Т. О. Савчук, О. В. Ольшанська

Технології комп’ютерного проєктування

Міністерство освіти і науки України
Вінницький національний технічний університет

Т. О. Савчук, О. В. Ольшанська

Технології комп’ютерного проєктування

Електронний навчальний посібник
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2024

УДК 004.41(075)
С13

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 25.01.2024 р.)

Рецензенти:
Анатолій КУЛІК, доктор технічних наук, професор, завідувач кафедри біофізики, інформатики та мед. апаратури Вінницького національного медичного університету ім. М. І. Пирогова
Євген ПАЛАМАРЧУК, кандидат технічних наук, професор кафедри АІІТ, Директор Центру дистанційної освіти ВНТУ
Олеся ВОЙТОВИЧ, кандидат технічних наук, доцент кафедри захисту інформації, директор Центру забезпечення якості освіти ВНТУ

Савчук, Т. О.
С13 Технології комп'ютерного проєктування : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / Т. О. Савчук, О. В. Ольшанська. – Вінниця : ВНТУ, 2024. – 125 с.

У навчальному посібнику наведено теоретичні відомості за тематикою освітньої компоненти «Технології комп'ютерного проєктування», приклади їх застосування для розв'язування практичних завдань, а також запропоновано кейси для перевірки здобутих знань та вмінь.

УДК 004.41(075)

ЗМІСТ

ВСТУП

В процесі розвитку науки і техніки системи, пристрої й споруди, що створюються людиною, стають дедалі складнішими. Одночасно вимоги, які висуваються до термінів проєктування нових виробів, стають більш жорсткими. За цих умов некомп'ютерні методи проєктування стають неефективними. Створення та широке застосування систем комп'ютерного проєктування (СКПР) стало вкрай необхідним. Для розв'язання цієї важливої задачі необхідна відповідна підготовка фахівців.

Концепції багаторівневих СКПР, що здійснюють повний цикл проєктування, а також розвиток теорії штучного інтелекту надали підгрунтя для розвитку та впровадження нових підходів до комп'ютерного проєктування складних об'єктів і систем.

Не зважаючи на появу нових цікавих напрямків в галузі науки і техніки, комп'ютерне проєктування й зараз лишається невід'ємною складовою частиною науково-технічного прогресу. До того ж за роки існування СКПР фахівцями було висунуто багато нових ідей, розроблено велику кількість сучасних методів та алгоритмів проєктування складних об'єктів.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО КОМП'ЮТЕРНЕ ПРОЄКТУВАННЯ

1.1 Сучасні технології проєктування складних об'єктів

Розвиток науки та техніки передбачає зростання складності задач та проєктованих засобів, і, відповідно, збільшення обсягу робіт під час їх дослідження та розроблення, що визначило актуальність задачі автоматизування процесів розробки засобів як складних обєктів, тобто розвитку комп'ютерного проєктування.

Забезпечити підтримку найбільш трудомістких етапів розробки складних програмних об'єктів на етапах аналізу та проєктування дозволяє використання CASE-засобів.

Появі CASE-технологій сприяли [1]:

- впровадження мережевих технологій, що забезпечило можливість об'єднання зусиль окремих виконавців у єдиний процес проєктування шляхом використання розподіленої бази даних, яка містить необхідну інформацію про проєкт;
- розвиток модульного і структурного програмування та наявність відповідних аналітиків і програмістів;
- зростання продуктивності комп'ютерів, що дозволило використовувати ефективні графічні засоби та автоматизувати етапи проєктування.

Під терміном CASE-засоби (Computer-AidedSoftware/ SystemEngineering) розуміють засоби, що реалізують технологію аналізу, проєктування, розробки і супроводу складних систем програмного забезпечення на основі взаємопов'язаних програмних продуктів [1]; набір інструментів і методів для проєктування програмного забезпечення, який допомагає забезпечити високу якість розроблюваних програм, оптимізацію їх структури. CASE-технологія базується на методології системного

аналізу, під яким розуміють наукову дисципліну, що розробляє загальні принципи дослідження складних об’єктів і процесів з урахуванням їхнього системного характеру, концентруючись на початкових етапах розробки. В рамках CASE-технології системний аналіз призначений для відділення проєктування від програмування. У розробці відповідно до CASE-технологією виділяються побудова архітектури та її подальша реалізація, тому системний аналіз називають структурним системним аналізом або просто структурним аналізом.

CASE-засоби – це програмні засоби, що підтримують методологію проєктування інформаційних систем, а також набір інструментальних засобів, що підтримують процеси створення та супроводу інформаційних систем [1], моделювання предметної області, включно й аналіз і формування вимог, проєктування процесів та баз даних, генерацію коду, тестування, документування, забезпечення якості, конфігураційне управління й управління проєктом.

Використання CASE-технологій дозволяє:

- поліпшити якість створюваного програмного забезпечення за рахунок використання комп’ютерного проєктування;
- прискорити процес проєктування і розробки;
- забезпечити підтримку супроводу розробки;
- забезпечити підтримку технологій повторного використання;
- забезпечити всіх учасників проєкту, включно й замовників, єдиною графічною мовою для розуміння структури;
- використовувати бази даних проєкту (репозиторію) для зберігання всієї інформації про проєкт, що може розподілятися між розробниками відповідно до їх прав доступу;
- підтримувати групову роботу над проєктом в мережі, експорт-імпорт будь-яких фрагментів проєкту, а також планування, контроль, керівництво і взаємодію складових складного об’єкта;

- автоматично генерувати документацію, зокрема програм, за результатами проєктування з урахуванням змін у часі;
- автоматично верифікувати і контролювати проєкт на функціональну повноту і обґрунтованість рішень на всіх рівнях розробки складного об’єкта;
- створення моделі складного об’єкта з його кодів і інтеграцію отриманих моделей в проєкт, автоматичне оновлення документації у разі зміни кодів і т. п. за рахунок використання засобів реінжинірингу та зворотного реінжинірингу.

Більшість існуючих CASE-засобів основано на парадигмі «метод – нотація – засіб» і методологіях структурного, об’єктно-орієнтованого аналізу та проєктування, що використовують документацію у вигляді діаграм або текстів, зв’язків між моделями системи, динаміку поведінки складного об’єкта та архітектури програмних засобів. CASE-технології пропонують новий, оснований на автоматизації підхід до концепції життєвого циклу програмного забезпечення. За використання CASE змінюються всі фази життєвого циклу, водночас найбільші зміни стосуються фаз аналізу та проєктування.

До CASE-засобів прийнято відносити будь-яке програмне забезпечення, що автоматизує процеси його життєвого циклу та має такі характерні особливості:

- наявність потужних графічних засобів, що використовуються для опису і документування ІС та забезпечують зручний інтерфейс з розробником;
- інтеграцію окремих компонентів CASE-засобів з метою забезпечення керованості процесом розробки ІС;
- використання репозитарію – спеціальним чином організованого сховища проєктних метаданих.

Всі сучасні CASE-засоби класифікуються за типами і категоріями. Класифікація за типами передбачає функціональну орієнтацію CASE-засобів на процеси під час проєктування складних об’єктів. Класифікація за категоріями визначає ступінь інтегрованості за виконуваними функціями.

Нині існує класифікація CASE-засобів за такими ознаками:

- за типами – визначає функціональну орієнтацію CASE-засобів на будь-які процеси життєвого циклу;
- за категоріями – визначає рівень інтегрованості за виконуваними функціями (локальні засоби, комплект частково інтегрованих засобів, повністю інтегровані засоби);
- за ступенем інтегрованості з системами керування базами даних (СКБД);
- за доступними платформами;
- за застосовуваними методологіями та моделями систем і БД.

Класифікуючи CASE-засоби за можливостями можна виділити такі типи:

1. Верхні CASE-системи (Upper CASE) – засоби аналізу, що використовуються для побудови та аналізу моделей предметної області (BPWinLogicWorks). Ці системи відповідають основним поняттям терміна CASE, тому їх також називають нормальними.

2. Середні CASE-системи (Middle CASE) – засоби аналізу і проєктування, які підтримують поширені методології проєктування та використовуються для створення проєктних специфікацій (VantageTeamBuilder,Designer/2000). Забезпечують формування архітектури системи, складових та інтерфейсів системи, алгоритмів і пристроїв даних.

3. Засоби розробки застосунків (Power Builder, 4GL (Uniface Compuware)), SQL Windows, а також генератори кодів.

4. Засоби реінжинірингу, призначені для аналізу програмних кодів і схем баз даних та створення на їх базі моделей і проєктних специфікацій (Vantage Team Builder, ERwin).

5. Засоби проєктування баз даних – засоби моделювання даних, генерування схем баз даних для поширених систем управління базами даних (Erwin, DataBaseDesigner).

Саме BPwin (All Fusion Process Modeler) і ERrwin (All Fusion Erwin Data Modeler) нині є найбільш популярними CASE-засобами, що входять в пакет All Fusion Modeling Suite, який інтегрує комплекс CASE-засобів для розробки програмного забезпечення. Цей пакет призначений для проєктування та аналізу баз даних та інформаційних систем і містить продукти:

All Fusion Process Modeler (BPwin);

All Fusion Erwin Data Modeler (ERwin);

All Fusion Data Model Validator (інструмент для перевірки структури баз даних і створених в ERrwin моделей);

All Fusion Model Manager (середовище для роботи групи проєктувальників на ERrwin і BPwin);

All Fusion Component Modeler (моделювання компонентів програмного забезпечення та генерації об’єктного коду застосунків на основі створених моделей).

BPwin дозволяє моделювати будь-які бізнес-процеси та проєктувати організаційну структуру з мінімізацією витрат і підвищенням загальної ефективності роботи організації. Крім того, BPwin дозволяє значно полегшити сертифікацію на відповідність міжнародному стандарту системи менеджменту якості ISO9000. ERwin надає можливість створювати багатокористувацькі інформаційні моделі, автоматизувати процеси збору і перевірки даних. ERwin також дозволяє моделювати практично будь-які бізнес-процеси підприємства для підвищення ефективності роботи організації та зниження витрат. У лінійку продуктів ERwin входять CASE-засоби проєктування, супроводу та документування

баз даних, функціонального моделювання бізнес-процесів і перевірки моделей даних.

Правильний вибір CASE-засобів під час автоматизації процесів проєктування дозволяє провести оптимізацію інформаційних систем, значно підвищити їх ефективність, знизити ймовірність помилок, а також скоротити проєктні витрати.

Проєктування програмного забезпечення за допомогою CASE-систем передбачає:

- попереднє вивчення проблеми, результатом чого є чітка постановка задачі і, як наслідок, розробка технічного завдання;
- деталізацію обмежень і функцій програмної системи, результатом чог є розробка функціонально-логічної моделі складного об’єкта.
- фізичне моделювання (наприклад, певна модульна структура програми), інфологічне проєктування БД, деталізація програмної складової, її складових тощо.

Результат порівняння життєвого циклу програмного забезпечення за традиційної розробки і розробки з використанням CASE-засобів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння життєвого циклу програмного забезпечення за традиційної розробки і розробки з використанням CASE-засобів

Традиційна технологія розробки	**Розробка за допомогою CASE-технології**
Основні зусилля – на кодування і тестування	Основні зусилля – на аналіз і проєктування
«Паперові» специфікації	Швидке ітеративне макетування
Ручне кодування	Автоматична генерація машинного коду
Тестування ПЗ	Автоматичний контроль проєкту
Супровід програмного коду	Супровід проєкту

Розрізняють такі моделі життєвого циклу програмного забезпечення:

– каскадна модель;

– спіральна модель.

Процес розробки програмного забезпечення за каскадною схемою наведено на рисунку 1.1.

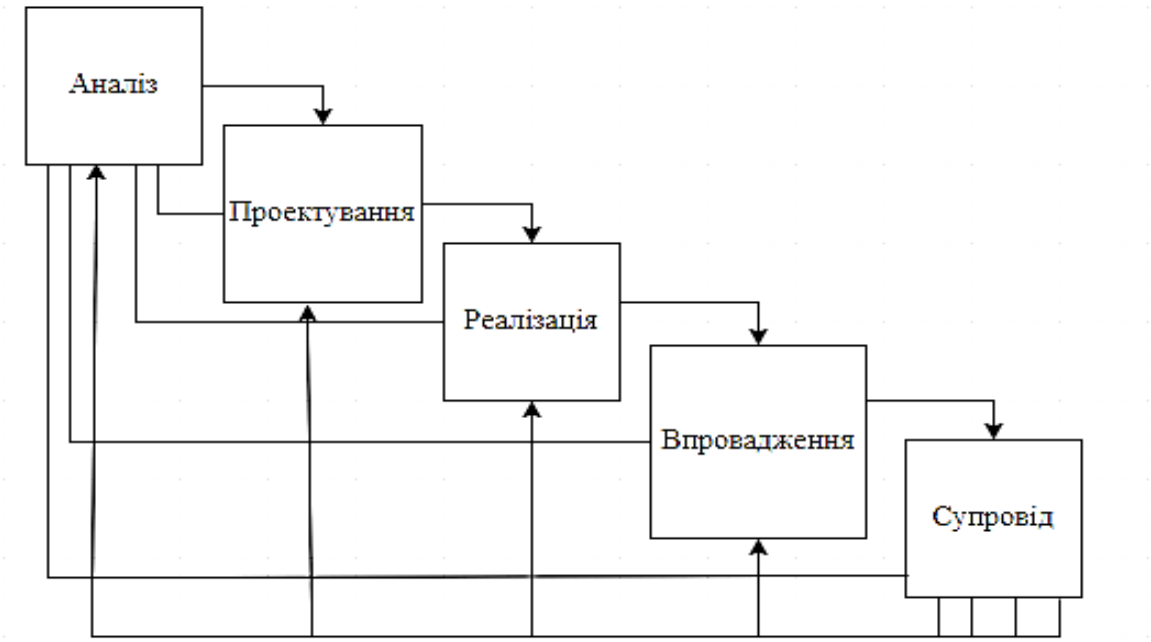


Рисунок 1.1 – Реальний процес розробки за каскадною схемою

Процес розробки програмного забезпечення за спіральною схемою наведено на рисунку 1.2.

Найважливішими (базовими) принципами реалізації CASE-підходу є:

– поділ (декомпозиція);

– ієрархічне впорядкування складових складного об’єкта;

– абстрагування від несуттєвих деталей (з їх «приховуванням»);

– принцип формалізації;

– принцип концептуальної спільності (структурний аналіз – структурне програмування – структурне тестування);

– принцип несуперечності – обґрунтування доцільності і узгодженість функціонування складових;
– принцип логічної і фізичної незалежності даних.

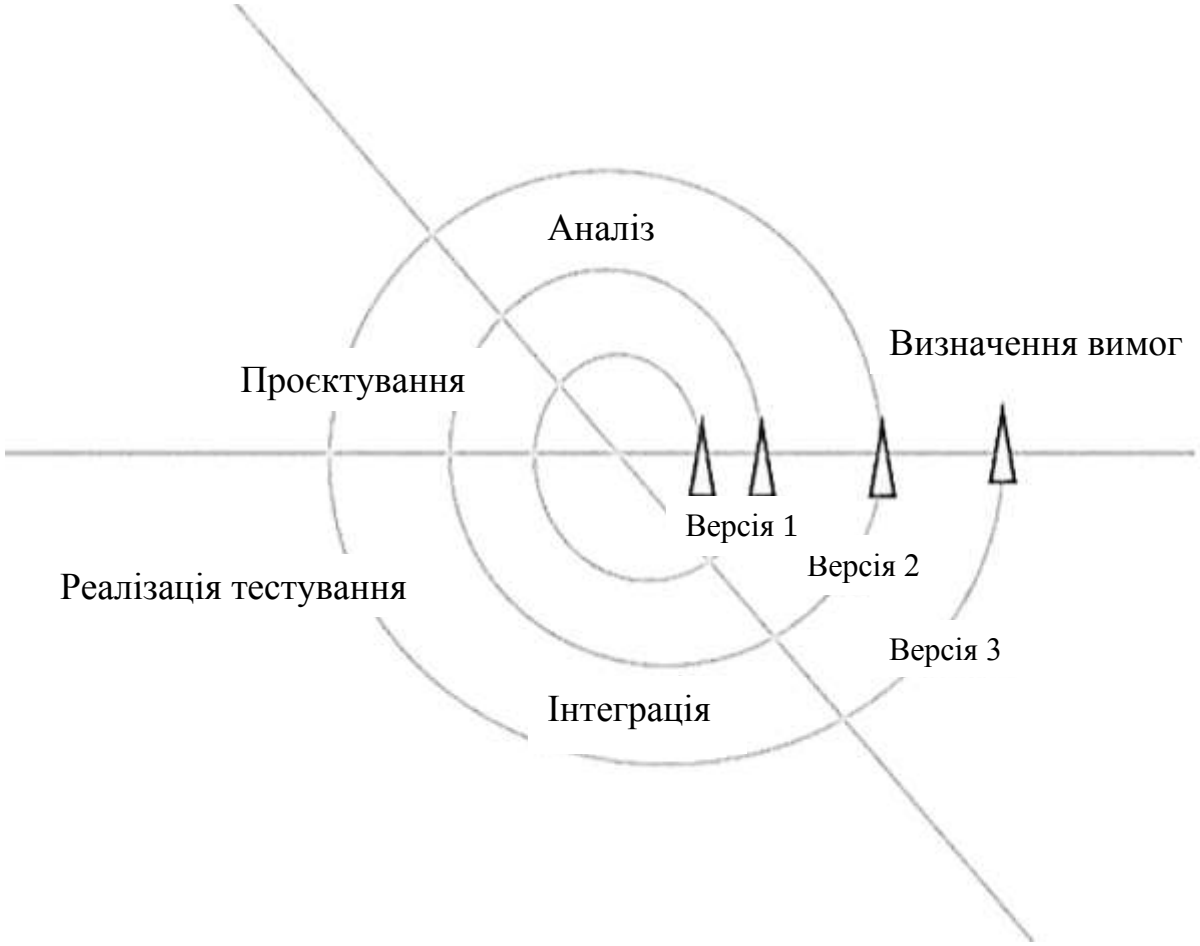


Рисунок 1.2 – Спіральна модель життєвого циклу програмного забезпечення

Застосування CASE-підходу до процесу проєктування базується на таких методологіях:

- SADT (StructuredAnalysisTechnique) – методологія структурного аналізу і проєктування;
- DEF0 (IntegratedDefinitionFunctionModeling) – методологія функціонального моделювання для створення функціональної моделі, що відображає структуру і функції системи;

- UDEF1– методологія для побудови інформаційної моделі щодо структури і змісту інформаційних потоків складного об’єкта;
- UDEF2 – методологія побудови динамічної моделі поведінки складових з урахуванням змін у часі;
- UDEF3 – методологія опису сценаріїв та послідовності операцій під час виконання функцій кожної скадової;
- UDEF4 – об’єктно-орієнтована методологія, відображає взаємодію об’єктів;
- UDEF5 – методологія онтологічного дослідження складного об’єкта, що дає можливості користування словником термінів і правил для подання станів складного об’єкта в певний момент часу;
- DFD (DataFlowDiagram) – методологія моделювання потоків даних для опису процесів обміну даними;
- UML – (UnifiedModelingLanguage) мова візуального моделювання, основана на об’єктно-орієнтованому підході, що дозволяє подати результати проєктування складного об’єкта на кожному рівні.

Крім CASE-технології, в процесі розробки складних об’єктів використовують такі технології.

CAD-технологія (ComputerAidedDesign) – технологія автоматизованого проєктування як система програмних засобів, що реалізовує проєктування, за якого всі проєктні рішення або їх переважну частину отримують внаслідок обрахунків і складання математичних моделей на ЕОМ. Наприклад, САПР друкованих плат (ECAD – ElectronicCAD/EDA – ElectronicDesignAutomation).

CAE-технологія (Computer-aidedengineering) – технологія автоматизованої розробки. До основних функцій цієї технології відносять аналітичні дослідження, розрахунок станів, а також дослідження перехідних процесів на макрорівні, імітаційне моделювання

функціонування складних об’єктів на основі моделей масового обслуговування та мереж Петрі.

CAM-технологія (Computer-aided manufacturing) – технологія автоматизованого виробництва. До основних функцій цієї технології відносять розробку технологічних процесів; розробку програм управління для технологічного обладнання з числовим програмним управлінням (ЧПУ); моделювання процесів обробки в процесі обробки; генерація постпроцесорів для конкретних типів обладнання з ЧПУ (NC – NumericalControl); розрахунок норм часу обробки тощо [2].

CALS-технологія (Computer Acquisition and LifeCycle Support) як безперервна інформаційна підтримка життєвого циклу продукту – технологія комплексної комп'ютеризації сфер промислового виробництва, мета якої – уніфікація і стандартизація специфікацій промислової продукції на всіх етапах її життєвого циклу. У CALS-системах передбачено зберігання, обробку і передачу інформації в комп'ютерному середовищі, оперативний доступ до даних в потрібний час і в потрібному місці. В межах цієї технології характерним є вирішення завдань інтеграції всіх процесів під час життєвого циклу, визначення основним середовищем передачі даних глобальної мережі Internet, незалежність розташування в мережі вузлів інформаційної взаємодії, незалежність від типу інформації (маркетингові, конструкторсько-технологічні, виробничі дані, комерційна і юридична інформація і т. д.). Предметом CALS є технології інформаційної інтеграції, а метою застосування – підвищення ефективності діяльності всіх учасників створення, виробництва і користування складним об’єктом, що проєктується [2].

Потрібно зазначити, що часткова автоматизація часто не дає очікуваного підвищення ефективності отримання проєктних рішень, а тому кращим є впровадження інтегрованих САПР, що автоматизують основні етапи проєктування складних об’єктів. Подальше підвищення

ефективності виробництва та конкурентоспроможності продукції, що випускається, можливе за рахунок інтеграції систем проєктування, управління та документообігу. Така інтеграція лежить в основі створення комплексних систем автоматизації, в яких крім функцій власне САПР реалізуються засоби для автоматизації функцій управління проєктуванням, документообігу, планування виробництва, обліку і т. п., що можливе у разі детального вивчення самих процесів проєктування, їх суті з метою максимально автоматизувати такі процеси.

1.2 Блочно-ієрархічний підхід до проєктування складних об’єктів

Блочно-ієрархічний підхід (БІП) оснований на структуризації описів об'єкта з розділенням описів на ряд ієрархічних рівнів відповідно до детальності відображення в них властивостей об'єкта і його складових. Кожному ієрархічному рівню проєктування властиві свої форми, документація, матеріали, апаратура, сукупність мов і моделей, методів здобуття описів деякого ієрархічного рівня. Складний об’єкт розглядається як складна система пов'язаних і взаємодіючих між собою складових, яка подається у вигляді блочно-ієрархічної структури з урахуванням рівнів деталізації. Основна перевага БІП – спрощення процесу проєктування. Водночас:

– під час переходу з деякого рівня $К_1$, на якому розглядається складний об’єкт S, на сусідній, нижчий рівень $К_2$ відбувається розподіл складного об’єкта S на складові і розгляд замість складного об’єкта S його окремих складових;
– розгляд кожної із складових на рівні $К_2$ з більшим ступенем деталізації, ніж на рівні $К_1$, приводить до формування завдань проєктування приблизно однакової складності з погляду можливостей сприйняття людиною і можливостей розв’язання за допомогою наявних засобів проєктування;

– використання понять складного об’єкта і його складових на кожному ієрархічному рівні, тобто якщо складовими проєктованого складного об’єкта S вважалися складові S_k, то на сусідньому, нижчому рівні $К_2$ (з вищим ступенем деталізації) ті самі складові S_k розглядаються вже як складні об’єкти.

На рисунку 1.3 наведено блочно-ієрархічну структуру об’єкта проєктування з урахуванням рівнів деталізації.

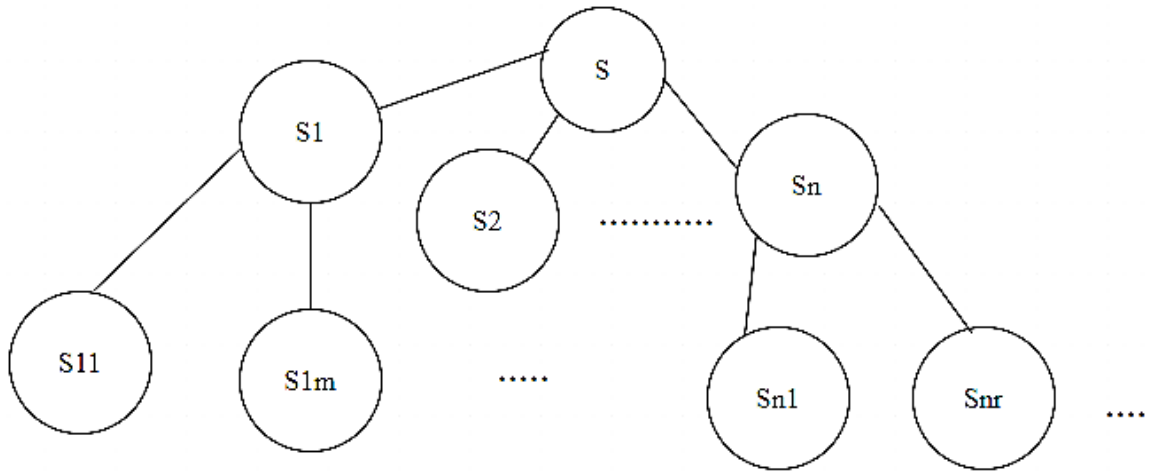


Рисунок 1.3 – Блочно-ієрархічна структура об’єкта проєктування з урахуванням рівней деталізації

Концепція БІП містить:

– розбиття складного об’єкта проєктування, що зводиться до проєктування простіших складових з врахуванням взаємодії між ними;
– локальну оптимізацію, що передбачає поліпшення параметрів / характеристик усередині кожної із складових;
– абстрагування як побудова моделей, що відображають лише важливі в заданих умовах параметри/характеристики об'єктів;
– повторюваність, що залежить від досвіду проєктування.

Розглянемо подання складного об’єкта за блочно-ієрархічним підходом. Нехай на складний об’єкт покладається задача моніторингу

валютних пар. В такому випадку відповідна система з урахуванням блочно-ієрархічного підходу може мати вигляд, поданий на рисунку 1.4.

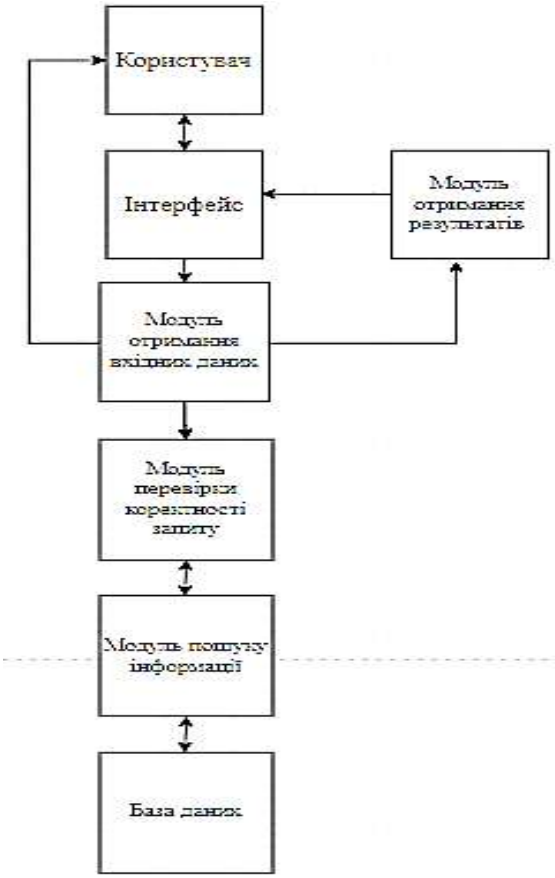


Рисунок 1.4 – Система моніторингу валютних пар з урахуванням блочно-ієрархічного підходу

У цьому випадку функції розподіляться між модулями системи таким чином. До функцій користувача віднесемо створення запиту. Інтерфейс відповідатиме за введення вхідних даних, виведення вихідних даних, пошук інформації. Функції модуля отримання вхідних даних зводяться до отримання даних від попереднього модуля, а також передачі даних до модуля перевірки коректності запиту. До функцій бази даних віднесемо збереження інформації, оновлення інформації, забезпечення доступу до інформації. Модуль отримання результатів отримує інформацію від модуля перевірки коректності запиту, а також виконує передачу даних до інтерфейсу.

Як інший приклад розглянемо задачу проєктування інформаційної системи «Путівник-порадник абітурієнта». Відповідна система, з урахуванням блочно-ієрархічного підходу, матиме вигляд, поданий на рисунку 1.5. Серед функцій модуля перевірки коректності запиту – забезпечення пошуку інформації; отримання відповіді від модуля пошуку інформації. У разі помилки під час перевірки переходить до модуля «інтерфейс», у разі коректної відповіді переходить до модуля виведення результатів. Функціями модуля пошуку інформації є забезпечення з'єднання з базою даних, пошук інформації за запитом. Після отримання відповіді від бази даних система виводить результат.

Функції інформаційної системи розподілимо між модулями системи таким чином. До функцій інтерфейсу, вхідними даними якого є дані для ідентифікації користувача та введена спеціальність, віднесемо авторизацію та аутентифікацію користувача, введення / виведення даних, підтвердження введених даних. Вихідними даними цього модуля можуть виступати дані щодо знайдених університетів (номери телефонів; документи, необхідні для вступу; вартість навчання; загальні відомості про університет).

Модуль зчитування даних оперуватиме вхідними даними, введеними користувачем та виконуватиме функції отримання даних з модуля «Інтерфейс», збереження даних від користувача у Базі Даних, а також обробки отриманих даних.

База Даних матиме вхідними даними інформацію з модуля зчитування даних та виконуватиме функції збереження даних, оновлення даних, видалення даних, пошуку інформації. Модуль аналізу знайдених даних виконуватиме функції отримання даних з Бази Даних; обробки знайдених даних; передачі даних в модуль структурування даних. До функцій модуля структурування проаналізованих даних входить отримання даних від

модуля знайдених даних, а також формування структурованих даних за запитом абітурієнта.

Рисунок 1.5 – Інформаційна система «Путівник-порадник абітурієнта» з урахуванням блочно-ієрархічного підходу

Саме такий підхід проєктування складних об’єктів у сучасних технологіях комп’ютерного проєктувння реалізовано з використанням нотацій IDEF.

Кейс 1. Запропонуйте структуру складного об'єкта з урахуванням зв'язків між його складовими (до 6 складових). Поясніть функціональне навантаження кожної складової. Відповідь обгрунтуйте.

1.3 Нотації IDEF

Розглянемо основні нотації IDEF(Icam DEFinition) для подання результатів проєктування складних об’єктів, що використовуються в сучасних технологіях комп’ютерного проєктування.

