Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель модуля управління імітаційної системи

(тема)

Виконав:
студент II курсу, групи СПзм-18-2
Наумов ІА.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: проф. Горбачов В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ ____________ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ____________
(підпис)
“____”______________ 20__ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Наумову Ігорю Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель модуля управління імітаційної системи

затверджена наказом по університету від “ 30 ” березня 2020 р. № 43 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2020 р.

3. Вхідні дані до роботи

VHDL

Модуль управління

Імітаційна системи

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної області

Об’єктно-орієнтовані моделі складних систем

Об’єктно-орієнтована концепція імітаційного моделювання

Модуль управління імітаційної системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (слайдів) 13 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд предметної області.	31.03.20-12.04.20	
	Аналіз посилань за темою роботи.		
2	Вибір та обгрунтування методики	13.04.20-21.04.20	
	дослідження		
3	Вибір інструментальних засобів	22.04.20-27.04.20	
4	Проведення експериментів	28.04.20-05.05.20	
	Оформлення ПЗ	06.05.20-10.05.20	
6	Подання атестаційної роботи	11.05.20-13.05.20	

Дата видачі завдання 30 березня 2020 р.

Студент ____________________
(підпис)

Керівник роботи ____________________ проф. Горбачов В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 105 с., 17 рис., 4 табл, 1 дод., 41 джерел.

МІКРОКОНТРОЛЕР, АРХІТЕКТУРА, РЕГІСТР, ЦИКЛ ШИНИ, ПРОГРАМНА МОДЕЛЬ, КОНВЕЙЄР.

Об’єктом дослідження є алгоритмічна модель складної дискретно-собітійної системи та створена на її основі структура імітаційної системи моделювання. Проведено аналіз характеристик електронних компонентів і запропоновано вимоги їх модельних системних рівнів. Розглянуті засоби технологічного опису електронних компонентів системного рівня на основі технологічного процесу. Розроблена структура та представлені основні компоненти об’єктно-орієнтованих імітаційних систем. Робочий алгоритм імітації дискретних системних процесових методів. Програмно реалізовані алгоритми: складання очевидних планових програм і алгоритм модифікацій тимчасової координати та управління очевидними умовами.

ABSTRACT

Master’s thesis: 105 pages, 17 figures, 4 tables, 1 appendics, 41 sources.

DISCRETE-EVEN SYSTEM, MANAGING PROGRAM of MODELING, SIMULATION MODEL, PROGRAM MODEL, ELECTRONIC SYSTEMS.

Object of research of the degree project is the algorithmic model complex discrete - even systems and creation on its basis of structure of simulation system.

The analysis of the characteristics of electronic components is released and the requirements to their models of a system level are offered. The means of the structural description of electronic components of a system level on a basis of processor of the approach are considered.

The structure and the basic components of object-oriented simulation systems are offered. The processor-based algorithm of simulation of discrete systems is developed.

The algorithms of scheduling of events, algorithm of modification of temporary coordinate, are creating.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RAM – оперативна пам’ять

ROM – пам’ять тільки для читання

OOM – об’єктно-орієнтоване моделювання

EOM – електронна обчислювальна машина

ОС – операційна система

ВСТУП

Рішення сучасних задач керування галузями народного господарства, проектування і дослідження технічних, економічних, організаційних та інших систем вимагає залучення фахівців різних профілів. Їх ефективне співробітництво можливе лише за умови наявності загальної методології, у рамках якої проводиться дослідження. Така методологія зветься «системний аналіз». Один з найважливіших інструментів системного аналізу – комп’ютерне моделювання.

Метод моделювання з успіхом застосовують у таких областях, як автоматизація проектування, організація роботи обчислювальних комплексів, організація роботи транспорту, сфера обслуговування, аналіз різних сторін діяльності людини (охорона навколишнього середовища, керування водними ресурсами, екологічні проблеми, енергетика тощо.), автоматизоване керування виробничими й іншими процесами. Важливо підкреслити, що моделювання використовується при проектуванні, створенні, впровадженні, експлуатації систем, а також на різних рівнях їхнього вивчення – починаючи від аналізу роботи елементів та закінчуючи дослідженням систем у цілому при їхній взаємодії з навколишнім середовищем.

Ще в недалекому минулому комп’ютерне моделювання застосовувалося тільки висококваліфікованими фахівцями в наукових дослідженнях і дорогих технічних проектах. Це були космічні і ядерні програми, проекти, зв’язані зі створенням нових засобів озброєнь. Поворотним моментом у технології комп’ютерного моделювання стала поява персонального комп’ютера, а також прогресивної методології об’єктно-орієнтованого моделювання (ООМ). ООМ інтенсивно розвивається, охоплюючи всі нові області і надаючи аналітикові нові й ефективні засоби проектування систем із заданими властивостями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація моделей

Перш ніж дати класифікацію моделей, відзначимо, що вихідним пунктом при побудові моделі реальної системи є опис.

Опис - сукупність відомостей про досліджувану систему й умови, за яких необхідно провести дослідження. Опис зображується звичайно у вигляді схем, текстів, формул, таблиць експериментальних даних, що характеризують передбачувану структуру і функціонування системи, а також навколишнє середовище.

Таким чином, опис являє собою першу спробу чітко викласти закономірності, характерні для досліджуваної системи. Опис звичайно самостійного значення не має, а служить основою для побудови моделі системи.

Через багатозначність поняття "модель" у науці і техніці зручно ввести визначення моделі як її характерну рису щодо опису. Модель відтворює опис з великими або меншими спрощеннями, що залежать від намірів дослідника і засобів, що є в його розпорядженні. При цьому повинен досягатися розумний компроміс між точністю відтворення і складністю необхідних для цього засобів.

За основу класифікації моделей буде покладено спосіб подання властивостей системи [4,5]. Попередньо всі моделі можна поділити на дві групи: матеріальні (фізичні) та математичні.

Фізичні моделі. Фізична модель – сукупність об’єктів фізичної природи, що відображають основні геометричні, фізичні і функціональні властивості досліджуваної системи. Фізичні моделі можуть збігатися або не збігатися за фізичною природою із системою-оригіналом. Можна виділити наступні типи фізичних моделей: натурні й аналогові.

Натурні моделі – це реальні системи, що досліджується. Їх називають ще макетами або дослідними зразками. Натурні моделі мають високий ступінь адекватності і збігаються по фізичній природі із системою-оригіналом, що забезпечує високу точність і вірогідність результатів моделювання.

Застосування методу повного макетування при проектуванні мікропроцесорних систем (МПС) є прикладом використання натурних моделей в галузі комп’ютерної техніки. Суть методу полягає в тому, що відлагодження технічних засобів, програмного забезпечення і їхньої спільної роботи можливе тільки після виготовлення макета МПС. Однак цей метод має ряд істотних недоліків [6]: потрібна розробка додаткових апаратно-програмних засобів для організації режимів налагодження на макеті, що є не тривіальною задачею і по трудомісткості іноді порівнянна з розробкою самого макета; динамічні характеристики макета в більшості випадків гірше відповідних характеристик реальної системи, що не гарантує працездатності реальної системи при будь-яких специфікованих умовах її функціонування.

Аналоговими моделями називаються системи, що мають фізичну природу, що відрізняється від системи-оригіналу, але схожі з оригіналом процеси функціонування. Обов’язковою умовою при цьому є однозначна відповідність між параметрами об’єкта, що досліджується і його моделлю, а також тотожність безрозмірних математичних описів процесів, що відбуваються у них. Для створення аналогової моделі потрібний математичний опис досліджуваної системи.

Як аналогові моделі використовують механічні, гідравлічні, пневматичні системи, але найбільш широке застосування одержали електричні й електронні аналогові моделі. У цих моделях сила струму або напруга є аналогами фізичних величин іншої природи. Особливістю аналогових моделей є їхня гнучкість і простота адаптації до зміни і виміру кількісних значень параметрів і характеристик системи, що досліджується.

Аналогові моделі використовуються при дослідженні засобів обчислювальної техніки на рівні логічних елементів і електричних ланцюгів, а також на системному рівні, коли функціонування системи описується, наприклад, диференціальними або алгебраїчними рівняннями.

Для ілюстрації застосування аналогових моделей розглянемо найпростіший приклад формалізації процесу функціонування двох елементарних систем різної фізичної природи: механічної (коливання маятника, рисунок 1.1,а) та електричної (коливальний контур, рисунок 1.1,б).

На рисунку 1.1, а $\P(t)$ – відхилення центра маси пружинного маятника від положення рівноваги A в момент часу t, m – його маса, $\Psi\P(t)$ – сила жорсткості, що діє на маятник з боку пружини (Ψ – коефіцієнт жорсткості пружини). Тоді рівняння малих коливань маятника має вигляд:

$$m\frac{d^2\P(t)}{dt^2} + \Psi\P(t) = 0. \tag{1.1}$$

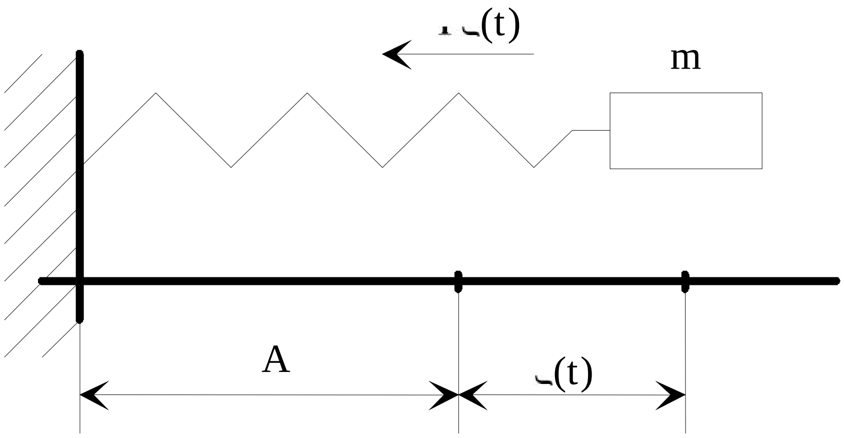
Позначивши $\mu_0 = \frac{\Psi}{m}$ і $\P(t) = Z$, можемо записати рівняння (1.1) у загальній формі рівняння вільних коливань:

$$\frac{d^2 Z}{dt^2} + \mu_0 Z = 0. \tag{1.2}$$

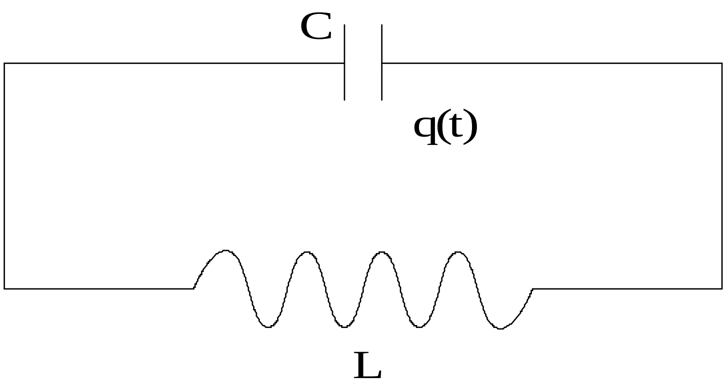
Розглянемо тепер вільні коливання в електричному контурі (рисунок 1.1,б). Якщо позначити ємність конденсатора C, його заряд у момент часу t через $q(t)$, а індуктивність котушки L, то рівняння коливань прийме вид:

$$L\frac{d^2 q(t)}{dt^2} + \frac{q(t)}{C} = 0.$$

Позначивши $\mu_0 = \frac{1}{LC}$, $q(t) = Z$, прийдемо знову до (1.2).



а)



б)

Рисунок 1.1 – Елементарні системи: а) пружинний маятник; б) коливальний контур

З наведених співвідношень виходить, що закономірності, властиві коливальному контурові (наприклад, залежність амплітуди і частоти коливань від його параметрів L та C), можна вивчати на моделі, що являє собою пружинний маятник, і навпаки.

Математичні моделі. До математичних моделей віднесемо концептуальні, аналітичні й імітаційні моделі.

Одним з етапів створення моделі є розробка концептуальної моделі системи.

Концептуальна модель відтворює опис в абстрактних термінах і поняттях, виявляючи основні (з погляду дослідника) причинно-наслідкові зв’язки, властиві досліджуваній системі, і які суттєві в рамках визначеного дослідження. Концептуальна модель системи описує тільки ті явища, що відбуваються в системі, що суттєві і повинні враховуватися при проведенні планованих досліджень із моделлю.

Наприклад, та сама комп’ютерна система може наводиться різними концептуальними моделями, що будуються в залежності від мети дослідження. Так, концептуальна модель багатопроцесорного комплексу, що відображає проходження потоків команд і даних, буде відрізнятися від моделі, що відображає використання оперативної пам’яті [5] .

Розробка концептуальної моделі вимагає досить глибоких знань системи, тому що треба обґрунтувати не тільки те, що повинно ввійти в модель, але і те, що може бути відкинуте без істотних спотворювань результатів моделювання. Концептуальна модель повинна, насамперед, забезпечити однозначне тлумачення понять і відносин, через які виражається мета дослідження, формується список задач дослідження і відбувається їхнє ув’язування в деяку узагальнену схему проведення досліджень.

Аналітична модель відтворює опис за допомогою абстрактної мови, зокрема за допомогою аналітичних співвідношень, що відбивають процес функціонування системи. До того ж рівень абстрагування залежить від тих питань, на які дослідник хоче одержати відповідь за допомогою моделі, і в якомусь ступені визначає вибір аналітичної моделі. Для складання моделі можна використовувати будь-які математичні засоби – алгебраїчне, диференціальне й інтегральне числення, теорію множин, теорію алгоритмів тощо.

Кожна конкретна система характеризується набором властивостей, під якими розуміються параметри, що відображають поведінку системи, що моделюється і враховують умови її функціонування у взаємодії з зовнішнім середовищем. Взаємодія системи та середовища показано на рисунку 1.2.

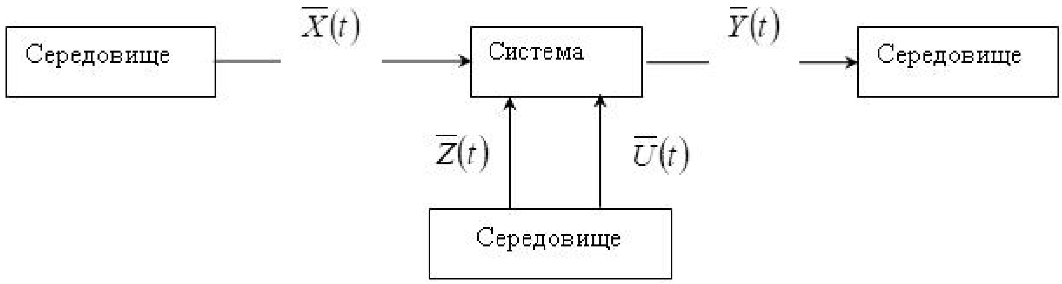


Рисунок 1.2 – Взаємодія системи і середовища

З рисунка 1.2 видно, що процес функціонування системи можна навести у вигляді безлічі взаємодіючих між собою параметрів зовнішнього середовища і системи:

$\overline{X}(t)$ – вектор вхідних параметрів, що впливають на систему. Значення цих параметрів можуть бути обмірювані, але можливість впливу на них відсутнє. Наприклад, інтенсивність вхідного потоку заявок.

$\overline{Z}(t)$ –вектор збуджуючих параметрів, значення яких випадковим образом змінюються в часі. Наприклад, зміна напруги в мережі, температури.

$\overline{U}(t)$ – вектор керуючих (або внутрішніх) параметрів системи, на які можна робити прямий вплив відповідно до вибору розробника або поставлених вимог. Наприклад, число процесорів, обсяг оперативної пам’яті.

$\overline{Y}(t)$ – вектор вихідних параметрів, значення яких визначається характером процесів, що протікають у системі. Ці параметри характеризують стан системи як результат сумарного впливу вхідних, керуючих і збуджуючих параметрів. Практично – це показники ефективності системи.

У загальному випадку $\overline{X}(t)$, $\overline{Z}(t)$, $\overline{U}(t)$, $\overline{Y}(t)$ є непересічними множинами і містять як детерміновані, так і стохастичні складові. При моделюванні системи $\overline{X}(t)$, $\overline{Z}(t)$, $\overline{U}(t)$ є незалежними, а $\overline{Y}(t)$ – залежними параметрами.

Аналітична модель системи являє собою сукупність співвідношень, що зв'язують вихідні параметри $\overline{Y}(t)$ з параметрами системи $\overline{U}(t)$, вхідними сигналами $\overline{X}(t)$ і $\overline{Z}(t)$.

Процес функціонування ідеалізованої системи можна описати в рамках поняття «стан» і «перехід зі стану в стан». Під станом системи будемо розуміти ознаку, що характеризує визначену розробником сукупність значень параметрів, що описують систему. Тоді стану системи можна подати у вигляді вектора:

$$\overline{S}' = \left(S_1', S_2', ..., S_k'\right), \ \overline{S}'' = \left(S_1'', S_2'', ..., S_k''\right)$$

тощо, де $S_i' = S_i(t')$, $S_i'' = S_i(t'')$ – значення і-тої ознаки в моменти часу, відповідно, t′ та t″.

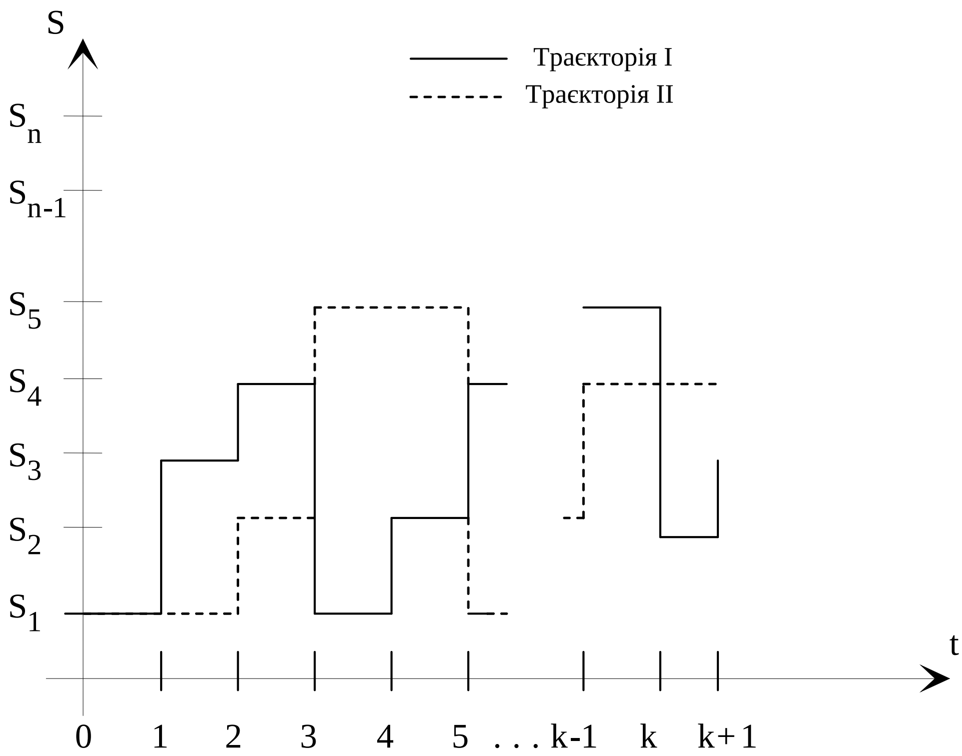


Рисунок 1.3 – Можливі траєкторії процесу

Сукупність усіх можливих значень станів називається простором станів об’єкта моделювання. Якщо розглядати процес функціонування системи як послідовну схему станів, то вони можуть бути інтерпретовані як координати точки в k-мірному фазовому просторі. Причому кожної реалізації процесу буде відповідати деяка фазова траєкторія (рисунок 1.3).

Відповідно до цього, моделювання - це подавання динаміки моделі за допомогою просування її від одного стану до іншого за заданими формальними правилами. Для опису динаміки імітаційної моделі необхідно скористатися співвідношеннями, що зв’язують характеристики станів системи $S_i(t)$ з її параметрами і часом, а також початковими умовами S_i^0 для початкового моменту часу t_0.

Таким чином, математичну модель системи можна надати у вигляді:

$$M = \{S^0, F, G, \overline{X}(t), \overline{Z}(t), \overline{U}(t)\}, \tag{1.3}$$

де F і G – відповідно оператори переходів та виходів такі, що:

$$\overline{S}(t) = G\left(\overline{X}(t), \overline{Z}(t), \overline{U}(t), \overline{S}(2), 2 < t\right) \tag{1.4}$$

$$\overline{Y}(t) = F\left(\overline{X}(t), \overline{Z}(t), \overline{U}(t), \overline{S}(2), 2 < t\right). \tag{1.5}$$

Власне кажучи, вираз (1.3, 1.4 або 1.5) є аналітичною моделлю досліджуваної системи. Дуже часто сукупність залежностей вихідних характеристик системи від часу $y_i(t)$ для всіх і називають вихідною траєкторією $\overline{Y}(t)$. У загальному випадку оператори F та G можуть бути задані у вигляді функції, функціонала, логічних умов, в алгоритмічній та табличній формах.

Імітаційна модель являє собою формальний, тобто виконаний на деякій формальній мові, опис логіки функціонування об’єкта дослідження, що відтворює як структуру досліджуваної системи, залежності, що існують між її параметрами, так і процеси функціонування системи в часі. Дуже часто імітаційна модель виражається у вигляді алгоритму.

