNAVAIL. Ця властивість дозволяє моделювати поломки техніки, регламентні роботи, перерви у касирів на здачу грошей і продавців – на прийом товару, роботу з перервами на обід, тижневий і річний ритми. Часи поломок генеруються аналогічно моментам прибуття, інтервали між ними зазвичай підпорядковані розподілу Вейбулла.

FAVAIL A

Блок встановлює вказаний операндом A пристрій у стан готовності.

FUNAVAIL A,B,C,D,E,F,G,H

Блок переводить пристрій у стан неготовності для вхідних транзактів. Операнди:

A – ім'я або номер пристрою;

B – розпорядження подальших дій для транзакта, що займав пристрій. Операнд необов'язковий, може бути RE, CO (REmove – видалити, COntinue – продовжити) або Null;

C – наступний блок для транзакта, що займав пристрій;

D – номер параметра для реєстрації часу, що залишився до закінчення обслуговування, якщо транзакт, що займав пристрій, видаляється зі списку майбутніх подій;

E – розпорядження подальших дій для транзактів, обслуговування яких було перерване до виконання блока FUNAVAIL. Операнд має бути RE, CO або Null;

F – новий блок для транзактів, виконання яких було перерване раніше;

G – подальші дії для транзактів, що чекали можливості зайняти пристрій до виконання блока FUNAVAIL. Операнд має бути RE, CO або Null;

H – новий блок, до якого перейдуть транзакти, що чекали можливості зайняти пристрій до виконання блока FUNAVAIL.

Складність блоку FUNAVAIL пояснюється тим, що він має справу з трьома типами транзактів:

1) тими, що займають пристрій (операнди B – D);

2) тими, що очікують поновлення обслуговування, перерваного блоком FUNAVAIL (операнди E – F);

3) тими, що очікували можливості зайняти пристрій до виконання FUNAVAIL (операнди G – H).

Транзакти, що надійшли в період неготовності, будуть затримані і не допущені до пристрою. Якщо в операндах записана опція RE, транзакти перестають претендувати на зайняття пристрою. Якщо в полі G для транзактів, що чекали можливості зайняти пристрій, вказано RE, то для подальшого перенаправлення транзактів треба заповнити поле H.

Якщо використовується опція CO, то транзакти, що займали пристрій, можуть продовжувати його займати навіть у період неготовності. У статистичних даних з використання пристрою цей час вразовуватиметься.

Якщо вказується блок нового призначення, транзакти звільняються від раніше встановлених зв'язків і прямують до нового блока. Затримані і перервані транзакти, керовані операндами G і H, не можуть бути перенаправлені без опції RE. Транзакт, що зайняв пристрій і керований операндами B – D, а також перервані транзакти, керовані операндами E і F, можуть знову претендувати на заняття пристрою, навіть будучи направленими за новим призначенням. Це робиться вказівкою зміненого призначення без використання опції RE.

Операнди B – D використовуються для управління транзактом, що володів пристроєм. Якщо B = CO, цей транзакт не позбавляється володіння пристроєм, але може отримати нове призначення через операнд C. При B = RE він видаляється і може продовжувати шлях у моделі, не заходивши в RETURN або RELEASE для даного пристрою. Тут операнд C обов'язковий.

Реалізація затримки на час обслуговування

Припустимо, що транзакт тільки що увійшов у блок SEIZE, тобто зайняв пристрій. Після виконання відповідної підпрограми для даного блоку планувальник відразу ж намагається просунути транзакт у наступний послідовний блок моделі. Існує мало обмежень на те, яким повинен бути цей блок. Це, наприклад, може бути інший блок SEIZE, що описує інший пристрій. Так робиться, якщо необхідно, наприклад, виконати

операцію з одночасним зайняттям і робітника, і якого-небудь інструменту для виконання певного виду обслуговування. Якщо робітник й інструмент моделюються деякою парою пристроїв, то транзакт одночасно повинен зайняти обидва ці пристрої, перш ніж може початися обслуговування.

Зазвичай, транзакт займає пристрій для того, щоб негайно почати в ньому своє обслуговування, яке триває деякий час. Протягом цього часу транзакт повинен припинити рух у моделі. Лише після завершення обслуговування він має потрапити у блок RELEASE і звільнити пристрій.

Для затримки транзакта на певний відрізок модельного часу використовують блок ADVANCE. Найчастіше цей проміжок задають випадковою змінною. Формат запису блоку ADVANCE:

ADVANCE A, B

Операнди:

A – задає середній час обслуговування;

B – задає половину поля допуску для значення операнда A.

Операнди A і B мають такий самий сенс, як і відповідні операнди блока GENERATE. Кожен з операндів A і B може бути іменем, числом, виразом у дужках, СЧА або СЧА*параметром.

Як і у блоці GENERATE, модифікатори можуть бути двох типів: модифікатор-інтервал і модифікатор-функція, отже, блок ADVANCE підраховує час затримки (приросту модельного часу) такими самими способами.

Якщо задані операнди A і B, і B не визначає функцію, то обидва операнди підраховуються (якщо вони не константи) і за час затримки береться значення випадкової величини, рівномірно розподіленої на проміжку [A–B, A+B]. Для розігрування випадкових чисел можна вибрати будь-який генератор рівномірно розподілених випадкових чисел (задається на сторінці Random Numbers у журналі Settings моделі в полі блока ADVANCE).

Приклади:

ADVANCE 15

ADVANCE 60,30

У першому прикладі обслуговування транзакта триватиме 15 одиниць модельного часу. У другому прикладі тривалість обслугодування транзакта рівномірно розподілена на інтервалі від 30 до 90 одиниць модельного часу.

Так само, як і у блоці GENERATE, незважаючи на спосіб обчислення часу затримки, значення операнда B не повинне перевищувати значення операнда A.

Для задання часу затримки, розподіленого за законом, відмінним від рівномірного, можна використовувати модифікатор-функцію, яку записують у полі операнда B. Під час звертання до функції планувальник визначає деяке число – значення функції, результат множення якого на значення операнда A використовується як час затримки.

Блок ADVANCE ніколи не перешкоджає потраплянню до нього транзакта. Будь-яка кількість транзактів може перебувати в цьому блоці одночасно. Блок ADVANCE можна розташовувати в будь-яких місцях моделі, а не лише між блоками SEIZE і RELEAZE. Час затримки може дорівнювати нулю. У такому випадку транзакт у блоці ADVANCE не затримується і переходить до наступного блока.

Лише блоки GENERATE і ADVANCE дають змогу розмістити транзакти у списку майбутніх подій. За допомогою цих блоків моделюють тривалість певної події або проміжок часу між настанням якихось подій.

10.2. Забезпечення пріоритетного обслуговування

Те, як впливає рівень пріоритетів транзактів на обробку, видно при розгляді списку поточних подій. При перенесенні транзактів зі списку майбутніх подій у список поточних подій транзакти у списку поточних подій займають положення, що є останнім усередині відповідного пріоритетного класу. Більш того, чим вище пріоритетний клас, тим ближче до початку списку поточних подій розташовується транзакт.

Оскільки планувальник GPSS проглядає список поточних подій з початку, він намагається просувати спочатку високопріоритетні транзакти. Пізніше, при перегляді він намагається просувати і транзакти, які мають низькі пріоритети.

Необхідність надання деяким транзактам пріоритетів часто витікає вже з постановки задачі.

Наприклад. При моделюванні одноколійної ділянки залізниці з роз'їздом посередині пріоритет може бути наданий одному з напрямів, пасажирським поїздам перед товарними і т. п. На автодорожніх перехрестях пріоритет визначається відомою водіям системою правил і загальною ситуацією. При моделюванні роботи технічних систем аварії мають пріоритет вище ніж основні транзакти.

Транзакти можуть мати цілочисельні пріоритети (за замовчуванням – нульовий). Пріоритет призначається при генерації (операнд E блоку GENERATE) або блоком PRIORITY в довільному місці моделі.

Блок PRIORITY присвоює транзакту потрібний пріоритет. Пріоритет впливає на порядок опрацювання транзактів процедурою перегляду і порядок займання ними апаратних об’єктів.

Блок PRIORITY має такий формат запису:

PRIORITY A,B

Операнд A задає нове значення пріоритету і може бути іменем, цілим числом, СЧА або СЧА*параметром. Нове значення пріоритету може бути меншим, більшим або дорівнювати поточному значенню пріоритету. Поле B визначає режим BUFFER, у ньому вказується значення "BU".

Загалом процедура перегляду списку поточних подій намагається просунути активний транзакт до наступного блока. Однак, якщо в полі B блока PRIORITY записано слово BU, то після присвоєння транзакту нового значення пріоритету блок PRIORITY стає еквівалентним блоку BUFFER. Транзакт вводиться до списку поточних подій у кінець свого нового пріоритетного класу, процедура перегляду повертається до початку списку поточних подій і починає перегляд знову. Оскільки блок BUFFER не затримує транзакти, які увійшли в цей блок, то транзакт буде опрацьований процедурою перегляду знову без зміни значення модельного часу.

Наприклад:

PRIORITY 7

Тут активному транзакту присвоюється пріоритет, який дорівнює 7.

Блок PRIORITY призначає новий рівень пріоритету і потім пересуває транзакт у списку поточних подій так, щоб він став останнім серед транзактів з відповідним значенням пріоритету. Ця операція виконується і в тому випадку, коли нове значення пріоритету дорівнює старому. Бувають випадки, коли потрібно змінити розташування транзакта в тому самому класі пріоритетів, в якому він перебуває.

Після того як планувальник завершить просування транзакта, який увійшов до блока PRIORITY, він автоматично повертається до початку перегляду списку поточних подій. Повернення до початку списку необхідне, оскільки у випадку, коли блок PRIORITY зменшив пріоритет активного транзакта, то планувальник пропустить усі транзакти, які перебувають у

списку поточних подій, починаючи з того місця, де був розташований активний транзакт до входження у блок PRIORITY, і закінчуючи його новим місцем у списку. Повернення до початку списку забезпечує перегляд усіх пропущених транзактів у той самий момент модельного часу.

Старшинство у пріоритеті само по собі не спричиняє переривання початого обслуговування – реалізується так званий відносний пріоритет, що стосується тільки очікування. Організація пріоритетного обслуговування з перериванням програмується спеціально. Інші дисципліни можуть бути запрограмовані роботою зі списком майбутніх подій. Вводити пріоритети має сенс тільки при великому завантаженні і малій питомій вазі пріоритетних заявок. У цих умовах він диференціює показники обслуговування – суттєво покращує їх для високопріоритетних заявок за рахунок деякого погіршення для низькопріоритетних. Інше призначення пріоритетів – модифікація логіки моделі.

Пріоритет і часові вузли

Призначення пріоритетів може потребуватися для забезпечення правильної послідовності обробки квазіодночасних подій ("часових вузлів"):

1. При одночасному прибутті заявки і закінченні обслуговування спочатку повинне завершитися обслуговування. Підвищення пріоритету обслуговуваного транзакта перед входом у блок ADVANCE дозволить запобігти необґрунтованій відмові у прийомі заявки. Так відбувається, наприклад, при вході і виході пасажирів автобуса – спочатку забезпечується вихід. Ще один часовий вузол утворюють вхід останнього пасажира і відправлення автобуса – пасажир повинен встигнути увійти.

2. При одночасному надходженні машини на автозаправну станцію і закриттям станції в кінці робочого дня машина повинна бути обслужена. Для цього транзакту-машині при генерації повинен бути привласнений вищий пріоритет, ніж часовому транзакту.

Висновки

1. Кожному транзакту в досліджуваній системі у відповідність ставиться сегмент моделі, тобто самостійна послідовність зв'язаних блоків, яка описує весь життєвий цикл тимчасових елементів одного класу, починаючи з їх появи в системі і закінчуючи їх видходом з неї. Модель може включати декілька сегментів.

2. Кожен сегмент моделі починається з одного або декількох блоків GENERATE, які створюють транзакти, і закінчується блоком TERMINATE, який ці транзакти видаляє з моделі. Кожен транзакт, що з'явився в моделі, послідовно обробляється блоками, які входять у сегмент. Одночасно в моделі обробляється тільки один транзакт, решта транзактів моделі чекає своєї черги на обробку.

3. Переривання початого обслуговування практикується лише при винятковій важливості транзакта.

4. Пріоритет транзактів встановлюється при їх появі в моделі і може бути змінений у будь-якому її місці. Старшинство у пріоритеті впливає тільки на порядок обслуговування транзактів у моделі і не викликає переривання.

Контрольні запитання та завдання

1. Дайте визначення транзакта, обслуговуючого пристрою, черги.
2. Дайте визначення черги, обслуговування.
3. Назвіть блоки, що забезпечують побудову моделі типу "СМО з одним пристроєм". Особливості їх роботи.
4. Дайте визначення дисципліни обслуговування.
5. Яким чином у GPSS World забезпечується пріоритетне обслуговування?
6. Дайте визначення переривання обслуговування.
7. Яким чином у GPSS World організуються переривання обслуговування?

Тема 11. Блоки, що забезпечують побудову моделі типу "багатоканальні СМО". Засоби GPSS World, що використовуються для забезпечення точності результатів імітаційного моделювання

11.1. Блоки, що забезпечують побудову моделі типу "багатоканальні СМО"

Два та більше пристроїв, призначених для обслуговування, можуть бути змодельовані на GPSS відповідною кількістю пристроїв, розталованих поруч, тобто паралельно. Це стосується випадку, коли окремі

пристрої є різними, тобто мають різні характеристики, наприклад, інтенсивність обслуговування.

Однак часто паралельно працюючі пристрої є однорідними. Наприклад, автоматична телефонна станція, автобус, ПК, що працює в мультипрограмному режимі, тощо. GPSS передбачає для моделювання однорідних паралельних пристроїв спеціальний засіб – багатоканальний пристрій. БКП може використовуватися кількома транзактами одночасно. Обмежень на кількість БКП у моделі немає. Для того щоб їх розрізняти їм дають різні імена. БКП визначається до його використання командою STORAGE. Її формат такий:

<ім'я БКП> STORAGE A.

Операнд А визначає ємність БКП, під якою розуміють кількість обслуговуючих каналів у БКП або кількість пристроїв, об'єднаних у БКП.

У моделі можна організовувати функціонування БКП у двох режимах:

займання та звільнення БКП;

доступність і недоступність БКП.

Займання та звільнення БКП імітують блоки ENTER (увійти) і LEAVE (вийти, залишити). Формати блоків:

ENTER A,B

LEAVE A,B

А – ім'я або номер БКП;

В – кількість пристроїв, що вивільняється.

Операнд А в обох блоках використовують для позначення імені відповідного БКП. Операнд В задає кількість пристроїв (каналів), які мають бути зайняті у блоці ENTER або звільнені у блоці LEAVE у момент потрапляння до них транзакта. Зазвичай, поле операнда В порожнє, і в цьому випадку за замовчуванням займається або звільняється один канал. Якщо В = 0, то блок вважається недієздатним.

Під час входження транзакта у блок ENTER операнд А слугує для знаходження БКП із зазначеним іменем. Якщо такого БКП немає, то відбувається зупинка внаслідок помилки. Якщо БКП існує і задано операнд В, то планувальник його підраховує, округлює до цілого, й отриманий результат використовує для оцінювання вільної ємності (кількості вільних каналів). Транзакт може увійти до блока ENTER, якщо БКП перебуває в доступному стані і достатньо ємності для виконання

замовлення. В іншому випадку транзакт переводиться до списку затримки пристрою згідно з його пріоритетом.

Коли транзакт заходить до блока ENTER, планувальник виконує такі дії:

збільшує на 1 лічильник входжень БКП;

збільшує на значення операнда B (за замовчуванням на 1) поточний вміст БКП;

зменшує на значення операнда B (за замовчуванням на 1) доступну ємність БКП.

Якщо транзакт перед входженням у блок ENTER подає запит на більшу кількість каналів, ніж та, яка визначена командою STORAGE, тобто її операнд A менший від операнда B блока ENTER, то виникає помилка.

БКП можна перевизначити, тобто змінити ємність ще однією командою STORAGE з тим самим іменем. Наприклад:

Bkp STORAGE 3

Повторний опис:

Bkp STORAGE 5

Імітацію обслуговування протягом певного проміжку часу в БКП також здійснюють блоком ADVANCE.

БКП, окрім ступеня зайнятості, характеризується поточною доступністю. На відміну від сукупності пристроїв, БКП може виявитися недоступним лише в цілому. Режим доступності БКП вводиться блоком SAVAIL, режим недоступності – блоком SUNAVAIL.

Недоступність тільки забороняє вхід транзактів у БКП, їх обробка продовжується. Здійснити імітацію виходу БКП з ладу, при якому всі транзакти, що перебувають у БКП на обслуговуванні, губляться, блоками SUNAVAIL і SAVAIL неможливо.

Недоступність БКП моделюють за допомогою блока SUNAVAIL. Формат блока такий:

SUNAVAIL A

Блок SUNAVAIL переводить БКП A у стан недоступності.

Якщо в час переведення в недоступний стан у БКП перебували транзакти, тобто поточний вміст БКП не дорівнював нулю, то обслуговування цих транзактів продовжується, поки поточний вміст БКП не буде дорівнювати нулю. Транзакти, які намагаються зайняти БКП у час

перебування його в недоступному стані, не заходять до блока ENTER і заносяться до списку затримки БКП.

Перебування в недоступному стані триває до того часу, поки транзакт не увійде до блока SAVAIL. Формат блока:

SAVAIL A

Блок SAVAIL встановлює БКП A в доступний стан.

Якщо в момент відновлення доступності БКП у його списку затримки були транзакти, їм надається можливість зайняти БКП відповідно до дисципліни "first-fit-with-skip" (перший придатний з перервою). Ті транзакти, запити яких не можуть бути задоволені, залишаються у списку затримки.

Усі зміни враховуються у статистиці, що збирається про функціонування БКП.

11.2. Засоби GPSS World, що використовуються для забезпечення точності результатів імітаційного моделювання

Оцінка точності результатів моделювання пов'язана з побудовою довірчих інтервалів для вихідних змінних (відгуків) моделі. Кількість реалізацій (прогонів моделі) та час прогону для кожної реалізації моделі визначають точність результатів. Якщо модель детермінована, то для отримання точних результатів моделювання достатньо одного прогону. У загальному випадку дані одного прогону моделі є одиничною вибіркою або часовим рядом. Часовий ряд – кінцева реалізація випадкового процесу, тобто в результаті кожного прогону моделі утворюються часові ряди для кожного відгуку моделі досліджуваних стохастичних процесів. Для стохастичних моделей розглядають два режими роботи: перехідний і стаціонарний. Стаціонарний режим визначається стаціонарним процесом на виході моделі.

Якщо модель працює в перехідному режимі, то необхідну кількість прогонів моделі можна розрахувати за формулами:

$$N = t_{\alpha}^{2} \frac{p(1-p)}{\varepsilon^{2}} + 1 \qquad (11.1)$$

або

$$N = \frac{t_\alpha^2 \sigma^2}{\varepsilon^2} + 1, \qquad (11.2)$$

де p – імовірність настання деякої події, що визначає стан модельованої системи;

σ^2 – середнє квадратичне відхилення;

ε – точність, може бути задана рівною ± 5% від середнього значення величини, для якої будується довірчий інтервал при $\alpha = 0.95$;

t_α – табличне значення критерія Стьюдента.

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. Так, у GPSS World для цього призначена процедура ANOVA. Після останнього прогону моделі 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.

Розглянемо детальніше процедуру ANOVA.

До більшості складних систем застосовний принцип Парето, згідно з яким 20 % факторів визначають властивості системи на 80 %. Тому першочерговим завданням дослідника імітаційної моделі є відсіювання незначущих факторів, що дозволить зменшити розмірність задачі оптимізації моделі.

Аналіз дисперсії (Analysis of Variation – ANOVA) оцінює відхилення спостережень від загального середнього. Потім варіація розбивається на частини, кожна з яких має свою причину. Залишкова частина варіації (residual), яку не вдається повязати з умовами експерименту, вважається його випадковою похибкою. Похибка оцінюється в середньоквадратичних відхиленнях. Зазвичай тільки ті ефекти, які перевищують величину СКВ, вважаються значущими. Для підтвердження значущості використовується спеціальний тест – F-статистика.

При імітації похибки дослідів можливо зробити незалежними, застосовуючи в різних дослідах різні послідовності випадкових чисел. Ці похибки не обовязково повинні бути нормально розподіленими із загальною дисперсією, однак F-критерій мало чутливий до порушення нормальності.

Нижче наведений приклад ANOVA-таблиці.

ANOVA

Джерело варіації	Сума квадратів	Степені свободи	Середній квадрат	F	Критичне значення F (p = 0.05)
A	28.000	2	14.000	5.600	3.89
B	21.000	3	7.000	2.800	3.49
AB	11.000	6	1.833	0.733	3.00
Похибка	30.000	12	2.500		
Всього	90.000	23			

Рівень		Лічильник	Середнє значення	Min	Max	95 % інтервал
A	B					
1	1	2	3.000	2.000	4.000	[0.564, 5.436]
1	2	2	1.000	1.000	1.000	[-1.436, 3.436]

Унизу таблиці загальна сума квадратів розділена, і компоненти її повязані з факторами та їх взаємодіями. Залишкова сума квадратів показана у рядку "Похибка". Середня сума квадратів похибкового члена використовується для оцінки СКВ експеримента.

Кожна сума квадратів має дільник, що дорівнює числу степенів свободи. Фактори і їх взаємодія у статистичній моделі представляються в таблиці окремими рядками. У кожному рядку маємо суму квадратів і степені свободи, повязані з цією оцінкою. З них виводяться інші необхідні величини. Просте ділення дає середній квадрат; діленням цієї величини на середньоквадратичну похибку з нижнього рядка таблиці отримуємо F-статистику для цього ефекту. Наведена таблиця показує, що ефект фактора A значущий, а ефекти B і AB незначущі.

Іноді експеримент не виявляє ефект, навіть якщо він реально існує. Одна з цілей – зробити таку ситуацію якомога найменш імовірною. Для цього є два шляхи: збільшити F-статистику або зменшити критичне значення. Бажано видалити частини суми квадратів похибок, повязані з будь-яким важливим фактором, що не включений в аналіз, – тоді F-статистика стане більшою. При природньо-наукових експериментах це досягається проведенням їх в максимально однорідних умовах. При

імітаційному моделюванні треба врахувати додаткові фактори, які впливають на результати експерименту.

Два інших підходи повязані зі збільшенням числа степенів свободи похибкового члена. Перший полягає у збільшенні кількості реплік. Це найбільш затратний, але простий і надійний шлях. Друга можливість повязана з плануванням експеримента і вдосконаленням статистичної моделі, покладеної в основу ANOVA. Середній квадрат похибки фактично і є залишком після видалення інших квадратів. Якщо віднайдемо прийнятний шлях зберегти більше даних після віднімання ефектів, отримаємо оцінку СКВ з більшим числом степенів свободи. Критичне значення F-статистики буде меншим, збільшиться потужність критерія. Саме це відбувається при ігноруванні деяких взаємодій.

У багатофакторних експериментах часто видаляють взаємодії високого порядку, щоб покращити ту інформацію про головні фактори і взаємодії низького порядку, які маються. Якщо взаємодії високого порядку відсутні, додаткові степені свободи використовуються для кращої оцінки F-статистики. Крім того, збільшення числа степенів свободи означає менше критичне значення F. У противному випадку краще використовувати додаткові репліки.

Не всі екперименти потребують "реплічного" вимірювання. При використанні метода латинських квадратів ANOVA здатна отримати оцінку СКВ і без того. Коли ANOVA не може розрахувати F-статистику, необхідно додати репліки, або видалити зі статистичної моделі взаємодії висого порядку. Обидва ці підходи потребують зміни в організації виклику ANOVA. Однак більш ефективними методами покращення статистичної значущості результатів є зменшення їх дисперсії.

Висновки

1. Багатоканальні пристрої призначені для моделювання однорідних паралельних пристроїв. БКП визначаються до їх використання в моделі.

2. Обслуговування транзакта багатоканальним пристроєм не може бути перерване.

3. Оцінка точності результатів моделювання пов'язана з побудовою довірчих інтервалів для вихідних змінних (відгуків) моделі. Для цього в GPSS World призначена процедура ANOVA

Контрольні запитання та завдання

1. Дайте визначення багатоканального пристрою.
2. Назвіть відмінності пристрою від багатоканального пристрою з ємністю 1.
3. Назвіть блоки, що забезпечують побудову моделі типу "багатоканальні СМО". Особливості їх роботи.
4. Назвіть засоби GPSS World, які використовуються для забезпечення точності результатів імітаційного моделювання. Дайте їх коротку характеристику.

Тема 12. Стандартні числові й логічні атрибути та їх використання в моделях. Функції в GPSS World. Їх використання в моделях

12.1. Стандартні числові й логічні атрибути. Їх використання в моделях

У процесі моделювання GPSS World автоматично реєструє і коректує інформацію, подану системними числовими атрибутами різних об’єктів, які входять до моделі. Значна частина інформації доступна лише інтерпретатору. Проте до деяких атрибутів об'єктів може звертатися і програміст (розробник моделі), маніпулюючи їх значеннями згідно з логікою моделі. Такі атрибути називаються стандартними числовими атрибутами (СЧА). СЧА можуть мати дійсні, цілі, логічні і рядкові значення. Тип значення у випадку його очевидності не вказується. Усі часові й усереднені показники вважаються дійсними.

Кожен об'єкт GPSS має свій набір СЧА. Окрім СЧА об'єктів існують ще системні числові атрибути, до яких користувач може звертатися в моделі, але не може змінювати їх значення. Доступ до СЧА дає можливість керувати процесом руху транзактів, наприклад, обмежувати розмір або час перебування у черзі. Повний перелік СЧА, класифікованих за об’єктами GPSS, наведено в додатку А.

Імена СЧА зарезервовані. За способом визначення імен СЧА можна розбити на три групи.

Ім'я СЧА в першій групі складається з двох частин. Перша частина вказує групове ім'я, яке ідентифікує тип об'єкта і тип інформації про

об'єкт. Друга частина ідентифікує конкретного члена групи. Групове ім’я складається з однієї-двох літер, фіксованих для інформації про об’єкти певного типу (наприклад, Q – поточне значення довжини черги, SC – загальне число входів у БКП, FT – середній час використання пристрою одним транзактом тощо). Об’єкти GPSS можуть бути ідентифіковані за допомогою числових або символьних імен.

Ідентифікація СЧА залежить від виду адресації, яких у GPSS передбачено дві: пряма і непряма.

У разі прямої адресації СЧА визначають як:

1. Якщо об’єкт ідентифіковано за допомогою номера, то посилання на його СЧА записується так:

СЧАj, де j – номер об’єкта (додатне ціле число).

Наприклад, QM8 – максимальна довжина черги номер 4, FR5 – коефіцієнт використання пристрою номер 5, R7 – кількість вільних каналів у БКП номер 7.

2. Якщо зазначають символьне ім’я об’єкта, то посилання на його СЧА записується так:

СЧА$<ім'я>, де <ім'я> – ім’я об’єкта.

Наприклад, Q$Cherga – поточне значення довжини черги з іменем Cherga, FC$Pristry – загальне число входів у пристрій з іменем Pristry, CA$Myspisok – середнє число транзактів у списку користувача з іменем Myspisok.