Нотація IDEF0 виникла під час формалізації процесу створення складного об’єкта як такого, що містить етапи:

- аналіз;
- синтез структури складного об’єкта та взаємодії його складових;
- синтез складових складного об’єкта;
- трасування складових складного об’єкта в єдине ціле;
- тестування функціонування складного об’єкта;
- запуск складного об’єкта;
- використання складного об’єкта.

Основу IDEF0 становить методологія Дугласа Т. Росса і мова опису систем SADT (structured analysis and design technique), за якою все, що відбувається в складному об’єкті та його складових, називають функціями. Кожній функції ставиться у відповідність складова (модуль). На IDEF0−діаграмі складова позначається прямокутником. Інтерфейси, за допомогою яких складова взаємодіє з іншими складовими або із зовнішніми об’єктами, зображаються стрілками, що входять в складову або виходять з неї. Вхідні стрілки показують, які умови мають бути одночасно виконані, щоб функція, описувана блоком, була виконана. У IDEF0-моделях використовується як природна, так і графічна мови.

Компонентами синтаксису IDEF0 є діаграми, складові, стрілки і правила.

IDEF0 висуває вимоги, щоб у діаграмі було не менш трьох і не більше шести складових. Ці обмеження підтримують складність діаграм і моделі на рівні, доступному для читання, розуміння і використання [3].

Складові – це функції, які визначаються як діяльність, процес, операція, дія або перетворення. Складові на діаграмах зображаються прямокутниками. Складова подає функцію назвою з використанням дієслів.

Стрілки є даними, пов’язанимиі з функціями.

Правила визначають, як потрібно застосовувати компоненти; діаграми забезпечують формат графічного та словесного опису моделей.

В IDEF0 кожна сторона складової має особливе, цілком певне призначення. Ліва сторона блока призначена для входів, верхня – для управління, права – для виходів, нижня – для функцій.

Кожна складова може бути декомпозованою на більш детальній діаграмі. Входи і виходи визначають інтерфейси між складовими, проте, за необхідності, їх можна об’єднувати.

Складові в IDEF0 мають бути пронумеровані. Номери складових слугують однозначними ідентифікаторами для системних функцій і автоматично організують ці функції в ієрархію моделі. Використовуючи номери складових і оцінюючи, як одна складова впливає на іншу, можна організувати модель за принципом функціонального домінування. Це дозволяє узгодити ієрархічний порядок функцій в моделі з рівнем впливу кожної функції на іншу частину складного об’єкта.

Одна IDEF0-діаграма складна сама по собі, оскільки вона містить від трьох до шести складових, пов’язаних множиною стрілок. Для опису складного об’єкта потрібно кілька таких діаграм. Діаграми, зібрані і пов’язані разом, є IDEF0-моделлю [3].

У IDEF0 додатково до правил синтаксису діаграм існують правила синтаксису моделей. Синтаксис IDEF0-моделей дозволяє розробнику визначити межі моделі, пов’язати діаграми в одне ціле і забезпечити точне узгодження між діаграмами. IDEF0-модель є ієрархічно організованою сукупністю діаграм. Декомпозиція формує межі, і кожен блок в IDEF0 розглядається як формальна межа частини цілого складного об’єкта. Іншими словами, блок та стрілки, які його стосуються, визначають точну межу діаграми, що є декомпозицією цієї складової. Така діаграма, що називається діаграмою-нащадком, описує всі лише її складові та зв’язки. Складова, що декомпозується, називається батьківською складовою, а діаграма, що її містить, відповідно, батьківською діаграмою. Таким чином, IDEF0-діаграма є декомпозицією деякого складного об’єкта.

Принцип обмеження складного об’єкта зустрічається на кожному рівні. Одна складова і її зв’язки (впливи та реакції) на найвищому (верхньому) ієрархічному рівні використовуються для визначення меж складного об’єкта. Такий опис визначає загальну функцію, виконувану складним об’єктом. Діаграма, що складається з однієї складової та її впливів і реакцій, визначає межу складного об’єкта та називається контекстною діаграмою моделі складного об’єкта. IDEF0-моделі в процесі структурної декомпозиції подають проєктування «згори-донизу». Спочатку декомпозується одна складова, яка є межею моделі, на одній діаграмі, яка має від трьох до шести складових, потім декомпозується одна (або більше) з цих складових на інший діаграмі з трьома-шістьома складовими і т. д. Назва діаграми збігається з назвою складової, що декомпозується. Результатом цього процесу є модель, діаграма верхнього рівня якої описує складний об’єкт в загальних термінах, а діаграми нижнього рівня описують дуже деталізовані аспекти та операції складного об’єкта [3].

Можна виділити такі основні підходи до декомпозиції [3]:

1. Функціональна декомпозиція (декомпозиція базується на функціональних взаємовідношеннях дій складного об’єкта). Перевага надається докладному висвітленню необхідних обмежень на функції складного об’єкта, а не їх послідовність.

2. Декомпозиція відповідно до вже відомих стабільних складових. Це призводить до створення набору моделей, по одній моделі на кожну підсистему або важливу компоненту. Потім для опису всієї системи має бути побудована складна модель, яка об’єднує всі окремі моделі.

3. Декомпозиція, основана на відстеженні «життєвого циклу» складного об’єкта чи його складових.

4. Декомпозиція з фізичного процесу. Результатом такої декомпозиції буде виділення функціональних стадій, етапів завершення або кроків виконання.

Таким чином, кожна діаграма являє собою деяку закінчену частину всієї моделі. У методології IDEF0 ідентифікується кожна діаграма цієї моделі за допомогою того, що називається «номер складової». Номер складової для контекстної діаграми має такий вигляд:

назва моделі або абревіатура/A-0,

де A – (Activity у функціональних діаграмах).

Наприклад, номером складової для контекстної діаграми моделі Інтернет-порталу є ІП/A-0. Номером складової діаграми, що декомпозує контекстну діаграму, є той самий номер вузла, але без дефіса (наприклад, ІП/A0). Всі інші номери вузлів утворюються за допомогою додавання до номера вузла батьківського діаграми номера складової, що декомпозується. Номер вузла на першій діаграмі – ІП/A0, а номер вузла на другий діаграмі – ІП/A1. Діаграма ІП/A1 декомпозує складову 1 діаграми ІП/A0. Перший нуль під час утворення номера вузла прийнято опускати, тому замість ІП/A01 пишеться ІП/A1.

Для ідентифікації версії діаграм використовується С-номери, які слугують для зв’язування діаграм при русі як вгору, так і донизу по ієрархії моделі. Зазвичай С-номер діаграми, що декомпозує певну складову, вперше з’являється безпосередньо під цією складовою на батьківський діаграмі. Це утворює «спрямований донизу» зв'язок від батьківської діаграми до діаграми-нащадка.

Як тільки утворюється спрямований донизу зв'язок, на діаграмі-нащадку формується посилання на батьківську діаграму. В області контексту IDEF0-бланка (правий верхній кут) зображається кожна складова батьківської діаграми маленькими квадратиками, заштриховується квадратик декомпозовного блока і розміщується С-номер батьківської діаграми біля заштрихованого квадратика. Це утворює «спрямований догори» (до батьківської діаграми) зв'язок. Метод з’єднання

діаграм за допомогою однозначно визначених номерів гарантує, що саме потрібна версія діаграми стане частиною моделі. Іншими словами, у разі використання С-номерів здійснюється ретельний контроль за введенням нових діаграм в ієрархію моделі.

Початок моделювання в IDEF0 [3, 4] означає створення діаграм А-О та АО, які потім можуть рецензуватися. Ці дві діаграми повністю розповідають все про досліджувану систему з мінімальним ступенем деталізації. Створюючи їх, розробник робить початкову спробу декомпозувати систему і потім узагальнювати отриману декомпозицію. Декомпозиція (діаграма АТ) висвітлює найбільш важливі функції і складові складного об’єкта. Об’єднання (діаграма А-О) трактує складний об’єкт як «чорний ящик», дає йому назву і визначає найбільш важливі впливи та реакції.

Розглянемо деякі рекомендації з використання IDEF0 моделі.

Перш ніж почати моделювання, розробник проводить підготовку до нього, збирає інформацію, декомпозує складний об’єкт і узагальнює цю декомпозицію. Підготовка містить вибір мети моделі, вибір погляду, з якого буде представлено модель, тип створюваної моделі і передбачуване використання побудованої і перевіреної моделі. Декомпозуючи складний об’єкт, необхідно, насамперед, звернути увагу на вхідні та вихідні дані. Декомпозиція всього складного об’єкта починається зі складання списку основних типів та основних функцій. Потім ці списки забезпечуються коментарями для ідентифікації основних типів, як даних, так і функцій складного об’єкта або їх різних поєднань. Нарешті, списки з коментарями використовуються для створення діаграми А0, яка потім узагальнюється за допомогою діаграми А-0.

IDEF0-діаграми подають межі функцій і обмеження, що накладаються на них, причому обмеження мають бути присутніми у всіх системах. Без обмежень функціональна IDEF0-діаграма являє собою схему потоків даних.

Після складання списку даних складається список функцій. У такому випадку кілька різних типів даних можуть використовуватися однією функцією. Для кожної конкретної функції визначається її відношення до груп даних. Доцільно поєднувати функції в «агрегатні». Це дозволить привести модель до 3–6 функціональних угруповань. Потрібно, щоб ці угруповання мали один і той самий рівень складності, містили приблизно однаковий обсяг функціональності і функції, в кожному з них мають бути подібні операції і цілі.

Початковий зміст діаграми А0 забезпечують списки даних і функцій. Для правильного опису складного об’єкта змісту потрібно надати форму. У IDEF0 це робиться за допомогою побудови діаграми.

Доцільно дотримуватися такого порядку:

- розташувати блоки на сторінці;
- нарисувати основні стрілки, що подають обмеження;
- нарисувати зовнішні стрілки;
- нарисувати всі стрілки, що залишилися.

Правильне розташування блоків є найважливішим етапом побудови діаграми. Складові розташовуються відповідно до їх домінуванням (за ступенем важливості або за порядком проходження). Найбільш домінантна складова зазвичай розташовується у верхньому лівому куті, а найменш домінантна – в нижньому правому.

Також наводяться обмеження як основні стрілки. Це є другою важливою частиною побудови діаграми А-0. В подальшому об’єкт діаграми декомпозується на 3–6 системних функцій, що зображаються складовими.

Основними стрілками, що подають обмеження, завжди є зовнішні стрілки, тобто стрілки, що подають дані, які надходять з безпосереднього оточення діаграми. Наступним кроком у побудові діаграми є розміщення інших зовнішніх стрілок.

Узагальнення є останнім важливим кроком початкового етапу моделювання. Для будь-якої IDEF0-діаграми створюється батьківська діаграма, що містить її контекст, де під контекстом розуміється складова з набором впливів та реакцій. Діаграма А-0 має кілька призначень. По-перше, вона оголошує загальну функцію всієї системи. По-друге, вона дає множину основних типів або наборів даних, які використовує або формує складний об’єкт. По-третє, А-0 діаграма відображає відношення між основними типами даних.

Побудова діаграми А-0 завершує початковий етап моделювання. Подання загального вигляду системи, отриманого за допомогою діаграм А−0 і А0 є основноюа метою на початковому етапі побудови IDEF0-моделі.

Продовження моделювання ґрунтується на тих самих методах і виводить модель на наступний рівень деталізації. Для цього потрібно створити окрему діаграму для кожного блока діаграми верхнього рівня, потім побудувати діаграми для всіх блоків нових діаграм, і так до тих пір, доки модель не буде описувати об’єкт з потрібним для досягнення мети ступенем деталізації. Таким чином, продовження моделювання є рекурсивним процесом.

Для створення моделей даних можна використовувати дві нотації:

- методології проєктування реляційних баз даних IDEF1X, як одного з підходів до семантичного моделювання даних, основано на концепції «суть-зв’язок» з основним призначенням – побудова концептуальної схеми реляційної бази даних, яка була б незалежною від програмної платформи та її кінцевої реалізації;
- IE (Information Engineering).

Процес побудови інформаційної моделі на основі нотації IDEF1X [4] складається з таких основних кроків:

- визначення сутностей;

- визначення залежностей між сутностями;
- задання первинних та альтернативних ключів;
- визначення атрибутів сутностей;
- приведення моделі до необхідного рівня нормальної форми;
- перехід до фізичного опису моделі.

Нині модель Чена «сутність – зв’язок» (EntityRelationship), стала фактичним стандартом під час моделювання баз даних. Загальноприйнятою стала її скорочена назва – ER-модель предметної області, а більшість сучасних CASE-засобів містять інструментальні засоби для опису даних саме у форматі цієї моделі.

Всі CASE-системи мають розвинені засоби документування процесу розробки БД, автоматичні генератори звітів дозволяють підготувати звіт про поточний стан проєкту БД з докладним описом об’єктів БД та їх відношень як в графічному вигляді, так і у вигляді готових стандартних друкованих звітів, що істотно полегшує ведення проєкту.

На сьогодні не існує єдиної загальноприйнятої системи позначень для ER-моделі і різні CASE-системи використовують різні графічні нотації. Метод IDEF1, розроблений Т. Ремей (T. Ramey), також оснований на підході П. Чена і дозволяє побудувати модель даних, еквівалентну реляційній моделі в третій нормальній формі. Нині на основі вдосконалення методології IDEF1 створено її нову версію – методологію IDEF1X.

IDEF1X дозволяє на основі простих графічних зображень моделювати інформаційні взаємозв’язки та відмінності між реальними об’єктами, фізичними та абстрактними залежностями, що існують серед реальних об’єктів, та інформацією, що відноситься до реальних об’єктів.

IDEF1X використовує основні поняття реляційних баз даних: сутність, атрибут, зв'язок та ключ. Крім того, IDEF1X додатково оперує

низкою понять, правил та обмежень, такими як домени, views, первинні, зовнішні та сурогатні ключі тощо.

Стандарт та методологія IDEF1X є спеціалізованим інструментом, призначеним для розробників реляційних баз даних.

Кейс 2. Запропонуйте структуру складного об'єкта з урахуванням зв'язків між його складовими (до 6 складових) за нотацією IDEF0. Поясніть функціональне навантаження кожної складової. Відповідь обгрунтуйте.

1.4 Характеристики об’єктів проєктування

Під проєктуванням технічних систем розуміють комплекс робіт з дослідження, розрахунків та конструювання, які мають за мету отримання технічної документації, необхідної для створення нового об’єкта, його реконструкції, модернізації, або реалізації нових процесів, що задовольняють задані вимоги.

Проєктування – це процес складання опису, необхідного для створення об’єкта, що ще не існує, який здійснюється перетворенням первинного опису (технічного завдання), оптимізацією заданих характеристик об’єкта та алгоритму його функціонування, усуненням протиріч первинного опису та послідовним поданням деталізованих описів об’єкта різними мовами для різних етапів проєктування [5].

Технологія (teche + logos, тобто «майстерність+навчання») комп’ютерного проєктування (Computer-AidedDesign) полягає у використанні комп'ютерних систем для полегшення створення, зміни, аналізу та оптимізації проєктів як сукупність методів, виробничих процесів та програмно-технічних засобів, об’єднаних у технологічний ланцюжок, що забезпечує виконання процесів проєктування з метою підвищення їх надійності та оперативності за зменшення трудомісткості використання інформаційного ресурсу [5].

Серед загальних підходів комп'ютерного проєктування потрібно відзначити такі:

– відсутність легких принципів проєктування складних обєктів і систем;
– прагнення одночасно підвищити точність, зменшити вартість, підвищити надійність, підвищити швидкодію, розширити функціональність;
– основний принцип проєктування – навчання, наслідок – досвід проєктування;
– блочно-ієрархічний підхід до проєктування об’єктів.

Загалом, технологія проєктування складного об’єкта містить:

1) покрокову процедуру проєктування, що визначає послідовність технологічних операцій проєктування;
2) критерії і правила, що використовуються для оцінення результатів виконання проєктування складного об’єкта на кожному рівні проєктування;
3) визначення нотацій для опису проєктованого складного об’єкта.

Визначимось з поняттям «об’єкт проєктування» (ОП).

Під об’єктом проєктування будемо розуміти середовище або процес, в контексті яких знаходиться предмет. Предмет проєктування – це продукт, образ якого подано в проєкті. Прикладами ОП може бути процес розпізнавання образів, процес моніторингу прогресу навчання, інтелектуальна система оцінювання ситуації або системи певного призначення.

Усі параметри будь-якого об’єкта проєктування можна розділити на:

- вхідні параметри / характеристики (параметри / характеристики впливу);
- вихідні параметри / характеристики (реакції).

Отже, вхідними параметрами об’єкта проєктування будемо називати такі параметри / характеристики, за допомогою яких можна змінити стан

об'єкта [5, 6]. Залежно від складності об’єкта проєктування кількість вхідних параметрів / характеристик може бути різна. В складних об’єктах кількість може доходити до тисяч, навіть десятків тисяч. Будемо відображати, в математичній формі, всі вхідні параметри / характеристики відповідним вектором $\overline{X_{вх.}}$, а за необхідності виділення конкретного *j*-го вхідного параметра $x_{вх.j}$. В загальному випадку

$$\overline{X_{вх.}} = (x_{вх.1}, x_{вх.2}, ..., x_{вх.n}),$$

де n – загальна кількість вхідних параметрів/характеристик.

Серед важливих характеристик об’єктів проєктування потрібно відзначити точність, час (швидкодію), надійність, вартість.

Точність характеризується похибкою. Залежно від розмірності значення похибки поділяють на абсолютні та відносні. Під абсолютною похибкою Δ розуміють різницю між реальним значенням вихідного сигналу Y та його можливим ідеальним значенням Y_0:

$$\mathbf{\Delta = Y - Y_0}. \qquad (1.1)$$

Відносна похибка не має розмірності і визначається як частка від ділення абсолютної похибки на реальне значення вихідного сигналу:

$$\mathbf{\delta = \Delta/Y = (Y - Y_0)/Y}. \qquad (1.2)$$

Іноді відносну похибку подають у відсотках. Загалом точність має особливо важливе значення для аналогових або аналого-цифрових пристроїв.

Часові характеристики визначають динамічні властивості пристрою, тобто швидкістю реалізації ним необхідних функцій. Наприклад, тривалість функціонування $T_ф$ як затримка між моментом подачі вхідного сигналу (впливу) та появою вихідного сигналу (реакції); період

функціонування $T_ц$ як інтервал часу між двома послідовними перетвореннями вхідного сигналу. Для модема швидкодія буде визначатися його часовою діаграмою, часом встановлення та поновлення зв'язку, швидкістю передачі, протоколами зв'язку тощо.

Надійність визначає здатність пристрою виконувати покладені на нього функції, зберігаючи свої експлуатаційні характеристики в заданих межах протягом певного проміжку часу. Кількісною характеристикою надійності є ймовірність безвідмовної роботи. На практиці звичайно застосовується середній час безвідмовної роботи, тобто середній час функціонування аналого-цифрового пристрою до появи першої відмови.

Вартість визначає в узагальненому вигляді всі витрати, пов'язані з практичною реалізацією пристрою. Сюди відносять собівартість пристрою, а також масу та габаритні розміри.

Всі наведені характеристики описують окремі аспекти функціонування систем або пристроїв. Тому робилися спроби отримати узагальнені характеристики. Проте на практиці такі критерії звичайно не застосовують, а задаються якоюсь однією характеристикою (як правило, собівартістю С) та її оптимізують і дивляться, щоб інші характеристики знаходились в певних межах.

Проєктування, що здіснюється людиною у взаємодії з ЕОМ, називають комп'ютерним. Ступінь автоматизації може бути різною і оцінюється часткою δ проєктних робіт, які виконує ЕОМ без участі людини. Якщо δ=0 проєктування називають ручним або некомп'ютерним, якщо δ=1 – автоматичним. Найчастіше ступінь автоматизації коливається в межах 0,5 – 0.7.

Кейс 3. Наведіть приклади ОП та його вхідні й вихідні параметри/характеристики.

1.5 Етапи та методи проєктування складних об’єктів

Процес проєктування складних систем звичайно складається з таких етапів.

Етап науково-дослідних робіт (НДР) – етап попереднього проєктування. Як правило, новий складний об’єкт певного призначення за характеристиками має бути удосконаленим з погляду виконання покладених на нього функцій. Для досягнення поставленої мети необхідні результати досліджень щодо сучасних наявних рішень, їх ґрунтовний порівняльний аналіз, а також пошук нових підходів до розв’язання проблем, побудови структур та технічних засобів тощо. Результатом виконання цього етапу є формування мети та постановка задачі проєктування.

Етап ескізного проєктування або дослідно-конструкторських робіт (ДКР) – етап створення ескізного проєкту, в якому відображено результати детального опрацювання можливостей побудови складного об’єкта.

Етап технічного (робочого) проєктування – етап детального опрацювання всіх схемних, конструкторських та технологічних рішень, що фіксуються в технічному проєкті складного об’єкта.

Крім названих етапів під час проєктування існують і інші, більш деталізовані, необхідність в яких визначається процесом розробки та особливостями подальшого використання.

На будь-якій стадії або етапі проєктування можуть виявитися помилки або неоптимальність прийнятих рішень, і, таким чином, необхідність або доцільність їх перегляду. Такі ситуації є характерними для проєктування та зумовлюють його ітераційний характер. Може бути також виявлена необхідність корегування технічного завдання.

В цьому випадку відбувається чергування процедур зовнішнього та внутрішнього проєктування, що особливо характерно для ранніх стадій (НДР, ДКР). У цьому випадку до зовнішніх процедур проєктування

відносять процедури формування або корегування технічного завдання, а до внутрішніх – процедури реалізації сформованого завдання.

З урахуванням блочно-ієрархічного підходу до проєктування, залежно від порядку, в якому виконуються етапи проєктування, розрізняють методи висхідного та низхідного проєктування.

Метод висхідного проєктування (проєктування «знизу догори») характеризується розв’язанням задач проєктування складових більш низьких ієрархічних рівнів (вищий рівень деталізації) перед розв’язанням задач проєктування складових більш високих рівнів (нижчий рівень деталізації). Відбувається «збирання» результатів проєктування простіших складових за ієрархічним принципом (по рівнях) до функціонально повного об’єкта проєктування.

Суть методу висхідного проєктування: розв’язування задач проєктування складових вищих ієрархічних рівнів базується на розв’язуванні задач проєктування складових нижчих рівнів.

Метод низхідного проєктування, як проєктування «зверху донизу», починається з верхнього рівня, на якому складний об’єкт розглядається як ціле, а потім виконуються етапи проєктування складових першого рівня, другого і т. д. На кожному рівні встановлюється структура і взаємозв'язок складових, визначаються значення їх характеристик.

У цьому випадку знайдені значення характеристик мають розглядатися як технічне завдання для проєктування на наступному, більш деталізованому нижчому рівні.

Суть методу низхідного проєктування: розв’язування задач проєктування складових вищих ієрархічних рівнів передує розвязування задач проєктування складових нижчих рівнів.

Під час проєктування складних об’єктів і систем як hard, так і soft-засобів застосовують методи низхідного і висхідного проєктування.

1.6 Рівні проєктування

Практика проєктування електронної та обчислювальної техніки привела до виділення горизонтальних та вертикальних рівнів проєктування, наведених в таблиці 1.2 [5, 6].

Горизонтальні рівні виділяють згідно з врахуванням характеру властивостей об’єкта. Основними горизонтальними рівнями є: функціональний, алгоритмічний та конструкторський.

Таблиця 1.2 – Горизонтальні рівні

Функціональний	**Алгоритмічний**	**Конструкторський**
Структурний (системний)	Проєктування схем узагальнених алгоритмів функціонування складного об’єкта	Фізична реалізація складного об’єкта
Функціонально-логічний	Проєктування схем алгоритмів функціонування окремих складових, проєктування модулів та їх взаємодії	Фізична реалізація окремих складових складного об’єкта
Компонентний	Програмний, мікропрограмний, макроси, модулі	Базові компоненти

Функціональне проєктування пов’язано з розробкою складного об’єкта або його складових (модулів, блоків), шо виконують визначені функції.

Алгоритмічне проєктування пов’язано з розробкою алгоритмів функціонування складного об’єкта або його складових та зі створенням математичного (зокрема програмного) забезпечення.

Конструкторське проєктування охоплює коло питань, пов’язаних з конструкторською реалізацією результатів функціонального проєктування.

Рівні проєктування можна також розділяти згідно зі ступенем деталізації, з яким відображаються властивості об’єкта, що проєктується (так звані вертикальні або ієрархічні рівні).

Відповідно до назви, функціональний рівень проєктування вирішує питання розподілу функцій з урахуванням рівня деталізації складного об’єкта – від загальної задачі, що покладається на складний об’єкт, до окремих функцій, що розв’язуються його відповідними складовими. Необхідно враховувати, що рівень деталізації різних складових може бути різним і залежить від мети проєктування.

На структурному (системному) рівні ведеться узагальнений розгляд об’єкта загалом, розподіл його на окремі складові.

На функціонально-логічному рівні проєктуються окремі складові відповідно до функції, що їм призначена. Водночас досліджується узгодженість у виконанні покладених функцій кожною складовою з урахуванням рівня деталізації, а також розробляються тести для перевірки коректності функціонування складного об’єкта.

На компонентному рівні розробляються окремі компоненти складного об’єкта відповідно до рівня деталізації.

Алгоритмічне проєктування починається з розробки схем алгоритмів функціонування складного об’єкта загалом. Потрібно передбачити особливості виконання покладених на складний об’єкт функцій з урахуванням можливих значень вхідних даних. З підвищенням рівня деталізації складного об’єкта розробляються схеми алгоритмів функціонування його окремих складових, проєктуються відповідні модулі та їх взаємодія.

Конструкторський рівень проєктування складних об’єктів вирішує питання реалізації результатів проєктування на функціональному та алгоритмічному рівнях з урахуванням рівня деталізації.

1.7 Склад СКПР

Сучасні СКПР створюються за такими принципами [5, 6]:

- СКПР – система, в якій взаємодіють людина та машина. Колектив розробників є складовою частиною системи проєктування, виконуючи проєктні роботи в діалозі з системою проєктування;
- комплексна автоматизація всіх рівнів проєктування. Введення комп'ютерного проєктування на деяких рівнях, за умов збереження застарілих ручних форм проєктування на інших рівнях, менш ефективне, ніж комплексна автоматизація всього процесу;
- інформаційна узгодженість підсистем та програм проєктування. Програми, які входять в пакет прикладних програм деякої підсистеми СКПР, мають бути інформаційно узгодженими, тобто допускати можливість спільного виконання заданої проєктної процедури без втручання людини в процес переходу від однієї програми до іншої;
- відкритість СКПР. Властивість відкритості означає можливість внесення змін в систему під час її експлуатації;
- поєднання традиційного та комп'ютерного проєктування. Цей принцип має значення в тих випадках, коли комп'ютерне проєктування впроваджується вже на діючому підприємстві з установленою структурою, певними взаємовідносинами між підрозділами тощо.

Складовими структурними частинами СКПР, жорстко пов'язаними з організаційною структурою проєктної організації, є підсистеми, у яких за допомогою спеціалізованих комплексів засобів зважується функціонально закінчена послідовність задач САПР. За призначенням підсистеми поділяють на проєктувальні і обслуговувальні. Проєктувальні підсистеми мають об'єктну орієнтацію і реалізують певний етап (рівень) проєктування або групу безпосередньо пов'язаних проєктних задач. Приклади проєктувальних підсистем: підсистема ескізного проєктування виробів, підсистема проєктування інтерфейсу, підсистема проєктування процесів

обробки даних. Обслуговувальні підсистеми мають загальносистемне застосування і забезпечують підтримку функціонування проєктувальних підсистем, а також оформлення, передачу і виведення отриманих результатів. Приклади обслуговувальних підсистем: автоматизований банк даних, підсистеми документування, підсистема графічного введення-виведення.

Компонентами СКПР є різні види забезпечення – технічне, математичне, програмне, лінгвістичне, інформаційне, методичне та організаційне [6].

Технічне забезпечення – сукупність технічних (апаратних) засобів, що використовуються в СКПР для переробки, зберігання, передачі інформації, організації спілкування людини з ЕОМ, виготовлення проєктної докуметації. Основу технічного забезпечення становлять ЕОМ з різноманітними периферійними пристроями. До технічного забезпечення СКПР відносять також засоби оргтехніки, вимірювальне обладнання для отримання даних, які використовують в процесі проєктування.

Математичне забезпечення – сукупність математичних моделей, методів, алгоритмів для розв’язання задач комп’ютерного проєктування. Математичне забезпечення реалізується в програмному забезпеченні СКПР.

Програмне забезпечення – сукупність програм разом з необхідною документацією, яка призначена для використання в СКПР.

Лінгвістичне забезпечення – сукупність мов, що використовують в СКПР для відображення інформації про об’єкти, які проєктуються, процес та засоби проєктування.

Інформаційне забезпечення – документи, що містять описи стандартних проєктних процедур, типових проєктних рішень, комплектуючих та інші дані, а також файли на магнітних носіях з електронними версіями зазначених документів.

Методичне забезпечення – документи, в яких відображено склад, правила відбору та експлуатації засобів комп’ютерного проєктування. Іноді поняття методичного забезпечення розширяють, відносячи до нього лінгвістичне та математичне забезпечення.

Організаційне забезпечення – положення, інструкції, накази, штатні розклади, кваліфікаційні вимоги та інші документи, які регламентують організаційну структуру підрозділів проєктного підприємства та їх взаємодію з комплексом засобів комп’ютерного проєктування.

2 ТЕХНОЛОГІЇ СТРУКТУРНОГО ПРОЄКТУВАННЯ

2.1 Основні задачі структурного проєктування

Задачі структурного проєктування складних систем поділяються на задачі аналізу та синтезу. Головна мета цього етапу – спроєктувати систему загалом так, щоб вона задовольняла поставлені вимоги, не вдаючись до розгляду деталей.

Проєктні процедури синтезу майже не автоматизовані, їх комп'ютерне розв'язання вдається отримати лише в окремих випадках. Процедури аналізу, як правило, автоматизовані. Однак отримання математичних моделей на структурному рівні потребує від користувача значно більших зусиль та витрат часу, ніж на інших ієрархічних рівнях. Тому процес проєктування переважно відбувається в діалоговому режимі: користувач пропонує можливі варіанти структур, а ЕОМ їх оцінює (виконує верифікацію). Генерацію варіантів можна частково перекласти на ЕОМ на підставі одного з підходів до структурного синтезу [5, 6]. На підставі отриманих результатів користувач вносить зміни у варіанти, приймає рішення, корегує математичні моделі, програмні засоби тощо.

2.2 Формалізація процедур синтезу

Процедури системного синтезу класифікуються за рядом ознак [6].