Алгоритм – це четвірка: A =<S, LS, O, P>, обчислювальний процес якого моделює функціонування системи, що моделюється. Тут S - множина станів системи; l - множина областей значень, що можуть приймати ці стани; ОБ - множина операторів, кожний з яких розглядається як відображення значень станів у нові значення; P – множина спускових функцій операторів, кожна з яких Pj – розглядається як предикат, визначений у просторі значень станів, що приймає істинне значення у випадку, якщо оператор $Oj \in O$ може почати перетворення.

Процес зміни значень станів, здійснюваний операторами алгоритму, називається обчислювальним процесом. В імітаційній (алгоритмічній) моделі обчислювальний процес відповідає поведінці системи, що моделюється. Часто алгоритмічну модель називають імітаційною моделлю. Важливо відзначити, що при моделюванні динамічних систем в алгоритм імітаційної моделі необхідно вводити спеціальний стан, зміна якого відповідає перебігу часу. Такий елемент називається модельним часом. Спускові функції операторів, а отже, і послідовність перетворень значень станів залежать від модельного часу, забезпечуючи тим самим моделювання тимчасових категорій реального світу.

Математичні моделі можна класифікувати на детерміновані та імовірносні (стохастичні); статичні і динамічні; дискретні та безперервні.

Детерміновані моделі не містять елементів випадковості, тобто можна вважати, що в цьому випадку стохастичного впливу зовнішнього середовища $\overline{Z}(t)$ та стохастичних складових параметрів системи відсутні. Вихідні параметри системи визначаються, для цього випадку таким чином:

$$\overline{S}(t) = G\left(\overline{X}(t), \overline{Z}(t), \overline{U}(t), \overline{S}(\tau), \tau < t\right)$$

$$\overline{Y}(t) = F(\overline{X}(t), \overline{U}(t), \overline{S}(²), ² < t)\,. \qquad (1.6)$$

Очевидно, що детермінована модель є певним варіантом стохастичної моделі.

Статичні моделі служать для опису поводження системи в який-небудь момент часу, тому вирази (1.4 та 1.5) можуть бути записані так:

$$\overline{S}(t) = G(\overline{X}, \overline{Z}, \overline{U}, \overline{S})$$

$$\overline{Y} = F(\overline{X}, \overline{Z}, \overline{U}). \qquad (1.7)$$

Динамічні моделі є математичним описом поведінки системи в часі t, тобто відображають його динамічні властивості. Їхня узагальнена форма представляється співвідношеннями (1.4 та 1.5).

Розрізняють два основних типи динамічних систем: з дискретним станом; та безперервною множиною станів.

Якщо функція $S(t)$, що зв'язує характеристики станів системи з її параметрами і часом, може приймати лише кінцеве і лічильне число значень, має місце модель з дискретними станами або дискретна модель [28]. Наприклад, при дослідженні систем масового обслуговування в якості такої ознаки звичайно використовують число заявок у системі. Відповідно, зміна числа заявок у системі інтерпретується як перехід системи в новий стан. Для ідентифікації станів моделі можна використовувати натуральні числа. Так, якщо $S(t)$ знаходиться в стані S_j $(j = \overline{0,n})$ у момент t, говорять, що $S(t) = j$.

З іншого боку, якщо припустима безліч станів функції $S_i(t)$ являє собою кінцевий або нескінченний безперервний інтервал, говорять, що має місце модель з безперервними станами або безперервною моделлю. Прикладом безперервної множини станів може служити зміна форми краплі, що падає.

На практиці можливі також змішані випадки, коли деякі параметри системи можуть бути ідентифіковані як дискретні, а інші – як безперервні. Так у «житті» тієї ж краплі води, що утвориться в нещільно закритому крані, можна виділити як дискретні стани («висить» - «летить» -«впала»), так і безперервні (зміна форми під час падіння).

Зміна станів системи може відбуватися або у фіксовані моменти часу, множина яких дискретна або безупинна. Відповідно до цього розрізняють моделі з дискретним часом і моделі з безперервним часом зміни станів.

У загальному випадку час у моделі системи може розглядатися на інтервалі моделювання (O,T). Якщо час розглядається як дискретна величина, тоді $T = m\Delta t$, де Δt - відрізок одиниці часу, а m - число інтервалів дискретизації.

1.2 Основні етапи моделювання

З розвитком комп’ютерної техніки найбільш ефективним методом дослідження складних систем є моделювання. Тому однією з актуальної задач підготовки спеціалістів є освоєння теорії та методів моделювання в рамках системного підходу. При цьому розроблювач моделі повинний уміти не тільки будувати моделі систем, що досліджується, аналізувати їхню динаміку і можливість керування машинним експериментом з моделлю, але і судити про адекватність створюваних моделей, про границі її застосовності і правильно організувати моделювання системи.

Перш ніж розглядати математичні й імітаційні аспекти моделювання, необхідно враховувати загальні методологічні питання, характерні для практичного використання широкого класу моделей.

Процес створення моделі проходить кілька стадій. Він починається з вивчення системи, зовнішніх впливів і завершується розробкою або вибором моделі та комп’ютерної програми. Комп’ютерне моделювання припускає наявність наступних етапів створення і використання моделей [13,14]:

- складання змістовного опису об’єкта, що досліджується;
- розробка концептуальної моделі;
- розробка математичної моделі;
- калібрування і перевірка придатності моделі;
- програмування моделі;
- планування і проведення машинних експериментів;
- аналіз результатів моделювання.

Складання змістовного опису об’єкту, що досліджується. Цей етап виконується замовником і розробником моделі спільно. Здійснюється постановка задачі і складання технічного завдання на весь обсяг передбачуваних робіт. Сукупність зведень про об’єкт моделювання подається у вигляді схем, текстів, таблиць експериментальних даних, що характеризують передбачувану структуру і функціонування системи. У вихідній документації одночасно може бути перебільшення інформації про одні аспекти поводження системи і недолік інформації про інші. Існує також інформація про зовнішні впливи і навколишнє середовище. Як правило, інформації, представленої у вихідній документації, буває недостатньо для опису поводження складної системи і вивчення різних сторін її поводження. Найчастіше це і не потрібно. Задача може виявитися значно важчою. Тому необхідно спочатку чітко визначити мету майбутнього дослідження, а потім відповідно до цієї мети переробити весь обсяг вихідної інформації і заповнити відсутню інформацію. Такий процес назвемо складанням змістовного опису складної системи.

Рекомендується наступна послідовність дій при складанні змістовного опису складної системи: вибір показників ефективності, що відбивають мети моделювання; деталізація опису режимів функціонування системи; доповнення опису інформацією про зовнішнє середовище. На цьому етапі дослідник широко застосовує опитування інженерно-технічного персоналу або фахівців, що мають досвід роботи з прототипом проектованої або реально існуючої системи, поводження якої необхідно досліджувати.

Розробка концептуальної моделі. Маючи змістовний опис досліджуваної системи, розробник приступає до створення концептуальної моделі. При побудові простих моделей змістовний опис практично являє собою спрощене алгоритмічне відображення реальної системи.

Важливим етапом побудови концептуальної моделі системи є етап її декомпозиції. Відомо, що будь-яка система, у тому числі й обчислювальна - це насамперед цілісна сукупність підсистем і елементів. Неодмінною властивістю кожної системи є її поділення, тобто можливість подання її у вигляді сукупності підсистем і елементів [4]. У цю сукупність включаються всі підсистеми й елементи, що забезпечують цілісність системи. З іншого боку, кожна підсистема складається також із сукупності підсистем, що, у свою чергу, можуть бути розбиті на підсистеми тощо. Вибір рівнів декомпозиції (деталізації) залежить від цілей моделювання й обсягу попередніх знань про властивості підсистем. Декомпозиція досліджуваної системи дає ряд переваг розробнику: кожна підсистема може бути реалізована автономно (математично, експериментально та ін.); полегшується керування моделлю й ухвалення рішення про істотність або неістотність кожної підсистеми; підвищується ефективність робіт із програмування моделі (за рахунок їхнього розподілу).

Наступним кроком створення концептуальної моделі є вибір параметрів системи. При цьому необхідно керуватися таким. У модель повинні увійти всі ті параметри системи, що допускають варіювання в процесі моделювання, визначення характеристик, що цікавлять дослідника при конкретних зовнішніх впливах на заданому інтервалі часу функціонування системи. Інші параметри повинні бути по можливості виключені з моделі. Опис кожного параметра повинний даватися в наступній формі: визначення і коротка характеристика; ідентифікатор і одиниця виміру; діапазон змін і місце застосування в моделі.

Серед етапів побудови концептуальної моделі зупинимося на етапі побудови структури моделі з указівкою зв’язків, що відображають функціональну взаємодія елементів і підсистем, а також взаємодію основних процесів, що відбуваються у системах. На підставі виконання попередніх етапів розроблювач моделі одержує перелік взаємодіючих процесів, елементів і підсистем, а також перелік відповідних параметрів. Тепер на підставі даних про функціонування системи, процесів взаємодії її елементів і підсистем, встановлюються зв’язки між компонентами системи. Далі варто перевірити ці зв’язки на інформаційну забезпеченість.

Процес побудови концептуальної моделі завершується наступними діями: оцінка потреби використання обраного типу моделі; аналіз точності апроксимації функціональних дій у системі і відповідності прийнятих гіпотез; перевірка вірогідності вихідної інформації; перегляд постановки задачі на моделювання й оцінці майбутніх точнісних характеристик моделювання. Зазначена послідовність дій являє собою перевірку відповідності концепції моделі реальній системі.

Розробка математичної моделі. Концептуальна модель і кількісні дані є основою для розробки математичної моделі. Створення математичної моделі переслідує дві мети: дати формальний опис структури і процесу функціонування системи для однозначності їхнього розуміння; спробувати подати процес функціонування у вигляді, що допускає аналітичне дослідження системи.

Розробка єдиної методики побудови математичних моделей, мабуть, не можлива. Це обумовлено великим різновидом класів систем. Системи можуть бути статичні та динамічні, з постійною або тимчасовою структурою, з жорстким або гнучким програмним керуванням тощо. У практиці моделювання систем в галузі системотехніки і системного аналізу раціонально використовувати типові математичні схеми [13,14]: системи масового обслуговування, що моделюють дискретні і дискретно-безперервні стохастичні системи; мережі Петрі, що використовують при структуризації причинних зв’язків і моделюванні систем з паралельними процесами; скінченні та імовірнісні автомати, що використовують для моделювання дискретно-детермінованих систем; диференціальні рівняння, що є математичним апаратом безперервно-детермінованих систем; мережі кусково-лінійних агрегатів, що дозволяють описувати поводження безперервних та дискретних, детермінованих і стохастичних систем та ін.

Комп'ютерні системи є досить складним об'єктом моделювання. У цьому зв'язку вибір ступеня детальності моделі є однією з важливих проблем, яку необхідно буде вирішити розробнику. Більш детальна модель, очевидно, буде точнісною і матиме ширшу галузь застосування, ніж менш детальна. Однак, безсумнівно, її проектування, програмування, перевірка, документування і використання обходяться набагато дорожче. Таким чином, при виборі ступеня деталізації, розробник повинний знайти точний і важко визначуваний компроміс між вимогами дослідження і вартістю моделі. Наприклад, тимчасова здатність, дозволяючи, визначає найменший інтервал часу моделі між послідовними подіями. Іншими показниками детальності можуть бути число станів і типи подій, що розглянуті в моделі.

Побудова гнучкої моделі є ефективним способом вирішення проблем непередбаченого розвитку робіт з дослідження системи. Для цього модель повинна легко пристосовуватися до змін, яких вимагає процес калібрування[11], а також до змін, що відображають модифікації досліджуваної системи.

Калібрування і перевірка придатності моделі. Основне питання, що виникає при використанні методу моделювання, полягає в тому, наскільки добре модель відображує систему, яка моделюється або, іншими словами, модель адекватна досліджуваній системі. Адекватність – властивість моделі відображати з заданою точністю зв'язки та співвідношення досліджуваної системи [8]. Тільки стверджувана відповідь на це питання може додати результатам дослідження необхідну вірогідність. Модель, поведінка, якої дуже відрізняється від поведінки системи, що моделюється, практично не потрібна. Однак модель не повинна бути повним відображенням системи, інакше нема сенсу її створення. У процесі розробки моделі адекватність штучно порушуються в результаті створення концептуальної, аналітичної та (або) імітаційної моделі, а також у результаті ідеалізації зовнішніх впливів і режимів функціонування, виключення тих або інших параметрів та ін. Перелічені та інші фактори можуть стати причиною того, що результати моделювання будуть істотно відрізнятися від реальних.

Розглянемо методику оцінки адекватності моделі для випадку, коли модельована система існує.

Адекватність моделі будемо визначати оцінюючи точність відображення станів системи $\overline{S}$ (1.4) та вихідних параметрів системи $\overline{Y}$ (1.4). Під точністю розуміють гранично припустиму погрішність оцінки, наприклад $\overline{Y}$ на основі моделі:

$$\Delta y_i \le \Delta_i , \tag{1.8}$$

де $\Delta y_i = \left| y_i^0 - y_i^m \right|$ – відхилення i-го параметра оригіналу y_i^0 та моделі y_i^m;

Δ_i – гранично припустима величина відхилення Δy_i .

Таким чином, модель вважається адекватною, якщо $\overline{\Delta y} \le \overline{\Delta}$.

Це визначення точності схематично ілюструється на рисунку 1.3. Система S при вхідних впливах $\overline{X}$ має показники ефективності $\overline{Y}$. Показники ефективності можуть бути скалярами, скалярними функціями параметрів системи, розподілами або послідовностями значень величин, розглянуті як випадкові перемінні та ін. Щоб спростити наші міркування, припустимо, що показники ефективності існуючої системи, вироблені для порівняння з показниками моделі, не піддані помилкам. Однак відомо, що це, взагалі кажучи, невірно.

На рисунку 1.3 S' – це модель досліджуваної системи S. S', що керується моделлю вхідних впливів X', показує ефективність $\overline{Y'}$. Порівняння значень однойменних показників у $\bar{y}$ та $\bar{y}'$ дає міру точності моделі.

Якщо точність моделі незадовільна, то модель повинна бути змінена, а процес перевірки повторений. Ця операція називається калібруванням. Ітеративна процедура калібрування схематично показана на рисунку 1.4. Процедура закінчується, як тільки дослідники визначать точність моделі задовільною.

Однак практичне використання такої оцінки адекватності часто викликає великі труднощі.

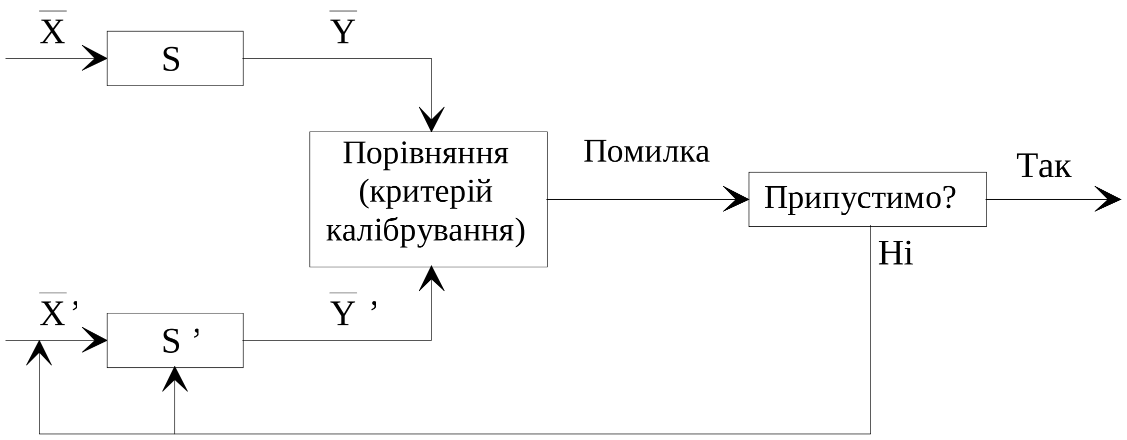


Рисунок 1.4 – Схема ітеративної процедури калібрування

По-перше, для систем, що проектуються або модернізуються, відсутня у (1.8) інформація про значення характеристики y^o. А моделюються, як правило, саме такі системи. Можна порівнювати характеристики моделі і деякої системи-аналога, але тоді буде однаковий ступінь недовіри до цих характеристик. По-друге, система оцінюється не по однією, а по безлічі характеристик, у яких може бути різна величина відхилення. По-третіх, характеристики можуть бути випадковими величинами і функціями, а часто і нестаціонарними функціями. Для стохастичних систем може виявитися, що статичні характеристики, отримані на моделі з високим ступенем адекватності, точними, за відповідні характеристики, обчислені за результатами вимірів на реальній системі. Це пояснюється тим, що результати моделювання визначаються за великим числом реалізацій, у той час як кількості вимірів на реальній системі завжди обмежені. По-четверте, відсутня можливість апріорного точного завдання граничних відхилень Δ. Проте перевіряти адекватність необхідно, тому що за невірними результатами моделювання можуть бути прийняті неправильні рішення.

На практиці оцінка адекватності, для даного випадку, звичайно проводиться шляхом експертного аналізу розумності результатів моделювання. Можна виділити наступні види перевірок:

- перевірка моделей елементів (у сумнівних випадках варто деталізувати елемент або провести додатковий аналіз);
- перевірка моделі зовнішніх впливів (прийняті припущення, апроксимації і гіпотези необхідно оцінити математичними методами);
- перевірка концептуальної моделі функціонування системи (виявляються помилки постановки задачі);
- перевірка математичної моделі;
- перевірка способів виміру й обчислення вихідних характеристик (виявляються помилки рішення);
- перевірка програмної моделі (аналізується відповідність операцій і алгоритмів функціонуванню програмної і математичної моделі, проводяться контрольні розрахунки при типових і граничних значеннях перемінних, виявляються інструментальні помилки програмування).

Модель не можна відкалібрувати для всіх реальних умов, у яких вона повинна використовуватися. Це практично неможливо через, узагалі говорячи, велике число таких умов і їхньої непередбачуваності. Більш того, якби було пророблено вичерпне калібрування з даними вимірів, модель виявилася б марної, оскільки більше не треба було б завбачати ефективність системи – вона була б оцінена експериментально. Модель звичайно використовується в умовах, у яких її точність не була, а часто і не могла бути перевірена. Таким чином, калібрування звичайно обмежене декількома комбінаціями вхідних параметрів системи. Якщо система існує і її можна виміріти, ці комбінації варто вибирати таким чином, щоб ці експерименти були здійсненні і по можливості прості. Модель під час калібрування повинна керуватися вхідними перемінними, що представляють обрані комбінації системних параметрів.

При заданих визначених вимогах до точності в термінах максимальних припустимих помилок, ми говоримо, що модель придатна для деякої вхідної умови, якщо вона задовольняє цим вимогам, коли її вхідні перемінні відповідають цій умові. Область придатності моделі є безліч вхідних умов, для яких придатна модель. Зрозуміло, розмір цієї області залежить від того, наскільки строгі вимоги точності. При калібруванні моделі ми домагаємося її придатності до вхідних умов, обраних для калібрування. Важким є питання, як можна визначити, чи придатна модель також і до тих умов, що зустрінуться при її практичному використанні?

Відповісти на це питання було б можна, якби була відома придатна завжди процедура перевірки. На жаль, це не так. Тому розробники моделей змушені прибігати до методів "для даного випадку", частковим тестам, а крім усього, до здорового глузду. Деяке відчуття стійкості моделі можна одержати, виконуючи після її калібрування небагато додаткових, ретельно відібраних порівнянь типу, показаних на рисунку 1.3 з використанням нових експериментальних даних. Якщо доступні дані не можна доповнити новими, розробники моделі можуть спробувати збільшити надійність моделі, використовуючи частину даних для її калібрування, а залишок — для часткової перевірки. Часто буває, корисна апостеріорна перевірка. Вона складається в порівнянні передбачувань моделі і результатів вимірів після того, як буде проведена яка-небудь зміна в системі, передбачена в дослідженні. Якщо модель виявиться прийнятно точною, упевненість користувачів у її стійкості виправдано зростає.

Інша, навіть більш важлива проблема полягає в тому, як проектувати досить стійкі моделі. У загальному випадку можна додержуватися принципу: чим ближче структура моделі до структури системи і чим вище ступінь детальності, тим більше область придатності моделі. На жаль, дуже докладна модель досить дорога.

Ступінь стійкості моделі можна підвищити ретельним калібруванням. У загальному випадку варто очікувати, що збільшення ступеня детальності не погіршить стійкість програми, що моделюється. Іноді, однак, у процесі калібрування існує точка, за якою стійкість може погіршитися через надмірне підстроювання моделі до приватних вхідних умов, що використовувались для її калібрування. На жаль, дуже важко визначити момент, коли ця точка досягнута.

Програмування моделі. На цьому етапі виконуються наступні дії.

Складається план створення і використання програмної моделі. Як правило, програма моделі створюється за допомогою засобів автоматизації моделювання на ЕОМ. Тому в плані вказуються: тип ЕОМ; засіб автоматизації моделювання; зразкові витрати пам’яті ЕОМ на створення програми моделі та її робочих масивів; витрати машинного часу на один цикл роботи моделі; оцінки витрат на програмування і налагодження програми моделі.

Потім дослідник приступає до програмування моделі. Технічним завданням на програмування служить опис математичної або імітаційної моделі. Специфіка робіт із програмування моделі залежить від засобів автоматизації моделювання, що доступні дослідникові. Не існує значних відмінностей створення програми моделі від звичайного автономного налагодження програмних модулів великої програми або пакета програм. Відповідно до тексту робиться розподіл моделі на блоки. На відміну від звичайного автономного налагодження програмних модулів, при автономному налагодженні блоків програмної моделі обсяг робіт істотно збільшується, оскільки для кожного модуля необхідно створити і налагодити свій імітатор зовнішнього оточення. Досить істотно вивірити реалізацію функцій модуля в модельному часі й оцінити витрати машинного часу на один цикл роботи моделі як функцію від значень параметрів моделі.