У разі непрямої адресації СЧА визначають так:

СЧА*j, де j – номер параметра активного транзакта (додатне ціле число), який містить необхідне значення;

СЧА*$<ім'я>, де <ім'я> – ім’я параметра активного транзакта, який містить необхідне значення. Знак $ використовується як роздільник і його можна не використовувати. По суті записи і СЧА*<ім'я> ідентичні.

У загальному випадку непряму адресацію позначають так:

СЧА*параметр.

Наприклад, Q*4 – поточне значення довжини черги, номер якої є значенням параметра 4 активного транзакта, SR*Pristry – коефіцієнт використання БКП, номер якого міститься в параметрі з іменем Pristry активного транзакта.

Непряма адресація реалізується за допомогою блока ASSIGN через запис номера об'єкта в параметрі транзакта.

Другу групу, як частковий випадок першої групи, становлять матричні СЧА. Матричний СЧА MX може містити до трьох ідентифікаторів (непрямих адрес). Наприклад:

MX*Result(*Rjadok,*Stovpec) є посиланням на матрицю, номер якої міститься в параметрі активного транзакта з іменем Result, а елемент матриці – номер рядка і номер стовпця – визначають значеннями параметрів з іменами Rjadok і Stovpec відповідно того самого активного транзакта.

Третю групу складають стандартні числові атрибути A1, AC1, C1, M1, PR, TG1, XN1, Z1, які називають атомарними. Ці СЧА, на відміну від СЧА першої і другої груп, не потребують зазначення номера або імені об’єкта.

СЧА ***за ознакою доступності*** до них користувача можна розділити на дві групи.

До першої групи належать атрибути, які в частині, що стосується їхнього формування, доступні лише GPSS World. Користувач може їх використати у виразах, але не може їх змінювати у процесі моделювання. Ця група містить найбільшу кількість СЧА.

Друга група охоплює СЧА обчислювальних об’єктів, які може змінювати розробник моделі і використовувати їх у виразах:

FNj (FN$Resultat) – обчислене дійсне значення функції номер j (з іменем Resultat);

Vj (V$Zminna) – обчислене значення дійсної або цілочисельної змінної номер j (з іменем Zminna);

BVj (BV$Bulev) – обчислене дійсне значення булевої змінної номер j (з іменем Bulev).

Стандартні числові атрибути дозволяють відобразити залежність поводження транзактів від стану системи: затримки чи траєкторії від параметра транзакта, довжини черг до пристроїв від години доби; тарифів – від відстані, години доби і дня тижня тощо. СЧА можуть бути використані як входи в матрицю. Другий аспект застосування СЧА – забезпечити нестандартні збір і обробку статистики.

Стандартні числові атрибути можуть використовуватися як операнди практично в будь-якому типі блоків. Також значення будь-якого СЧА може входити до більшості операторів опису об'єктів. Особливістю СЧА є те, що вони забезпечують користувачеві доступ до характеристик стану системи у процесі моделювання.

12.2. Функції в GPSS World. Їх використання в моделях

Генерування випадкових чисел у GPSS

Загалом джерелом "випадковості" в моделях GPSS є одна або кілька функцій, звертаючись до яких можна отримати випадкові значення з вибірки рівномірно розподілених на проміжку [0; 1] чисел. Розігрування значення відповідно до деякого розподілу, зазвичай, здійснюють в два етапи:

1) розігрується випадкове число з вибірки чисел, рівномірно розподілених на проміжку [0; 1];

2) отримане число певним способом перетворюється на еквівалентне йому значення, яке вже належить до вибірки чисел потрібного нам розподілу.

У GPSS World передбачена необмежена кількість генераторів рівномірно розподілених псевдовипадкових чисел на проміжку [0; 1]. Звертаються до генератора без попереднього його оголошення за допомогою СЧА RNn, де n – номер генератора. Початкове згенероване генератором число збігається з номером n генератора. Наприклад, запис RN823 означає, що генератор номер 823 запускається початковим числом 823.

Під час моделювання може виникнути необхідність дослідження:

різних варіантів моделі із заданням одного і того ж самого вхідного потоку випадкових чисел;

одного або кількох варіантів моделі із заданням різних потоків випадкових чисел.

Перша необхідність легко реалізується, оскільки генератори дають змогу відтворювати одні й ті ж самі послідовності рівномірно розподілених випадкових чисел.

У другому випадку GPSS надає можливість змінювати початкові числа перших семи RN1,...,RN7 генераторів і, як наслідок, формувати різні послідовності.

Для цього призначена команда RMULT такого формату:

RMULT A,B,C,D,E,F,G

Операнд A – початкове число для RN1, B – для RN2, ..., G – для RN7.

Наприклад, запис

RMULT ,23,74,,,59

означає встановлення початкових чисел для генераторів 2, 3 і 6. Початкові значення генераторів 1, 4, 5 і 7 залишаються без змін.

У GPSS World псевдовипадкові числа генеруються за 32-бітним мультиплікативним конгруентним алгоритмом, який дає змогу отримати 2147483646 унікальних псевдовипадкових чисел, тобто з періодом 231-2, не враховуючи 0.

СЧА RN набуває цілих значень від 0 до 999 включно, а під час обчислення випадкових функцій використовується випадкове число з інтервалу від 0 до 0,999999 включно.

У GPSS World генератори випадкових чисел використовують не лише у блоках GENERATE і ADVANCE, а й під час опрацювання одночасних подій та у блоках TRANSFER, які працюють у режимі статистичної передачі. Користувач може вибрати будь-який генератор, знайшовши в меню Edit – Settings на закладці Random Numbers поле відповідного блока (за замовчуванням – RN1).

Генерування випадкових чисел у GPSS World як для рівномірного, так і для інших типових розподілів найзручніше здійснювати за допомогою вмонтованих бібліотечних генераторів випадкових чисел (див. підрозділ 12.3). Однак з цією метою можна також використовувати й функції, які в GPSS призначені для обчислення величин, заданих табличними залежностями.

Примітка. Оператори CLEAR і RESET не змінюють значень індексів і множників датчиків.

Табличні функції

Табличну функцію GPSS World визначає оператор FUNCTION, після якого йдуть пари можливих значень аргумента й функції. Оператор FUNCTION має такий формат:

<ім'я> FUNCTION A,B

Тут <ім'я> – ім'я функції (числове або символьне).

Операнд A – аргумент функції, який може бути іменем, додатним цілим числом, рядком, виразом у дужках, значенням будь-якої іншої функції, СЧА або СЧА*параметром. Якщо як аргумент функції використовується випадкове число RNj, то значеннями аргумента будуть числа, рівномірно розподілені в інтервалі 0<=RNj<1.

Операнд B складається з однієї літери, яка визначає тип функції, й цілого додатного числа, яке задає кількість координат Операнд B задає

тип функції й кількість координат (пара точок аргумент-функція X(i) й Y(i)).

У GPSS World можна використовувати такі п’ять типів функцій:

C – неперервна числова функція. Значення функції обчислюється методом лінійної інтерполяції й береться як ціле число;

D – дискретна числова функція. Значення функції береться рівним значенню на правому кінці інтервалу як ціле число;

E – дискретна атрибутивна функція. У ній можна використати будь-який СЧА, крім матриць;

L – спискова числова функція, аргументом якої є відрізок натурального ряду, починаючи з 1;

M – спискова атрибутивна функція. Значенням функції може бути будь-який СЧА.

За кожним оператором опису FUNCTION ідуть оператори для задання координат (X(i) й Y(i)) функції. Не допускається використання коментарів між оператором опису FUNCTION й операторами, що задають значення функції.

При написанні пар чисел, які визначають координати X та Y точки функції, необхідно дотримуватися таких правил:

запис має починатися в позиції 1;

значення координат X(i) й Y(i) однієї точки функції відділяються комою;

набори координат відокремлюють символом "/" або переходом до наступного рядка і розташовують у довільній кількості рядків;

кількість пар даних, зазначених в операнді B оператора FUNCTION, має збігатися з кількістю пар, відокремлених символом "/" у списку даних функції;

значення X розташовують у порядку зростання;

кожна функція повинна мати, принаймні, дві описані точки;

списки даних функції не мають полів коментаря.

Значення, задані командою FUNCTION, зберігаються у формі чисел з плаваючою крапкою подвійної точності.

Функції всіх типів мають єдиний СЧА з назвою FN, значенням якого є обчислене значення функції. Обчислення функції відбувається під час потрапляння транзакта у блок, який містить посилання на СЧА FN з іменем функції.

Неперервні числові функції (C). Функції типу C використовують для отримання випадкових чисел з неперервного рівномірного розподілу. Коли значення аргумента неперервної числової функції попадає в інтервал між двома заданими значеннями X(i) і X(i+1), програма виконує лінійну інтерполяцію для визначення значення функції, що знаходиться в інтервалі між Y(i) і Y(i+1). Результат береться як ціле число (десяткові знаки відкидаються).

У списку даних функції змінні X та Y мають бути цілими або дійсними числами. Наприклад:

```
Myfn FUNCTION RN12,C4
.1,12.3/.3,15.2/.6,7.5/1,45
```

Значення неперервної функції Myfn обчислюється наступним чином. Після звертання до функції Myfn відбувається розігрування випадкового числа, яке надалі використовується як аргумент функції (нехай було розігране значення X=.34). Потім переглядається таблиця для визначення інтервалу значень накопиченої частоти, до якого потрапило розігране випадкове число (інтервал [.3; .6]). Після цього виконується лінійна інтерполяція на цьому інтервалі й обчислюється значнення функції в розіграній точці:

$$Y = \frac{x - x(i+1)}{x(i) - x(i+1)} \times Y(i) + \frac{x - x(i)}{x(i+1) - x(i)} \times Y(i+1) =$$

$$= \frac{.34 - .6}{.3 - .6} \times 15.2 + \frac{.34 - .3}{.6 - .3} \times 7.5 = 14.173$$

Оскільки від отриманого результату береться ціла частина, то значення функції Myfn у точці X=.34 дорівнює 14.

Значення неперервної функції обчислюється за допомогою лінійної інтерполяції, тому для всіх точок на одному інтервалі накопиченої частоти ймовірність їхнього використання є однаковою. Унаслідок цього процедура розіграшу неперервної рівномірно розподіленої випадкової величини є простою. Припустимо, що деяка випадкова величина рівномірно розподілена на проміжку [5; 10]. Тоді ймовірність того, що значення змінної менше ніж 5, дорівнює 0, а ймовірність того, що вона менша ніж 10, дорівнює 1. Ці два значення ймовірності використовуються як значення накопиченої частоти для визначення неперервної функції:

Chas FUNCTION RN4,C2

0,5/1,10

Незважаючи на те, що значенням другої пари, використаної для визначення функції, є 10, функція ніколи не набуває значення 10, оскільки СЧА RN під час обчислення функції не може перевищувати 0,999999. Отож найбільшою цілою частиною отримуваного значення є 9.

Дискретні числові функції (D). Дискретні числові функції задають одне й те саме значення функції Y(i) для всіх значень аргумента X(i–1)<X<=X(i). Значення функції береться рівним значенню функції на правому кінці інтервалу. Нецілі значення функцій приводяться до цілих шляхом виділення цілої частини.

У списку даних функції типу D значення аргумента X мають бути виразом, а значення функції Y – ім'ям, цілим або дійсним числом. Наприклад:

Myfd FUNCTION RN3,D5

.14,2/.4,7/.68,8/.77,11/1,15

Значення дискретної функції Myfd обчислюється наступним чином. Після звертання до функції Myfd відбувається розігрування випадкового числа, яке надалі використовується як аргумент функції (нехай було розігране значення X=.7). Потім переглядається таблиця для визначення інтервалу значень накопиченої частоти, до якого потрапило розігране випадкове число (інтервал [.68; .77]). Значення функції Myfd у точці X=.7 дорівнює значенню функції на правому кінці цього інтервалу, тобто дорівнює 11.

Якщо результатом розігрування мають бути цілі числа, то перевага надається неперервній функції перед дискретною. Припустимо, що необхідно розігрувати випадкові числа на проміжку від 100 до 234. Функцію можна описати так

Gra FUNCTION RN32,C2

0,100/1,235

Зауважимо, що друга пара тут (1; 235), а не (1; 234). Це зроблено для забезпечення можливості отримання максимального значення 234.

Якщо ж для розігрування випадкових чисел на тому самому проміжку використати дискретну функцію, то потрібно буде рахувати 134 пари накопичених частот і задати 134 пари для визначення функції.

Спискові числові функції (L). У багатьох випадках значеннями аргумента X(i) є неперервні послідовності цілих чисел 1,2,3...n.

Машинний час, необхідний для обчислення значень таких функцій, може бути значно зменшено, якщо ці функції описані як табличні. У цьому випадку значення аргумента X в операторах, що задають значення координат функції, програмою введення GPSS не розглядаються і приймаються рівними X(1)=1, X(2)=2,..., X(i)=i,..., X(n)=n.

Інтерпретатор використовує аргумент функції для прямого звертання до масиву заданих величин функції. Таким чином, не потрібен послідовний перегляд таблиці, пов'язаної з функцією.

Значення Y(i) мають бути записані у відповідних полях операторів, що задають значення координат. Для зручності програміста в операторах можуть бути записані значення X, хоча вони ніколи не переглядаються. Якщо значення аргумента виходить за межі інтервалу (1,n), то видається повідомлення про помилку на етапі виконання.

У функції типу L аргумент X може бути лише цілим числом, його розглядають як порядковий номер. Значення функції Y – ім'ям, цілим або дійсним числом. За порядковим номером знаходять значення функції. Значення X мають починатися з 1 і збільшуватися на 1 для кожної наступної пари даних. Наприклад:

```
Myfunl FUNCTION Q$Cherga,L5
1,Stan1/2,Stan2/3,Stan3/4,Stan4/5,Stan5
```

У цьому прикладі якщо довжина черги з іменем Cherga дорівнюватиме 3, то значення функції Myfunl дорівнюватиме Stan3.

Дискретні атрибутивні функції (E). Дискретні атрибутивні функції (E) подібні дискретним числовим функціям (D). У дискретній атрибутивній функції типу E аргумент X має бути виразом, а Y – ім'ям, цілим або дійсним числом, СЧА або виразом у дужках. Наприклад:

```
Myfune FUNCTION P5,E4
1,X$Use1/4,V$Use2/10,S$Comp/12,FN$Myfun
```

Функцію типу E обчислюють так само, як і функцію типу D. Відмінність полягає лише в тому, що після ідентифікування аргументу X відбувається обчислення відповідного СЧА або виразу в дужках. Наприклад, для значення 5-го параметра транзакта (P5), який дорівнює 4, відбувається обчислення арифметичної змінної з іменем Use2.

Спискові атрибутивні функції (M). Спискові атрибутивні функції (M) подібні списковим числовим функціям (L). Спискова атрибутивна

функція типу M відрізняється від спискової числової функції типу L тим, що значення функції можуть бути не лише ім'ям, цілим або дійсним числом, а й СЧА або виразом у дужках. Аргумент є також цілого типу, його розглядають як порядковий номер, тому він має починатися з 1 і збільшуватися на 1 для кожної наступної пари даних. Наприклад:

```
Myfunm FUNCTION     X$Zbereg,M4
1,Q$Cherga/2,N$Term/3,S$Mys/4,V$Zmin
```

Обчислюється функція типу M так само, як функція типу L, лише після ідентифікації аргумента – порядкового номера відбувається обчислення відповідного СЧА або виразу в дужках. Наприклад, при X=4 обчислюється арифметична змінна з іменем Zmin.

Звернення до функції можна записувати в операндах блоків, у виразах, як аргумент інших функцій.

Звернення до функції можна записувати в операнді B блоків ADVANCE або GENERATE. У цьому випадку вона називається "функцією-модифікатором". Значення цієї функції обчислюється з подвійною точністю й множиться на значення операнда A. Отриманий результат використовують як потрібну часову затримку в цьому блоці.

12.3. Оператори опису деяких імовірнісних розподілів

Розглянемо PLUS-процедури генерування випадкових чисел з типовими теоретичними розподілами. У зв’язку з тим, що форма задавання і навіть кількість параметрів (з урахуванням зсуву) розподілів у GPSS World часто відрізняються від стандартних, то розглянемо і їх характеристики. Виклик кожної функції оформлений як оператор присвоювання із зазначенням в його лівій частині типу значення, що повертається (Real – дійсний, Integer – цілий).

Усі процедури викликаються по їх вказівнику. Якщо виникає необхідність задати початковий ДВЧ, підставляється тільки номер останнього – без попереднього RN. У всіх випадках через Min і Max позначені відповідно найменше і найбільше значення генерованої випадкової величини, m – зсув, s – масштабний параметр ($m > 0$, $s > 0$), α і β – параметри форми, які в операторах виклику замінюються їх латинськими аналогами. Усі аргументи мають бути типу Вираз.

У GPSS World передбачено 24 імовірнісних розподіли.

Неперервні розподіли

1. Рівномірний розподіл:

$$Real = UNIFORM(RNj, a, b),$$

де $a = \min$, $b = \max$.

Характеристики:

щільність розподілу на інтервалі (a,b): $f(x) = \frac{1}{b-a}$, $a \le x \le b$;

середнє значення: $\nu = \frac{a+b}{2}$;

дисперсія: $D = \frac{(b-a)^2}{12}$.

2. Експоненціальний розподіл:

$$\mathrm{Real} = EXPONENTIAL(\mathrm{RNj}, \mathrm{m}, \mathrm{s}).$$

Характеристики:

щільність розподілу: $f(x) = \frac{1}{s} exp\left(-\frac{x-m}{s}\right)$, $x \ge m$;

середнє значення: $\nu = m + s$;

дисперсія: $D = s^2$.

3. Нормальний розподіл (розподіл Гауса):

$$Real = NORMAL(RNj, m, s).$$

Характеристики:

щільність розподілу: $f(x) = \frac{1}{s\sqrt{2\pi}} exp\left(-\frac{(x-m)^2}{2s^2}\right)$;

середнє значення: $\nu = m$;

дисперсія: $D = s^2$.

4. Трикутний розподіл:

$$\mathrm{Real} = TRIANGULAR(\mathrm{RNj}, \mathrm{a}, \mathrm{b}, \mathrm{c}),$$

де $a = \min$, $b = \max$, $c = \mathrm{mod}\, e$.

Характеристики:

щільність розподілу: $f(x)=\begin{cases}\dfrac{2(b-x)}{(b-a)(c-a)}, a\le x\le c,\\ \dfrac{2(b-x)}{(b-a)(b-a)}, c\le x\le b.\end{cases}$;

середнє значення: $\nu=\dfrac{a+b+c}{3}$;

дисперсія: $D=\dfrac{a^2+b^2+c^2-ab-ac-bc}{18}$.

5. Бета-розподіл:

$$\text{Real}=BETA(\text{RNj},\min,\max,\alpha,\beta).$$

Аргументи α,β – додатні дійсні параметри.

Характеристики:

щільність розподілу:

$$f(x)=\frac{(x-\min)^{\alpha-1}(1-(x-\min))^{\beta-1}}{(\max-\min)^{\alpha+\beta-1}B(\alpha,\beta)},\ x\in(\min,\max),$$

де $B(\alpha,\beta)=\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$ – бета-функція;

середнє значення: $\nu=\min+(\max-\min)\dfrac{\alpha}{\alpha+\beta}$;

дисперсія: $D=(\max-\min)^2\dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

Примітки.

При $\alpha=\beta=1$ розподіл вироджується в рівномірний (тобто $BETA(\text{RNj},\min,\max,1,1)=UNIFORM(\text{RNj},\min,\max)$.

При $\alpha=1,\ \beta=2$ розподіл перетворюється в лівий трикутний.

При $\alpha=2,\ \beta=1$ розподіл перетворюється в правий трикутний.

6. Розподіл крайніх значень типу A:

$$\text{Real} = EXTVALA(\text{RNj}, \text{m}, \text{s}).$$

Щільність розподілу: $f(x) = \exp\left(-e^{\frac{x-m}{s}} \right)$.

7. Розподіл крайніх значень типу B:

$$\text{Real} = EXTVALB(\text{RNj}, \text{m}, \text{s}).$$

Щільність розподілу: $f(x) = \exp\left(-e^{-\frac{x-m}{s}} \right)$.

8. Гама-розподіл:

$$\text{Real} = GAMMA(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Характеристики:

щільність розподілу: $f(x) = \dfrac{s^{-\alpha}(x-m)^{\alpha-1}}{\Gamma(\alpha)} e^{-\frac{x-m}{s}}$, $x \geq m$,

де $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$ – гама-функція;

середнє значення: $\nu = \alpha s$;

дисперсія: $D = \alpha s^2$.

Примітки:

при $\alpha = 1$ розподіл зводиться до експоненціального (тобто $GAMMA(\text{RNj}, \text{m}, \text{s}, 1) = EXPONENTIAL(\text{RNj}, \text{m}, \text{s})$);

при цілих додатніх α розподіл зводиться до ерланговського відповідного порядку.

9. Обернений розподіл Гауса:

$$\text{Real} = INVGAUSS(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Щільність розподілу:

$$f(x) = \sqrt{\frac{\alpha}{2\pi(x-m)^2}} \exp\left(-\frac{\alpha(x-m-s)^2}{2s^2(x-m)} \right), \; x \geq m.$$

10. Розподіл Вейбулла:

$$\text{Real} = \mathit{WEIBULL}(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Характеристики:

щільність розподілу: $f(x) = \alpha s^{-\alpha}(x-m)^{\alpha-1}\exp\left(-\left(\frac{x-m}{s}\right)^{\alpha}\right),\ x \geq m;$

середнє значення: $\nu = \frac{s}{\alpha}\Gamma\left(\frac{1}{\alpha}\right) + m;$

дисперсія: $D = \frac{s^2}{\alpha}\left(2\Gamma\left(\frac{2}{\alpha}\right) - \frac{1}{\alpha}\Gamma^2\left(\frac{1}{\alpha}\right)\right).$

Тут $\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1}e^{-t}dt$ – гама-функція.

Примітка. Розподіл Вейбулла при $\alpha = 2$ перетворюється в розподіл Релея, а при $\alpha = 1$ – в експоненціальний розподіл (тобто $\mathit{GAMMA}(\text{RNj}, \text{m}, \text{s}, 1) = \mathit{EXPONENTIAL}(\text{RNj}, \text{m}, \text{s})$.

11. Обернений розподіл Вейбулла:

$$\text{Real} = \mathit{INVWEIBULL}(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Щільність розподілу: $f(x) = \exp\left(-e^{-\frac{\alpha\ln(x-m)}{s}}\right),\ x \geq m.$

12. Розподіл Лапласа:

$$\text{Real} = \mathit{LAPLACE}(\text{RNj}, \text{m}, \text{s}).$$

Характеристики:

щільність розподілу: $f(x) = \frac{1}{2s}\exp\left(-\frac{|x-m|}{s}\right);$

середнє значення: $\nu = m;$

дисперсія: $D = 2s^2.$

13. Логістичний розподіл:

$$\text{Real} = LOGISTIC(\text{RNj}, \text{m}, \text{s}).$$

Характеристики:

щільність розподілу: $f(x) = \dfrac{e^{\frac{x-m}{s}}}{s\left(1+e^{\frac{x-m}{s}}\right)^2}$;

середнє значення: $\nu = m$;

дисперсія: $D = \dfrac{(\pi\beta)^2}{3}$.

14. Логарифмічний розподіл Лапласа:

$$\text{Real} = LOGLAPLACE(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Характеристики:

щільність розподілу: $f(x) = 0.5\exp\left(-\alpha\left|\ln\left(\dfrac{x-m}{s}\right)\right|\right) = 0.5\ln\left|\dfrac{s}{x-m}\right|^{\alpha}$;

середнє значення: $\nu = m$;

дисперсія: $D = \dfrac{(\pi\beta)^2}{3}$.

15. Логарифмічно логістичний розподіл:

$$\text{Real} = LOGLOGIS(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Щільність розподілу: $f(x) = -\dfrac{1}{\exp\left(\alpha\ln\left(\dfrac{x-m}{s}\right)+1\right)}$.

16. Логарифмічно нормальний розподіл:

$$\text{Real} = LOGNORMAL(\text{RNj}, \text{m}, \text{s}, a).$$

Характеристики:
щільність розподілу:

$$f(x) = \frac{1}{(x-m)a\sqrt{2\pi}} \exp\left(-\frac{(\ln(x-m)-s)^2}{2a} \right),\ x \geq m;$$

середнє значення: $\nu = \exp\left(s + \frac{a^2}{2} + m \right)$;

дисперсія: $D = e^{2s+a^2}(e^{a^2} - 1)$.

17. Розподіл Парето:

$$\text{Real} = PARETO(\text{RNj}, \text{m}, \text{s}).$$

Характеристики:

щільність розподілу: $f(x) = \frac{sm^s}{x^{s+1}},\ x \geq m;$

середнє значення: $\nu = \frac{sm}{s-1}$ існує при $s > 1$;

дисперсія: $D = \frac{sm^2}{(s-1)^2(s-2)}$ скінченна при $s > 2$.

18. Розподіл Пірсона типу V:

$$\text{Real} = PEARSON5(\text{RNj}, \text{m}, \text{s}, \alpha).$$

Характеристики:

щільність розподілу: $f(x) = \left(\frac{s}{x-m} \right)^{\alpha+1} \frac{e^{-\frac{s}{x-m}}}{s\Gamma(\alpha)},\ x \geq m,$

де $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$ – гама-функція;

середнє значення: $\nu = \frac{s}{\alpha - 1} + m$ існує при $\alpha > 1$;

дисперсія: $D = \dfrac{s^2}{(\alpha-1)^2(\alpha-2)}$ скінченна, якщо $\alpha > 2$.

19. Розподіл Пірсона типу VI:

$$\text{Real} = PEARSON6(\text{RNj}, \text{m}, \text{s}, \alpha, \beta).$$

Характеристики:

щільність розподілу: $f(x) = \dfrac{\left(\dfrac{x-m}{s}\right)^{\alpha-1}}{sB(\alpha,\beta)\left(1+\dfrac{x-m}{s}\right)^{\alpha+\beta}}$, $x \geq m$,

де $B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$ – бета-функція;

середнє значення: $\nu = \dfrac{s\alpha}{\beta-1} + m$ існує при $\beta > 1$;

дисперсія: $D = \dfrac{s^2\alpha(\alpha+\beta-1)}{(\beta-1)^2(\beta-2)}$ скінченна, якщо $\beta > 2$.

Дискретні розподіли

20. Дискретний рівномірний розподіл:

$$Integer = DUNIFORM(RNj, min, max).$$

Характеристики:

імовірність: $p(x) = \dfrac{1}{\max - \min + 1}$, $x \in \{\min, \min+1, \ldots, \max\}$;

середнє значення: $\nu = \dfrac{min + max}{2}$;

дисперсія: $D = \dfrac{(max - min + 1)^2 - 1}{12}$.

21. Геометричний розподіл:

$$Integer = GEOMETRIC(RNj, p),$$

де p – імовірність успіху при одній спробі.

Характеристики:

імовірність: $p(x) = p(1-p)^x, x = 0,1,\ldots;$

середнє значення: $\nu = \frac{1-p}{p};$

дисперсія: $D = \frac{1-p}{p^2}.$

22. Розподіл Пуассона:

$$Integer = POISSON(RNj, \nu),$$

де ν – середнє значення.

