Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи дизайну системи документообігу університету

(тема)

Виконав:

студент II курсу, групи СПм-22-3

Діян В.Р.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

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

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

# ЗАВДАННЯ

## НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Діян Володимир Романович
(прізвище, ім’я, по батькові)

1. Тема роботи Методи дизайну системи документообігу університету

затверджена наказом по університету від “ 1 ” квітня 2023 р. № 257Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

3.1 Інформаційна структура університету

3.2 Метод проектування паралельних застосунків COMET

3.3 Система моделювання General purpose Simulation System

4. Перелік питань, що потрібно опрацювати у роботі

4.1 Дослідження основних методів системного аналізу;

4.2 Розробка моделі системи електронного документообігу університету;

3) Розробка бази даних деканату;

4) Оцінка ефективності параметрів моделі системи електронного документообігу

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) — 17 слайдів

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

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

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Вибір метода проектування | 01.04.24-31.05.24 | |
| 2 | Розробка архітектури системи Вибір апаратних і програмних засобів проектування | 01.06.24-02.06.24 | |
| 3 | Розробка структурної моделі проекту | 03.06.24-05.06.24 | |
| 4 | Розробка програмної моделі проекту | 03.06.24-05.06.24 | |
| 5 | Оформлення матеріалів | 06.06.24-11.06.24 | |
| 6 | Підготовка до захисту | 11.06.24-12.06.24 | |
| | | | |
| | | | |
| | | | |
| | | | |

Дата видачі завдання 1 квітня 2024 р.

Студент _____
(підпис)

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

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 99 с., 21 рис., 11 табл., 2 дод., 16 джерел.

ПАРАЛЕЛЬНІ ЗАСТОСУНКИ, РОЗПОДІЛЕНІ ЗАСТОСУНКИ, АРХІТЕКТУРНЕ ПРОЕКТУВАННЯ, МОДЕ-ЛЮВАННЯ ПАРАЛЕЛЬНИХ СИСТЕМ

Метою даної роботи є проведення аналізу технологій розробки архітектури та програмного забезпечення паралельного розподіленого застосунку. Пропонується розробка та впровадження електронної системи документообігу в адміністративній системі вищого навчального закладу.

У ході виконання кваліфікаційної роботи вирішена задача розробки електронної системи обігу документів університету. Її вибір базується на показниках ефективності системи.

# ABSTRACT

Master’s thesis: 99 pages, 21 figures, 11 tables, 2 appendices, 16 sources.

PARALLEL APPLICATIONS, DISTRIBUTED APPLICATIONS, ARCHITECTURAL DESIGN, PARALLEL SYSTEMS MODELING

The purpose of this work is to analyze the architecture and software development technologies of a parallel distributed application. It is proposed to develop and implement an electronic document management system in the administrative system of a higher educational institution.

In the course of the qualification work, the task of developing an electronic document circulation system of the university was solved. Its selection is based on system performance indicators.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ…… .8
ВСТУП……..9
1 АНАЛІЗ ПРОБЛЕМИ І ПОСТАНОВКА ЗАВДАННЯ ДОСЛІДЖЕННЯ…11
1.1 Аналіз проблеми…….....11
1.2 Завдання дослідження…… 12
2 МЕТОДОЛОГІЇ СИСТЕМНОГО АНАЛІЗУ …….. 15
2.1 Розвиток методів системного аналізу…… …. .15
2.2 Модель «кодування-усунення помилок»…… … 16
2.3 Модель водоспаду…… … 17
2.4 Тимчасові прототипи…… … .20
2.5 Еволюційні прототипи…… . .22
2.6 Спіральна модель…… ……..……23
2.7 Об'єктно-орієнтовані методи…… ……..…27
2.8 Метод COMET…… ……..……28
2.9 Формальні методи…… ……..……29
2.10 Порівняння методу COMET з іншими методами розробки ПЗ……30
2.11 Висновки…… ……31
3 ЕЛЕКТРОННА СИСТЕМА ДОКУМЕНТООБІГУ З ВИКОРИСТАННЯМ UML…… ……..…33
3.1 Моделювання вимог……33
3.2 Аналітичне моделювання……...……37
3.3 Проектне моделювання…….…..48
3.4 Висновки…….....…51
4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ……52
4.1 Концептуальна структура бази даних……...…52

4.2 Логічна структура бази даних……..……...….54
4.3 Схема даних…….……60
4.4 Нормалізація та схема функціональних зв'язків……..…62
4.5 Реалізація бази даних…….…….65
4.6 Висновки…….……..…67
5 ВИБІР КРИТЕРІЇВ І ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ…….….68
5.1 Планування експерименту ……..…68
5.2 Застосування імітаційної моделі системи ……...……..…71
5.3 Висновки……..…78
ВИСНОВКИ ……..…82
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ……..……..84
ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІФІКАЦІЙНОЇ РОБОТИ ….86
ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ НА GPSS……..….96

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LAN - Локальна мережа (англ.Local Network)

CS – Складна система (англ., Complex System)

DES - Дискретно – событийная сист (англ., Discrete Event Sistem)ема

ES - Электронные системы (англ., Electronics Sistem)

PM - Программная модель (англ., Program Model)

DM Розробка даних (англ., Data Mining)

ЕС - Електронні системи

ПМ - Програмна модель

ФД - Функціональна дія

УПМ - Керуюча програмна модель

ОД - Область даних

ОС - Операційна система

ПЗ - Програмне забезпечення

ОС - Операційна система

# ВСТУП

Практики розробки об’єктно-орієнтованого програмного забезпечення все частіше впроваджуються складні системи, включаючи системні додатки, що працюють у режимі реального часу та паралельні. На практиці, правда, поточний Об'єктно-орієнтовані методи проектування програмного забезпечення все ще переважно орієнтовані на створення статичних моделі класу. Динамічні архітектурні моделі, що відображають загальні поведінкові властивості програмні системи часто створюються з використанням спеціальних методів, при цьому мало уваги приділяється результуючі наслідки продуктивності або надійності, поки проект не досягне реалізації. Зусилля до аналіз надійності та продуктивності цих архітектур відбуваються через опортуністичні, а не через систематичних підходів і за своєю суттю є громіздкими, ненадійними та неповторними.

Один із засобів підвищення впевненості в тому, що проект об’єктно-орієнтованої архітектури відповідає вимогам функціональні та продуктивні вимоги полягає в інтеграції формалізму з об'єктно-орієнтованим специфікація. Використовуючи цю техніку, артефакти об’єктно-орієнтованого дизайну все ще фіксуються в такому форматі як уніфікована мова моделювання (UML) [1], яка є інтуїтивно зрозумілою для архітектора програмного забезпечення. Конкретний метод, який використовується в цьому дослідження [2] полягає в інтеграції методу COMET з проектами об’єктно-орієнтованої архітектури зафіксовані в термінах моделей поведінки UML. Зокрема, у цій статті буде описано метод систематично переводити проект архітектури програмного забезпечення UML у базову модель COMET за допомогою а набір попередньо визначених шаблонів на основі набору поведінкових ролей об’єктів. Розподілена обробка має такі переваги:

- гнучка конфігурація - один і той самий застосунок допустимо конфігурувати різними способами, розмістивши його на потрібній кількості вузлів;

- більш локалізоване управління та адміністрування - розподілену підсистему, що виконується на своєму власному вузлі, можна спроектувати так, що вона буде автономною, тобто практично незалежною від інших підсистем, що працюють на інших вузлах;

- поступове розширення системи - якщо навантаження сильно зростає, систему легко розширити за рахунок додавання нових вузлів;

- зменшення витрат - найчастіше розподілене рішення виявляється дешевшим за централізоване, особливо якщо взяти до уваги стрімко зменшувану вартість і зростаючу продуктивність мікрокомп'ютерів;

- балансування навантаження - у деяких додатках загальне навантаження на систему може бути розподілене між різними вузлами;

- зменшення часу відгуку - запити користувачів локальних систем обробляються швидше.

Перехід до використання паралельних розподілених систем сприяв зміні вимог, що висуваються до програмного забезпечення. Реакцією на ці зміни стало широке поширення об'єктно-орієнтованих методів проектування. Об'єктно-орієнтовані концепції є особливо важливими для аналізу та проектування програмного забезпечення, оскільки вони стосуються фундаментальних питань адаптованості та розвитку.

Прикладом є метод COMET - метод архітектурного проектування та моделювання паралельних систем, що дає змогу проектувати паралельні, засновані на обміні повідомленнями додатки, забезпечуючи водночас високий ступінь структурованості. Таким чином, метою роботи є розробка архітектури та програмного забезпечення паралельного розподіленого додатка електронної системи обігу документів на основі методу COMET.

# 1 АНАЛІЗ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

## 1.1 Аналіз проблеми

Аналіз архітектури паралельного проектування та проектування в реальному часі Після створення основної моделі COMET створюються тестові випадки на основі варіантів використання сценарії з моделі UML. Ці тестові випадки використовуються для виконання моделі COMET з цією метою моделювання виконання архітектури та аналізу результуючої поведінки. Результати від моделювання використовується для оцінки архітектури об'єкта з точки зору бажаного результату та бажаного результату продуктивність. Ця імітаційна вправа може бути застосована до як низького, так і високого рівня абстракції  щоб отримати видимість у бажаних поведінкових аспектах архітектури. Наприклад, можна вибрати просто провести тестування чорної скриньки, розмістивши маркери введення на місця акторів, виконання моделювання, а потім спостереження за отриманими значеннями токенів на вихідних місцях актора. Крім того, якщо потрібне більш детальне дослідження, інженер може скористатися COMET ієрархічну побудову та спостерігати за такими аспектами, як поведінка стану змінюється всередині держави представлення COMET залежного об’єкта.

На додаток до можливостей моделювання, інструмент, який використовується в цій роботі, також має дуже потужний інструмент продуктивності [8], який можна використовувати для аналізу аспектів продуктивності одночасного керування архітектура програмного забезпечення. Два приклади, що ілюструють аналітичні можливості, можна знайти в [2]. До  коротко проілюструйте аналіз ефективності для цього документу про позицію, спочатку розглянемо випадок аналізу ефекти періодичних вхідних подій, які обробляються об'єктом керування за допомогою однієї черги повідомлень.

Таким чином, одним із вірогідних аспектів для аналізу є потенціал для

резервування черги на основі періодичного введення швидкість. Діаграму на малюнку 3(а) було побудовано для такого архітектурного сценарію з використанням генерованих даних інструментом продуктивності, змінюючи частоту періодичного введення між 100 мс, 200 мс та 500 мс. Як можна побачити на цій діаграмі, було визначено, що частота періодичної обробки 500 мс призвела до відсутності відставання в черзі, тоді як швидкості 100 мс і 200 мс призвели до значного відставання. Ці результати можуть потім дозвольте інженеру-програмісту скорегувати дизайн архітектури, щоб задовольнити продуктивність обмеження.

Інше застосування інструменту продуктивності можна побачити у проведенні наскрізного часу аналіз системи з декількома одночасними об'єктами. Для проведення цього аналізу були введені команди видається системі під час моделювання режиму роботи з використанням оціненої номінальної швидкості введення для операційна система. Інструмент продуктивності використовувався для моніторингу змін у вихідних даних системи та порівняти час спостережуваної зміни вихідного сигналу з часом, коли була видана оригінальна вхідна команда. Потім ці результати можна порівняти з продуктивністю вимоги, щоб визначити, чи справді система задовольняє необхідну пропускну здатність. За мати можливість проводити цю форму аналізу з паралельного дизайну програмного забезпечення та коригувати дизайн відповідно, ми отримуємо впевненість, що остаточний дизайн задовольнить необхідні характеристики вимоги системи.

## 1.2 Завдання дослідження

В дипломній роботі потрібно дослідити структуру і функції вищого навчального закладу. В якості прикладу роботи взято Харківський національний університет радіоелектроніки в Україні. Цей університет складається з дев’яти факультетів, основна задача яких – координація студентів на протязі їхнього часу навчання в університеті, починаючи з

першого курсу та видачи диплом про вищу освіту. Крім цього є 32 кафедри, які займаються навчальною, методичною та науково-дослідницькою роботою. Кафедри навчають студентів гуманітарним, фундаментальним, загально інженерним і спеціальним дисциплінам. Відділ аспірантури і докторантури готовий докторів наук по 13 спеціальностям, кандидатів наук по 26 спеціальностям.