Завершуються роботи при автономному налагодженні компонентів моделі підготовкою форм подання вхідних і вихідних даних моделювання.

Далі переходять до другої перевірки вірогідності програми моделі системи. У процесі цієї перевірки встановлюється відповідність операцій у програмі та описі моделі. Для цього розробляється зворотний перехід програми до схеми моделі (ручне "прокручування"), що дозволяє знайти грубі помилки в програмі.

Після виключення грубих помилок ряд блоків поєднується і починається комплексне відлагодження моделі з використанням тестів. Відлагодження по тестах починається з декількох блоків, потім у цей процес залучають усе більше число блоків моделі. Відзначимо, що комплексне відлагодження програми моделі набагато складніше відлагодження пакетів прикладних програм, оскільки помилки динаміки моделювання в цьому випадку знайти значно складніше внаслідок квазіпаралельної роботи різних компонентів моделі. По завершенні комплексного відлагодження програми моделі необхідно знову оцінити витрати машинного часу на один цикл розрахунків на моделі. При цьому корисно одержати апроксимацію часу моделювання на один цикл імітації.

Наступною дією є складання технічної документації на модель складної системи.

Етапи планування машинних експериментів і аналіз результатів моделювання розглянуті в [6, 9] та їхній аналіз виходить за рамки цієї роботи.

1.3. Постановка задач дослідження.

Аналіз предметної області дозволяє підвести деякі підсумки. В процесі моделювання дослідник має справу з чотирма об’єктами: системою (реальної, проектованої), математичної (формальної) моделлю системи, алгоритмічної (машинної) моделлю і програмної реалізації машинної моделі. Формальна модель (або формальна схема) дискретно-подієвих систем архітектурного (системного) рівня будується як композиція поведінкових (функціональних) або структурних моделей. Тому тут необхідно вирішувати наступні завдання:

- визначення базових моделей (листя дерева), з яких будуються більші моделі;

Як ці моделі з'єднуються один з одним, утворюючи ієрархічну структуру. Досвід, накопичений в області моделювання, потреба в моделюванні все більш складних систем і можливості сучасних обчислювальних ресурсів зумовили появу нових поглядів на архітектуру і функції алгоритмічної моделі та її програмної реалізації. Це пов'язано з тим, що, по-перше, моделі складних систем, як правило, містять багато модулів відповідають різним елементам і підсистем об'єкта, що моделюється. Виникає запитання організації межмодульного інтерфейсу, яка є типовою для системного програмування. По-друге, окремі елементи і підсистеми можу бути описані за допомогою різних математичних схем, наприклад, звичайними диференціальними рівняннями, кінцевими автоматами, в термінах теорії масового обслуговування і т.д. По-третє, моделі складних систем можуть розроблятися багатьма людьми. Очевидно, що розробнику моделі повинні бути надані кошти дозволяють:

- замінити програмування конструюванням з готових елементів;
- розробку проблемно-орієнтованих бібліотек елементів;
- автоматичну компоновку елементів в алгоритмічну модель. У даній роботі проекті поставлено завдання розробки машинного алгоритму моделювання дискретних систем з урахуванням вимог, які були зазначені вище.

2 ОБ'ЄКТНО-ОРІЄНТОВАНІ МОДЕЛІ СКЛАДНИХ СИСТЕМ

2.1 Ієрархічні рівні представлення систем в об'єктно-орієнтованій технології

Основним методом аналізу та проектування складних систем є ієрархічний метод, або метод декомпозиції складного об'єкта на підсистеми (блоки). Той факт, що багато складних систем мають ієрархічну структуру, є головним чинником, який дозволяє вивчати такі системи. Ієрархія [15] - ранжування або упорядкована система абстракцій.

Абстракція [15]– це такі істотні характеристики деякого об'єкта, що відрізняють його від всіх інших видів об'єктів і, таким чином, чітко визначають особливості даного об'єкта з погляду подальшого розгляду й аналізу.

В об'єктно-орієнтованих технологіях моделювання використовуються два види ієрархії.

Ієрархія «ціле/частина» - показує, що деякі абстракції включені в розглянуту абстракцію як її частини, наприклад, лампа складається з цоколя, нитки накалювання і колби (мікроконтролер складається з). Цей варіант ієрархії використовується в процесі розбивки системи на різних етапах проектування: на логічному рівні - при декомпозиції предметної області на об'єкти, на фізичному рівні - при декомпозиції системи на модулі.

Ієрархія «загальне/частка» - показує, що деяка абстракція є окремим випадком іншої абстракції, наприклад, пам'ять, конкретні типи пам'яті: RAM, ROM. Використовується при розробці структури класів, коли складні класи будуються на базі більш простих шляхом додавання до них нових характеристик і, можливо, уточнення наявних.

Очевидно, що декомпозицію моделюємого об'єкта на кінцеве число підсистем здійснюють зі збереженням зв'язків між підсистемами, які забезпечують облік взаємодії підсистем. Якщо одержані в такий спосіб підсистеми все ще складні, метод декомпозиції застосовують для кожної з них. Процедуру розбивки підсистем продовжують до одержання таких підсистем, які в умовах розглянутої задачі будуть визнані досить простими і зручними для безпосереднього математичного опису.

Декомпозиція системи проводиться виходячи обраному рівню деталізації (абстракції) моделі, який, у свою чергу, визначається трьома факторами: цілями моделювання; обсягом апріорної інформації про систему; вимогами до точності та вірогідності результатів моделювання. Рівні деталізації іноді називають стратами, а процес виділення рівнів - стратифікацією.

Таким чином, у результаті послідовного застосування методу декомпозиції одержуємо багаторівневу ієрархічну систему компонентів (абстракцій), яку зручно зображати у виді дерева (рис 2.1). Очевидно, що різним рівням ієрархії абстракцій відповідають відповідні рівні цього дерева. На рівні листів дерева визначається поводження елементів проекту найнижчого рівня.

Наприклад, сучасний мікропроцесор містить кеш - пам’ять, ядро процесора і шини, до яких підключаються чипсети, контролери. Кожний з цих блоків, наприклад, ядро процесора, можна розкласти на більш прості складові: модуль з конвеєрною обробкою, пристрій керування, який у свою чергу можна також розкласти на більш прості елементи типу логічне «AND», інвертори, тригери і т.д.

Ми бачимо, що на кожному рівні деталізації існує своє поняття блоку або підсистеми, його закону функціонування і зовнішніх, стосовно цього блоку, впливів. Тоді під елементом системи будемо розуміти такий блок системи, що виконує визначену функцію (функції) і не підлягає декомпозиції на даному рівні деталізації. Множину типів елементів кожного рівня будемо називати базисом рівня. Вибір достатньої множини типів блоків для заданої предметної області є основною проблемою об’єктно-орієнтованного проектування.

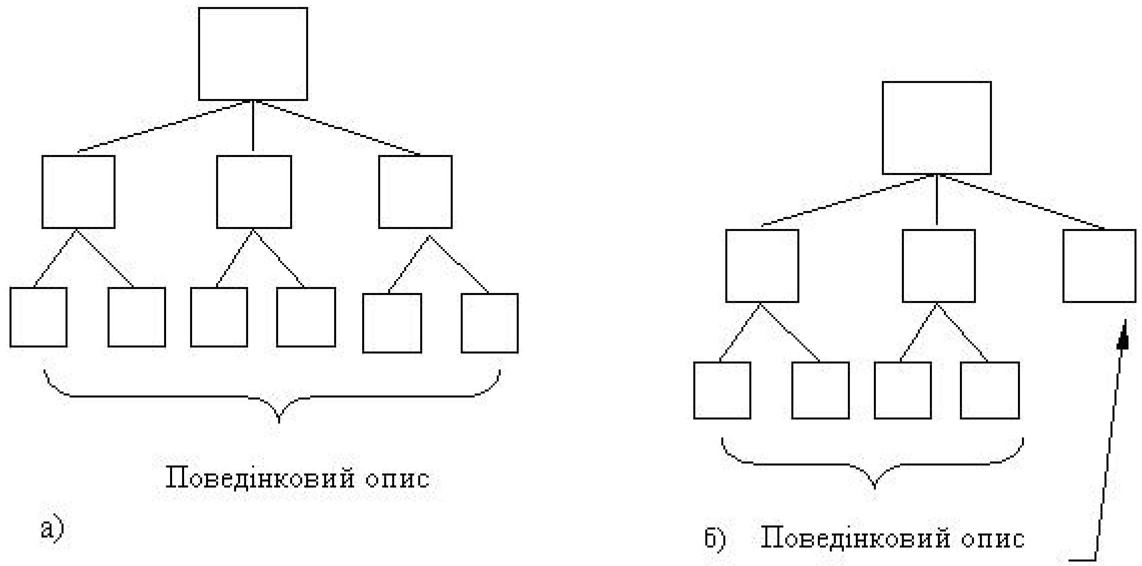


Рисунок 2.1 – Багаторівнева ієрархічна система блоків, представлена у виді (а) повного і (б) неповного дерева

Кількість рівнів ієрархії завжди обмежена. Підвищення рівня деталізації (зниження рівня абстракції) системи повинне проводитись до такого ступеня, щоб для кожного блоку були відомі або могли бути отримані залежності «вхід-вихід». Підвищення рівня деталізації опису системи дозволяє одержати більш точну її модель, але ускладнює процес моделювання і веде до росту витрат часу на його проведення.

Метод декомпозиції породжує серйозні проблеми при створенні САПР:

- визначення рівнів ієрархії та базисів для них;
- відображення з одного базису в інший та ін.;
- розробка формального апарату опису досліджуваної системи.

Застосування методу декомпозиції може базуватися на двох способах представлення системи та її компонентів: структурному і поведінковому (функціональному).

Структурний спосіб передбачає опис системи, підсистеми або блоку як сукупності взаємозалежних фізичних компонентів більш низького рівня, тим самим визначаючи базис цього рівня. Структурна форма ієрархії проекту має на увазі процес декомпозиції або розбивки проекту так, що на будь-якому рівні, який вибирається для моделювання, модель системи будується як сукупність взаємозалежних елементів, визначених для даного рівня.

Поведінковий спосіб передбачає опис системи, підсистеми, блоку або елемента по функціональних залежностях «вхід/вихід» за допомогою деякої формальної моделі. Для листів дерева проекту, тобто для елементів системи, цей опис визначається деякою власною формальною процедурою, а не описується з використанням інших процедур. Очевидно, що при структурному підході поведінкова модель використовується для опису елементів рівня листів дерева проекту.

Різні частини проекту можуть мати поведінкові описи на різних рівнях, тому поведінкова модель деякого проекту може існувати на будь-якому рівні. На рисунку 2.1(а) показане "повне" дерево проекту, де весь поведінковий опис формується на тому самому рівні. На рисунку 2.1(б) показаний проект, представлений у формі неповного дерева, де поведінкові описи відносяться до різних рівнів. Подібна ситуація виникає тому, що розробнику найчастіше бажано будувати й аналізувати взаємозв’язки між системними компонентами ще до завершення проектування. Таким чином, не обов’язково мати специфікації всіх системних компонентів, наприклад, на рівні логічних вентилів, щоб одержати можливість контролю проекту в цілому на відсутність помилок. Такий контроль здійснюється за допомогою багаторівневого моделювання, тобто моделювання, при якому поведінкові описи моделей компонентів відносяться до різних рівнів ієрархії. Важливим додатковим достоїнством подібного підходу є його сприяння підвищенню ефективності моделювання.

Структурні або функціональні компоненти системи, що володіють самостійною поведінкою (тобто здатні виконувати деякі дії, які залежать від отриманих повідомлень і власних станів), будемо називати об’єктом.

С точки зору розробника апаратних засобів можна виділити шість основних рівнів ієрархії [16], представлених на рисунку 2.2.

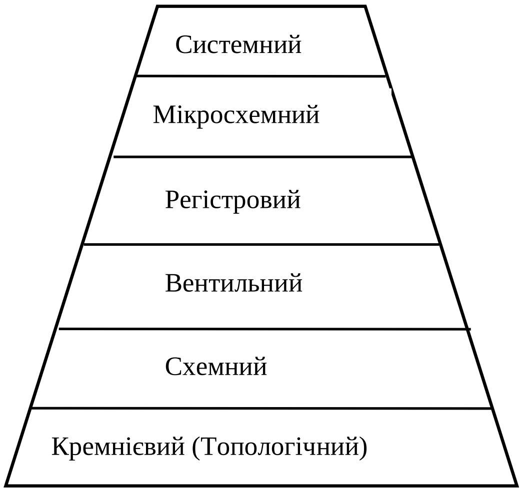


Рисунок 2.2 – Рівні ієрархії представлення електронних систем

Це системний, мікросхемний (або ІС), регістровий, вентильний, схемний і топологічний рівні. З рисунка видно, що ієрархія рівнів представлення має форму усіченої піраміди. Розширення піраміди донизу відображає збільшення ступеня деталізації, тобто кількості елементів, який необхідно враховувати при описі проектованого пристрою на цьому рівні.

У таблиці 2.1 приведена характеристика рівнів - вказуються елементи структури та поведінкове представлення для кожного рівня.

На самому нижньому рівні, кремнієвому, в якості базових примітивів використовуються геометричні форми, що представляють області дифузії полікремнію і металізації на поверхні кремнієвого кристала. Поєднання цих форм ніби імітує процес виготовлення кристала з погляду розроблювача.

На наступному, більш високому рівні, схемному, представлення проекту формується за допомогою з’єднання традиційних активних і пасивних елементів в електричній схемі: резисторів, конденсаторів і транзисторів. Поєднання цих компонентів використовується для моделювання поводження електричної схеми, яке виражається за допомогою залежностей між напругами і струмами. Для поведінкового опису на цьому рівні можна використовувати диференціальні рівняння.

Таблиця 2.1 – Ієрархія моделей

Рівень	Структурні примітиви	Формальний апарат для поведінкового представлення
Системний	Центральні процесори, комутатори, канали, шини, що запам'ятовуючі пристрої й ін.	Системний аналіз, теорія ігор, теорія масового обслуговування й ін.
Мікросхемний	Мікропроцесори, ЗУПВ, ПЗП, контролери, і ін.	Вхідні-вихідні залежності, ГСА
Регістровий	Регістри, АЛП, лічильники, мультиплексори, дешифратори	Теорія цифрових автоматів, таблиці істинності, ГСА
Вентильний	Логічні вентилі, тригери	Алгебра логіки, системи логічних рівнянь
Схемний	Транзистори, діоди, резистори, конденсатори	Теорія електричних ланцюгів, системи лінійних, нелінійних, диференціальних рівнянь
Кремнієвий	Геометричні об'єкти	немає

Третій рівень - рівень логічних вентилів, - традиційно відіграє основну роль при проектуванні цифрових схем і систем. Тут використовуються такі базові елементи, як логічні вентилі AND, OR і NOT та різні типи тригерів. Поєднання цих примітивів дозволяє обробляти комбінаційні та послідовні логічні схеми. Формальний апарат для поведінкового опису на цьому рівні - булева алгебра.

Вище вентильного рівня в ієрархії знаходиться регістровий рівень. Тут базові елементи - це такі компоненти, як регістри, лічильники, мультиплексори й арифметико-логічні пристрої (АЛП). Поведінкове представлення проекту на регістровому рівні можливо з використанням таблиць істинності, таблиць станів і мов регістрових передач.

Над регістровим рівнем знаходиться рівень мікросхем (або ІС). На мікросхемному рівні елементами виступають такі компоненти, як мікропроцесори, пристрої основної пам’яті, послідовні і паралельні порти та контролери переривань. Хоча границі мікросхем є і границями моделей елементів, можливі й інші ситуації. Так, набір мікросхем, що спільно утворюють один функціональний пристрій, можна представити як один елемент. Показовим прикладом тут може служити моделювання разрядно-модульного процесора. Можливий і альтернативний варіант - коли елементи представляють окремі секції однієї мікросхеми, наприклад, на етапі аналізу технічного завдання та декомпозиції. Головною особливістю тут є те, що елементом представляється великий блок логіки, де для довгих і найчастіше сходяться трактів обробки даних необхідно представляти залежності виходів від входів. Як і у випадку елементів нижчих рівнів, елементи мікросхемного рівня не будуються ієрархічно з більш простих примітивів, а являють собою єдині об’єкти-моделі. Так, якщо потрібно моделювати послідовний порт вводу-виводу (універсальний асинхронний прийомопередавач, УАПП), то відповідну модель не будують шляхом поєднання більш простих функціональних моделей таких блоків, як регістри і лічильники, тут сам УАПП стає базовою моделлю. Моделі такого типу важливі для виробників комплексного устаткування, які отримують мікросхеми в інших фірм-виробників, але не знають їхньої внутрішньої структури рівня логічних вентилів, оскільки це є звичайно секретом фірми. Поведінковий опис моделі мікросхемного рівня будується на основі вхідної-вихідної залежності кожної конкретної ІС-алгоритма, реалізованого даною ІС.

Верхній рівень - це системний рівень. Елементами цього рівня є - процесор, пам’ять і комутатор (шина) і ін. Поведінковий опис на цьому рівні включає такі основні дані і характеристики, як, наприклад, показник швидкодії процесора або пропускна здатність тракту обробки даних (біт/с).

З таблиці 2.1 видно, що структурні або поведінкові характеристики сусідніх рівнів деякою мірою перекриваються. Наприклад, і на регістровому і на мікросхемному рівні може використовуватися представлення за допомогою ГСА. Однак структурне представлення для обох рівнів відрізняється, тому вони і розділяються. Мікросхемний і системний рівні мають по суті ті самі елементи, однак вони абсолютно різні по своїх поведінкових характеристиках. Так, поведінкові моделі рівня ІС дозволяють обчислювати детальні окремі реакції у виді значень цілих чисел і бітів. А поведінковому представленню системного рівня властиве серйозне обмеження - воно служить переважно для моделювання пропускної здатності системи або визначення стохастичних параметрів системи. На практиці, представлення проекту на системному рівні використовується головним чином для порівняльної оцінки різних архітектур. Взагалі, варто використовувати моделі різного рівня, якщо вимоги або поведінкового, або структурного характеру різні.

2.2 Загальна характеристика взаємодії компонентів складних систем

Як було показано в попередньому розділі, складна система є багаторівневою конструкцією, яка складається із взаємодіючих елементів, поєднаних у підсистеми різних рівнів.

Взаємодію компонентів складної системи розглянемо на прикладі взаємодії електронних компонентів складних електронних систем у рамках механізму обміну сигналами. Сигнали передаються по каналах зв’язку, прокладеним між елементами складної системи. Початком даного каналу є вихід (вихідний порт) елемента, що видає сигнал, а кінцем - вхід (вхідний порт) елемента, що приймає сигнал. У роботі будемо розглядати ідеальні канали, тобто канали, що передають сигнали миттєво і без перекручувань (тобто характеристики сигналу на початку каналу і наприкінці його збігаються в будь-який момент часу). Це не є обмеженням, тому що не ідеальні канали необхідно розглядати як самостійні елементи системи, функціонування яких зводиться до відповідних затримок і перекручувань сигналів.

При побудові математичної моделі складної системи, необхідно враховувати взаємодію її з зовнішнім середовищем. У загальному випадку, зовнішнє середовище розглядається як деяка сукупність суб’єктів, які впливають на елементи складної системи, а також зазнають впливів, які надходять від елементів складної системи. Тому взаємодія з зовнішнім середовищем, може бути розглянута як деякий інтерфейс між елементами складної системи та суб’єктами зовнішнього середовища. Без утрати спільності можна розглядати модель відповідного каналу зв’язку аналогічну тій, котра описує канал, який реалізує взаємодію елементів усередині самої складної системи. Різниця складається тільки в тому, що, на відміну від елементів складної системи, суб’єкти зовнішнього середовища не є предметом дослідження і розглядаються тільки як джерела сигналів для елементів складної системи або адресати від елементів складної системи. У першому випадку характеристики сигналів повинні бути задані разом з іншими параметрами системи. В другому випадку характеристики вихідних сигналів елементів виробляються моделлю.

Механізм обміну сигналами, як математична модель взаємодії елементів складної системи між собою або з об’єктами зовнішнього середовища, включає набори наступних складових [17]:

- процес формування вихідного сигналу елементом, що видає сигнал;
- визначення адреси передачі для кожної характеристики вихідного сигналу;
- проходження сигналів по каналах зв’язку та компонування вхідних сигналів для елементів, які приймають сигнали;
- реагування елемента, який приймає сигнал, на вхідний сигнал, що надійшов.

Перша і четверта складові відносяться до побудови моделей функціонування самих елементів складної системи. Тут коротко необхідно відзначити, що елементи, які видають і приймають сигнали, є динамічними системами тих або інших класів, тому при побудові їхніх математичних моделей необхідно вибирати такі функцій (або оператори) переходів у новий стан та виходів для цих елементів, щоб забезпечувався досить адекватний опис поводження формалізуємих об’єктів з урахуванням їхньої взаємодії.

Функціональна модель елемента системи несе наступну інформацію:

- сукупність вхідних портів, через які надходять зовнішні сигнали;

- сукупність вихідних портів, через які відбувається видача зовнішніх сигналів;

- сукупність параметрів і змінних стану: звичайно присутні дві змінних стану – фазовий стан (фаза) і параметр сигма (при відсутності зовнішніх подій система перебуває в поточному фазовому стані протягом проміжку часу, обумовленого значенням параметра сигма);

- внутрішня функція переходу, що визначає наступний стан, у який система перейде після закінчення часу, що задається параметром сигма;

- функція виходів, яка генерує зовнішній вихідний сигнал, що безпосередньо передує внутрішньому переходові;

- функція тимчасового випередження, що задає часи внутрішніх переходів, при наявності змінної сигма ця функція просто видає значення сигми.