Характеристики:

імовірність: $p(x) = \frac{\nu^x}{x!}e^{-\nu}, x = 0,1,\ldots;$

дисперсія: $D = \nu^2.$

23. Біноміальний розподіл:

$$Integer = BINOMIAL(RNj, t, p),$$

де t – лічильник бернулієвих спроб;

p – імовірність успіху в кожній.

Характеристики:

імовірність: $p(x) = \frac{t!}{x!(t-x)!}p^x(1-p)^{1-x}, x \in \{0,1,\ldots,t\};$

середнє значення: $\nu = tp;$

дисперсія: $D = tp(1-p).$

24. Від'ємний біноміальний розподіл:

$$Integer = NEGBINOM(RNj, c, p),$$

де c – потрібна кількість успішних бернулієвих спроб, p – імовірність успіху у спробі.

Характеристики:

імовірність: $p(x) = \frac{(c+x-1)!}{x!(c-1)!}p^c(1-p)^x, x = 0,1,\ldots;$

середнє значення: $\nu = \frac{c(1-p)}{p}$;

дисперсія: $D = \frac{c(1-p)}{p^2}$.

Примітка. При $c = 1$ розподіл вироджується в геометричний.

Висновки

1. Стандартні числові атрибути забезпечують доступ до характеристик стану системи у процесі моделювання, забезпечують нестандартний збір і обробку статистики. Доступ до СЧА дає можливість керувати процесом руху транзактів.

2. Для отримання випадкових чисел у GPSS World передбачені генератори випадкових чисел.

3. Розробник, використовуючи оператор FUNCTION може задати п’ять типів табличних функцій: неперервну, дискретну і спискову числові, дискретну і спискову атрибутивні.

4. У GPSS World передбачено 24 імовірнісних розподіли: 19 неперервних і 5 дискретних.

Контрольні запитання та завдання

1. Дайте визначення стандартних числових і логічних атрибутів (СЧА). Назвіть їх види. Яку роль вони відіграють у моделях?

2. Для чого в GPSS World використовуються функції? Яким чином записуються функції та де використовуються їх значення?

3. Які імовірнісні розподіли реалізовані в GPSS World? Назвіть оператори їх опису.

Тема 13. Збережувані величини і матриці. Змінні та вирази. Зміна маршрутів транзактів

13.1. Збережувані величини і матриці

Транзакти не можуть безпосередньо посилатися один на одного. Їхнє спілкування реалізується через збережувані величини. Збережувані величини можуть бути скалярними і матричними.

Блоки, які змінюють значення параметрів транзактів

Кожен транзакт може мати будь-яку кількість параметрів з довільною інтерпретацією суті кожного з них. У момент генерування транзакта всі його параметри (які використовують в моделі) мають нульові значення. Для присвоювання параметрам початкових значень або зміни цих значень призначений блок ASSIGN, який має такий формат:

ASSIGN A,B,C

Операнд А задає номер параметра, якому присвоюється значення. Він може бути іменем, додатним цілим числом, виразом у дужках, СЧА, СЧА*параметром. Операнд А може супроводжуватися знаком "+" або "–" (якщо треба збільшити або зменшити значення параметра транзакта, вказаного в операнді А).

Операнд В визначає значення, яке потрібно додати, відняти або яким треба замінити значення в параметрі, заданому операндом А. Якщо такого параметра немає, то він автоматично створюється зі значенням, рівним 0. Операнд В може бути таким самим, як і операнд А, крім того, числом і рядком.

Операнд С (необов'язковий) задає номер або ім'я функції. У випадку використання операнда С значення операнда В множиться на значення функції. Отриманий результат множення стає значенням, яке змінює значення параметра, заданого операндом А. Оскільки операнд С визначає номер функції або її ім'я, то не потрібно використовувати СЧА FN перед ним. Функція в цьому випадку є ***функцією-модифікатором***. Крім того, операнд С може бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Приклади запису блока ASSIGN:

```
ASSIGN    5,23.2
ASSIGN    3+,Q$Cherga
ASSIGN    4–,33.5,7
ASSIGN    Imen,"Klon"
ASSIGN    Chas–,(X5+Q5),Nord
```

У першому прикладі параметру номер 5 присвоюється число 23,2. У другому прикладі до значення параметра 3 додається значення поточної довжини черги з ім'ям Cherga. У третьому прикладі від значення параметра номер 4 віднімається результат множення числа 33,5 на обчислене значення функції номер 7. У четвертому прикладі параметру з

іменем Imen присвоюється рядок Klon. У п’ятому прикладі обчислюються вираз у дужках і функція з іменем Nord, і результат їхнього добутку віднімається від значення параметра з іменем Chas.

У GPSS World значення параметрів транзактів можна також змінювати за допомогою блока PLUS. Блок PLUS обчислює вираз і записує його в параметр. Блок PLUS має такий формат:

PLUS A,B

Операнди:

A – вираз, який може бути ім'ям, числом, рядком, виразом у дужках, СЧА або СЧА*параметром;

B – номер параметра транзакта, в якому зберігається результат. Може бути ім'ям, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Наприклад:

PLUS (P$Result+1.25#Q4),Res

Коли транзакт потрапляє у блок PLUS обчислюється вираз у дужках, записаний у полі операнда A. Потім отриманий результат присвоюється параметру з іменем Res. Якщо такого параметра немає, то він автоматично створюється.

Змінити значення параметра транзакта можна також за допомогою блока INDEX. Формат блоку:

INDEX A,B

Операнди:

A – задає номер параметра транзакта, який може бути ім'ям, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Операнд B – числове значення, яке додається до значення параметра, вказаного операндом A. Може бути іменем, числом, виразом у дужках, СЧА або СЧА*параметром.

Отриманий результат заноситься в перший параметр активного транзакта. Наприклад:

INDEX Dod,(Normal(3,Met,Ser)+X$Ost)

Після потрапляння транзакта у блок INDEX обчислюється вираз у дужках, записаний у полі операнда B, і результат додається до значення параметра з іменем Dod. Отриманий результат присвоюється параметру номер 1 транзакта. Якщо цього параметра не існує, то він автоматично створюється. Якщо немає параметра, зазначеного у полі операнда A, то

відбувається зупинка внаслідок помилки "Звертання до неіснуючого параметра".

Блок MARK використовують для занесення до активного транзакта або його заданого параметра значення абсолютного модельного часу. Цей блок має такий формат:

MARK A

Операнд A – номер параметра транзакта, в який автоматично записується значення абсолютного модельного часу в момент потрапляння цього таранзакта у блок MARK. Може бути ім'ям, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Наприклад:

```
MARK
MARK    Vhid
```

У першому прикладі операнд A не використовується. За замовчуванням транзакту, що увійшов в цей блок, встановлюється поточний час замість часу надходження транзакта до моделі. Цей поточний час використовують для визначення СЧА M1. СЧА M1 обчислюється як різниця значення абсолютного модельного часу (СЧА AC1) і часу, присвоєного транзакту, який дорівнює або часу надходження транзакта до моделі, або часу останнього проходження транзактом блока MARK без операнда A.

У другому прикладі операнд A використовується. Тому значення абсолютного модельного часу заноситься в параметр з іменем Vhid транзакта, який увійшов до цього блока MARK. Якщо цього параметра немає, то він автоматично створюється.

Блок MARK можна використовувати, якщо необхідно визначити час проходження транзакта через певну ділянку моделі. Для цього використовують СЧА MP. Наприклад, якщо на вході ділянки моделі розташувати блок

```
MARK    Marker
```

то на виході цієї ділянки СЧА MP$Marker буде містити різницю між поточним модельним часом і часом, занесеним у параметр Marker блоком MARK.

Зауважимо, що у блоках ASSIGN, PLUS, INDEX і MARK під час запису в поля операндів номера або ім'я параметра транзакта СЧА P не використовується.

У параметр транзакта можна записати результат вибору об’єктів за умовою або кількість об'єктів, що відповідають заданій умові. Для цього використовують блоки SELECT і COUNT.

SELECT X A,B,C,D,E,F

Блок вибирає об'єкт за умовою й поміщає його номер у параметр активного транзакта. Операнди:

X – логічний оператор або показчик відношення;

A – операнд для запису номера обраного елемента;

B – нижня межа діапазону пошуку;

C – верхня межа діапазону пошуку;

D – опорне значення (база порівняння) з операндом E;

E – ім'я класу СЧА, обов'язковий тільки для умовної форми;

F – номер блока призначення, якщо не відібраний жоден об'єкт.

Приклади:

```
SELECT    LE 3,1,10,.70,FR
SELECT    min Minicherga,1,7, ,Q
SELECT    FV TTT,1,5,,,Mit1
```

У першому прикладі вибирається з 10 пристроїв той, у якого коефіцієнт використання менше або дорівнює .70, номер пристрою (тобто результат вибору) записується в 3-й параметр активного транзакта. У другому прикладі вибирається найкоротша черга з семи існуючих і записується її номер у параметр транзакта з ім'ям Minicherga. У третьому прикладі вибирається доступний пристрій з 5-ти можливих і записується його номер у параметр TTT активного транзакта. Якщо доступних пристроїв не виявиться, транзакт направляється до блока з міткою Mit1.

COUNT X A,B,C,D,E

Блок підраховує число об'єктів, що відповідають заданій умові, і записує результат у параметр активного транзакта. Операнди:

X – логічний оператор або покажчик відношення;

A – операнд для прийняття результату;

B – нижня межа діапазону об'єктів, що перевіряються;

C – верхня межа діапазону;

D – опорне значення для операнда E, обов'язкове тільки при наявності умови;

E – специфікатор класу об'єктів (для умовного режиму перевірки).

Приклади:

```
COUNT    LE 3,1,10,.70,FR
COUNT    E Minicherga,1,7,12,Q
COUNT    FV TTT,1,5
```

У першому прикладі в 3-й параметр активного транзакта записується кількість пристроїв з 10 можливих, у яких коефіцієнт використання менше або дорівнює .70. У другому прикладі в параметр з ім'ям Minicherga записується кількість черг (з 7-ми можливих), які містять 12 транзактів. У третьому прикладі підраховується кількість доступних пристроїв з 5-ти можливих і записується результат у параметр TTT активного транзакта.

Комірки зберігання

У GPSS використовують комірки пам’яті, початкові значення яких можуть бути задані перед моделюванням і до яких можна звернутися з будь-якого місця моделі під час моделювання. Ці комірки називають ***комірками зберігання***. Значення комірок зберігання змінюються лише за прямою вказівкою користувача. Перед початком моделювання встановлюють значення комірок зберігання, які дорівнюють нулю. За бажанням користувача окремим коміркам зберігання можуть бути присвоєні ненульові початкові значення за допомогою команди INITIAL. Команду INITIAL застосовують за відсутності активного транзакта або ж тоді, коли не можна використати значення параметрів транзактів. Вона має такий формат:

INITIAL A,B

Операнд А під час ініціалізації комірки зберігання може бути Хдодатне_ціле_число, Х$<ім’я>. Тобто в команді можна відразу зазначати або ім’я або номер збережуваної величини.

Операнд В – початкове значення, яке привласнюється збережуваній величині, або UNSPECIFIED (не визначене). Може бути іменем, числом, рядком або UNSPECIFIED. Якщо операнд В не використовують, то значення комірки дорівнює 1.

Наприклад:

```
INITIAL   X7,123.45
INITIAL   X$Res,Q$Cherga
INITIAL   X$Komirka,V$Myzmin
INITIAL   X$Chas,"Rezult"
INITIAL   X4
```

У першому прикладі в комірку номер 7 заноситься число 123.45, у другому – в комірку з іменем Res записується поточна довжина черги з іменем Cherga, у третьому – в комірку з іменем Komirka – значення змінної користувача Myzmin, у четвертому – в комірку з іменем Chas заноситься рядкова константа Rezult, а в п’ятому прикладі операнд B не використовується, тому комірці зберігання номер 4 присвоюється значення 1.

У процесі моделювання значення комірки зберігання змінюється з входженням транзакта у блок SAVEVALUE (зберегти величину). Формат запису блоку SAVEVALUE:

SAVEVALUE A,B

Операнд A задає номер або ім'я збережуваної величини. Він може бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Операнд A може супроводжуватися знаком "+" або "–" (якщо треба збільшити або зменшити значення збережуваної величини, вказаної в операнді A).

Операнд B визначає значення, яке потрібно додати, відняти або яким треба замінити значення збережуваної величини, заданої операндом A. Якщо такої збережуваної величини немає, то вона автоматично створюється зі значенням, рівним нулю. Операнд B може бути таким самим, як і операнд A, крім того, числом і рядком.

Приклади:

```
SAVEVALUE     6+,X3
SAVEVALUE     Dva–,V$Imc
SAVEVALUE     Tret,–18
SAVEVALUE     4, (3.7#X$Nom+Q6)
SAVEVALUE     15,"Rezult number…"
SAVEVALUE     P9,V$Zmin
```

Значення операндів A змінюються лише із входженням транзактів до цих блоків. У першому прикладі значення комірки зберігання номер 6 з входженням транзакта у блок SAVEVALUE збільшується на значення комірки зберігання з номером 3. У другому прикладі значення комірки зберігання з іменем Dva зменшується на обчислене значення арифметичної змінної Imc. У третьому прикладі значення комірки зберігання з іменем Tret заміщується числом –18. У четвертому прикладі обчислюється вираз у дужках і присвоюється комірці номер 4. У п’ятому вміст комірки номер 15 заміщується рядком. Якщо комірки з таким іменем не

існує, то вона створюється. У шостому прикладі обчислюється значення змінної користувача Zmin, отриманий результат призначається комірці зберігання, номер якої записаний у параметрі номер 9 активного транзакта (одночасно старе значення комірки зберігання знищується).

Комірки зберігання мають єдиний СЧА з назвою X, значенням якого є поточне значення відповідної комірки зберігання. СЧА X в операнді A блока SAVEVALUE, як видно з прикладів, не використовується. Під час запису інших операндів посилання на СЧА X обов'язкове. Посилання на цей СЧА застосовують також у виразах, функціях, в команді INITIAL.

Матриці комірок зберігання

Поняття комірки зберігання в GPSS застосовують і до матриць. Ці об'єкти дають змогу впорядкувати значення, які потрібно зберегти у вигляді матриць. Матрицю необхідно спочатку описати. Команда опису матриці має такий формат:

<ім'я матриці> MATRIX A,B,C,D,E,F,G

Мітка <ім'я матриці> визначає ім'я матриці і повинне бути іменем (не може бути числом). Операнд A не використовують, його залишено для сумісності зі старішими версіями GPSS. Операнди B, C, D, E, F, G можуть бути лише цілими додатними числами.

Операнд B задає кількість рядків матриці – максимальну кількість елементів у першому вимірі, операнд C – кількість стовпців матриці – максимальну кількість елементів у другому вимірі. Операнди D, E, F, G задають максимальну кількість елементів у третьому, четвертому, п'ятому і шостому вимірах відповідно. Наприклад, команда

```
Matr    MATRIX   ,5,7
```

визначає матрицю з іменем Matr, яка містить 5 рядків і 7 стовпців.

Матриці мають єдиний СЧА з назвою MX, за допомогою якого можна звертатися до будь-якого елемента матриці. Значеннями рядків і стовпців можуть бути імена, цілі числа і параметри транзактів. Приклади:

```
MX1(2,5)
MX$Tab(P3,P$Stovp)
MX$Tab(V$Ser,P4)
```

У першому прикладі визначається елемент матриці номер 1, який перебуває на перетині другого рядка і п'ятого стовпця. У другому прикладі для визначення елемента матриці з іменем Tab використовуються значення параметра номер 3 і параметра з іменем Stovp тран-

закта, а у третьому – ім’я змінної користувача з іменем Ser і параметр номер 4 транзакта.

Зауважимо, що у першому прикладі матриця має не ім’я, а номер, хоч у команді визначення матриці, як було зазначено вище, на місці мітки <ім'я матриці> можна використовувати лише ім’я. Для присвоєння матриці номера замість імені потрібно для запису команди визначення матриці використати команду EQU. Наприклад:

```
Rozr    EQU     7
Rozr    MATRIX  ,3,5
```

У цьому прикладі ім’я Rozr матриці замінене на номер 7, за яким (MX7) потрібно звертатися до матриці. Якщо в команді MATRIX замість імені Rozr відразу записати число 7, то під час трансляції виникне зупинка із зазначенням помилки "Пропущена мітка оператора".

Перед початком моделювання або виконання команди CLEAR ON значення всіх елементів матриці мають дорівнювати нулю. Деяким або всім елементам за бажанням розробника можна присвоїти значення, які не дорівнюють нулю, або вони можуть бути переведені в невизначений стан командою INITIAL. Наприклад:

```
INITIAL   MX8(2,7),15.3
INITIAL   MX$Matr(3,8),25
INITIAL   MX$Mac(4,5),Kontr
INITIAL   Petit,3
INITIAL   Zsuv
INITIAL   Mist,UNCPECIFIED
```

У першому прикладі елементу матриці номер 8, який розташований на перетині другого рядка і сьомого стовпця, присвоюється початкове значення 15.3, у другому – елементу (3,8) матриці з іменем Matr – значення 25, у третьому – елементу (4,5) матриці з іменем Mac – значення змінної користувача Kontr. У четвертому прикладі замість операнда зазначено ім’я матриці Petit, тому всім елементам цієї матриці присвоюється значення 3. У п’ятому прикладі також замість операнда A стоїть ім’я матриці Zsuv, але операнд B не використовується, тому всім елементам матриці присвоюється значення 1. У шостому прикладі замість операнда B стоїть ключове слово UNCPECIFIED, тому всі елементи раніше створеної матриці Mist будуть переведені в невизначений стан. Цим переведенням матриця готується для використання в експе-

рименті, в якому можуть бути пропущені, тобто не отримані під час експерименту дані. Невизначений стан елемента матриці засвідчує те, що даних немає. Якщо ж елементи матриці були б переведені в нуль, то це б сприймалось як отримання даних експерименту, які дорівнюють нулю, що не відповідає дійсності.

Після експерименту матриця з його результатами передається бібліотечній процедурі дисперсійного аналізу (ANOVA), яка опрацьовує елементи, що перебувають у стані UNCPECIFIED, як пропущені дані замість нульових значень.

Оператор MATRIX створює матрицю в поточній моделі. Матриця не може бути видалена з поточної моделі. Якщо оператор MATRIX вилучається з моделі, яка є у процесі виконання, то зв'язок з матрицею в цій моделі залишається. Матрицю можна перевизначити повторно іншим оператором MATRIX з тією ж міткою. Перед використанням матриця має бути визначена в операторі MATRIX. Матриця обмежена максимальним обсягом пам'яті, встановленим у журналі налаштувань моделі.

Для запису у процесі моделювання значень у матриці, а також для збільшення або зменшення значень, записаних у матриці, призначений блок MSAVEVALUE. Формат його запису такий:

MSAVEVALUE A,B,C,D

В операнді A задають ім'я або номер матриці. Крайнім правим символом операнда може бути знак + (режим накопичення) або знак – (режим віднімання). В операнді B задають номер рядка, в операнді C – номер стовпця матриці. Операнд D визначає значення, яке має зберігатися, додаватися або відніматися. Усі операнди є обов'язковими. Вони можуть бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Наприклад:

```
MSAVEVALUE   4,5,P7,321
MSAVEVALUE   X$Matric–,3,7,V$Serv
MSAVEVALUE   Matr+,4,(Myst#5–3),M1
MSAVEVALUE   (Kra+Q$Cherga)+,(Roz–7),4,P$Odn
MSAVEVALUE   3,Riad,Stovp,"Dovgyna"
```

У першому прикладі з входженням транзакта до блоку MSAVEVALUE значення елемента матриці номер 4, розташованого на перетині рядка 5 і стовпця, номер якого міститься в параметрі 7 цього транзакта, замінюється числом 321. У другому прикладі значення елемента на перетині

рядка 3 і стовпця 7 матриці, номер якої міститься у комірці зберігання Matric, зменшується на обчислене значення арифметичної змінної Serv.

У третьому прикладі номер стовпця знаходять як результат обчислення виразу в дужках (Myst#5–3), а потім значення визначеного елемента матриці Matr збільшується на величину відносного часу M1 перебування транзакта в моделі.

У четвертому прикладі номер матриці і номер рядка елемента визначають як результати обчислення виразів у дужках (Kra+Q$Cherga) і (Roz–7) відповідно. Після цього значення знайденого елемента збільшується на величину, яка міститься в параметрі з іменем Odn активного транзакта.

У п’ятому прикладі елементові матриці номер 3, розташованому на перетині рядка і стовпця, які визначають змінними користувача Riad і Stovp, присвоюється рядок "Dovgyna". Якщо змінним користувача раніше командою EQU не будуть присвоєні відповідні значення, то виникне зупинка внаслідок помилки.

Зупинка внаслідок помилки виникає також в тому випадку, коли із входом транзакта до блоку MSAVEVALUE не буде знайдена матриця з потрібним іменем або номером, або не буде знайдений елемент з відповідними номерами рядка та стовпця.

СЧА MX у блоці MSAVEVALUE при записі операнда A, як і СЧА X у блоці SAVEVALUE при записі операнда A, не використовується. Посилання на СЧА MX застосовується у виразах, функціях, операторі INITIAL.

З розглянутого формату команди опису матриці випливає, що матриця GPSS може мати до шести вимірів. Однак, як видно з формату блоку MSAVEVALUE, він дає змогу використовувати лише двовимірні матриці, тобто в ньому доступні лише перші два виміри. Решта індексів дорівнюють одиниці.

Якщо є необхідність використання матриць більшої розмірності, то потрібно створити одну або кілька PLUS-процедур. PLUS-процедури мають доступ до всіх елементів всіх матриць. Матриці, визначені командою MATRIX, є глобальними і доступні всім PLUS-процедурам. На час виконання PLUS-процедури можуть бути створені тимчасові матриці з локальною областю видимості.

LOGIC X A

Блок призначений для вмикання, вимикання або інвертування стану логічного ключа. Якщо логічний оператор X дорівнює S або R, то ключ A

встановлюється у стан "увімкнений" (Set) або "вимкнений" (Reset) відповідно. Якщо логічний оператор – I, то стан ключа інвертується. Блок LOGIC у режимі інвертування дозволяє чергувати альтернативи різного роду – маршрути, функції, числові значення й т. п.

13.2. Змінні та вирази

Алфавіт мови GPSS складається з латинських букв, цифр і спеціальних символів. Великі й малі букви розрізняються тільки в рядкових константах і коментарях. Рекомендується службові слова набирати великими літерами, а індивідуальні імена об'єктів – малими, починаючи із великої.

Арифметичні, умовні та логічні оператори

Обчислювальні вирази є комбінацією математичних операторів, бібліотечних функцій, СЧА і констант, що задовольняють правилам елементарної алгебри. Вони обчислюються згідно з ієрархією операторів в напрямку зліва направо. Порядок обчислень можна змінити за допомогою дужок.

Для позначення операцій, які виконують над елементами виразу, призначені арифметичні, умовні і логічні оператори. У GPSS World використовують такі оператори (у порядку убування пріоритету):

^ – піднесення до степеня;

– множення; / – ділення; \ – ділення націло;

@ – цілочисельний залишок;

+,– – додавання й віднімання;

>=, 'GE' – більше або дорівнює; <=, 'LE' – менше або дорівнює; >, 'G' – більше; <, 'L' – менше; =, 'E' – дорівнює;

!=, 'NE' – не дорівнює;

&, 'AND' – логічне множення;

|, 'OR' – логічне додавання;

'NOT' – заперечення.

Якщо оператор потребує певного типу даних, то дані перетворюються автоматично. Наприклад, якщо над рядковою величиною виконується числова операція, то буде використаний числовий еквівалент цього рядка символів.

Оператор "#" використано для операції множення для того, щоб була можливість застосовувати оператор "*" для позначення непрямої

адресації. Проте користувач може змінювати значення операторів "#" і "*" (прапорець Switch * and #, в меню Edit – Settings – Simulation).

З математичних функцій дозволені ABS (абсолютне значення), ATN (арктангенс), COS (косинус), SIN (синус), TAN (тангенс), EXP (експонента), LOG (логарифм натуральний), SQR (корінь), INT (виділення цілої частини). Аргументи математичних функцій повинні бути взяті в дужки. Усі кути задаються в радіанній мірі.

Аргументи бібліотечних математичних функцій у всіх випадках автоматично перетворюються у числові значення. Допустимі значення аргументів – вираз. Обчислювані числові значення функцій – дійсного типу.

Змінні користувача

GPSS надає користувачеві можливість мати свої змінні для зберігання потрібних під час моделювання числових і рядкових даних. Змінні користувача створюються за допомогою команди EQU. Команда EQU створює іменований об'єкт (змінну користувача) і привласнює йому обумовлене виразом числове значення:

<ім'я змінної> EQU <вираз>

Приклади:

```
Myzminna  EQU   0.45
Rjad      EQU   15
Stovp     EQU   9
Name      EQU   "Name"
```

У наведених перших трьох прикладах змінним користувача Myzminna, Rjad, Stovp присвоюються числові значення, а змінній Name – рядок.

За допомогою змінних користувача імена об’єктів GPSS можна замінювати номерами. Для цього необхідно на початку моделі визначити змінні користувача командою EQU.

Імена, задані командою EQU, які в моделі використовують як мітки об’єктів, вважають мітками.

Змінні користувача можна застосовувати в PLUS-виразах, змінювати командами EQU або присвоюючими операторами в PLUS-процедурах. Якщо під час виконання експерименту з імітаційною моделлю передбачається дослідження впливу деяких чинників на шуканий показник, то чинники обов’язково мають бути змінними користувача.

Арифметичні змінні й арифметичні вирази

Арифметичні змінні дозволяють обчислювати арифметичні вирази, що складаються із стандартних числових атрибутів (СЧА). У виразі змінної використовують оператори, арифметичні дії і виклики бібліотечних функцій. Арифметична змінна є СЧА, який визначає користувач.

Якщо моделювання виконується не в режимі сумісності з GPSS/PC, то арифметична змінна в GPSS World може бути визначена командою VARIABLE або командою FVARIABLE, оскільки дія цих команд, а отже, й обчислювані значення, однакові. Формати команд:

<ім'я змінної> VARIABLE A

<ім'я змінної> FVARIABLE A

Операнд A – це вираз, який задає користувач.

Величиною арифметичної змінної є величина заданого користувачем арифметичного виразу. Значення арифметичної змінної можна використовувати як:

операнд; у цьому випадку значення арифметичної змінної може вказувати: номер об’єкта j; j логічного атрибута (блок GATE); номер параметра транзакта (блоки ASSIGN, INDEX, LOOP, MARK, SPLIT);

значення атрибута;

аргумент функції;

значення залежної змінної атрибутивної функції;

аргумент таблиці;

операнд іншої арифметичної або булевої змінної.

Приклади:

```
Bakir   VARIABLE   R$Mybkp+S$Mybkp
Kard    VARIABLE   Q$Cherga–Q$Bula
Dart    VARIABLE   P7+Q$Mycher/3
Lasso   VARIABLE   N$Tonic@12
```

У першому прикладі змінна Bakir є величиною, яка дорівнює сумі вільної ємності і поточного вмісту БКП Mybkp. Цей приклад показує, що розробник може задавати арифметичну змінну, величина якої є ємністю пристрою. У другому прикладі змінну Kard визначають як поточну довжину черги Cherga мінус поточна довжина черги Bula. У третьому прикладі змінну Dart обчислюють спочатку діленням поточної довжини черги Mycher на 3, а потім додаванням результату до значення параметра 7 активного транзакта. У четвертому прикладі Lasso – цілий залишок від ділення на 12 кількості входжень до блоку з іменем Tonic.