Науково-дослідницька частина організовує наукову діяльність: про ведення наукових конференцій, виконання науково-дослідних та інших робіт, захист інтелектуальної власності. Навчально-методичне управління в університеті забезпечує організацію, контроль і аналіз навчального процесу по кафедрам і деканатам. Міжнародний відділ координує роботу навчально-наукових підрозділів університету по питанням навчання іноземних громадян, міжнародного співробітництва, забезпечення міжнародних наукових програм. Крім того, в університеті є ще такі відділи: інформаційно-розрахунковий центр, видавничо-поліграфічний центр, навчально-технічний телевізійний центр, центр технологій дистанційного навчання, бібліотека та багато інших відділів. Усі відділи та лабораторії належать проректорам і ректору університету. Завдяки дослідженням структури діючого університету та на основі отриманої в результаті моделі можливо було створити електронних служб університету та опис взаємодій між ними. Таким чином, постановка завдання полягає в наступному. Задана система університету: підрозділами, їх функціями, зв'язками між відділами. Потрібно розробити його модель із застосуванням сучасного методу проектування системи та оцінити ефективність роботи університету на основі отриманої моделі.

Актуальність роботи полягає в тому, що така модель дозволить розробити ефективну систему електронного документообігу. Модель також може застосовуватися для подальших досліджень або ознайомлення нових співробітників університету. На даний момент введена в експлуатацію система урахування студентів, викладачів , навчальної літератури, спільності в деканаті навчання студентів на іноземних мовах. Майбутня система

електронних служб університету призначена для вирішення завдань надійності, забезпечення доступності даних та інформаційної безпеки .

З короткого опису структури університету випливає, наскільки складними є зв'язки та взаємодія між компонентами цієї системи, а також і сама система. Створення моделі структури університету та її оцінка з метою дослідження та можливої оптимізації процесу представляє практичну новину (цінність) для користувачів системи (співробітників університету, студентів, абітурієнтів). Для моделювання такої складної динамічної системи необхідно застосувати методи системного аналізу та проектування, як згадувалося в підрозділі 1.1. Аналіз існуючих методів представлений в розділі 2. Наукова новина полягає в застосуванні методу COMET для опису моделі університету.

Ціль дослідження є аналіз інформаційної системи університету з метою створення системи електронного документообігу та сервісів в університеті.

Для досягнення поставленої цілі необхідно вирішити наступні завдання:

1) дослідження основних методів системного аналізу та вибір методу, який буде здійснюватися моделюванням;

2) побудова моделі системи електронного документообігу та сервісів в університеті вибраним методом;

3) побудова бази даних деканат, як основної структурної складової університету;

4) оцінка ефективності моделі системи електронного документообороту з використанням імітаційного моделювання, з метою її оптимізації.

# 2 МЕТОДОЛОГІЇ СИСТЕМНОГО АНАЛІЗУ

## 2.1 Розвиток методів системного аналізу

Системний аналіз має на меті представити залежність небезпечної події від менших, більш основних подій. Це робиться для того, щоб відповісти на різні запитання, наприклад:

Скільки речей має піти не так, щоб призвести до небезпечної події (чи потрібні нам додаткові засоби захисту)?

Чи є будь-які залежності нестерпними (чи слід шукати за своєю суттю безпечніші альтернативи)?

Як порівняти дві альтернативи дизайну (яку альтернативу нам вибрати)?

На ці запитання часто можна відповісти на основі якісної оцінки системного аналізу, але в аналізі основних небезпек зазвичай проводиться кількісна оцінка аналізованої системи, щоб відповісти на запитання «як часто я очікую цю подію?» ”. Результати аналізу зазвичай використовуються для відносних, а не абсолютних цілей. У таких випадках відносні частоти для альтернативних пропозицій використовуються як допомога для судження.

У системному аналізі прийнято використовувати графічні системи (діаграми). Це сприяє передачі аналізу між аналітиками та особами, які приймають рішення, і підтримує обговорення того, як уявлення про небезпеку розвивається в ході аналізу. З одного боку, діаграми допомагають візуалізувати створену модель залежності, з іншого боку, вони можуть обмежити складність проблем, які можна описати. Можна стверджувати, що це не є серйозним обмеженням, оскільки, якщо система настільки складна, то ймовірно, бажаною є безпечніша система. Найпоширенішими методами системного аналізу є:

Блок-схема (аналіз системи)

Аналіз дерева несправностей (аналіз системи)

Аналіз дерева подій

Діаграма причина-наслідок

Аналіз людських помилок

Подальше обговорення та джерела інформації наведено в посиланні [2], а чудовий вступ до методів (окрім аналізу блок-схеми) можна знайти в «Рекомендаціях США (США) щодо процедур оцінки небезпеки» [3].

Аналіз блок-схем зазвичай застосовується до досить простих систем, тоді як синтез дерева відмов застосовується до більш складних систем. Моделювання дерева подій зазвичай використовується для аналізу того, як система реагує на критичну подію. Причинно-наслідкові діаграми, більш складна техніка, яка долає деякі обмеження дерев несправностей, описана в посиланні [4], але не настільки широко застосована. «Рекомендації США щодо процедур оцінки небезпеки» [3] пропонують деякі вказівки щодо того, коли застосовна кожна техніка, тобто на якому етапі життєвого циклу закладу ця техніка буде найбільш корисною. Однак існує мало об’єктивних вказівок щодо того, як вирішити, коли потрібен детальний аналіз системи. Що стосується ідентифікації небезпеки, це поки що залишається на розсуд відповідальних осіб.

## 2.2 Модель «кодування–усунення помилок»

Дана модель була першою спробою представити модель процесу створення ПЗ, яка змогла б учести його основні особливості та зробити його керованим [8]. Вона описується таким чином:

1) поставити задачу;

2) виконати її до успішного завершення або відміни;

3) перевірити результат;

4) повторити при необхідності з 1-го кроку.

Звичайно, така модель не структурувала процес розробки, і говорити про можливості її ефективного застосування, особливо в крупних проектах, не має сенсу.

## 2.3 модель водоспаду

Існує п’ять етапів методології Waterfall: вимоги, проектування, впровадження, перевірка та обслуговування. Нижче ми обговорюємо кожен етап і мету кожного з них. Вимоги. На цьому етапі ви окреслюєте загальну картину вимог вашого проекту. За словами доктора Кріса Меттманна, головного спеціаліста з технологій та інновацій (CTIO) Лабораторії реактивного руху NASA, це «заяви високого рівня, які можна реалізувати багатьма різними способами». Наприклад, вимогою може бути те, що програмне забезпечення B2B обробляє мільйон транзакцій на день або обслуговує спільноту з 10 000 одночасних користувачів.

Дизайн. Коли ви зрозумієте вимоги проекту, наступним кроком стане пошук способів проектування рішень, які їм відповідають. Наприклад, якщо вимога передбачає обробку одного мільйона користувачів на день, ви повинні розглянути можливості, які найкраще підтримають це на етапі проектування. Як пояснив Меттманн, «нам, мабуть, не слід мати єдиного серверу, оскільки він нестійкий. Тож, можливо, наша конструкція говорить про те, що ми повинні мати надлишковість із кількома внутрішніми серверами, щоб, якщо один з них піде, ми все одно могли досягти мети обробки мільйона транзакцій на день».

Реалізація. Під час цього етапу ви вибираєте один із своїх потенційних проектів і використовуєте технологію для його реалізації. Це може включати збір даних і перевірку того, чи здатний проект підтримувати вимоги.

Перевірка. На цьому етапі ви берете реалізацію, створену на четвертій фазі, і перевіряєте, чи відповідає вона вашим вимогам. Якби, наприклад, початковою вимогою було обробляти один мільйон транзакцій на день, ви б перевірили, чи можливо це. Якщо ви зіткнетеся з проблемами (скажімо, ви можете виконувати лише 500 000 транзакцій на день), ось куди ви повинні повернутися та перевірити, де могли виникнути проблеми, пояснив Меттманн.

Технічне обслуговування. Проект не завершується після того, як він пройшов валідацію та перевірку. Систему ще потрібно підтримувати. Під час технічного обслуговування ви «розробляєте стратегії для оновлення та модернізації». Це включає в себе виправлення систем, оновлення систем, впровадження оновлення програмного забезпечення або тестування на помилки та їх виправлення, якщо вони трапляються.

Методологія водоспаду: переваги та недоліки

Одна з переваг Waterfall полягає в тому, що він має фіксований графік і бюджет, оскільки цілі проекту є конкретними та окресленими з самого початку. Після встановлення мети проекту методологія Waterfall не передбачає частого зворотного зв’язку чи співпраці з клієнтом, окрім встановлених етапів або результатів для кожної фази. Це полегшує для керівників проектів планування та спілкування із зацікавленими сторонами чи діловими партнерами. Однак, хоча це може допомогти з плануванням, це також практично лише тоді, коли клієнт має чітку та фіксовану кінцеву мету і не потребує участі в процесі розробки проекту.

Одним із недоліків цієї методології є те, що вирішення неочікуваних проблем може бути складним і своєчасним. «Коли ви переходите від фази до фази, фази можуть інформувати одна одну», — пояснив Маттманн. Наприклад, труднощі на етапі впровадження можуть свідчити про те, що у вас був поганий дизайн. Проблема в тому, що «ви можете ніколи цього не зрозуміти, доки не запровадите або не зробите перевірку та перевірку». У таких ситуаціях жорсткість повного завершення одного етапу перед переходом до наступного може подовжити часові рамки проекту. Найкращий спосіб запобігти цьому, коли дослідник використовуєте Waterfall.

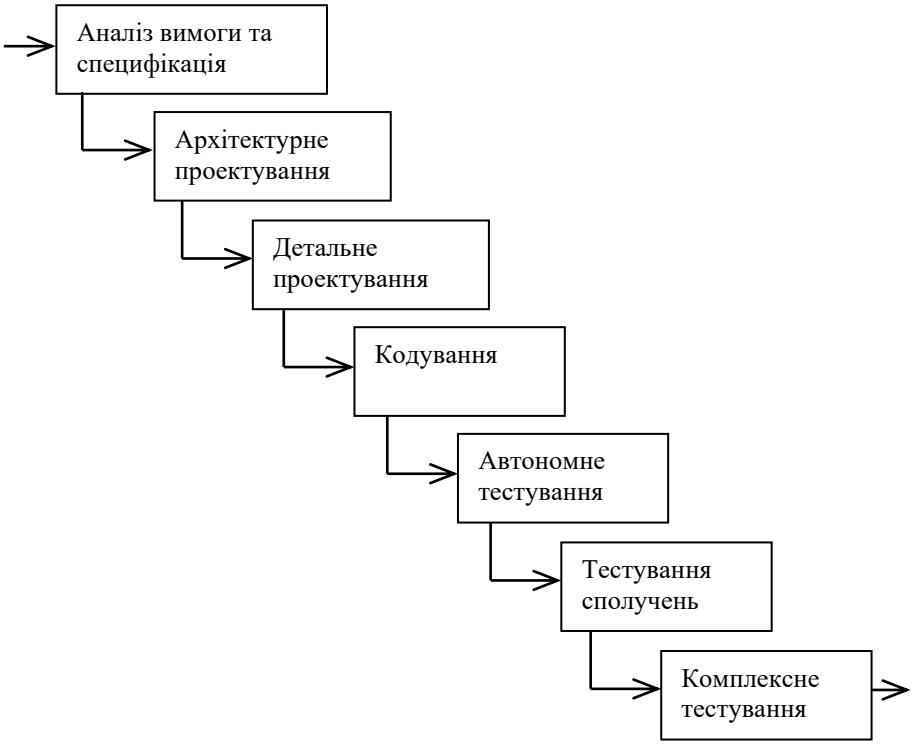


Рисунок 2.1 – Модель водоспаду

При подальшому використанні моделі модель водоспаду однієї з найбільших проблем виявилося можливе невідповідність кінцевого продукту і початкових вимог, які до нього пред'являлися. Причина полягає в тому, що працездатний продукт з'являється вже на останніх етапах розробки. Так як роботу на різних етапах зазвичай виконують різні спеціалісти, і проект передається від однієї групи до іншої, в результаті можна отримати не зовсім те, що передбачалося спочатку.