Третя складового механізму обміну сигналами пов’язана з урахуванням затримок і перекручувань при проходженні сигналів через неідеальні канали. Розгляд фізичного каналу зв’язку як самостійного елемента складної системи та побудова математичної моделі його у виді динамічної системи відповідного типу ґрунтується на результатах і висновках спеціальних наук, пов’язаних з теорією передачі інформації. Дана система є цілком формалізованою, отже, система має тільки ідеальні канали зв’язку.

Таким чином, нам залишається розглянути останню - другу складову математичної моделі взаємодії елементів складної системи - адресацію характеристик вихідних сигналів і їхнє компонування у вхідні сигнали елементів.

Практично, ця частина математичної моделі взаємодії елементів складної системи формалізує структуру системи, що моделюється, тобто описує, як ці моделі елементів з’єднуються один з одним, утворюючи ієрархічну структуру.

Структурна модель несе в собі наступну інформацію:

- сукупність вхідних портів, через які надходять зовнішні сигнали;

- сукупність вихідних портів, через які відбувається видача зовнішніх сигналів;

- з’єднання зовнішніх входів, тобто приєднання вхідних портів моделі системи до одного або більше вхідних портів елементів,– за рахунок цього одержувані моделлю системи вхідні сигнали адресуються визначеним моделям-елементам;

- з’єднання зовнішніх виходів, тобто приєднання вихідних портів елементів до вихідних портів моделі системи,– за рахунок цього згенеровані елементом вихідні сигнали можуть бути спрямовані на визначений вихідний порт моделі системи і передані на зовні;

- інтервальні з’єднання, що здійснюють з’єднання вихідних портів одних елементів із вхідними портами інших елементів, коли в одному з елементів виникне сигнал; його можна направити на вхідні порти заданих елементів (а також на вихідний порт усієї складеної моделі).

З одного боку, частина формалізованої схеми взаємодії - формування вихідних сигналів і реагування на вхідні сигнали, - включається безпосередньо в процес побудови математичних моделей елементів складної системи; з іншого боку, без цього не можуть бути правильно визначені множини вхідних і вихідних сигналів. Таким чином, побудова математичних моделей поводження елементів і їхніх зв’язків - єдиний процес; відрив одного від іншого практично неможливий.

2.3 Формальні схеми моделей елементів складних систем

Представлення об’єкта, що моделюється, у виді багаторівневої конструкції з елементів називається декомпозицією (структуризацією) об’єкта. У структурованій системі об’єктами матеріального світу являються елементи та зв’язки між ними. Об’єднання елементів у підсистеми – операція формальна, чисто структурна; вона не вносить яких-небудь нових об’єктів матеріального світу в сукупність елементів системи та зв’язків між ними і не виключає наявних.

Основний висновок, що випливає з принципу декомпозиції, полягає в наступному: математична модель складної системи складається з математичних моделей елементів і математичної моделі взаємодії між елементами.

У даному параграфі ми зосередимо свою увагу на математичних моделях елементів складної системи. Зовсім очевидно, що характер математичної моделі будь-якого об’єкта матеріального світу істотно залежить від його властивостей і особливостей функціонування. Однак, напевне, не менш важливу роль відіграють призначення моделі, ті дослідження, що передбачається на цій моделі проводити. Тому процес функціонування того самого об’єкта може бути описаний по-різному, у залежності від постановки задач моделювання.

Серйозну допомогу при побудові математичної моделі може надати володіння типовими математичними схемами, найбільше широко використовуваними в арсеналі моделювання.

Математичну схему можна визначити як ланку при переході від змістовного до формального опису процесу функціонування системи з урахуванням впливу зовнішнього середовища: тобто має місце ланцюжок «описова модель – математична схема – математична (аналітична або (та) імітаційна) модель».

У практиці моделювання об’єктів в області системного аналізу на початкових етапах дослідження системи раціонально використовувати типові математичні схеми: диференціальні рівняння, кінцеві та імовірнісні автомати, системи масового обслуговування тощо.

Вхідною інформацією при побудові математичних моделей процесів функціонування систем служать дані про призначення й умови роботи досліджуваної (проектованої) системи. Ця інформація визначає основну мету моделювання системи і дозволяє сформулювати вимоги до розроблюваної математичної моделі. Причому рівень абстрагування залежить від кола тих питань, на які дослідник системи хоче одержати відповідь за допомогою моделі, і в якомусь ступені визначає вибір математичної схеми.

Введення поняття «математична схема» дозволяє розглядати математику не як метод розрахунку, а як метод мислення, як засіб формулювання понять, що є найбільш важливим при переході від словесного опису системи до формального представлення процесу її функціонування у виді деякої математичної моделі (аналітичної або імітаційної). При користуванні математичної схеми в першу чергу дослідника системи S повинне цікавити питання про адекватність відображення у виді конкретних схем реальних процесів у досліджуваній системі, а не можливість одержання відповіді (результату рішення) на конкретне питання дослідження. Наприклад, представлення процесу функціонування інформаційно-обчислювальної системи колективного користування у виді 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 одержання результатів у явному виді.

Таким чином, при побудові математичних моделей процесів функціонування систем можна виділити наступні основні підходи:

- безперервно-детермінований (наприклад, диференціальні рівняння);
- іскретно - детермінований (кінцеві автомати);
- дискретно-стохастичний (імовірнісні автомати, марковські процеси);
- безперервно-стохастичний (системи масового обслуговування);

- агрегатний.

Безперервно-детерміновані моделі (D-схеми). Розглянемо особливості безперервно-детермінованого підходу на прикладі використання в якості математичних моделей диференціальних рівнянь. Диференціальними рівняннями називаються такі рівняння, у яких невідомими будуть функції однієї або декількох змінних, причому в рівняння входять не тільки функції, але й їхні 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.

Звичайно в таких математичних моделях у якості незалежної змінної, від якої залежать невідомі шукані функції, служить час t. Так як математичні схеми такого виду відображають динаміку досліджуваної системи, то вони називаються D-схемами (англ. dynamic system).

Використання D-схем [2] дозволяє формалізувати процес функціонування безперервно-детермінованих систем та оцінити їхні основні характеристики, застосовуючи аналітичний або імітаційний підхід, реалізований у виді відповідної мови для моделювання безперервних систем.

Дискретно - детерміновані моделі (F-схеми). Особливості дискретно - детермінованого підходу на етапі формалізації процесу функціонування систем розглянемо на прикладі використання теорії автоматів в якості математичного апарату. Теорія автоматів [18] – це розділ теоретичної кібернетики, у якому вивчаються математичні моделі – автомати. На основі цієї теорії система представляється у виді автомата, який переробляє дискретну інформацію та змінює свої внутрішні стани лише в припустимі моменти часу. Поняття «автомат» варіюється в залежності від характеру конкретно досліджуваних систем, від прийнятого рівня абстракції та доцільного ступеня спільності.

Автомат можна представити як деякий пристрій (чорна шухляда), на яке подаються вхідні сигнали і знімаються вихідні і яке може мати деякі внутрішні стани. Кінцевим автоматом називається автомат, у якого множина внутрішніх станів і вхідних сигналів (а отже, і множини вихідних сигналів) є кінцевими множинами.

Абстрактно кінцевий автомат (англ. finite automat) можна представити як математичну схему (F-схему), яка характеризується шістьма елементами: кінцевою множиною X вхідних сигналів (вхідним алфавітом); кінцевою множиною [Y] вихідних сигналів (вихідним алфавітом); кінцевою множиною Z внутрішніх станів (внутрішнім алфавітом або алфавітом станів); початковим станом z0, $z0 \in Z$; функцією переходів $\varphi(z, x)$; функцією виходів $\psi(z, x)$. Автомат, що задається F-схемою:

$$F = \langle Z, X, Y, \varphi, \psi, z0 \rangle.$$

Функціонує в дискретному автоматному часі, моментами якого є такти, тобто межуючі друг з другом рівні інтервали часу, кожному з яких відповідають постійні значення вхідного та вихідного сигналів і внутрішні стани.

Таким чином, поняття F-автомата в дискретно-детермінованому підході до дослідження на моделях властивостей об’єктів є математичною абстракцією, зручною для опису широкого класу процесів функціонування реальних об’єктів в автоматизованих системах керування. Такими об’єктами в першу чергу варто назвати елементи та вузли комп’ютеру, пристрою контролю, регулювання та керування, системи тимчасової і просторової комутації в техніці обміну інформацією і т.д. Для всіх перерахованих об’єктів характерна наявність дискретних станів і дискретний характер роботи в часі, тобто їхній опис за допомогою F-схем є ефективним. Але широта їхнього 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.

Дискретно-стохастичні моделі (Р-схеми). Розглянемо особливості побудови математичних схем при дискретно-стохастичному підході до формалізації процесу функціонування досліджуваної системи. Оскільки сутність дискретизації часу при цьому підході залишається аналогічною розглянутим кінцевим автоматам, то вплив фактора стохастичності простежимо також на різновиді таких автоматів, а саме на ймовірносних (стохастичних) автоматах.

У загальному виді ймовірносний автомат (англ. probabilistic automat) можна визначити як дискретний потактний перетворювач інформації з пам’яттю, функціонування якого в кожному такті залежить тільки від стану пам’яті в ньому та може бути описане статистично [19].

Застосування схем ймовірносних автоматів (Р-схем) має важливе значення для розробки методів проектування дискретних систем, які виявляють статистично закономірне випадкове поводження, для з’ясування алгоритмічних можливостей таких систем і обґрунтування границь доцільності їхнього використання, а також для розв’язання задач синтезу за обраним критерієм дискретних стохастичних систем, що задовольняють заданим обмеженням.

Безперервно-стохастичні моделі (Q-схеми). Особливості безперервно-стохастичного підходу розглянемо на прикладі використання в якості типових математичних схем систем масового обслуговування, (англ. queueing system), які будемо називати Q-схемами. Системи масового обслуговування являють собою клас математичних схем, розроблених у теорії масового обслуговування і різних додатках для формалізації процесів функціонування систем, що по своїй суті є процесами обслуговування. У розділах 3 та 4 даного посібника докладно розглядаються теоретичні та практичні аспекти Q-схем.

Процесам обслуговування можуть бути представлені різні по своїй фізичній природі процеси функціонування економічних, виробничих, технічних і інших систем, наприклад: потоки постачань продукції деякому 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 їхнього функціонування.

Перераховані типові математичні схеми, природно, не можуть претендувати на можливість опису на їхній базі всіх процесів, що відбуваються у великих інформаційно-керуючих системах. Для таких систем у ряді випадків більш перспективним є застосування об’єктно-орієнтованих моделей [21-23].

До класу об’єктно-орієнтованих моделей відносяться агрегатні моделі, котрі дозволяють описати широке коло об’єктів дослідження з відображенням системного характеру цих об’єктів. Саме при агрегатному описі складний об’єкт (система) розчленовується на кінцеве число частин (підсистем), зберігаючи при цьому зв’язки, що забезпечують взаємодію частин.

Агрегатні моделі (А-схеми). Найбільш відомим загальним підходом до формального опису процесів функціонування систем є підхід, запропонований Н. П. Бусленко [12]. Цей підхід дозволяє описувати поводження безперервних і дискретних, детермінованих і стохастичних систем, тобто в порівнянні з розглянутими є узагальненим (універсальним) і базується на понятті агрегатної системи (англ. aggregate system), що представляє собою формальну схему загального виду, яку будемо називати А-схемою.

2.4 Процеси як засіб опису складних електронних систем на системному рівні

У цьому розділі розглянемо формальну схему, названу процесом, яку будемо використовувати для побудови поведінкових моделей складних систем. Головним стимулом застосування таких формальних схем є уніфікація формального опису як системи, що моделюється так і імітаційних систем.

Наведемо процес, відразу відзначивши, що ідеального визначення поки не існує. Є всі підстави стверджувати, що архітектура сучасного комп’ютеру багатопроцесорна, якщо під процесором будемо розуміти будь-який пристрій у складі комп’ютеру, здатний автоматично виконувати припустимі для нього дії в деякому визначеному порядку. Тоді крім центрального процесора можна назвати процесором контролер дисків, мережну або відеокарту або будь-який інший апаратний ресурс, що входить до складу комп’ютеру. Між процесорами в системі існують інформаційні і керуючі зв’язки.

Тоді стосовно процесора *процесом* будемо називати деяку діяльність або дії (обчислювальні) у визначеному порядку (відповідно до алгоритму), що виконуються процесором.

Стан процесу. У період свого існування процес проходить через ряд характерних дискретних станів. Зміну станів (рисунок 2.3) процесу можуть викликати різні події. Прийнято розрізняти наступні характерні окремі стани [20]:

- породження – підготовлюються умови для першого виконання процесу. Створення або породження процесу складається з багатьох операцій, включаючи: присвоєння імені процесові; включення цього імені в список імен процесів, відомих системі; визначення початкового пріоритету процесу; виділення процесові початкових ресурсів.

- готовність – процес не виконується, але для його виконання надані всі необхідні в сучасний момент ресурси;

- активний стан, або стан виконання – процес виконується, наприклад, на деякому процесорі;

- чекання або блокування – раніше виконуваний процес не виконується, а очікує появи деякої події, щоб одержати можливість продовжити виконання;

- закінчення - нормальне або аварійне завершення виконання процесу, після якого ресурси йому вже не надаються.

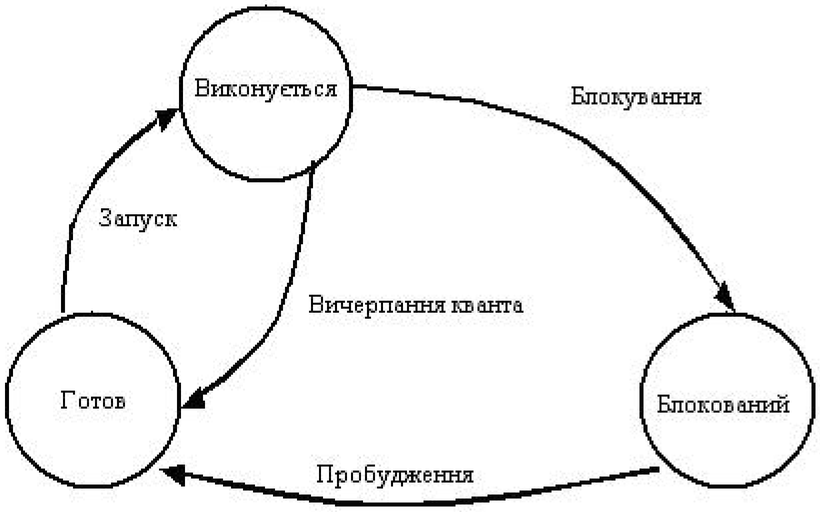


Рисунок 2.3 – Діаграма переходів процесу зі стану в стан

Графи процесів. Застосування поняття процес як формальної моделі елементів системи дозволяє застосувати теорію графів для опису й аналізу структури досліджуваних систем.

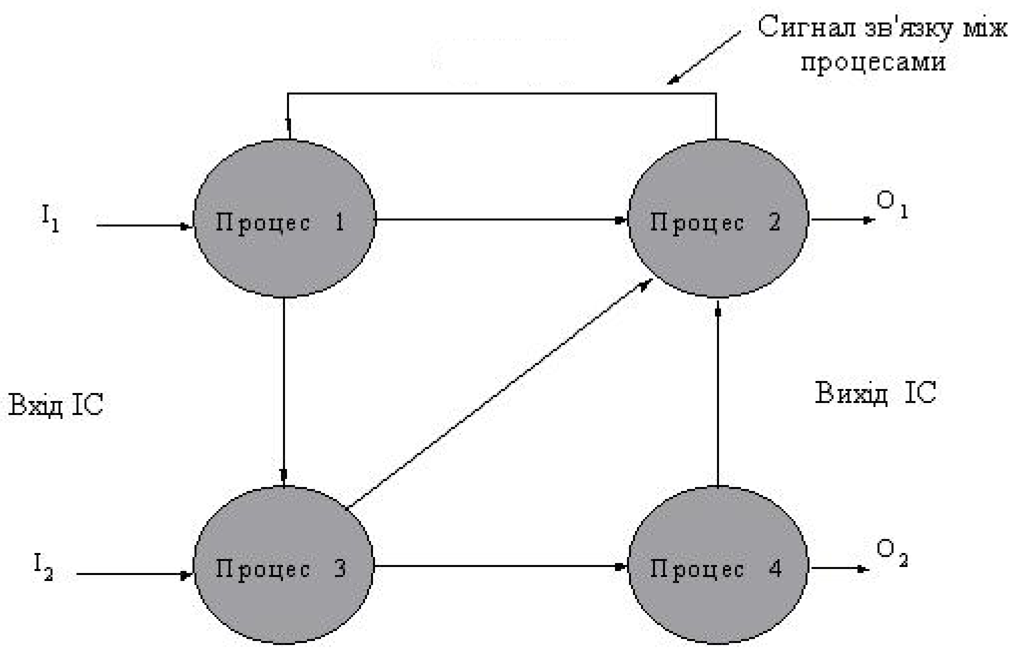


Рисунок 2.4 – Граф процесів

На рисунку 2.4. показане представлення типової моделі системного рівня у виді графа. Цей граф відображає розбивку функцій моделі на підфункції, причому кожен вузол графа являє собою деяку підфункцію - процес. Дуги графа відповідають шляхам проходження сигналів між вузлами-процесами. Кожна дуга може мати мітку-позначення виду S, де S - ім’я сигналу. Таке представлення моделі системного рівня у виді графа ми будемо називати графом процесів. Це дуже корисне представлення для моделей системного рівня, оскільки воно дає нам можливість моделювати складні потоки сигналів у рамках моделі при використанні високого рівня представлення. Приклади використання подібного графа для моделювання приводяться нижче.

Описаний вище граф моделі процесів - це вид графа з керуванням, обумовленим сигнальними ланцюгами IC і відповідними затримками. Однак моделі можуть, крім того, мати функціональну розбивку. Таку функціональну розбивку можна зробити і для вузлів графа моделі процесів (рисунок 2.5).

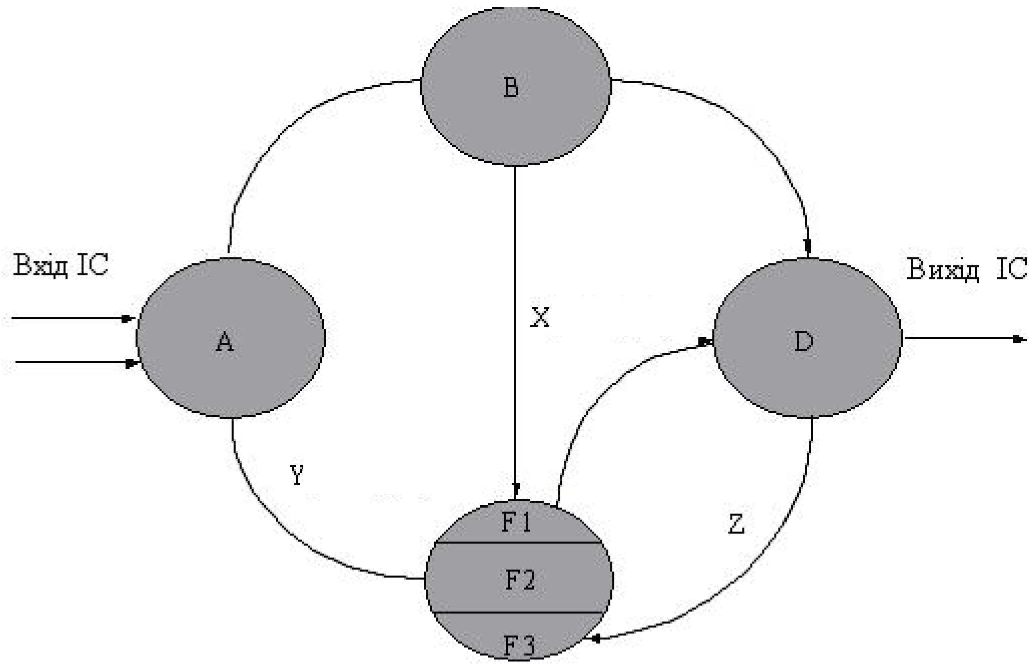


Рисунок 2.5 – Функціональна розбивка

Для вузла C зроблена декомпозиція на три функції: F1, F2, і F3. Кожній функції можна призначити відповідний процес. Таким чином, модель вузла C, при моделюванні мовою VHDL, може бути описана як

```
process (X, Y)   ---- Функція F1
   --------------
   --------------
end process

process (X, Z)   ---- Функція F2
   --------------
   --------------
end process

process (Y, Z)   ---- Функція F3
   --------------
   --------------
end process
```

Приклад 2.1 – Приклад програми на VHDL

Розглядаючи граф на рисунок 2.5, відзначимо, що вузол C має три входи: X, Y і Z, і що в приведеному вище прикладі кожний із процесів вузла C активується своєю підмножиною цих входів.

У деяких моделях функціональна розбивка є єдиною формою використовуваної розбивки.

Формальне визначення процесу. Процес може бути представлений у виді багатополюсника (рисунок 2.6), що функціонує в часі, причому час приймає значення $t \in T, T = [0, \infty)$. Процес має входи, виходи, внутрішні стани; кожен процес виконує визначений алгоритм, який задає поводження моделюємої функції системи та характеризується режимом роботи. Процеси моделі можуть виконуватися паралельно в часі, обмінюючись інформацією за допомогою каналів зв’язку.