Значення арифметичного виразу обчислюється, коли активний транзакт потрапляє в блок, оператор якого серед своїх операндів містить посилання на арифметичні змінні. Підраховані значення є дійсними числами.

У GPSS World проміжні результати і значення СЧА не округлюються. Округлення може задавати лише сам користувач. Для цього можна використати операцію ділення націло, або бібліотечну функцію INT(A).

Наприклад:

```
Zminna1   VARIABLE    MX$Matr(P$Str,P$Sto)\Kor
Zminna1   VARIABLE    INT(MX$Matr(P$Str,P$Sto)/Kor)
```

Якщо процес моделювання виконується в режимі сумісності з GPSS/PC, то обчислення одного і того ж арифметичного виразу, заданого командами VARIABLE і FVARIABLE, здійснюється по-різному, оскільки в GPSS/PC значення всіх СЧА є цілого типу.

У режимі сумісності під час опису арифметичної змінної командою VARIABLE всі проміжні обчислення і кінцевий результат округлюються до цілого значення. Дійсні змінні (з плаваючою крапкою) визначає команда FVARIABLE. У цьому випадку округлюється до цілого значення лише кінцевий результат, тобто дійсні змінні є також цілочисельні.

Арифметична змінна і змінна з плаваючою крапкою в режимі сумісності з GPSS/PC не можуть мати однакових імен. Якщо вони мають однакові імена, то під час обчислення використовують останній з цих двох описів. Це відбувається тому, що СЧА V використовують як для звертання до арифметичних змінних, так і до змінних з плаваючою крапкою. Спосіб обчислення змінної V є оператором опису цієї змінної.

Розходження між результатами, отриманими під час обчислення змінних з плаваючою і фіксованою крапками, можна побачити з наведених нижче прикладів:

```
Obch1     FVARIABLE    10#(11/4)
Obch2     VARIABLE 10#(11/4)
```

Значення змінної Obch1 буде дорівнювати 27, оскільки число 10 потрібно помножити на 2.75 і від результату 27.5 взяти цілу частину. Змінна Obch2 дорівнює 20, оскільки результат проміжної операції 2.75 буде округлений до 2.

Оскільки GPSS World надає можливість використовувати вирази в операндах, то доцільно для спрощення моделей вираз, який неодноразово використовують у різних частинах моделі, визначити як змінну лише один раз, а потім посилатися на нього.

Булеві змінні

Булеві змінні дають можливість приймати рішення залежно від стану і значення багатьох об'єктів GPSS, використовуючи для цього лише один блок. У булевій змінній перевіряється одна або декілька логічних умов. Результатом перевірки є 1, якщо задані умови задовольняються, і 0, якщо вони не задовольняються. Булевими змінними є логічні вирази, складені з різних стандартних числових атрибутів, у тому числі й інших булевих змінних.

Булеву змінну визначають командою BVARIABLE. Формат команди:

<ім'я> BVARIABLE A

Операнд А – логічний вираз. У виразі можна використовувати три типи операторів: логічні, булеві й оператори відношення. Кінцевий результат перетворюється в ціле значення 0, якщо дорівнює нулю, або в ціле значення 1, якщо відмінний від нуля.

Логічні оператори пов'язані з об'єктами апаратної категорії і використовуються для визначення стану цих об'єктів. Існують такі логічні оператори:

FVj дорівнює 1, якщо пристрій j доступний, інакше – 0;

FIj дорівнює 1, якщо пристрій j обслуговує переривання, інакше – 0;

SFj дорівнює 1, якщо БКП j заповнений повністю, інакше – 0;

SEj дорівнює 1, якщо БКП j порожній, інакше – 0;

SVj дорівнює 1, якщо БКП j доступний, інакше – 0;

LSj дорівнює 1, якщо логічний ключ j увімкнений, інакше – 0.

Під j розуміють номер або ім’я об’єкта.

Приклади:

```
Stan1    BVARIABLE    FV$Apparat1
Stan2    BVARIABLE    SF$Apparat2
Stan3    BVARIABLE    SV$Apparat3
Stan4    BVARIABLE    LS2
```

У першому прикладі булева змінна Stan1 дорівнює 1, якщо пристрій Apparat1 доступний, і – 0, якщо недоступний. У другому прикладі булева змінна Stan2 дорівнює 1, якщо БКП Apparat2 повністю заповнений. У

третьому прикладі булева змінна Stan3 дорівнює 1, якщо БКП Apparat3 доступний. У четвертому прикладі булева змінна Stan4 дорівнює 1, якщо логічний ключ номер 2 увімкнений.

Оператори відношення здійснюють алгебраїчне порівняння операндів. Наприклад:

```
Vidn1    BVARIABLE    V$Myzmin'G' 5
Vidn2    BVARIABLE    Q7'LE'P$Porjadok
Vidn3    BVARIABLE    X$Matr'GE'P8
```

Булева змінна Vidn1 дорівнює 1, якщо змінна Myzmin більше 5, інакше дорівнює 0. У другому прикладі булева змінна Vidn2 дорівнює 1, якщо поточна довжина черги номер 7 не первищує параметра транзакта з ім'ям Porjadok. У третьому прикладі булева змінна Vidn3 дорівнює 1, якщо значення збережуваної величини Matr не менше ніж значення параметра 8.

Булевих операторів є два: OR – АБО і AND – І. Оператор АБО перевіряє виконання хоча б однієї з умов, а оператор І потребує виконання обох умов. Наприклад:

```
Poriv1    BVARIABLE    FI$Prib'OR'SF8
Poriv2    BVARIABLE    FI$Prib'AND'SF$Gos
Poriv3    BVARIABLE    (V2'G'5)'AND'(FV$Prib1'OR'FV$Prib2)
```

Булева змінна Poriv1 дорівнює 1, якщо виконується одна з умов: пристрій Prib обслуговує переривання або БКП номер 8 незаповнений. Булева змінна Poriv2 дорівнює 1, якщо виконуються обидві умови: пристрій Prib обслуговує переривання і БКП з іменем Gos незаповнений. У третьому прикладі змінна Poriv3 дорівнює 1, якщо виконуються обидві умови: значення змінної номер 2 більше ніж 5 і доступний один з пристроїв Prib1 або Prib2.

Дужки у третьому прикладі потрібні лише для задання певних булевих співвідношень. Дужки треба використовувати лише в тих випадках, коли вони необхідні.

Якщо булева змінна задається СЧА, як наприклад:

```
Myzmin    BVARIABLE    V$Acc
```

то обчислюється значення арифметичної змінної на ім'я Acc, і якщо воно відмінне від нуля, то значення булевої змінної Myzmin буде дорівнювати 1, інакше – 0.

13.3. Зміна маршрутів транзактів

Блок GATE (впустити) в GPSS World використовується для визначення стану пристроїв, БКП і логічних ключів без зміни цього стану. Також блок GATE дозволяє змінювати маршрут руху транзакта за станом об'єкта, який перевіряється. Стан завжди має логічне значення. Формат блока:

GATE X A,B

Операнди:

X – вид умови, що перевіряється. При виконанні умови активний транзакт переходить до наступного блока;

A – ім'я або номер об'єкта, який перевіряється. Тип елемента повинен бути погоджений з умовою;

B – номер або ім'я блока, до якого направляється транзакт при невиконанні умови.

Умовний оператор X неявно задає тип об’єкта, який перевіряється. Допустимі значення:

для пристроїв: **U / NU** (зайнятий / вільний), **I / NI** (обслуговує переривання / неперерваний), **FV / FNV** (доступний / недоступний);

для БКП: **SE / SNE** (порожній / непорожній), **SF / SNF** (заповнений / незаповнений), **SV / SNV** (доступний / недоступний);

для логічних ключів: **LS / LR** (включений / виключений);

для транзактів: **M** – у блоці, заданому в полі A, у стані синхронізації перебуває транзакт, який належить до тієї самої сім'ї, що й транзакт, який перебуває у блоці GATE або робить спробу увійти до нього; **NM** – у блоці, заданому в полі A, у стані синхронізації немає жодного транзакта, який належить до тієї самої сім'ї, що й транзакт, який робить спробу увійти до блока GATE.

Операнди A і B можуть бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Блок GATE працює у двох режимах:

відмови входження до блока;

дозволу входження та альтернативного виходу.

Якщо операнд B не використовується, GATE діє в режимі відмови. У випадку невиконання заданої умови транзакт переводиться до списку повторних спроб тестованого об’єкта. Коли стан об’єкта змінюється, заблокований транзакт знову активізується і повторюється перевірка

заданої блоком GATE умови. Якщо ця умова виконується, то транзакту дозволяється увійти до блока GATE і далі перейти до наступного за порядком блока.

У поєднанні з БКП GATE може використовуватися для формування пачок постійного обсягу, що дорівнює ємності БКП.

Для розгалуження маршруту на два напрямки використовуються двопозиційні логічні ключі: замок є/немає, червоний або зелений сигнал світлофора, табличка "Місць немає", "Перерва". У випадку триколірного світлофора використовують блок TEST і числову змінну (VARIABLE).

Приклади:

GATE NU Prystriy

GATE NU Pribor1,Mit

GATE SNF Pribor2,Mitka

У першому прикладі блок GATE не пропустить транзакта за умови зайнятості пристрою з іменем Prystriy. У другому прикладі у випадку зайнятості пристрою з іменем Pribor1 транзакт направляється до блока з міткою Mit. У третьому прикладі, якщо БКП з іменем Pribor2 незаповнений, тобто є вільні канали, то задана у блоці GATE умова виконується і транзакт буде направлений до наступного блока. Якщо БКП буде заповнений, то транзакт направляється до блока з міткою Mitka.

Блок TEST (перевірити) порівнює значення зазначених величин і управляє напрямком руху транзакта, ґрунтуючись на результаті порівняння.

TEST X A,B,C

Операнди:

X – вид умови, що перевіряється;

A, B – порівнювані величини;

C – номер або ім'я блока, до якого направляється транзакт при невиконанні умови.

Умовний оператор X може бути вираженим одним з шести умовних операторів: L – менше; LE – менше або дорівнює; E – дорівнює; NE – не дорівнює; G – більше; GE – більше або дорівнює.

Операнди A і B можуть бути іменем, числом, рядком, виразом у дужках, СЧА або СЧА*параметром.

Оскільки логічні значення в GPSS/W кодуються нулем й одиницею, у блоці TEST можна зіставляти й логічні значення.

Якщо операнд C зазначений, то транзакт завжди може увійти в блок TEST і залежно від співвідношення значень операндів A та B буде переданий або до наступного блока, або до блока, вказаного операндом C. Якщо поле операнда C порожнє, то транзакт у випадку невиконання умови не зможе увійти до блока TEST. Він переводиться до списку повторних спроб всіх об'єктів, які беруть участь у перевірці умови. Коли стан будь-якого з цих об'єктів змінюється, то транзакт зі списку повторних спроб активізується і перевірка здійснюється знову. Якщо умова виконується, то транзакт отримує дозвіл на вхід у блок TEST.

Блок TEST та булеві змінні можна використовувати для перевірки стану апаратних об'єктів.

У булевій змінній допускається використання таких логічних операторів:

F дорівнює 1, якщо пристрій зайнятий, інакше – 0;

SF дорівнює 1, якщо БКП заповнений повністю, інакше – 0;

SE дорівнює 1, якщо БКП порожній, інакше – 0;

SV дорівнює 1, якщо БКП доступний, інакше – 0,

а також СЧА пристроїв та БКП:

FC – кількість транзактів, які займали пристрій за допомогою блоків SEIZE і PREEMPT;

FR – коефіцієнт використання пристрою;

FT – середній час обслуговування пристроєм одного транзакта;

S – кількість зайнятих каналів БКП;

SA – середнє значення кількості зайнятих каналів БКП;

SC – лічильник використання БКП;

SR – коефіцієнт використання БКП;

ST – середній час використання одного каналу БКП.

Отже, СЧА SE, SF і SV можна використовувати як у блоці GATE, так і в булевих змінних блока TEST.

Приклад використання булевої змінної та блока TEST для перевірки стану пристрою:

```
Bul   BVARIABLE      F$Server
TEST      E BV$Bul,0,Mitka
```

Послідовність проходження транзактами блоків у моделі також можна змінити за допомогою блоків TRANSFER, DISPLACE та LOOP.

Блок TRANSFER (передати) змінює траєкторію руху активного транзакта. Формат блока:

TRANSFER A,B,C,D

Операнди:

A – режим блока, обирається з множини {BOTH, ALL, PICK, FN, P, SBR, SIM}, може бути також дробовим числом, ім'ям, константою, СЧА;

B – номер або ім'я блока (у режимі P – ім'я параметра);

C – номер або ім'я блока (у режимі FN або P – збільшення);

D – збільшення номера блока для режиму ALL. За замовчуванням дорівнює 1.

Усі режими блока TRANSFER, крім безумовного, вибіркові, тобто відрізняються один від одного способом вибору чергового блока, до якого має бути направлений активний транзакт. Операнд A задає цей режим вибору. Блок TRANSFER може працювати в дев'яти режимах:

, (за замовчуванням) – безумовний;

. – статистичний, тобто вибір випадковим чином одного з двох блоків;

BOTH – послідовний вибір одного з двох блоків;

ALL – послідовний вибір одного з кількох блоків;

PICK – вибір випадковим чином одного з кількох блоків;

FN – функціональний;

P – параметричний;

SBR – підпрограмний;

SIM – одночасний.

Операнд A, крім перерахованих значень, також може бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Операнди B і C визначають можливі імена наступних блоків або їхнє розташування. Вони також можуть бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Використання цих значень розглянемо під час ознайомлення з конкретними режимами роботи. Якщо операнд B відсутній, то планувальник записує замість нього ім'я або номер блока, який розташований після блока TRANSFER.

Розглянемо режими роботи блока TRANSFER.

Режим безумовної передачі. У цьому режимі операнд A не використовують, а операнд B зазначає ім'я блока, до якого направляється транзакт. Блок TRANSFER завжди відкритий для входження транзактів. Наприклад:

TRANSFER ,Mit1

Потрапивши у блок TRANSFER, транзакт відразу намагається увійти у блок з міткою Mit1. Якщо транзакт отримує відмову на входження, то він залишається у блоці TRANSFER.

Статистичний режим. Значення операнда A визначає імовірність, з якою транзакт направляється до блока, зазначеного в полі операнда C. З імовірністю 1–A транзакт направляється до блока, вказаного в полі операнда B (до наступного блока, якщо поле B порожнє).

Імовірність у полі A може бути задана безпосередньо десятковим дробом, який починається з крапки. Наприклад блок

TRANSFER .75,Tut,Tam

з імовірністю 0,75 направляє транзакти до блока з іменем Tam, а з імовірністю 0,25 – до блока з іменем Tut.

Якщо поле A починається не з десяткового дробу і не містить одного з ключових слів – ознак інших режимів роботи, то його значення розглядається як кількість тисячних часток імовірності передачі. Наприклад, попередній блок TRANSFER можна записати також у такому вигляді:

TRANSFER 750,Tut,Tam

Числове значення операнда A можна задавати будь-яким СЧА. Можливі такі випадки:

1) значення операнда A менше або дорівнює нулю;
2) значення операнда A дорівнює або більше ніж 1000;
3) значення операнда A більше від нуля, але менше ніж 1000.

Якщо підраховане значення операнда A менше або дорівнює нулю, то буде здійснюватися безумовне направлення транзакта до блока B. Якщо значення операнда A більше або дорівнює 1000, то буде здійснюватися безумовна передача транзакта до блока C. У третьому випадку блок TRANSFER працює у звичайному режимі.

Наприклад, запис

TRANSFER P5,,Tam

означає, що тризначне число, записане в параметрі 5 транзакта, який потрапив у блок TRANSFER, інтерпретується як імовірність (в частках тисячі) того, що транзакт передаватиметься до блока Tam, а в решті випадків – до наступного блока, оскільки операнд B не використовується.

Режим статистичної передачі зручно використовувати для розгалуження маршруту руху транзактів на два напрямки.

Можна визначити генератор – джерело випадкових чисел для блока TRANSFER. Для цього, вибравши в меню Edit – Settings закладку Random Numbers, треба ввести номер генератора в полі блока TRANSFER.

Режим BOTH. Якщо в полі операнда A використовується ключове слово BOTH (обидва), то блок TRANSFER працює в режимі BOTH. У цьому режимі кожний транзакт, увійшовши в блок TRANSFER, перевіряє дві можливості. Спочатку він намагається увійти до блока B (або до наступного блока, якщо поле B порожнє), а якщо це не вдається, тобто блок B не дозволяє входження, то – до блока C. Якщо й ця спроба невдала, то транзакт затримується у блоці TRANSFER до такої зміни умов у моделі, які б дозволили увійти до одного з блоків B або C. Блоку B надається перевага, якщо така можливість виникне одночасно.

Приклад:

```
……
TRANSFER BOTH,,Mitka
SEIZE Prist1
ADVANCE 15,5
RELEASE Prist1
……
Mitka SEIZE Prist2
ADVANCE 20
RELEASE Prist2
……
```

Транзакти потрапляють у блок TRANSFER, який працює в режимі BOTH. Якщо в момент надходження транзакта пристрій Prist1 може його прийняти, то блок TRANSFER направить транзакт до наступного блока, тобто до блока SEIZE. Якщо ж у момент надходження транзакта пристрій не може його прийняти (пристрій зайнятий, обслуговує переривання чи перебуває у стані неготовності), то транзакт буде направлений до блока SEIZE з іменем Mitka. Якщо жоден блок не зможе прийняти транзакт, то він залишається у блоці TRANSFER доти, поки якийсь пристрій не прийме його.

При моделюванні замість блока TRANSFER, який працює в режимі BOTH, можна використати також блок GATE або TEST, замінивши рядок програми

```
TRANSFER BOTH,,Mitka
```

на такий:

```
GATE NU Prist1,Mitka
```

або на такий:

```
TEST E F$Prist1,0,Mitka
```

У цьому варіанті моделі транзакт проходить до блока SEIZE, якщо пристрій вільний (перевірка проводиться відповідно блоком GATE або TEST).

Режим ALL. Блок TRANSFER у режимі ALL, як і в режимі BOTH, виконує роль "диспетчера", але на відміну від блока BOTH, він розподіляє транзакти між кількома пристроями.

Для роботи в режимі ALL необхідно в полі операнда A використати ключове слово ALL. У цьому режимі, увійшовши у блок TRANSFER, транзакт перевіряє можливість входження до будь-якого блока, починаючи з блока, зазначеного операндом B, і закінчуючи блоком, вказаним операндом C. Операнд D визначає крок d зміни номера блока, який перевіряється. Значення кроку дає змогу опитувати певні блоки, розташовані між тими, які задані операндами B і C.

Спочатку транзакт, увійшовши у блок TRANSFER, намагається увійти до блока B. Якщо блок B зайнятий, то транзакт намагається увійти до блоків з номерами v+d, v+2d, ..., x=v+nd, де v – номер блока B, x – номер блока C, n – деяке натуральне число.

Якщо операнд C не використовують, то перевіряється лише один блок. Оскільки, зазвичай, у полях операндів B і C записують мітки блоків, то блоки потрібно розташовувати так, щоб під час присвоєння номера різниця між номерами блоків, зазначених операндами B і C, була кратною крокові, вказаному в полі операнда D. Наприклад:

```
TRANSFER ALL,Mit1,Mit2,4
```

Тут режим ALL допустимий, якщо різниця між номерами, присвоєними блокам Mit1 і Mit2, кратна 4. У моделі це може виглядати так.

Приклад. Замовлення розподіляють між трьома пристроями. Чергове замовлення обслуговується тим пристроєм, який першим звільнився.

```
      GENERATE 10,5
      TRANSFER ALL,Mit1,Mit2,4
;Імітування роботи пристрою 1
Mit1  SEIZE Prib1
```

```
        ADVANCE 10
        RELEASE Prib1
        TERMINATE
;Імітування роботи пристрою 2
        SEIZE Prib2
        ADVANCE 15
        RELEASE Prib2
        TERMINATE
;Імітування роботи пристрою 3
Mit2    SEIZE Prib3
        ADVANCE 20,10
        RELEASE Prib3
        TERMINATE
;Таймер
        GENERATE 480
        TERMINATE 1
```

Після надходження транзакта у блок TRANSFER послідовно перевіряється можливість входження спочатку до блока SEIZE, позначеного міткою Mit1. Потім, якщо цей блок не дозволяє входження, до другого блока SEIZE без мітки, оскільки різниця між номерами цього блока та блока з міткою Mit1 дорівнює 4. Якщо входження до другого блока SEIZE неможливе, то перевіряється третій блок SEIZE з міткою Mit2. Якщо й він зайнятий, то транзакт залишається у блоці TRANSFER, і з кожною зміною поточного модельного часу перевірки можливості входження знову повторюються спочатку, тобто з блока SEIZE з міткою Mit1.

У блоці TRANSFER операнди B і C можуть бути не мітками, а номерами блоків. Зокрема, якщо в моделі попереднього прикладу записати

```
        TRANSFER    ALL,3,11,4
```

то результат моделювання буде такий самий (нумерація починається з блока GENERATE).

Зауважимо, що у режимі ALL (як і у режимі BOTH) у випадку, коли можливий перехід до декількох блоків, то блоки з меншими номерами мають деяку перевагу перед блоками з більшими номерами.

Режим PICK вибирає подальше призначення транзакта в інтервалі від B до C (послідовні номери) з рівними ймовірностями незалежно від можливого блокування.

Для роботи блока TRANSFER у режимі PICK необхідно в полі операнда A використати ключове слово PICK. У цьому режимі з послідовності блоків з номерами v, v+1, v+2, ..., n, де v – номер блока, зазначеного операндом B, а n – номер блока, вказаного операндом C, випадково вибирається номер одного блока, до якого має перейти транзакт. Усі блоки, враховуючи й вказані операндами B і C, вибираються з однаковою ймовірністю 1/(n–v+1). Транзакт намагається перейти лише до вибраного для нього блока. Якщо транзакт не може відразу перейти до наступного блока, то він чекатиме у блоці TRANSFER до того часу, поки не буде знята умова блокування. Номер блока C має бути більший або дорівнювати v+1.

Приклад. Замовлення розподіляються між чотирма пристроями, які мають однаковий час обслуговування. Ймовірність потрапляння до будь-якого з пристроїв P=1/4=0,25. Якщо в режимах BOTH і ALL чергове замовлення займало той пристрій, який раніше звільнявся, то в режимі PICK замовлення чекає на звільнення того пристрою, на який воно було розподілене. Оскільки блоки, до яких передаються замовлення, в моделі розташовані не послідовно, то використовується послідовність блоків TRANSFER у режимі безумовної передачі.

```
        GENERATE 10,5
        TRANSFER PICK,Mit1,Mit2
Mit1    TRANSFER ,Mit3
        TRANSFER ,Mit4
        TRANSFER ,Mit5
Mit2    TRANSFER ,Mit6
;Імітування роботи пристрою 1
Mit3    SEIZE Prib1
        ADVANCE 10,5
        RELEASE Prib1
        TERMINATE
;Імітування роботи пристрою 2
Mit4    SEIZE Prib2
        ADVANCE 10,5
        RELEASE Prib2
```

```
        TERMINATE
;Імітування роботи пристрою 3
Mit5    SEIZE Prib3
        ADVANCE 10,5
        RELEASE Prib3
        TERMINATE
;Імітування роботи пристрою 4
Mit6    SEIZE Prib4
        ADVANCE 10,5
        RELEASE Prib4
        TERMINATE
;Таймер
        GENERATE 480
        TERMINATE 1
```

Режим функції діє, коли операнд A є FN. Під час входження транзакта у блок TRANSFER підраховується значення функції, ім'я якої задане операндом B. Якщо результат не цілий, то від нього береться ціла частина. Для визначення номера наступного блока отримане значення додається до значення операнда C (значення останнього може бути записане нулем). Якщо блок з обчисленим номером зайнятий, то транзакт залишається у блоці TRANSFER до того часу, поки не зможе перейти саме до цього блока. Наприклад:

```
        TRANSFER    FN,Vybir,P4
        TRANSFER    FN,8,P4
```

У першому прикладі транзакт направляється до блока, номер якого планувальник визначає як суму цілої частини обчисленого значення функції з іменем Vybir і значення параметра 4 цього транзакта. У другому прикладі все так само, лише функція задана не іменем, а номером.

Режим параметра. Для задання параметричного режиму потрібно в полі операнда A вказати ключове слово P. У цьому режимі активний транзакт направляється до блока, номер якого визначається як сума значення параметра та значення операнда C. Якщо операнд C не використовують, то значення параметра приймається за номер нового блока. Наприклад:

```
        TRANSFER    P,Param,5
        TRANSFER    P,4,5
```

У першому прикладі транзакт направляється до блока, номер якого дорівнює сумі значення параметра з іменем Param і значення операнда C, тобто 5. У другому прикладі все так само, лише параметр заданий не іменем, а номером.

Режим підпрограми. Для задання підпрограмного режиму потрібно в полі операнда A зазначити ключове слово SBR. У цьому режимі активний транзакт завжди направляється до блока, номер якого вказаний операндом B. Номер блока TRANSFER заноситься в параметр, вказаний операндом C. Наприклад:

```
TRANSFER   SBR,Mitka,Par
```

Увійшовши у блок, транзакт переходить до блока з міткою Mitka, а номер блока TRANSFER записується в параметр з іменем Par. Якщо такого параметра немає, то планувальник його створює.

Для повернення з підпрограми використовують блок TRANSFER у параметричному режимі.

Ця технологія дозволяє організувати перехід транзакта з поверненням, аналогічний передачі керування на підпрограму з поверненням на продовження основної програми.

Режим SIM (одночасний) використовують у випадку, коли потрібне одночасне виконання кількох умов. Кожен транзакт має свій індикатор затримки SIM. У цьому індикаторі записується результат кожної спроби входження транзакта до наступного блока. Якщо планувальник виявляє умови, які перешкоджають входженню транзакта до блока, то індикатор SIM цього транзакта дорівнює 1. Якщо всі умови переходу до наступного блока задовольняються, то індикатор SIM дорівнює нулю. Якщо не виконується хоча б одна з умов, то індикатор SIM дорівнює 1. У таких випадках операнд C вказує блок, у якому перевірялась перша умова, і транзакт виконує перевірки всіх умов, поки вони не будуть задовольнятися одночасно. Блок ADVANCE також встановлює індикатор SIM, який дорівнює нулю. Перевірка станів пов’язана з блоками, які можуть затримувати транзакти. Для перевірки умов використовують блок GATE.

Приклад:

```
Mit   GATE SE Bkp1
      GATE SE Bkp2
      GATE SE Bkp3
      TRANSFER SIM,,Mit
      ENTER Bkp4
```

У наведеному прикладі транзакт не може перейти до блока ENTER, якщо БКП Bkp1, Bkp2 і Bkp3 одночасно порожні. Під час входження транзакта у блок TRANSFER перевіряється значення його індикатора SIM. Якщо транзакт був затриманий в якомусь з блоків GATE, то його індикатор дорівнює 1, і під час перевірки індикатора у блоці TRANSFER виявляється, що транзакт був затриманий. Тому він відсилається до того блока, де здійснювалася перша перевірка (Mit), і вся послідовність перевірок повторюється.