За цілями синтезу та змістом отримуваних результатів виділяють такі процедури структурного синтезу:

- вибір принципів побудови та функціонування технічних об'єктів;
- вибір технічного рішення;
- синтез технічної документації.

Вибір принципів побудови та функціонування технічних об'єктів здійснюється на стадіях передпроєктних досліджень та науково-дослідних робіт. Його мета – встановлення фізичних, інформаційних, організаційних

принципів тощо. Під час проєктування обчислювальних систем змістом цієї процедури є вибір архітектурних рішень та побудова структурних схем.

Вибір технічного рішення має за мету отримання функціональних рішень складових складного об’єкта відповідно до рівня деталізації, визначення маршрутів інформаційних потоків тощо.

Синтез технічної документації полягає в автоматичному перетворенні даних про складні об’єкти та їх складові, що знаходяться у внутрішньому форматі СКПР, в текстову та креслярську документацію, оформлену згідно з правилами відповідних стандартів щодо конструкторської документації.

За рівнем складності формалізації процедур синтезу виділяють п’ять рівнів.

Задачі першого рівня складності з’являються там, де структура об’єкта, що проєктується, визначена наперед результатами раніше виконаних досліджень і синтез зводиться до вибору числових значень параметрів для заданої структури. Задачі першого рівня – це задачі параметричного синтезу.

Задачі другого рівня складності полягають у виборі структури з кінцевої множини варіантів за таких умов:

1) всі варіанти заздалегідь відомі або їх можна легко отримати;
2) потужність множини варіантів настільки мала, що можливий повний перебір за їх порівняння між собою.

Задачі третього рівня складності також зводяться до вибору варіантів в кінцевій множині, але потужність множини достатньо велика, щоб зробити неможливим повний перебір.

Задачі четвертого рівня складності характеризуються вибором варіанта структури в множинах, потужність яких апріорно невідома та не виключена можливість, що вона необмежена.

Задачі п’ятого рівня складності пов’язані з пошуком рішень, основаних на нових, раніше невідомих або на таких ідеях та принципах, що не використовувалися. В задачах попередніх рівнів існування рішень не викликало сумнівів та необхідно було знайти найкраще або прийнятне рішення. В задачах п’ятого рівня складності отримання рішення рівноцінне отриманню принципово нового типу технічних об’єктів.

За типом структур, що синтезуються, розрізняють процедури одновимірного, схемного та геометричного синтезу.

Одновимірний синтез полягає в розробці складових найнижчого рівня. Приклади таких складових – розробка алгоритмів введення / виведення інформації, алгоритмів ідентифікації користувача.

Схемний синтез пов’язаний з розробкою hard-засобів різного рівня деталізації – розробка функціональних, структурних, принципових схем тощо, які відображають результати проєктування складових складного об’єкта.

Геометричний синтез виконується за конструювання складних об’єктів та їх складових. Він пов’язаний з їх фізичною реалізацією (розташуванням об’єкта або його складових).

На системному рівні найбільш поширені два підходи до структурного синтезу:

1) зведення задачі синтезу до задачі дискретного математичного програмування та її розв’язання методами скороченого перебору;
2) використання експертних систем, які містять знання про відомі структури та елементи систем і способи генерації нових структур.

Багато задач формування оптимальної структури складного об’єкта та визначення його складових зводяться до задачі дискретного математичного програмування:

$$\min F(X),\ X \in X_D, \qquad (2.1)$$

де $\mathbf{X}= (x_1, x_2, \ldots, x_n)$, x_i – кількість складових *i*-го типу в складному об'єкті, що проєктується;

$\mathbf{X}_D$ – область допустимих значень, яка визначається обмеженнями виду

$$\mathbf{y_j (X) \le y_{тj}}, \tag{2.2}$$

де $\mathbf{Y} = (y_1(X), y_2(X), \ldots, y_m(X))$, $y_j(X)$ – *j*-й вихідний параметр складного об'єкта (середній час розв'язання задачі, пропускна здатність, ймовірність відмови, точність тощо);

$\mathbf{Y}_т=(y_{т1}, y_{т2}, \ldots, y_{тm})$ – вектор заданих технічних вимог;

F(**X**) – цільова функція, яка формується згідно з одним із способів постановки екстремальних задач [6].

Задача (2.1) є комбінаторною, оскільки на кожен x_i накладено певні обмеження. В загальному випадку вона відноситься до класу NP-повних задач, а це означає, що із зростанням розмірності x_i кількість варіантів зростає експоненціально. Ефективні наближені методи розв'язання вдається отримати лише для окремих випадків загального формулювання.

В експертних системах знання про структуру систем та мереж найчастіше подаються у вигляді І-АБО графів як різновиду семантичних мереж [7].

Наприклад, вершина І відображає комп'ютерну мережу. Вона може мати радіальну, кільцеву або розподілену структуру. Оскільки тут існує багато альтернатив, це будуть вершини типу АБО і так далі.

Розробка семантичної мережі у вигляді графа І-АБО не викликає принципових труднощів. Складніше формалізувати правила вибору оптимального маршруту в графі згідно з вимогами технічного завдання. Розробка його здійснюється згідно з системними правилами (продукціями) і становить суть структурного синтезу з використанням експертних систем.

2.3 Формалізація процедур аналізу

На системному рівні функціонування систем розглядається з інформаційного погляду за повного абстрагування від фізичної суті процесів, які відбуваються в системі. В СКПР за допомогою математичних методів моделюються процеси введення, обробки та виведення даних. Застосовуються як аналітичні, так і імітаційні моделі. Можливості отримання аналітичних моделей для складних систем обмежені, тому аналіз найчастіше виконується методом імітаційного моделювання (як найбільш універсальним).

На системному рівні найчастіше для аналізу використовують моделі систем масового обслуговування (СМО) та апарат мереж Петрі.

2.3.1 Системи масового обслуговування

Теорія масового обслуговування – розділ прикладної математики, який вивчає явища та процеси, пов’язані із задоволенням певного попиту. СМО складається з статичних об’єктів, які називають обслуговувальними апаратами (ОА), та динамічних об’єктів, що носять назву заявок [8, 9].

Заявки, що надходять на проєктування складових, можуть утворювати черги (в іншому випадку СМО називають недоступними). За структурою СМО можуть бути одно- чи багатоканальними (рис. 2.1), одно- чи багатофазними (рис. 2.2).

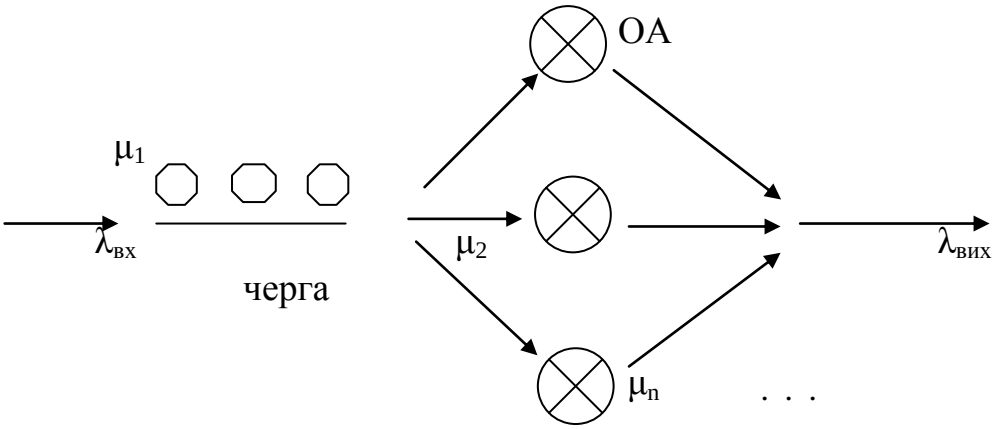


Рисунок 2.1 – Багатоканальна СМО

Під $\lambda_{вх}$ розуміють інтенсивність вхідного потоку заявок на проєктування складових складних об’єктів, тобто кількість заявок, що приходять на СМО за одиницю часу.

Під μ_i розуміють інтенсивність обслуговування заявки на проєктування складових складних об’єктів *i*-м автоматом, тобто кількість заявок, які можна обслужити за одиницю часу.

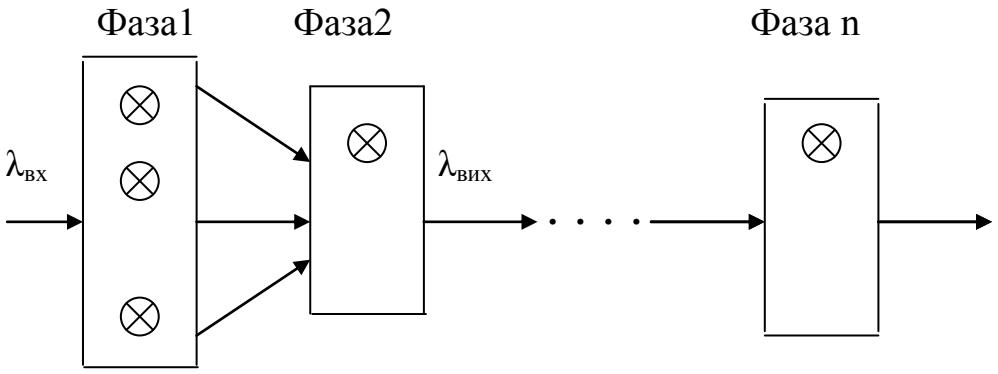


Рисунок 2.2 – Багатофазна СМО

Функціонування СМО виглядає як процес проходження заявок на проєктування складових складних об’єктів через систему. Коли заявка надходить для обслуговування на деякий ОА, змінна *u*, що характеризує стан заявки, набуває значення «обслуговування» (1), а змінна *w*, що характеризує стан ОА – значення «зайнято» (1). Якщо ОА не зайнятий обслуговуванням, то змінна *w* має значення «вільний» (0). Якщо заявка надходить на вхід ОА, зайнятого обслуговуванням іншої заявки, то утворюється черга на вході ОА. Змінна *u*, що характеризує стан заявки, яка знаходиться у черзі, має значення «очікування» (0). Таким чином змінні *u, w* набувають одне з двох можливих значень, тобто є булевими змінними. Стан ОА характеризується також довжиною черги *r* на вході, яка дорівнює кількості заявок.

Правило, згідно з яким заявки на проєктування складових складних об’єктів надходять на обслуговування з черги, називається дисципліною

обслуговування. Величина, яка виражає переважне право на обслуговування, називається пріоритетом.

Якщо всі заявки мають рівні пріоритети, дисципліна обслуговування називається безпріоритетною. Відомі правила FIFO та LIFO відносяться до безпріоритетних дисциплін обслуговування.

Надходження заявок на проєктування складових складних об’єктів може відбуватись через однакові, неоднакові, але певні, та випадкові проміжки часу. Вхідний потік вимог до об’єкта проєктування також може бути однорідним та неоднорідним. Однорідний потік утворюється заявками, на обслуговування яких в середньому витрачається однаковий час. Відповідно, неоднорідний потік утворюється, якщо середній час на проєктування складових складних об’єктів істотно відрізняється.

Таким чином, потік заявок можна розглядати як потік певних подій φ(t) (рис. 2.3):

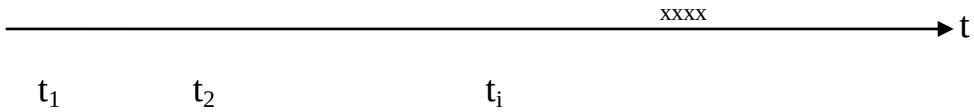


Рисунок 2.3 – Потік заявок

Потік заявок може описуватись різними законами розподілу: рівномірним, нормальним, пуасонівським, ерлангівським.

Простіший потік характеризується трьома властивостями:

- стаціонарність;
- ординарність;
- відсутність післядії.

Вхідний потік заявок називається стаціонарним, якщо ймовірність $p_n(\tau)$ того, що за час τ відбудеться n подій (надійде n заявок на проєктування складових складних об’єктів) , не залежить від розташування

τ на осі часу, а залежить лише від довжини τ та інтенсивності вхідного потоку λ (рис. 2.4).

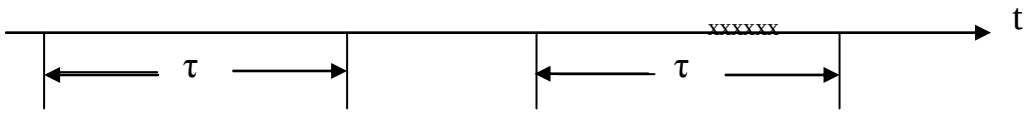


Рисунок 2.4 – Стаціонарний потік заявок

Ординарність означає, що ймовірнсть появи в один і той самий момент часу двох і більше заявок на проєктування складових складних об’єктів є нескінченно малою величиною:

$$\lim_{\Delta t \to 0} p_{>1}(\Delta t)=0 \qquad (2.3)$$

Відсутність післядії полягає в тому, що ймовірність приходу за відрізок часу τ заявок на проєктування складових складних об’єктів у кількості *n* не залежить від того, скільки заявок вже надійшло в СКПР.

За цих умов потік заявок на проєктування складових складних об’єктів описується законом Пуасона:

$$p_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \ \lambda=\text{const}. \qquad (2.4)$$

Аналітичні моделі СМО можливо отримати тільки для систем, що мають вказані властивості. Оскільки більшість систем на практиці не підпадають під визначення «простіших», для їх аналізу використовують методи імітаційного моделювання.

2.3.2 Мережі Петрі

Під час опису складного об’єкта важливо визначити його стани та можливі переходи з одного стану в інший. Ця задача з використанням мереж Петрі може бути означена таким чином. Опис мережі Петрі містить складові [6, 7]

$$\mathbf{C = (P, T, I, O)}, \qquad (2.5)$$

де $P = \{p_1, p_2, \ldots, p_n\}$ – вектор станів складного об’єкта (позицій);

$T = \{t_1, t_2, \ldots, t_n\}$ – кінцева множина умов зміни станів (переходів);

$I(t_j)$ – вхідна функція відображення умов t_j зміни станів в комплекті станів складного об’єкта P;

$O(t_j)$ – вихідна функція відображення умов t_j зміни станів в комплекті станів складного об’єкта P;

p_i є вхідним станом умов t_j зміни станів в комплекті станів складного об’єкта P t_j, якщо $p_i \in I(t_j)$;

p_i є вихідним станом умов t_j зміни станів в комплекті станів складного об’єкта P t_j, якщо $p_i \in O(t_j)$ [15].

Входи і виходи умов зміни станів складного об’єкта P являють собою комплекти його станів. Комплект є узагальненням множини, в яку вносяться елементи, що можуть багаторазово повторюватись.

Структура мережі Петрі – це сукупність станів складного об’єкта та умов зміни станів складного об’єкта. Умова – це предикат або логічний опис стану системи. Умова може набувати значення «істина» або значення «хиба». Відповідно, граф мережі Петрі має два типи вузлів. Вузлом позначається стан складного об’єкта, а планкою – умова зміни станів складного об’єкта. Орієнтовані дуги (стрілки) з’єднують стан складного об’єкта з умовою зміни стану складного об’єкта, водночас деякі дуги спрямовані від стану складного об’єкта до умови зміни стану складного об’єкта, а інші – від умов зміни стану складного об’єкта до стану складного об’єкта.

Приклад графічного подання мережі Петрі як дводольного орієнтованого мультиграфа наведено на рис. 2.5.

Опис мережі Петрі, яка зображена графічно на рис. 2.5, виглядає так:

$C = (P, T, I, O)$;

$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$;

$T = \{t_1, t_2, t_3, t_4, t_5\}$;

$I(t_1) = \{p_1\}$;

$I(t_2) = \{p_3\}$;

$I(t_3) = \{p_2, p_3\}$;

$I(t_4) = \{p_4, p_5, p_5\}$;

$I(t_5) = \{p_2\}$;

$O(t_1) = \{p_2, p_3\}$;

$O(t_2) = \{p_3, p_5, p_5\}$;

$O(t_3) = \{p_2, p_4\}$;

$O(t_4) = \{p_4\}$;

$O(t_5) = \{p_6\}$.

Мережа Петрі є мультиграфом, тому що допускається існування кратних дуг від однієї вершини до іншої. Оскільки дуги є спрямованими, то це орієнтований мультиграф.

За допомогою мереж Петрі моделюються процеси, що подаються у вигляді послідовності станів складного об’єкта. Зміна стану складного об’єкта відбувається, якщо виконано умови зміни стану складного об’єкта. Активність стану складного об’єкта (можливість змінитися за умовою) відображається за допомогою маркерів (фішок), що розташовують всередині вузлів мережі, які відповідають станам складного об’єкта. Активність стану складного об’єкта відображається у векторі станів значенням «1» у відповідній позиції, що була зазначена під час опису мережі Петрі. Значення «0» у позиції вектора станів означає, що стан складного об’єкта на цьому кроці є неактивним, тобто не може бути

зміненим. Так, в мережі Петрі, що наведена на рис. 2.5, початкове маркування у стані (позиції) p_1 виглядатиме так: P = {1, 0, 0, 0, 0, 0}.

Моделювання процесів функціонування складного за допомогою мережі Петрі відбувається шляхом аналізу процесів зміни станів складного об'єкта, що відповідає пересуванню маркерів (фішок) між вершинами-станами складного об'єкта (позиціями).

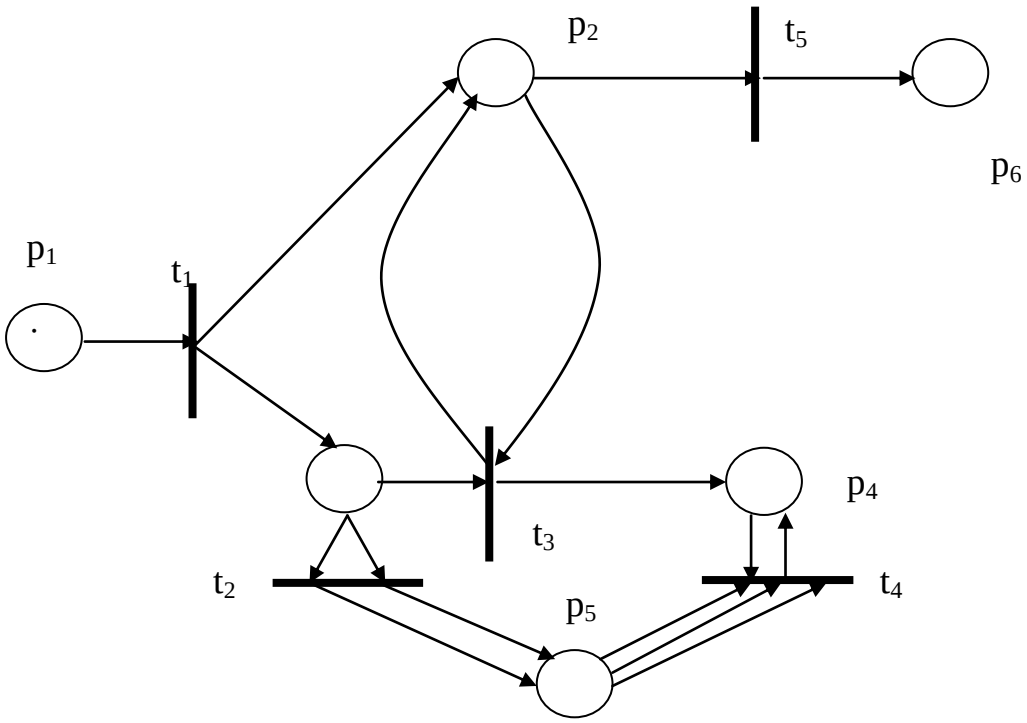


Рисунок 2.5 – Приклад мережі Петрі

Як приклад розглянемо задачу моделювання роботи станка-автомата. Станок-автомат знаходиться в стані очікування до тих пір, поки не з'явиться деталь, яку він має обробити та послати на доставку. Умовами такої системи є:

а) станок-автомат очікує;

б) деталь прийшла та очікує;

в) станок-автомат обробляє деталь;

г) деталь оброблена.

Подіями будуть:

1. Деталь надійшла.
2. Станок-автомат починає обробляти деталь.
3. Станок-автомат закінчує обробляти деталь.
4. Деталь відсилається на доставку.

Може виникнути запитання: чи не доцільно об’єднати події 2 та 3 в одну подію: оброблення деталі. Таке рішення є логічним, проте, входить в протиріччя з одним з головних принципів моделювання з використанням мереж Петрі – події мають відбуватись миттєво. Щоб обійти це обмеження, процес оброблення (поштучно) виглядає як дві події – початок та кінець оброблення, між якими знаходиться умова: деталь обробляється.

Мережа Петрі на рис. 2.6 ілюструє модель наведеного вище станка-автомата.

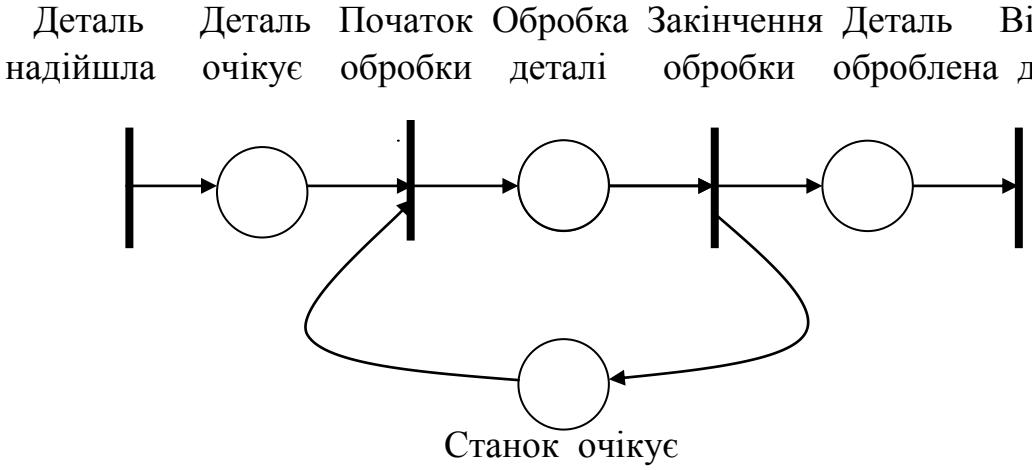


Рисунок 2.6 – Мережа Петрі для станка-автомата

Серед інших, до основних властивостей мереж Петрі є відносять такі.

Обмеженість означає, що кількість фішок в будь-якій позиції не може перевищувати деякого заданого числа K (K – обмеженість).

Безпека визначається умовою K=1, тобто кількість фішок в безпечній мережі Петрі не перевищує одиниці.

Досяжність передбачає, що з будь-якого стану, який визначений як початковий, можливий перехід в будь-який інший стан. Аналіз досяжності дозволяє визначити множину маркувань, в які можливі переходи в процесі функціонування складного об'єкта. Потрібно зазначити, що до аналізу досяжності певних маркувань зводиться багато задач, зокрема проєктування засобів для виконання певних функцій.

Збережність – це властивість мережі зберігати (не збільшувати і не зменшувати) кількість фішок в усіх позиціях разом. Для строго збереженої мережі Петрі

$$\sum \mu(p_i) = \sum_{i=1}^{n} \mu_0(p_i), \qquad (2.6)$$

де $\mu_0(p_i)$ – кількість фішок за початкового маркування;

$\mu(p_i)$ – кількість фішок в *i*-й вершині мережі Петрі на аналізовному кроці функціонування складного об'єкта;

i – *i*-тий вузол мережі Петрі (*i*-тий стан складного об'єкта p_i), $i = \overline{1, n}$;

n – кількість станів складного об'єкта.

Збережність важлива у разі моделювання збереження ресурсів.

Живучість (активність) означає, що з будь-якого стану складного об'єкта можливий перехід в будь-який інший його стан. Аналіз живучості дозволяє виявити наявність тупиків, зациклювань, блокувань тощо, наприклад, в процесі передачі повідомлень в обчислювальній мережі або під час виконання паралельних програм в багатопроцесорній системі.

Існують два основних методи аналізу вказаних властивостей мереж Петрі. Перший з них базується на побудові дерева досяжності, другий – на матричних рівняннях. Ми розглянемо лише перший з них. Детальний опис матричного підходу можна знайти в [6, 10].

Розглянемо марковану мережу Петрі, подану на рис. 2.7. Опис мережі Петрі має вигляд:

$$P = \{p_1, p_2, p_3\};$$

$T = \{t_1, t_2, t_3\}$;

$I(t_1) = \{p_1\}$;

$I(t_2) = \{p_1\}$;

$I(t_3) = \{p_2, p_3\}$;

$O(t_1) = \{p_1, p_2\}$;

$O(t_2) = \{p_2, p_3\}$;

$O(t_3) = \{p_3\}$;

Її початкове маркування – (1, 0, 0). В цьому початковому маркуванні дозволено два переходи: t_1 і t_2, оскільки стан p_1 можна змінити за виконання умов t_1 і t_2 (із стану p_1 виходять направлені дуги до умов t_1 і t_2). Нові активні стани в дереві за виконання умов t_1 і t_2 визначаються у векторі станів відповідно як (1, 1, 0) та (0, 1, 1), як показано на рисунку 2.8.

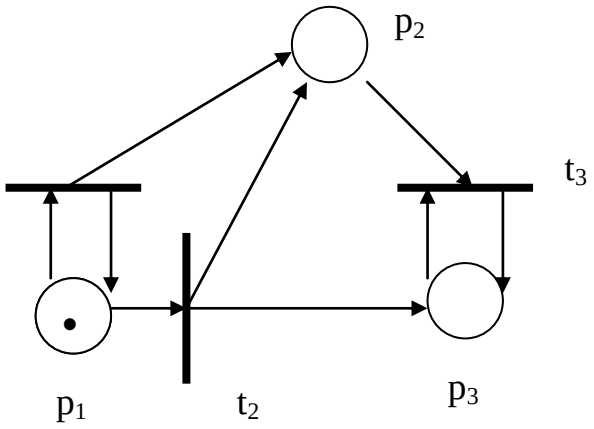


Рисунок 2.7 – Маркована мережа Петрі

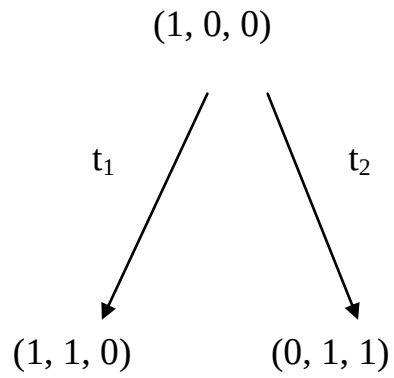


Рисунок 2.8 – Перший крок побудови дерева досяжності

Тепер необхідно розглянути всі маркування, досяжні з нових маркувань. Маркування (1, 1, 0) потрібно розглядати як суперпозицію маркувань (1, 0, 0), що означає активність стану p_1 та (0, 1, 0), що означає активність стану p_2. Активний стан p_1 може бути змінений за виконання умов t_1 і t_2, що приведе до векторів станів (1, 1, 0) та (0, 1, 1). Активний стан p_2 може бути змінений за виконання умови t_3, що приведе, за аналогічними міркуваннями, до векторів станів (0, 0, 1) та (0, 0, 1). Дерево досяжності на цьому кроці набуде вигляду, як показано на рисунку 2.9. Якщо стан складного об’єкта після виконання умови повторюється, це зазначають в дереві досяжності зміною символу у відповідному стані (на інший символ обраного алфавіту (у прикладі це w)). Процедуру потрібно проводити до тих пір, поки не повторяться вже описані в дереві ситуації зміни станів складного об’єкта.

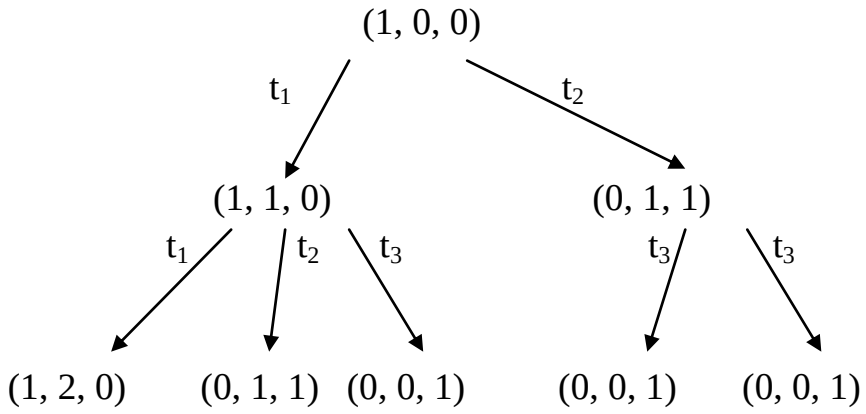


Рисунок 2.9 –Другий крок побудови дерева досяжності

Дерево досяжності має кінцевий вигляд як показано на рисунку 2.10.

Дерево досяжності можна використовувати для розв’язання задач аналізу безпеки складного об’єкта, збереженості ресурсів, надійності функціонування тощо.

Теорія класичних мереж Петрі динамічно розвивається. В той самий час з’являються нові різновиди мереж Петрі: часові (в яких умова (перехід)

спрацьовує не миттєво, а протягом певного проміжку часу), стохастичні (із заданими ймовірностями виконання умов (переходів)) та інші.

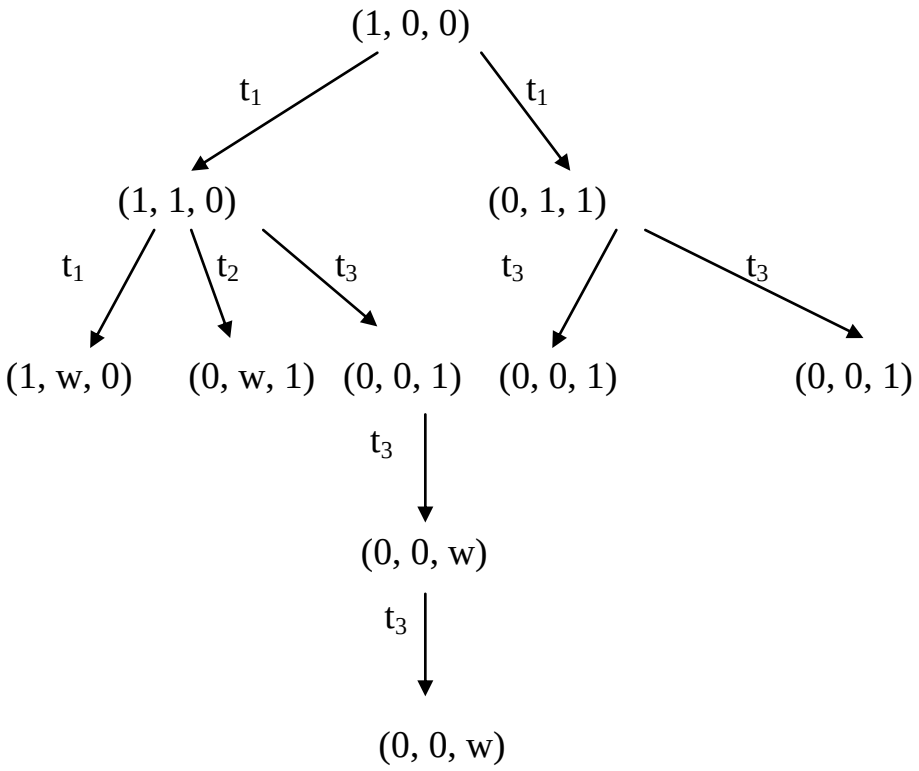


Рисунок 2.10 – Третій крок побудови дерева досяжності

Кейс 4. Опишіть функціонування складного об’єкта з використанням мереж Петрі, визначіть властивості та побудуйте відповідне дерево досяжності.