Входи процесу поділяються на дві групи: керуючі й інформаційні. Інформаційні входи $x = (x_1, x_2, ..., x_n)$ приймають значення з алфавіту X и є функцією часу $x(t)$. Керуючі входи $c = (c_1, c_2, ..., c_l)$ приймають значення з алфавіту C и також є функцією часу $c(t)$. Для спрощення опису, будемо вважати, що всі сигнали x_i і c_i є елементарними, тобто не можуть бути представлені у виді множини каналів (шин). Якщо процес працює зі складними сигналами, вони представляються у виді декількох елементарних сигналів. Алфавіти X і C у нашому випадку є об’єднаннями алфавітів кожного з елементарних сигналів. На практиці ж, алфавіти елементарних сигналів можуть виявитися різними, тому потрібно говорити про множину алфавітів для окремих сигналів.

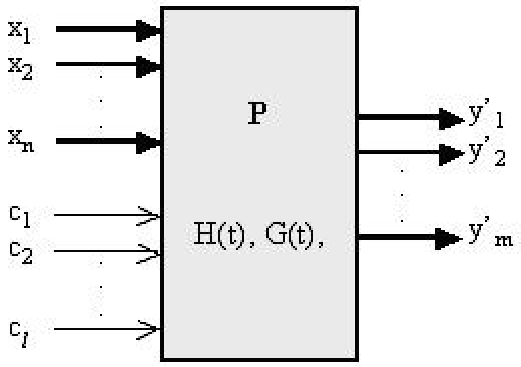


Рисунок 2.6 – Модель процесу

Розбивка сукупності вхідних сигналів на дві основні групи виконана у відповідності до їхнього впливу на функції системи. Одні сигнали просто представляють проходження в системі даних на різних стадіях обробки. Це інформаційні сигнали. Поводження системи мало залежить від того, яка інформація проходить по відповідних каналах. Справді, які б значення не були подані на входи x, вони все рівно будуть прийняті процесом і потрібним образом оброблені. Результати обчислень у будь-якому випадку будуть записані у внутрішню пам’ять і передані на вихід. Логіка роботи системи істотних змін не перетерплює при різних значеннях на цих входах.

Інша ситуація спостерігається при розгляді $c(t)$. Крім того, що ці сигнали також передають оброблювані дані, вони визначають моменти запуску або, навпаки, припинення виконання тих або інших функцій процесу.

Таким чином, можна сказати, що інформаційні сигнали синхронізуються керуючими.

На відміну від входів, виходи процесу не мають розподілу на керуючі й інформаційні, чому – це стане зрозумілим трохи пізніше. Вихід також є вектором $y' = (y_1', y_2', ..., y_m')$, що представляє собою функцію часу $y'(t)$. Кожен вихідний елементарний сигнал є структурою, що описується трійкою:

$$y_i' = (y_i, \Delta t_i, a_i),$$

де y_i - наступне значення виходу та процесу. Сигнали y_i набирають значення з алфавіту Y.

Δt_i - значення інтервалу часу, через який на виході і-ого процесу повинне встановитися значення $y_i . \Delta t \in T, T = [0, \infty)$;

a_i - вектор додаткових атрибутів, які впливають на спосіб моделювання затримки на даній лінії. Цей параметр буде розглянутий трохи пізніше. $a_i \in A$.

Також процес оперує локальними (внутрішніми) даними, що представляються змінними в традиційному їхньому розумінні. У якості локальних даних процесу можуть бути внутрішні стани, які характеризують можливі режими роботи процесу, момент часу, на якому припинений його внутрішній алгоритм, вказівник поточної виконуваної операції алгоритму процесу (IP) і т.д. Будемо позначати внутрішні стани процесу вектором $\bar{z} = (z_1, z_2, ..., z_k)$. Елементи цього вектора набирають значення з алфавіту Z і називаються фазовими координатами. Зі зміною часу t стани також змінюються, тобто є функцією часу $z(t)$. У цьому випадку вони називаються фазовими траєкторіями процесу.

Як уже було сказано, у кожен момент часу процес знаходиться у визначеному режимі роботи $m_i \in M, M = \{active, sleep, ...\}$, де active –

активний стан процесу, в якому відбувається виконання його алгоритму, sleep – стан чекання процесу і т.д.

В активному стані кожен процес характеризується поводженням, яке задається імітаційним алгоритмом. Форма представлення алгоритму може бути довільною: аналітичні вирази, операторні вирази, графи, таблиці, словесні описи, програма мовою моделювання і т.д. В імітаційному моделюванні найбільш поширена операторна форма представлення алгоритму – форма представлення у виді так званого операторного виразу. Операторний вираз [17] являє собою спеціально організовану послідовність логічних умов і операторів (дій). Оператори можуть бути простими, умовними і циклічними. Прості оператори виконують різні операції по перетворенню даних і позначаються прописними буквами латинського алфавіту, наприклад, D_1, D_2 і т.д. Послідовності операторів на листі розділяються комами. Умовні оператори складаються з двох частин (являють собою конкатенацію) – перевірки умови та виконання оператора або групи операторів, якщо умова істинна. При цьому умова береться в круглі дужки, а виконувані оператори – у квадратні. Наприклад, $(x_i \le 1)[D_1,(x_i > 0)[D_2, D_3]]$ і т.д. Завжди істинна умова позначається U, а помилкова – F. Циклічний оператор – це конкатенація умовних операторів. Кількість виконання циклу називається його кратністю. Наприклад, $n - m + 1$ - кратний цикл, у якого m – нижній параметр, а n – верхній, позначається так:

$$\prod_{i=m}^{n}(U_i)[D_i],$$

При виконанні такого оператора здійснюється ітеративне виконання умовного оператора для кожного значення $i \in [m, n]$. При цьому якщо умова істинна, виконується оператор D_i, інакше виконання циклу закінчується. Будь-яка послідовно виконуєма група операторів називається операторним вираженням. Операторний вираз може також бути представлений одним оператором. Всі оператори алгоритму процесу оперують даними, що описують стан процесу, його синхронні й асинхронні входи і виходи.

Розглянемо тепер динаміку процесу. У момент часу $t_0=0$, тобто на початку моделювання, відбувається ініціалізація всіх процесів моделі. Вона полягає у виконанні оператора ініціалізації процесу D_{init}. Відповідно до цього оператора усім внутрішнім змінним процесу привласнюються їхні початкові значення, вказівник поточної команди IP установлюється на перший оператор алгоритму для кожного процесу, а також усі процеси моделі переводяться в режим ready. Далі кожен процес моделі починає виконання алгоритму. При функціонуванні процесу часову вісь $t \in [0, \infty)$ вважають розділеного на відрізки, границями яких є моменти зміни вхідних керуючих або вихідних сигналів. Такі моменти часу називаються особливими. Стани в особливі моменти часу називаються особливими станами. Тоді вся динаміка процесу описується чергуванням особливих станів. У ході виконання алгоритму кожен процес може змінювати значення своїх внутрішніх станів, обчислювати нові значення виходів і затримок, які потрібно відробити по кожному з виходів, що змінюють значення. У загальному виді алгоритм процесу може бути реалізований за допомогою наступних двох груп операторів. Перша група складається з операторів керування станом процесу:

$$z(t_2) = H_1(z(t_1), x(t_1), c(t_1)),$$

де t_1 – поточне значення модельного часу;

t_2 – наступний момент модельного часу, значення якого відрізняється від t_1 на нескінченно малу величину;

H_1 – є оператором переходів, а оператор очікуваання wait буде розглянуто нижче:

wait ((список сигналів керування), (умова),(час)).

Друга група операторів визначає виходи процесу:

$$y(t_2) = G_1(z(t_1), x(t_1), c(t_1));$$

$$\Delta t(t_2) = G_2(z(t_1), x(t_1), c(t_1));$$

$$a(t_2) = G_3(z(t_1), x(t_1), c(t_1));$$

$$y'(t_2) = G_4(y(t_2), \Delta t(t_2), a(t_2)),$$

де оператор G_1 обчислює значення сигналу y на виході процесу, оператор G_2 обчислює значення інтервалу часу, через який на виході процесу повинне установитися значення $y; \Delta t \in T, T = [0, \infty)$; оператор G_3 визначає атрибути, які впливають на спосіб моделювання затримки на даній лінії, а оператор G_4 формує векторну структуру вихідного сигналу y'.

Практично він відповідає такому операторному виразу:

$$y_i' := y_i \,\&\, \Delta t_i \,\&\, a_i \,,$$

де & - операція конкатенації, застосована для формування вектора y_i' із трьох елементарних сигналів. Дія такого оператора полягає у формуванні вектора виду (y_i, t_i, a_i) і миттєвій передачі його на вихід y_i' процесу. Помітимо, що цей оператор тільки передає результати функцій G_1, G_2 і G_3 на вихід, але не моделює затримку.

Усі приведені оператори можуть задаватися з використанням будь-яких засобів, включаючи словесні описи або алгоритми.

Серед усіх операторів алгоритму процесу зупинимося більш докладно на операторі очікування wait. Алгоритм кожного процесу виконується доти, поки не виконається оператор очікування. Оператор очікування призначений для переходу процесу з активного стану в стан очікування, або стан блокування.

$$wait\left(\{c_i\}_{i=0}^{n}, U, t_d\right).$$

Виконання оператора приводить до того, що виконання алгоритму припиняється, процес переходить у режим очікування (sleep), у якому він знаходиться доти, поки не зміниться хоча б один з керуючих сигналів, приведених у списку $\{c_i\}_{i=0}^{n}$, n – кількість керуючих сигналів у списку оператора wait. Якщо це відбулося, процес обчислює умову U – умова у операторі wait. У випадку істинності цієї умови виконання алгоритму продовжується з наступного після wait оператора, інакше процес залишається в стані sleep. Незалежно від зміни сигналів або виконання умов процес переходить у режим active після закінчення інтервалу часу, заданого параметром (t_d), з моменту переходу процесу в режим очікування. Повний оператор ний вираз, котрий описує перемикання процесу у режим очікування, можна представити у такому вигляді:

$$m_i := sleep, D_{ret}, \left(\bigvee_{i=0}^{n} \overline{stable}(c_i)\right)\left[(U)\left[m_i := active\right]\right], (t_m \geq t_w)\left[m_i := active\right], D_{ret},$$

де m_i – режим роботи процесу;

D_{ret} – оператор, який повертає керування системі, якщо m_i = sleep;

stable(c_i) – умова, який приймає значення «істина», якщо сигнал c_i не змінював своє значення в даний момент часу;

t_m – поточне значення модельного часу;

$t_w=t_1+t_d$ – значення модельного часу, коли процес повинний активізуватися, незалежно від інших умов.

Виходить, що оператор wait керує режимом роботи процесу, який є частиною стану процесу z. Таким чином, можна вважати, що оператор wait є реалізацією однієї з підфункцій функції переходів H_1 процесу.

Модельний час не змінюється, поки хоча б один процес системи знаходиться в активному режимі. Якщо для всіх процесів системи виконалися оператори wait, і немає жодного активного процесу, програма керування моделюванням розраховує наступне значення модельного часу, коли в системі відбудуться зміни керуючих входів хоча б одного з процесів. Тоді будемо говорити, що в системі відбувається подія, модельний час приймає розрахункове значення і виконується підпрограма, що визначає, які з процесів моделі переходять в активний режим. Далі, всі процеси, що стали активними, знову виконуються, планують нові події в системі в наступні моменти часу, та знову переходять у режим sleep. Описана послідовність дій виконується циклічно, поки не виконаються умови закінчення моделювання.

Виникає питання – як повинні поводитися процеси, виконання алгоритмів яких дійшло до кінця. Щоб відповісти на це питання, потрібно згадати, що кожен процес моделює функцію системи. Після розгляду кожної з описаних вище типових функцій складних ЕС стає очевидним, що функція, яка закінчила виконання, може бути знову активована керуючими сигналами, тобто при виконанні деяких умов буде виконуватися із самого початку. Таким чином, ясно, що останнім оператором будь-якого процесу буде оператор передачі керування знову на початок його алгоритму.

Виходячи зі сказаного, алгоритм, який моделює поводження будь-якого процесу, можна представити у виді операторної схеми наступного виду:

$$D_{init}, \prod(U)\left[\left[D_1,...,D_{i-1}, wait_i\right]_1,...,\left[D_j,...,D_{k-1}, wait_k\right]_b\right],$$

де D_{init} – операторний вираз, який виконує ініціалізацію внутрішніх змінних процесу;

D_i – інші оператори процесу, серед яких можуть бути й оператори призначення.

З цього виразу видно, що алгоритм процесу будується з ініціалізуючого оператора D_{init} і нескінченного циклу (тому що керуюча ним умова завжди істинна). У циклі міститься b операторних виразів, кожний з яких закінчується оператором wait. Кожний операторний вираз заданим образом впливає на виходи системи та її внутрішні стани, після чого оператор wait переводить процес у режим очікування, даючи, таким чином, системі моделювання можливість виконати інші процеси моделі та відреагувати на дії даного процесу. Слід зазначити, що оператор wait повинний знаходиться в алгоритмі процесу хоча б один раз, інакше процес ніколи не вийде з активного режиму і буде знову та знову віконувати ітерації циклу. У такому випадку модель стає некерованою з боку системи моделювання.

2.5 Висновки

У цьому розділі пропонується використовувати процес як формальну схему поведінкового опису елементів системи. Такий вибір пояснюється наступними причинами:

- процесорна модель являє собою систему, яка складається з поєднаних каналами зв'язку «чорних ящиків», що допомагає формалізувати концептуальні представлення, існуючи у користувача щодо об'єктів моделюємої системи, а також сприяє використанню візуального моделювання. Також процеси припускають використання графічних засобів для представлення моделі.

- процес являє собою досить універсальний перетворювач інформації, що сприймає вхідні дані і видає вихідні сигнали. Це дуже зручно для здійснення функціональної декомпозиції системи при розв'язанні різних задач синтезу й аналізу.

- процесорні моделі є узагальненням різноманітних машинних моделей систем і формалізують такі фундаментальні для імітаційного моделювання поняття, як стан, структура, сигнали, події і т.д. Процесорні моделі не викликають труднощів при організації квазіпаралелізму в моделях і є зручними для реалізації на цифрових ЕОМ.

3 ОБ’ЄКТНО-ОРІЄНТОВАНА КОНЦЕПЦІЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

3.1 Вступ

Реальні складні системи (СС) можна досліджувати за допомогою двох типів математичних моделей: аналітичних та імітаційних. В аналітичних моделях поведінка СС записується у вигляді деяких функціональних співвідношень або логічних умов. Найбільш повне дослідження вдається провести в тому випадку, коли отримані явні залежності, що зв’язують шукані величини, що шукаються з параметрами СС. Для побудови аналітичних моделей існує могутній математичний апарат (системи лінійних, нелінійних і диференціальних рівнянь, теорія імовірностей, математична статистика, теорія масового обслуговування тощо). Наявність математичного апарата і відносна швидкість опису поводження СС сприяла повсюдному поширенню аналітичних моделей у різних галузях науки і техніки.

Однак застосування аналітичних моделей можливо для порівняно простих систем. Для складних систем дослідникові найчастіше приходиться йти на спрощення зображення реальних явищ, що відбуваються в системі. Як приклад, розглянемо основні припущення й обмеження, що властиві аналітичним моделям СМО, що використовуються для дослідження обчислювальних систем зокрема. Однорідність потоків подій значно спрощує математичну модель системи, але робить її непридатною для випадку пріоритетного вхідного потоку заявок. У цьому випадку процеси, що відбуваються в системі, хоча й описуються процесом загибелі і розмноження, але значно складним і більшої розмірності [9,36]. Ще одне обмеження - неможливість обслуговування однієї заявки одночасно декількома приладами. На практиці, дуже часто заявка захоплює кілька приладів і монополізує їх. Ця обставина повинна трактуватися як паралельна робота декількох пристроїв, що також приводить до значного ускладнення математичної моделі системи, роблячи її іноді нерозв'язної [9]. Що стосується припущення про експонентний час обслуговування і найпростіший вхідний потік, то це буває далеко не завжди. Наприклад, якщо потік подій, що переводить систему з одного стану в інший, є потік відмовлень якого-небудь вузла, то більш вірніше припустити, що час безвідмовної роботи вузла, який залишився, залежить від того, скільки часу вузол уже проробив. При цьому час перебування вузла в робочому стані є випадковою величиною, розподіленою не за показовим законом. У підсумку має місце процес, що залежить від передісторії. Для цього процесу написати рівняння, що виражають у загальному випадку імовірності станів як функції часу практично неможливо. Дійсно, для марківського процесу обчислюється імовірність того, що в момент $t + \Delta t$ система буде в стані Si, з огляду не тільки на те, у якому стані вона була в момент t, незважаючи, скільки часу вона була в цьому стані. Для немарківської системи цей прийом уже не придатний, обчислюючи імовірність переходу з одного стану в інший за час Δt, потрібно буде враховувати, скільки часу вона пробула в даному стані. Це приводить, до того, що замість звичайних диференціальних рівнянь, до рівнянь у частинних похідних, тобто до набагато складнішого математичного апарата, за допомогою якого тільки в рідких випадках можна одержати результат [37].

Підсумувати сказане можна в таким чином, є система S, функціонування якої не вкладається в рамки марківського процесу, тому при розрахунку показників її ефективності дослідник повинний вибрати визначену стратегію:

- зневажити відхиленнями і використовувати марківську модель, тобто свідомо неадекватну реальній системі, одержавши натомість можливість використовувати добре розроблену методику розрахунку;

- використовувати адекватну реальній системі математичну модель, а, отже, і складні математичні методи розрахунку показників ефективності, що приведуть до заміни рішення системи лінійних рівнянь на рішення системи нелінійних рівнянь великої розмірності або до заміни рішення системи звичайних диференціальних рівнянь у частинних похідних на рішення системи диференціальних рівнянь;

- використовувати адекватну реальній системі імітаційну модель і імітаційні методи моделювання для оцінки показників ефективності систем.

Однозначні рекомендації дати важко. Все залежить від вимог експерименту, від важливості результатів, якщо вони використовуються, як орієнтована оцінка показників ефективності, то ступінь адекватності моделі може бути невисока. Для отримання точних результатів необхідно працювати з моделлю, що має високий ступень адекватності.

На підставі досвіду імітаційного моделювання й аналізу літературних джерел [12, 38] можна рекомендувати дослідникові використовувати імітаційну модель складної системи при вирішенні своїх задач у наступних випадках:

Якщо не існує закінченої постановки задачі дослідження і йде процес пізнання об’єкта моделювання. Імітаційна модель служить засобом вивчення явища.

Якщо аналітичні методи існують, але математичні процедури настільки складні й трудомісткі, що імітаційне моделювання дає простіший спосіб розв’язання задачі.

Коли крім оцінки впливу параметрів СС бажано здійснити спостереження за поводженням компонентів СС протягом визначеного періоду, у тому числі зі зміною термінів протікання процесів.

Коли імітаційне моделювання виявляється єдиним способом дослідження складної системи через неможливість спостереження явищ у реальних умовах.

Коли вивчаються нові ситуації в СС, про які мало що відомо або невідомо нічого. У цьому випадку імітація служить для попередньої перевірки нових стратегій і правил прийняття рішень перед проведенням експерименту на реальній системі.

Однак імітаційні моделі поряд з характерними для них достоїнствами мають ряд істотних недоліків. Розробка гарної імітаційної моделі часто обходиться дорожче створення аналітичної моделі і вимагає великих тимчасових витрат. Іноді може показатися, що імітаційна модель точно відображує реальне положення справ у СС, а в дійсності це не так. Р. Шеннон показав у [39], що імітаційна модель, принципово, не точна і ми не в змозі виміряти ступінь цієї неточності. Проте, імітаційне моделювання є одним з найбільше широко використовуваних методів при вирішенні задач синтезу й аналізу СС. Із достоїнств імітаційного методу виділимо: можливість опису поводження компонент СС на високому рівні деталізації, відсутність обмежень на вигляд залежностей між параметрами імітаційної моделі і станом зовнішнього середовища СС, можливість дослідження динаміки взаємодії компонент у часі і просторі параметрів системи. Зазначені достоїнства забезпечують імітаційному часові широке поширення.

3.2 Сутність імітаційного моделювання

Основну ідею імітаційного моделювання розглянемо на прикладі обслуговування касиром клієнтів у банку. Аналітична модель такої системи (одноканальна система масового обслуговування з необмеженою чергою) була докладно розглянута у розділі 4.3. У цьому розділі покажемо альтернативний метод моделювання – метод імітаційного моделювання.

Клієнти прибувають у банк, якщо касир зайнятий, очікують обслуговування в єдиній до нього черзі, обслуговуються і потім залишають банк. Для простоти припускаємо, що інтервал часу між клієнтами (або момент прибуття клієнта) і час обслуговування його касиром відомі для кожного клієнта (табл. 3.1). Ціль полягає в ручній імітації функціонування описаної вище системи для того, щоб визначити, який відсоток часу касир вільний (зайнятий) і який середній час перебування клієнта в банку.

Таблиця 3.1 – Моменти прибуття і час обслуговування клієнта

Номер клієнта	Момент прибуття клієнта (t_i), хв.	Інтервал часу між клієнтами (Δt_i), хв.	Час обслуговування, хв.
1	3.2	3.2	3.8
2	10.9	7.7	3.5
3	13.2	2.3	4.2
4	14.8	1.6	3.1
5	17.7	2.9	2.4
6	19.8	2.1	4.3
7	21.5	1.7	2.7
8	26.3	4.8	2.1
9	32.1	5.8	2.5
10	36.6	4.5	3.4

Тому що імітаційне моделювання - це динамічне відображення змін стану системи з часом, то в першу чергу повинні бути визначені стани системи. Визначимо стан системи за кількістю клієнтів у банку.