Також і дослідження показали, що окрім виявлених в ході роботи над проектом також можуть бути значні помилки в специфікаціях вимог, і вони зазвичай виявляються на останніх етапах проектування системи. Таким чином, вартість їх виробів найбільша для таких моделей. Відповідно дуже складно виправити помилки проектування та ефективності на етапах, коли система вже працездатна. К тому же такий підхід зовсім не пристосований для зміни вимог замовника в ході розробки. Щоб уникнути появи таких помилок на останніх етапах проектування, були розроблені моделі

тимчасових прототипів і еволюціонуючи прототипів (evolutionary prototypes).

## 2.4 Тимчасові прототипи

Цей метод заснований на моделі водоспаду і застосовується для уточнення вимог користувача по інтерфейсу системи. На рисунку 2.2 приведена частина діаграми, що відноситься до розширення методу водоспаду тимчасовими прототипами.

Швидке прототипування, також відоме як швидка перевірка, коли до нього навмисно підходять із явними гіпотезами, стосується процесу тестування ідеї якомога швидше та дешевше. Основна концепція полягає в тому, що краще завчасно та з невеликими витратами виявити, що команда рухається в неправильному напрямку, замість того, щоб зазнати невдачі після того, як уже вклали час і гроші в вдосконалення небажаного продукту чи послуги (див. розділ «Виявлення незадоволених потреб). : проблеми, які потребують вирішення”). Коли провідні новатори кажуть «створити культуру, яка охоплює та відзначає невдачі», вони насправді мають на увазі сприймати швидкі, дешеві невдачі, методично керуючись навмисним експериментуванням. Добре побудовані гіпотези, упорядковані за тим, що є найважливішим для успіху та найменш відомим чи зрозумілим, визначають великі припущення, які мають бути правдивими, щоб досягти бажаного результату. Експерименти, значно прискорені за допомогою методів, які ми обговоримо в цьому розділі, потім можуть бути розроблені та впроваджені, щоб перевірити, чи людина рухається в правильному напрямку. Таким чином, вивчення того, що працює, а що ні, за кілька днів чи тижнів замість місяців чи років є ефективним скасуванням гіпотези, а не провалом. Випробовуючи ідею якнайшвидше та якомога дешевше, команда отримає уявлення про те, як її вдосконалити, і у них залишиться більше часу та ресурсів для внесення необхідних покращень на основі правдоподібних доказів, отриманих у реалістичному контексті. Навіть якщо ідея

невідшліфована або недостатньо розроблена, швидке прототипування може бути використано на ранніх стадіях процесу, щоб визначити, які аспекти продукту працюють, які припущення щодо продукту витримуються та який інтерес споживачів до продукту існує (див. розділ «Проведення глибокого дослідження»). Дослідження ринку"). Цей процес є перевагою порівняно з тестуванням продукту після того, як було витрачено велику кількість часу та грошей, оскільки може виявитися, що споживачі не мають попиту чи інтересу до продукту, і весь час та гроші були витрачені даремно. Перевірені дані швидких дешевих тестів швидкого прототипування надають знання, які команда може використати для повторної оцінки та створення продукту, який краще підходить для бажаної кінцевої мети.

Однак ми вважаємо, що за своєю суттю всі ці методи можна класифікувати на три основні групи: тест паром, фальшивий інтерфейс і фальшивий бекенд. Таким чином, вони перевіряють, чи хоче хтось те, що планує побудувати підприємець, як люди використовуватимуть це та чи досягає це бажаних результатів.

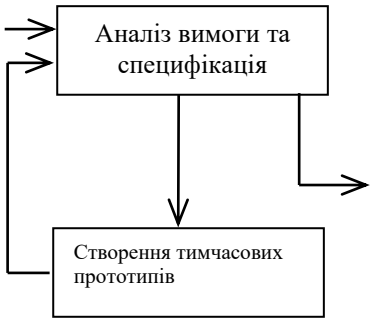


Рисунок 2.2 – Уточнення вимог за допомогою тимчасових прототипів

Крім цього, такий прототип можна використовувати на етапі архітектурного проектування для створення експериментальних прототипів проекту, перевіряючи логіку алгоритмів і одержувану продуктивність, як показано на рисунку 2.3.

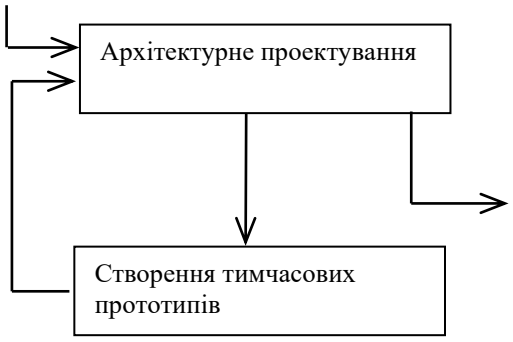


Рисунок 2.3 – Перевірка архітектури за допомогою тимчасових прототипів

Відповідальною стороною цього методу для замовника є додаткові витрати ресурсів на створення прототипів, тоді як зацікавлений розробник наддасть, як можна більш деталізовані прототипи, щоб знизити можливість отримання неадекватної системи.

## 2.5 Еволюційні прототипи

Інкрементна модель заснована на тому, що ПЗ можна ввести в експлуатацію по частинам. Задача розробляється на відносно незалежні складові, які розробляються по окремості. Як видно з малюнка 2.4, послідовно створюються проміжні прототипи з більш широкою функціональністю для отримання готової системи. Цей підхід застосовується для отримання працездатних модулів системи як можна швидше, з наступним доповненням функціональності. Так само можна протестувати критично важливі компоненти системи за швидку дію.

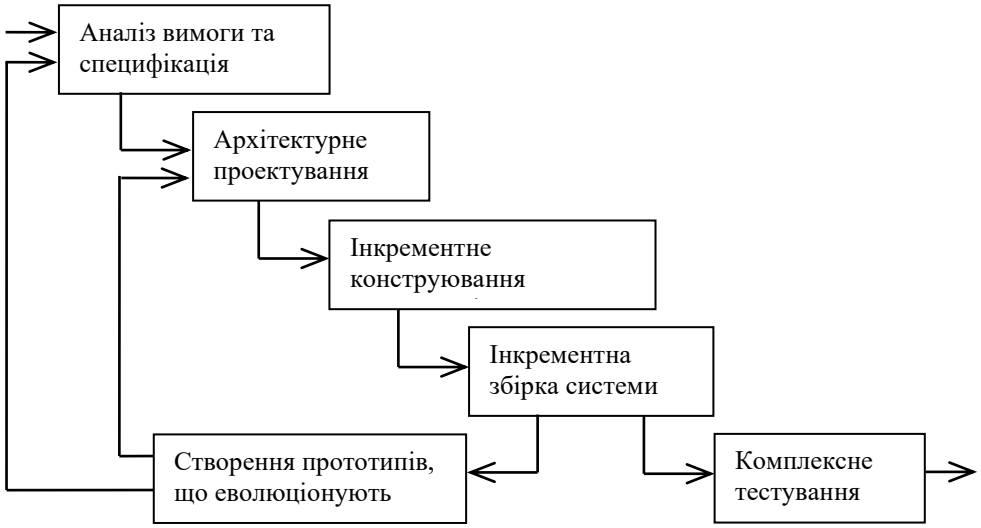


Рисунок 2.4 – Життєвий цикл інкрементної розробки системи

До недоліків необхідно віднести той факт, що поділ на функціональні блоки сповільнює весь процес, так як виникає необхідність забезпечення їх взаємодій між собою, витрачаються ресурси на виділення окремих модулів і керування, отримані в результаті складної системи . Обе модифікації моделі водоспаду можна поєднувати – використовувати тимчасові прототипи для уточнення вимог, а після складання специфікацій застосовувати інкрементний життєвий цикл.

## 2.6 Спіральна модель

У сфері розробки програмного забезпечення були розроблені різні методології для оптимізації процесу та забезпечення ефективного управління проектами. Однією з таких методологій є спіральна модель, яка пропонує гнучкий та ітеративний підхід до життєвих циклів розробки програмного забезпечення.

У цій статті я детально досліджу життєвий цикл розробки програмного забезпечення Spiral Model, її переваги, процес, стратегії впровадження та потенційні випадки використання. Тож давайте зануримося у світ спіральної моделі та дізнаємося, як вона може покращити процес розробки програмного

забезпечення.

Розуміння життєвого циклу розробки програмного забезпечення спіральної моделі

Спіральна модель — це модель процесу розробки програмного забезпечення, орієнтована на ризики, яка поєднує в собі елементи як водоспаду, так і ітераційної методології розробки. Він наголошує на постійному вдосконаленні та аналізі ризиків протягом усього життєвого циклу розробки програмного забезпечення (SDLC). Давайте розберемо спіральну модель на її ключові компоненти та детально розберемо кожну фазу.

Фаза 1 — Планування. Фаза планування знаменує початок Спіральної моделі. На цьому етапі визначаються цілі проекту, вимоги та обмеження. Команда розробників проводить детальний аналіз здійсненності, щоб оцінити життєздатність проекту. Основні дії на етапі планування включають визначення обсягу проекту, встановлення цілей проекту та визначення потенційних ризиків і невизначеності.

Фаза 2 — Аналіз ризиків. На етапі аналізу ризиків команда розробників визначає, оцінює та зменшує потенційні ризики, пов’язані з проектом. Ця фаза передбачає комплексну оцінку ризиків для визначення впливу та ймовірності різних ризиків. Команда оцінює альтернативні рішення та обирає найкращий підхід для ефективного усунення виявлених ризиків.

Фаза 3 — Інженерія. Фаза проектування зосереджена на фактичній розробці програмного продукту. Цей етап передбачає проектування архітектури, розробку програмних модулів і проведення частого тестування та інтеграції. Спіральна модель дозволяє ітераційну розробку, дозволяючи команді вдосконалювати та вдосконалювати продукт поступово.

Етап 4 — Оцінка. Етап оцінювання передбачає суворе тестування та оцінку програмного продукту. Команда оцінює розроблені функції відповідно до встановлених вимог і цілей. Будь-які невідповідності або відхилення документуються та негайно усуваються. Цей етап також включає

відгуки користувачів і залучення, щоб переконатися, що програмне забезпечення відповідає їхнім очікуванням.

Переваги спіральної моделі. Спіральна модель пропонує кілька переваг, які роблять її популярним вибором у розробці програмного забезпечення. Ось деякі основні переваги:

Гнучкість: спіральна модель забезпечує гнучкість і враховує зміни на будь-якому етапі процесу розробки. Ця гнучкість особливо корисна під час роботи зі складними проектами або змінними вимогами.

Управління ризиками: наголос на аналізі та зменшенні ризиків виділяє Спіральну модель. Завдяки ранньому виявленню та усуненню ризиків модель допомагає мінімізувати невдачі проекту та підвищує шанси на успіх.

Залучення клієнтів: спіральна модель заохочує залучення клієнтів протягом усього процесу розробки. Регулярні відгуки та співпраця гарантують, що кінцевий продукт відповідає очікуванням клієнта.

Поступове вдосконалення: ітераційний характер спіральної моделі дозволяє поступове вдосконалення програмного продукту. Кожна ітерація спирається на попередню, включає відгуки та вдосконалення.

Стратегії реалізації спіральної моделі. Реалізація спіральної моделі вимагає ретельного планування та виконання. Нижче наведено кілька стратегій для забезпечення плавного впровадження спіральної моделі у ваші проекти розробки програмного забезпечення:

Стратегія 1. Завчасне визначення ризиків проекту. Щоб максимізувати переваги спіральної моделі, дуже важливо визначити потенційні ризики на ранній стадії проекту. Проведіть ретельний аналіз ризиків на етапі планування та створіть ефективні стратегії управління ризиками.

Вплив компаній-розробників програмного забезпечення на зростання бізнесу

Стратегія 2. Залучення зацікавлених сторін. Залучення зацікавлених сторін, у тому числі клієнтів і кінцевих користувачів, до всього процесу розробки є життєво важливим. Їхній внесок і відгуки дають цінну

інформацію, яка сприяє успіху програмного продукту.

Стратегія 3. Ітеративна розробка та тестування. Спіральна модель процвітає завдяки ітераційній розробці та циклам тестування. Зробіть акцент на регулярному тестуванні та інтеграції, щоб швидко виявляти та виправляти проблеми. Такий підхід забезпечує розробку надійного та якісного програмного продукту. На рисунку 2.5 наведена спіральна модель процесу розробки. Тут радіальна координата відповідає затратам, а угловая – завершення поточного циклу розробки системи. Ця модель у першому квадранті містить цілий, альтернативи та обмеження поточного циклу спіралі, у другому – аналіз ризиків та оцінку альтернативи, у третьому – розробку та перевірку, а четвертий квадрант показує завдання, які будуть вирішуватися на наступному квадранті спіралі. Число вітрів спіралі залежить від особливостей цього проекту.