Під час затримки транзактів у блоках ASSEMBLE, GATHER або MATCH індикатор SIM не дорівнює 1.

Режим SIM використовують зрідка, оскільки за допомогою булевих змінних і блока TEST можна ефективніше організувати управління станом великої кількості об’єктів.

Блок LOOP (зациклити) призначений для організації в моделі циклів. Він модифікує параметр й управляє рухом активного транзакта, виходячи з результату цієї модифікації. Формат блока:

LOOP A,B

Операнди:

A – параметр транзакта або параметр циклу, в якому міститься число – кількість повторень якого-небудь сегмента моделі;

B – мітка блока, з якого починається цикл.

Обидва операнди можуть бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Наприклад:

LOOP KilPovt,Pochat

Нехай блок з міткою Pochat є початком циклу, тобто розташований раніше від блока LOOP. Коли транзакт, пройшовши ділянку моделі, яка починається блоком з міткою Pochat, увійде до блока LOOP, значення його параметра з іменем KilPovt зменшується на 1. Якщо це значення не дорівнює нулю, то транзакт переходить до блока з міткою Pochat, тобто цикл повторюється. Якщо ж після віднімання 1 значення параметра циклу дорівнює нулю, отже, виконана задана кількість повторень, то транзакт переходить до наступного блока.

Блок LOOP можна використовувати для моделювання СМО з відмовами і заданою кількістю повторних спроб.

DISPLACE A,B,C,D

Блок DISPLACE призначений для відшукання будь-якого транзакта і переміщення його до нового блока зі збереженням інтервалу модельного часу. Блок має такий формат:

A – номер транзакта, який потрібно перемістити;

B – мітка блока, до якого передається транзакт, вказаний операндом A;

C – номер або ім'я параметра цього транзакта, до якого записується час, що залишився до завершення його обслуговування, якщо транзакт перебував у списку майбутніх подій;

D – мітка альтернативного блока для активного транзакта.

Усі операнди можуть бути іменем додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром. Наприклад:

DISPLACE (P6+7),Mit1,Result,Mit2

Тут операнд A є виразом у дужках. Цей вираз обчислюється та округлюється до цілого. Отриманий результат є номером транзакта, який треба перемістити. Далі блок DISPLACE розшукує цей транзакт. Для останнього можливі випадки:

транзакт є в моделі і не перебуває у списку майбутніх подій;

транзакт є в моделі і перебуває у списку майбутніх подій;

транзакта із зазначеним номером немає в моделі.

У першому випадку транзакт передається до блока з іменем Mit1. У другому випадку визначається час, який залишився до його повернення у процес моделювання, і записується в параметр з іменем Result. Якщо параметра з таким іменем немає, то він автоматично створюється. Транзакт також передається до блока з міткою Mit1. У третьому випадку, тобто коли в моделі немає транзакта з потрібним номером, активний транзакт, який увійшов до блока DISPLACE, скеровується до блока з міткою Mit2. Якщо операнда D немає, то активний транзакт переходить до наступного блока.

Коли транзакт переходить до нового блока, він вилучається зі списків:

майбутніх подій;

відкладених переривань (для транзактів, які здійснюють переривання);

затримки (у порядку пріоритету);

користувача;

повторних спроб.

У той же час він не вилучається зі списків:

поточних подій;

переривань (для перерваних транзактів);

груп.

Під час пересування перервані виконання у пристроях не припиняються. Це означає, що транзакт продовжує займати пристрій.

Висновки

1. Зв'язок між транзактами реалізується через збережувані величини, які бувають скалярними і матричними. Значення параметрів транзактів і збережуваних змінюються лише за прямою вказівкою користувача. Перед початком моделювання за замовчуванням вони дорівнюють нулю.

2. Матриці створюються у поточній моделі й описуються оператором MATRIX. Матриця обмежена максимальним обсягом пам’яті, встановленим у журналі налаштувань моделі. Матриця не може бути видалена з поточної моделі, її можна перевизначити повторно іншим оператором MATRIX з тією ж міткою.

3. Для проведення розрахунків (арифметичних, перевірки умов) призначені обчислювальні вирази, які є комбінацією математичних операторів, бібліотечних функцій, СЧА і констант, що задовольняють правилам елементарної алгебри.

4. Для зберігання потрібних під час моделювання числових і рядкових даних використовуються змінні.

5. Булеві змінні дають можливість приймати рішення залежно від стану і значення об'єктів GPSS.

6. Блоки GATE, TEST, TRANSFER, LOOP і DISPLACE дозволяють змінити траєкторію руху транзакта по моделі.

7. Блоки GATE, TEST і TRANSFER використовують для моделювання різних СМО з втратами замовлень.

Контрольні запитання та завдання

1. Дайте визначення збережуваної величини і матриці в GPSS World. Як і для чого вони використовуються в моделях?

2. Назвіть особливості роботи з матрицями в GPSS World.

3. Дайте визначення змінної та виразу в GPSS World. Як і для чого вони використовуються в моделях?

4. Яким чином організуються посилання на збережуваня величини, матриці, комірки матриць, змінні.

5. Назвіть блоки, що змінюють маршрути транзактів у моделі. Вкажіть особливості їх роботи.

6. Назвіть та опишіть режими роботи блока TRANSFER.

Тема 14. Використання таблиць у GPSS World

14.1. Накопичення статистики в GPSS World

Для реєстрації статистичних даних у GPSS передбачені два типи об'єктів – черги і таблиці.

Статистика про черги повинна містити таку інформацію:

кількість транзактів, які приєдналися до черги;

кількість транзактів, які фактично приєднались до черги, і кількість таких, що потрапили на обслуговування без черги;

максимальне значення довжини черги;

середня довжина черги;

середній час очікування в черзі.

GPSS забезпечує можливість отримання такої статистики за допомогою так званого ***реєстратора черги***. У випадку використання реєстратора черги в тих точках моделі, де ресурси обмежені, планувальник автоматично збирає статистику, яка описує очікування (якщо воно є) в цих точках. Очікування транзакта починається при виникненні в моделі будь-якої умови, яка блокує його просування. Однак вимірювання тривалості очікування починається тільки при вході у блок QUEUE (стати в чергу) і закінчується коли транзакт входить в блок DEPART (покинути чергу). Ця пара блоків і є реєстратором черги. Для отримання статистики про транзакти, заблоковані перед яким-небудь блоком моделі, блоки QUEUE і DEPART розташовують перед і після цього блока відповідно. Транзакт може одночасно перебувати в кількох чергах.

Блок QUEUE збільшує довжину черги. Формат блока:

QUEUE A,B

Операнд A задає номер або ім’я черги, а операнд B – кількість одиниць, на яку збільшується поточна довжина черги під час входження транзакта у блок QUEUE. За замовчуванням B = 1.

Блок DEPART зменшує довжину черги. Формат блока:

DEPART A,B

Операнд A задає номер або ім’я черги, довжину якої треба зменшити. Операнд B задає кількість одиниць, на яку зменшується довжина черги. Якщо операнд B не використовується, то за замовчуванням довжина черги зменшується на 1.

Операнди A і B у цих блоках можуть бути іменем, додатним цілим числом, виразом у дужках, СЧА або СЧА*параметром.

Приклади:

QUEUE Cherga

QUEUE P5,P$Kilkst

У першому прикладі блок QUEUE збільшує на одиницю довжину черги Cherga після входження кожного транзакта. У другому прикладі довжина черги, номер якої заданий у параметрі номер 5 транзакта, збільшується на кількість одиниць, задану в параметрі з ім’ям Kilkst.

DEPART Cherga

DEPART V5,(V$Result+3.2)

У першому прикладі блок DEPART зменшує довжину черги Cherga на одиницю. У другому прикладі операнди A і B задані арифметичною змінною з номером 5 і виразом у дужках, який також містить арифметичну змінну з ім'ям Result. Під час входження транзакта у блок DEPART змінна й вираз у дужках обчислюються та округлюються. Після цього довжина черги, номер якої заданий значенням змінної V5, зменшується на округлене значення виразу в дужках.

Зауваження. Відсутність реєстратора не виключає існування черги. Тобто насправді черги в моделі будуть, а статистичної інформації про них у стандартному звіті не буде.

14.2. Поняття таблиці, її складові, використання таблиць. Оператори опису таблиць

Користувач може також зібрати додаткову статистичну інформацію. Для отримання щільності розподілу, його інтегральних відносних частот, середнього та стандартного відхилення деяких аргументів, якими можуть

бути СЧА (наприклад, часу перебування транзакта в моделі або затримки в її окремих частинах, довжин черг, вмісту БКП тощо), використовують ***статистичні таблиці*** TABLE і QTABLE.

Таблиці використовують для отримання вибіркових розподілів деяких випадкових величин. Таблиця складається з частотних класів, куди заноситься число потраплянь конкретного числового атрибута в кожен конкретний частотний клас. Для кожної таблиці обчислюється також математичне очікування і середньоквадратичне відхилення. Ця статистика є стандартною для всіх таблиць. Після завершення експерименту з моделлю результати, які містяться в таблицях, виводяться у Звіт (сама таблиця) і у Вікно Таблиць (гістограма розподілу).

Оператор опису таблиці TABLE має такий формат:

<ім'я таблиці> TABLE A,B,C,D

Операнди:

A – аргумент таблиці – елемент даних, частотний розподіл якого буде табулюватися. Операнд може бути ім'ям, цілим, виразом у дужках, СЧА або СЧА*параметр;

B – верхня межа першого інтервалу. Операнд може бути числом або рядком. Перший з інтервалів має ширину від $-\infty$ до величини, зазначеної в полі B, включно, другий – це проміжок (B; B+C] і так далі. Останній інтервал містить усі значення, більші від останньої межі;

C – ширина частотного інтервалу (різниця між верхньою і нижньою межею кожного частотного класу). Операнд може бути числом або рядком;

D – число частотних інтервалів. Це число не може перевищувати 8 191. Операнд може бути додатним цілим числом.

Для збирання елементів даних транзакт має увійти у блок TABULATE з тим же ім'ям таблиці, що визначене у блоці TABLE.

TABULATE A,B

Блок TABULATE табулює поточне значення заданого аргументу. Спосіб табуляції залежить від режиму роботи таблиці, який визначається оператором опису таблиці TABLE.

A – номер або ім'я таблиці, в яку табулюється значення аргумента. Операнд може бути іменем, виразом у дужках, лише додатним числом, СЧА або СЧА*параметром;

B – кількість одиниць, які повинні бути занесені в той частотний інтервал, куди потрапило значення аргумента. За замовчуванням B

дорівнює одиниці. Операнд B може бути іменем, виразом у дужках, лише додатним цілим числом, СЧА або СЧА*параметром.

Блок TABULATE записують у тому місці моделі, яке відповідає досліджуваному об’єктові. Коли транзакт входить у блок TABULATE, то для пошуку таблиці використовується операнд A. Якщо такої таблиці немає, то виникає помилка виконання. Таблиця має бути визначена оператором TABLE. Таблиця змінюється відповідно до операндів оператора TABLE.

Для збирання елементів даних транзакт повинен увійти в блок TABULATE з тим же ім'ям таблиці, що визначене у блоці TABLE.

Коли транзакт входить у блок TABULATE, оцінюється аргумент таблиці (операнд A оператора TABLE). Якщо він менше або дорівнює операнду B оператора TABLE, то вибирається перший частотний клас таблиці.

Якщо аргумент таблиці не підходить для цього класу, то клас вибирається шляхом ділення значення аргумента на операнд C оператора TABLE. Нижня межа частотного класу включається в попередній клас. Якщо таблиці не достатньо для розміщення цього значення, то вибирається останній частотний інтервал. Потім вибирається ціле число з частотного класу і лічильник збільшується на величину, визначувану операндом B оператора TABULATE. За умовчанням збільшення відбувається на 1. Наприкінці роботи оператора TABULATE змінюються значення середнього і стандартного відхилення аргумента таблиці.

Приклад:

```
ChasRem   TABLE      P$Remont,7.5,2.5,12
          TABULATE   ChasRem
```

Оператор TABLE описує таблицю з іменем ChasRem. Аргументом таблиці є СЧА P$Remont з верхньою межею першого інтервалу 7,5, шириною 2,5 і кількістю інтервалів 12. Якщо значення аргумента потрапляє до певного частотного класу, то його частота збільшується на 1, оскільки операнд B у блоці TABULATE відсутній.

Крім таблиці TABLE, використовують Q-таблиці QTABLE, які призначені лише для отримання розподілу часу перебування транзакта в черзі. Формат опису таблиці QTABLE такий самий, як і TABLE:

<ім'я таблиці> QTABLE A,B,C,D

Відмінність полягає лише в тому, що операнд A задає ім'я черги. Призначення операндів B, C і D таке саме, як і в команді опису TABLE. Операнд D може бути нулем або додатним числом.

Наприклад:

```
Wtime    QTABLE    Dovzh,10.5,2.5,15
```

Тут Dovzh – ім'я черги, а не СЧА, як у випадку використання TABLE. Під час проходження транзакта через блоки QUEUE і DEPART його час очікування фіксується, і до лічильника частотного інтервалу таблиці, до якого потрапило це значення, додається 1. Одночасно в таблиці накопичується інформація для обчислення середнього значення і середнього квадратичного відхилення часу очікування.

Варто звернути увагу, що в разі використання QTABLE інформація до таблиці заноситься автоматично під час входження транзакта до блоків QUEUE і DEPART і жодних спеціальних заходів, тобто блока TABULATE, для цього не потрібно. Після завершення моделювання зібрана в таблиці інформація також виводиться у стандартному звіті GPSS.

Перші рядки обох таблиць містять: ім'я таблиці, середнє значення табульованого аргумента (MEAN), його середнє квадратичне відхилення (STD.DEV.), а також значення кількості транзактів, які очікують виконання спеціальної умови, що залежить від стану даної таблиці (RETRY).

Кожен з наступних рядків містить інформацію про один з частотних інтервалів. У стовпці RANGE розташовані нижня та верхня межі частотного класу; FREQUENCY – сумарна частота, яка формується під час потрапляння табульованого аргумента до зазначених меж; CUM.% – величина накопиченої частоти у відсотках до загальної кількості значень табульованого аргумента.

Таблиця TABLE (QTABLE) може бути перевизначена іншим оператором TABLE (QTABLE), з тією ж самою міткою, що і перша.

14.3. Трасування

Блоки TRACE і UNTRACE мають такий формат запису:

TRACE і **UNTRACE**

Блок TRACE виконує дії, необхідні для відслідковування руху транзактів (повідомлень) до тих пір, поки вони не увійдуть у блок UNTRACE.

При вході транзакта у блок TRACE включається його індикатор траси. Після цього при кожному вході транзакта в черговий блок в журнал видається повідомлення трасування. Відстежувана ділянка програми обмежується блоком UNTRACE (TERMINATE).

Трасовані транзакти разом зі встановленим покажчиком трасування можуть бути записані у вікно даних. Можна зупинити видачу трасування у вікні даних шляхом натиснення будь-якої клавіші. Повторне натиснення клавіші повертає дисплей у попередній стан.

Трасовані транзакти записуються в робочий журнал (Session Journal), якщо ви використовуєте його. Можна визначити (у SETTINGS.GPS) файл, який отримуватиме список команд і повідомлень, записаних у журнал. Це корисно для відладки і реєстрації результатів.

Блок UNTRACE служить для припинення трасування.

Оператори TRACE і UNTRACE можуть використовуватися при ручному моделюванні для зміни покажчика трасування активного транзакта. При вставці блока TRACE в інтерактивному режимі його дія поширюється на єдиний транзакт.

Якщо вікно блоків (BLOKS WINDOW) відкрите, то блок позначається червоним кольором, коли транзакт з його покажчиком трасування входить у блок.

У графічних вікнах можна викликати трасування повідомлень (транзактів) шляхом натиснення клавіш Alt+L. Повторне натиснення клавіш Alt+L відміняє трасуючі повідомлення в рядку трасування. Це не залежить від проходу повідомлень через блоки TRACE / UNTRACE.

Висновки

1. Реєстратори черг призначені для збору статистики за чергами, але їх відсутність у моделі не виключає існування черги.

2. Черги є статистичними об'єктами, тому списки транзактів, що в них перебувають, не складаються. Використання черг ніяким чином не впливає на внутрішні операції GPSS.

3. Статистичні таблиці призначені для отримання розподілів випадкових величин.

4. Для табулювання часу перебування транзактів у черзі використовуються Q-таблиці.

Контрольні запитання та завдання

1. Які елементи GPSS World дозволяють збирати та накопичувати статистику?
2. Назвіть складові таблиць GPSS.
3. Якими операторами задаються таблиці в GPSS World?
4. Яким чином відбувається збір статистики у таблицях?
5. Яким чином у GPSS World організується збір статистики за чергами?
6. Назвіть призначення таблиць TABLE і QTABLE. Чим вони відрізняються?

Тема 15. Списки користувача та блоки для їх формування. Групи і сімейства транзактів

15.1. Списки користувача та блоки для їх формування

Під час руху в моделі транзакти можуть бути заблоковані, наприклад, блоками GATE або TEST. Якщо заблоковані транзакти перебувають у списку поточних подій, то у випадку, коли їх багато, інтерпретатор витрачає велику кількість часу на перегляд списку з метою вибору чергового транзакта для просування. Для економії машинного часу заблоковані транзакти доцільно заносити до ***списків користувача*** й залишати їх там до того часу, поки не будуть виконані умови, які дозволять подальший рух цих транзактів. Крім того, розташування транзактів у списках користувача дає змогу організувати різні дисципліни черг, які відрізняються від дисципліни "першим прийшов – першим обслуговується" (FIFO), реалізованої у списку поточних подій.

Список користувача є деяким буфером, до якого можуть тимчасово заноситися транзакти, виведені зі списку поточних подій. На відміну від списків поточних і майбутніх подій транзакти вводяться у список користувача й виводяться з нього не автоматично, а згідно з рішенням користувача відповідно до логіки моделі за допомогою спеціальних блоків.

Для введення транзактів до списку користувача призначений блок LINK (ввести у список), який може бути використаний у двох режимах:

умовному й безумовному. Обмежимося розглядом лише безумовного режиму, в якому блок LINK має такий формат запису:

LINK A,B,C

Операнди:

A – ім’я або номер списку користувача, до якого автоматично заноситься транзакт після входження у блок LINK. Може бути ім'ям, виразом у дужках, СЧА, СЧА*параметр, додатним цілим числом;

B – визначає принцип упорядкованості, тобто визначає, в яке місце списку користувача треба внести активний транзакт. Допустимі значення: FIFO (транзакт передається в кінець списку); LIFO (транзакт передається на початок списку); PR (транзакти впорядковуються за спаданням пріоритету); P (транзакти розташовуються у порядку зростання значення параметра); M1 (транзакти розташовуються в порядку зростання відносного часу перебування в моделі).

У полі операнда B можна використовувати й інші СЧА, крім зазначених вище СЧА транзактів: арифметичну змінну, функцію, а також вираз у дужках. У цьому випадку здійснюється обчислення вказаного СЧА для активного транзакта й для решти транзактів, які вже перебувають у списку користувача, починаючи з початку черги. Після цього відбувається впорядкування транзактів у списку користувача за спаданням обчисленого значення. Прямої можливості побудувати список за спаданням значень параметра немає, проте опція BACK блока UNLINK при виборі зі списку сповна її замінює. При визначенні порядку включення у список пріоритет поточного транзакта в загальному випадку ігнорується;

C – блок призначення для активного транзакта у випадку, коли він не приєднується до списку користувача (індикатор зв'язку списку користувача встановлений у нуль); індикатор зв’язку вмикається (встановлюється в 1), якщо він був вимкнений.

У найпростішому варіанті (C не використовується) транзакт завжди поміщається у список користувача A і видаляється з усіх інших списків, виключаючи групи транзактів і списки переривань.

Наприклад, блок

```
LINK   5,FIFO
```

заносить транзакти в кінець списку користувача номер 5 у порядку їхнього надходження до блока. Блок

```
LINK   Spisok,P$Znach
```

заносить транзакти у список користувача з іменем Spisok, впорядковуючи їх за зростанням значення параметра з іменем Znach.

Умови, унаслідок виконання яких транзакт заноситься до списку користувача, в безумовному режимі перевіряються засобами, передбаченими розробником моделі. Наприклад, направити транзакт до списку користувача за умови зайнятості пристрою можна так:

```
        ……
        GATE    NU  Prilad,Mit
        SEIZE   Prilad
Mit     LINK    Spisok,FIFO
        ……
```

Якщо пристрій Prilad зайнятий, то блок GATE не допускає транзакта до блока SEIZE, а направляє його до блока LINK з іменем Mit, і транзакт заноситься у кінець списку користувача з іменем Spisok.

У цьому прикладі припускалося, що список користувача необмежений, тобто до нього можна заносити будь-яку кількість транзактів. У реальних системах список користувача можна використовувати для імітування, наприклад вхідного накопичувача, ємкість якого, зазвичай, обмежена. Це обмеження можна реалізувати так:

```
Emn  EQU        5
     GATE       NU  Prilad,Mit
     SEIZE      Prilad
Mit  TEST       L  CH$Spisok, Emk,Out
     LINK       Spisok,FIFO
Out  TERMINATE
```

Якщо пристрій Prilad зайнятий, то блок GATE не допускає транзакта до блока SEIZE, а направляє його до блока TEST з іменем Mit, який розташований перед блоком LINK. Якщо поточна довжина списку користувача з іменем Spisok менша від заданої (Emk), то транзакт проходить до списку користувача; в іншому випадку він направляється до блока TERMINATE з міткою Out.

Для виведення одного або кількох транзактів зі списку користувача й занесення їх назад до списку поточних подій призначений блок UNLINK (вивести зі списку), який має такий формат:

UNLINK X A,B,C,D,E,F

Блок виводить транзакти зі списку користувача.

Операнди:

X – умовний оператор відношення між значеннями D і E, при якому відбувається видалення;

A – номер або ім'я списку користувача, з якого виводять транзакти;

B – мітка блока, до якого переходять виведені зі списку транзакти;

C – вказує кількість транзактів, які виводяться зі списку користувача, або ключове слово ALL для виведення всіх транзактів, які є у списку (за замовчуванням ALL);

D – порівнюваний атрибут (атрибут транзакта, який підлягає перевірці, булева змінна для перевірки або BACK для виведення членів з кінця списку користувача);

E – довідкове значення, з яким порівнюється операнд D (не потрібний, якщо D – булева змінна);

F – номер або ім’я блока, куди переходить транзакт, який виходить з блока UNLINK, якщо зі списку користувача не виведено жодного транзакта. Якщо операнд C не використовується, то транзакт, який виходить, переходить до наступного блока незалежно від кількості виведених транзактів.

Операнди D і E разом з умовним оператором X визначають спосіб і умови виведення транзактів зі списку користувача. Значення оператора X такі самі, як і у блоці TEST. У випадку, коли умовний оператор X потрібно використовувати, але він не вказаний, то за замовчуванням він набуває значення E (рівність). Якщо операндів D та E немає, то не використовують й умовний оператор X. У цьому випадку транзакти виводяться з початку списку, а кількість таких транзактів визначається обов’язковим операндом C.

Операнди D, E й умова необов'язкові. Якщо вони відсутні, виводяться всі транзакти від початку списку до його кінця або до досягнення ліміту (операнд C).

Розглянемо три можливі варіанти операнда D.

1. Якщо операнд D є булевою змінною. Тоді операнд E і оператор X не використовуються. Булева змінна обчислюється відносно транзакта зі списку користувача. Якщо результатом є одиниця, тобто умова виведення виконується, то транзакт виводиться. Кількість транзактів, які виводяться, визначає операнд C. Однак виведено може бути і менше, ніж вказано операндом C: за кількістю результатів обчислення булевої

змінної, які не дорівнюють нулю. Крім того, і транзактів у списку користувача може бути менше, ніж зазначено операндом C.

2. Якщо операнд D є ключовим словом BACK, то витягуються транзакти з кінця списку в кількості, визначеній операндом C. У цьому випадку операнд E й умовний оператор X не використовуються.

3. У решті випадків операнд D обчислюється стосовно транзакта зі списку користувача і використовується як номер параметра цього транзакта, значення якого і є кінцевий результат. Цей результат порівнюється з результатом обчислення операнда E.

Якщо операнд D задає параметр, а операнд E не використовується, то значення параметра транзакта зі списку користувача порівнюється зі значенням такого самого параметра транзакта, який ініціює виведення. Якщо вони рівні, то транзакт виводиться зі списку користувача. У цьому випадку кількість транзактів, які виводяться, також визначає операнд C.

Наприклад, блок

```
UNLINK   5,Mit,1
```

виводить транзакт зі списку користувача з номером 5 і направляє його до блока з міткою Mit.

Блок

```
UNLINK   Spisok,Sat,1,BACK
```

виводить зі списку користувача з іменем Spisok один транзакт з кінця списку і направляє його до блока з міткою Sat. Блок

```
UNLINK   E  P$Obl,Mit1,ALL,Rist,P$Rist,Mit2
```

виводить зі списку користувача, номер якого записаний у параметрі Obl транзакта, що ініціює виведення, і направляє до блока з іменем Mit1 всі транзакти, значення параметра Rist яких дорівнює значенню однойменного параметра транзакта, який ініціює виведення. Якщо таких параметрів у списку немає, то транзакт, який ініціює виведення, буде направлений до блока з іменем Mit2, а в іншому випадку – до наступного блока.

Зазначимо такі особливості виконання блока UNLINK. По-перше, якщо операнди D і E містять посилання на СЧА транзактів, то операнд C обчислюється відносно транзактів у списку користувача, а операнд E – відносно активного транзакта. По-друге, після виведення транзактів зі списку планувальник продовжує або починає просування транзакта з найвищим пріоритетом, а якщо вони рівні, то віддає перевагу транзакту-ініціатору виведення.

Приклад, який показує використання списків користувача для організації нестандартних дисциплін обслуговування. В одноканальній СМО з очікуванням потрібно організувати таку дисципліну обслуговування, коли пріоритет віддається замовленням з найменшим часом обслуговування

```
        GENERATE    (Uniform(3,2,10))
        ASSIGN      Tmin,(Uniform(4,1,15))
        GATE   NU   Prilad,Mit
Mit2    SEIZE       Prilad
        ADVANCE     P$Tmin
        RELEASE     Prilad
        UNLINK      Spisok,Mit2,1
        TERMINATE
Mit     LINK        Spisok,P$Tmin
        GENERATE    1000
        TERMINATE   1
        START       1
```

До параметра Tmin активних транзактів у блоці ASSIGN записується випадковий час обслуговування, обчислений за допомогою вмонтованого генератора рівномірного розподілу. Якщо пристрій Prilad вільний, то блок GATE пропускає транзакт до блока SEIZE, і пристрій займається на час P$Tmin. Якщо ж у момент надходження транзакта пристрій зайнятий, то блок GATE направляє транзакт до блока LINK з міткою Mit, де транзакт вводиться до списку Spisok із впорядкуванням за зростанням часу обслуговування, записаного в параметрі P$Tmin. Блок UNLINK у момент звільнення пристрою виводить з початку списку транзакт з найменшим часом обслуговування, забезпечуючи тим самим задану дисципліну.

BUFFER

Блок поміщає активний транзакт у список поточних подій услід за транзактами з таким же пріоритетом, і перегляд списку поточних подій починається спочатку. Остання обставина і є основною метою застосування BUFFER.