3 ФУНКЦІОНАЛЬНО-ЛОГІЧНЕ ПРОЄКТУВАННЯ

3.1 Основні задачі етапу функціонально-логічного проєктування

Основними задачами функціонально-логічного рівня проєктування складного об’єкта є розробка його функціонала. Функціонально-логічний рівень проєктування складного об’єкта, таким чином, наслідує структурний рівень проєктування. Це потребує розв’язання задач синтезу складного об’єкта та аналізу можливих рішень чи виконання покладених функцій. Потрібно визнати, що задачі синтезу в автоматизованому режимі вирішити, як правило, не вдається, оскільки: по-перше, не всі процедури синтезу формалізовані; по-друге, обсяг обчислень із зростанням розмірності задачі збільшується не лінійно, а експоненційно (тобто в багатьох випадках не існує ефективних обчислювальних алгоритмів для розв’язання таких задач). Навпаки, задачі аналізу розв’язуються в автоматизованому режимі просто, як задачі, що базуються на порівнянні.

Крім того, на етапі функціонально-логічного проєктування вирішується задача синтезу тестів для контролю та діагностування коректності виконання покладених на складний об’єкт функцій.

3.2 Математичне забезпечення комп’ютерного проєктування складних об’єктів

З погляду блочно-ієрархічного підходу, математичний опис складного об’єкта на мікрорівні є описом процесів, що протікають в неперервному середовищі. Елементами рівня, у цьому разі, виступають складові, функціонування яких може бути описаним диференціальними рівняннями у частинних похідних. Макрорівень – подання середовища, в якому проводиться моделювання як дискретного простору, перехід до зосереджених математичних моделей. Вихідні параметри мікрорівня є вхідними для макрорівня. Елементи метарівня – це складні вузли проєктованого об’єкта, складовими яких також є складові макрорівня.

Серед основних вимог, що висуваються до математичних моделей, особливої уваги заслуговують точність, універсальність та економічність.

Точність відбиває ступінь відповідності значень параметрів реального об'єкта значенням, що надаються математичною моделлю. Проте, кількісно оцінити точність математичної моделі досить складно з таких причин:

1. Реальні об'єкти і їх математичні моделі характеризуються не одним, а множиною параметрів.

2. Моделі складаються для багаторазового використання під час аналізу варіантів об'єкта певного класу – оцінка точності неоднозначна.

3. Спектр значень параметрів об'єкта звичайно ототожнюють з експериментально отриманими (без похибки експерименту).

Економічність – кількість параметрів, що враховуються під час її аналізу.

Універсальність – здатність використовувати математичні моделі для різноманітності варіантів застосування.

Як зрозуміло, всі зазначені вимоги до математичних моделей є суперечливими.

Математичні моделі складних об’єктів класифікують за низкою ознак. Розглянемо основні з них.

За характером відображуваних властивостей математичні моделі складних об’єктів поділяють на:

– функціональні моделі, що відображають процеси функціонування складного об’єкта у вигляді систем рівнянь;
– структурні моделі, що відображають лише структурні властивості складного об’єкта.

До функціональних математичних моделей можна віднести:

– статичні моделі, що відображають аналітичні залежності між вхідними змінними й внутрішніми зв’язками та вихідними змінними на певному рівні ієрархії без урахування їх змін у часі;
– динамічні моделі, відображають аналітичні залежності між вхідними змінними й внутрішніми зв’язками та вихідними змінними на певному рівні ієрархії з урахуванням їх змін у часі, як похідними змінних;
– дискретні (логічні) моделі;
– неперервні моделі.

До структурних математичних моделей складних об’єктів, що поділяються за характером змінних, можна віднести:

– фазові моделі – моделі, які оперують з фазовими змінними (функціями часу), наприклад, імітаційні моделі відображають поведінку системи незалежно від логічних значень, її параметрів: вхідні змінні (впливи) і вихідні змінні (реакції) подають у вигляді залежностей відповідних фазових змінних у часі;
– факторні моделі, в яких вхідні значення визначаються змінними в моделі, які виступають факторами, наприклад, аналітична модель (аргумент носить нелогічний характер), алгоритмічна модель (параметри чи змінні подаються системою рівнянь, яка визначає алгоритм розв’язування задачі).

Макромодель складного об’єкта відображає аналітичні залежності між вхідними змінними та вихідними змінними на певному рівні ієрархії з/без урахування їх змін у часі.

Повна математична модель складного об’єкта відображає аналітичні залежності між вхідними змінними й внутрішніми зв’язками та вихідними змінними на всіх рівнях ієрархії з урахуванням їх змін у часі, як похідними змінних.

Серед особливостей моделювання складних об'єктів уваги заслуговує таке.

1. Стан складових складного об’єкта характеризується фазовими змінними одного типу, які позначаються як інформація (водночас фізична природа змінної (струм, напруга) не конкретизується).
2. Фазові змінні доцільно подавати в дискретній формі, оскільки інформація має цифрову форму.
3. Аналіз функціональних схем відбувається в дискретному часі.

Математичні моделі складних об’єктів подаються у явній формі або неявній формі.

Явна форма подання математичної моделі складного об’єкта передбачає чітку залежність вихідного вектора $\boldsymbol{Y}$ змінних (залежних змінних) від вхідного вектора $\boldsymbol{(W, V, M, N)}$ змінних (незалежних змінних):

$$\boldsymbol{Y = Z\,(W, V, M, N)}, \qquad (3.1)$$

де $\boldsymbol{Y}$ – вектор вихідних залежних змінних;

$\boldsymbol{W, V, M, N}$ – вхідні незалежні змінні;

$\boldsymbol{Z}$ – функціонал, що забезпечує явний опис покладених функцій на складний об’єкт.

Якщо вихідний вектор $\boldsymbol{Y}$ змінних (залежних змінних) помістити з одного боку у рівнянні разом з вхідним вектором $\boldsymbol{(W, V, M, N)}$ змінних (незалежних змінних), отримаємо неявну форму подання математичної моделі складного об’єкта

$$\boldsymbol{F(W, V, M, N, Y) = 0}, \qquad (3.2)$$

де $\boldsymbol{F}$ – функціонал, що забезпечує неявний опис покладених функцій на складний об’єкт.

Розглянемо математичний опис складного об’єкта різними математичними моделями з урахуванням блочно-ієрархічного підходу.

Нехай складний об’єкт А (як показано на рисунку 3.1) може бути подано на першому ієрархічному рівні множиною взаємодіючих модулів M_1, M_2, M_3 та M_4, кожен з яких, й собі, на другому ієрархічному рівні може бути подано множиною складових C_{ij}, де і – номер модуля складного об’єкта А; j – номер складової в модулі складного об’єкта А.

Приймемо такі позначення індексів:

ст – статична модель;

дин – динамічна модель;

ПММ – повна математична модель;

макро – макромодель.

Крім того в індексі має бути зазначено об’єкт опису. Отже,

$Fст_A$ є описом статичної моделі об’єкта А;

$Fдин_{M3}$ є описом динамічної моделі об’єкта M_3;

$Fмакро_{M3}$ є описом макромоделі об’єкта M_3;

$Fпмм_A$ є описом повної математичної моделі об’єкта А.

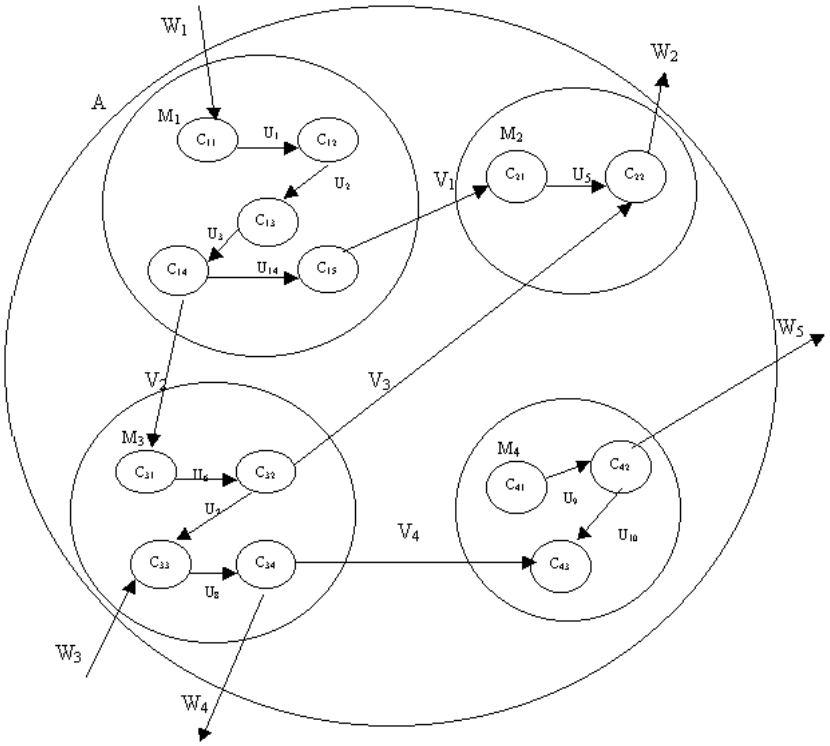


Рисунок 3.1 – Подання складного об’єкта А з урахуванням блочно-ієрархічного підходу

Тоді, відповідно до означення, зазначені математичні моделі матимуть вигляд:

$Fc\text{т}_\text{A}$ ($W1$,$W2$, $W3$,$W4$, $W5$, $V1$, $V2$, $V3$, $V4$) = 0;

$F\text{дин}_\text{A}$ ($W1$,$W2$, $W3$,$W4$, $W5$, $W1'$,$W2'$, $W3'$,$W4'$, $W5'$, $V1$, $V2$, $V3$, $V4$, $V1'$, $V2'$, $V3'$, $V4'$) = 0;

$Fc\text{т}_{M2}$($W2$, $V1$, $V3$, $U5$) = 0;

$F\text{дин}_{M3}$ ($W3$, $W4$,$V2$, $V3$, $V4$, $U6$, $U7$, $U8$, $W3'$, $W4'$, $V2'$, $V3'$ $V4'$, $U6'$, $U7'$, $U8'$) = 0;

$F\text{макро}_{M3}$ ($W4$, $W3$, $V2$, $V3$, $V4$) = 0;

$F\text{пмм}_\text{A}$ ($W1$,$W2$, $W3$,$W4$, $W5$, $W1'$,$W2'$, $W3'$,$W4'$, $W5'$, $V1$, $V2$, $V3$, $V4$, $U1$, $U2$, $U3$, $U14$, $U5$, $U6$, $U7$, $U8$, $U9$, $U10$, $V1'$, $V2'$, $V3'$, $V14'$, $U1'$, $U2'$, $U3'$, $U4'$, $U5'$, $U6'$, $U7'$, $U8'$, $U9'$, $U10'$) = 0;

F макро_A ($W1$,$W2$, $W3$, $W4$,$W5$) = 0.

Методи синтезу математичних моделей складного об’єкта поділяють на такі групи.

Група 1 –методи одержання математичних моделей складових складного об’єкта і математичних моделей складного об’єкта на різних ієрархічних рівнях з використанням неформальних (евристичних) прийомів і процедур – підбору виду математичних співвідношень моделі. До цієї групи методів відносять:

- теоретичні методи, що базуються на використанні фізичних закономірностей з урахуванням властивостей, які знаходять відображення в моделях процесів (допускається ряд припущень, що відображають особливості вузького класу моделей об’єкта). Наприклад, алгоритмічні моделі;
- експериментальні методи, що базуються на використанні експериментально отриманих залежностей між важливими

параметрами/характеристиками, зазначеними користувачем. Наприклад, факторні моделі.

Група 2 – методи одержання повної математичної моделі складного об’єкта із заданих математичних моделей складових з урахуванням блочно-ієрархічного підходу. Передбачається, що кожна із складових повністю визначена стосовно вхідних і вихідних змінних, а, отже, процедури одержання моделей цими методами можуть бути повністю формалізованими.

Кейс 5. Запропонуйте структуру складного об’єкта, що виконує певну функцію. Передбачте не менше 3 рівнів деталізації, не менше 5 складових, визначивши функціональне навантаження кожної з них. Наведіть функціональні моделі (статичну, динамічну, повну математичну та макромодель) кожної складової окремо й складного об’єкта загалом.

3.3 Моделювання функціонування складного об’єкта

Моделювання функціонування складного об’єкта, що у своїй структурі не має внутрішніх зворотних зв’язків (комбінаційного типу), є порівняно простим. В процесі моделювання відбувається послідовне поширення сигналів від входів до виходів складного об’єкта. У цьому випадку послідовно активуються відповідні моделі складових – спеціальні процедури, які зберігаються в бібліотеках баз даних. Моделювання складного об’єкта з наявними внутрішніми зворотними зв’язками (послідовнісного типу) є складним процесом, оскільки на входах деяких складових можуть змінюватись значення сигналів внаслідок наявності ланцюгів зворотних зв’язків. Тому моделювання функціонування таких складних об’єктів – процес ітераційний.

3.3.1 Ранжування складних об’єктів

В процесі моделювання функціонування складного об’єкта важливим є визначення порядку реалізації рівнянь моделі згідно з поширенням сигналів в реальному складному об’єкті. З цією метою потрібно розв’язати задачу ранжування складного об’єкта.

Складний об’єкт є відранжованим, якщо всім його складовим та ланцюгам присвоєно ранги.

Під час ранжування складного об’єкта діють такі правила.

Для складного об’єкта комбінаційного типу:

– 1-й ранг надається таким складовим, всі входи яких підключені до вхідних ланцюгів складного об’єкта;
– (k+1)-й ранг надається складовим, хоча б один вхід яких з’єднується з виходом складової k-го рангу, а інші входи – з виходами складових не вище k-го рангу.

Щоб вирішити задачу ранжування для складного об’єкта послідовнісного типу, потрібно виявити внутрішні зворотні зв’язки. Внаслідок розриву таких ланцюгів зворотних зв’язків отримуємо складний об’єкт комбінаційного типу, що впорядковується за правилами, як було вказано вище.

Таким чином, алгоритм ранжування складного об’єкта послідовносного типу містить кроки [6,11]:

Перший етап. Вхідним ланцюгам складного об’єкта надається 1-й ранг.

Другий етап. Послідовно розглядаються всі складові, яким ранг ще не присвоєно. Якщо хоча б один вхідний ланцюг складової не має рангу, визначити його ранг складової неможливо. В іншому випадку

$$r(d_i) = \max_{k_j \in A} r(k_j),$$

де A – множина входів складової;

k_j– ранг j–го вхідного ланцюга.

Вихідним ланцюгам складової d_i надається ранг $(r(d_i) + 1)$, якщо $r(d_i) \neq 0$, де $r(d_i)$ – ранг складової d_i.

Для схем з шинною структурою можлива ситуація, коли ланцюг містить вихідні ланцюги декількох складових. В такому випадку присвоїти ранг шині можна лише тоді, коли визначено ранги всіх складових, виходи яких об'єднуються шиною, як найвищого рангу вихідного ланцюга, що входить до шини.

Якщо складова не має вхідних ланцюгів (це може статися після розриву зворотних зв'язків), їй надається 1-й ранг (найнижчий ранг, такий, що надається вхідним ланцюгам складного об'єкта).

Другий етап повторюється до виконання однієї з умов:

а) всім складовим було присвоєно ранги;

б) жодній складовій неможливо присвоїти ранг. В цьому випадку переходимо до 3-го етапу.

Третій етап. В складному об'єкті виділяється контур і визначається ланцюг зворотного зв'язку. Для цього будується ланцюг складових, яким ще не присвоєно ранг. Знаходимо складову, що не має рангу. Серед її вхідних ланцюгів є такі, що не мають рангу (інакше складовій було б присвоєно ранг). Обираємо один з ланцюгів і також вносимо в ланцюг. Далі знаходимо, вихід якої складової з'єднаний з цим входом; його також вносимо в ланцюг. Цей процес продовжуємо, поки не зустрінемо складову, яка вже належить до ланцюга. Це вказує на існування контура. Щоб знайти сам контур, потрібно визначити, де в ланцюгу ця складова зустрілася вперше. Всі складові, які розташовані в ланцюгу праворуч від цього місця, утворюють контур.

Припустимо, що складено ланцюг складових

$$d_{i1}, d_{i2}, \ldots, d_k, d_{ik+1}, \ldots, d_{is},$$

ранги яким ще не присвоєно. Якщо під час дослідження складової d_{is} наступною визначається складова d_{ik}, яку вже внесено до ланцюга, то отримуємо контур складових

$$d_{ik}, d_{ik+1}, \ldots, d_{is-1}, d_{is}.$$

Вихідний ланцюг будь-якої з цих складових можна вважати ланцюгом зворотного зв'язку. Після визначення контура зворотного зв'язку переходимо до четвертого етапу.

Четвертий етап. Обираємо ланцюг зворотного зв'язку та розриваємо його. Під час вибору вихідного ланцюга складової контура, перевага надається ланцюгу, який є вхідним ланцюгом більш, ніж одної складової складного об'єкта. Після розриву зворотного зв'язку вихідний ланцюг складової d_{ik+1} вилучається з списку вхідних ланцюгів d_{ik}. Після розриву ланцюга зворотного зв'язку переходимо до другого етапу ранжування складного об'єкта. Робота алгоритму закінчується, коли всім складовим складного об'єкта буде присвоєно ранги. Алгоритм є швидкодійним та просто реалізується на ЕОМ за комп'ютерного проєктування складного об'єкта, хоча й не гарантує, що кількість зворотних зв'язків буде мінімальною.

Розглянемо алгоритм ранжування на прикладі складного hard-об'єкта, поданого на рис. 3.2.

1. Вхідним ланцюгам надається перший ранг:

$$r(k_1) = r(k_2) = r(k_3) = r(k_4) = r(k_5) = r(k_6) = r(k_7) = r(k_8) = 1.$$

2. *1-а ітерація*. Ранг складової d_1 дорівнює максимальному рангу її вхідних ланцюгів:

$$r(d_1) = \max [r(k_1), r(k_2)] = 1.$$

Оскільки ранги складових d_4 та d_6 не визначено, неможливо присвоїти ранг вихідному ланцюгу k_9. Також не вдається присвоїти ранги складовим d_2, d_3, d_6, тому що ранги ланцюгів k_9 та k_{10} невідомі.

Визначаємо ранг складової d_5 та її вихідного ланцюга:

$$r(d_5) = \max [r(k_4), r(k_5)] = 1;$$

$$r(k_{10}) = r(d_5) + 1 = 2.$$

2-а ітерація.

$$r(d_4) = \max [r(k_{10}), r(k_6)] = 2.$$

3-я ітерація. Під час третьої ітерації не присвоєно ранг жодній новій складовій, тому переходимо до наступного кроку.

3. Визначається складова з неприсвоєним рангом. Припустимо, це складова d_2. Вносимо її в ланцюг як початкову. Вхідному ланцюгу k_9 складової d_2 не присвоєно ранг, тобто $r(k_9) = 0$. Ланцюг k_9 містить виходи декількох складових. Складову d_6 ($r(d_6) = 0$) вносимо в ланцюг $[d_2, d_6]$. Вхідному ланцюгу k_{14} складової d_6 не присвоєно ранг, тобто $r(k_{14})=0$. В ланцюг $[d_2, d_6, d_8]$ вносимо складову d_8, вихід якої належить ланцюгу k_{14}. Далі, вносячи в ланцюг складові d_3 та d_7, отримуємо $[d_2, d_6, d_8, d_3, d_7]$.

Для складової d_7 маємо $r(k_9) = 0$ та $r(k_{12}) = 0$. Обираємо один ланцюг, припустимо k_{12}. Переходимо до складової d_3, яка вже зустрічалася в ланцюгу. Відповідно, складові d_3 та d_7 об’єднано в контур.

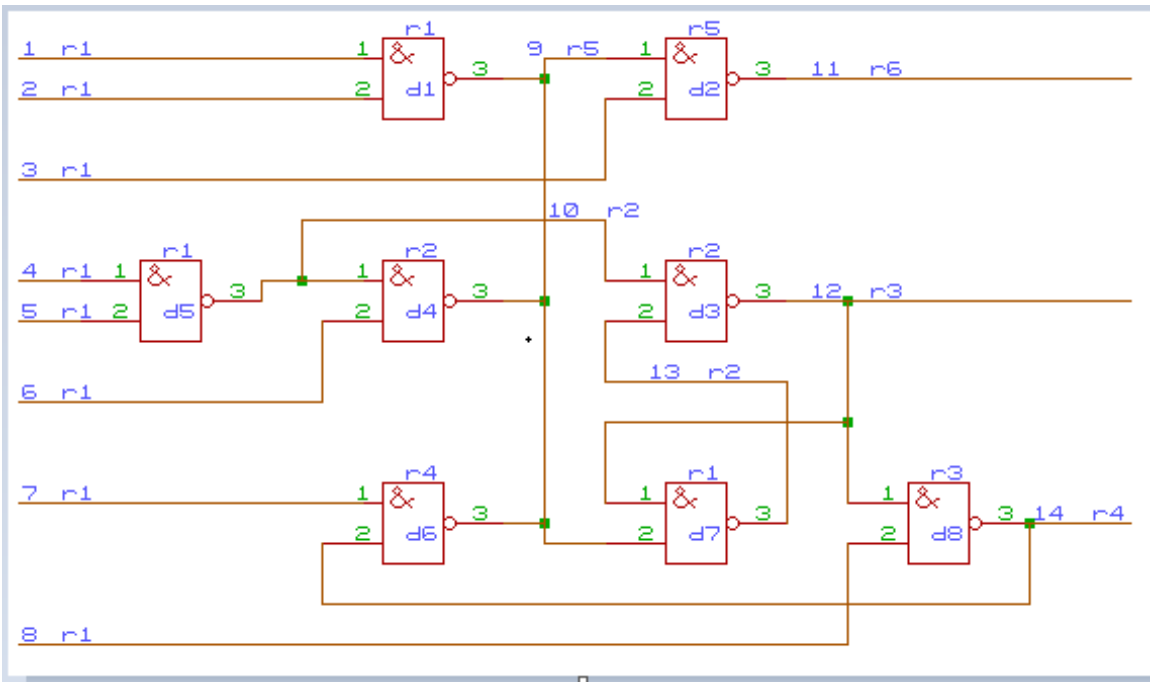


Рисунок 3.2 – Приклад складного об’єкта

4. Вибирається вихідний ланцюг k_{12} складової d_3 як ланцюг зворотного зв'язку. Він є входом для двох складових d_7 та d_8. Ланцюг k_{12} вилучається зі списку вхідних ланцюгів складової d_7.
5. Жодній новій складовій ранг присвоїти не вдається.
6. Формується ланцюг, починаючи зі складової d_2: $[d_2, d_6, d_8, d_3, d_7]$.

В цьому випадку для складової d_7 лише $r(k_9) = 0$. Ланцюг k_{12} з списку вхідних ланцюгів складової d_7 вилучено. Ланцюг k_9 містить вихід складової d_6, для якої $r(d_6) = 0$.

Переходимо до складової d_6, яку вже внесено в ланцюг. Відповідно, складові d_6, d_8, d_3, d_7 об'єднано в контур.

7. Обираємо вихідний ланцюг k_9 складової d_6 як ланцюг зворотного зв'язку. Він є вхідним ланцюгом для двох елементів d_2 та d_7. Ланцюг k_9 вилучається зі списку вхідних ланцюгів елемента d_7.
8. *1-а ітерація*. Елемент d_7 не має вхідних ланцюгів і йому надається перший ранг:

$$r(d_7) = 1; \quad r(k_{13}) = 2.$$

2-а ітерація.

$$r(d_3) = \max [r(k_{10}), r(k_{13})] = 2;$$

$$r(k_{12}) = r(d_3) + 1 = 3.$$

3-я ітерація.

$$r(d_8) = \max [r(k_{12}), r(k_8)] = 3;$$

$$r(k_{14}) = r(d_8) + 1 = 4.$$

4-а ітерація.

$$r(d_6) = \max [r(k_{14}), r(k_7) = 4.$$

Тепер можна визначити ранг ланцюга k_9, оскільки визначено ранги всіх елементів, виходи яких належать цьому ланцюгу:

$$r(k_9) = \max [r(d_6), r(d_4), r(d_2)] + 1 = 5.$$

5-а ітерація.

$$r(d_2) = \max [r(k_9), r(k_3)] = 5;$$

$$r(k_{11}) = r(d_2) + 1 = 6.$$

Таким чином, всім складовим і ланцюгам складного об’єкта присвоєно ранги. Ранжування завершено.

3.3.2 Аналіз функціонування складного об’єкта

Для запису математичної моделі складного об’єкта в загальному випадку зручно ввести такі позначення: U– вектор вхідних змінних складного об’єкта; Y та V – вектори вихідних та внутрішніх змінних всіх складових складного об’єкта для поточного t та майбутнього t′ часу, відповідно. Для кожної складової складного об’єкта змінні **V′** для моменту майбутнього часу t′ визначаються як для $(t+\Delta t_j)$, тобто значення затримок Δt_j можуть бути різними для різних змінних.

Вихідні змінні складного об’єкта є вихідними змінними деяких складових і, відповідно, входять до векторів V та V′. Тоді математична модель складного об’єкта є системою рівнянь

$$\mathbf{Y} = \mathbf{F}(\mathbf{V}, \mathbf{U}) \qquad (3.3)$$

з дискретними змінними Y, V, U, в якій F – це оператор перетворення дискретних змінних.

Модель (3.3) називається асинхронною моделлю складного об’єкта.

Найбільш точно поведінку логічних схем відображає асинхронне моделювання. Асинхронна модель будується на припущенні, що кожна *j*-та складова складного об’єкта затримує сигнал на час Δt_j. Для аналізу перехідного процесу функціонування складного об’єкта необхідно розглядати як асинхроне в межах одного такту його роботи. Це означає, що складові переключаються асинхронно зі зміною станів сигналів на їх входах, але тривалість перехідного процесу не перевищує тривалості такту роботи складного об’єкта ΔT, який має бути кратним абстрактній затримці

Δt_j і не має бути меншим за максимальну затримку сигналу складовими в складному об’єкті. Оскільки тривалість знаходження в певному стані вхідних сигналів дорівнює такту роботи складного об’єкта, то до моменту зміни сигналів на входах складного об’єкта перехідний процес закінчується.

Якщо всі затримки $\Delta t_j \neq 0$, то асинхронна модель (3.3) є сукупністю рекурентних співвідношень, що дозволяють за відомих значень змінних V та заданої функції U(t) обчислити значення вектора V′ у всьому часовому діапазоні, тобто побудувати часові діаграми роботи складного об’єкта.

Асинхронні моделі є універсальними, оскільки можуть застосовуватись для аналізу як асинхронних, так і синхронних схем, дослідження перехідних процесів та дослідження усталених станів складного об’єкта. Проте використання асинхронних моделей супроводжується великими витратами машинного часу.

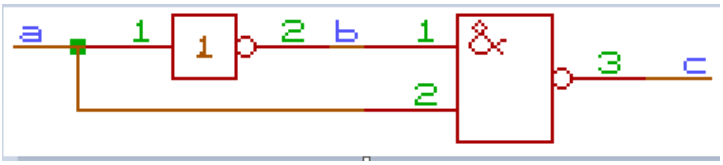
Синхронні моделі (тобто такі, в яких всі затримки $\Delta t_j = 0$) використовують для перевірки коректності з’єднань складових в складному об’єкті, для виявлення ризиків збою, під час побудови тестів. Якщо в синхронній моделі складного об’єкта використовуються булеві змінні, то модель називається двозначною моделлю. Двозначні моделі найбільш економічні, моделювання в двійковому алфавіті найбільш швидке, проте за їх допомогою можливо вирішувати лише частину задач, наприклад, виявлення «грубих» помилок (неправильне виконання функцій) в процесі побудови складного об’єкта. Але вони не можуть вирішувати ряд інших задач, наприклад, виявлення ризиків збою.

Ризиком збою називають можливість появи хибних сигналів на виході складного об’єкта або його складової [6, 12]. Якщо сигнали на виході складової для двох сусідніх наборів вхідних сигналів A та B залишаються однаковими, але під час перехідного процесу можлива поява хибного

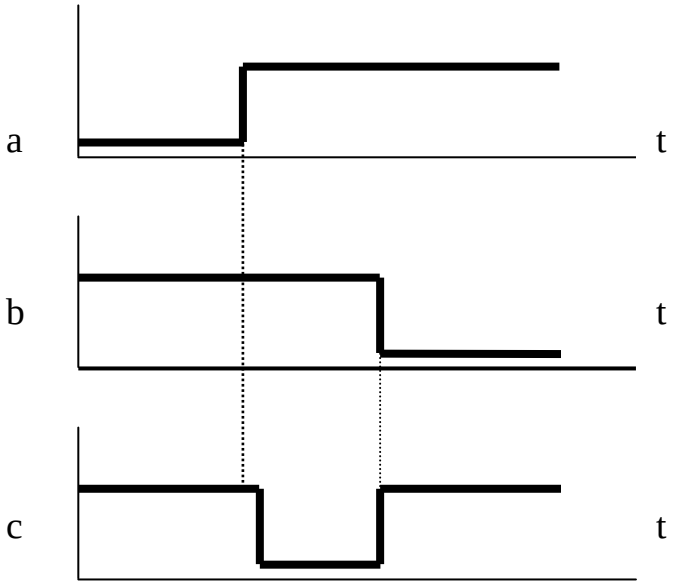
сигнала протилежного значення, то така ситуація називається статичним ризиком збою.

Статичний ризик збою ілюструється на рис. 3.3. Внаслідок затримки сигналу в точці *b* відносно точки *a*, зміна сигналу в точці *a* з 0 на 1 призводить до появи на виході *c* хибного сигналу: сигнал *a* змінюється (0→1) раніше сигналу *b* (1→0).