Відзначимо, що стан системи змінюється в результаті впливу на неї двох типів подій:

події, зв’язаною з прибуттям клієнта в банк (будемо називати його «прибуття клієнта на обслуговування»);

події, зв’язаною з завершенням обслуговування касиром і наступним відходом клієнта (будемо називати його «закінченням обслуговування клієнта»).

Для ілюстрації імітації, стан системи буде визначатися як результат обробки упорядкованих у часі згаданих подій.

Результат ручної імітації поводження системи відповідно до вихідних даних, наведені в таблиці 3.2. При цьому передбачається, що в початковий момент часу в системі немає клієнтів, касир вільний і перший клієнт перебуває в момент часу, рівний 3.2 хв.

Таблиця 3.2 – Ручне моделювання системи

Номер клієнта (1)	Момент прибуття, хв. (2)	Момент початку обслуговування, хв. (3)	Момент відходу, хв. (4)	Час чекання в черзі, хв (5)=(3)-(2)	Час перебування в банку, хв (6)=(4-(2))
1	3,2	3,2	7,0	0,0	3,8
2	10,9	10,9	14,4	0,0	3,5
3	13,2	14,4	18,6	1,2	5,4
4	14,8	18,6	21,7	3,8	6,9
5	17,7	21,7	24,1	4,0	6,4
6	19,8	24,1	28,4	4,3	8,6
7	21,5	28,4	31,1	6,9	9,6
8	26,3	31,1	33,2	4,8	6,9
9	32,1	33,2	35,7	1,1	3,6
10	36,6	35,7	40,0	0,0	3,4

У таблиці 3.2 перший і другий стовпці узяті з таблиці 3.1. Час початку обслуговування, наведений у третьому стовпці, залежить від того, чи залишив попередній клієнт банк. Він приймається рівним найбільшому значенню з часу прибуття даного клієнта і відходу попереднього. Час відходу, наведений у четвертому стовпці, обчислюється як сума відповідного елемента третього стовпця і часу обслуговування даного клієнта, що визначається за таблицею 6.1. Середні значення часу очікування в черзі і часу перебування клієнта в банку відповідно дорівнюють 2.6 і 5.81 хвилинам.

Таблиця 3.2 містить підсумкову інформацію, що відноситься до клієнта, але не містить відомості про касира і розмір черги до нього. Для одержання такої інформації необхідно досліджувати події, пов’язані з цією ситуацією.

Логіка обробки подій приходу і відходу (або реакції системи на ці події) залежить від стану системи в момент настання цих подій. При настанні події «прибуття клієнта на обслуговування» подальша ситуація визначається станом касира. Якщо касир вільний, він переходить у стан «зайнятий» і приступає до обслуговування клієнта. При цьому планується подія «відхід клієнта» у момент часу, дорівнює поточному часу плюс тривалість його обслуговування. Якщо ж касир зайнятий, обслуговування клієнта не може початися, й, отже, він стає в чергу і довжина черги збільшується на одиницю. Логіка обробки події «закінчення обслуговування клієнта» залежить від довжини черги. Якщо в черзі є хоча б один клієнт, касир залишається в стані «зайнятий», довжина черги зменшується на одиницю і для першого клієнта з черги планується подія відходу. Якщо ж черга порожня, касир переводиться в стан «вільний».

У таблиці 6.3 приводиться подійно-орієнтований опис стану касира і числа клієнтів у банку (події розташовані в хронологічному порядку). На рисунку 3.1 приведені графіки зміни значень цих перемінних станів у часі. Результати імітації показують, що протягом перших сорока хвилин роботи в банку в середньому одночасно знаходилося 1.7 клієнта, а касир був вільний 20% часу.

Для розташування подій приходу і відходу в хронологічному порядку необхідно вести запис (список) подій. Це здійснюється шляхом фіксування моментів настання наступної події приходу і наступної події відходу. Порівняння цих моментів визначає потім вибір однієї з подій для обробки. Такий упорядкований список подій іноді називають календарем подій.

Наведений приклад ілюструє цілий ряд важливих понять. Ми переконалися в тім, що в будь-який момент імітаційного часу модель знаходиться в деякому стані. При настанні подій стан системи змінюється відповідно до логіко-математичних відносин, що пов’язані з цими подіями, іншими словами, відповідно до логіки функціонування системи.

Таблиця 3.3 – Подійно-орієнтований опис імітації роботи банківського касира

Час події, хв.	Номер клієнта	Тип події	Довжина черги	Число клієнтів	Стан касира	Час простою касира, хв.
0,0	-	Початок	0	0	Вільний	-
3,2	1	Прихід	0	1	Зайнятий	3,2
7,0	1	Відхід	0	0	Вільний	
10,9	2	Прихід	0	1	Зайнятий	3,9
13,2	3	Прихід	1	2	Зайнятий	
14,4	2	Відхід	0	1	Зайнятий	
14,8	4	Прихід	1	2	Зайнятий	
17,7	5	Прихід	2	3	Зайнятий	
18,6	3	Відхід	1	2	Зайнятий	
19,8	6	Прихід	2	3	Зайнятий	
21,5	7	Прихід	3	4	Зайнятий	
21,7	4	Відхід	2	3	Зайнятий	
24,1	5	Відхід	1	2	Зайнятий	
26,3	8	Прихід	2	3	Зайнятий	
28,4	6	Відхід	1	2	Зайнятий	
31,1	7	Відхід	0	1	Зайнятий	
32,1	9	Прихід	1	2	Зайнятий	
33,2	8	Відхід	0	1	Зайнятий	
35,7	9	Відхід	0	0	Вільний	
36,6	10	Прихід	0	1	Зайнятий	0,9
40,0	10	Відхід	0	0	Вільний	

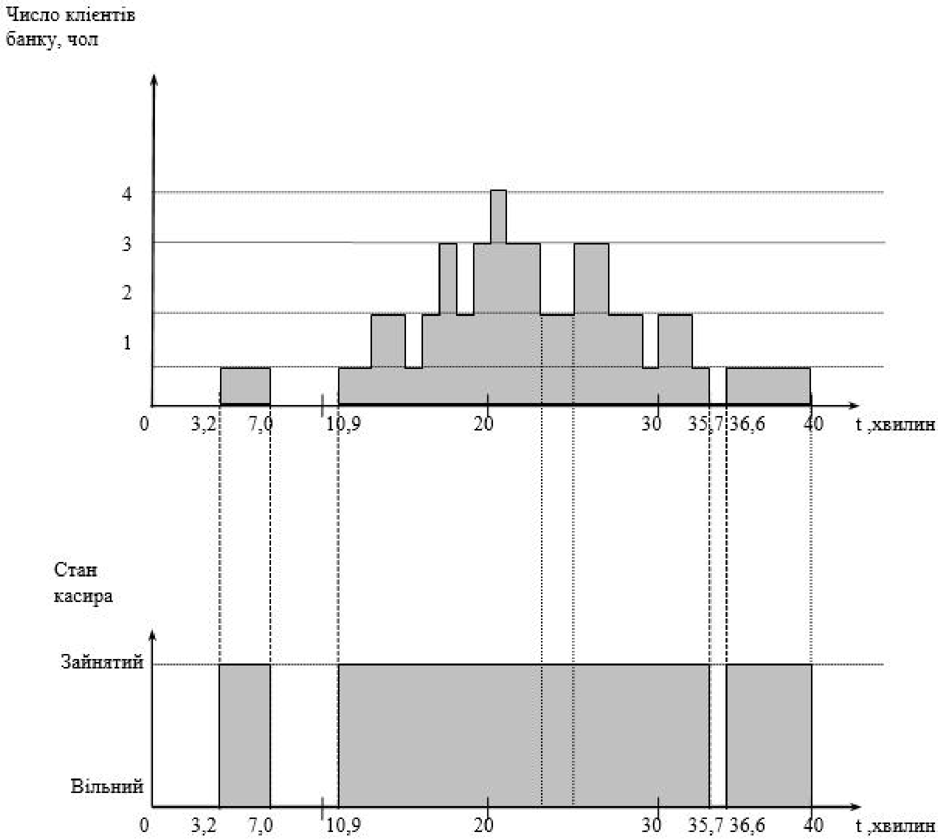


Рисунок 3.1 – Графічне зображення результатів моделювання роботи банку

Таким чином, події визначають динамічне поводження системи. При заданому початковому стані, логічній схемі обробки кожної події і методу визначення випадкових величин наша проблема в більшому ступені стає обчислювальною проблемою. Істотним елементом схеми обчислень є список подій, що забезпечує механізм для запису й упорядкування майбутніх подій. Ще одним важливим моментом є те, що ми можемо розглядати зміни стану системи в двох аспектах:

- як процес, з яким зіштовхується клієнт при обслуговуванні (точка зору клієнта);
- як послідовність подій, що викликають зміну стану касира (точка зору касира).

3.3 Принципи розробки імітаційних моделей

Розглянемо основні принципи алгоритмізації або побудови імітаційної моделі функціонування об'єкта дослідження. Для демонстрації цього скористаємося прикладом, котрий був розглянутий у попередньому розділі. Очевидно, що при побудові імітаційної моделі такої системи необхідно вміти [32]:

- моделювати події, що виникають у системі і приводять до зміни її станів;
- моделювати реакцію системи на ці події.

Моделювання реакції системи на події. У попередньому підрозділі були розглянуті два типи подій, що переводять систему зі стану в стан: прихід заявки на обслуговування і закінчення обслуговування.

Із таблиці 3.4 видно, що в системі виникає ще одна подія, що переводить обслуговуючий прилад з вільного стану в зайнятий, назвемо його "початок обслуговування". Ця подія виникає в двох випадках, коли йому передує основна подія: "надходження заявки" і прилад вільний (черга порожня) або "закінчення обслуговування" і черга не порожня.

Моделювання подій. До подій у даному випадку відносимо всі те, що змінює стани системи:

- прихід заявки на обслуговування;
- закінчення обслуговування.

Моделювання події прихід заявки на обслуговування, практично, зводиться до моделювання вхідного потоку заявок: у момент часу, коли надійшла чергова заявка, необхідно планувати час приходу наступної.

Це робиться в два етапи:

- розігрується випадкове число відповідно з законом розподілу інтервалу часу між заявками. Отримане значення інтерпретується як величина інтервалу часу, що повинний пройти перш, ніж з'явиться наступна заявка;

- значення тимчасового інтервалу додається до поточного значення часу. Результат вказує на момент часу в майбутньому, коли надійде наступна заявка.

Геометрична інтерпретація описаного процесу, з використанням часової осі модельного часу, представлена на рисунку 3.2.

Таблиця 3.4 – Реакція системи на події

Основна подія	Дія, що вона викликає.
Прихід заявки на обслуговування	Планування приходу наступної заявки Перевірка стану обслуговуючого приладу: Прилад вільний? НІ: заявка стає у чергу; ТАК: заявка надходить на обслуговування, це викликає: - перехід обслуговуючого приладу з вільного стану в зайнятий; та - планування події “закінчення обслуговування”.
Закінчення обслуговування	Перевірка стану черги. Чи є там заявка на обслуговування? НІ: перехід обслуговуючого приладу з зайнятого стану у вільний. ТАК: надходження заявки на обслуговування, це викликає: - просування заявки в черзі; та - планування події “закінчення обслуговування”.

Розглянемо тепер планування події закінчення обслуговування, що пов’язано з часом обслуговування заявки приладом. Подія закінчення обслуговування планується в той момент, коли заявка надходить на обслуговування, тобто відбувається подія початок обслуговування. Цю подію можна назвати внутрішньою або допоміжною, воно не змінює стани системи, і момент її появи залежить від моменту закінчення обслуговування поточної заявки і приходу заявки на обслуговування.

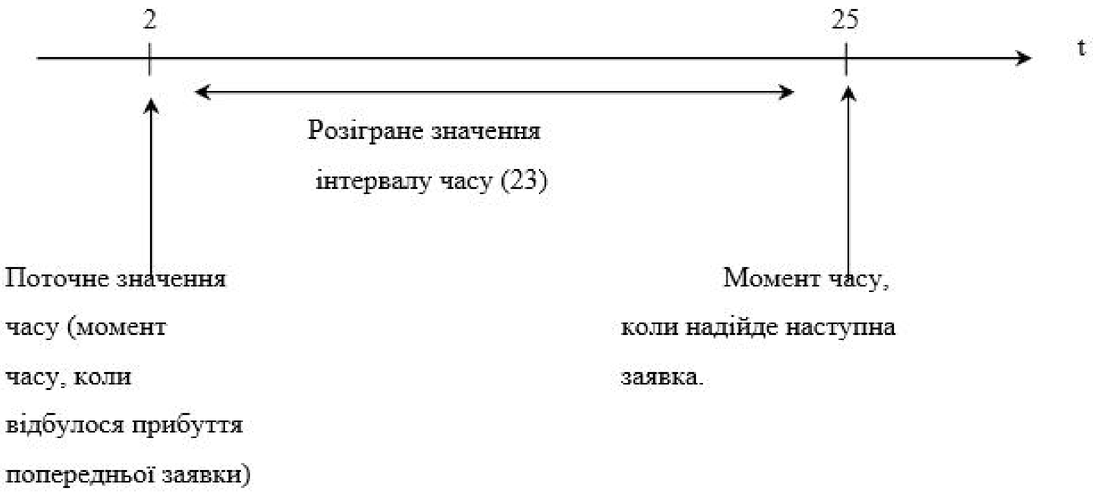


Рисунок 3.2 – Часова діаграма, що ілюструє прихід наступної заявки

Зважаючи на те, що час обслуговування tобсл. теж є незалежною випадковою величиною, подія «закінчення обслуговування» планується також, як і в процедурі планування приходу наступної заявки:

- розігрується значення інтервалу часу відповідно з законом розподілу часу обслуговування;
- цей інтервал додається до значення часу, яке відповідає початку обслуговування заявки.

Геометрична інтерпретація описаного процесу, з використанням часової осі модельного часу, наведена на рисунку 3.3.

Розглянемо планування події початок обслуговування. Тому що вона виникає в двох випадках, коли їй передує основна подія: надходження заявки та прилад вільний (черга порожня) або закінчення обслуговування і черга не порожня, то момент часу настання події початок обслуговування попередньо не планується, от чому вона є допоміжною (іншими словами її можна обчислити).

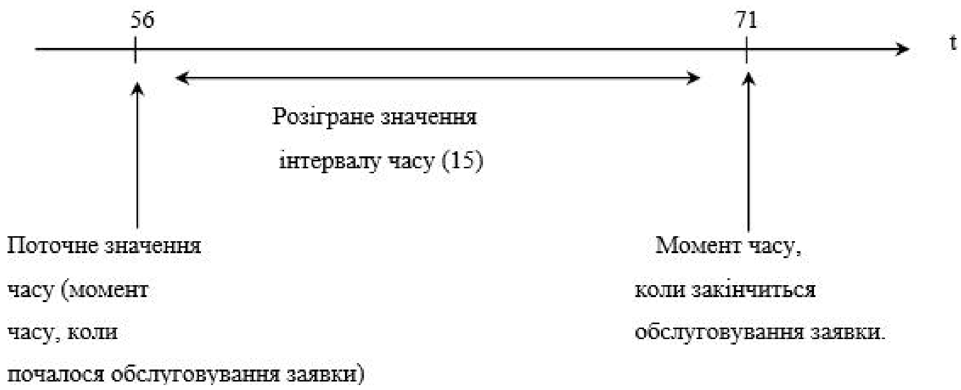


Рисунок 3.2 – Часова діаграма, що ілюструє планування завершення обслуговування заявки

Для демонстрації розглянутих принципів моделювання побудуємо імітаційну модель одноканальної СМО з необмеженою чергою (рисунок 3.4), алгоритм функціонування якої був докладно розглянутий вище. Припустимо, що в потоці заявок, що надходять, інтервал часу має рівномірне $R[\sim, \ ]$, а час обслуговування має нормальний розподіл N (m, σ). У процесі моделювання становить інтерес оцінка наступних характеристик: середнього часу очікування заявки в черзі і середньої кількості заявок у черзі.

Для проведення статистичного моделювання такої системи обслуговування необхідні: підпрограми імітації випадкових чисел, розподілених за рівномірним та нормальним законом. При побудові схеми алгоритму використовуватимемо наступні позначення:

t_0 – початок моделювання;

T- кінець моделювання;

n – поточне число обслугованих заявок;

t_j‘- момент часу надходження j-й заявки;

t_j‘‘- момент часу закінчення обслуговування j-й заявки;

t_m - таймер модельного часу або поточний час;

Δt – інтервал часу між заявками;

$t_{обсл}$ – тривалість обслуговування;

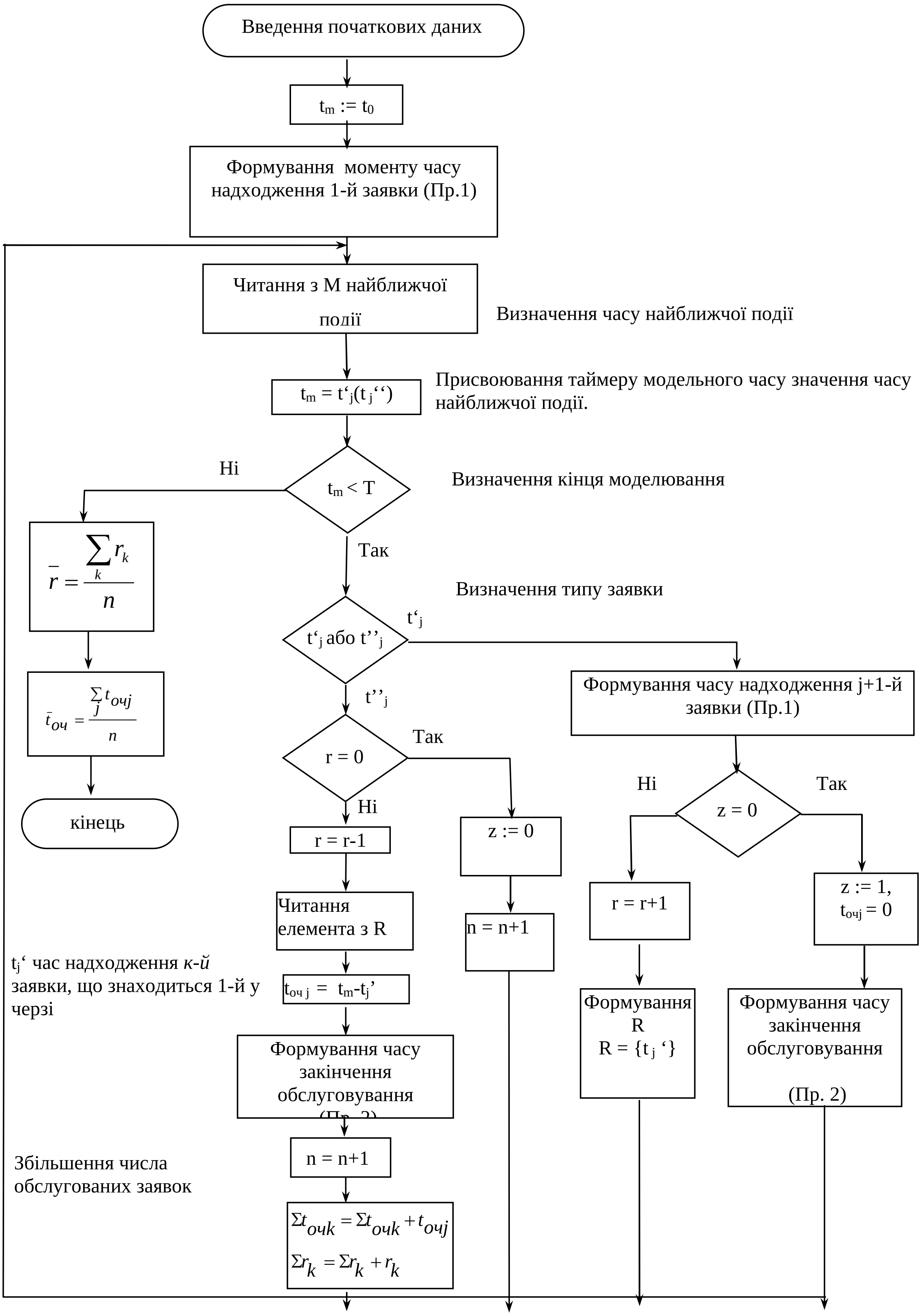


Рисунок 3.4 – Імітаційна модель одноканальної СМО з необмеженою чергою

z – прапор зайнятості приладу: z=0 – прилад вільний, z=1 – прилад зайнятий;

r_k – поточне значення черги;

$t_{ожj}$ – час очікування j-й заявки в черзі;

M={t_j‘,t_j‘‘} – упорядкований масив моментів часу основних подій або список основних подій;

R={t_j‘} - упорядкований масив часу надходжень j-ої заявки, що знаходяться на момент t_m у черзі;

При розробці алгоритму прийняті наступні обмеження:

а) j-я заявка, для якої t_j‘>T не розглядається;

б) j-я заявка, для якої t_j‘‘<T вважається обслугованою;

в) j-я заявка, для якої t_j‘<T, а t_j‘‘>T вважається такою, що отримала відмову.

Необхідні статистичні характеристики СМО обчислюються таким чином:

$$\bar{t}_{оч} = \frac{\sum_j t_{очj}}{n};$$

$$\bar{r} = \frac{\sum_k r_k}{n}.$$

Таким чином, розглянуті Пр1. та Пр2.