15.2. Групи і сімейства транзактів

Між транзактами, що перебувають у моделі, може існувати деякий зв'язок. Наприклад, процеси промислового виробництва будь-якого

складного виробу – "залізного" або програмного – зазвичай проходять три стадії: "зверху вниз" при проектуванні, паралельне виготовлення і збирання з комплексною відладкою. Незалежні роботи, які виконуються паралельно, не можуть забезпечуватися одним транзактом. Транзакт, що запускає їх, повинен розщеплюватися з подальшим збиранням компонентів. Очевидною є необхідність розпізнавання приналежності останніх до єдиного агрегату. Операції об'єднання необхідні також для систем з порогом обслуговування (наприклад, зібрати складний виріб можливо за наявності всіх деталей; автобус відправиться, коли буде зайнято в ньому не менше заданої кількості місць, тощо). З іншого боку, бажано мати можливість виконувати деякі групові дії над однорідними транзактами.

Усі транзакти в моделях належать до множин, які називають ***сімействами*** (***сім'ями, ансамблями***). Як правило, в моделі співіснують декілька різних сімейств. Кожен транзакт є членом тільки одного сімейства. Транзакт, введений у модель блоком GENERATE, отримує числову мітку – номер сімейства і стає першим і поки єдиним членом нового сімейства. Його нащадки (і можливо нащадки цих нащадків), що формуються при входженні у блок SPLIT, включаються до того ж сімейства.

Сімейства корисні, коли процес повинен чекати настання деяких подій. З поняттям сімейства пов'язані ***списки синхронізації*** – у цьому випадку транзакти чекають у них членів свого сімейства.

Управляти рухом транзактів залежно від їх приналежності до сімейств можна за допомогою блоків ASSEMBLE, GATHER, MATCH. За допомогою блока GATE і логічних умов M / NM вхід транзакта в будь-який блок можна поставити в залежність від наявності елементів того ж сімейства в іншому блоці.

Розглянемо блоки, призначені для обробки транзактів, що належать до одного сімейства.

SPLIT A,B,C

Блок SPLIT (розщепити) виконує функцію копіювання транзакта, що входить до нього і називається ***початковим*** або ***породжуючим***. Операнди:

A – задає число створюваних копій. Усі копії формуються в момент входу транзакта до блока SPLIT. Сам транзакт робить спробу перейти до наступного за номером блока;

B – задає номер або ім’я наступного блока, до якого переходять копії транзакта-оригінала, причому значення операнда B обчислюється для кожної копії окремо;

C – задає номер параметра, що використовується для присвоєння копіям послідовних номерів. Якщо операнд C не використовується (за замовчуванням), номерів транзактів, що виходять з блока SPLIT, в їх параметрах немає.

Після проходження блока SPLIT транзакт-оригінал переходить до наступного блока, а всі копії направляються за адресою, вказаною операндом B, або до наступного блока, якщо операнда B немає (за замовчуванням).

Таким чином, якщо операнд A дорівнює k, то з блока вийдуть k+1 транзакт. Надалі породжуючий транзакт і копії є рівноправними і можуть проходити знову через будь-яке число блоків SPLIT. Кожна нова копія, створена блоком SPLIT, стає членом сімейства транзактів, породженого одним вхідним транзактом, згенерованим блоком GENERATE. Транзакти, що належали до одного сімейства, об'єднуються планувальником у список. Якщо блоком SPLIT транзакти не були пронумеровані, то усередині сімейства транзактів неможливо встановити, який із транзактів сімейства породжуючий. Число транзактів в сімействі може бути довільним. Якщо копія транзакта входить до блока SPLIT, то вторинна копія стає членом того ж сімейства, що й первинна. Крім значень параметрів до кожної копії записуються значення пріоритету та позначка часу початкового транзакта. Копії почергово надходять до списку поточних подій, причому кожна копія розміщується в кінець відповідного пріоритетного класу.

Кожен транзакт, створений блоком GENERATE, є окремим сімейством (номер транзакта дорівнює номеру сімейства, тобто кількість сімейств дорівнює кількості згенерованих у моделі транзактів). Таким чином, кількість сімейств у системі довільна і сімейство існує доти, поки в ньому є хоча б один транзакт.

Приклади:

```
SPLIT   7
SPLIT   12,Mitka
SPLIT   3,Mitka,7
```

У першому прикладі з блока SPLIT вийде 1+7 транзактів. Сім копій слідком за породжуючим транзактом будуть направлені до наступного

блока, оскільки операнд B не використовується. Номери в параметр породжуючого і копій транзактів не записуються, оскільки операнд C не заданий. Копії мають той же пріоритет, значення параметрів і час входу в модель, що і породжуючий транзакт.

У другому прикладі блок SPLIT згенерує 1+12 транзактів. Породжуючий транзакт перейде до наступного блока, а 12 копій – до блока з міткою Mitka. Номери в параметри транзактів теж не записуються.

У третьому прикладі блок SPLIT генерує три копії, які направляються до блока з міткою Mitka. Кожна копія має той же пріоритет, час входу в модель і значення параметрів, що і породжуючий, за винятком параметра номер 7. У параметр номер 7 кожної копії буде записаний порядковий номер. Якщо параметр номер 7 породжуючого транзакта не визначений заздалегідь, він буде створений, і йому буде привласнене значення 0. Припустимо, що в даному прикладі значення параметра номер 7 породжуючого транзакта дорівнює 0. Після виходу з блока SPLIT у параметр номер 7 породжуючого транзакта буде записана 1, а в параметр номер 7 копій – 2, 3 і 4 відповідно. Якщо ж параметр номер 7 породжуючого транзакта визначений заздалегідь і його значення дорівнює n, тоді в параметрі номер 7 породжуючого транзакта і копій після виходу з блока SPLIT будуть записані n+1, n+2, n+3 і n+4 відповідно.

```
SPLIT   3,P$Adresa,Adresa
```

Цей приклад демонструє можливість направлення копій транзактів до послідовно розташованих у моделі блоків. Якщо параметр з іменем Adresa містить номер блока, припустимо n, то перша копія буде направлена до блока n+2, друга – до блока n+3, третя – до блока n+4. Ці ж номери будуть записані в параметр з ім'ям Adresa. Породжуючий транзакт зі значенням n+1 параметра з ім'ям Adresa перейде до наступного блока.

ASSEMBLE A

Блок ASSEMBLE (об'єднати) об'єднує задане число транзактів, що належать до одного сімейства, в один транзакт (тобто здійснює збирання заданого числа транзактів). Після об'єднання з блока виходить тільки один транзакт, який переходить до наступного за номером блока. При цьому транзакт зберігає всі свої колишні властивості.

Операнд А задає лічильник транзактів, що беруть участь у збиранні. Значення лічильника необов'язково має дорівнювати чисельності сімейства. Операнд А може бути ім'ям, додатнім цілим числом, виразом у дужках, СЧА, СЧА*параметром. Зауважимо, що:

для кожного сімейства в даному блоці ASSEMBLE одночасно може виконуватися тільки одна операція об'єднання;

у даному блоці ASSEMBLE може паралельно виконуватися операція об'єднання двох і більше сімейств;

для кожного сімейства операції об'єднання можуть одночасно виконуватися в кількох блоках ASSEMBLE.

Оскільки збирання продовжується протягом якогось проміжку модельного часу, блок ASSEMBLE має список синхронізації (парності). У ***список синхронізації*** поміщаються перші транзакти кожного сімейства, що увійшли у блок ASSEMBLE. Вони очікують приходу транзактів зі своїх сімейств. При входженні транзакта у блок ASSEMBLE перевіряється, чи є в списку синхронізації транзакт того ж сімейства. Можливі два випадки: транзакта даного сімейства у списку немає або він є. Розглянемо ці випадки.

1. Транзакта такого ж сімейства у списку синхронізації немає, тобто транзакт, що увійшов, буде першим з сімейства, яке збирається. Тоді обчислюється операнд А, округлюється до цілого числа і зменшується на 1 (один транзакт вже прибув). Залежно від отриманого результату S, що зберігається в комірці (лічильник збирання) першого транзакта збираємого сімейства, виконуються такі дії:

S<0 – відбувається зупинка через помилку "Лічильник збирання не додатний";

S=0 – транзакт зразу робить спробу увійти в наступний блок (тобто необхідно було зібрати тільки один транзакт);

S≥1 – транзакт виключається зі списку поточних подій і поміщається у список синхронізації блока ASSEMBLE.

2. У списку синхронізації вже є транзакт того ж сімейства, що і транзакт, який щойно увійшов у блок ASSEMBLE. Тоді транзакт, який прибув, знищується. Лічильник збирання, що зберігається в комірці транзакта, який перебуває у списку синхронізації, зменшується на 1. Коли цей лічильник збирання стане рівним нулю (тобто у блок ASSEMBLE увійшло задане число транзактів), то транзакт, що очікує, виводиться зі списку синхронізації. Якщо обслуговування цього тран-

закта не було перерване жодним з пристроїв, він робить спробу перейти до наступного блоку і у випадку успіху повертається у список поточних подій.

Зауваження. Перерваному транзакту забороняється покидати блок ASSEMBLE доти, поки всі переривання не будуть закінчені. Тому будувати модель рекомендується таким чином, щоб транзакти, обслуговування яких було перерване без видалення (звільнення пристрою), не входили у блок ASSEMBLE.

Приклади:

Блок

ASSEMBLE 5

збирає 5 транзактів одного сімейства, 4 знищується, один переходить у наступний блок. Блок

ASSEMBLE *4

збирає число транзактів, що дорівнює значенню параметра номер 4 транзакта сімейства, який увійшов у блок ASSEMBLE першим.

Приклад використання блоків SPLIT і ASSEMBLE [17].

Розглянемо фрагмент програми моделювання паралельних операцій введення-виведення:

```
        SEIZE       CPU
        SPLIT       1,PCA2
        ADVANCE     FN$TIM1
        RELEASE     CPU
PAE1    ASSEMBLE    2
PCA2    SEIZE       ARM
        ADVANCE     FN$TIM2
        SEIZE       CHAN
        ADVANCE     FN$TIM3
        RELEASE     CHAN
        RELEASE     ARM
        TRANSFER    ,PAE1
```

Транзакт займає пристрій (центральний процесор – CPU), і породжує одну копію, за допомогою якої моделюється операція введення-виведення. Потім вихідний транзакт моделює виконання програми процесором, затримуючись на відповідний час у блоці ADVANCE, після чого транзакт вивільнює процесор і входить до блока ASSEMBLE, в якому очікує закінчення операції введення-виведення. Тим часом копія

моделює виконання операції введення-виведення і займає пристрій доступу (пристрій ARM), канал, затримується на час моделювання операцій зчитування або запису у блоці ADVANCE, потім вивільнює канал і пристрій доступу та надходить до блока ASSEMBLE. Після цього вихідний транзакт може продовжувати рух.

Блок GATHER (зібрати) діє аналогічно ASSEMBLE і має ті ж властивості. На відміну від ASSEMBLE в ньому при накопиченні заданого в першому операнді числа транзактів рух продовжують ***всі*** вони в порядку FIFO. Формат блока:

GATHER A

Операнд А задає число належних одному сімейству транзактів, яке необхідно зібрати при їх русі одним шляхом (початкове число лічильника збирання). Операнд А може бути ім'ям, додатним цілим числом, виразом у дужках, СЧА, СЧА*параметром.

Блок GATHER може одночасно виконувати збирання транзактів кількох сімейств і також має список синхронізації. При входженні транзакта у блок GATHER перевіряється, чи є у списку синхронізації транзакт того ж сімейства. Можливі два випадки: транзакта даного сімейства у списку немає або він є. Розглянемо ці випадки.

1. Транзакта того ж сімейства у списку синхронізації немає, тобто транзакт, що увійшов, буде першим з сімейства, яке збирається. Тоді обчислюється операнд А, округлюється до цілого числа і зменшується на 1 та запам’ятовується отриманий результат S у комірці транзакта – лічильнику збирання. Залежно від отриманого результату S можливі такі дії:

S<0 – відбувається зупинка через помилку "Лічильник збирання не додатний";

S=0 – транзакт зразу робить спробу увійти в наступний блок (тобто необхідно було зібрати тільки один транзакт);

S≥1 – транзакт поміщається у список синхронізації блока для очікування прибуття інших транзактів свого сімейства.

2. У списку синхронізації вже є транзакти того ж сімейства, до якого належить транзакт, який щойно увійшов у блок GATHER. Він також поміщається у список синхронізації, а лічильник збирання зменшується на 1. Якщо його значення стане рівним нулю (тобто у блок GATHER увійшло задане число транзактів даного сімейства), всі зібрані транзакти

виключаються зі списку синхронізації і поміщаються у список поточних подій.

Зауваження. Якщо серед зібраних транзактів одного сімейства виявляться перервані транзакти, то після закінчення збирання вони не переводяться у список поточних подій. Перервані транзакти перебуватимуть у блоці GATHER, але не у списку синхронізації, доти, поки всі перервані обслуговування не закінчаться. Тому в моделі необхідно передбачити, щоб перервані без видалення (звільнення пристрою) транзакти не входили без необхідності у блок GATHER.

Блок MATCH (синхронізувати) призначений для синхронізації руху транзактів одного сімейства, які рухаються різними шляхами. Для синхронізації необхідні два блока MATCH, які знаходяться у відповідних місцях моделі і називаються спряженими. Формат запису:

MATCH A

Операнд А кожного блока MATCH вказує мітку або номер спряженого йому блока. Операнд може бути іменем, додатним цілим числом, виразом у дужках, СЧА, СЧА*параметром.

Наприклад:

```
Mit1    MATCH    Mit2
Mit2    MATCH    Mit1
```

У моделі ці блоки будуть поміщені окремо в паралельних шляхах руху транзактів.

При вході транзакта у блок MATCH операнд А обчислюється і округляється до цілого числа. За набутим значенням визначається спряжений блок MATCH. При його відсутності відбувається зупинка за помилкою.

Якщо спряжений блок є, перевіряється наявність в ньому транзакта з того ж сімейства. Якщо у блоці немає жодного транзакта даного сімейства, транзакт, що надійшов, поміщається у список синхронізації і чекатиме в ньому входу транзакта свого сімейства у спряжений блок MATCH.

Під час надходження такого транзакта у спряжений блок MATCH обидва транзакта одного сімейства будуть виключені зі списку синхронізації й одночасно будуть пропущені у наступні за блоком MATCH блоки.

Одна і та ж пара блоків MATCH може одночасно синхронізувати будь-яке число пар транзактів з різних сімейств. Транзакти одного

сімейства також можна синхронізувати в будь-якому числі пар блоків MATCH.

Зауваження 1. Якщо один з транзактів, що синхронізуються, будучи перерваним увійшов у блок MATCH, йому не дозволяється вийти з нього до тих пір, поки всі його перервані обслуговування не будуть закінчені. Тому будувати модель потрібно так, щоб транзакти, обслуговування яких перерване без видалення (без звільнення пристроїв), не входили в блоки MATCH.

Зауваження 2. Блок MATCH може бути спряженим сам собі. При цьому його дія буде аналогічною дії блока GATHER з операндом A, що дорівнює 2. Приклад запису в моделі:

```
Mit1    MATCH    Mit1
```

У сімейств немає імен, на які можна було б посилатися при описі логіки моделі. За допомогою блока ADOPT можна змінити приналежність активного транзакта до сімейства. Формат блока:

ADOPT A

Операнд A задає ім'я або номер сімейства, до якого буде належати активний транзакт.

Приналежність транзактів до сімейств визначається тільки *генетично*. З іншого боку, є можливість їх *довільного групування*. Кожен транзакт може одночасно належати будь-якому числу різних груп. Транзакт стає членом групи, пройшовши через блок JOIN (приєднати). Формат блока:

JOIN A,B

Блок додає активний транзакт до групи транзактів або число – до числової групи. Операнди:

A – номер групи, в яку буде доданий новий член;

B – значення, яке включається в числову групу.

Якщо B специфікує число, блок JOIN оперує з числами, інакше – з транзактами. Працюючи з числами, JOIN обчислює операнди A і B. Потім число, задане в операнді B, включається в числову групу, вказану A. Оперуючи з транзактами, блок JOIN включає вхідний транзакт у групу, специфіковану операндом A.

Нові члени поміщаються у список групи в порядку їх надходження. Групи чисел відрізняються від груп транзактів, навіть якщо мають однакові номери. Після блока JOIN активний транзакт переходить у наступний блок.

На відміну від транзактів, що входять у списки, включений в групу транзакт не стає пасивним і продовжує рухатися, залишаючись в групі до завершення своєї траєкторії. Видалення транзакта з групи можливе блоком TERMINATE, об'єднанням його з іншими транзактами в блоці ASSEMBLE або видаленням з групи (блок REMOVE).

REMOVE X A,B,C,D,E,F

Блок видаляє члени з груп. Його операнди:

X – відношення між D і E, при якому виконується видалення;

A – номер групи, з якої проводиться видалення;

B – максимальне число транзактів (може бути ALL, ім'я, константа, СЧА), що видаляються;

C – значення, яке має бути видалене з числової групи, в режимі транзактів поле C не використовується;

D – номер параметра транзакта, який визначає члени групи, що видаляються, або PR, якщо ознакою для видалення є пріоритет;

E – довідкове значення, з яким порівнюється операнд D активного транзакта;

F – альтернативний блок для активного транзакта, коли F не заданий, а також при результативному видаленні, тразакт продовжує рух по моделі.

Блок REMOVE діє в числовому режимі, якщо використовується операнд C. Інакше він працює з групою транзактів.

Найбільш використовувані комплексні варіанти:

– самовидалення – якщо B, D, E не задані, транзакт видаляється;

– лічильник – D, E не задані, кількість транзактів вказана операндом B, видаляється з групи (якщо не вистачає – всі). ALL в цьому операнді видаляє всю групу;

– управління за параметром або пріоритетом – використовуються операнди B, D, E. Операнд B указує кількість (можливо, ALL), D – параметр, що вивчається, або PR, E – порівнюваний СЧА. Умова видалення записується після REMOVE, за замовчуванням мається на увазі рівність;

– екстремаль – умова має вигляд MAX або MIN. Аналогічно попередньому випадку B, C, E не застосовуються;

– у чисельній моді REMOVE використовує операнди A, C і, можливо, F. Операндом C вказується шукане числове значення.

Знайдений елемент покидає групу, за відсутності – активний транзакт переходить за адресою F.

Шлях транзакта по моделі можна зробити залежним від його приналежності до групи, включивши в маршрут блок EXAMINE.

EXAMINE A,B,C

Блок застосовується для перевірки членства в числовій групі або в групі транзактів і визначає рух транзакта, не змінюючи склад групи. Він працює або в "числовому" режимі, або в режимі "транзакта". Якщо використовується операнд B, режим числовий. Операнди:

A – ідентифікатор групи, яка перевіряється;

B – вказує, яке числове значення розшукується;

C – альтернативний блок для активного транзакта, якщо члени групи не знайдені.

Приклад:

EXAMINE Color,P$Col,Necol

Якщо числова група з ім'ям Color не містить значення, записане в параметрі транзакта з ім'ям Col, то активний транзакт перейде до блока з міткою Necol. Якщо значення виявиться членом групи, активний транзакт переходить до наступного по порядку блока.

У режимі транзактів перевіряється приналежність активного транзакта до групи, вказаної операндом A.

SCAN X A,B,C,D,E,F

Блок працює тільки з групами транзактів. Він дозволяє тому транзакту (не обов'язково членові групи A), що увійшов до блока, вибрати перший елемент групи, параметр або пріоритет (B) якого задовольняють умові. Операнди:

X – відношення між B і C для вибору члена групи;

A – група, що перевіряється;

B – атрибут (PR або параметр), що перевіряється;

C – довідкове значення, з яким порівнюватиметься операнд B;

D – PR або номер параметра, значення якого має бути призначене параметру активного транзакта;

E – номер параметра, що набуває значення;

F – нове призначення для того транзакта, що увійшов до блока SCAN, при неуспіху.

Операнд B вказує, який атрибут транзакта перевірятиметься. Блок SCAN вибирає з групи перший транзакт, який задовольняє умові, і

запам’ятовує заданий атрибут у параметрі транзакта, що увійшов у блок SCAN. Якщо активний транзакт не має такого параметра, то він створюється.

Якщо використовується умова І/АБО для операндів B, C, то перевіряється перший член групи. Якщо умовою є MIN або MAX відшукується транзакт з екстремальним значенням вказаного атрибута. Якщо таких транзактів декілька, то береться перший з них. Якщо умовою є MIN або MAX, то операнд C не використовується. За умовчанням операнд C дорівнює нулю. Якщо умови перевірки не задані, береться перший транзакт групи. Після вибору транзакта, його атрибут, специфікований операндом D, записується в параметр транзакта, заданий операндом E. Далі транзакт, що увійшов у блок SCAN, переходить до наступного блока.

Операнди B і D завжди відносяться до члена групи, який перевіряється. Тут може використовуватися будь-який СЧА. Якщо для обчислення СЧА потребується транзакт, то використовується той, що перевіряється. Результатом є номер параметра, значення якого і є кінцевим результатом. Якщо умовний операнд X не заданий (за замовчуванням мається на увазі рівність), але задані B і C, то порівнювані значення мають бути рівними.

Якщо у групі не знайдеться жодного транзакта, який задовольняє умові, то транзакт, що увійшов у блок SCAN, переходить відповідно до операнда F, або до наступного блока.

Приклади:

```
SCAN   MIN   Grup,Rozmir,,Nomer,50
```

У цьому прикладі переглядаються всі транзакти з групи Grup і вибирається той, який має найменше значення параметра Rozmir. Якщо таких транзактів декілька, то береться перший з них. Для нього обчислюється операнд D. Потім параметру з номером 50 активного транзакта привласнюється значення параметра Nomer. Якщо група виявиться пустою, то ніяких дій не виконується.

```
SCAN   GE   Part,P$Nabir,7,P$Nomer,Mit
```

У цьому прикладі при вході транзакта у блок SCAN група транзактів з іменем Part переглядатиметься виявлення першого транзакта з параметром Nabir, який перевищить або дорівнюватиме 7. Коли транзакт буде знайдений, значення його параметра Nomer буде передане відповідному параметру транзакта, що увійшов. Потім активний транзакт

переходить до наступного блока. Якщо у групі не виявиться жодного такого члена, то наступним буде блок з міткою Mit.

Модифікувати параметр або пріоритет групи вцілому або її окремих членів можна за допомогою блока ALTER. При цьому місцеперебування членів групи є несуттєвим.

ALTER X A,B,C,D,E,F,G

Блок змінює пріоритет або параметр вибраних членів групи транзактів. Операнди:

X – покажчик відношення (буквенний!);

A – група транзактів, члени якої перевіряються на необхідність зміни;

B – максимальне число змін транзактів (за замовчуванням ALL);

C – змінюваний атрибут (PR або номер, або ім'я параметра транзакта – члена групи);

D – замінююче значення;

E – PR або параметр, який використовується для перевірки відношення;

F – базове значення, з яким порівнюється операнд E;

G – альтернативний вихід для активного транзакта за відсутності змін.

Блок ALTER вибирає транзакти з групи і змінює один з атрибутів цих транзактів. Вказаному в C атрибуту відібраних членів групи привласнюється значення D. Змінені транзакти не змінюють свою траєкторію руху, але можуть бути перенаправлені згідно з операндом G.

При відсутності покажчика відношення й операндів E або F (без перевірок) змінюються всі транзакти – до верхньої межі (операнд B). Операнд E визначає, який атрибут перевірятиметься. Він може порівнюватися з мінімумом або максимумом атрибутів всіх членів групи при MIN або MAX як вказівника відношення. Тоді будуть змінюватися всі перевірені транзакти, які мають мінімальний або максимальний атрибут. У цьому випадку операнд F не використовується.

Атрибут члена групи можна порівняти з операндом F за вказаним відношенням. В операнді E мається на увазі параметр транзакта, який перевіряється. У випадку, якщо будь-який інший операнд є СЧА, що відноситься до транзакта, то він обчислюється для транзакта, який увійшов у блок.

Відношення між атрибутом транзакта (операнд E) і перевірюваним значенням (операнд F) визначає умови зміни транзакта. За замовчуванням покажчиком відношення вважається рівність.

Операнд G вказує транзакту, який увійшов у блок, альтернативний маршрут коли: 1) жоден транзакт не був змінений; 2) задане операндом B значення лічильника змін не може бути досягнене. Якщо операнд G не використовується, транзакт переходить до наступного по порядку блока.

Приклади:

ALTER Grup,ALL,Znak,123

У цьому прикладі всім транзактам – членам групи Grup, встановлюється параметр з іменем Znak рівним 123.

ALTER NE Grup,5,Znak,45,P$Param,12,MitOut

У групі з іменем Grup відшукуються транзакти, у яких параметр з іменем Param не дорівнює 12. Першим 5 транзактам, які зустрінуться при перевірці, параметр з іменем Znak встановиться в 45. Якщо 5 транзактів не можуть бути знайдені при перевірці, транзакт робить спробу увійти в блок з міткою MitOut, інакше транзакт переходить до наступного блока.

Контроль приналежності транзактів до групи відбувається через списки їх номерів. У блоках JOIN, REMOVE, EXAMINE є можливість працювати в числовій моді – з вільно призначеними числами замість номерів. Розробник моделі, наприклад, затребувати список всіх зайнятих пристроїв. Блоки SCAN і ALTER цей режим не підтримують.

На кожну групу створюється FIFO-список. У режимі транзактів у списку фігурують номери транзактів, у числовому – довільні числа, назначені користувачем. У межах групи числа мають бути різними. Числова мода зазвичай використовується для збереження номерів пристроїв і БКП. Вибір моди визначається першим блоком, що посилається на групу у процесі прогону. Блоки JOIN, REMOVE, EXAMINE працюють з групою в будь-якому режимі, SCAN і ALTER – тільки в режимі транзактів. Крім транзактів, групи можуть скадатися і з пристроїв і з БКП (нарізно), що задовольняють умові відбору. Відповідно все сказане вище про транзакти (виключаючи рух) відноситься і до них.

Висновки

1. Списки користувача дозволяють моделювати будь-яку дисципліну обслуговування крім "першим прийшов – першим обслугований".

2. Розщеплення транзактів з подальшим їх збиранням дозволяє моделювати роботи, що паралельно виконуються над одним транзактом.

3. Операції об'єднання дозволяють моделювати системи з порогом обслуговування, а також дають можливість виконувати групові дії над однорідними транзактами.

4. Приналежність транзактів до сімейств визначається тільки генетично, але розробник моделі має можливість довільно групувати їх. Транзакт може належати одночасно до кількох груп, але тільки до одного сімейства.

5. Транзакти, включені у групи, залишаються активними, тобто продовжують рухатися і залишаються у групі до завершення своєї траєкторії.

Контрольні запитання та завдання

1. Дайте визначення списку користувача. Для чого він призначений?

2. Які блоки призначені для формування списків користувача? Назвіть особливості роботи цих блоків.

3. Дайте визначення групи і сімейства транзактів. Назвіть відмінності між ними. Для чого вони використовуються в GPSS World?

4. Які блоки призначені для роботи з групами та сімействами транзактів? Назвіть особливості роботи цих блоків.

Тема 16. Сучасний стан імітаційного моделювання. Основні сфери використання імітаційних моделей

16.1. Сучасний стан і розвиток імітаційного моделювання в Україні та за кордоном

В Україні і колишньому СРСР становлення моделювання, як наукової і прикладної дисципліни, пов'язане з ім'ям члена-кореспондента АН СРСР Бусленко Н. П.