Динамічний ризик збою припускає можливість багатократної зміни значень сигналу на виході під час переходу від вхідного набору A до набору B, коли вихідний сигнал схеми змінюється на протилежний.



а)



б)

Рисунок 3.3 – Статичний ризик збою: а) фрагмент складного об’єкта; б) часові діаграми функціонування

Зміна 0→1 сигналу *d* раніше зміни 0→1 та 1→0 сигналів *a* та *b* під час переходу від вхідного набору A=*abd* = 010 до набору B=*abd*=101 призводить до динамічного ризику збою (рис. 3.4).

З часової діаграми роботи складових бачимо, що динамічний ризик збою є наслідком статичного ризику збою. В процесі аналізу функціонування складних об’єктів для виявлення ризику збою необхідно враховувати часове непогодження вхідних сигналів складових.

Під час аналізу функціонування складного об’єкта часто використовують різні види моделювання, які дозволяють відтворити поведінку складного об’єкта або його окремих складових у разі подачі набору вхідних сигналів. Методи моделювання носять універсальний характер і застосовуються в тому випадку, коли прямі методи перевірки не розроблено або виявляються занадто складними.

Під час синхронного моделювання функціонування складного об’єкта досліджується лише логіка роботи складного об’єкта на основі ідеалізованої моделі. Моделювання функціонування складного об’єкта зводиться до розв’язання системи рівнянь, кожне з яких моделює логіку функціонування однієї складової. Процес зміни станів складових складного об’єкта в процесі моделювання синхронізується упорядкуванням складових за рівнями запуску (згідно з алгоритмом ранжування). Моделювання починається з задання вектору вхідних змінних на незалежних входах складного об’єкта, далі моделюється функціонування складових 1-го, 2-го та наступних рангів. Для складних об’єктів зі зворотними зв’язками моделювання повторюється декілька разів, поки складний об’єкт не перейде в стійкий стан або не виникнуть коливання сигналів.

Для виявлення статичного ризику збою в складних об’єктах комбінаційного типу використовується трійкове моделювання, а саме – крім значень «0» та «1» вводиться значення сигналу під час перехідного процесу, яке позначається *x*. Розглянемо, для прикладу, реалізацію принципу трійкового моделювання на складових, що реалізують операції І, АБО, НЕ (табл. 3.1) [6, 8].

Таблиця 3.1 – Таблиці істинності для функцій І, АБО, НЕ для трійкового моделювання

І			АБО			НЕ	
A	B	Y	A	B	Y	a	Y
0	0	0	0	0	0	0	1
0	X	0	0	X	x	x	X
0	1	0	0	1	1	1	1
X	0	0	X	0	x	-	-
X	X	X	X	X	x	-	-
X	1	X	X	1	1	-	-
1	0	0	1	0	1	-	-
1	X	X	1	X	1	-	-
1	1	1	1	1	1	-	-

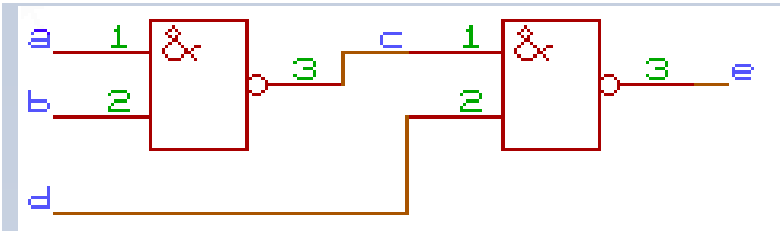
Під час трійкового моделювання функціонування складного об’єкта, крім вхідних наборів А та В, використовується набір А/В, що визначає стан складного об’єкта під час перехідного процесу.

Якщо значення вихідних сигналів складової для вхідних наборів А та В збігаються, а для перехідного набору А/В зафіксовано невизначений стан *x*, на виході складової може з’явитися хибний сигнал і складний об’єкт в процесі функціонування може мати статичний ризик збою.

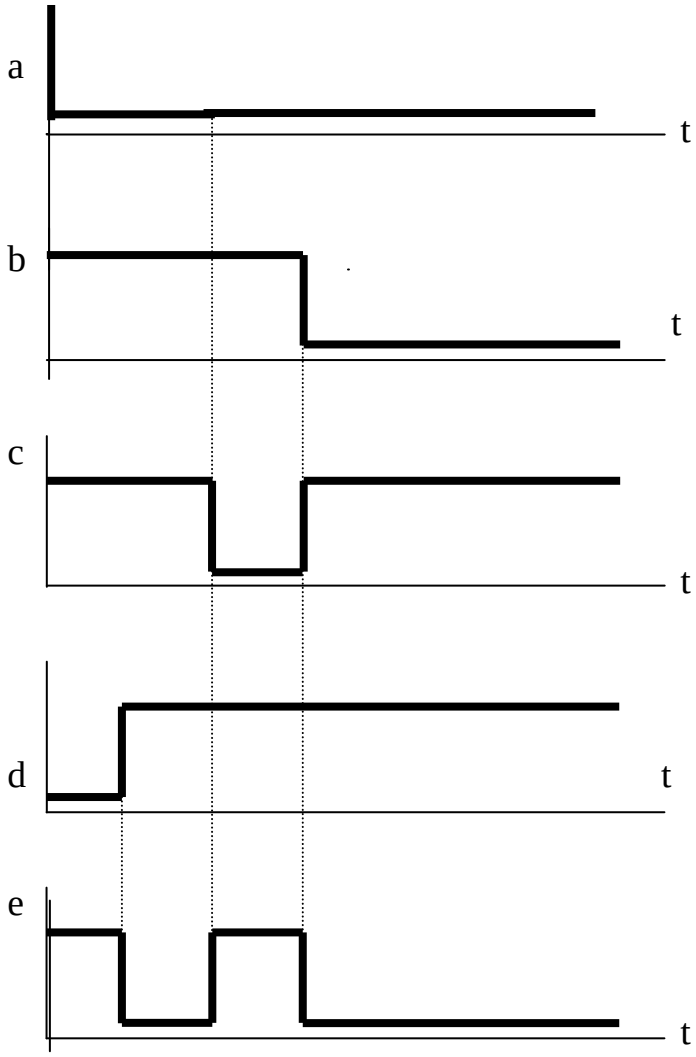
Для складного об’єкта, наведеного на рис. 3.4, а), часові діаграми функціонування наведено на рис. 3.4, б), а значення сигналів за трійкового моделювання наведено в табл. 3.2.

Таблиця 3.2 –Значення сигналів за трійкового моделювання

	a	b	d	c	e
A	0	1	0	1	1
A/B	*x*	*x*	*x*	*x*	*x*
B	1	0	1	1	0



а)



б)

Рисунок 3.4 – Динамічний ризик збою: а) фрагмент складного об’єкта; б) часові діаграми функціонування

Розглянемо точку *c*. Оскільки для вхідних наборів A та B значення сигналу не змінюється, а для перехідного набору A/B змінюється, у функціонуванні складного об’єкта наявний статичний ризик збою. Таким чином, трійкове моделювання дозволяє виявити статичний ризик збою.

Проте, наявність в схемі динамічного ризику збою трійкове моделювання виявити не може. З метою виявлення динамічного ризику збою застосовують п’ятизначне моделювання. Приклад реалізації функцій І, АБО, НЕ та особливості моделювання функціонування складного об’єкта з використанням п’ятизначного алфавіту описано в [1].

Внутрішній стан схеми з пам’яттю визначається сигналами зворотного зв’язку. Вхідний набір може змінити значення кількох зворотних зв’язків. Затримки в схемі викликають змагання сигналів, які можуть виявитися критичними. Залежно від послідовності зміни значень сигналів зворотного зв’язку схема переходить в різні стійкі стани. Критичні змагання сигналів можуть бути виявлені за допомогою трійкового моделювання.

Для складного об’єкта послідовнісного типу трійкове моделювання триває до тих пір, поки значення сигналів на виході складного об’єкта не перестануть змінюватись. Моделювання починається від деякого заданного стійкого стану, який визначається сигналами зворотного зв’язку та вхідним набором А. Багатократним розв’язанням систем рівнянь функціонування складного об’єкта моделюється для перехідного набору А/В та набору В. Якщо в результаті хоча б один сигнал зворотного зв’язку набуває значення *x*, то в складному об’єкті мають місце критичні змагання сигналів.

Під час моделювання функціонування складного об’єкта з перехідним набором А/В значення *x* з’являються на виходах лише тих складових, сигнали яких можуть змінитися. Якщо значення *x* набуде ланцюг зворотного зв’язку, і наступний набір В не змінює цього значення зворотного зв’язку, а лише підтримує його, має місце неоднозначна поведінка складного об’єкта. Саме така ситуація і носить назву критичних змагань сигналів. Необхідно зауважити, що трійкове моделювання не враховує конкретних значень затримок в складному об’єкті, припускаючи

найгіршу комбінацію затримок його складових. Під час проєктування складного об’єкта звичайно враховують конкретні затримки.

Для фрагмента складного об’єкта, наведеного на рис. 3.5, трійкове моделювання показує наявність статичного ризику збою та критичних змагань.

Маємо вхідний набір A = *abcd* = 1111 і стан складного об’єкта, який визначається сигналом ланцюга зворотного зв’язку *h* = 1. Нехай за трійкового моделювання перехідний набір є таким A/B = 11*x*1. Ланцюг зворотного зв’язку набуває значення *x*, що свідчить про наявність критичних змагань. Насправді хибний сигнал на виході складової d2 не з’являється, і ланцюг зворотного зв’язку утримує те саме значення, оскільки сигнал *c* змінюється раніше сигналу *e* (рис. 3.6).

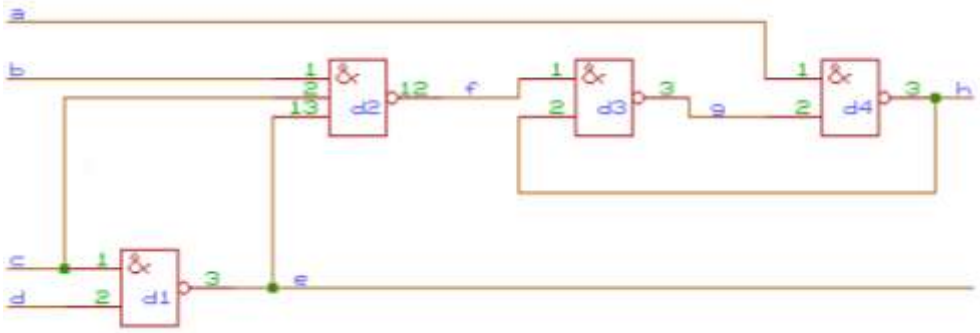


Рисунок 3.5 – Приклад фрагмент складного об’єкта

Моделювання функціонування складного об’єкта здійснюється в дискретні моменти часу з кроком, який дорівнює абстрактній затримці Δt_j. За кожний такт моделювання сигнали в складному об’єкті просуваються за час, що дорівнює Δt_j. Складова знаходиться в стійкому стані, якщо її реакція не протирічить збудженню. Якщо всі складові знаходяться в стійкому стані, то і складний об’єкт перебуває в стійкому стані.

Асинхронні моделі дозволяють відтворити часову послідовність усіх подій для складного досліджуваного об’єкта, тобто побудувати часові діаграми функціонування складного об’єкта для різних вхідних наборів.

Завдяки їм, зокрема, можна встановити, чи є в схемі критичні змагання сигналів.

Але більша точність та універсальність асинхронних моделей досягається за рахунок збільшення трудомісткості обчислень в процесі аналізу. Якщо під час синхронного моделювання обчислення рівнянь виконується лише один або два рази, то за використання асинхронних моделей розв'язання системи рівнянь звичайно виконується $\Delta T/\Delta t_{j\ max} > 1$ раз, де Δt_j − максимальна затримка складовими у певному складному об'єкті; ΔT − тривалість такту роботи складного об'єкта.

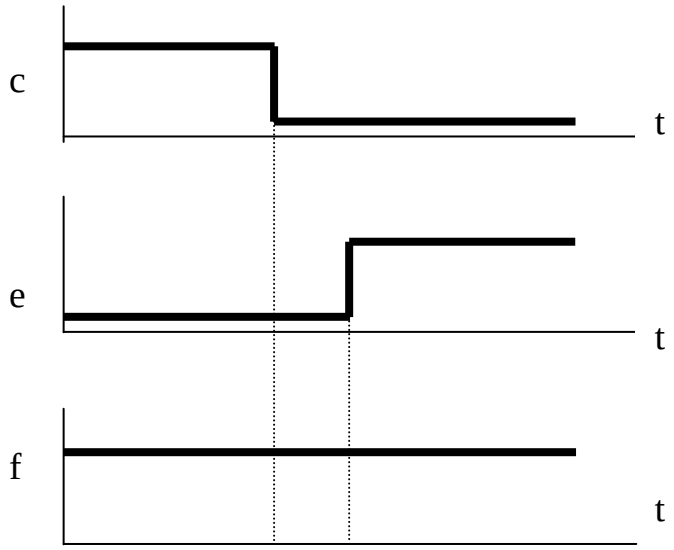


Рисунок 3.6 – Часові діаграми функціонування фрагмента складного об'єкта, наведеного на рис. 3.4

Асинхронне моделювання, так само як і синхронне, може виконуватись в двозначному, тризначному і п'ятизначному алфавіті сигналів.

Алгоритм аналізу функціонування складного об'єкта, який описується асинхронною моделлю (3.4), потребує виконання потужних обчислень, складність яких стрімко зростає з підвищенням його складності. Тому у випадку комп'ютерного проєктування складного об'єкта для аналізу його

функціонування частіше використовуються алгоритми, які реалізують *подійний* метод.

Основна ідея подійного методу полягає в виконанні обчислень лише за рівняннями тільки активних складових, тобто складових, у яких хоча б на одному вході відбулася подія (змінилась вхідна змінна). На кожній ітерації обчислювального процесу існує своя група активних складових. Тому в алгоритмі подійного методу на кожному кроці встановлюється, які складові є активними аби виконати звертання лише до їх моделей. В складних об'єктах на кожному такті, як правило, активізується не більше кількох відсотків від загальної кількості складових. Отже, використання подійного методу дозволяє істотно скоротити витрати машинного часу під час аналізу функціонування складного об'єкта.

Для розв'язання систем логічних рівнянь, що описують функціонування складного об'єкта, застосовуються ітераційні методи: метод простої ітерації, метод Зейделя без ранжування та з ранжуванням. Перед розв'язанням системилогічних рівнянь, в кожному з цих методів мають бути задані вектор вхідних сигналів **U** та вектор початкового наближення $\mathbf{Y_0}$ для вектора **Y**. Розглянемо зазначені методи.

Метод простої ітерації. Цей метод передбачає виконання ітерацій за формулою

$$\mathbf{Y}_i = \mathbf{F}\ (\mathbf{Y}_{i-1}, \mathbf{U}), \qquad (3.4)$$

де $\mathbf{Y}_i$ – значення вектора **Y** на і-й ітерації.

Тобто, якщо $\mathbf{Y}_i = \mathbf{Y}_{i-1}$, то розв'язок знайдено; якщо $\mathbf{Y_i} \neq \mathbf{Y_{i-1}}$, то виконується нова ітерація; якщо ітераційний процес не сходиться, то це свідчить про помилки проєктування складного об'єкта. На практиці вважають, що процес не сходиться, якщо умова $\mathbf{Y}_i = \mathbf{Y}_{i-1}$ не досягається за заданої кількості ітерацій.

Алгоритм методу простої ітерації аналогічний алгоритму асинхронного аналізу функціонування складного об’єкта за умов, коли всі складові складного об’єкта мають однакові затримки. Такий збіг не випадковий: за методом простої ітерації визначається усталене значення **Y**; таке саме усталене значення **Y** буде знайдено у випадку асинхронного моделювання до моменту, коли перехідні процеси в схемі закінчаться.

Як зазначено вище, асинхронний аналіз функціонування складного об’єкта має високу трудомісткість; така сама висока трудомісткіть притаманна методу простої ітерації. Зменшити обсяг обчислень вдається, якщо побудувати ітераційний процес за методом Зейделя. Особливість ітерації за методом Зейделя полягає в тому, що у разі обчислення чергового елемента вектора $\mathbf{Y}_i$ в праву частину (3.5) там, де це можливо, підставляються не елементи вектора $\mathbf{Y}_{i-1}$, а ті елементи вектора $\mathbf{Y}_i$, які вже обчислені до цього моменту. Кількість ітерацій в методі Зейделя істотно залежить від порядка, в якому реалізуються рівняння моделі.

В методі Зейделя без ранжування рівняння моделі перераховуються в довільному порядку.

В методі Зейделя з ранжуванням рівняння розташовуються в тому порядку, який відповідає проходженню інформаційних потоків через складові складного об’єкта. Тоді для аналізу складного об’єкта без внутрішніх зворотних зв’язків потрібна буде лише одна ітерація. В складних об’єктах зі зворотними зв’язками метод Зейделя з ранжуванням потребує кількох ітерацій, але їх кількість істотно менша, ніж в методі простої ітерації.

Як бачимо, метод Зейделя без ранжування за трудомісткістю обіймає проміжне становище між методом простої ітерації та методом Зейделя з ранжуванням. Подальше скорочення обсягу обчислень досягається за допомогою подійного методу, в якому скорочується трудомісткіть кожної

ітерації. Подійний метод можна застосовувати в межах методів простої ітерації та Зейделя під час синхронного та асинхронного моделювання.

3.4 Генерація тестів для складного об’єкта

Перевірка коректності функціонування складного об’єкта відбувається шляхом подачі на його входи деяких вхідних наборів з множини можливих вхідних наборів і порівняння очікуваних результатів функціонування з реакцією складного об’єкта. За допомогою тестів визначається наявність або відсутність несправностей в складному об’єкті. Будь-яка несправність з класу несправностей, для пошуку яких будується тест, має проявитись хоча б на одному наборі.

За допомогою діагностичного тесту локалізується тип та місце несправності в складному об’єкті. Діагностичний словник містить інформацію про всі типи несправностей в складному об’єкті та відповідні реакції на його виходах.

Основними характеристиками тестів є:

1. Повнота тесту. Повний тест – це тест, реалізація якого дозволяє виявити всі несправності із заданого класу несправностей. Кількісною оцінкою повноти контролю є відношення кількості виявлених несправностей до загальної кількості можливих несправностей складного об’єкта заданого класу.

2. Трудомісткість контролю. Трудомісткіть контролю безпосередньо пов’язана з кількістю наборів в тесті. Тому для оцінення трудомісткості звичайно використовують довжину тесту як кількість векторів вхідних змінних, що виявляють заданий тип помилки.

3. Для діагностичних тестів важливою характеристикою також є ступінь локалізації несправностей з множини всіх можливих несправностей складного об’єкта. Звичайно тест дозволяє локалізувати місце несправності з точністю до складової, з точністю

до групи складових, з точністю до блоку певного функціонального призначення тощо, відповідно до блочно-ієрархічного підходу.

Під час побудови тестів до уваги береться обмежений клас несправностей, які підлягають виявленню та локалізації. Як правило, в цей клас вносять несправності:

- константний нуль, коли на виході складової присутнє постійне значення логічного нуля «0», що не залежить від значень сигналів на її входах;
- константна одиниця – те саме, але за наявності на виході значення логічної одиниці «1».

Звичайно тести розробляються для пошуку поодиноких несправностей. Задача побудови тестів для пошуку кратних несправностей, з одного боку потребує великих витрат машинного часу, а з іншого – не має практичного сенсу, оскільки доведено, що тести, які виявляють поодинокі несправності, дозволяють знайти також і кратні.

Найбільшої популярності під час побудови тестів набули такі методи:

- ймовірностні процедури;
- d-алгоритм;
- булева різниця;
- еквівалентна нормальна форма [6, 9].

В усіх названих методах утворюється активізований шлях до деякого виходу складного об’єкта. Відрізняється лише спосіб обчислення вхідних сигналів для побудови активізованого шляху.

Найбільш універсальним є ймовірнісний метод, який передбачає підбір тестів на основі псевдовипадкових вхідних послідовності, які спочатку проходять перевірку коректності, а потім повноти або діагностичних властивостей. Процедура закінчується, якщо отримано тест заданої повноти або ефективність нових наборів стає занадто малою.

Ефективність виражається кількістю несправностей, які перевіряються цією послідовністю.

Універсальним способом визначення характеристик тесту є моделювання коректно функціонуючого складного об'єкта S_0 та усіх його модифікацій S_1, S_2, …, S_k, в яких містяться несправності 1, 2, …, k з наступним порівнянням реакції складного об'єкта S_j з реакцією складного об'єкта S_0. Несправність перевіряється тестом, якщо реакції схем S_j та S_0 відрізняються між собою. Несправність вводиться в модель зміною відповідних констант моделі.

В цьому випадку треба моделювати роботу схеми (1 + | F |) разів, де F – множина несправностей, для яких будується тест. Таким чином можна отримати тести як для складного об'єкта комбінаційного типу, так і для складного об'єкта послідовнісного типу. Побудова тестів за допомогою ймовірнісних процедур описана в [13].

За допомогою ймовірнісного методу звичайно отримують тести, повнота яких значно менша за 100 %. Для пошука наборів, що покривають несправності, які не охоплені випадковим тестом, застосовують детерміновані методи синтезу. Це пояснюється тим, що таких несправностей залишається небагато, а використанню детермінованих методів віддають перевагу за синтезу тестів для порівняно нескладних складних об'єктів.

Більшість детермінованих методів базується на таких положеннях:

1. Вхідні набори тесту підбирають почергово для всіх типів несправностей з заданного списку.
2. Для кожної несправності визначають шляхи, які включають складові складного об'єкта і з'єднують несправну складову з іншими складовими складного об'єкта.
3. Для виявлення несправності необхідно забезпечити проходження протилежного за значенням до типу помилки сигналу від

аналізованої складової складного об’єкта до його входів, тобто подати на входи складових, що входять до активного шляху, такі значення сигналів, які дозволять виявити справність функціонування складової.

4. Для виявлення несправностей та активізації шляхів підбирають відповідний вектор вхідних змінних (набір змінних), який і становить черговий вектор у діагностичному тесті.

Найбільш наочно процес побудови тестів подає активізація одновимірних шляхів. Е. Е. Рот [14] формалізував послідовність дій з активізації шляхів у вигляді d-алгоритму, який став одним з фундаментальних методів синтезу тестів.

d-алгоритм синтезує тест для несправності заданого типу, що існує в складному об’єкті. Цей алгоритм являє собою неявну процедуру перебору в процесі обчислення зворотного шляху. Для кожної складової складного об’єкта можна обрати вхідні сигнали, які перевіряють всі її константні несправності. Основна проблема полягає в тому, що ми не можемо безпосередньо подавати сигнали на входи складного об’єкта і спостерігати реакцію на його виході, оскільки він «захований» всередині складного об’єкта. А щоб судити про його реакцію на виходах складного об’єкта, потрібно визначити активізований шлях від складової E_0, що містить несправність, через складові E_1, E_2, …, E_{m-1} до складової E_m, вихід якої є одночасно виходом складного об’єкта.

Значення вихідного сигналу складової E_i має залежати лише від значення вихідного сигналу тільки складових E_{i-1} (E_i та E_{i-1} включено до активного шляху). Інші значення вхідних сигналів складової E_i обираються таким чином, щоб змусити складову E_i «нести повну відповідальність» за значення вихідного сигналу складової E_{i-1}. Узагальнюючи це положення, можна досягти такої ситуації, за якої про значення вихідного сигналу

складової E_0 можна було судити, спостерігаючи за виходами складової E_m. Цей процес носить назву фази просування вперед або прямої фази.

Визначивши активний шлях, необхідно знайти первинний вхідний набір, який забезпечить всі можливі вхідні значення сигналів складових E_0, E_1, …, E_m. Для цього прокладається зворотний шлях за схемою від входів складових E_0, E_1, …, E_m до первинних входів складного об'єкта. Цей процес носить назву оберненої фази або фази просування назад.

Під час активізації одновимірного шляху може випадково активізуватися і інший шлях, який приховує несправності основного активізованого шляху. Такого роду явище може виникати кожен раз, коли два або більше шляхи, що розгалужуються від місця несправності, потім знову сходяться в одній точці, а парність кількості інверсій вздовж шляхів неоднакова. Слабке місце одновимірного методу полягає в тому, що кожен раз дозволяється активізувати лише один шлях, навіть за розгалужень, які сходяться.

Основна ідея алгоритмічного методу полягає в одночасній активізації всіх можливих шляхів від місця несправності до всіх виходів схеми. Ця ідея дозволяє позбутися основної вади одновимірного методу та приводить до d-алгоритму. Спочатку обирається тест для несправності, що розглядається. Далі генеруються всі можливі шляхи від місця несправності до всіх виходів складного об'єкта. На кожному кроці видаляються шляхи, які неможливо активізувати через розгалуження, що сходяться. Цей крок є узагальненням прямої фази одновимірного методу. Нарешті формується первинний набір, який реалізує всі умови, розроблені під час виконання прямої фази.

В процесі підбору значень вхідних сигналів складової можливий випадок, коли невдалий вибір пізніше призведе до несумісності обраних значень сигналів. Якщо це відбудеться, алгоритм має «дати зворотний шлях», змінюючи вибір до тих пір, поки несумісність буде усунена.

d-алгоритм можна використовувати, щоб отримати один тест для кожного типу несправності та об’єднати їх у діагностичний тест. Можна визначити множини поодиноких та кратних несправностей, що виявляються цим тестом, і для кожної складової складного об’єкта обчислити, за яких несправностей змінюється значення її вихідного сигналу.

За відповідної модифікації d-алгоритм дозволяє знаходити тести виявлення декількох несправностей, розрізняти дві несправності, а також виявляти несправності, що відносяться до класу можливих несправностей.

Кейс 6. Розробіть діагностичний тест для виявлення заданого типу помилку для складової складного об’єкта.

4 КОНСТРУКТОРСЬКЕ ПРОЄКТУВАННЯ

Конструкторське проєктування – один з найважливіших рівнів проєктування складних об’єктів. Початковими даними для конструкторського проєктування є результати структурного та функціонального проєктування. Зі свого боку, результати конструкторського проєктування (конструкторська документація) є основою технологічної реалізації.

Для розробки конструкцій складних об’єктів загалом характерним є висхідне проєктування: спочатку розробляються окремі функціональні модулі, блоки та найскладніші об’єкти. Також на кожному з цих рівнів проєктування послідовно розв’язуються такі задачі:

- компонування складових конструкції у вузли заданого ієрархічного рівня;
- розташування складових конструкції;
- трасування з’єднань між складовими.

Розв’язання задач конструкторського проєктування передбачає також виготовлення пакета конструкторської документації.

4.1 Компонування складових складних об’єктів

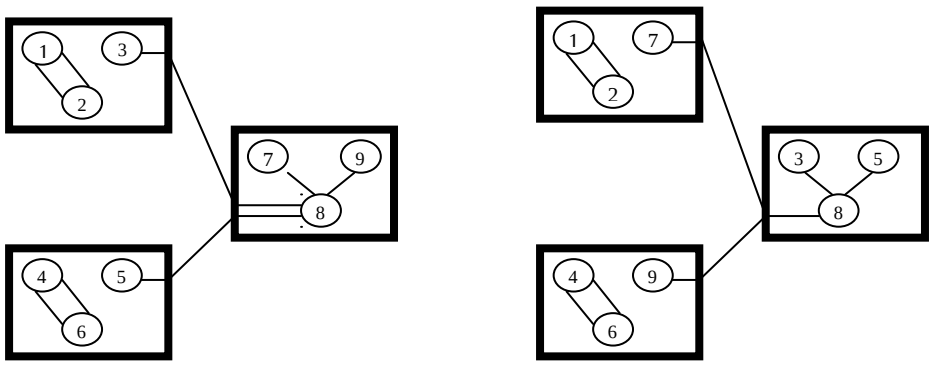
Функціональний поділ складного пристрою має вигляд ієрархічного дерева. Конструктивно пристрій побудовано за модульним принципом, згідно з яким він поділяється на декілька констуктивних одиниць, які також поділяються на конструктивні одиниці більш низького рангу (з вищим рівнем деталізації) і т. д. В процесі компонування визначається однозначна відповідність між функціональним та конструктивним поділом складного об’єкта.

Модуль як мінімальна конструктивна одиниця, яку в процесі експлуатації можна замінити, звичайно складається з декількох функціональних або базових складових.

В процесі реалізації компонування мають виконуватись вимоги:

1. Максимальне використання складових. Водночас найчастіше використовується мінімізація кількості складових m.
2. Типізація, тобто мінімальна кількість різних типів складових.
3. Мінімальна кількість міжвузлових з'єднань f або мінімальна кількість зовнішніх зв'язків всіх складових f^*.

Звичайно вимоги третьої групи є головними, оскільки мінімізація числа зовнішніх зв'язків f веде до скорочення сумарної довжини з'єднань в процесі розв'язання всієї задачі технічного проєктування. Компонування має сприяти полегшенню виконання наступних етапів розташування складових та трасування з'єднань. Зменшенню кількості ланцюгів між складовими різних вузлів за постійної кількості зовнішніх зв'язків f сприяє мінімізація сумарної кількості зовнішніх зв'язків всіх вузлів f^*. Це наочно показано на рис. 4.1, де наведено два різних варіанти компонування дев'яти складових в три вузли. Компонування еквівалентні відносно критерію f, але не еквівалентні відносно критерію f^*. Неважко помітити, що за зменшення f^* має місце тенденція до зменшення f та навпаки.



а) б)

Рисунок 4.1 – Варіанти компонування конструктивних вузлів для а) f = 2, f^* = 4; б) f = 2, f^* = 3

4. Дотримання конструктивних обмежень:

- кількість складових n_q вузла q не може перевищувати граничного значення n:

$$\boldsymbol{n_q \le n}, \tag{4.1}$$

де q = 1, 2, …, m.

- кількість зовнішніх виводів f^*_q вузла q не може перевищувати граничного значення M:

$$\boldsymbol{f^*_q \le M}, \tag{4.2}$$

де q = 1, 2, …, m.

5. Нерозривність функціонального призначення складових.

6. Мінімальна зв’язність складових за назвами, що сприяє полегшенню розташування складових.

Всі відомі наближені алгоритми компонування можна віднести до однієї з двох груп:

1. Алгоритми послідовного компонування складових.
2. Ітераційні алгоритми послідовного поліпшення компонування складових.

Спільним для всіх алгоритмів першої групи є вибір на кожному кроці для компонування одного елемента з максимальним значенням обраної норми S^a. Норма S^a в загальному випадку є функцією від деякого числа факторів:

$$\mathbf{S^a = f\,(S^a_1, S^a_2, \ldots, S^a_n)}. \tag{4.3}$$

В алгоритмах використовують такі фактори.