Основна ціль спіральної моделі полягає в зниженні ризиків проекту, аналіз яких виконується на початку кожної витки спіралі. Після аналізу ризиків можна визначити, які види діяльності потрібно ¬виконати при проектуванні системи. Для кожного процесу розробки програми підходить своя модель. За допомогою спіральної моделі можна визначити, яка саме модель: модель водоспаду, інкрементної розробки , тимчасових прототипів або друга, - підходить для даної системи більше всього.

Спіральна модель дозволяє вирішити основні види проблем при проектуванні ПЗ, викликані змінами вимог до проекту, змінами параметрів проекту (років, бюджету, якості) або тимчасових замовлень. Вона передбачає усунення завдань з меншими пріоритетами там і зміну шляху проекту. Всі зміни вносяться з урахуванням основного критерію - збереження термінів проекту.


Рисунок 2.5 – Спіральна модель процесу розробки

## 2.7 Об'єктно-орієнтовані методи

Широко поширена в даний час об'єктно-орієнтована методологія, яка має класи, методи класів, об'єкти, такі властивості ООМ як поліморфізм, інкапсуляція та спадкування. Відповідно форми і методи, засновані на даній методології. Наприклад, метод OOSE (Object-Oriented Software Engineering) застосовує об'єктно-орієнтований аналіз, проектування та програмування. [12]

На основі цього методу в 1995 році був створений метод RUP (Rational Unified Approach) - один із найбільш поширених методів комплексного управління процесом розробки. Крім цього на основі ітеративної моделі фірмою Microsoft розроблений аналог методу RUP – MSF (Microsoft Solutions Framework), а також аналогічний підхід компанії Borland – ALM ( Application Lifecycle Management). Метод RUP систематизує процес створення ПЗ із застосуванням UML (Unified Modelling Language). [5] Розробка проекту методом RUP складається з чотирьох фаз із використанням послідовності дій.

Наступні послідовності дій розподіляються по фазам і можуть виконуватися паралельно:

1 определение тренований;

2 аналіз системи;

3 проектування;

4 реалізація;

5 тестування.

Об'єктно-орієнтована методологія ROOM (Real-Time Object Oriented Modelling) призначена для розробки системи реального часу на рівні схеми та рівня деталізації (з використанням мов програмування) [6]. Згідно з цією методологією передбачається створення системи з об’єктів із завданням взаємодій між ними. Така модель ефективна в великих проектах, а також там, де застосовуються засоби швидкої розробки (RAD, Rapid Application Development), засновані на цих технологіях і містять готові бібліотеки класів.

Однак аналіз використання методів на практиці показує, що в RAD-проектуванні не часто використовуються ООМ, так як основна ціль RAD полягає в швидкому проектуванні з готових компонентів. Услід за цими особливостями методології, ООМ застосовується в масштабних проектах, з обов’язковими етапами аналізу та проектування за встановленим - стандартами.

## 2.8 Метод COMET

Метод COMET (method of development object oriented and distributed systems) є одним із методів проектування паралельних додатків, у тому числі розподіленого та реального часу. У відповідності з визначенням, даними в [11], метод проектування являє собою систематичний опис етапів створення проекту за умови, що вимоги до системи вже сформульовані. Модель життєвого циклу в COMET - це ітеративний процес розробки на основі концептів прецедентів. Ітеративний процес розробки методом COMET

показаний на рисунку 2.6. Кожен прецедент описує послідовність взаємодій між кількома акторами.

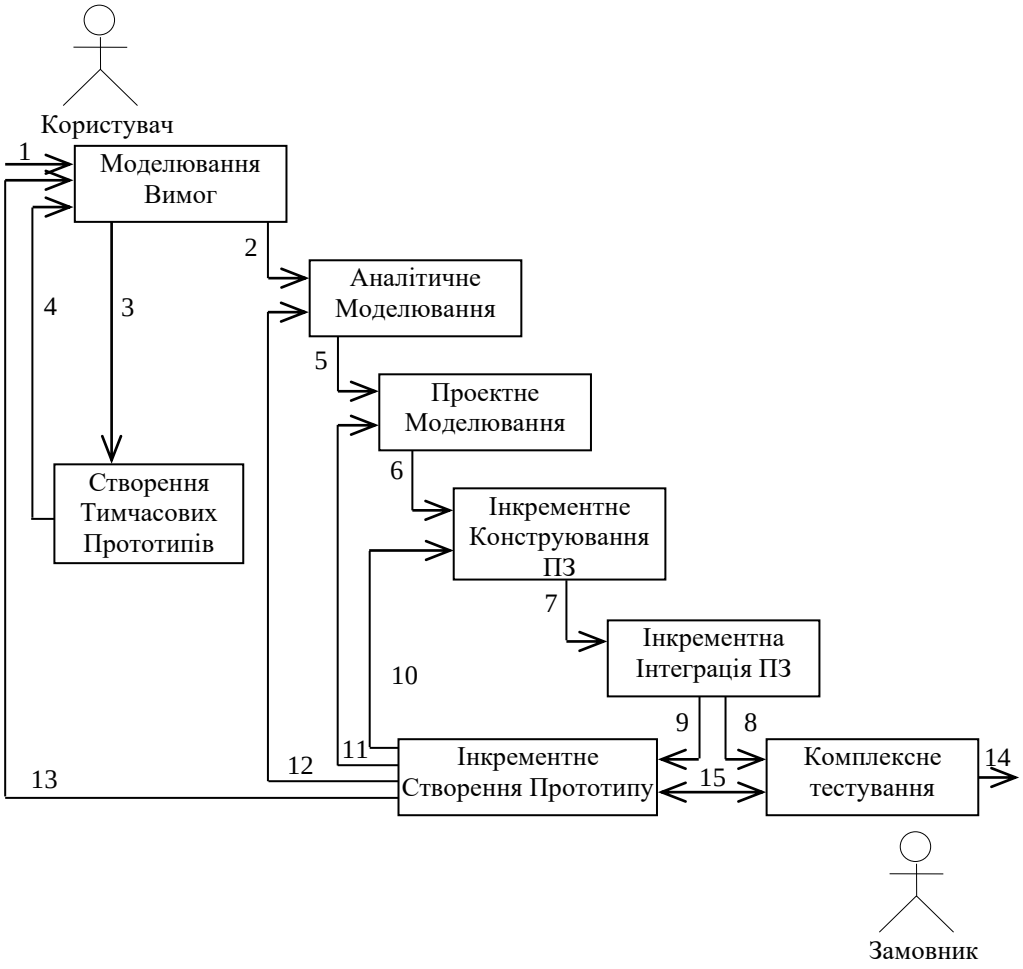


Рисунок 2.6 - Модель життєвого циклу розробки об'єктно-орієнтованого ПЗ в методі COMET

## 2.9 Формальні методи

Крім цих методів проектування, аналіз яких був зроблений вище, в підрозділі коротко розглянуті інші методи. Одним із таких видів є форма - льних методів проектування систем, заснованих на математикі. Розвиток багатьох інженерних дисциплін вийшло саме на математичній основі. Наскільки програми мають точно визначену поведінку , вони можуть розглядатися як математичні об’єкти. Формальні методи допомагають

вирішити задачу забезпечення надійності про граммних продуктів; вони можуть бути застосовані як при аналізі вимог для забезпечення точності формулювання вимог, так і в процесі реалізації для забезпечення відповідності коду програми сформульованим вимогам.

## 2.10 Порівняння методу COMET з іншими методами

Метод паралельного проектування для систем реального часу (CODARTS). (Gomaa 1993) заснований на сильних сторонах попереднього паралельного проектування, проектування в реальному часі, і ранні об'єктно-орієнтовані методи проектування, наголошуючи на приховуванні інформації структурування модулів і паралельне структурування завдань. Octopus (Awad, Kuusela, and Ziegler 1996) — це метод проектування в реальному часі, заснований на про випадки використання, статичне моделювання, взаємодію об’єктів і діаграми станів. Для систем реального часу ROOM (Selic, Gullekson, and Ward 1994) є об’єктно-орієнтованою системою реального часу. метод проектування, який тісно пов’язаний із CASE (Computer-Assisted Software Engineering) під назвою ObjecTime.

ROOM базується на акторах, які активні об’єкти, які моделюються за допомогою варіації діаграм станів під назвою ROOMcharts. Модель ROOM можна виконати, тому її можна використовувати як ранню прототип системи. Бур (1996) представив цікаву концепцію під назвою карта варіантів використання (на основі про концепцію варіантів використання) для вирішення питання динамічного моделювання великого масштабу системи.

Для розробки програмного забезпечення реального часу на основі UML Дуглас (2004, 1999) надав вичерпний опис того, як UML можна застосувати до реального часу системи. Попередня версія методу COMET для проектування одночасного, реального часу, і розподілених програм, які базуються на UML 1.3, описано в Gomaa (2000). Цей новий підручник розширює метод COMET, базуючи його на UML 2, збільшуючи акцент на

архітектурі програмного забезпечення та звертаючись до широкого кола програмні додатки, включаючи об'єктно-орієнтовані програмні архітектури, клієнт/сервер програмні архітектури, сервіс-орієнтовані архітектури, компонентне програмне забезпечення. Огляд архітектури, архітектури паралельного програмного забезпечення та програмного забезпечення в реальному часі та архітектури лінійки програмних продуктів.

У цій главі були представлені об’єктно-орієнтовані методи та нотації, архітектурний дизайн програмного забезпечення та UML. У цьому розділі коротко описано еволюцію методів проектування програмного забезпечення, об’єктно-орієнтованого аналізу та методів проектування, а також одночасних, розподілені та методи проектування в реальному часі.

## 2.11 Висновки.

У другому розділі вирішена задача і дослідження основні методів системного аналізу. У результаті аналізу провідних методологій в області проектування системи можна зробити наступні висновки. У процесі еволюції моделей розробки ПЗ, як було описано в цьому розділі, нові методи не замінили повністю старими. Кожен із створених за цей період методів має свою область застосування . Також може виявитися, що для якоїсь задачі не існує ще оптимального методу проектування. Тоді на основі існуючих методів розробляється новий для вирішення цієї задачі.

Було також показано, що класичні моделі типу "водоспаду", передбачувані четверте визначення вимог до проекту, не працюють ефективно в умовах зміни вимог і жорстких термінів. У таких ситуаціях найбільш ефективними є різні ітеративні підходи, такі як USDP, спіральна модель і COMET. Вони дозволяють швидко розробити працездатний прототип і поступово нарощувати його функціональні можливості. Як виявилося, головне різницю між ітеративними підходами полягає в методі визначення ключових (найбільше) найважливіших вимог до системи. Середи вищеописаних методів для наступного моделювання було обрано метод

COMET як найбільш підходящий метод для системного аналізу заданої об'єктно-орієнтованої розподіленої системи.

У третьому розділі буде описаний системний аналіз, обраний вище методом об’єктно-орієнтованої розподільної системи, яка є системою університету. Аналіз буде представлений з точки зору документів, що циркулюють по цій системі у вигляді перспективного проектування - електронної системи документообороту університету.

# 3 ЕЛЕКТРОННА СИСТЕМА ДОКУМЕНТООБОРОТА З ВИКОРИСТАННЯМ UML

## 3.1 Моделювання вимог

Вибір методу COMET для розробки моделей електронного документообігу в університеті в якості методу аналізу та проектування зазвичай використовується для його адаптації для розробки розподілених прикладних програм і систем реального часу. У цій голові приводиться поетапне проектування моделей, вибраним методом.

Згідно з правилами методу, проектування системи починається з моделювання вимог до майбутньої системи. При цьому сама система розглядається як чорний ящик, враховуються лише її зовнішні характеристики . На етапі моделювання вимог визначаються функціональні - вимоги до системи в термінах акторів і прецедентів. Проводиться аналіз задач - це декомпозиція (розбиття на більш прості фрагменти); та проектування - це синтез, або композиція, рішення.

У розглянутій системі можна виділити такі основні користувачі, як студенти, викладачі, деканати та інші учасники освітнього процесу. Студенти вступають в університет з отриманням диплома. Для досягнення цієї цілі їм доведеться вирішити деякі завдання чи. На рисунку 3.1. наведена діаграма сторони моделювання вимог з використанням нотацій UML, де акторами є студент з однієї та університет з іншої.