Методологічною основою для розвитку імітаційного моделювання стали роботи Бусленко Н. П., Глушкова В. М., Мойсеєвого Н. Н., Марчука Т. І., Коваленко І. Н. У колишньому СРСР склалися чотири школи у галузі імітаційного моделювання. *Московська* – Мойсеєвий Н. Н.,

Ємельянов С. В., Калашников В. В., Воскресенська Т. З., Немчинов Б. В., Андріанов А. Н., Бичків С. П., Хорошилов А. І., Чорненький В. М., Лутков В. І., Бусленко В. Н., Рівес Н. Я. – на чолі з Бусленко Н. П. *Ленінградська* – Фомін Б. Ф., Аврамчук Є. Ф. – на чолі з член-кореспондентом АН СРСР Вавіловим А. А. *Новосибірська* – Нечипоренко М. І., Чинін Г. Д., Окольнішников В. В., очолювана академіком АН СРСР Марчуком Г. І. *Київська* – Марьяновіч Т. П., Літвінов В. В., Коваленко І. Н., Калініченко Л. А., Гусев В. В., Жук К. Д., Яровицький Н. В., Бакаєв А. А., Нікітін А. І. – на чолі з академіком АН СРСР Глушковим В. М.

Вказані школи не мали офіційного статусу, та їх керівники мали високий науковий авторитет і публікації основоположних робіт з питань комп'ютерного імітаційного моделювання, а їх учні очолили наукові колективи, які виконали і продовжують виконувати великий обсяг досліджень за такими традиційними напрямами у галузі імітаційного моделювання:

розвиток методології, методів і технологій моделювання;

розробка засобів і систем моделювання на базі універсальних алгоритмічних мов моделювання;

розробка пакетів моделювання широкого призначення;

розробка проблемно-орієнтованих пакетів моделювання.

Представники Київської школи імітаційного моделювання на чолі з академіком Глушковим В. М. вели розробки і дослідження по всіх вказаних напрямах у галузі імітаційного моделювання. Інститут кібернетики є піонером у галузі розробки методів і засобів імітаційного моделювання на основі мов моделювання високого рівня для вітчизняних ЕОМ всіх поколінь.

Природно, що творці засобів і систем імітаційного моделювання в Україні використовували в міру можливості зарубіжний досвід розробки таких систем. Історія становлення і розвитку імітаційного моделювання в Україні і в СРСР пов'язана з відповідними етапами у світовій практиці в даній галузі.

Визначальними факторами в історії імітаційного моделювання були генерації мов моделювання. Проте упродовж всієї історії змінювалися концепції, парадигми програмування, платформи, що вплинуло на специфіку відповідних етапів.

Відомі фахівці в галузі імітаційного моделювання Р. Ненсі і Ф. Кивіат у своїх роботах визначили декілька етапів у практиці розвитку імітаційного моделювання.

Етап 1 (1955 – 1960). Програми для завдань моделювання розроблялися на основі таких загальновідомих універсальних мов як FORTRAN і ALGOL.

Етап 2 (1961 – 1965). З'явилися перші мови моделювання: GPSS, SIMSCRIPT, SIMULA, CSL, SOL. Була розроблена так звана концепція погляду на світ (world view).

Етап 3 (1965 – 1970). З'явилося друге покоління мов моделювання GPSS V, SIMSCRIPT II.5, SIMULA 67.

Етап 4 (1971 – 1978). Розвиток вже розроблених мов і засобів моделювання, орієнтоване передусім на підвищення ефективності процесів моделювання і перетворення моделювання в більш простий і швидкий метод дослідження складних систем.

Роботи Зейглера і Йорена зіграли важливу роль у вирішенні проблеми таксономії імітаційних моделей (вони запровадили мета-концепції моделі і схеми експерименту).

Етап 5 (1979 – 1984). Роки переходу від програмування до розвитку моделей. Основний акцент був перенесений на ідентифікацію інтегрованих засобів імітаційного моделювання.

Процес моделювання включає такі етапи, як створення моделі, програмування, проведення імітаційних експериментів, обробку й інтерпретацію результатів моделювання. Проте традиційно перевага віддавалася етапу програмування. Схема моделювання, що виникає при цьому, багато в чому повторює схему проведення натурних випробувань і зводиться лише до імітації траєкторій вивчених моделей. З появою імітаційних моделей змінилася концепція моделювання, яка тепер розглядається як єдиний процес побудови і дослідження моделей, що має програмну підтримку. Тепер основним є формальне поняття моделі, яке не тільки пояснює динаміку системи, але і служить предметом математичних досліджень. Стає можливим достовірний аналіз багатьох практично важливих властивостей моделі (стаціонарних розподілів, малої ймовірності, чутливості, надійності і достовірності результатів моделювання). Ці властивості особливо суттєві при дослідженні високовідповідальних і крупномасштабних систем, де ціна помилки особливо висока.

Етап 6 (1985 – 1994). Перенесення програмного забезпечення для імітаційного моделювання на персональні ЕОМ з використанням засобів графічного інтерфейсу (для візуалізації й анімації процесів моделювання).

Етап 7 (1995 – 1998). Розробка засобів технологічної підтримки процесів розподіленого імітаційного моделювання на мультипроцесорних ЕОМ і мережах.

До моменту початку розробки (1966 р.) першої в Україні мови моделювання систем із дискретними подіями були відомі закордонні аналоги таких мов, як SIMSCRIPT, SOL, і SIMULA. Та за умовами того часу не було можливості придбати відповідне програмне забезпечення за кордоном. Була можливість ознайомитися з відповідними розробками тільки на рівні публікацій. Тому в Інституті кібернетики академії наук України було прийняте рішення про розробку власної мови і системи моделювання.

Системи імітаційного моделювання, розроблені в Інституті кібернетики відповідно до вищеназваних етапів, наведені в табл. 16.1. Усі представлені системи є оригінальними розробками, виконаними на рівні зарубіжних досягнень. У процесі створення цих систем були розроблені вхідні мови, методології і технології моделювання, забезпечена їх програмна реалізація. Важливо відзначити, що всі без винятку роботи проводилися у зв'язку з виконанням відповідальної прикладної тематики й отримали широке впровадження і застосування.

Протягом 1966 – 1968 рр. в Інституті кібернетики під керівництвом Марьяновіча Т. П. виконувалися роботи зі створення мови і системи моделювання систем з дискретними подіями СЛЕНГ, що і поклало початок розвитку методів імітаційного моделювання в Україні. Як прототип була вибрана мова SOL.

Розробка такої мови зажадала в першу чергу вирішення проблеми формалізації досліджуваних систем. Традиційно методи імітаційного моделювання реалізують процес формалізації досліджуваних систем на основі таких понять, як "концептуальна модель" і "узагальнена схема функціонування".

Високорозвинуті мови імітаційного моделювання спираються на відповідні концептуальні бази (набори понять), у термінах яких і формулюються (представляються) концептуальні моделі досліджуваних систем і процесів.

Таблиця 16.1

Основні етапи розвитку методів імітаційного моделювання

№ п/п	Етапи	Країна	Мови і системи моделювання	Клас досліджуваних систем
1	1955 – 1960		FORTRAN, ALGOL	дискретні, неперервно-дискретні
2	1961 – 1965	USA UK Norway	GPSS, SOL, SIMSCRIPT CSMP360 CSL SIMULA	дискретні неперервно-дискретні дискретні дискретні
3	1966 – 1970	USA Norway Ukraine	GPSS V, SIMSCRIPT II.5 SIMULA 67 СЛЕНГ	дискретні неперервно-дискретні дискретні
4	1971 – 1978	USA UK Russia Ukraine	GASP IV CADSIM DEMOS, ECSL MODEL-6 АЛСИМ- БЕСМ НЕДІС	неперервно-дискретні неперервно-дискретні дискретні дискретні дискретні неперервно-дискретні
5	1979 – 1984	USA UK Ukraine	GPSS/H, HOCUS, MICROSAINT 3.1, MIC-SIM INTER-SIM АЛСИМ-2 ТАЇС	дискретні дискретні дискретні дискретні дискретні
6	1985 – 1994	USA UK Germany Ukraine	SLAM II PC System Animation PC Model ESL SIMPLE ++ НЕДІС-90	неперервно-дискретні дискретні дискретні неперервно-дискретні неперервно-дискретні
7	1994 – 1998	USA UK Germany Ukraine	Розробка технологій і засобів підтримки розподіленого імітаційного моделювання	

Склад концептуальної бази формується залежно від проблемної орієнтації мови моделювання (для систем з дискретними подіями, неперервних систем і гібридних систем). Так, концептуальна база систем моделювання дискретних процесів включає такі поняття, як об'єкт (процес), клас об'єкта, атрибут об'єкта, схема поведінки об'єкта, пріоритет об'єкта, подія, час, список подій.

Відмітимо, що в умовах широкого використання парадигми об'єктно-орієнтованого програмування, вказана множина понять є загальноприйнятою в багатьох мовах і системах програмування. Проте на початку 70-х років (до появи мови СИМУЛА – 67) використовувалися різні понят-

тя для опису і представлення процесів функціонування складних систем: повідомлення, процес, активності та ін. Розробники мови СИМУЛА – 67 поклали початок об'єктно-орієнтованому представленню досліджуваних систем.

Узагальнена схема описує в термінах відповідної концептуальної бази процес функціонування досліджуваних систем.

Тут відомо два підходи: подієвий і процесний.

Процесний підхід представляє функціонування будь-якої системи як дії, що розвиваються в часі, і взаємодії паралельно протікаючих процесів. Кожен процес – це ланцюжок подій, виконання кожної такої події призводить до зміни стану системи. Усі події впорядковані в часі і на послідовному комп'ютері виконуються квазіпаралельно.

При подієвому підході в системі виділяються класи подій. Управління процесом моделювання полягає у виборі й активізації програми відповідної події.

Основні розробки Інституту кібернетики в галузі мов і засобів імітаційного моделювання базувалися на процесному підході.

Таким чином, метод імітаційного моделювання вимагає розробки концептуальної бази, узагальненої схеми, мови моделювання й імітаційної системи, орієнтованих на відповідну галузь застосувань.

У 1973 році в Інституті кібернетики була завершена робота зі створення і реалізації на ЕОМ БЕСМ-6 системи АЛСИМ-БЕСМ. Система призначалася для дослідження обчислювальних систем і мереж. У системі були виділені три мовні рівні: мова опису моделей, мова управління моделюванням, мова управління завданнями. Система знайшла застосування при рішенні завдань радіолокації, протиповітряної оборони, вирішенні завдань аналізу і розподілу ресурсів.

Протягом 1973 – 1975 рр. в Інституті кібернетики були виконані роботи із створення мови моделювання НЕДІС і відповідної імітаційної системи, призначених для моделювання широкого класу реальних систем дискретної, неперервної і неперервно-дискретної природи. Розробка системи НЕДІС базувалася на використанні досвіду розробки і впровадження системи СЛЕНГ. Мова НЕДІС увібрала в себе окремі деталі з відомих у той час за публікаціями мов СИМУЛА-67 і АЛГОЛ-68. За своїми можливостями система НЕДІС близька до систем на базі таких мов, як SIMULA-67 і GASP-IV. Система НЕДІС не мала аналогів у практиці вітчизняного програмування СРСР. Розробники системи НЕДІС,

окрім робіт з упровадження і супроводу системи, виконали великий обсяг робіт з адаптації системи НЕДІС в різних прикладних областях. Вбудований в мову НЕДІС механізм бібліотечних вступів і висновків дозволяв створювати багаторівневі бібліотеки додатків.

Система використовувалася для проведення досліджень і різного роду проектних робіт у таких галузях: проектування обчислювальних машин, систем і мереж передачі даних; підземні пожежі у вугільних шахтах; технологічні процеси на залізничному транспорті; планування ремонтних і профілактичних робіт для різних парків літаків; проектування засобів і систем космічної техніки; проектування технологічних процесів у суднобудуванні; системи управління і контролю в конверторному виробництві; системи управління і контролю засобами і системами зв'язку на морських судах.

У 1979 – 1980 рр. були виконані роботи зі створення моделюючого комплексу АЛСИМ – 2. Його математичне забезпечення підтримувало вирішення завдань проектування обчислювальних систем і мереж.

Моделюючий комплекс АЛСИМ – 2 включав два типи підсистем і засобів. До першого типу відносяться засоби планування й управління проектуванням і генерації документальної частини проекту. Друга група підсистем забезпечувала вирішення завдань проектування. До складу комплексу АЛСИМ – 2 була включена система управління інформацією, що базується на мові визначення даних (для опису схем структур даних, що зберігаються в базі даних) і мові маніпулювання даними (для обміну з базами даних, корекції баз даних, аналізу, синтезу і перетворення даних).

Система АЛСИМ – 2 широко використовувалася при дослідженні процесів функціонування військово-морських баз Тихоокеанського узбережжя.

Система алгоритмічного моделювання ТАІС розроблена на початку 80-х років. Система базувалася на спеціально розробленій мові АЛГОРИТМ-80, призначеній для відпрацювання проектних рішень на рівні міжрегістрових і міжмодульних передач, для проектування (представлення) і дослідження (моделювання) апаратури обчислювальних систем в основному на етапі алгоритмічного і структурно-алгоритмічного опрацьовування. За допомогою системи ТАІС був промодельований макроконвеєрний обчислювальний комплекс, розроблений в Інституті кібернетики.

Протягом 1991 – 1993 рр. в Інституті кібернетики виконувалися роботи зі створення технологічної системи програмування НЕДІС-90 і реалізації її на ПЕОМ IBM PC AT/386. Система призначена для оперативної розробки проблемно-орієнтованих мов для найширшого кола застосувань, наприклад: дискретні, неперервні і гібридні моделі інформаційних, економічних, біологічних та інших систем, планування й обробка результатів експериментів; специфікація пристроїв у системах проектування, логічне моделювання, синтез описів нижнього рівня (схем). Користувачі системи дістають можливість будувати власні функціональні еквіваленти таких мов, як SIMULA, GASP-IV, VHDL і їх розумно спеціалізовані діалекти без побудови нових компіляторів. Створювані проблемно-орієнтовані мови можуть бути як імперативними, так і декларативними. Розроблена технологія створення нових мов моделювання для різних застосувань базується на використанні механізму контекстних модулів.

Система побудована на основі компілятора з базової мови об'єктно-орієнтованого програмування НЕДІС-90, що регламентує виключно методи побудови нових визначень на основі використовуваної системи позначень. Система реалізована в 1994 р. як компілятор на мові С для комп'ютерів, сумісних з IBM PC.

Значні роботи були виконані в ІК зі створення проблемно-орієнтованих пакетів на основі мови НЕДІС. Це такі пакети, як СИМПО (Процеси пожежогасінні у вугільних шахтах), САУККС (Системи контролю й управління комплексом зв'язку на суднах), ПАРК (Залізничні транспортні системи), МЕРЕЖА (орієнтований на мережі передачі даних, мережний і транспортний рівні), КОМПЛЕКС (Обчислювальні системи).

Великий обсяг робіт був виконаний в Інституті кібернетики зі створення засобів імітаційного моделювання для систем, формалізоване представлення яких забезпечується за допомогою моделей скінченних автоматів Мура з детермінованими виходами, кусочно-лінійних агрегатів, логіко-диференціальних рівнянь, моделей теорій оптимальних рішень, інтегро-функціональних моделей. Програмній реалізації вказаних засобів передував етап досліджень зі створення теоретико-концептуальної бази відповідних математичних моделей і методів. Перелік цих розробок представлений у табл. 16.2.

До прикладних розробок для персональних ЕОМ відносяться пакети МОРЖ (Складні технічні системи, 1990 р.), СУЗИ (Система базу-

вання суден, 1991 р.), РЕМ (Система управління запасами, 1991 р.). Вхідні мови вказаних пакетів є непроцедурними, як інструментальні засоби використовувалася мова С і СУБД Fox Pro.

Таблиця 16.2

Модельно-орієнтовані імітаційні пакети

Найменування пакета, рік створення	Клас досліджуваних систем	Тип математичних моделей	Вхідна мова (мова програмування)
АМОС-М, 1979	Системи з дискретними подіями	Кусочно-лінійні агрегати	FORTRAN
ДЕПРОГ, 1981	Прогнозування демографічних факторів	Дискретно-ініціальні автомати Мура з детермінованими виходами	PL-1
ВОКОН, 1981	Часова оцінка якості обслуговування населення	- " -	PL-1
ПАТ-1, 1983	Фізіологічні системи, системи дихання і кровообігу	Рівняння теорії оптимальних рішень	FORTRAN
ЕСКІЗ, 1983	Нелінійні динамічні системи	Диференціальні і логіко-динамічні рівняння	FORTRAN
МРС-1, 1984	Економічні і біологічні системи	Інтегро-дифференційні динамічні моделі	FORTRAN

У 1975 – 1977 рр., а потім в 1978 – 1980 рр. розроблені відповідно перша і друга версії пакету прикладних програм ДІСМ, орієнтованого на моделювання систем з дискретними подіями. ДІСМ базувався на мові PL/1, був реалізований на ЄС ЕОМ під управлінням ОС ЄС.

У 1980 р. була завершена робота зі створення програмного комплексу ПОСИМЕЯ – ФОСИМ, орієнтованого на моделювання систем з дискретними подіями. Комплекс забезпечував автоматизацію процесу конструювання імітаційних моделей у діалоговому режимі.

Розроблені пакети використовувалися при дослідженні виробничих процесів у морському порту, виробничої діяльності суден тралового флоту, систем обробки даних у реальному масштабі часу, управління пасажирським міським транспортом, планування робіт служби мережі

Міністерства зв'язку СРСР, задач управління і статистичної обробки в інформаційно-обчислювальному центрі ЦСУ Молдавської РСР.

Серед інших робіт зі створення проблемно-орієнтованих пакетів моделювання можна назвати розробки Вінницького політехнічного інституту (засоби автоматизації ерготехнічних систем), Львівського Обчислювального Центру Інституту прикладних проблем механіки і математики (інструментальні засоби моделювання систем з мікропроцессорним управлінням); Київського НДІ автоматизації будівельних систем (засоби імітаційного моделювання для дослідження технологічних процесів на будівельних об'єктах), Київського політехнічного інституту.

Таким чином, Україна завдяки піонерським роботам Інституту кібернетики має значний досвід розробки і впровадження в різних прикладних областях засобів і систем імітаційного моделювання. Величезна географія впроваджень свідчить про значний вплив вказаних розробок на вирішення таких загальнодержавних і національних проблем, як прийняття відповідальних проектних рішень у різних прикладних областях; підготовка і навчання наукових і науково-технічних фахівців найбільш сучасним інструментаріям досліджень на базі ЕОМ; накопичення і використання досвіду досліджень у різних прикладних областях у стандартній для всіх користувачів і дослідників формі.

Українські розробники багато уваги приділяли питанням популяризації методів і засобів імітаційного моделювання (читали лекції в різних вузах країни, виступали з доповідями на Всесоюзних, Республіканських і Міжнародних семінарах та конференціях, виконували великий обсяг робіт з авторського супроводу, надавали систематичну методичну допомогу з питань розробки імітаційних моделей в різних застосуваннях).

На підставі спільної роботи з користувачами відпрацьовувалися методики побудови імітаційних моделей і проведення модельних експериментів. При цьому особлива увага приділялася питанням розуміння суті методу імітаційного моделювання і формування відповідного світогляду програміста.

Високорозвинуті мови моделювання перевершили своїм значенням їх початкове призначення і стали важливим фактором у пізнанні світу й отриманні інформації про нього. З їх появою реальною стала можливість вивчати досліджувані системи у всій їх складності, не втискуючи їх в моделі, зручні для застосування тих або інших відомих математичних

методів аналізу. Процес розробки імітаційних моделей дозволяє осмислити дійсність (виявити взаємозалежності, необхідні заходи, тимчасові співвідношення, необхідні ресурси), крім того, з'являється можливість упорядкувати наші нечіткі або суперечливі поняття і невідповідності. Така модель змушує розробника організувати його задуми, оцінити і перевірити їх обґрунтованість.

Розпад Радянського Союзу та реформи призвели до втрати звязків між науковими колективами й окремими вченими, припинення активної діяльності багатьма з них, призупинення чи припинення низки перспективних розробок.

Аналіз WWW-ресурсів в Україні і Росії показав, що немає спеціалізованих інформаційних вузлів з проблем імітаційного моделювання. У той же час у країнах Європи, США й інших цій проблемі приділяється значна увага. Існує Міжнародне товариство комп'ютерного моделювання (SCS) у США, яке має свої філії в Європі (Європейське відділення SCS-Europe), Федерація європейських товариств з моделювання (EUROSIM), Інститути наук з моделювання (McLeod Institute of Simulation Science) – підрозділ SCS з центрами в багатьох країнах, і коледжі імітаційного моделювання. Україна і Росія також входять до складу EUROSIM, проте, практично ці філії не працюють.

З цієї проблеми проводиться цілий ряд щорічних конференцій і випускається велика кількість наукових журналів.

З появою першої об'єктно-орієнтованої мови в 1967 р. СИМУЛА-67 її можливості були використані фахівцями з імітаційного моделювання. На базі цієї мови були створені системи імітаційного моделювання. Україна займала одну з лідируючих позицій з цієї тематики в СРСР. Пригадаємо, що одні з кращих мов імітаційного моделювання, такі як СЛЕНГ, АЛСИМ, НЕДІС, були створені Інститутом кібернетики АН України.

Проблематикою моделювання займається також Інститут проблем моделювання в енергетиці АН України, який випускає власний журнал "Електронне моделювання". Нажаль, академічні інститути в Україні втратили свої лідируючі позиції в галузі імітаційного моделювання. Нині цими питаннями займаються у вищих навчальних закладах Києва, Одеси, Донецька, Вінниці, Чернігова й інших містах.

У Національному технічному університеті України "КПІ" накопичений значний досвід роботи в галузі комп'ютерного моделювання в різних сферах. Ще в 1977 р. в КПІ була проведена Перша всесоюзна конференція з моделювання. Таким чином, настала необхідність у створенні Центру імітаційного моделювання. Основний напрям цього центру пов'язаний з вирішенням проблем синтезу й аналізу систем різної природи і надання методичної, інформаційної і програмної підтримки для фахівців у галузі імітаційного моделювання.

Якщо поставити питання "Навіщо потрібне імітаційне моделювання?", відповідь буде простою – для прийняття рішень. Простіше вчитися на помилках, які отримують у процесі моделювання, ніж на своїх власних. Тому існує велика кількість лабораторій з цієї тематики при університетах у США і в Європі, які вирішують конкретні завдання з поліпшення продуктивності фірм, обґрунтування їх структур, вибору найкращих технологічних процесів та ін.

Завдання Центру імітаційного моделювання:

поширення досвіду з моделювання в різних сферах діяльності людини;

надання інформації про стан імітаційного моделювання в Україні і в країнах СНД;

поширення й узагальнення світового досвіду з комп'ютерного моделювання з метою розробки методичних основ і їх впровадження у вищих навчальних закладах України;

розробка україномовної термінології з моделювання і її поширення;

створення робочих груп для розробки проектів з використанням імітаційного моделювання.

16.2. Мови та системи моделювання

Будь-який алгоритм моделювання може бути записаний на універсальній процедурно орієнтованій мові. Перевагами подібної організації роботи є:

можливість роботи практично на будь-якій ЕОМ;

відсутність необхідності у спеціальному програмному забезпеченні та його вивченні;

"прозорість" логіки моделі не тільки для розробника, а й для широкого кола програмістів;

доступність вбудованих функцій, бібліотечних підпрограм, засобів статистичного аналізу, редагування виведення та інших можливостей, наданих штатними системами програмування широкого призначення;

відсутність обмежень на склад і логіку моделі, виключаючи накладені конфігурацією і режимом її використання;

можливість управління експериментом і реалізації методів зменшення дисперсії результатів.

Можливість створення мов моделювання базується на двох обставинах:

1) навіть у складних моделях з тисячами елементів кількість видів дій залишається дуже обмеженою. Такі дії природно описувати як підпрограми – якщо є потреба, з додатковими параметрами;

2) імітаційні моделі досить широких класів реальних процесів мають схожу логіку, що визначається загальною математичною моделлю.

У загальному випадку моделювання забезпечується вхідною мовою, компілятором, системою інтерпритації окремих конструкцій мови, бібліотекою стандартних підпрограм, банком даних (для проблемно орієнтованих систем), засобами відладки тощо. Тому правильніше користуватися терміном "система моделювання".

Мови моделювання дозволяють писати імітаційні програми в такій формі, яка нагадує опис модельованої системи, причому так, що незначним змінам опису системи відповідають незначні зміни у програмі. Системи моделювання спрощують опис моделі, дозволяють будувати модель і працювати з нею в діалоговому режимі, часто включають в себе банки даних по предметних областях.

Перетворення станів елементів системи зазвичай повязане з виконанням арифметичних операцій над відповідними змінними. Тому мова моделювання повинна включати в себе базисні обчислювальні засоби. Специфічні модельні засоби мають забезпечити зручність опису реальних обєктів, імітації подій, паралельного виконання процесів та їх взаємодії. Найбільш характерною особливістю систем моделювання є наявність виділеної змінної – системного часу. Сервісні засоби мають забезпечити зручний інтерфейс, засоби відладки, збирання й обробку статистичного матеріалу.

Логіка моделювання дискретних процесів в основному вбудована в універсальний інтерпритатор і при написанні програми моделювання

мається на увазі. Природно, що програма моделі виявляється коротшою, ніж на універсальній мові, оскільки в останньому випадку моделююча програма є синтезом опису моделі та її інтерпретатора. Компілятори забезпечують контроль синтаксичної правильності програми. Це забезпечує швидкість і правильність складання програм моделювання, особливо при можливості інтерактивної роботи з моделлю.

Відомо багато систем моделювання (більше 500). Серед найбільш використовуваних згадуються Симула і GPSS. Необхідно також назвати мову Симскрипт (створену на базі Фортрана), алголоподібну мову SOL, ПЛИС (інтергує можливості ПЛ/1 і GPSS), Паскаль Плюс (розширений вбудованими засобами моделювання), система СМДП (діалогова версія GPSS). Низка мов моделювання була створена в Радянському Союзі (про них згадувалося в попередньому розділі).

Останнім часом посилюється тенденція до створення універсальних мов. На цьому шляху можливі дві стратегії:

"статична" мова з величезним набором засобів, яка потребує формулювання прикладних задач в її термінах (приклад – *GPSS*);

"динамічна" мова, яка допускає створення на основі ретельно відібраного ядра спеціалізованих класів для різних додатків (приклад – *Симула 67*).

Максимальне спрощення опису задач і підготовки відправних даних простіше за все досягається спеціалізацією систем моделювання. Чим вужча проблемна орієнтація, тим менше роботи з програмування. Так можна добитися повної параметризації системи, і тоді завдання конкретної моделі полягатиме лише у вказівці числових значень параметрів. Така система матиме обмежене застосування. Тому реальні спеціалізовані системи допускають синтез довільних структур з включених у базу даних типових елементів. Прикладами можуть бути системи *СЛЕНГ* і *NetCracker* моделювання комп'ютерних мереж.