Фактор кон’юнкції S^a_1, значення якого дорівнює числу з’єднань між досліджуваною складовою a та підмножиною складових A_q, що призначені модулю.

Фактор диз'юнкції S^a_2, значення якого дорівнює числу з'єднань, інцидентних досліджуваній складовій *a*, але не інцидентних підмножині A_q складових, призначених модулю, що формується.

Суть алгоритмів послідовного поліпшення компонування складових полягає в тому, що деякі складові міняються місцями з іншими складовими або переносяться на вільні місця незаповнених вузлів. Перед виконанням ітераційного алгоритму має бути задано початкове компонування, яке визначається довільно або після виконання іншого алгоритму. Зміна складових відбувається для покращення обраного критерію.

4.2 Розташування складових

Розташування – це визначення положення модулів, складових в складному об'єкті, що проєктується. Надалі розглянемо узагальнену задачу розташування складових.

Початковими даними для розташування є:

- схема з'єднань конструктивних складових деякого модуля, отримана за результатами компонування;
- конструктивні параметри складових (обсяг пам'яті, розміри);
- параметри монтажного простору складного об'єкта (якщо це hard-об'єкт).

Найбільш привабливою метою розташування потрібно вважати максимальне зниження спотворень логічних сигналів та максимальне полегшення трасування складових. Можна виділити такі практичні критерії розташування:

- мінімізація сумарної довжини L всіх з'єднань складного об'єкта;
- мінімізація довжини найдовшого з'єднання складного об'єкта M (критерій усереднення довжини);
- мінімізація кількості перетинів з'єднань, що відносяться до різних сигналів;
- максимізація кількості зв'язків з простою конфігурацією.

Всі відомі алгоритми розташування складних об’єктів можна поділити на три групи [6, 8, 10]:

1. Ітераційні алгоритми послідовного поліпшення розташування, які використовують перестановки складових.
2. Алгоритми послідовного розташування складових.
3. Алгоритми, що використовують силові функції.

Ітераційні алгоритми послідовного поліпшення розташування було запропоновано в роботі Глейзера. Початкове розташування складових складного об’єкта задається певним чином, і в подальшому використовуються попарні перестановки з метою зменшення сумарної довжини зв’язків. Існує багато робіт, де запропоновано різні модифікації ітераційного алгоритму. Зокрема в роботі [9] показано можливість застосування циклічних перестановок для додаткового поліпшення розташування, коли попарні перестановки не приводять до бажаного результату. Потрібно зазначити, що ефективність ітераційних алгоритмів значною мірою залежить від початкового розташування, оскільки отриманий розв’язок приводить до локального мінімуму функції, що оптимізується.

Алгоритми послідовного розташування елементів. В цих алгоритмах початкову множину складових складного об’єкта розташовують за певну кількість кроків, меншу за n, оскільки до початку розташування частина складових отримують фіксовані позиції. На кожному кроці обирається чергова нерозташована складова x_i та встановлюється в незайняту позицію p_i. Надалі складова не пересувається.

Алгоритми, що використовують силові функції. Вперше метод силових функцій було запропоновано в роботі Мілє [1] для розташування додаткових верхівок під час розв’язання однієї з модифікацій задачі Штейнера. В процесі реалізації методу між розміщуваними вершинами вводяться сили притягання, пропорційні числу зв’язків і відстані між ними,

і після розв’язання відповідної системи нелінійних диференціальних рівнянь визначаються координати вершин, за яких система знаходиться в рівновазі.

Правила вибору та установлення чергової складової визначають конкретні алгоритми. В найпростішому випадку вибір складової базується на оціненні числа зв’язків a_{ij} складової x_i з розташованими (з множини $\mathbf{X}^p$) та нерозташованими (з множини $\mathbf{X}^н$) складовими:

$$\mathbf{F(x_i) = \sum_{x_j \in X^p} a_{ij} - \sum_{x_j \in X^н} a_{ij}.} \qquad (4.4)$$

З усіх можливих складових вибирається той, у якого значення $F(x_i)$ максимальне. В більш складних випадках функція $F(x_i)$ враховує параметри та характеристики модулів. За наявності декількох складових з однаковим значенням функції $F(x_i)$ застосовують правила «уточнення вибору»: випереджувальний перегляд, обчислення вторинної функції або довільний вибір.

Найпростіше правило установлення використовує оцінку сумарної довжини зв’язків $d(x_i, x_j)$ складової x_i з розташованими складовими з множини $\mathbf{X}^p$:

$$\mathbf{F(p_i) = \sum_{x_j \in X^p} a_{ij}\, d(x_i, x_j)} \qquad (4.5)$$

Вибирається саме та з позицій p_i, для якої значення $F(p_i)$ мінімальне.

Послідовні алгоритми приводять до менших витрат часу та пам’яті ЕОМ, ніж ітераційні.

4.3 Трасування складових складних об’єктів

Початковими даними для задачі трасування є параметри та характеристики складного об’єкта з розташованими складовими. Під час розв’язання задачі трасування потрібно з’єднати між собою всі складові відповідно до спроєктованої структури складного об’єкта. Розглянемо задачу трасування на прикладі проєктування складного об’єкта hard-типу.

Функція якості трасування (критерій оптимальності) в такому випадку має враховувати такі параметри:

- сумарну довжину зв’язків;
- кількість кутів в трасі.

В процесі складання списку зв’язків визначається, які складові мають бути з’єднаними, тобто формуються списки координат контактів складових, які підлягають з’єднанню.

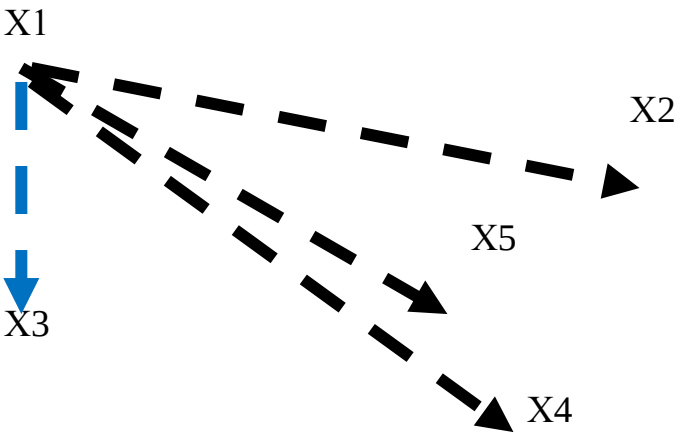
Як проводити трасу чергового зв’язку визначає алгоритм прокладання трас.

На першому етапі необхідно вирішити, в якій послідовності потрібно з’єднувати складові одного ланцюга, щоб сумарна довжина всіх зв’язків була мінімальною. Ця задача зводиться до задачі побудови мінімального зв’язувального дерева.

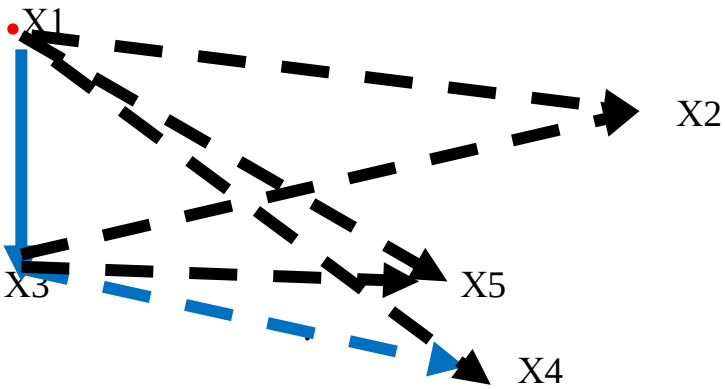
Найбільш поширений розв’язок цієї задачі базується на алгоритмі Прима.

На першому кроці алгоритму для довільного контакту знаходиться найближча вершина та з’єднується з ним ребром. На наступних (n-2) кроках з множини непідключених контактів вибирається той, що знаходиться найближче до групи вже зв’язаних контактів та з’єднується ребром. На рис. 4.2 показано фрагмент дерева (x_1, x_2, x_3, x_4, x_5) після четвертого кроку алгоритму, формується траса, що має мінімальну відстань d_{ij} серед всіх можливих. Водночас, перелік кроків формування

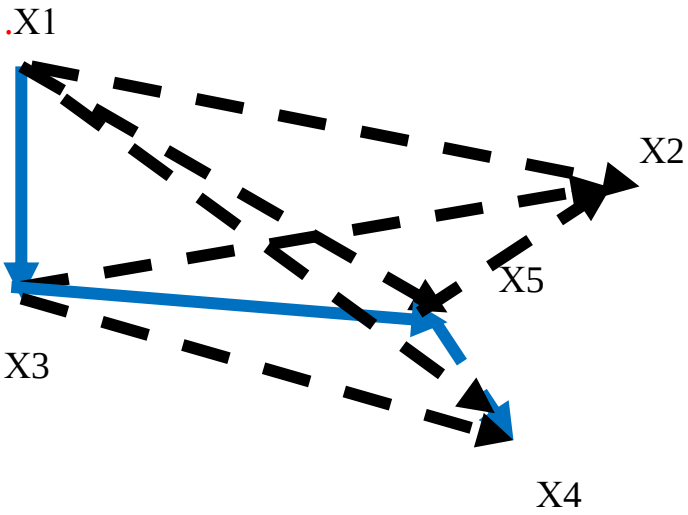
траси буде таким (рис. 4.2). Розглянемо відстані від початкової точки X1: [(x_1, x_2) (x_2, x_3) (x_2, x_4) (x_4, x_5)] з метою обрання мінімальної з них.



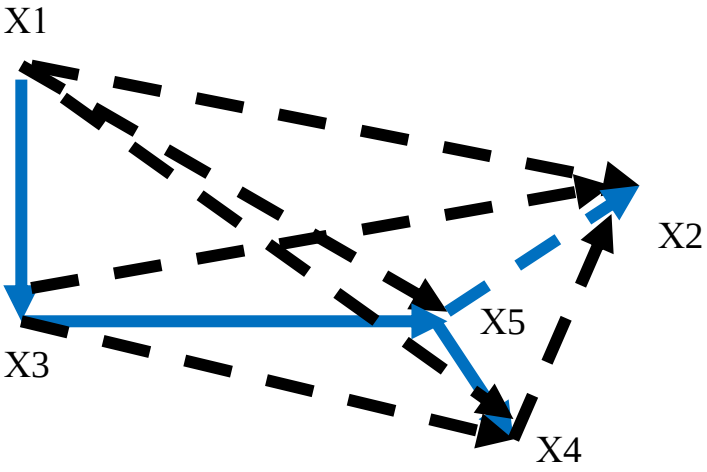
min(d(X1,X2), d(X1,X3), d(X1,X4), d(X1,X5)) = d(X1,X3)



min(d(X1,X2), d(X1,X4), d(X1,X5), d(X3,X2), d(X3,X4), d(X3,X5)) = d(X3,X5)



min(d(X1,X2), d(X1,X4), d(X1,X5), d(X3,X2), d(X3,X4), d(X3,X5), d(X4,X2), d(X4,X5)) = d(X5,X4)



min(d(X1,X2), d(X1,X4), d(X1,X5), d(X3,X2), d(X3,X4), d(X3,X5), d(X4,X2), d(X4,X5), d(X5,X2)) = d(X4,X2)

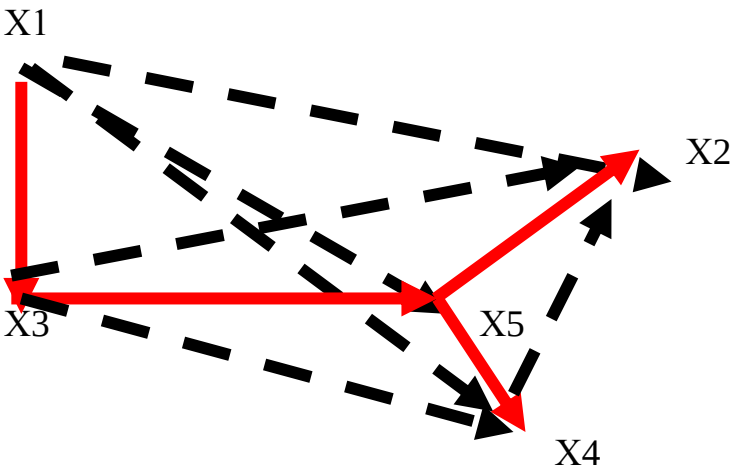


Рисунок 4.2 – Покрокова побудова траси за алгоритмом Прима

Якщо m=2 – задача зводиться до відомої теоретичної задачі про комівояжера.

Власне трасування зв’язків полягає в послідовній побудові трас з врахуванням заданих вимог та обмежень. Існуючі алгоритми проведення трас можна умовно поділити на групи, що базуються на ідеях хвильового та евристичного алгоритмів.

Хвильовий алгоритм або алгоритм Лі визначає шлях між двома комірками **A** та **B** дискретної решітки. Для оцінення якості шляху задається багатовимірна вагова функція $F = (f_1, f_2, \ldots, f_r)$, де f_i характеризує деякий параметр шляху (довжину, кількість зв’язків,

кількість кутів в трасі тощо). Під час визначення шляху всім коміркам приписують масу $M = (m_1, m_2, \ldots, m_r)$. Вводиться поняття заборонених комірок, тобто таких, які не можуть бути внесені до шляху.

В алгоритмі чітко виділяються дві частини: поширення хвилі та формування траси.

Під час поширення хвилі, починаючи з комірки **A**, коміркам, сусіднім до вже розглянутих, крім заборонених, приписують значення маси, що відрізняється на одну умовну одиницю маси, тобто формується фронт нової хвилі. У цьому випадку хвиля маси поширюється під кутом, кратним $\pi/2$. Ця процедура триває до тих пір, поки всім коміркам, зокрема комірці **B,** не буде присвоєно значення маси. На зворотному шляху від комірки **B** до **A** зі зменшуваними значеннями маси на одну умовну одиницю маси визначається шлях між **A** та **B** з мінімальним значенням функції F.

Для спрощення процедури проведення шляху під час поширення хвилі коміркам надаються шляхові координати.

В роботі алгоритму Лі прийнято квадратну форму комірок і сусідніми вважаються квадрати з спільним ребром. Роботу алгоритму Лі ілюструє приклад, наведений на рис. 4.3, де наведено різні варіанти сформованих трас та обрано одну з них (виділено жирною лінією).

Алгоритми Лі визнають найуніверсальнішими серед локально-оптимальних алгоритмів трасування. Їх недоліки – великий обсяг обчислень та необхідність використання великих обсягів пам’яті. З метою скорочення витрат пам’яті та машинного часу запропоновано багато різних модифікацій хвильових алгоритмів.

Скорочення витрат пам’яті та часу ЕОМ можна досягти, застосовуючи променеві алгоритми, запропоновані Абрайтісом.

Основна ідея променевих алгоритмів полягає в дослідженні не всіх комірок, а тільки частини з них по деяких заданих напрямках [6].

Розглянемо двопроменевий варіант алгоритму. В цьому варіанті від кожного контакту x_i та x_j поширюється по два промені та кожний новий фронт містить не більше чотирьох елементів.

Поширення кожного променя припиняється, якщо всі сусідні квадрати, що обираються з числа можливих, зайняті або заборонені (блокування променя). Якщо трасу ще не прокладено, вектори пересувань в обох напрямах змінюють свої напрями на протилежні, а робота алгоритму продовжується. Трасу прокладено, якщо промені від різних вершин перетинаються.

Нехай є точки А та В дискретної решітки, що мають бути внесеними до траси з використанням променевого алгоритму, як показано на рис. 4.4.

В променевому алгоритмі Абрайтиса виділяються два етапи:

1. Визначення пріоритетних напрямів пересувань у дискретній площині з кожної з точок, що з'єднуються: вверх чи вниз / вліво чи вправо. Черговість напрямів променів по горизонталі чи вертикалі не є принциповою та передбачається алгоритмом комп'ютерного трасування. Напрям прокладання променя визначається на підставі аналізу координат точок, що підлягають з'єднанню. Наприклад, для завдання, що подано на рисунку 4.5, за умови, що спочатку обираємо прокладання променя по вертикалі, матимемо А → В: вниз – вправо; В → А: вверх – вліво.
2. Прокладання траси за допомогою визначених векторів пересувань у дискретній площині з кожної з точок, що з'єднуються: N догори чи вниз / L вліво чи вправо. N, L – коефіцієнт, що визначає максимальну кількість дискретних комірок, що можуть бути пройденими за один крок, є цілим додатнім числом. Може мати значення від 1 до максимальної кількості дискретних комірок за певним виміром у дискретній площині.

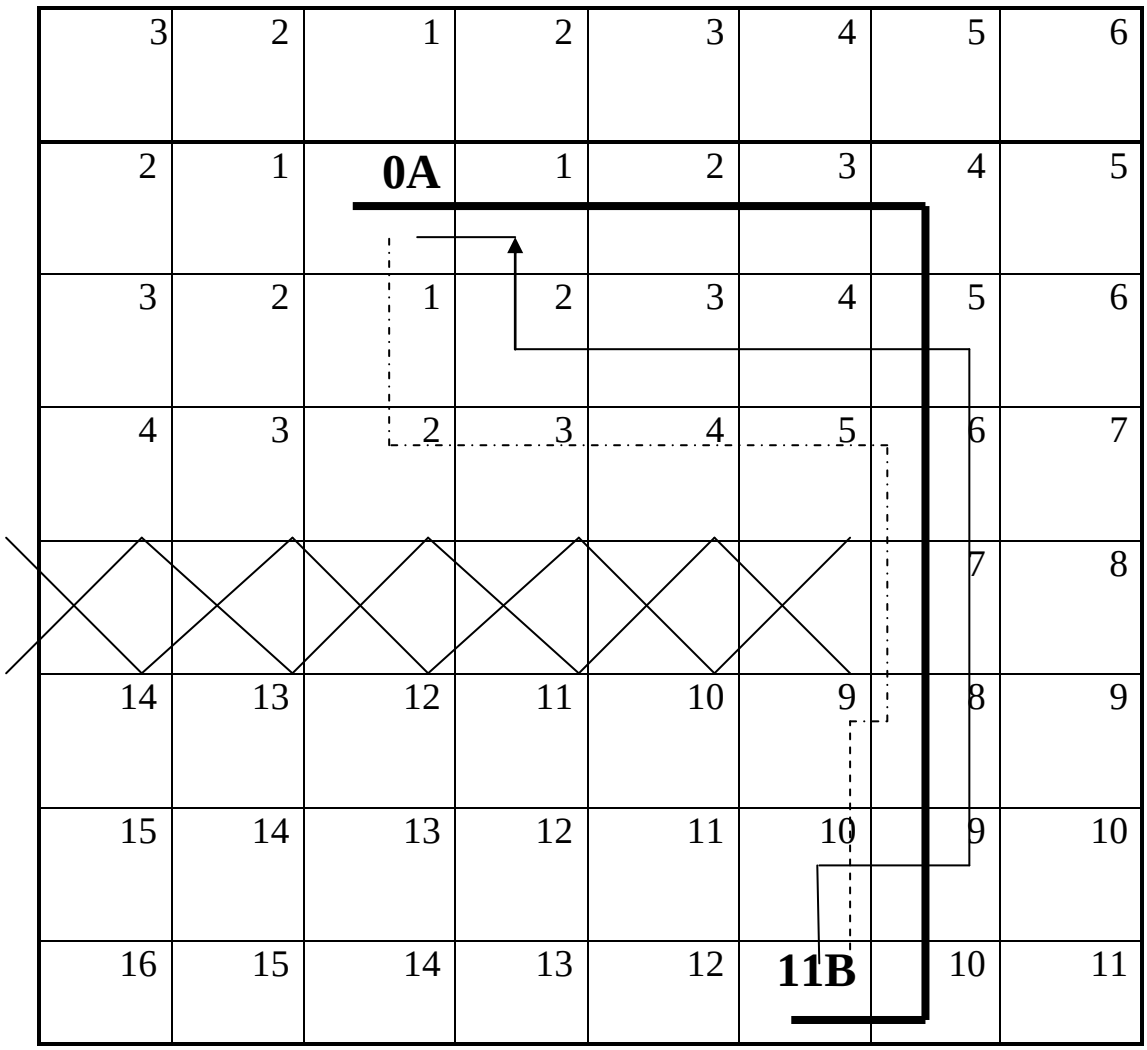


Рисунок 4.3 – Приклад застосування хвильового алгоритму трасування

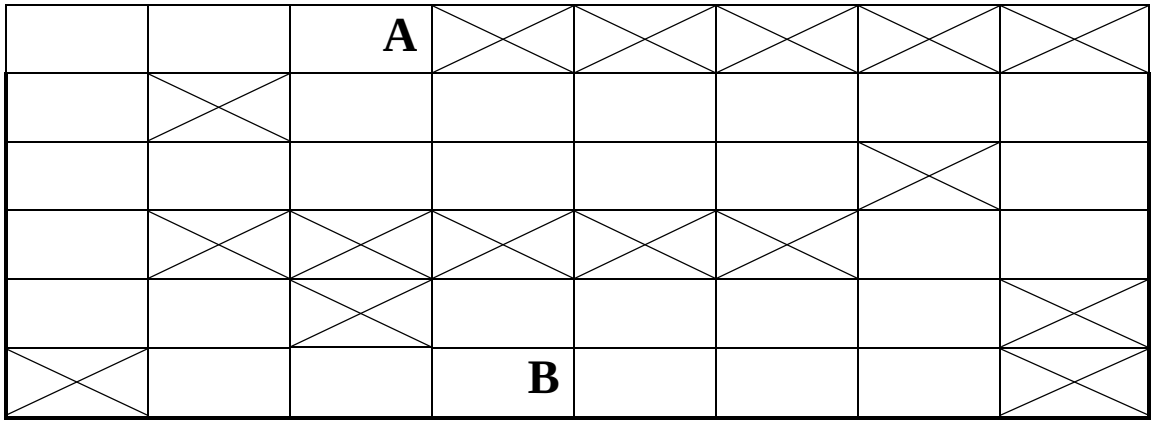


Рисунок 4.4 – Завдання для побудови траси

Водночас, кожен крок, прокладений із кожної з точок, позначають як Z(v), де Z – ім′я точки, з якої виходить промінь; v – номер кроку. Якщо в поточному кроці за одним із напрямів промінь прокладено або неможливе прокладання променя в цьому напрямі, переходять до наступного напряму в цьому кроці. На кожному кроці промені прокладаються з кожної точки спочатку за першим вказаним напрямом, далі – за другим.

Шлях знайдено, якщо промені перетнулись. Шлях траси визначають прокладені промені.

Тоді, для означеного прикладу, отримаємо:

Етап 1.

A → B: вниз – вправо;

B → A: вверх – вліво.

Етап 2.

A → B: 3 вниз – 6 вправо;

B → A: 5 вверх – 4 вліво.

Пріоритетні напрями пересувань виділено жирною лінією. Після прокладання у першому кроці променя з точки B, за другим напрямом у цьому кроці прокладання променя є неможливим. На другому кроці прокладання променя з точки B, напрями вектора пересувань змінено на протилежні, оскільки за раніше визначеними напрямами вектора пересувань з точки B промені не можуть бути прокладеними. Отож, вектор пересувань B → A зміниться за напрямами, а саме : 5 вниз – 4 вправо. Після виконання 2-го кроку алгоритму з обох точок прокладання променів, промені перетнуться, а, отже, траса, побудована із застосуванням раніше визначених векторів пересувань за алгоритмом Абрайтиса, матиме вигляд, поданий на рисунку 4.5.

Звичайно за допомогою променевого алгоритму вдається провести 80 % трас, а більш складні конфігурації, нерозведені траси прокладають за допомогою хвильового алгоритму або вручну. Для реалізації всіх

можливих променевих алгоритмів вводять спеціальні процедури виходу променів з тупикових ситуацій.

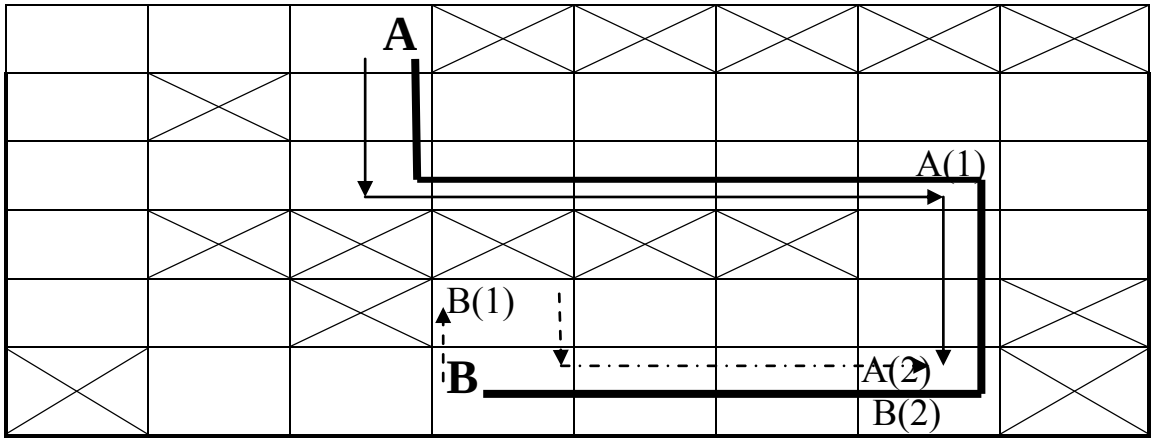


Рисунок 4.5 – Приклад застосування променевого алгоритму трасування

Евристичні алгоритми трасування з’єднань є більш швидкодійними. На відміну від хвильового алгоритму в них не аналізуються всі можливі траси для вибору оптимальної, а одразу намагаються прокласти трасу по найкоротшому шляху. Якщо на шляху зустрічаються перешкоди, то обминання здійснюється по першому вільному напрямку або згідно з правилами, що визначають порядок обминання перешкод. До початку роботи алгоритму задається пріоритетний порядок обминання перешкод. Такі алгоритми складно формалізуються.

5 UML – ДІАГРАМИ В ПРОЦЕСІ ПРОЄКТУВАННЯ СКЛАДНИХ ОБ'ЄКТІВ

Серед процедур структурного синтезу виділяють синтез технічної документації (див. підрозділ 2.2) як автоматичне перетворення даних про складні об'єкти та їх складові, що знаходяться у внутрішньому форматі СКПР, на текстову та креслярську документацію, оформлену згідно з правилами відповідних стандартів щодо конструкторської документації.

З цією метою використовують уніфіковану мову моделювання (UnifiedModellingLanguage або UML) у парадигмі об'єктно-орієнтованого програмування як мову позначень для створення абстрактної моделі складного об'єкта для визначення, проєктування, документування моделей, зорієнтованих на об'єкти систем програмного забезпечення. Особливо потрібно відзначити мову UML як базову нотацію візуального моделювання, що реалізується у CASE-засобах. Потрібно зауважити, що UML, як загальноприйнятий стандарт опису результатів проєктування складних об'єктів та їх складових, полегшує співпрацю з іншими її розробниками, але не виступає методом розробки.

UML забезпечує підтримку всіх етапів життєвого циклу складного об'єкта та його складових та надає для цих цілей ряд графічних засобів – діаграм.

До основних типів UML-діаграм, що використовуються для подання результатів проєктування складних об'єктів, найбільшу популярність здобули такі.

У разі створення концептуальної моделі складного об'єкта для опису бізнес-діяльності використовуються моделі бізнес-прецедентів та діаграми видів діяльності, а для опису бізнес-об'єктів –моделі бізнес-об'єктів та діаграми послідовності.

У разі створення логічної моделі складного об'єкта опис вимог задається з використанням діаграм прецедентів. На рівні функціонального

проєктування зазвичай використовують діаграми класів, діаграми послідовності та діаграми станів.

У разі створення фізичної моделі детальне проєктування виконується з використанням діаграм компонентів, діаграм розгортування.

UML-діаграми розділено на групи за різними ознаками:

- діаграми, що подають структуру складного об’єкта;
- діаграми, що подають поведінкові аспекти складного об’єкта;
- діаграми, що подають фізичні аспекти складного об’єкта.

Взаємодію різних видів UML-діаграм подано на рисунку 5.1.

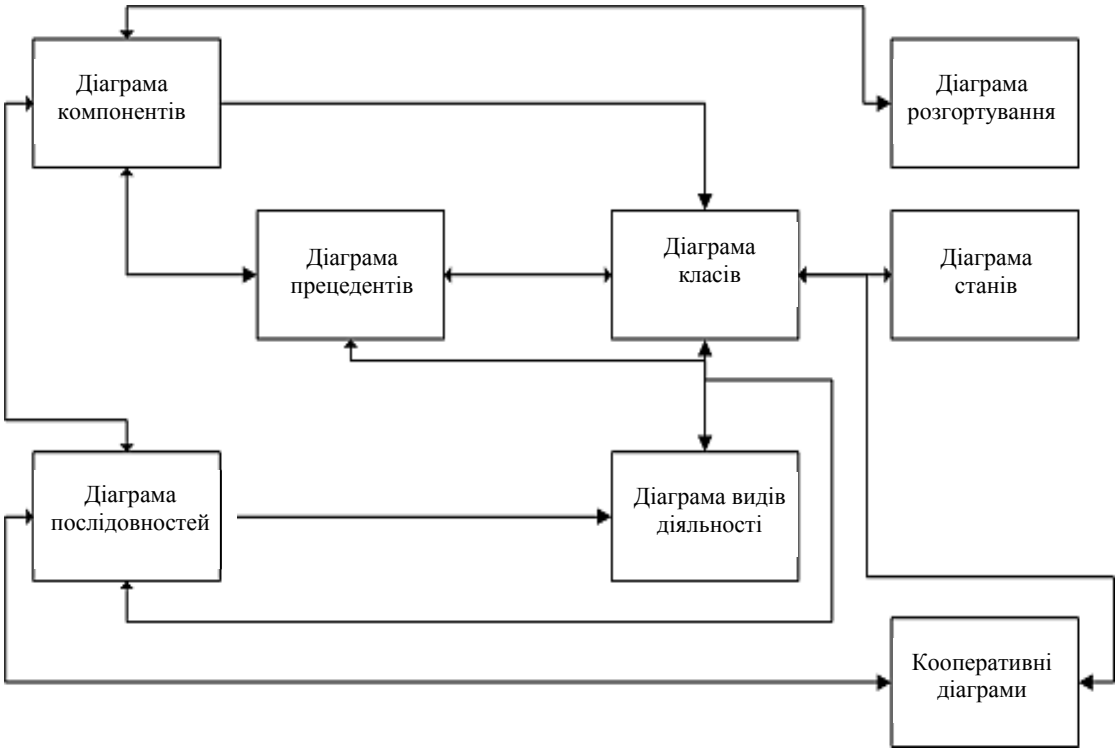


Рисунок 5.1 – Взаємодія різних видів UML-діаграм

Розглянемо основні типи UML-діаграм, що використовуються в процесі подання результатів проєктування складних об’єктів.