Прецедент починається з дії актора і складається із завершеної послідовності події з відомим результатом. Прецедент в залежності від його складності може складатися з інших прецедентів. На рисунку 3.1 показаний прецедент «Получити диплом» і його складові прецеденти, які в свою чергу можуть бути розбиті на ще більш детальну взаємодію за необхідності. У розділі детально наведено основні прецеденти заданої предметної області для її кращого розуміння. У подальшому описі попереднього аналізу системи

моделі будуть лише частково увійти у вид громіздкості всієї системи.

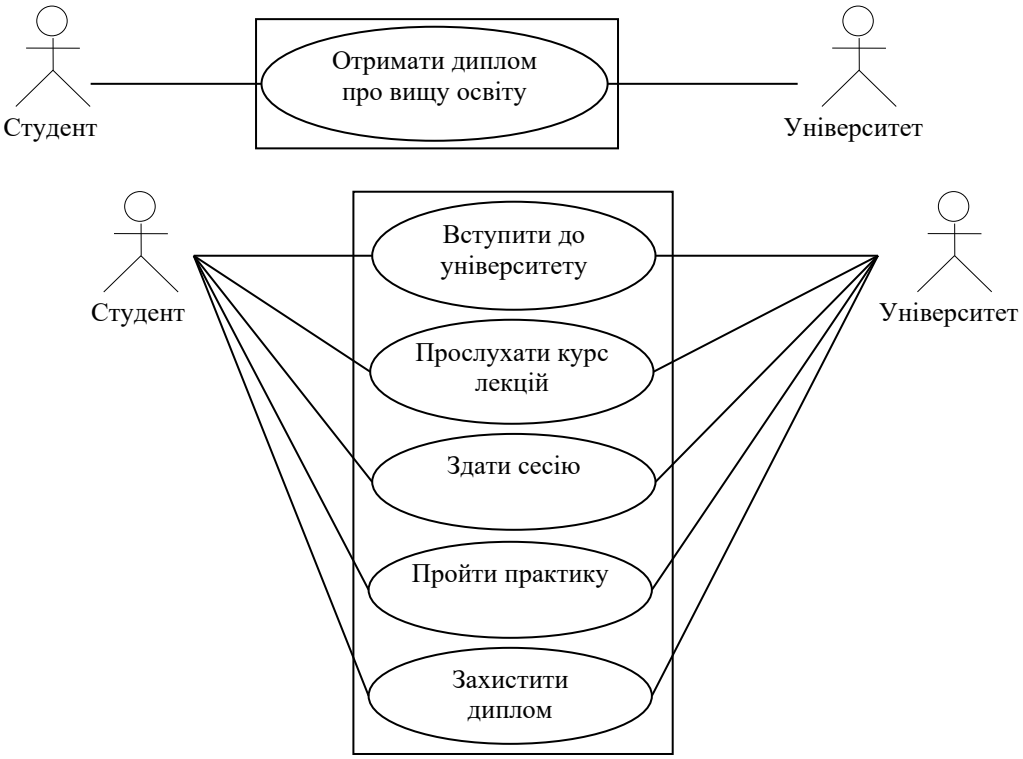


Рисунок 3.1- Прецедент «Получити диплом про вищу освіту»

Кожен прецедент задається своїм ім'ям, коротким змістом, переліком акторів-учасників, за умовами виникнення прецеденту, за поступовими взаємодіями, що приводять до потрібного результату, та іншими найбільш можливими подіями, що приводять до альтернативного результату. Нижче наведено опис прецеденту «Захист дипломів».

Ім'я прецеденту:

Захистити диплом

Зведення: Студент отримує навчання в університеті та отримує диплом

Актор: Студент

Передмова: Студент успішно здав усі екзаменаційні сесії на попередніх курсах навчання та отримує завдання на дипломну роботу. Опис: УМУ розподіляє навантаження викладачеві; Бухгалтерія нараховує зарплату викладачеві; Деканат виділяє наукового керівника; Науковий керівник

виділяє тему; Студент пише диплом; Керівник перевіряє диплом; Деканат призначає ДЕК; Студент захищає диплом; Якщо студент захистив диплом успішно, йому присуджують ступінь.

Альтернативи: - студент не встиг вчасно, і захист переносять на осінь; диплом студента не відповідав вимогам, і студента не допущено до захисту

На рисунку 3.2 наведено аналогічне моделювання вимог до системи зі сторони користувача – викладача.

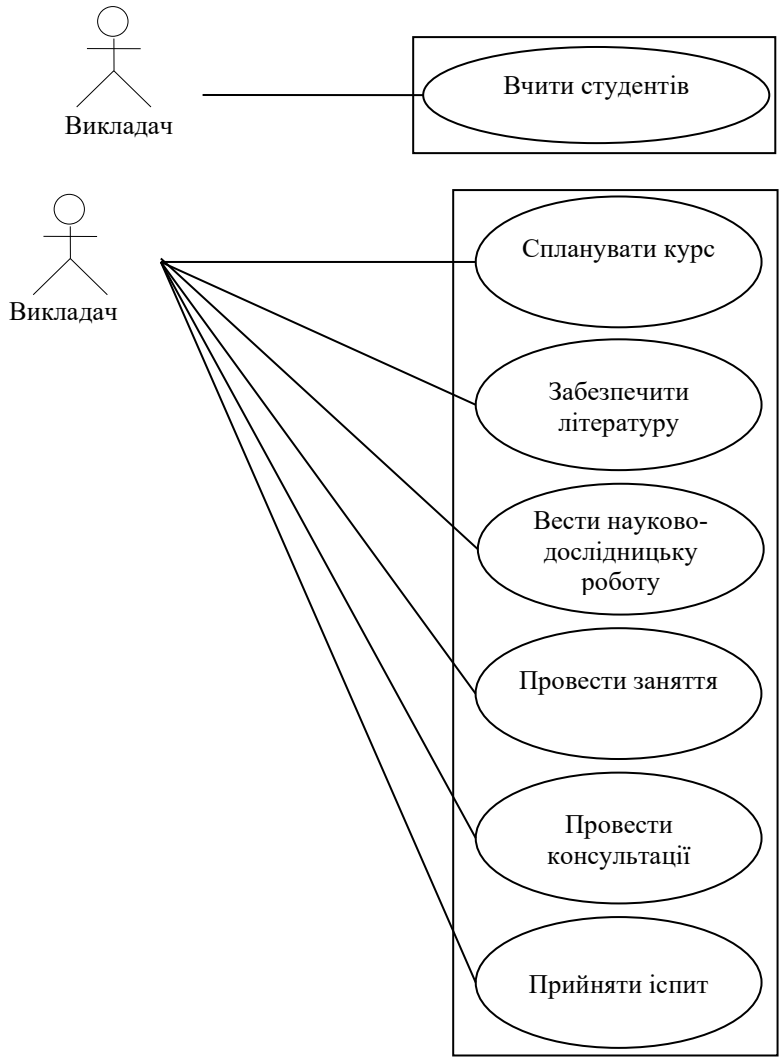


Рисунок 3.2 - Прецедент «Вчити студентів»

У завдання деканату в голові з деканом факультету входить організація взаємодії викладачів і студентів. Отже, прецеденти для актора «декан» і

«деканат» будуть такими, як зазначено на рисунку 3.3.

Такі прецеденти в свою чергу складаються з більш простих і кожен такий прецедент описується окремо. На рисунку 3.4 розписаний прецедент перекладу студентів на наступний курс, який є складовою частиною прецеденту контролю успішності студентів.

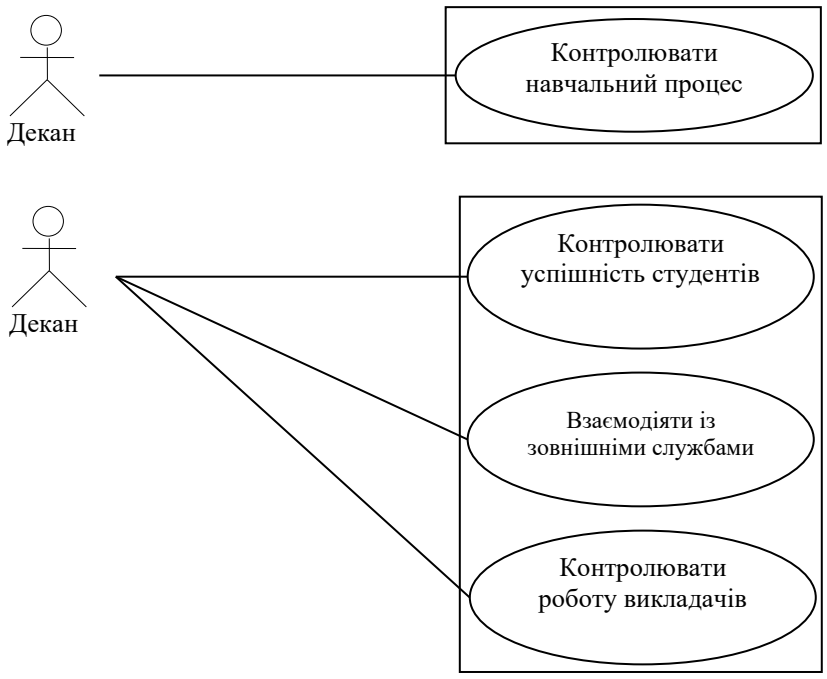


Рисунок 3.3 - Прецедент «Контролювати навчальний процес»

Такі ж прецеденти вказуються для кожного учасника в системі, але ¬всі його вимоги до майбутньої системи. Таким чином, можна задати необхідну функціональність, не розкриваючи внутрішні структури ¬системи.

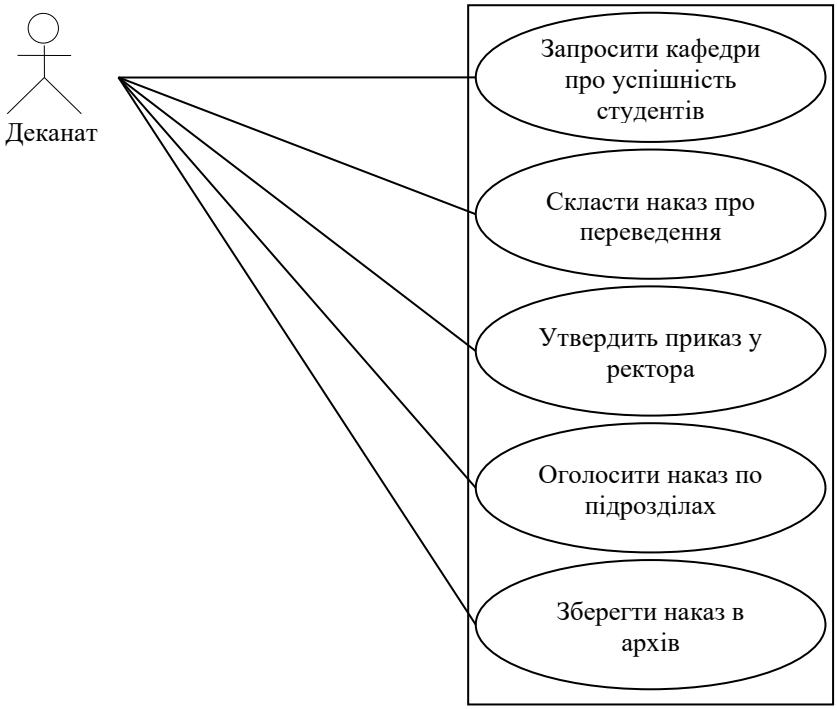


Рисунок 3.4 - Прецедент «Перевести студентів на наступний курс»

## 3.2 Аналітичне моделювання

На етапі аналітичного моделювання розглядається предметна область, будуються статична та динамічна моделі системи. Статична модель задає структурні відносини між класами предметної області. Для опису моделей на цьому етапі використовуються діаграми класів, діаграми кооперації або послідовності, діаграми стану. Модель складається з об’єктів, за критеріями розбиття на об’єкти, і їх взаємодій, виявляється, які об’єкти беруть участь в яких прецедентах .