3.4 Формалізація імітаційних моделей

Імітаційне моделювання в даний час є найефективнішим методом оцінки характеристик процесу функціонування складних систем. Нагадаємо, що імітаційна модель є логіко-математичним описом об’єкта дослідження, що відтворює як структуру системи, залежності, які існують між її параметрами, так і процеси функціонування системи в часі. Дуже часто імітаційна модель зображується у вигляді алгоритму (рисунок 6.4). В імітаційній (алгоритмічній) моделі або алгоритмі, що моделює внутрішні операції і внутрішня структура можуть досить сильно збігатися з тими, котрі існують у реальній системі. Для порівняння, при аналітичному моделюванні оцінки показників ефективності $\overline{Y}(t)$ системи,, що моделюється, обчислюється безпосередньо з математичної моделі за допомогою різних чисельних методів.

Основна увага в даному розділі приділятиметься алгоритмічним моделям або алгоритмам, які моделюють складні системи [12,14,40], що реалізуються на ЕОМ. Для їхнього опису доцільно розглянути базові поняття, що конкретизують динаміку моделей систем, що досліджуються. Передбачається, що систему можна описати в рамках понять "стан" і "перехід зі стану в стан" під дією відповідних подій.

Під станом системи розуміється комбінація значень параметрів $\overline{X}(t)$, $\overline{U}(t)$ (рисунок 1.2), а також значення деяких внутрішніх параметрів, що описують систему. Тоді стан системи можна представити у вигляді вектора:

$$\overline{S}' = \left(S_1', S_2', ..., S_k'\right),\ \overline{S}'' = \left(S_1'', S_2'', ..., S_k''\right)\ \text{тощо,}$$

де $S_i' = S_i(t')$, $S_i'' = S_i(t'')$ — значення деякого узагальненого і-го параметра в моменти часу, відповідно, t' та t''.

Сукупність усіх можливих значень станів $\{S_i'\}$ називається простором станів об’єкта моделювання. Якщо розглядати процес функціонування системи як послідовну схему станів $\overline{S}(t) = \left(S_1(t), S_2(t), ..., S_k(t)\right)$, то вони можуть бути інтерпретовані як координати точки в k-мірному фазовому просторі. Причому кожній реалізації процесу відповідатиме деяка фазова траєкторія.

Відповідно до цього, імітаційне моделювання – це зображення динаміки імітаційної моделі за допомогою її просування від одного стану до іншого за даними операційними правилами. Як правило, ці правила мають алгоритмічну основу.

Для опису динаміки імітаційної моделі необхідно скористатися специфікаціями, що зв'язують характеристики станів системи $S_i(t)$ з її параметрами і часом, а також початковими умовами S_i^0 для початкового моменту часу t_0.

Згадані операційні правила перетворяться до такого вигляду, щоб, по можливості, зробити зручним визначення $S_i(t_0 + \Delta t)$. У моделі виділяється деякий ідентифікатор для фіксації поточного значення t ("модельний час"). У початковий момент модельний час дорівнює t_0. До t_0 додається величина Δt й обчислюється момент часу $t_1 = t_0 + \Delta t$ зміни стану системи, а також самі стани $S_i(t_1)$; потім знову визначається час $t_2 = t_1 + \Delta t$ зміни станів тощо.

Моменти часу t_0, t_1, t_2, ... називаються особливими. Зміна станів системи, що відбувається в ці моменти часу, цілком визначає динаміку імітаційної моделі. Тому при машинній реалізації досить відтворювати лише ці зміни. Необхідно відзначити, що зміни в особливі моменти часу, можуть бути випадковими. Тому структура імітаційної моделі повинна забезпечувати обчислення розподілу імовірностей і-го параметра $S_i(t)$, $i = \overline{1,n}$.

Розрізняють два типи імітаційних моделей, що пов'язано з наступними обставинами.

У моделях одного типу переходи системи з одного стану в інший, ініціюється реальними подіями, що відбуваються в системі. Тоді послідовність зазначених подій складає суть функціонування системи. У цьому випадку динаміка моделі фактично повторює динаміку системи, тобто перехід від однієї події до іншої. Говорять, що відповідна алгоритмічна модель має подійний *характер.*

У моделях іншого типу події вводяться штучно внаслідок необхідності зображати деякий безперервний процес у дискретній ЕОМ. Це характерно для випадку, коли імітаційна модель є алгоритмічній реалізацією аналітичної моделі з метою її розв’язаня на ЕОМ. Прикладом є будь-який чисельний метод інтегрування диференціальних рівнянь. У подібних методах моменти настання подій визначаються кроком інтегрування. Крок інтегрування або призначається постійним, або, якщо це необхідно, змінюється у відповідності зі спеціально встановленим механізмом. Тому методи цього типу називають покроковими. При їхньому використанні динаміка моделі є дискретним наближенням реальних безперервних процесів.

Зупинимося на одній важливій особливості зазначеного алгоритмічного уявлення. Внаслідок послідовного характеру обробки інформації в обчислювальній однопроцесорній машині паралельні процеси, що відбуваються в моделі, перетворяться за допомогою обговорюваного алгоритму в послідовні. Такий спосіб уявлення має назву *квазіпаралельного процесу*. Протиріччя між паралельністю модельних процесів і послідовним характером квазіпаралельного процесу є причиною їхньої неповної відповідності.

Для деякого класу систем можливістю одночасного настання двох або більшої кількості подій можна зневажити. Уявимо тепер, що така можливість припускається. Тоді для завдання квазіпаралельного процесу визначити реакції моделі на кожну з подій недостатньо, необхідно вказати ще, в якій послідовності модель повинна реагувати на одночасні події, що відбуваються, тому що в алгоритмі, який моделює, вони обробляються послідовно. Ця проблема не виникає, якщо реакція моделі не залежить від порядку настання подій (комутативність подій). Інакше проблеми, що виникають, можуть бути вирішені тільки шляхом завдання необхідного порядку обробки одночасних подій. Іноді попередній теоретичний аналіз дозволяє зневажити можливістю одночасності подій: наприклад, якщо моделюється одноканальна система масового обслуговування з пуасонівським вхідним потоком, то імовірність одночасного надходження заявки і закінчення обслуговування дорівнює нулю.

Якщо в моделях подійного типу події, що наступають одночасно, є винятком і не ускладнюють задачу або через комутативність події, або через їхню малу імовірність, то в моделях покрокового типу одночасні події є правилом, а не виключенням. Тому при побудові покрокових моделей значна увага повинна бути приділена організації обробки одночасних подій. З цією проблемою тісно пов’язане питання вибору кроку Δt: зменшення кроку збільшує точність відтворення траєкторії моделі, зменшує число одночасних подій, що відбуваються, але приводить до збільшення часу імітації. Це питання досить докладно буде представлено у наступному розділі.

3.5 Поняття про модельний час. Керування модельним часом

Для того щоб зрозуміти механізм імітації функціонування складної системи на ЕОМ, розглянемо наступний приклад.

Об’єктом імітації є процедура обробки і-ої програми в мультипроцесорної ОС. Для і-ої програми можна виділити ряд стадій її обробки, кожна стадія характеризується відповідним часом τ_{ij} протягом якого виконується відповідна послідовність функціональних дій (ФД). Кожні стадії можна поставити у відповідність деякий стан системи. Нехай кожна стадія характеризується номером j. Припустимо, при j = 1 здійснюється завантаження в ОП і-ої програми. При цьому стан ОС позначається S_1. На цій стадії виконується послідовність функціональних дій $ФД_{i1}$: завантаження в програмний лічильник початкової адреси, установка (якщо потрібно) слова стану процесора тощо. На другій стадії здійснюється виконання програми, що є послідовністю функціональних дій $ФД_{i2}$. Нарешті, виведення результату й установка процесора у вихідний стан відбувається на третій стадії - функціональна дія $ФД_{i3}$. Обслуговування у ОС одночасно декількох програм є складним процесом, у якому кожна і-я програма може розглядатися як компонента K_i.

Зробимо наступне узагальнення. Функціонування компоненти K_i являє собою послідовність $ФД_{ij}$. Можна стверджувати, що перехід системи в стан S_{ij} і виконання нею послідовності дій $ФД_{ij}$ ініціюється подією S_{ij}. Кожна з подій у реальній системі пов’язано, як правило, з відповідною компонентою K_i. При цьому будь-яка функціональна дія $ФД_{ij}$ виконується на деякому інтервалі τ_{ij}. Для кожної K_i введемо поняття локального часу τ_{ij}. Тому що всі K_i в системі функціонують одночасно, очевидно й їхні локальні часи t_i змінюються одночасно, однак характер цих змін різний і визначається послідовністю часових інтервалів τ_{ij}.

Реальним функціональним діям $ФД_{ij}$, що відбуваються в системі, в імітаційній моделі (ІМ) ставиться у відповідність деяка їхня апроксимація $ФД'_{ij}$. Ступінь такого спрощення визначає рівень деталізації ІМ. Відмінності $ФД'_{ij}$ від $ФД_{ij}$ породжують помилки імітації реальної системи. В ІМ $ФД_{ij}$ зображуються парою ($ФД'_{ij}$, τ_{ij}), що виконується в такий спосіб. Спочатку реалізується $ФД'_{ij}$ при незмінному значенні t_i, а потім уже відображається зміна t_i на величину τ_{ij}, ініціюючи, таким чином, появу події S_{ij}.

На рисунку 3.5 наведений приклад розвитку процесу обробки K_i в системі координат ($ФД'_{ij}$, τ_{ij}). Звичайно, подібна система координат має умовний характер, оскільки по осі ординат не можна відкласти “значення” функціональної дії $ФД'_{ij}$. Подібне уявлення використовується тільки для зображення того факту, що зі зміною локального часу t_i деяка K_i виконує кілька різних $ФД_{ij}$. Так, відповідно до рисунка 3.5, K_i послідовно виконує $ФД_{i1}$, $ФД_{i2}$, $ФД_{i3}$, при цьому в системі відбуваються події S_{i0} ,S_{i1}, S_{i2}, S_{i3}.Причому з появою кожного нового S_{ij} відбувається відповідне виконання $ФД_{ij}$ і збільшення його часової координати t_i відповідно на величину τ_{i1}, $\tau_{i2,}$ τ_{i3}. В ІМ поява подій реалізується східчастою лінією (0,a, S_{i1}, b, S_{i2}, d, S_{i3}). Це означає, що в ІМ спочатку виконується $ФД'_{i1}$ при незмінному t_i, а потім вже відображається зміна t_i на величину τ_{i1}, ініціюючи появу події S_{i1}. Потім реалізується $ФД'_{i2}$ при незмінному часі t_i і відображається зміна t_i на величину τ_{i2}, ініціюючи в K_i здійснення події S_{i2}. Далі в два прийоми:

реалізація ФД'$_{i3}$ і наступна зміна t_i на величину τ_{i3} так ініціюється подія S_{i3}. Відзначимо при цьому, що в ІМ порядок появи події може бути й зворотним: спочатку змінюється t_i, а потім вже виконується ФД'$_{ij}$.

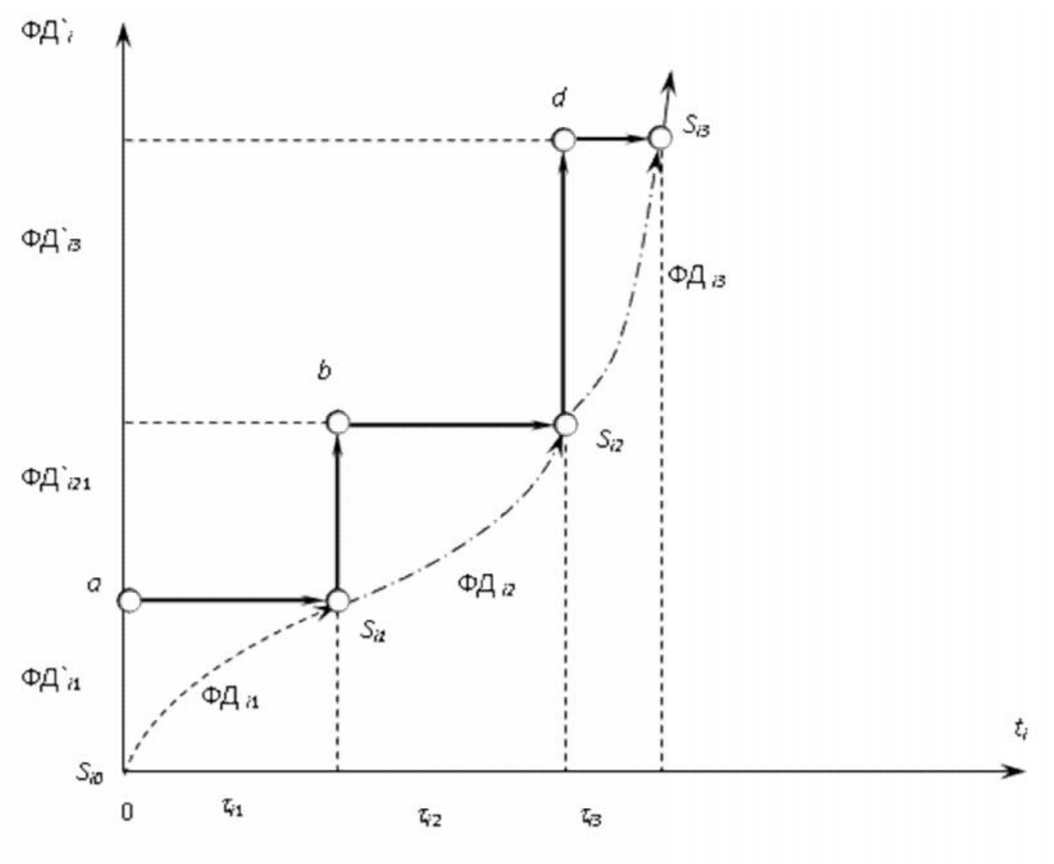


Рисунок 3.5 – Апроксимація функціональних дій об'єкта Кі у моделі системи

В ІМ кожне ФД'$_{ij}$ описується, у загальному випадку, деяким алгоритмом АЛ$_{ij}$. У ході імітації відбувається реалізація ФД'$_{ij}$ за відповідними алгоритмами АЛ$_{ij}$ і наступна зміна t_i на величину τ_{ij}. Таким чином, будь-яка ІМ описується набором деяких "молекул", кожна з яких містить у собі опис алгоритму АЛ$_{ij}$ виконання відповідного ФД'$_{ij}$ і оператора Mt$_{ij}$, що здійснює зміну тимчасової координати t_i на величину τ_{ij}. Пари (АЛ$_{ij}$, τ_{ij}) називатимемо ij-й активністю ІМ та позначати АК$_{ij}$. Кожна ij-я активність є записом поводження компоненти $К_i$ в імітаційній моделі системи. Реалізація цієї компоненти в ІМ приводить до появи в моделі системи події S_{ij}.

Якби на ЕОМ імітувалося поводження тільки однієї компонента системи, то виконання активностей в ІМ можна було б здійснити строго послідовно, і справа звелася б до перерахування часової координати t_i після чергового виконання алгоритму $АЛ_{ij}$. Насправді будь-яка складна система складається з декількох компонентів. Усі ці компоненти K_i можуть функціонувати одночасно. Це повинна відбивати ІМ. У більшості однопроцесорних ЕОМ у кожний момент часу може реалізуватися алгоритм тільки однієї з компонент моделі.

Щоб забезпечити імітацію паралельних подій реальної системи, уводять деяку глобальну перемінну t_m, що називають модельним (системним) часом, або таймером модельного часу. За допомогою цієї перемінної організуються синхронізація всіх подій S_{ij} у моделі і виконання алгоритмів $АЛ_{ij}$ компонентів K_i моделі системи. При реалізації ІМ використовується звичайно три уявлення часу:

t_r – реальний час системи, робота якої імітується за допомогою даної ІМ;

t_m – модельний час, за яким організується синхронізація подій у системі;

$t_э$ – машинний час імітації, що відображає витрати ресурсу часу ЕОМ на організацію імітації.

За допомогою модельного часу t_m реалізується квазіпаралельна робота компонент ІМ [14,17,38]. Приставка "квазі" у даному випадку відображає послідовний характер обслуговування подій у ІМ, які одночасно виникають у різних компонентах реальної системи.

Коректування тимчасових координат t_i декількох K_i в ІМ здійснюється за допомогою модельного часу t_m таким чином. Якщо значення t_i при виконанні $АЛ_{ij}$ декількох K_i збігаються (це означає, що в реальній системі відбувається одночасно кілька подій S_{ij}), то послідовно обслуговуються $АЛ_{ij}$, що збігаються за часом виконання, тобто ті, що мають однакові значення t_{ij}. Тут і далі під t_{ij} будемо розуміти конкретні значення t_i, при яких відбувається подія S_{ij}. При цьому модельний час t_m не змінюється до закінчення виконання всіх алгоритмів $АЛ_{ij}$, що збіглися за часом реалізації. Таким чином послідовно виконуються відповідні $ФД'_{ij}$ при незмінному значенні t_m. Після кожної реалізації $АЛ_{ij}$, що забезпечує виконання в ІМ $ФД'_{ij}$, виконується оператор Mt_i, котрий корегує часову координату t_i. Фактично це коректування зводиться до обчислення нового значення t_i за формулою:

$$t_i = t_m + \tau_{ij}.$$

Це нове значення часової координати t_i запам'ятовується і використовується надалі для визначення моменту нової активізації компоненти K_i в ІМ. Під активізацією компонента моделі K_i розумітимемо початок виконання наступної її активності (виконання алгоритму $АЛ_{ij}$ і оператора Mt_i коректування часової координати t_i).

Коли імітація подій Sij, що з'явилися одночасно, закінчена, тобто виконані відповідні алгоритми активностей $АЛ_{ij}$ та проведено коректування часових координат t_i, змінюється значення глобальної перемінної моделі t_m. Існують два способи зміни t_m: за допомогою фіксованих і перемінних інтервалів зміни модельного часу. Часто їх називають відповідно спосіб фіксованого кроку і спосіб кроків до наступної події.

Для того, щоб легше було уявити обидва способи організації зміни модельного часу, розглянемо наступний приклад (рисунок 3.6). Нехай у системі функціонують три компоненти K_i (i=1,2,3). При функціонуванні K_1 послідовно відбуваються чотири події (S_{11}, S_{12}, S_{13}, S_{14}) відповідно в чотири моменти часу t_1 (t_{11},t_{12},t_{13},t_{14}). Між цими моментами K_1 виконує чотири різних функціональних дії ($ФД_{11}$, $ФД_{12}$, $ФД_{13}$, $ФД_{14}$). Кожна з зазначених функціональних дій виконується протягом відповідних інтервалів часу (τ_{11}, τ_{12}, τ_{13}, τ_{14}). Апроксимація $ФД_{ij}$ здійснюється послідовністю ($ФД'_{1j}$, j=1,2,3,4). Так само компонента K_2 послідовно виконує три функціональні дії ($ФД_{2j}$, j=1,2,3) відповідно протягом трьох інтервалів часу (τ_{2j}), які в ІМ апроксимуються відповідно ФД'$_{2j}$. Нарешті, компонента К$_3$ також послідовно виконує три функціональних дії (ФД$_{3j}$, j=1,2,3) відповідно протягом трьох інтервалів часу (τ_{3j}), що апроксимуються в ІМ відповідно трьома ФД'$_{3j}$. Відзначимо, що кожна ФД'$_{ij}$ описується відповідним алгоритмом АЛ$_{ij}$ і реалізується відповідною активністю, що являє собою виконання цього алгоритму при незмінному значенні t_i та модифікацію часової координати t_i на величину τ_{ij}. Алгоритм функціонування в координатах (ФД'$_i$, t_i) зображений на рисунку 3.6. Протягом виконання активностей (АЛ$_{ij}$, τ_{ij}) у моделі відбуваються послідовно події S_{ij}.

Розглянемо імітацію подій S_{ij} у системі при кожному зі способів зміни t_m. При імітації за способом фіксованого кроку модельного часу, позначений на рисунку 6.6 як t'_m, змінюється щораз на величину кроку Δt. Тоді в моменти часу 0, Δt_1, $2\Delta t_1$, $3\Delta t_1$ тощо. В ІМ відбуваються відповідно події s_0, s_1, s_2, s_3 тощо. Моделювання в ІМ усієї системи кожної з цих подій, практично означає обслуговування керуючою програмою моделювання (КПМ) тих подій S_{ij}, що потрапляють до відповідного інтервалу Δt. Як видно з рисунка 6.6, події S_{11}, S_{31}, S_{21} відбуваються на інтервалі (s_0, s_1). Вважається, що вони відбуваються одночасно в момент $t'_m = \Delta t$. КПМ послідовно обслуговує ці події, що зводиться до ініціалізації та виконанню відповідних активностей (АЛ$_{11}$, τ_{11}; АЛ$_{31}$, τ_{31}; АЛ$_{21}$, τ_{21}). Потім у момент $t'_m = 2\Delta t$ КПМ ініціалізує подію s_2. Вважається, що події S_{12} і S_{22} також відбуваються одночасно, тому КПМ послідовно ініціює виконання активностей (АЛ$_{12}$, τ_{12}) та (АЛ$_{22}$, τ_{22}). Час t'_m одержує збільшення на заздалегідь обрану величину Δt, а моменти реагування КПМ на появу подій s_i не пов'язані з моментами t_{ij} появи подій у реальній системі. Таким чином, точність моделювання подій S_{ij} визначається кроком Δt зміни модельного часу t'_m і, як правило, усі події обслуговуються в точці, що відповідає верхній границі інтервала зміни t'_m.

При імітації за способом кроків до наступної події час t''_m змінюється в моменти m_i, що відповідають моментам t_{ij} появи подій у реальній системі.

Обробку подій S_{ij}, що одночасно з’являються в реальній системі, КПМ здійснює послідовно при незмінному модельному часі t''_m. Наприклад, події S_{12} та S_{22} у реальній системі відбуваються одночасно. У моделі спочатку реалізується, наприклад, алгоритм $АЛ_{12}$ та модифікується координата t_1 на величину τ_{12}, а потім виконується алгоритм $АЛ_{22}$ та модифікується координата t_2 на величину τ_{22}. Таким чином, t''_m щораз зсувається на величину, що дорівнює мінімальному значенню збільшення модельного часу τ_{ij}, яке одержують координати t_i.