5.1 UML-діаграма прецедентів/варіантів використання (use case diagram)

UML-діграма прецедентів (діаграми варіантів використання, use case diagram) − це узагальнена модель функціонування системи, що подає динамічні або поведінкові аспекти складного об’єкта. Суть цієї діаграми полягає в тому, що проєктований складний об’єкт подається у вигляді множини сутностей або акторів, що взаємодіють із складним об’єктом за допомогою так званих варіантів використання. Актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє із системою ззовні. Це може бути людина, технічний пристрій, програма або будь-який інший складний об’єкт, який може бути джерелом впливу на проєктований складний об’єкт так, як визначить сам розробник. Варіант використання (use case) слугує для опису сервісів, які складний об’єкт надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який здійснюється складним об’єктом під час діалогу з актором. Водночас, не йдеться про те, яким чином буде реалізовано взаємодію акторів із складним об’єктом [15].

Складовими діаграми такого типу є виконавець та прецедент.

Виконавець (дійова особа, Actor) – множина логічно пов'язаних ролей, виконуваних під час взаємодії з прецедентами або сутностями (складний об’єкт, система, підсистема або клас).

Розрізняють зовнішніх та внутрішніх виконавців.

Зовнішній виконавець використовує або використовується складним об’єктом, тобто породжує прецеденти діяльності.

Внутрішній виконавець забезпечує реалізацію прецедентів діяльності всередині складного об’єкта.

Прецедент – опис множини послідовних подій (включно й варіанти), що виконуються складним об’єктом, які приводять до спостережуваного Актором результату, і являє поведінку сутності, описуючи взаємодію між Актором і складним об’єктом. Прецедент не показує, як досягається певний результат, а тільки зазначає, які саме дії виконуються.

Основними цілями створення діаграм прецедентів є:

- визначення границь і контексту моделі предметної області на структурному рівні проєктування;
- формування загальних вимог до поведінки проєктованого складного об’єкта;

– розробка концептуальної моделі складного об’єкта для її подальшої деталізації;
– підготовка документації для взаємодії замовників і користувачів складного об’єкта.

Інтерфейс (interface) слугує для специфікації параметрів моделі складного об’єкта, які видно ззовні без зазначення внутрішньої структури. Мовою UML інтерфейс є класифікатором і характеризує лише обмежену частину поведінки сутності, що моделюється.

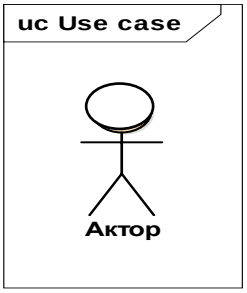
Стосовно діаграм прецедентів / варіантів використання, інтерфейси визначають сукупність операцій, які забезпечують необхідний набір сервісів чи функціональності для Акторів.

Формально інтерфейс еквівалентний абстрактному класу без атрибутів та методів з наявністю лише абстрактних операцій.

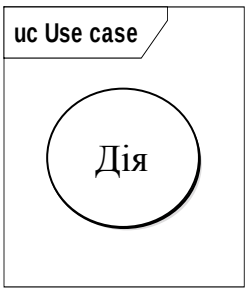
До діаграми вносять прецеденти, що задовольняють такі критерії:
– прецедент має описувати ЩО треба робити, а не ЯК;
– прецедент має описувати дії з погляду виконавця;
– прецедент має повертати виконавцю повідомлення;
– послідовність дій всередині прецеденту має бути неподільною.

В UML–діаграмах послідовності використовують такі позначення:

Виконавець (Актор)



Прецедент



Приклад UML-діаграми прецедентів медичного обслуговування пацієнта, що реалізується через множину інших, більш обмежених прецедентів, які відображають деталізацію подання функціонування центру, наведено на рисунку 5.2.

Між компонентами діаграми прецедентів/варіантів використання можуть існувати різні відношення, які описують взаємодію екземплярів одних Акторів та прецедентів/варіантів використання з екземплярами інших Акторів та варіантів. Один актор може взаємодіяти з кількома прецедентами/варіантами використання, тобто звертатись до кількох сервісів складного об’єкта. Зі свого боку, один прецедент/варіант використання може взаємодіяти з декількома Акторами, надаючи для них свій сервіс.

Потрібно зауважити, що два варіанти використання, визначені для однієї і тієї самої сутності, не можуть взаємодіяти один з одним, оскільки кожен із них самостійно описує закінчений варіант використання цієї сутності. Більш того, варіанти використання завжди передбачають деякі сигнали або повідомлення, коли взаємодіють із Акторами за межами складного об’єкта. У той самий час можуть бути визначені інші способи взаємодії з елементами всередині складного об’єкта.

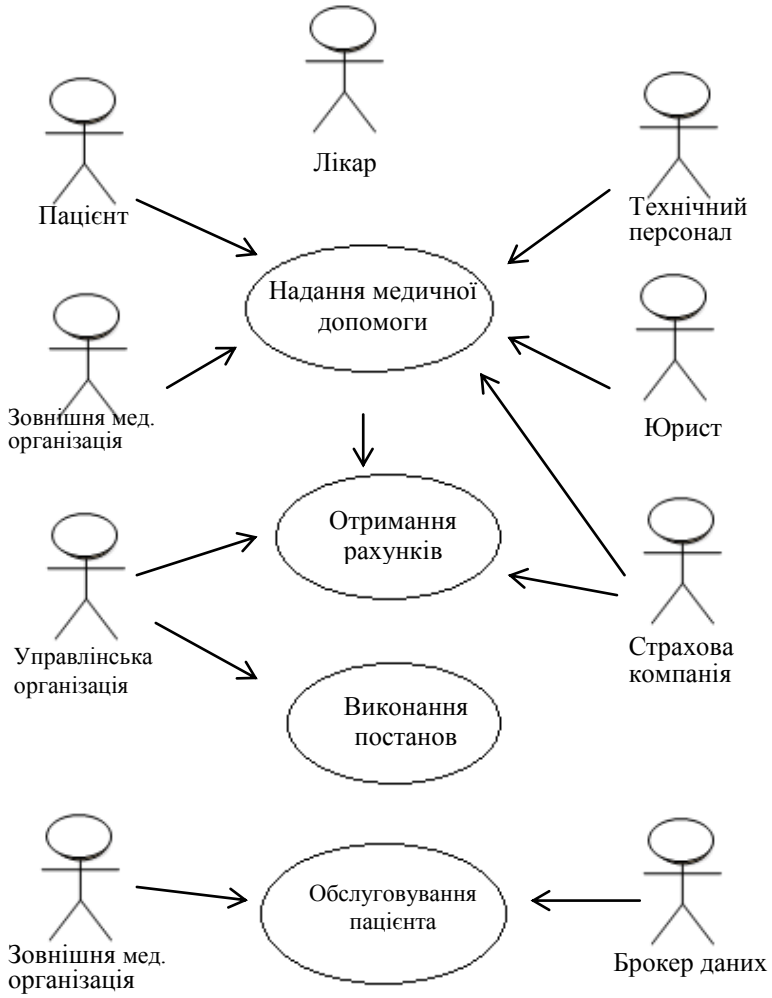


Рисунок 5.2 – UML-діаграма прецедентів медичного обслуговування пацієнта

У мові UML є кілька стандартних типів відношень між Акторами та прецедентами/варіантами використання, кожен з яких має особливі позначення:

- відношення асоціації (association relationship) – позначається неперервною лінією між актором та прецедентом / варіантом використання, що може мати додаткові умовні позначення, як ім’я й кратність ;
- відношення розширення (extend relationship) − позначається пунктирною лінією зі стрілкою (варіант відношення залежності) з ключовим словом «extend» (розширює), спрямованою від варіанта використання, який є розширенням для вихідного варіанта використання;
- відношення узагальнення (generalization relationship) – позначається неперервною лінією зі стрілкою у формі незафарбованого трикутника, що вказує на батьківський варіант використання. Ця лінія зі стрілкою має спеціальну назву – стрілка узагальнення;
- відношення включення (includerelationship) – позначається пунктирною лінією зі стрілкою (варіант відношення залежності) з ключовим словом «include» («включає»), спрямованою від базового варіанта використання до включеного.

Приклад UML-діаграми прецедентів організації освітнього процесу наведено на рисунку 5.3.

5.2 UML–діаграма взаємодії (interaction diagram)

Цей тип UML-діаграм, що подають поведінкові аспекти складного об’єкта, показує потік повідомлень між складовими складного об’єкта й основними асоціаціями між ними. Є альтернативою діаграми послідовності [16].

Особливою відмінністю такого типу діаграм є те, що зазначається не лише послідовність дій, але й їх черговість. Це має важливе значення в процесі реалізації алгоритмів, орієнтованих на різні сценарії функціонування.

Приклад UML-діаграми взаємодії для предметної області «Бібліотека» наведено на рисунку 5.4.

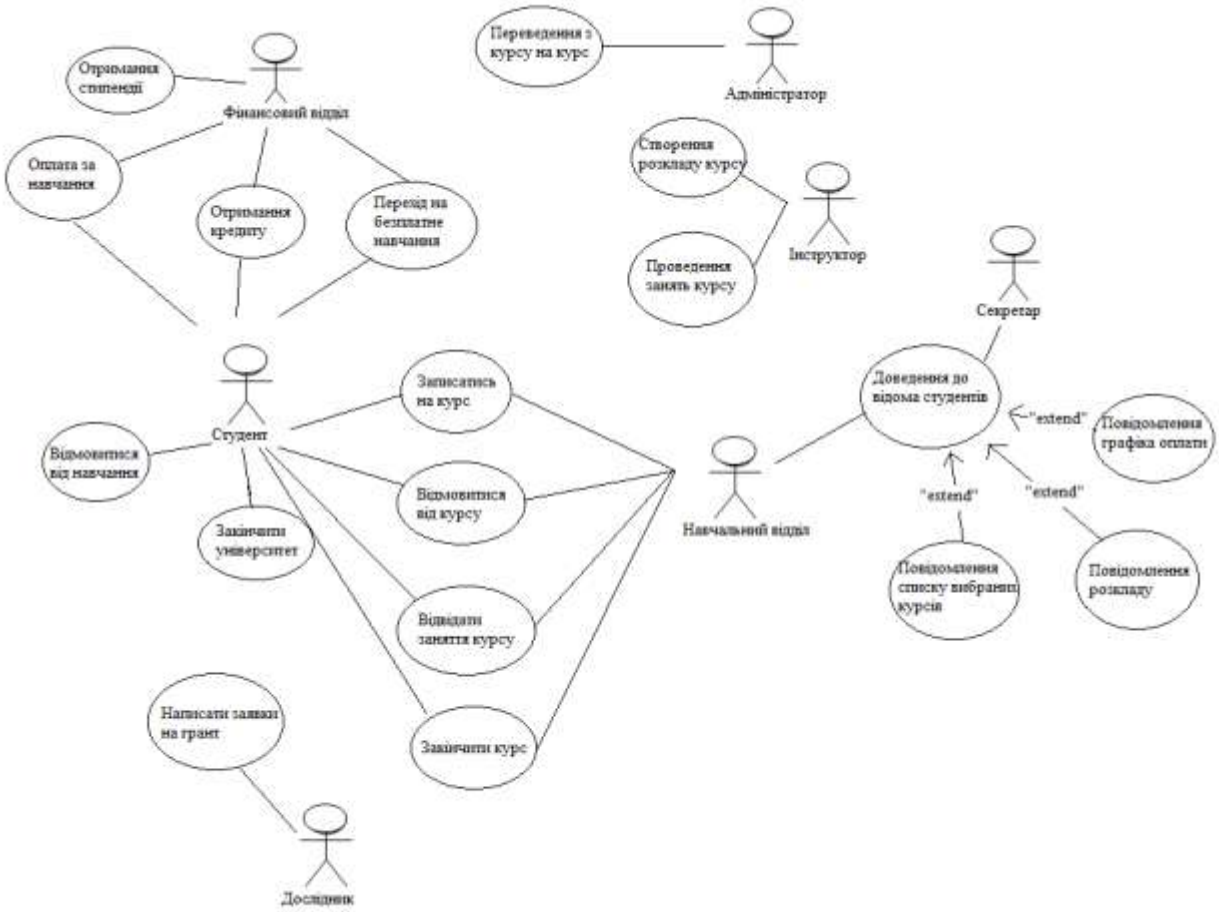


Рисунок 5.3 – UML-діаграма прецедентів організації освітнього процесу

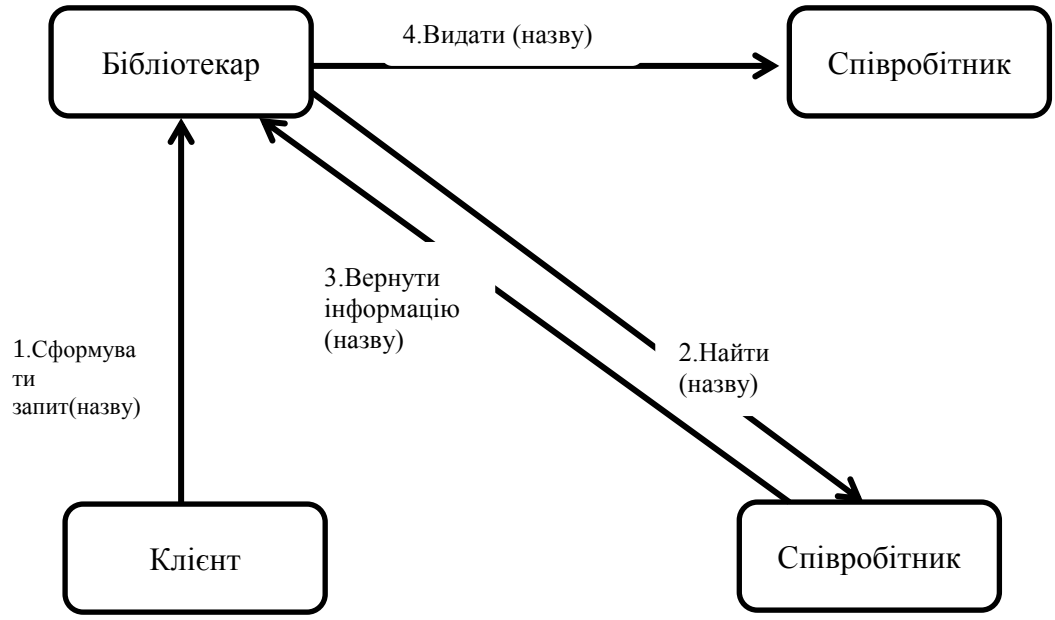


Рисунок 5.4 – UML-діаграма взаємодії для предметної області «Бібліотека»

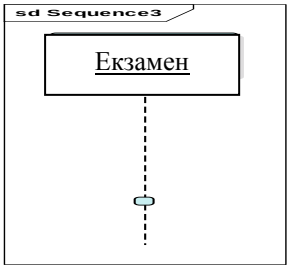
5.3 UML-діаграма послідовності (sequence diagram)

UML-діаграма послідовності (sequence diagram) не тільки відноситься до діаграм взаємодії, що описують поведінкові аспекти складного об’єкта, а ще й розглядає взаємодію об'єктів в часі.

Цей тип діаграм дозволяє відобразити тимчасові особливості передачі і прийому повідомлень об'єктами, а тому можна (і потрібно!) використовувати їх для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання тощо.

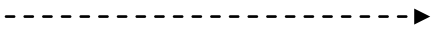
В UML-діаграмах послідовності використовують такі позначення [16, 17]:

– Об'єкти позначаються прямокутниками (з підкресленими іменами щоб відрізнити їх від класів);



– повідомлення (запити методів) – суцільні лінії зі стрілками;

– результати – пунктирні лінії зі стрілками;



– прямокутники на вертикальних пунктирних лініях під кожним з об'єктів – «час життя» (фокус) об'єктів.

Під час побудови UML-діаграми послідовностей крайнім ліворуч на діаграмі зображається об'єкт, який є ініціатором взаємодії. Правіше

зображається інший об'єкт, який безпосередньо взаємодіє з першим. Таким чином, всі об'єкти на діаграмі послідовності утворюють певний порядок, який визначається ступенем активності цих об'єктів в процесі взаємодії один з одним. Другий вимір діаграми послідовності – вертикальна тимчасова вісь, спрямована зверху донизу. Початковому моменту часу відповідає найвища частина діаграми. У цьому випадку взаємодії об'єктів реалізуються за допомогою повідомлень, які надсилаються одними об'єктами іншим. Повідомлення зображаються у вигляді горизонтальних стрілок з ім'ям повідомлення і утворюють порядок за часом свого виникнення. Таким чином, повідомлення, розташовані на діаграмі послідовності вище, ініціюються раніше тих, що розташовані нижче. Водночас масштаб на осі часу не вказується, оскільки діаграма послідовності моделює лише тимчасову впорядкованість взаємодій типу «раніше-пізніше». Лінія життя об'єкта (object lifeline) зображається пунктирною вертикальною лінією, що асоціюється з єдиним об'єктом на діаграмі послідовності. Лінія життя слугує для позначення періоду часу, протягом якого об'єкт існує у системі і, отже, може потенційно брати участь у її взаємодіях. Якщо об'єкт існує в системі постійно, то і його лінія життя має продовжуватися по всій площині діаграми послідовності від верхньої її частини до нижньої.

Окремі об'єкти, виконавши свою роль системі, можуть бути видалені, щоб звільнити займані ними ресурси. Для таких об'єктів лінія життя обривається на момент його видалення. Для позначення моменту видалення об'єкта в UML використовується спеціальний символ у формі латинської літери «X». Нижче цього символу пунктирна лінія не зображається, оскільки відповідного об'єкта в складному об’єкт вже немає, і цей об'єкт має бути виключений з наступних взаємодій.

Не обов'язково створювати всі об'єкти в початковий момент часу. Окремі об'єкти у системі можуть створюватися за необхідності. В цьому

випадку прямокутник такого об'єкта зображається не у верхній частині діаграми послідовності, а в тій її частині, яка відповідає моменту створення об'єкта. У цьому випадку прямокутник об'єкта вертикально розташовується в тій частині діаграми, яка по осі часу збігається з моментом його виникнення в системі. Отже, об'єкт обов'язково створюється зі своєю лінією життя та, можливо, з фокусом управління. У процесі функціонування об'єктно-орієнтованих систем одні об'єкти можуть перебувати у активному стані, безпосередньо виконуючи певні дії, чи стані пасивного очікування повідомлень з інших об'єктів. Щоб явно виділити подібну активність об'єктів, у мові UML застосовується спеціальне поняття, яке отримало назву *фокус управління* (focus of control).

Фокус управління зображається у формі витягнутого вузького прямокутника, верхня сторона якого позначає початок отримання фокусу управління об'єкта (початок активності), а її нижня сторона – закінчення фокусу управління (закінчення активності). Цей прямокутник розташовується нижче за позначення відповідного об'єкта і може замінювати його лінію життя, якщо протягом його життя він є активним. З іншого боку, періоди активності об'єкта можуть чергуватись з періодами його пасивності чи очікування. У цьому випадку такий об'єкт має кілька фокусів управління. Важливо розуміти, що одержати фокус управління може лише існуючий об'єкт, який у цей має лінію життя. Якщо ж певний об'єкт було видалено, то знову виникнути у системі він не може. Замість нього лише може бути створений інший екземпляр цього класу, який буде вже іншим об'єктом.

В окремих випадках ініціатором взаємодії в системі може бути актор чи зовнішній користувач. Тоді актор зображається на діаграмі послідовності першим об'єктом ліворуч зі своїм фокусом управління. Найчастіше актор і його фокус управління існуватимуть у системі постійно, відзначаючи характерну для користувача активність в

ініціюванні взаємодій із системою. Сам актор може мати своє ім'я або залишатися анонімним.

Кожна взаємодія описується сукупністю повідомлень, якими об'єкти, що беруть участь у ній, обмінюються між собою. У цьому сенсі повідомлення (message) є закінченим фрагментом інформації, який відправляється одним об'єктом іншому. У цьому випадку прийом повідомлення ініціює виконання певних дій, спрямованих на вирішення окремого завдання об'єктом, якому це повідомлення відправлено. На діаграмі послідовності всі повідомлення впорядковано за часом свого виникнення в модельованій системі. У такому контексті кожне повідомлення має напрям від об'єкта, який ініціює та надсилає повідомлення, до об'єкта, який його отримує. Зазвичай, повідомлення зображаються горизонтальними стрілками, що з'єднують лінії життя або фокуси керування двох об'єктів на діаграмі послідовності. Таким чином, у мові UML кожне повідомлення асоціюється з деякою дією, яка має бути виконана об'єктом, що його прийняв. Така дія може мати деякі аргументи чи параметри, залежно від конкретних значень яких можна отримати різний результат. Відповідні параметри буде мати і повідомлення, що викликає цю дію. Більше того, значення параметрів окремих повідомлень можуть містити умовні вирази, утворюючи розгалуження або альтернативні шляхи основного потоку керування.

Побудову діаграми послідовності доцільно починати з виділення з усієї сукупності тих і тільки тих класів, об'єкти яких беруть участь у взаємодії, що моделюється. Після цього всі об'єкти наносяться на діаграму із дотриманням порядку ініціалізації повідомлень. Необхідно встановити, які об'єкти існуватимуть постійно, а які тимчасово – лише на період виконання ними необхідних дій.

Коли об'єкти візуалізовано, можна приступати до специфікації повідомлень. Потрібно враховувати ті ролі, які відіграють повідомлення у

системі. За необхідності уточнення цих ролей потрібно використовувати їх різновиди та стереотипи. Для видалення об'єктів, що створюються на час виконання своїх дій, необхідно передбачити явне повідомлення.

Прості випадки розгалуження процесу взаємодії можна зобразити на одній діаграмі з використанням відповідних графічних примітивів. Кожний альтернативний потік управління може ускладнити розуміння побудованої моделі. Тому загальним правилом є візуалізація кожного потоку управління окремою діаграмою послідовності. У цій ситуації такі окремі діаграми потрібно розглядати спільно як одну модель взаємодії.

Подальша деталізація діаграми послідовності виконується за необхідності висвітлення окремих процесів у системі.

Розглянемо приклад побудови діаграми послідовності для процесу отримання студентом стипендії. Нехай алгоритм процесу отримання студентом стипендії міститиме кроки:

1. Отримати заліковку студента.
2. У заліковці виставлено всі бали за поточний семестр. Якщо у заліковці виставлено всі бали за поточний семестр, то продовжується алгоритм. Якщо ні – студент не отримає стипендію.
3. Порахувати бали студента.
4. Порахувати середній бал студента.
5. Порахувати додаткові бали.
6. Порахувати рейтинг студента.
7. Внести рейтинг студента до рейтингового списку студентів.
8. Якщо студент проходить поріг для отримання стипендії, то переходимо до визначення стипендії. Якщо студент не пройшов поріг для отримання стипендії – студент не отримає стипендію.
9. Якщо у студента в заліковці за поточний семестр усі бали більше або дорівнюють 90, то студент отримує підвищену стипендію. Якщо у

студента в заліковці за поточний семестр не усі бали більше або дорівнюють 90, то студент отримує звичайну стипендію.

10. Внести студента до списку стипендіатів.

UML-діаграму послідовності для процесу отримання студентом стипендії наведено на рисунку 5.5.

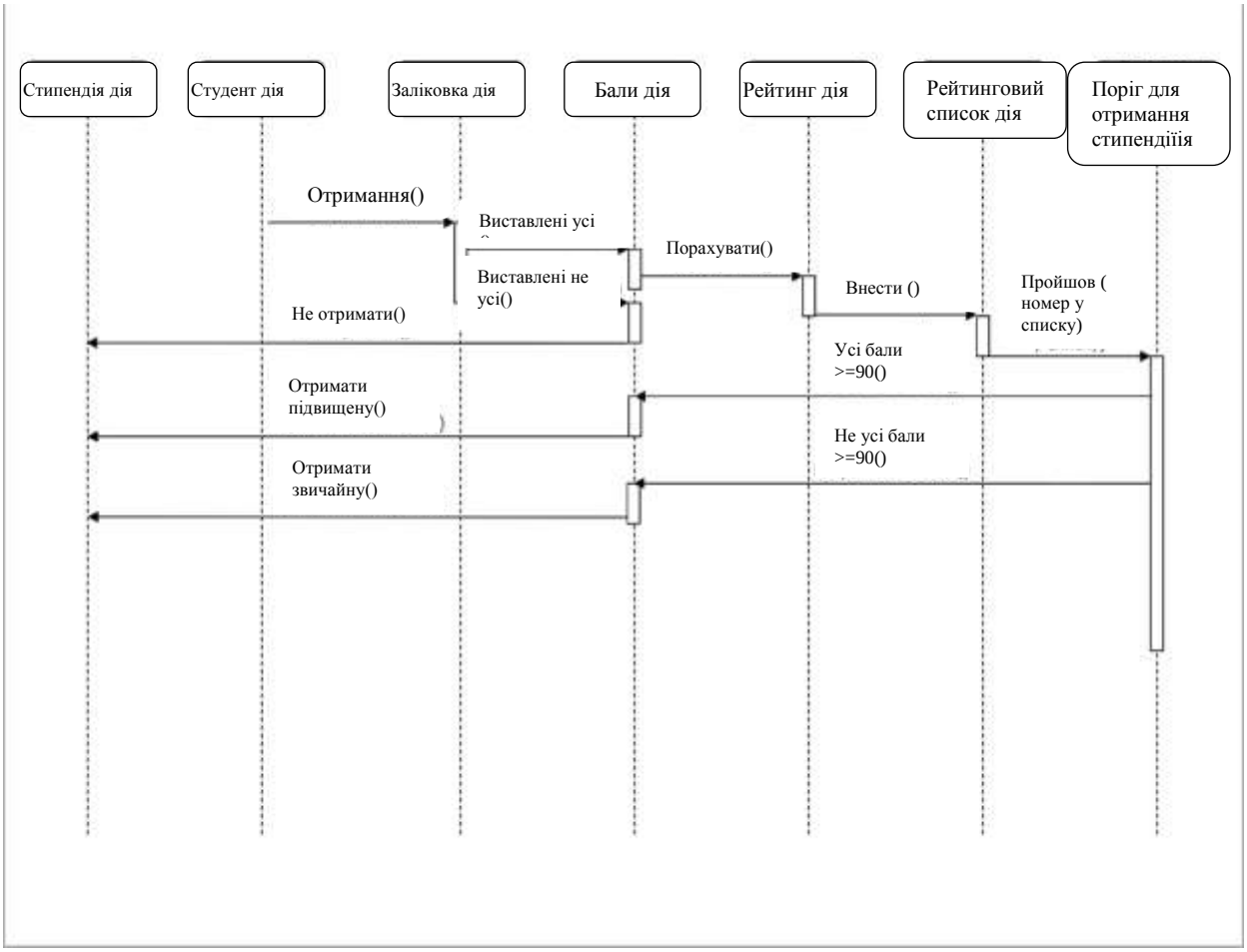


Рисунок 5.5 – UML-діаграма послідовності для процесу отримання студентом стипендії

5.4 UML-діаграма станів (statechart diagram)

Такий тип UML-діаграм використовується для того, щоб пояснити, яким чином працюють складні об'єкти.

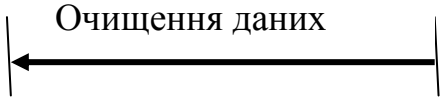
UML-діаграма станів [16, 17] подає динамічну поведінку сутностей, з урахуванням реакції сприйняття деяких конкретних подій, тобто показує, як складний об'єкт переходить з одного стану в інший, та описує процес

зміни станів представника певного класу, поведінка якого характеризується його реакцією на зовнішні події. Складний об’єкт, який реагує на зовнішні дії від інших систем або користувачів, називають реактивним.

В UML-діаграмах станів використовують такі позначення.

Прямокутники з округленими кутами зображають стани, через які проходить об'єкт протягом свого життєвого циклу.

Стрілками показують переходи між станами складного об'єкта, викликані виконанням методів описуваного діаграмою об'єкта.



Псевдостани:

– початковий, в якому знаходиться об'єкт відразу після його створення

;

– кінцевий, з якого об'єкт не може перейти в інший стан, якщо перейшов в нього

.

Приклад UML-діаграми станів процесу запису/очищення даних наведено на рисунку 5.6.

Рисунок 5.6 – UML-діаграма станів процесу запису/очищення даних

UML-діаграма станів також може демонструвати особливості функціонування складного об’єкта з урахуванням блочно-ієрархічного підходу. Приклад такої UML-діаграми станів проходження курсу навчання наведено на риснку 5.7.

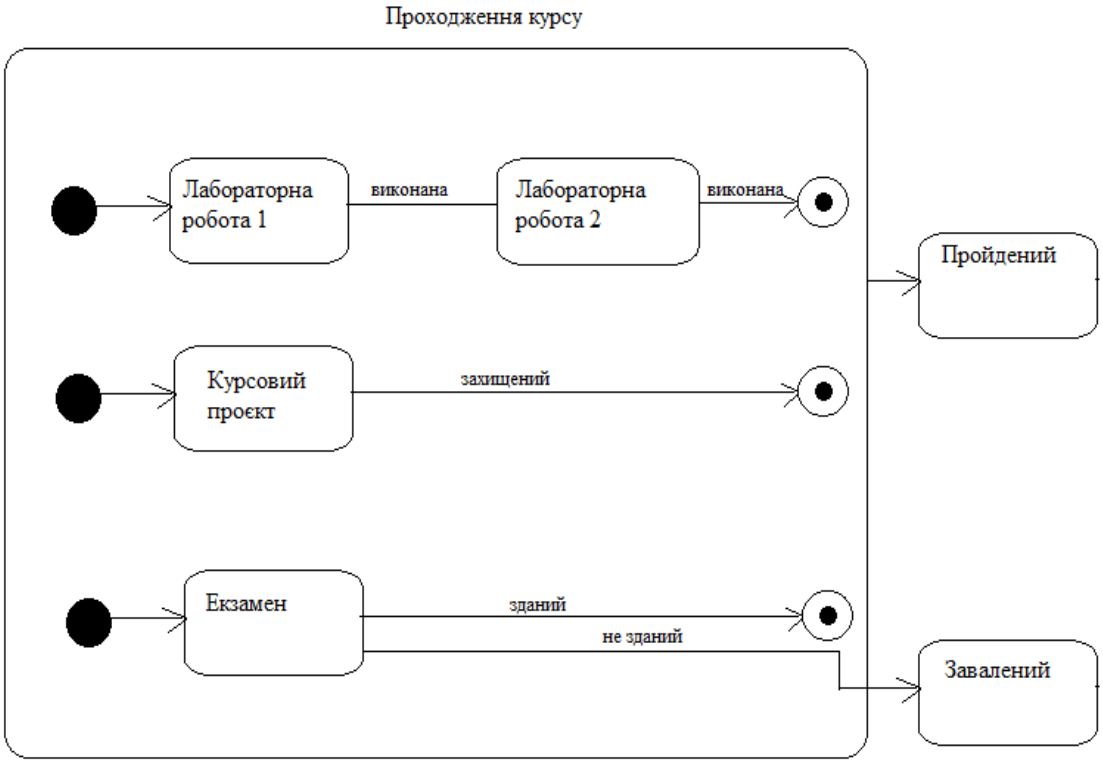


Рисунок 5.7 – UML-діаграма станів проходження курсу навчання

5.5 UML-діаграма об’єктів/класів (object diagram)

UML-діаграма об’єктів / класів [16, 17] являє собою набір статичних, декларативних складових моделі складного об’єкта, є графічним поданням структурних взаємозв'язків логічної моделі складного об’єкта, і, як кінцевий результат проєктування може застосовуватися як за низхідного проєктування, так і за висхідного проєктування.