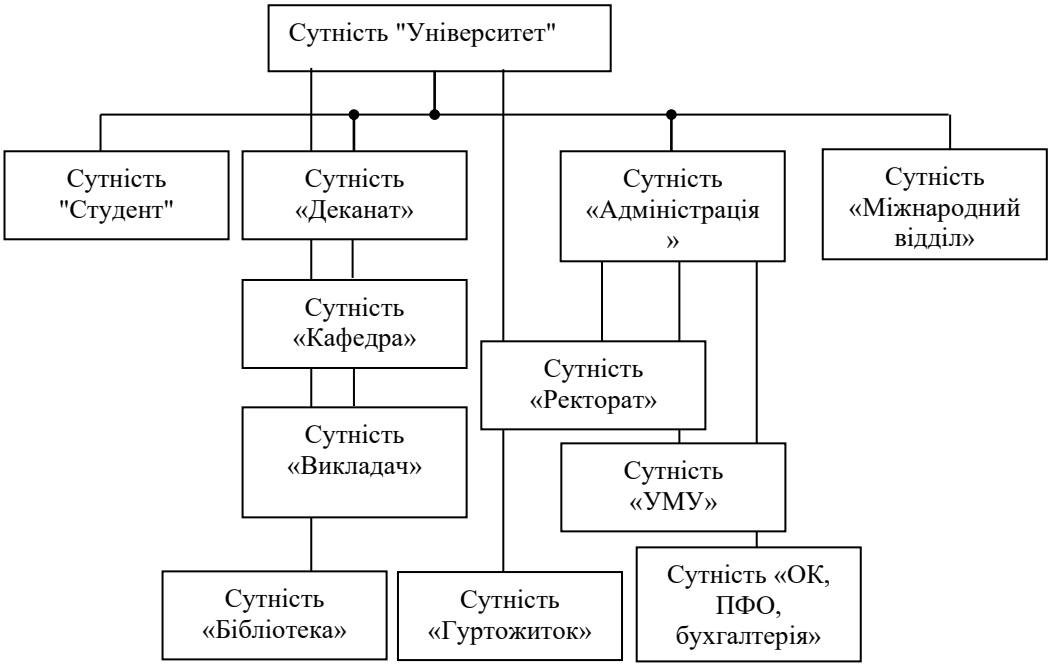


Рисунок 3.5 - Структура університету

На рисунку 3.5 - представлена структура сутності університету, взята на основі аналітичного моделювання.

### 3.2.1 Статичне моделювання

Статичне моделювання є першим кроком етапу аналізу чеського - моделювання. Це структурне представлення інформації про цінні аспекти системи, завдання класів, їх атрибутів і відносини між ними. Зміна статичної структури системи приймається за малоймовірне.

На рисунку 3.6 наведена концептуальна статична модель частини університету. На ней виявлені основні фізичні класи і відносини між ними. В університеті є ректорат, кілька деканатів і багато кафедр. До прикладу, кожен деканат являє собою складовий клас, що складається з класів «Декан», «Секретар» і т.д.

. Ректорат контролює роботу деканатів і кафедр. Кафедри мають у своєму складі викладачів (клас «Кафедра» не вказаний як складовий з метою

спрощення моделі). Викладачі навчають студентів, відчитуються про результати перед деканатом. Студенти контролюються деканатом.

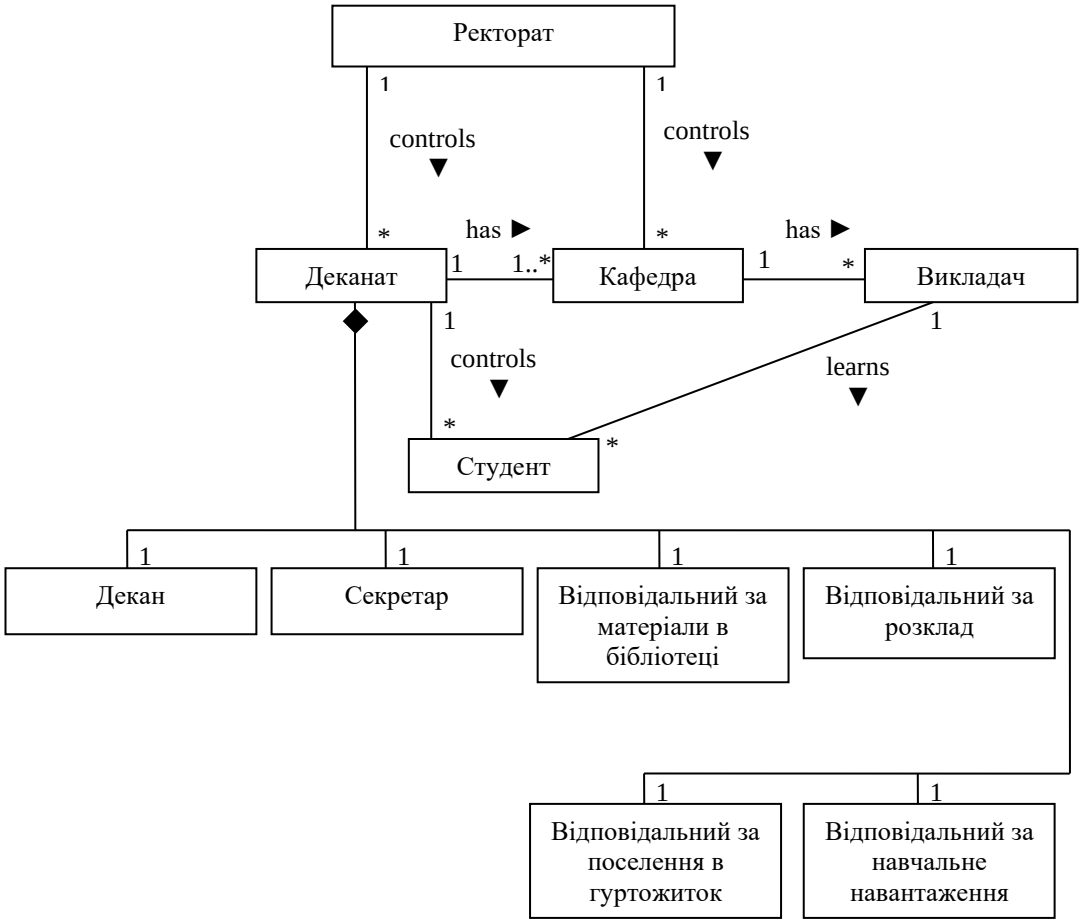


Рисунок 3.6 - Концептуальна статична модель структури університету

Моделювання системи університету проводиться з метою розробки системи електронного документообігу. Тому для уточнення статистики моделі побудована модель із власними класами.

На малюнку 3.7 показана спроектована статична модель нашої предметної області з діаграмами контексту класів електронної системи документообороту. За необхідності модель контексту системи може бути уточнена за допомогою діаграми класів, на яких відображаються інтерфейси між системою та зовнішніми класами. Це потрібно для систем, керуючих зовнішніми пристроями вводу-виводу і зовнішніми системами мами.

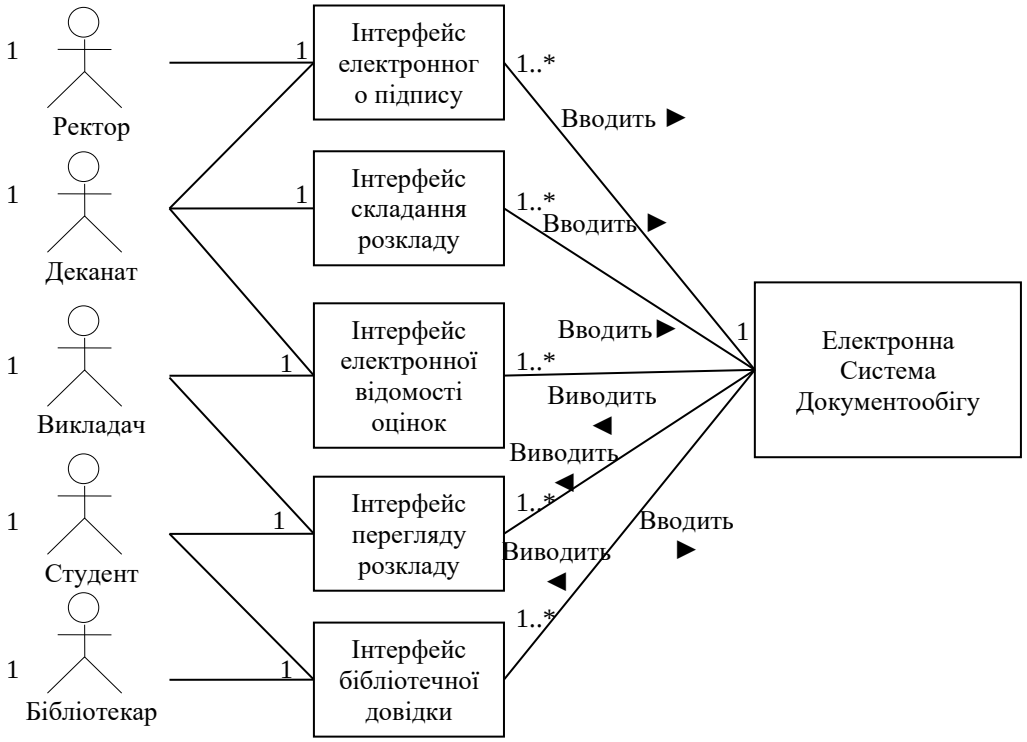


Рисунок 3.7 - Діаграма класів контексту ЕСД

На рисунку 3.8 зображена статична модель класів електронної системи документообороту університету, з відношеннями між інформаційно - насиченими класами, які призначені для зберігання даних . Оскільки в цій системі використовується багато різних даних, основна увага приділяється власним класам. На рисунках представлені класи для найбільш важливих сутностей, таких як «Розклад», «Навчальний план», «Приказ», «Відомість».

Перечень найважливіших класів з їх атрибутами наведено в наступному - розділі опису бази даних.

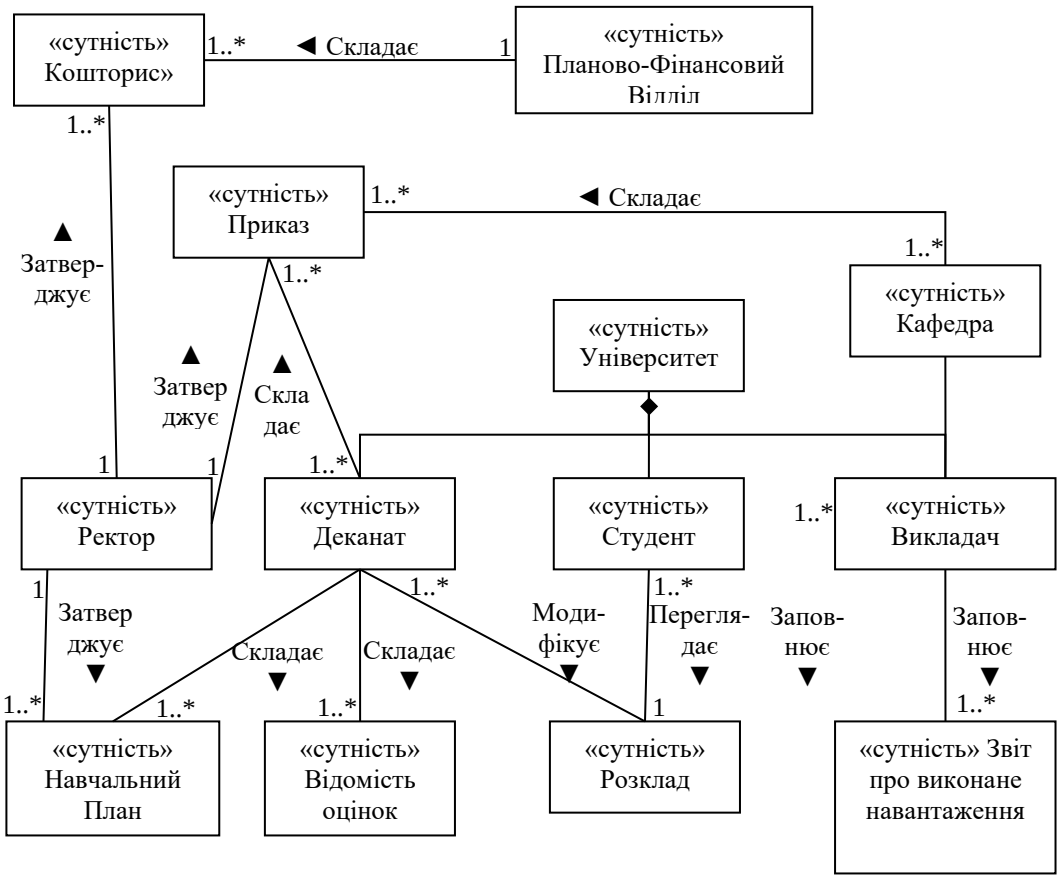


Рисунок 3.8 - Концептуальна статична модель ЕСД з власними класами

### 3.2.2 Декомпозиція на об'єкти

Суть цього етапу аналітичного моделювання складається в розробці програмних об'єктів, що беруть участь в системі. Виявлені на попередніх етапах зовнішні та сутнісні класи в електронній системі документообігу служать для виявлення та категоризації класів і об’єктів, які будуть реалізовані в програмній системі. Об’єкти можуть бути власними, інтерфейсними, керованими та пов’язаними з прикладною логікою.