Короткий огляд деяких нових систем і мов моделювання

Використовуючи можливості візуального моделювання і сучасні технології дизайну й анімації, можна значно прискорити дослідження. За кордоном зявилася велика кількість систем імітаційного моделювання. Комерційні симулятори спеціалізовані по галузях промисловості: *eM-Plant* (машинобудування), *DELMIA* (суднобудування), *NETRAC* (телекомунікації і зв'язок).

Розв'язання задач з переважанням логістичних аспектів можна отримати за допомогою будь-яких комерційних симуляторів: *GPSS*, *Simula*, *AutoMod*, *eM-Plant*, *Extend*, *ProcessModel*, *QUEST*, *SIMFACTORY*, *Taylor ED*, *WITNESS*, *Arena*.

Мова *C* і *C++* досить популярні. Основними перевагами їх є обєктна орієнтація і використання їх як інструмента для написання ядра операційних і модулюючих систем. Це полегшує інтеграцію з останніми нових створених (як правило у формі динамічних бібліотек) розширень та надбудов. Недоліками *C++* вважаються незручність програмування і складність супроводження програм. Основою цього та інших недоліків цієї мови є статична типізація в її основі. Ті ж недоліки властиві мовам програмування *Java*, *Object Pascal*, *Delphi*. Тому альтернативою є *SmallTalk* – об'єктно-орієнтована мова з динамічною типізацією.

Розглянемо декілька систем імітаційного моделювання.

Система *BPsim* спирається на апарат динамічних експертних систем і значною мірою вільна від типових недоліків. У ній визначені нові класи об'єктів: операції, ресурси, засоби, процеси, джерела і споживачі ресурсів, перехрестя, параметри. Параметри процесу задаються функцією від характеристик об'єктів і діляться на похідні (згортка різного типу характеристик) і консолідовані (згортка одноіменних характеристик операцій процесу). Опис причинно-наслідкових зв'язків задається спеціальними об'єктами.

Новий професіональний інструмент *AnyLogic* поєднує об'єктно-орієнтований підхід, візуальне проектування, дружній графічний інтерфейс, мову Java, агентні технології, теорію гібридних систем. Візуалізація динаміки процесів і статистична обробка випадкових параметрів є вбудованими і виконуються автоматично.

Система автоматизованого проектування *Modelling* є засобом візуального проектування дискретних і неперервно-дискретних імітаційних моделей (ДІМ і НДІМ). Розроблена аспірантами і студентами кафедри № 302 Московського авіаційного інституту, система *Modelling* є набором компонентів і шаблонів проектів для середовища програмування Borland Delphi.

Базова частина системи включає:

необхідний набір засобів формалізації: змінні стани імітаційної моделі (у т. ч. структурні, з можливістю задавання математичної моделі їх зміни у вигляді системи рівнянь алгебри або диференціальних),

візуальні засоби задавання графа взаємозв'язків між подіями моделі, засоби роботи зі списковими структурами даних, засоби генерації потоків випадкових величин;

інтерактивні засоби роботи з елементами моделі (як на етапі створення, так і по ходу моделювання);

засоби автоматизації збору й аналізу статистики, засоби її графічного представлення;

базова пробна версія засобів візуалізації ходу моделювання (у т. ч. з можливістю анімації);

інтерактивні засоби відладки (контроль ходу моделювання, управління точками останову, засоби візуального контролю стану елементів моделі, покрокова відладка і т. д.);

генератор звітів зі створюваних моделей (автоматичне створення звітів із зображенням схеми моделі й інформації про її структуру у форматах MS Word і HTML).

Головна сфера застосування системи імітаційного моделювання (CIM) *Modelling* – розробка імітаційних моделей і спеціалізованих CIM при вивченні і проектуванні складних комплексних систем, де використання стандартних засобів проектування (Mathlab, GPSS і подібних) незручне або неефективне.

Система *ProcessModel* (ProcessModel, Inc). Базуючись на сучасній технології імітації, *ProcessModel* містить в собі досвід багатьох років досліджень і розробок у галузі імітаційного програмного забезпечення. Разом з потужною базовою програмою, поставляються ще три додаткові компоненти: LIVE Animation (ЖИВА Анімація), OneStep Modeling (Моделювання за один крок) і Visual Staffing (Візуальна робота з ресурсами).

ProcessModell корисний для вдосконалення ділових процесів будь-якого типу включаючи обробку транзакцій, обслуговування покупців, виробничі і складальні операції, транспортні послуги. ProcessModel дозволяє уникнути неточностей при виробленні рішень, забезпечуючи користувача необхідними і надійними даними. Типові додатки ProcessModel включають: розклад роботи персоналу і графік робочого часу; пріоритетність і можливість переривання завдань; можливість вибору різних методів управління; планування ємкості операції або процесу; можливість задавання розміру партії обробки; розклад призначень; визна-

чення послідовності робіт; виробничий розклад; підвищення продуктивності; скорочення циклу обробки; зниження вартості; управління якістю; аналіз "вузьких місць"; оцінка вартості по операціях і ресурсах; розклад доступності ресурсів для перерв у роботі і простоїв.

ProcessModel дає можливість аналізу безлічі випробувань, підтримує визначувані користувачем змінні й атрибути, формує стандартні звіти, що настроюються користувачем.

ProcessModel дозволяє створювати ієрархічні моделі для кращої організації й управління великими модельними проектами. Групи розробників можуть створювати різні частини складної моделі, а потім, об'єднавши їх, моделювати процес в цілому.

Система *Arena* фірми System Modeling Corporation. Arena дозволяє будувати імітаційні моделі, програвати їх й аналізувати результати такого програвання. Оскільки імітаційне моделювання є універсальним засобом оптимізації процесів, моделі за допомогою Arena можуть бути побудовані для найрізноманітніших сфер діяльності – виробничих технологічних операцій, складського обліку, банківської діяльності, обслуговування клієнтів в ресторані і т. д. і т. п.

Імітаційна модель *Arena* включає такі основні елементи: джерела і стоки (Create і Dispose), процеси (Process) і черги (Queue). Джерела – це елементи, від яких у модель надходить інформація або об'єкти. Швидкість надходження даних або об'єктів від джерела зазвичай задається статистичною функцією. Стоком є пристрій для прийому інформації або об'єктів. Поняття черги близьке до поняття сховища даних – це місце, де об'єкти чекають обробки. Час обробки об'єктів (продуктивність) у різних процесах може бути різною. У результаті перед деякими процесами можуть накопичуватися об'єкти, які чекають своєї черги. Часто метою імітаційного моделювання є мінімізація кількості об'єктів у чергах. Тип черги в імітаційній моделі може бути конкретизований. Черга може бути схожа на стек – об'єкти, що прийшли останніми в чергу, першими відправляються на подальшу обробку (LIFO: last-in first-out). Альтернативою стеку може бути послідовна обробка, коли першими на подальшу обробку відправляються об'єкти, що прийшли першими (FIFO: first-in first-out). Можуть бути задані і складніші алгоритми обробки черги. Про-

цеси – це аналог робіт у функціональній моделі. В імітаційній моделі може бути задана продуктивність процесів.

Для побудови моделей *Arena* має набір засобів, які включають палітру інструментів, набір гідів тощо. Модулі типу Flowchart (зокрема Create, Dispose і Process) служать для відображення потоків об'єктів і можуть бути перенесені на робочий простір моделі drag-and-drop. Модулі типу Data (наприклад, Queue) служать для настройки параметрів моделі. Модуль Create є джерелом сутності в системі. Модуль Process відповідає за обробку сутності. Модуль Dispose є стоком сутності з системи.

16.3. Сфери застосування імітаційних моделей

У різних галузях техніки, в організації виробництва, в соціальній сфері й у військовій справі постійно виникає необхідність розвязання ймовірнісних задач, пов'язаних з роботою систем масового обслуговування різних видів запитів. Термін "масове обслуговування" передбачає багаторазову повторюваність ситуацій в тому чи іншому сенсі (багато заявок, що надійшли в систему і були обслуговані, велика кількість аналогічних систем, які перебувають в експлуатації) і статистичну стійкість картини. Висновки і рекомендації, отримані методами теорії масового обслуговування (ТМО), застосовні лише за наявності одного або обох названих факторів повторюваності.

За справедливим зауваженням видатного радянського фахівця у цій сфері Гнеденка Б. В., легше назвати ситуації, де не можна застосувати ТМО, ніж назвати всі сфери її потенційного застосування.

Наведемо деякі сфери застосування ТМО.

Техніка зв'язку. Методи ТМО використовуються для проектування телефонних станцій та мереж зв'язку.

Транспорт. Методи імітаційного моделювання досить успішно застосовуються при аналізі процесів дорожнього руху, проходження тунелів, черг біля світлофорів, технічного огляду автомашин, транспортних потоків на автомагістралях, вулицях міст, перехрестях, на окремих ділянках руху.

За допомогою моделей СМО можна описати роботу транспортних підприємств (наприклад пасажирські перевезення, перевезення вантажів, завантаження і розвантаження вантажного транспорту). Це дасть

можливість проаналізувати ефективність їх роботи, дозволить виявити недоліки в ній.

При моделюванні роботи залізничного транспорту можна визначити місце розташування додаткових колій, їх необхідну довжину та місця встановлення додаткових стрілкових переводів; розрахувати необхідну кількість вагонів і кількість локомативів.

Використавши імітаційні моделі можна промоделювати використання повітряного простору в зоні аеропорту, виявити ймовірності конфліктних ситуацій; промоделювати рух літаків в аеропорту (оцінити добову пропускну здатність, параметри затримок вильотів, час роботи аеропорту в режимі перевантження, час переміщення на льотовищі при злеті та посадці); проаналізувати процеси реєстрації пасажирів, диспетчерської служби аеропорту, резервування квитків.

Промисловість. Типові процеси виробництва зазвичай можна описати низкою моделей, які дозволять проводити аналіз вузьких місць, комплексне управління логістичними процесами, вибір ефективних стратегій управління запасами, оперативне і календарне планування тощо. Більшість відповідальних рішень приймається за результатами імітації реальних процесів для різних альтернатив.

ТМО у промисловості використовується при плануванні операцій збирання (розрахунок кількості ліній та місткості бункерів), гнучких автоматизованих виробництв, організації ремонту обладнання, роботи пунктів обслуговування (наприклад, інструментальних складів), розрахунку зон обслуговування (у ткацькому та прядильному виробництві), визначенні резервних забоїв у шахтах.

Ще один приклад – використання моделей СМО для аналізу роботи ремонтних служб. Основними факторами, які важливо врахувати при побудові моделі, можна назвати такі: кількість персоналу і технологічного устаткування, необхідного для виконання ремонтних робіт, наявність запасних деталей для заміни тощо.

За результатами моделювання можна зробити висновки про якість роботи реальних систем, виявити вузькі місця в їх роботі, ефективність і можливості покращення їх роботи.

Логістика. Під логістикою розуміють різні процеси переміщення матеріальних об'єктів. Там імітаційні моделі можна застосувати для

моделювання збиральних конвеєрів; систем транспортування вантажів усередені підприємства (крани, вантажчики тощо); стаціонарних та підвісних систем транспортування вантажів; складських процесів (приймання та переміщення вантажів, комплектація, пакування тощо); технологічних операцій, які виконуються роботами; ситуацій типу Crashtest (наслідків удару, зіткнення); фізичної зміни виробів при обробці.

Фінанси. У фінансовій сфері імітаційні моделі можна застосувати:

для оцінки ефективності і ступеня ризику маркетингових стратегій підприємства. Причому тут можна врахувати дії постачальників ресурсів і підприємств-конкурентів, вплив реклами на обсяги продажів продукції, коефіцієнти еластичності попиту;

для оцінки ризиків інвестиційних проектів;

для імітації фінансових торгів (прийнявши, що торгівля відбувається як подвійний аукціон).

Автоматизовані системи. ТМО використовується для оцінки своєчасності обслуговування заявок на обчислювальні роботи і підготовку даних, обробки результатів експериментів, управління технологічними процесами, при дослідженні продуктивності обчислювальних систем з асинхронно працюючими пристроями.

Інформатика. Імітаційні моделі СМО дозволяють промоделювати:

роботу розподілених обчислювально-управляючих систем; інформаційні потоки в корпоративній розподіленій системі;

роботу комутаційних систем з різними типами потоків викликів і часові затримки в них;

роботу локальних комп'ютерних мереж з різними топологіями;

динамічні процеси в розподілених базах даних комп'ютерних інформаційних систем тощо.

Торгівля. ТМО використовується для виначення кількості магазинів (торгових точок), продавців, фасувальників, вузлів розрахунку, торгових і касових автоматів.

Охорона здоров'я. Методи ТМО успішно застосовуються при визначенні необхідної кількості аптек, лікарняних ліжок, станцій і бригад швидкої допомоги, лікарів і молодшого медичного персоналу, потреб у діагностичній і лікувальній апаратурі, пропускної здатності спеціалізованих санаторіїв.

Комунальне господарство. Імітаційні моделі можна застосувати для моделювання й оцінки якості обслуговування обєктів ЖКГ (ремонт будівель, прибирання території, обслуговування ліфтів, ремонт тепломереж, мереж водопостачання та водовідведення, вивезення побутового сміття тощо).

Служба побуту. ТМО використовується при розрахунку мережі перукарень, майстерень різного призначення, кількості техніки і працівників на ній, транспортних засобів забезпечення послуг населенню, пунктів прийому комунальних платежів, кількості ремонтників в житлово-експлуатаційних конторах і трестах, співробітників бюро з працевлаштування, по обміну житлової площі тощо.

Органи юстиції і внутрішніх справ. ТМО використовується при розрахунку кількості судових органів та їх штатної чисельності, місткості виправно-трудових установ, чисельності співробітників слідчого апарату і оперативників, плануванні роботи контрольно-пропускних пунктів і пунктів митного огляду.

Наука й освіта. ТМО можна використати при дослідженні деяких видів природних процесів (реєстрація елементарних часток, фільтрація, дифузія, дія каталізаторів хімічних реакцій), запуску супутників зв'язку й обробці супутникової інформації, розрахунку кількості лабораторних установок, екзаменаторів (у тому числі автоматичних), можливостей навчально-дослідних виробництв, побутового і медичного обслуговування студентів; при проектуванні й аналізі роботи великих бібліотек.

Військова справа. ТМО використовується при проектуванні систем протиповітряної оборони (кожна ціль може розглядатися як "заявка" на обслуговування, тобто обстріл), організації охорони кордонів, розрахунку патрульних нарядів, в багатьох інших випадках типу розглянутих раніше, але застосовно до бойової і щоденної діяльності військ.

Зазначимо, що задачі вказаних типів виникає потреба розв'язувати не тільки при проектуванні наново створюваних систем і мереж обслуговування, але й у процесі експлуатації наявних: при збільшенні навантаження; зміні трудомісткості обробки заявок; виході з ладу, деградації чи модернізації техніки; зниженні кваліфікації персоналу; перегляді вимог до оперативності обробки заявок тощо.

Висновки

1. Вітчизняні вчені, створюючи мови і системи імітаційного моделювання, використовували закордонний досвід. Представники Київської школи імітаційного моделювання, очолюваної академіком АН СРСР Глушковим В. М., вели розробки і дослідження за різними напрямами у галузі імітаційного моделювання і зробили свій значний внесок у розробку і розвиток мов і систем імітаційного морделювання.

2. Основними вимогами, що висуваються до мови моделювання, є такі: мова моделювання має передбачати певну міру спрощення; має надавати користувачу можливість описувати систему блоками, що максимально відповідає тому понятійному апарату, який характерний для неї.

3. Теорію масового обслуговування можна використати майже у всіх сферах життя і діяльності людини.

Контрольні запитання та завдання

1. Охарактеризуйте сучасний стан імітаційного моделювання в Україні та за кордоном.

2. Дайте характеристику нових мов та систем моделювання.

3. У яких сферах людської діяльності застосовуються імітаційні моделі?

4. Наведіть приклади використання імітаційних моделей в економіці, інформатиці, комп'ютерних мережах, транспорті та інших сферах.

Рекомендована література

Основна

1. Бахвалов Н. С. Численные методы / Н. С. Бахвалов. – М. : Наука, 2000. – 286 с.

2. Бережная Е. В. Математические методы моделирования экономических систем : учебн. пособ. / Е. В. Бережная, В. И. Бережной. – 2-е изд. – М. : Финансы и статистика, 2006. – 432 с.

3. Венцель Е. С. Теория вероятностей / Е. С. Венцель, Л. А. Овчаров. – М. : Наука, 1969. – 368 с.

4. Ермаков С. М. Метод Монте-Карло и смежные вопросы / С. М. Ермаков. – Изд. 2-е. – М. : Наука, 1975. – 472 с.

5. Задачин В. М. Лабораторний практикум з навчальної дисципліни "Моделювання систем" : навч.-практ. посібн. / В. М. Задачин, І. Г. Конюшенко. – Харків : Вид. ХНЕУ, 2009. – 212 с. (Укр. мов).

6. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Моделювання систем" для студентів напрямку підготовки 0804 "Комп’ютерні науки" всіх форм навчання / укл. В. М. Задачин, І. Г. Конюшенко. – Харків : Вид. ХНЕУ, 2007. – 96 с.

7. Молчанов А. А. Моделирование и проектирование сложных систем / А. А. Молчанов. – К. : Вища школа, 1999. – 664 с.

8. Пономаренко В. С. Моделювання дискретних процесів : навч. посібн. / В. С. Пономаренко. – К. : ІСДО, 1993. – 180 с.

9. Робоча програма навчальної дисципліни "Моделювання систем" для студентів напряму підготовки "Комп'ютерні науки" всіх форм навчання / укл. В. М. Задачин, І. Г. Конюшенко. – Харків : Вид. ХНЕУ, 2008. – 44 с. (Укр. мов.).

10. Рыжиков Ю. И. Имитационное моделирование. Теория и технологии / Ю. И. Рыжиков. – М. : Альтекс–А, 2004. – 384 с.

11. Советов Б. Я. Моделирование систем / Б. Я. Советов, С. А. Яковлев. – М. : Высшая школа, 1998. – 319 с.

12. Томашевський В. М. Моделювання систем / В. М. Томашевський. – К. : Видавнича група BHV, 2005. – 349 с.

Додаткова

13. Васильев А. И. Имитационное моделирование информационных и вычислительных систем с использованием языка моделирования GPSS / А. И. Васильев. – Владивосток : Изд. ДВГТУ, 1998. – 48 с.

14. Васильев А. И. Имитационное моделирование систем массового обслуживания с использованием языка моделирования GPSS / А. И. Васильев, Н. Н. Хобта, И. В. Брязгин. – Владивосток : Изд. ДВПИ, 1984. – 36 с.

15. Вознесенский В. А. Статистические методы планирования эксперимента в технико-экономических исследованиях / В. А. Вознесенский . – М. : Статистика, 1974. – 192 с.

16. Гультяев А. MATLAB 5.2. Имитационное моделирование в среде Windows : практ. пособ. / А. Гультяев. – СПб. : КОРОНА принт, 1999. – 288 с.

17. Жерновий Ю. В. Імітаційне моделювання систем масового обслуговування : практикум / Ю. В. Жерновий. – Львів : Видавничий центр ЛНУ імені Івана Франка, 2007. – 307 с.

18. Кравченко П. П. Имитационное моделирование вычислительных систем средствами GPSS/PC / П. П. Кравченко, Н. Ш. Хусаинов. – Таганрог : ТРТУ, 2000. – 116 с.

19. Кузьменко В. М. Специальные языки программирования. Программные и инструментальные средства моделирования сложных систем / В. М. Кузьменко. – Харьков : Основа, 2001. – 124 с.

20. Кутузов О. И. Имитационное моделирование сетей массового обслуживания : учебн. пособ. / О. И. Кутузов, В. Н. Задорожный, С. И. Ол-зоева. – Улан-Удэ : Изд-во ВСГТУ, 2001. – 228 с.

21. Лычкина Н. Н. Имитационное моделирование экономических процессов : учебн. пособ. для слушателей программы eMBI / Н. Н. Лычкина. – М. : ГУУ, Ин-т информационных систем управления, 2005. – 152 с.

22. Ослин Б. Г. Технология имитационного моделирования систем массового обслуживания / Б. Г. Ослин. // В кн. : Материалы международной научно-технической конференции "Информационные системы и технологи". Т. 2. – Новосибирск : Изд-во НГТУ, 8 – 11 ноября 2000 г. – С. 320–325.

23. Руководство пользователя по GPSS World / пер. с англ. – Казань : Мастер Лайн, 2002. – 384 с.

24. Рыжиков Ю. И. Теория очередей и управления запасами : учебн. пособ. / Ю. И. Рыжиков. – СПб. : Питер, 2001. – 376 с.

25. Ситник В. Ф. Імітаційне моделювання : навч. посібн. / В. Ф. Ситник, Н. С. Орленко. – К. : КНЕУ, 1998. – 232 с.

26. Сытник В. Ф. Имитационное моделирование : учебн.-метод. пособ. / В. Ф. Сытник, Н. С. Орленко. – К. : КНЕУ, 1999. – 208 с.

27. Томашевский В. Н. Имитационное моделирование в среде GPSS / В. Н. Томашевский, Е. Г. Жданова. – М. : Бестселлер, 2003. – 416 с.

28. Томашевский В. Н. Имитационный проект автомобильного дорожного движения / В. Н. Томашевский, Д. С. Печенежский // Радиоэлектроника, автоматика, управление. – 2001. – № 1.

29. Томашевский В. Н. Моделирование дорожных знаков в имитационном проекте автомобильного дорожного движения / В. Н. Томашевский, Д. С. Печенежский. // Математическое моделирование. – 2001. – № 1(6).

30. Томашевский В. Н. Решение практических задач методами компьютерного моделирования / В. Н. Томашевский, Е. Г. Жданова, А. А. Жолдаков. – К. : Изд-во "Корнійчук", 2001. – 268 с.

31. Шеннон Р. Имитационое моделирование систем: искусство и наука / Р. Шеннон. – М. : Мир, 1978. – 418 с.

32. Шрайбер Т. Дж. Моделирование на GPSS / Т. Дж. Шрайбер ; пер. с англ. – М. : Машиностроение, 1980. – 592 с.

Ресурси мережі Internet

33. www.exponenta.ru
34. www.gpss.ru
35. www.minutemansoftware.com
36. www.model.exponenta.ru
37. www.model-im.narod.ru

Додатки

Додаток А
Таблиця А.1

Стандартні числові атрибути (СЧА)

Позначення	Призначення
1	2
Системні числові атрибути	
RNj	Число, що визначається j-м датчиком випадкових чисел (j = 1…999). Усі датчики генерують послідовність рівномірно розподілених випадкових чисел. Це ціле число змінюється від 0 до 999 включно, крім двох випадків його використання – як аргумент функції або елемент у змінній. У цих випадках RNj буде дробовим від 0 до 0,999999
C1	Поточне значення відносного часу. Автоматично змінюється системою й установлюється в 0 оператором CLEAR або RESET
AC1	Поточне значення абсолютного часу. Автоматично змінюється системою. Ця величина не змінюється під дією керуючого оператора RESET і встановлюється в 0 тільки під дією оператора CLEAR
TG1	Число, що дорівнює поточному значенню лічильника завершень
XN1	Повертає номер активного транзакта
Z1	Повертає розмір вільної оперативної пам'яті в байтах
M1	Час перебування в моделі транзакта, оброблюваного програмою в цей момент. Ця величина може змінюватися блоком MARK. Час перебування визначається так: M1 = поточне значення абсолютного часу – мітка часу генерації оброблюваного транзакта
PR	Пріоритет транзакта, оброблюваного в цей момент. Ця величина може змінюватися блоком PRIORITY
СЧА транзактів	
Pj	Значення параметра j поточного транзакта
MPj	Транзитний "параметричний" час (дорівнює різниці поточного абсолютного часу і значення, записаного в параметр транзакта)
A1	Номер групи, до якої належить активний транзакт
MBj	Прапорець синхронізації: = 1 – якщо транзакт у блоці j належить тій же родині, що й поточний транзакт; = 0 – у протилежному разі
СЧА блоків	
Nj	Загальне число входів транзактів у блок j
Wj	Поточне число транзактів у блоці j
СЧА багатоканальних пристроїв (БКП)	
Sj	Поточний вміст БКП j
Rj	Число вільних каналів БКП j
SRj	Коефіцієнт використання БКП j у тисячних частках
SAj	Середній вміст БКП j (ціла частина) за час моделювання
SMj	Максимальний вміст БКП j
SCj	Загальне число входів у БКП j
STj	Середній час перебування одного транзакта в БКП j
SEj	Ознака пустоти БКП j: 1 – повністю пустий; 0 – у протилежному разі
SFj	Ознака заповненості БКП j: 1 – заповнений; 0 – у протилежному разі
SVj	Ознака готовності БКП j: 1 – готовий; 0 – неготовий

Закінчення табл. А.1

1	2
СЧА обслуговуючих пристроїв	
Fj	Поточний стан пристрою j: 1 – якщо пристрій зайнятий; 0 – у протилежному разі
FIj	Ознака переривання пристрою j: 1 – пристрій перебуває в стані переривання (пристрій зайнятий транзактом, що перервав обслуговування іншого транзакта); 0 – у протилежному разі
FVj	Ознака доступності пристрою j: 1 – доступний; 0 – у протилежному разі
FRj	Коефіцієнт використання пристрою j
FCj	Загальне число входів у пристрій j
FTj	Середній час використання пристрою одним транзактом
СЧА черг	
Qj	Довжина черги j
QAj	Середня довжина черги j
QMj	Максимальна довжина черги j
QCj	Загальне число входів у чергу j
QZj	Число нульових входів у чергу j
QTj	Середній час перебування транзакта в черзі j, включаючи нульові входи
QXj	Середній час перебування транзакта в черзі j, не включаючи нульові входи
СЧА таблиць	
TBj	Обчислене середнє значення таблиці j
TCj	Обчислене середньоквадратичне відхилення для таблиці j
СЧА матриць і збережуваних величин	
Xj	j-те збережуване значення
MXj (a,b)	Вміст елемента матриці j, розміщеного в рядку a, стовпці b
СЧА обчислюваних об'єктів	
FNj	Обчислене значення функції j. Від значення функції береться ціла частина, за винятком тих випадків, коли це значення використо-вується як модифікатор у блоках GENERATE, ADVANCE або ASSIGN, або як аргумент функції
Vj	Обчислене значення змінної j
BVj	Обчислене значення булевої змінної j
СЧА списків і груп	
GNj	Поточне число членів у числовій групі j
GTj	Поточне число членів у групі транзактів з номером j
CHj	Поточне число транзактів у j-му списку користувача
CAj	Середнє число транзактів у j-му списку користувача
CMj	Максимальне число транзактів у j-му списку користувача
CCj	Загальне число транзактів, що входили у список
CTj	Середній час перебування транзакта в j-му списку користувача
LSj	Повертає стан логічного ключа j: 1 – встановлений; 0 – невстановлений

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Задачин Віктор Михайлович
Конюшенко Ірина Григорівна

МОДЕЛЮВАННЯ СИСТЕМ

Конспект лекцій

Відповідальний за випуск **Пономаренко В. С.**
Відповідальний редактор **Сєдова Л. М.**

Редактор **Дуднік О. М.**
Коректор **Байдак В. В.**

План 2010 р. Поз. № 106-К.
Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 16,75. Обл.-вид. арк. 20,94. Тираж прим. Зам. №

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи
Дк № 481 від 13.06.2001 р

Задачин В. М.
Конюшенко І. Г.

МОДЕЛЮВАННЯ СИСТЕМ

Конспект лекцій