Клас, як категорія, що використовується в процесі аналізу предметної області для складання словника предметної області (можуть бути як абстрактні поняття предметної області, так і класи, на які спирається розробка), має загальні атрибути та операції. Крім структури класів на відповідній діаграмі вказуються різні типи відношень між класами. Власне, діаграма класів є графом, вершинами якого є елементи типу «класифікатор», пов'язані різними типами структурних відношень. Сукупність типів таких відношень фіксована в мові UML та визначена семантикою цих типів відношень. Кожні з цих відношень мають власне графічне подання на діаграмі, що відбиває взаємозв'язки між об'єктами відповідних класів.

Базовими відношеннями або зв'язками у мові UML є:

- відношення залежності (dependency relationship)– позначається пунктирною лінією (між відповідними елементами) зі стрілкою на одному з її кінців («->» або «<-»). Таке позначення показує зв'язок окремих класів між собою; стрілка спрямована від класу-клієнта залежності до незалежного класу або класу-джерела;
- відношення асоціації (association relationship) – позначається неперервною лінією з додатковими спеціальними символами, що характеризують окремі властивості конкретної асоціації, такі як ім'я асоціації, а також імена та кратність класів-ролей асоціації. Водночас, ім'я асоціації не є обов'язковим елементом її позначення, але за його

наявності, воно записується з великої літери поряд з лінією відповідної асоціації;

- відношення узагальнення (generalization relationship) – позначається суцільною лінією з трикутною стрілкою одному з кінців, яка вказує більш загальний клас (клас-предок чи суперклас), та її початок – спеціальний клас (клас-нащадок чи підклас), що відповідає ієрархічному дереву, де клас-предок є коренем дерева, а класи-нащадки – його листям. Відмінність полягає у можливості відображення на діаграмі класів додаткової семантичної інформації, яка може вказує на різні теоретико-множинні характеристики цього відношення. У цьому випадку клас-предок на діаграмі може займати довільне положення відносно своїх класів-нащадків. Це можна використовувати для подання взаємозв'язків між пакетами, класами, варіантами використання та іншими елементами мови UML. На діаграмі класів це відношення описує ієрархічну будову класів і успадкування їх властивостей та поведінки. Водночас передбачається, що клас-нащадок має всі властивості та поведінку класу-предка, а також має свої власні властивості та поведінку, які відсутні у класу-предка;
- відношення реалізації (realization relationship), що має місце між двома елементами моделі у разі, якщо один елемент (клієнт) реалізує поведінку, задану іншим (постачальником). Відношення реалізації поділяється на відношення агрегації (aggregation relationship), яке позначається неперервною лінією, один із кінців якої – це незафарбований усередині ромб, що вказує на той із класів, який є «ціле», а інші класи є його «частинами», а також відношення композиції (composition relationship), що позначається неперервною лінією, один із кінців якої є зафарбованим усередині ромбом. Цей ромб вказує на той із класів, який є клас-композицією або «ціле». Інші класи є його «частинами».

Потрібно зазначити, що діаграма класів може містити інтерфейси, пакети, відношення і навіть окремі екземпляри, такі як об'єкти та зв'язки. Об’єкт, як екземпляр класу, ідентифікується значеннями атрибутів, що визначають його стан в поточний момент часу.

UML-діаграма об’єктів показує множину об'єктів – екземплярів класів (зображених на діаграмі класів) і відношень між ними в певний момент часу. Вона подає статичний вигляд складного об’єкта з погляду проєктування і процесів, як основу для сценаріїв, описуваних діаграмами взаємодії, та використовується для пояснення і деталізації діаграм взаємодії, а також діаграм послідовності.

Графічно клас зображається у вигляді прямокутника, який додатково може бути розділений горизонтальними лініями на розділи чи секції, де можуть вказуватися ім'я класу, атрибути (змінні) та операції (методи).

Приклад UML-діаграми класів наведено на рисунку 5.8.

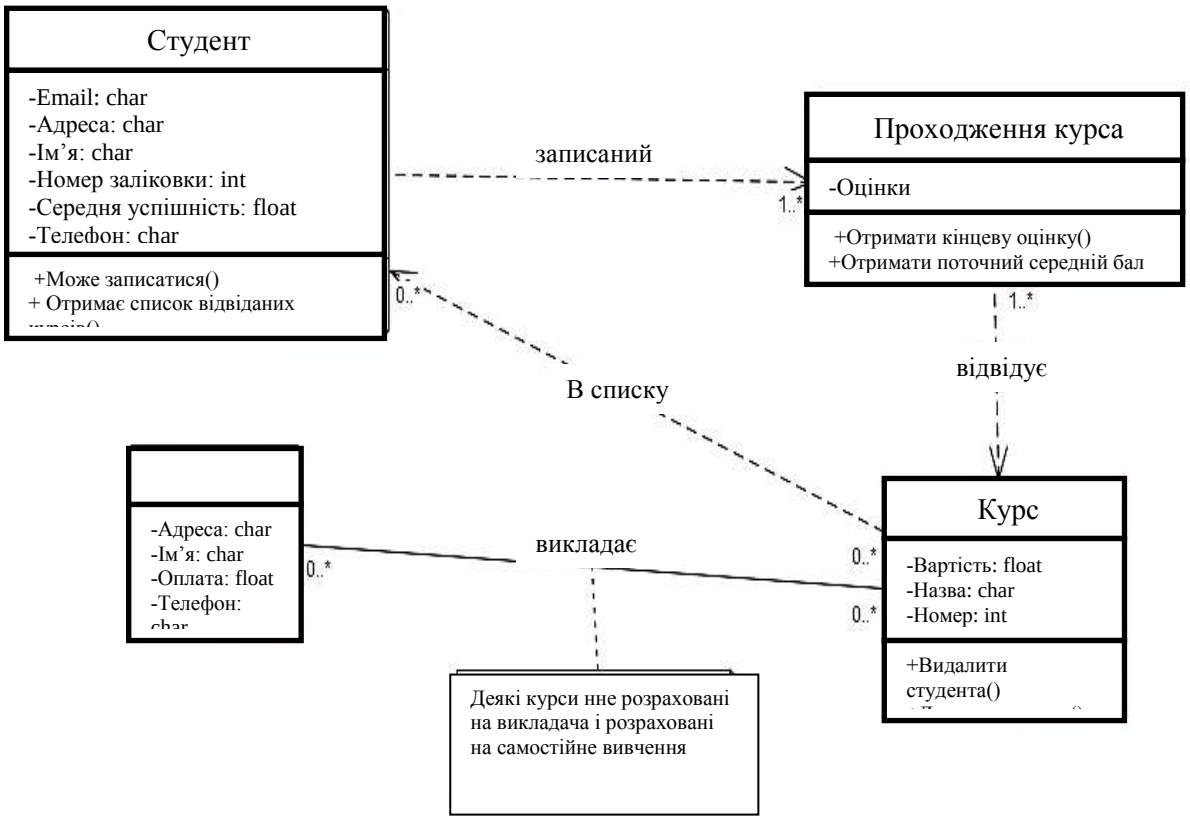


Рисунок 5.8 – UML-діаграма класів для організації процесу проходження курсу студентом

Об’єкт, як екземпляр класу, ідентифікується значеннями атрибутів, що визначають його стан в поточний момент часу.

UML-діаграма об’єктів показує множину об'єктів – екземплярів класів (зображених на діаграмі класів) і відношень між ними в певний момент часу. Вона подає статичний вигляд складного об’єкта, з погляду проєктування і процесів, як основу для сценаріїв, описуваних діаграмами взаємодії, та як такий, що використовується для пояснення і деталізації діаграм взаємодії, а також діаграм послідовності. Для графічного зображення об'єктів використовується такий самий символ прямокутника, що й у класів. Відмінності виявляються в процесі вказання імен об'єктів, які у разі об'єктів обов'язково підкреслюються. Запис імені об'єкта являє собою рядок тексту «ім'я об'єкта:ім'я класу», розділений двокрапкою.

Приклад UML-діаграми об’єктів наведено на рисунку 5.9.

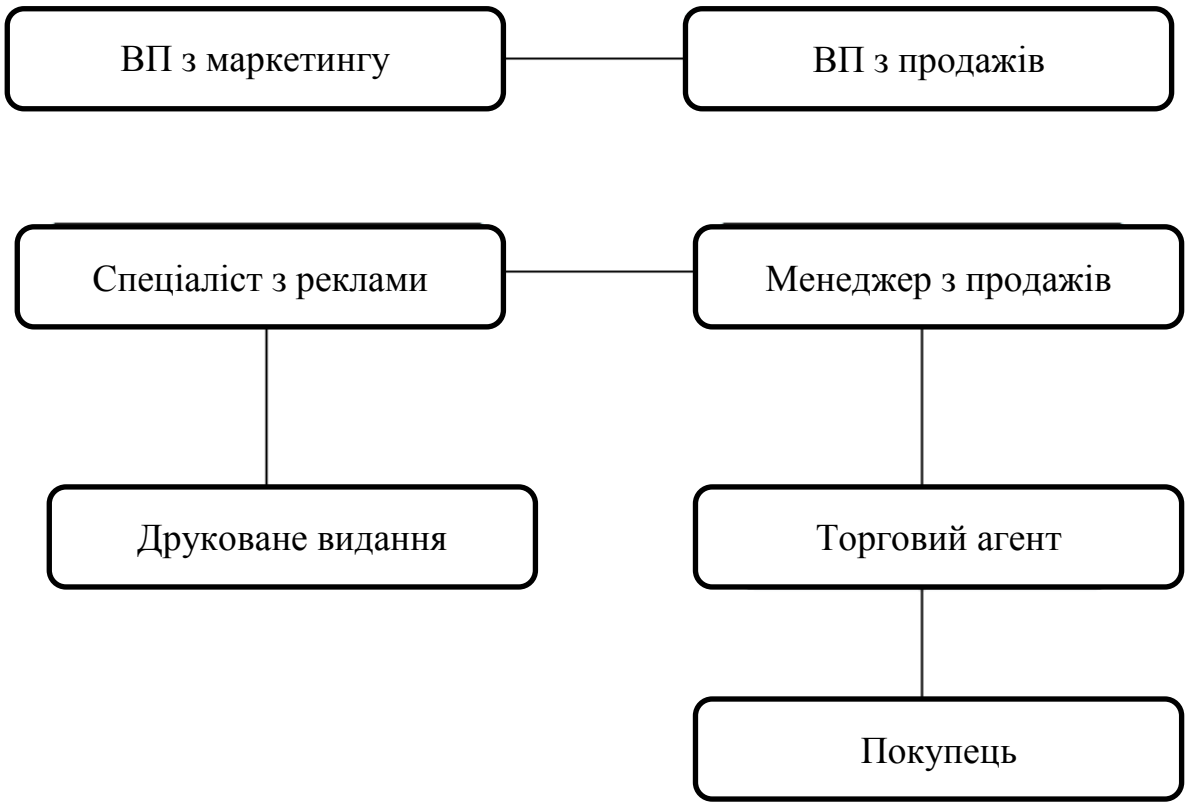


Рисунок 5.9 – UML-діаграма об’єктів предметної області «Маркетинг»

5.6 UML-діаграма розгортування (deployment diagram)

UML-діаграма розгортування [16, 17] показує топологію складного об’єкта і розподіл його складових по модулях, а також сполучення – маршрути передачі інформації між модулями складного об’єкта. Це єдина UML-діаграма, на якій застосовуються «тривимірні» позначення: вузли складного об’єкта позначаються кубами.

Діаграма розгортування призначена для візуалізації елементів та компонентів програми, що існують лише на етапі її виконання (runtime). У цьому випадку подано лише компоненти-екземпляри програми, які є виконуваними файлами або динамічними бібліотеками. Компоненти, які використовуються на етапі виконання, на діаграмі розгортування не показуються (наприклад, компоненти з вихідними текстами програм можуть бути лише на діаграмі компонентів, а не на діаграмі розгортування).

Діаграма розгортування містить графічні зображення процесорів, пристроїв, процесів та зв'язків між ними. На відміну від діаграм логічного подання, діаграма розгортування є єдиною для складного об’єкта загалом, оскільки має повністю відбивати особливості його реалізації. Розробка цієї діаграми, власне, є останнім етапом специфікації моделі складного об’єкта.

Основні цілі, що переслідуються під час розробки діаграми розгортування:

- визначити розподіл компонентів системи за її фізичними вузлами;
- показати фізичні зв'язки між всіма вузлами реалізації системи на етапі виконання;
- виявити вузькі місця у проєкті складного об’єкта та реконфігурувати його топологію для досягнення необхідної продуктивності.

Для забезпечення цих вимог діаграма розгортування розробляється спільно системними аналітиками, мережевими інженерами та системотехніками. Елементами діаграми розгортування є вузли (node), що на діаграмі розгортування зображаються у формі тривимірних кубів та мають власні імена. Самі вузли можуть подаватися як типи або їх

екземпляри. У першому випадку ім'я вузла записується без підкреслення і починається з великої літери. У другому – ім'я вузла-екземпляра записується як <ім'я вузла ':' ім'я типу вузла>. Ім'я типу вузла вказує на деякий різновид вузлів, що присутні у моделі складного об’єкта. Крім зображень вузлів на діаграмі розгортування вказуються відношення між ними. Як відношення виступають фізичні з'єднання між вузлами й залежності між вузлами та компонентами, зображення яких також можуть бути присутніми на діаграмах розгортування. З'єднання є різновидом асоціації та зображаються відрізками ліній без стрілок.

Приклад UML-діаграми розгортування для інформаційної системи «Обслуговування клієнта» наведено на рисунку 5.10.

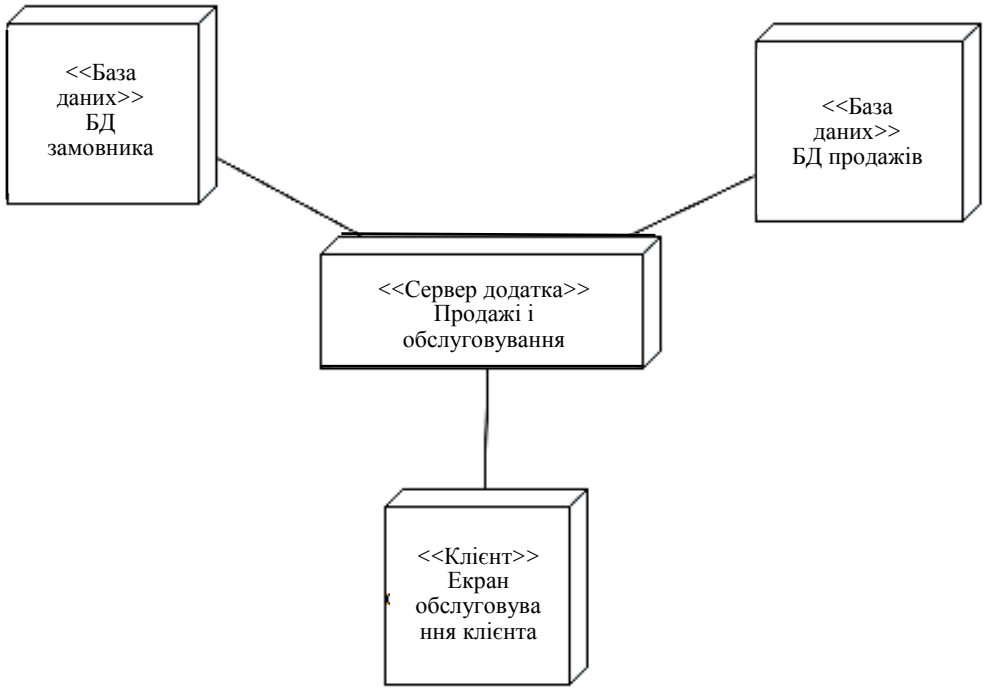


Рисунок 5.10 – UML-діаграма розгортування для інформаційної системи «Обслуговування клієнта»

Діаграми розгортування можуть мати складнішу структуру, що містить вкладені компоненти, інтерфейси та інші апаратні пристрої.

5.7 UML–діаграма активності/ видів діяльності (activity diagram)

UML-діаграма активності [16, 17] відображає виконавців і послідовність виконання відповідних алгоритмів функціонування складних об’єктів.

Такий тип діаграм слугує для візуалізації особливостей реалізації операцій класів, коли необхідно подати алгоритми їх виконання. Кожен стан може бути виконанням операції деякого класу або його частини, дозволяючи використовувати діаграми активності для опису реакцій на внутрішні події складного об'єкта. Водночас окремі елементарні обчислення можуть приводити до деякого результату або дії. На діаграмі активності відображається логіка та послідовність переходу від однієї діяльності (activity) до іншої, увага фіксується на результаті діяльності. Результат може призвести до зміни стану складного об’єкта або повернення деякого значення.

В UML-діаграмах активності використовують такі позначення.

- позначення стану («початок», «кінець»);
- дії;
- момент синхронізації дій (лінійка синхронізації, на якій сходяться або розгалужуються кілька стрілок).
- паралелограми (ромби) для позначення умови виконання послідовності дій.

Графічно стан дії зображається фігурою, що нагадує прямокутник, бічні сторони якого замінені опуклими дугами. Усередині цієї фігури записується вираз дії (action-expression), який має бути унікальним у межах однієї діаграми діяльності. Необхідно, щоб кожна діаграма діяльності мала єдиний початковий і єдиний кінцевий стан. Саму діаграму діяльності прийнято розташовувати таким чином, щоб дії прямували зверху донизу. У цьому випадку початковий стан зображатиметься у верхній частині діаграми, а кінцевий – у її нижній частині.

В процесі побудови діаграми активності використовуються лише нетригерні переходи, тобто такі, що спрацьовують відразу після завершення діяльності або виконання відповідної дії.

Такі переходи переводять діяльність у наступний стан відразу, щойно закінчиться дія попереднього стану. На діаграмі такий перехід зображається суцільною лінією зі стрілкою. Якщо зі стану дії виходить єдиний перехід, він може бути не позначений. Якщо таких переходів кілька, то спрацювати може лише один із них. Саме в цьому випадку для кожного з таких переходів має бути чітно записана умова у прямих дужках. У цьому разі для всіх переходів, що виходять з деякого стану, має виконуватися вимога істинності тільки одного з них. Подібний випадок зустрічається тоді, коли діяльність, що послідовно виконується, має розділитися на альтернативні гілки залежно від значення деякого проміжного результату.

Така ситуація отримала назву розгалуження, а для її позначення застосовується невеликий ромб, у якому немає жодного тексту. У цей ромб може входити тільки одна стрілка від стану дії, після виконання якого потік управління має бути продовжений по одній з взаємно виключних гілок.

Прийнято вхідну стрілку приєднувати до верхньої чи лівої вершини символу розгалуження. Вихідних стрілок може бути дві або більше, але для кожної з них явно вказується відповідна умова у формі виразу (наприклад, булевого).

Приклад UML-діаграми активності обслуговування пацієнта центру зовнішнім спеціалістом наведено на рисунку 5.11.

На рисунку 5.12 наведено приклад UML-діаграми активності обслуговування замовлення з урахуванням блочно-ієрархічного принципу.

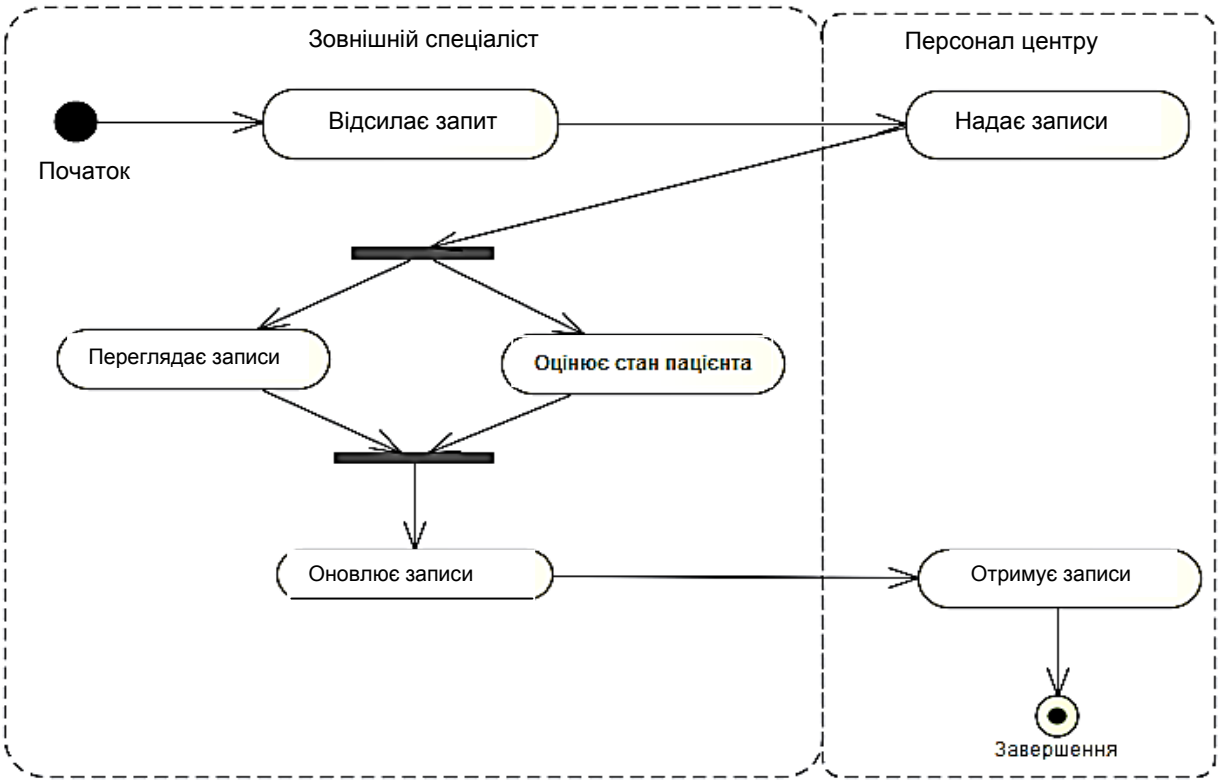


Рисунок 5.11 – UML-діаграми активності обслуговування пацієнта центру зовнішнім спеціалістом

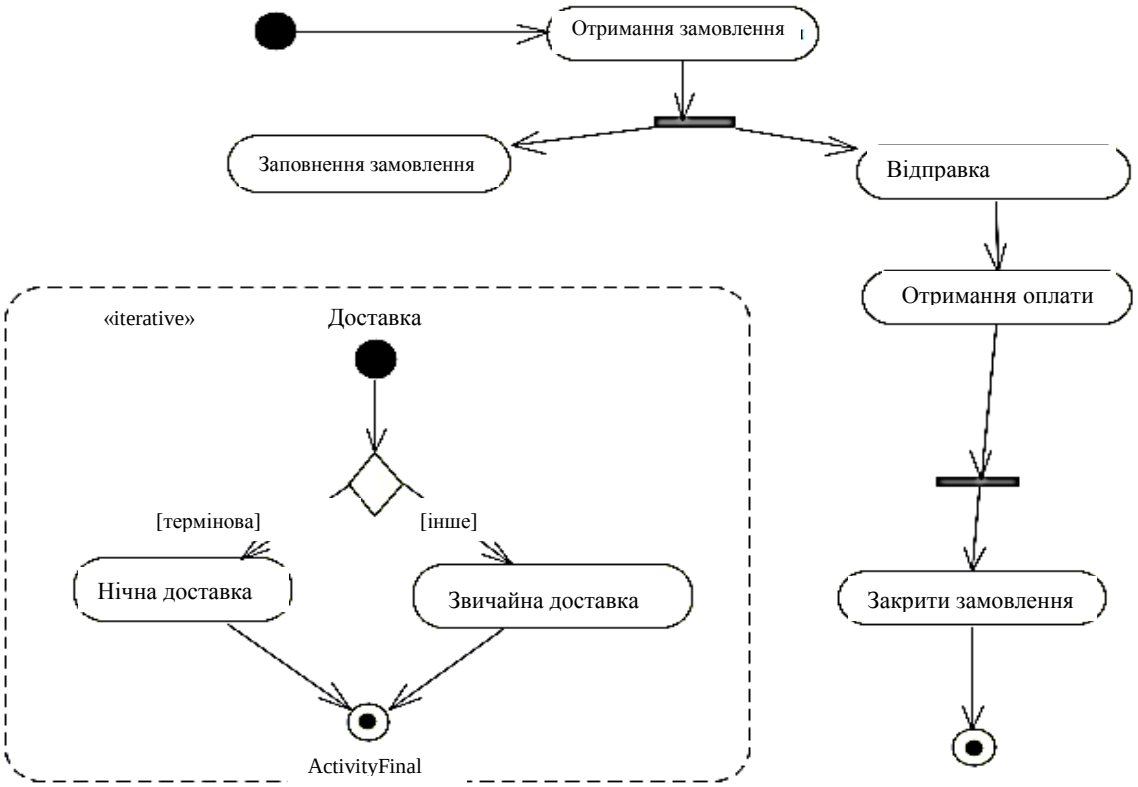


Рисунок 5.12 – UML-діаграма активності обслуговування замовлення з урахуванням блочно-ієрархічного принципу

Список використаних джерел

1. Савеленко О. К., Лисенко І. А., Іванченко О. О. С12 CASE-технології у проєктуванні інформаційних систем : навчальний посібник. – Кропивницький : Видавець Лисенко В. Ф., 2018. 240 с.
2. Олег Матвійків, Сергій Ткаченко, Володимир Хаханов. Інженерне проєктування складних об'єктів і систем Engineering Design of Complex Objects and Systems : навчальний посібник. – Львів : Кафедра САПР НУ «Львівська політехніка», 2016. 261 с. https://cad.lpnu.ua/picture/project/b2.pdf.
3. Методичні вказівки до самостійної роботи і практичних занять з дисципліни «Основи теорії систем та системний аналіз» для студентів денної і заочної форм навчання напряму підготовки 6.080101 «Геодезія, картографія та землеустрій» / Харк. нац. акад. міськ. госп-ва; уклад.: І. С. Глушенкова, Є. І. Кучеренко – Х. : ХНАМГ, 2010. 51 с.
4. Яшанов С. М., Дзус С. Б. Структурне та інформаційне моделювання в середовищі DESIGN/IDEF : навчально-методичний посібник. – К. : Вид-во НПУ імені М. П. Драгоманова, 2017. 73 с.
5. Кірчук Р. В., Герасимчук О. О., Завіша В. В. Сучасні інформаційні технології : навчальний посібник. – Луцьк : Технічний коледж Луцького НТУ, 2020. 134 с.
6. Роїк О. М., Савчук Т. О., Ткаченко О. М. Основи автоматизованого проєктування складних об’єктів і систем : навчальний посібник. – Вінниця : ВДТУ, 2001. 110 с.
7. Хох В. Д. Дослідження методів побудови експертних систем / В. Д. Хох, Є. В. Мелешко, М. С. Якименко // Системи управління, навігації та зв'язку, 2016. Вип. 4. С. 48–52. Режим доступу: http://nbuv.gov.ua/UJRN/suntz_2016_4_14.
8. Донченко М. В. Технології комп’ютерного проєктування : навчальний посібник. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2021. 364 с.
9. Мирончук В. Г., Єщенко О. А., Люлька Д. М., Якобчук Р. Л. Основи комп’ютерного проєктування : навчальний посібник. – К. : НУХТ, 2020. 360 с.

10. Барандич К. С., Подолян О. О., Гладський М. М. Системи автоматизованого проєктування : конспект лекцій. – К. : КПІ ім. Ігоря Сікорського, 2021. 97 с.
11. Методичні вказівки і завдання до самостійної роботи та контрольних робіт з дисципліни «Технології комп'ютерного проєктування» для денної та заочної форм навчання / Укладачі Т. О. Савчук, О. В Ольшанська. – Вінниця : ВНТУ, 2020. 77 с.
12. Савчук Т. О., Ольшанська О. В. Технології комп'ютерного проєктування : лабораторний практикум. Вінниця : ВНТУ, 2018. 122 с.
13. Дорогий Я. Ю., Дорога-Іванюк О. О. Сучасні технології автоматизованого проєктування і верифікації програм : конспект лекцій : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення», спеціалізації «Інженерія програмного забезпечення комп'ютерних систем» / КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2021. 89 с.
14. Куліков В. М. Підхід до побудови тестів перевірки цифрових пристроїв на надвеликих інтегральних схемах. // Інформаційні технології і безпека : зб. наук. пр. № 1(1) / ІСЗЗІ НТУУ «КПІ». Київ, 2011. С. 83–92.
15. Куліков В. М., Рябцев В. В., Паршуков С. С. Об'єктно-орієнтоване програмування для фахівців з кібербезпеки : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2023. 365 с.
16. Застосування uml для моделювання та проєктування інформаційних систем. Методичні вказівки до лабораторного практикуму з дисципліни «Об'єктноорієнтований аналіз та проєктування для студентів напряму підготовки 123 «Комп'ютерна інженерія» / Укл. Акименко А. М., Богдан І. В., Посадська А. С. – Чернігів : ЧНТУ, 2018. 37 с.
17. Методичні вказівки до виконання лабораторних робіт з дисципліни «Архітектура та проєктування програмного забезпечення» за розділом «Мова моделювання UML як засіб аналізу та проєктування програмних систем» / Уклад. Ткачук М. В, Сокол В. Є., Гамзаєв Р. О. та ін. Харків : НТУ «ХПІ», 2017. 60 с.

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Тамара Олександрівна Савчук
Ольга Вікторівна Ольшанська

Технології комп'ютерного проєктування

Навчальний посібник

Рукопис оформлено *О. Ольшанською*
Редактор *Т. Старічек*
Оригінал-макет виготовлено *Т. Старічек*

Підписано до видання 23.09.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-148.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.