Під категорією розуміється специфічно визначений розділ у системі класифікації. Таким чином, групуються об'єкти зі схожими моїми характеристиками. Рішення організувати класи в певній групі приймається для покращення розуміння розробляється системи. Щоб відрізняти види класів, застосовуються стереотипи. Стереотип – це під існуючим елементом

моделі класу, який використовується для ознайомлення способу використання або виду класу [11] .

На рисунку 3.9 зображена часткова класифікація класів проектів, що застосовуються в стереотипах «прикладення», «інтерфейс», «сутність », «прикладна логіка». У якості стереотипу «прикладна логіка» приймаються класи алгоритмів складання розписів, розподілу навантажень викладачів, розрахунку смети і інші. Сутностями є бази даних і документи, що поширюються в системі університету. У якості інтерфейсу вибрано стандартний інтерфейс персонального комп'ютера або термінала , тому детально не можна розглядати дану відповідь. Усі нові класи додаються до словника класів проектованої системи, створеного на етапі статичного моделювання.

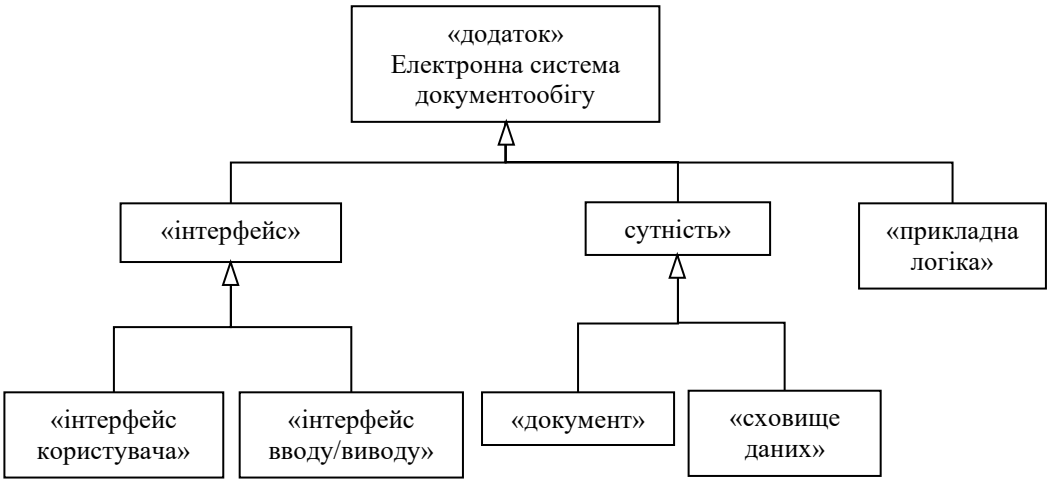


Рисунок 3.9 - Стереотипи класів додатки ЕСД

Разом із ідентифікацією класів проводиться класифікація об'єктів - системи. За критеріями класифікації об’єкти поділяються на інтерфейсні , сутнісні, керуючі та об’єкти прикладної логіки. Інтерфейсні об'єкти - реалізують інтерфейс системи із зовнішньою середою: устаткуванням , користувачем або іншою системою. Сутнісні об'єкти – це об’єкти з рідко змінюваною структурою. В них зберігається інформація. Керуючі об’єкти координують роботу інших об’єктів залежно від стану чи події. Об’єкти

прикладної логіки використовуються при наявності деяких алгоритмів обробки даних, наприклад алгоритму складання розписів за навантаженням викладачів і типу вільної аудиторії.

Після визначення класів і об'єктів-учасників система розбивається на - підсистеми. Система університету з розробкою на підсистеми наведена на рисунку 3.10. Ціль такого розбіжності полягає в тому, щоб розподілити сильно пов'язані об'єкти в одну підсистему, а слабо пов'язані – в різні. В рамках однієї підсистеми об'єкти мають загальну функціональність. Зазвичай в одну і ту ж підсистему потрапляють об'єкти, що беруть участь в одних і тих же прецедентах. Остаточно система розбивається на підсистеми вже після проведення динамічного моделювання, на одному з ранніх етапів етапу проектування.

Так, система університету розбивається на окремі підсистеми. Підсистема розписів включає в себе класи й об'єкти, спільна основна функція яких міститься в попередньому зборі інформації, складанні та видачі навчального розпису. Підсистема електронної відомості складається із засобів збору, зберігання та обробки інформації про результати екзаменаційних сесій. Підсистема документів електронного підпису повинна мати спеціальні засоби для затвердження приказів, навчальних планів, кошторисів та інших офіційних документів, які є основними засобами взаємодії між різними підрозділами університету.

У завершенні етапу статичного моделювання в досліджуваній системі мого університету повинні бути виділені клієнтські та серверні частини. Клієнтом є та частина системи, яка надсилає запити на серверну частину. Сервер відповідно обробляє запит і видає результат. В системному університеті клієнтом у більшості прецедентів є деканат. Деканат виправляє запити на кафедру про успішність студентів і про виконання попередньої - навантаження, на початку кожного семестру заявку в центрі складання - розписів, у бібліотеці – про наявність друкованих матеріалів. Відповідно об'єкти з іншою цією взаємодією є серверами.

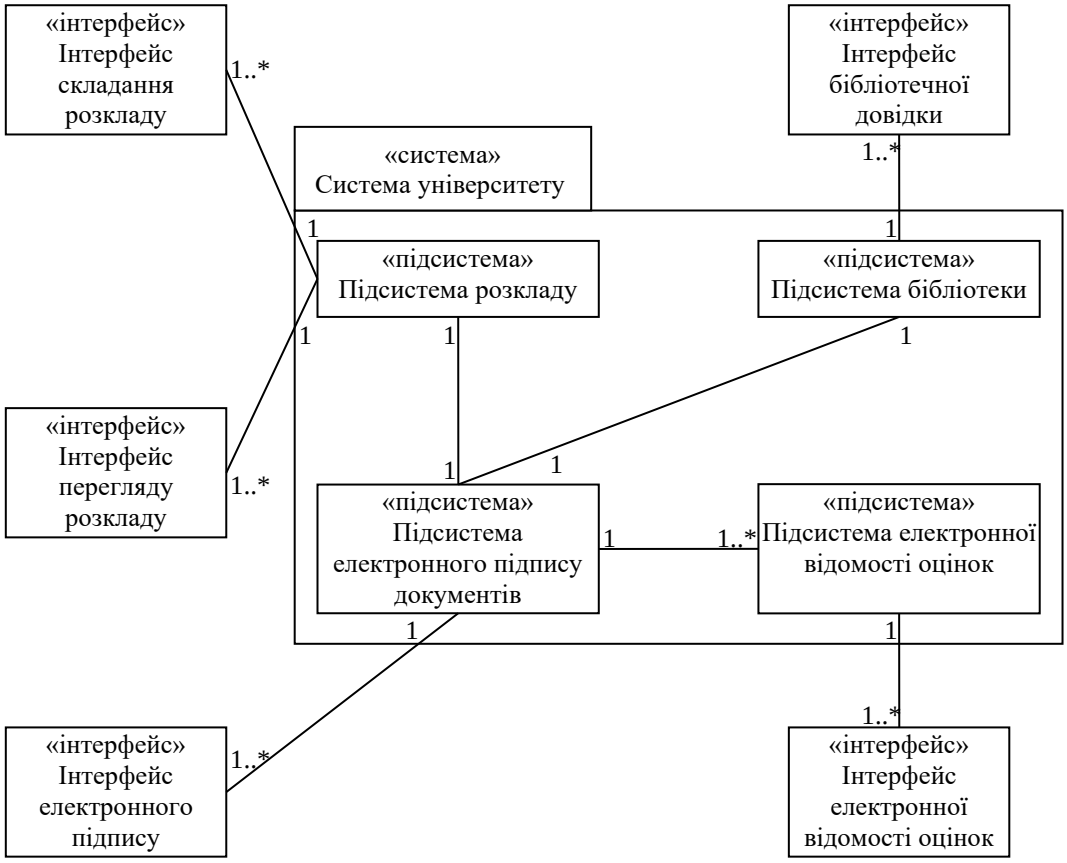


Рисунок 3.10 - Основні підсистеми та їхні зв'язки в системі університету

### 3.2.3 Моделювання стану

Є багато систем, функціонування яких визначається не тільки вхідними даними, але ще і станами системи на попередніх кроках. При проектуванні такої системи необхідно моделювати також її стан. Діаграми станів відображають динамічні аспекти системи. На цьому етапі система моделюється як кінцевий автомат. Скінченним автоматом називається автомат зі скінченним числом станів, що перебуває в будь-який момент часу лише в одному з них [11] .

Моделювання станів здійснюють за допомогою ієрархічних діаграм станів, де задають аспекти системи, що залежать від стану. На рисунку 3.11 представлена діаграма, складена для прецеденту «Перевести студентів на наступний курс» зі сторони підсистеми видання приказів.

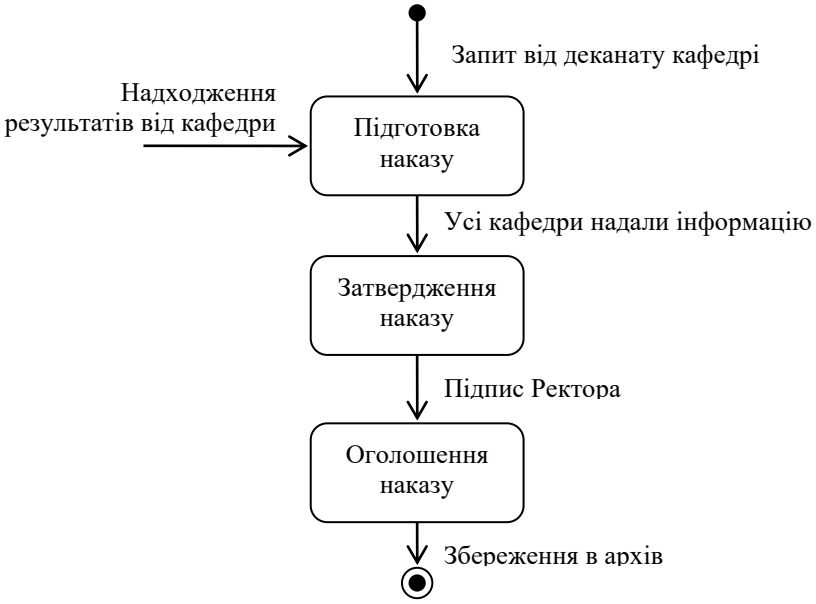


Рисунок 3.11 - Діаграма станів для підсистеми видання приказів

Спочатку діаграма, зображена на малюнку 3.11, перебуває в стані «Підготовка наказу». Перехід у початковий стан цієї діаграми відбувається при виникненні події «Запит деканату кафедрі». Наступні події «Надходження результатів від кафедри» не змінюють стану «Підготовка наказу» за умови, що не всі кафедри ще надали інформацію. Однак подія «Усі кафедри надали інформацію», викликає перехід у стан «Затвердження наказу". У свою чергу, подія «Підпис ректора» переводить систему в стан «Оголошення наказу». Після збереження наказу в архіві здійснюється перехід у кінцевий стан.

### 3.2.4 Динамічне моделювання

Завершальним етапом аналітичного моделювання є дина ми чеське моделювання, де досліджуються приведені властивості проектованої системи. Динамічні моделі описуються не тільки взаємодією об'єктів з іншими, але і поведінкою, що залежить від стану об'єктів. Як вихідні дані, на цьому етапі використовуються отримані раніше прецеденти. Для кожного

прецеденту визначаються наявні в нього об'єкти, їх взаємодія та динамічні властивості.