Незалежно від способу зміни t_m механізм регламентації зміни модельного часу звичайно передбачає виконання наступних дій:

- вибір подій у моделі, які необхідно обслужити при тому самому модельному часі t_m;
- обслуговування подій (ініціалізація активностей), що мають однаковий час ініціалізації;
- по закінченні обслуговування всіх одночасних (у межах кроку) подій визначення чергового значення модельного часу;
- коректування часової координати моделі t_m;
- перевірка умов закінчення моделювання або за часом завершення імітації, або за виконанням інших подій у системі.

Усі ці дії виконує КПМ. Кожна подія S_{ij} у моделі системи, що є результатом виконання активності $АЛ_{ij}$, обслуговується КПМ і вимагає для реалізації деякого ресурсу часу роботи ЕОМ. Тому чим частіше обслуговуються події КПМ, тим більше зростає машинний час $t_э$. З цього погляду спосіб фіксованого кроку зміни t_m є більш економічним, оскільки має місце групове обслуговування всіх подій S_{ij}, що потрапили усередину чергового кроку зміни Δt модельного часу.

На практиці перевага способові фіксованого кроку віддається в двох випадках. По-перше, коли події S_{ij} розподілені рівномірно на всьому інтервалі моделювання і дослідник може обирати інтервал зміни тимчасової координати Δt, що забезпечує мінімальну погрішність імітації. По-друге, коли подій дуже багато і вони з’являються групами. В усіх інших випадках спосіб завдання кроку до наступної події більш кращий. У тих випадках, коли події S_{ij} розподілені нерівномірно і з’являються вони через значні тимчасові інтервали τ_{ij}, спосіб завдання кроку до наступної події економічний і точніший ніж спосіб фіксованої зміни t_m. З урахуванням достоїнств і недоліків кожного зі способів зміни t_m намітилася спеціалізація їхнього застосування при моделюванні складних систем.

На практиці найбільше поширення одержав спосіб кроку до наступної події. Тому надалі розглядатимемо всі методологічні питання, пов’язані тільки з другим способом зміни модельного часу.

4 МОДУЛЬ УПРАВЛІННЯ ІМІТАЦІЙНОЇ СИСТЕМИ

4.1 Структура об’єктно-орієнтованих систем моделювання

Як уже встановлено, в ІМ відбувається апроксимація кожної $ФД_{ij}$ парою дій: спрощеною функціональною дією $ФД'_{ij}$ і модифікацією тимчасової координати і-ої компоненти моделі t_i. Ця пара дій являє собою “молекулу” ІМ і називається активністю $АК_{ij}$. Будь-яка $АК_{ij}$ складається з опису алгоритму $АЛ_{ij}$ і оператора модифікації тимчасової координати компоненти моделі Mt_{ij}. Для синхронізації виконання компонент ІМ на ЕОМ використовується глобальна змінна, називана модельним часом t_m. Крім модифікації модельного часу t_m і виконання операторів Mt_{ij} у функції КПМ входять: запуск на виконання алгоритмів $АЛ_{ij}$, організація взаємодії $АЛ_{ij}$ один з одним у ході імітації та перевірка умов закінчення імітації. Таким чином, кожна ІМ являє собою сукупність набору “об’єктів”, які відбивають поводження об’єкта імітації, і КПМ, що організує взаємодію цих “об’єктів” один з одним.

На рисунку 4.1 приведена структурна схема об’єкта моделювання і відповідної об’єктно-орієнтованої системи моделювання [14,42]. Для завдання вхідних умов, початкових значень параметрів і запуску ІМ до складу моделі включається підпрограма (ПП), яка організує початок імітації. ІМ створюється для вивчення на основі статистики, що характеризує поводження компонент K_i, реальної системи в різних режимах її функціонування. Для збору статистики моделювання в ІМ уводиться відповідна підпрограма, яка присутня в ІМ або в явному виді, або розосереджена по всіх компонентах ІМ. Нарешті, по досягненні умов закінчення імітації КПМ передає керування на підпрограму закінчення імітації, що найчастіше й забезпечує розрахунок характеристик поводження моделі системи і видає дослідникові результати моделювання. Всі ці підпрограми взаємодіють тільки з КПМ.

Таким чином, опис ІМ істотно розростається в обсязі в порівнянні з власне описом об’єкта імітації. Можна говорити про дві частини ІМ. Перша частина ІМ є змінною та створюється дослідником системи за допомогою наявних засобів автоматизації моделювання. Друга частина ІМ (ядро імітаційної системи) являє собою засоби реалізації автоматизації моделювання у вигляді: процедур, які забезпечують виконання операторів синхронізації моделі (наприклад, Mt_{ij}), підпрограм запуску і завершення імітації, КПМ і підпрограми збору статистики моделювання.

З рисунка 4.1 також видно, що будь-яка $ФД_{ij}$ у ІМ описується відповідною активністю $АК_{ij}$. Кожна $АК_{ij}$ являє собою опис алгоритму $АЛ_{ij}$ і оператора модифікації тимчасової координати компоненти моделі Mt_{ij} . КПМ передає керування на початок програми, що реалізує алгоритми $АЛ_{ij}$. Повернення на КПМ має місце при виконанні операторів Mt_{ij}. КПМ і службові підпрограми звичайно є універсальними і не змінюються при переході від однієї моделі до іншої. Однак принципи їхньої побудови та спосіб керування виконанням активностей залежать від класу об’єктів моделювання. Питання організації квазіпаралельного виконання активностей цікаві для фахівців зі створення засобів автоматизації проектування. Складання ж тексту опису ІМ і використання системних засобів при розробці $АЛ_{ij}$ є областю інтересів дослідників функціонування складних систем.

У моделях складних систем склад активностей у різних компонентах і характер їхньої взаємодії один з одним можуть бути різними. В одних системах усі $ФД_{ij}$ різні і для реалізації кожного алгоритму $АЛ_{ij}$ активності в моделі потрібне виконання своїх умов. В інших системах деякі $ФД_{ij}$ у різних компонентах аналогічні, збігаються за часом і приводять до однієї і тій ж події S_{ij}. Для реалізації в ІМ кожної такої групи $ФД'_{ij}$ одним алгоритмом $АЛ_{ij}$ потрібне виконання своїх визначених умов. Для деяких систем існує сильна взаємодія між $ФД_{ij}$, які можна апроксимувати математичними залежностями, що задаються явно, при створенні ІМ. У залежності від складу алгоритмів

$АЛ_{ij}$, наявності зв'язків між компонентами, цілей і задач моделювання вибирається той або інший спосіб представлення $К_i$ і реалізації активностей в імітаційних моделях.

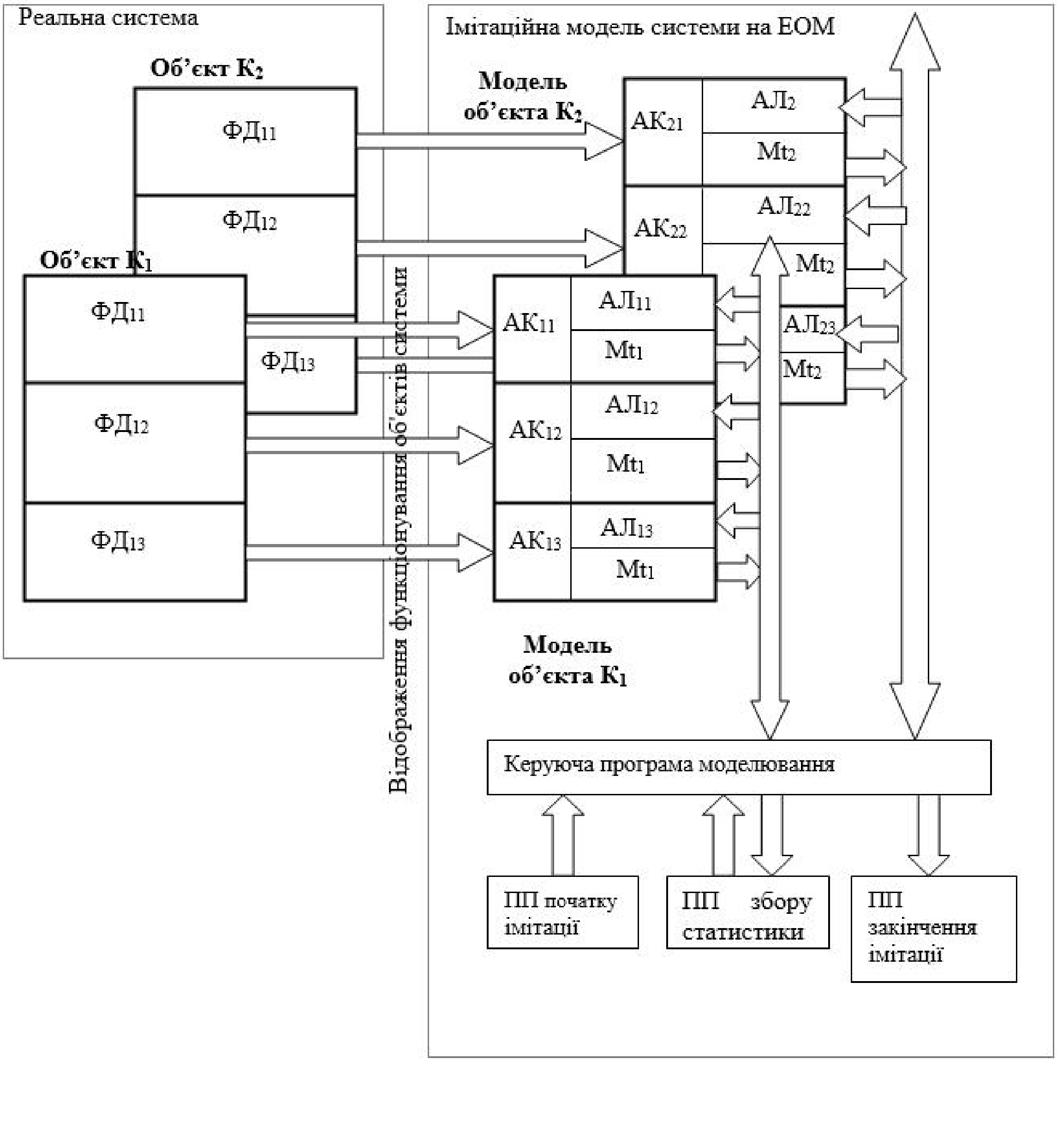


Рисунок 4.1 – Структурна схема об'єкта моделювання і відповідної об'єктно-орієнтованої системи моделювання

Найчастіше складання опису ІМ називають формалізацією об'єкта. Найбільше поширення одержали п'ять способів опису ІМ: безпосередньо активностями, апаратом подій, транзактами, агрегатами, процесами.

Кожному способові формалізації об’єкта моделювання відповідає свій спосіб організації квазіпаралелізму обслуговування КПМ активностей, з яких складена ІМ. Тому розрізняють відповідно наступні способи організації квазіпаралелізму в ІМ: перегляд активностей, складання розкладу подій, керування обслуговуванням транзактів, керування агрегатами, синхронізація процесів.

4.2 Організація квазіпаралелізма в імітаційних моделях

Одну й ту ж складну систему принципово можна представити кожним із зазначених у попередньому підрозділі способів формалізації, однак ІМ, побудовані на їхній основі, відрізнятимуться розмірами та кількістю ресурсів, що були затрачені на їхнє створення, випробовування та використання. Зупинимося на способі складання розкладу подій при організації квазіпаралельного обслуговування активностей з боку КПМ. Ця проблема становить інтерес не тільки для фахівців з розробки засобів автоматизації проектування, але і для користувачів цих засобів при створенні своїх ІМ [28].

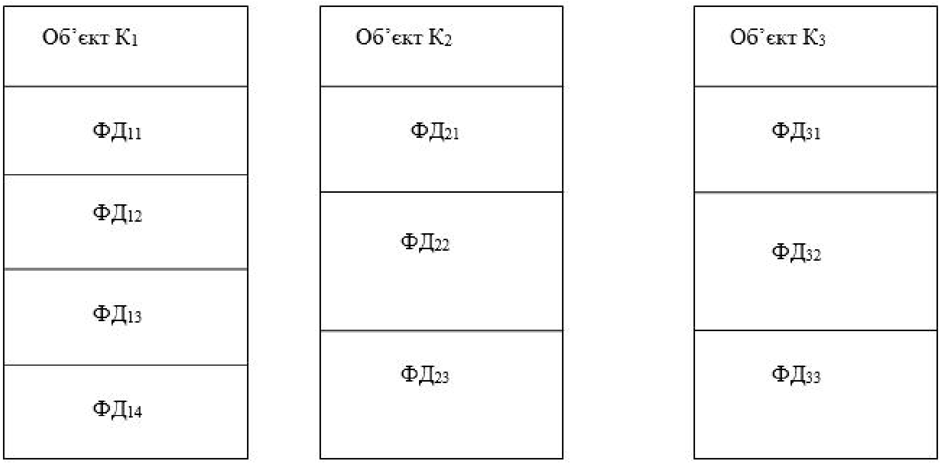


Рисунок 4.2 – Структура об’єкта, який використовується для ілюстрації способів формалізації і моделювання

Відмінні риси і принципи різної організації квазіпаралелізма в ІМ розглянемо на прикладі, у якому як об’єкт моделювання обрана система, що складається з трьох компонентів K_i (рисунок 4.2). Відповідно до часової діаграми переходу K_i до стану S_{ij} у системі координат $ФД'_{ij}$ час його виконання визначено наступне. Компонента K_1 виконує чотири функціональних дії ($ФД_{11}$, $ФД_{12}$, $ФД_{13}$, $ФД_{14}$) у часі t_1, що у ІМ описуються відповідно чотирма активностями. Аналогічним образом компонента K_2 і K_3 виконують по три функціональні дії ($ФД_{2j}$, $ФД_{3j}$, j=1,3) у часах відповідно t_2 і t_3. Опускаючи для простоти деякі деталі, розглянемо основні функції КПМ і принципи її взаємодії з компонентами моделі системи K_i при використанні способу перегляду активностей та складання розкладу подій.

ВИСНОВКИ

При реалізації даного програмного комплексу моделювання були вирішені наступні завдання:

- розглянуті характеристики електронних компонентів і запропоновані вимоги їх моделям системного рівня;

- розроблена процессная модель елемента складної електронної системи;

- організовано взаємодію процесів, як елементів складної системи;

- забезпечена універсальність моделі складної системи за рахунок введення широкого набору налаштувань параметрів системи;

- розроблена структура і алгоритм моделює програми з використанням процесного підходу;

- програмно реалізований алгоритми: складання черги планування подій і алгоритм модифікації тимчасової координати і управління чергою подій;

Всі завдання, поставлені в атестаційній роботі, виконані успішно. Даний програмний комплекс моделювання має передумови для подальшої модернізації і розширення функціональних можливостей системи моделювання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гультяев А. Визуальное моделирование в среде MATLAB: учебный курс – СПб.: Питер, 2000.- 432с.: ил.

2. Беньковии Е.С., Колесов Ю.Б., Саниченков Ю.Б. Практическое моделирование динамических систем – СПб.: БХВ-Петербург, 2002.- 454с.: ил.

3. Кухтенко А.И. Обзор основных направлений развития общей теории систем. – К.: Изд. Научного совета по кибернетике АН УССР,1969, - 74 с.

4. Моисеев Н.Н. Математические задачи системного анализа. – М.: Наука, 1981, - 488 с.

5. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых ситем.- М.: Мир, 1973. – 344с.

6. Иванников А.Д. Моделирование микропроцессорных систем. – М. Энергоатомиздат, 1990. – 144 с.

7. Ларионов А.М. и др. Вычислительные комплексы, системы и сети. – Л.: Энергоатомиздат, 1988. – 288 с.

8. Вейцман К, распределительные системы мини- и микро-ЭВМ. –М.: Финансы и статистика. 1982. -382 с.

9. Клейнрок Л. Теория массового обслуживания /Пер. с англ. И. И. Глушко. – М.: Машиностроение, 1979. – 432 с.

10. Бююль А., Цёфель П. SPSS: искусство обработки информации. Анализ статистических данных и восстановление скрытых закономерностйей: Пер. с нем./А. Бююль, П.Цёфель – Спб.: ООО «Диа СофтЮП», 2002.- 608 с.

11. Феррари Д. Оценка производительности вычислительных систем / Пер. с англ. А. И. Горлина, Ю. Б. Котлова и Л. В. Ухова; Под ред. В. В. Мартынюка. – М.: Мир, 1981. – 576 с.

12. Бусленко Н.П. Моделирование сложных систем. – М.: Наука, 1978.-400с.

13. Советов Б.Л., Яковлев С.А. Моделирование систем: Учебник для вузов по спец. «Автоматизация систем управления». – М.: Высш. шк., 1985. – 217 с. ил.

14. Максимей И.В, Имитационное моделирование на ЭВМ. – М.: Радио и связь, 1988. – 232 с.

15. Гради Буч, "Объектно-ориентированный анализ и проектирование, с примерами на С++", второе издание - СПб.: Бином, 2001 г.- 560 стр.

16. Дж. Амстронг. Моделирование цифровых систем на языке VHDL.- М.: Мир, 1992.- 175 с.

17. Технология системного моделирования /Е.Ф. Аврамчук, А.А. Вавилов, С.В. Емельянов и ДР.; Под общ. ред.. С.В. Емельянова и др. – М.: Машиностроение; Берлин: Техник, 1988. – 520 с.

18. Логическое проектирование дискретных устройств. Глушков В.М., Капитонова Ю.В., МищенкоА.Т.- Киев: Наук. Думка, 1987.- 264с.

19. Поспелов Д.А. Вероятностные автоматы.М.,1970

20. Столлингс В. Операционные системы, четвертое издание.: Пер. с англ.. – М.: Издательский дом «Вильямс», 2002.- 846 с.: ил. – Парал. тит. англ.

21. Э.Иордон, К.Аргила, "Структурные модели в объектно-ориентированном анализе и проектировании" – М.: изд. "Лори", 1999 г.

22. Г.С. Иванова, Т.Н. Нигушкина, Е.К., "Объектно-ориентированное программирование", уч. Пособие для вузов/Под ред. Г.С. Ивановой – М.: изд. МГУ им. Баумана, 2001 – 320с.

23. Э. Боггс, М. Боггс UML и Rational Rose – М.: изд. "Лори", 2001 г.

23. Lazowska T., Zahorjan J., Graham G., Sevcik, Quantitative System Performance. Computer System Analysis Using Queuing Network Models. Prentice-Hall Inc., 1984.-412 p.

24. Вентцель Е. С. Теория вероятностей. – М.: Наука, 1969. – 576 с.

25. Вероятностные методы в вычислительной технике: Учеб. пособие для вузов по спец. ЭВМ /А. В. Крайников, Б. А. Курдинов, А. Н. Лебедев и

др.; Под ред. А. Н. Лебедева и Е. А. Чернявского. – М.: Высш. шк., 1986. – 312 с.

25. Филлер В. Введение в теорию вероятностей и ее приложение: 2 т. – М., 1964. – 498 с.; 1967. – 752 с.

26. Вентцель Е. С. Исследование операций. – М.: Сов. радио, 1972. – 552 с.

27. Скороход А.В. Лекції з теорії віпадкових процесів: Навч. посібник.- К. : Либідь, 1990. – 168с.

28. Гнеденко Б. В., Бєляєв Ю. К., Соловйов А. Д. Математические методи в теории надежности.- М.: Наука, 1965.- 524 с.

29. Основы теории вычиислительных систем /С.А. Майоров, ТГ.И, Новиков, Т.И, Алиев и др.; Под ред. С.А. Майорова.- М.:Высш. Шк., 1978. – 408 с.

30. Дрим М., Коутский З. Вопросы получения случайных величин при помощи счетчиков. «Журнал вычислительной математики и математической физики», 1962, т. 2, V-VI, №3, с. 475.

31. Полляк Ю.Г. Вероятностное моделирование на электронных вычислительных машинах. М., Изд-во „Советское радио”, 1971 г.

32. Апанасович В.В., Тихоненко О.М. Цифровое моделирование стохастических систем: Учеб. Пособие для ун-тов по спец. „Радиофизика и электроника”. – Мн.: узд-во „Университетское”, 1986. – 127 с.: ил.

33. Томашевский В., Жданова Е. Имитационное моделирование в среде GPSS. – М.: Бестселлер, 2003. – 416 с.

34. Молчанов А.А, Моделирование и проектирование сложных систем. – К. Высшая шк.., 1988. – 359 с.

35. Хинчин А.Я. Математические методы теории массового обслуживания // Труды математического института им. В.А. Стеклова. – М.: Изд. АН СССР. – Т. 48, 1955. – 122 с.

36. Banks J., Discrete-event system simulation. Prentice-Hall Inc., 1995.- 548 p.

37. Шеннон Р. Имитационное моделирование систем – наука и искусство. –М.: Мир, 1978. -418.

38. Литвинов В.В., Т.П. Марьянович, Методы построоения имитационных систем. – К.: Наук. Думка, 1991. – 120 с.

39. Hamilton J., Nash D., Pooch Udo, Distributed simulation. CRC Press LLG, 1997.- 392 p.

40. Nain Ph., Basic Elements of Queuing Theory. Application to the Modeling of Computer Systems. Lecture Notes. INRIA. 1998.- 110 p.

41. Рыжиков Ю.И. Имитационное моделирование. Теория и технологии. – Спб.: Корона принт; М.: Альтек – А, 2004. – 384 с., ил.