Для моделювання предметної області застосовують один із видів діаграм взаємодії: діаграми кооперації або діаграми послідовності. Обидва різновиди діаграм представляють одну й ту саму інформацію в різному вигляді. Тому для виконання динамічного моделювання достатньо використовувати лише один вид діаграм.

Розглянемо рисунок 3.12, на якому зображено діаграму кооперації за прецедентом "Перевести студентів на наступний курс". Діаграма кооперації моделює послідовність участі об'єктів у прецеденті та вказує взаємодії між ними. Порядок проходження повідомлень, що відповідає опису прецеденту, позначається номерами. Порядкові номери повідомлень проставляються за визначеними стандартом UML правилами. Спочатку може слідувати рядок букв, що позначає прецедент, до якого належить ця діаграма кооперації. Далі йде порядковий номер події в послідовності повідомлень. Зовнішні події, що ініціювали ланцюг взаємопов'язаних подій, пронумеровані цілими числами. Номер внутрішньої події складається з номера зовнішньої події, що викликала її, і далі через крапку власного послідовного номера всередині цього ланцюга. Якщо в якомусь місці ланцюг повідомлень розгалужується, то головній послідовності присвоюють звичайну нумерацію, а до номера додаткової послідовності додають маленьку літеру за алфавітом. Наприклад, на малюнку 3.12 зовнішня подія "Наказ затверджено" має порядковий номер 4. Об'єкт "Інтерфейс електронного підпису документів" у відповідь на це повідомлення видає 3 паралельні повідомлення "Роздрукувати документ", "Зберегти документ" і "Розіслати всім". Ці альтернативні повідомлення з'являються паралельно і мають відповідно назви 4.1.a, 4.1.b і 4.1.c*. Повідомлення "Розіслати всім" розділяється на стільки копій, скільки відділів потрібно повідомити про видання нового наказу. Тому в його назві праворуч стоїть символ "*", що свідчить про множинну передачу повідомлення.

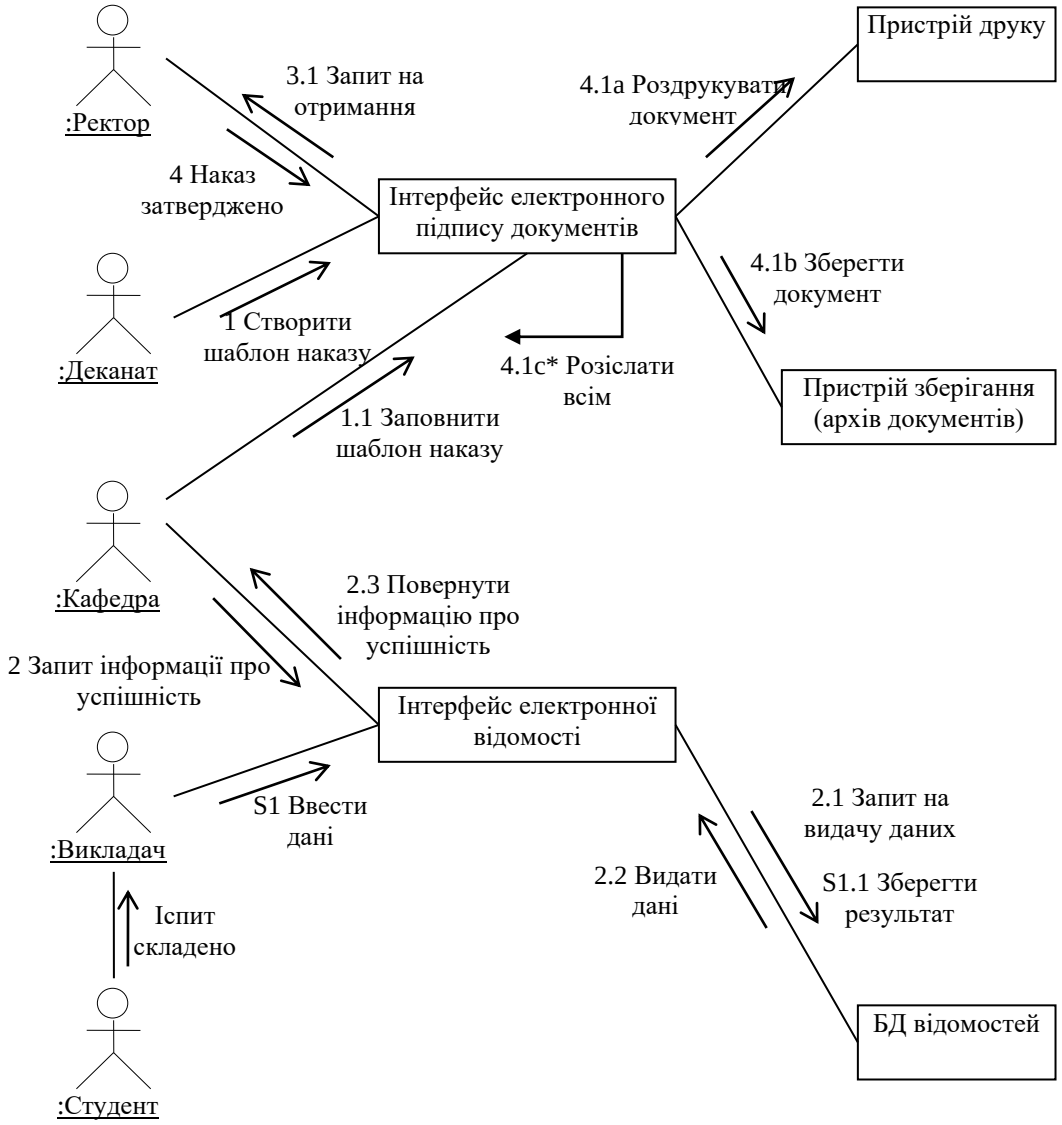


Рисунок 3.12 - Діаграма кооперації для підсистеми видання приказів за прецедентом «Перевести студентів на наступний курс»

У свою чергу діаграма послідовності показує ланцюг взаємодій в залежності від часу. На ньому вказуються об'єкти взаємодії та порядок відправки повідомлень. Повідомлення на діаграмі відображаються зверху вниз у порядку їх слідування, тому нумерувати їх не обов'язково. На діаграмах кооперації важко прослідкувати порядок слідування повідомлень. З іншого боку, недолік діаграми наслідків полягає в утрудненому розумінні зв'язків об'єктів між собою, особливо при великому числі взаємодіючих об'єктів .

## 3.3 Проектне моделювання

За аналітичним моделюванням слідує етап проектного моделювання. На цьому етапі йде розробка програмної архітектури системи. Таким чином, виконується задача відображення аналітичної моделі на проектну, а від предметної області здійснюється перехід до області рішення. На етапі проектного моделювання проводять декомпозицію системи за критеріями розбиття на підсистеми, де підсистеми далі розглядають як складові об'єкти, які взаємодіють шляхом обміну повідомленнями.

Для паралельних додатків, у тому числі розподілених і прикладе самого реального часу, виконуються наступні види дійсності. Повинна бути виконана консолідація моделі кооперації об'єктів. Така модель відображається консолідованими діаграмами кооперації. Далі визначається структура підсистем та інтерфейсів між ними зі створенням загальної архітектури програм і розділеними додатками на підсистеми.

Розподілений додаток розбивається на розподілені підсистеми, що проектуються як компоненти, що конфігуруються. Проектування архітектури розподіленої програми для розподіленого додатка відбувається за допомогою розподілу системи на розподілені підсистеми і вказівки інтерфейсів передачі повідомлень між ними. Після чого йде визначення характеристик об'єктів, зокрема пасивності та активності. Кожна підсистема розділяється на паралельні завдання (активні об'єкти) за допомогою критеріїв розбиття на завдання, при цьому для кожного завдання вказуються інтерфейси.

Уточнюються характеристики повідомлень, - які з них є синхронними (з відповіддю або без відповіді), а які - асинхронними. Проектують класи, що приховують інформацію (пасивні класи) для кожної підсистеми, задають операції кожного класу і параметри кожної операції. Розробляється детальний проект програми з урахуванням синхронізації завдань і обміну інформацією між ними, а також внутрішньої структури паралельних завдань.

Таким чином, на етапі проектного моделювання виконуються наступні -

кроки:

1) проводиться розробка аналітичної моделі, що включає синтез аналітичної моделі ческої моделі для отримання попередньої архітектури програми . На цьому етапі здійснюється перехід від аналізу до проектування . Якщо на стадії синтезу виявляються помилки, то відбувається повернення до етапу аналітичного моделювання. Сюди включається синтез діаграми, що складається з заснованих на прецедентах частинних діаграм, що складаються з кожного залежного від стану об’єкта. В результаті отримується загальна діаграма, що складається з даного об'єкта;

2) виконується синтез моделей кооперації об'єднанням усіх діаграм кооперацій в єдину модель кооперації для системи в цілому. Для більших систем розробляється модель кооперації підсистем. На цьому етапі може бути здійснено повернення з кроку 4;

3) відбувається синтез статичної моделі. Визначаються класи та відносини між ними. Уточнена статична модель є результатом, отриманим на основі статичної моделі предметної області. Якщо вона була надто концептуальною, то синтез проводиться на основі об'єднаної моделі кооперації;

4) проектується загальна архітектура системи. Додатки розбиваються на підсистеми, вказуються інтерфейси підсистем. Для виділення підсистем застосовуються критерії розбиття на підсистеми. Для кожної підсистеми розробляють діаграми кооперації та високорівневу діаграму кооперації для системи загалом;

5) проектується архітектура програми, заснована на розподілених компонентах. Для розподілених додатків розбиття на розподілені підсистеми відбувається на основі критеріїв розподіленості. Потрібно також визначити інтерфейс між компонентами системи на основі обміну повідомленнями. Цей крок необхідний тільки під час проектування розподілених додатків;

6) проектується архітектура паралельних задач для кожної підсистеми теми. Для цього виконується розбиття підсистеми на паралельні

задачі чи з критеріями декомпозиції та визначення цих задач та їх інтерфейсів. Далі слід розробка для кожної підсистеми діаграми паралельних кооперацій, на яких відображаються завдання та їх інтерфейси . Завершує цей крок документування проекту кожного завдання у вигляді специфікації поведінки завдання;

7) проводиться аналіз продуктивності проекту. Для цієї мети використовується метод планування в реальному часі, щоб визначити, наскільки проект відповідає вимогам щодо продуктивності. При необхідних - мостах можна кілька разів повторити кроки 5 і 6. Цей крок потрібен тільки при проектуванні додатків реального часу;

8) у кожній підсистемі проектуються класи, їх інтерфейси, операції та ієрархії спадкування. Для програм, якими для роботи необхідна база даних, створюється структура бази даних і класи-обгортки. Для кожного класу потрібна специфікація інтерфейсу. Проектування такої бази даних наведено в наступному розділі опису інформаційної системи деканата;

9) розробляється детальний проект кожної підсистеми. Розробка складається з кількох етапів. Створюється внутрішня структура складових задач, які містять вкладені пасивні об'єкти. Уточнюються деталі механізмів синхронізації для об'єктів, до яких можливий доступ зі сторони кількох завдань. Виділяються класи-роз'єми, інкапсулюючі деталі міжзадачної комунікації. Для кожної задачі виявляється і документується внутрішня логіка виникнення та обробки подій;

10) виконується більш детальний аналіз продуктивності кожної підсистеми додатків реального часу. У разі необхідності можна повторити кроки 6-9. Цей крок потрібен тільки при проектуванні додатків реального часу.

На етапі інкрементного конструювання ПЗ визначається, яку підмножину системи буде реалізовано на кожному кроці, з урахуванням прецедентів і об'єктів. Етап охоплює детальне проектування, кодування і тестування кожного з класів, що увійшли до відібраної підмножини. Такий

поетапний підхід дає змогу поступово створювати й об'єднувати фрагменти системи, доки її не буде зібрано повністю.

Етап інкрементної інтеграції передбачає тестування інтерфейсу всіх створених фрагментів. Система розглядається як прозорий ящик. Таким чином, система послідовно розширюється, при кожному розширенні формується інкрементний прототип. Якщо на якому-то розширенні виявлені помилки проектування, можна повернутися до етапів моделювання вимог, аналітичного або проектного моделювання.

Наводиться етап комплексного тестування про функціональне тестування системи в цілому, то є перевірка на відповідність функціональним вимогам . На цьому етапі система розглядається як чорний ящик.

## 3.4 Висновки

У третьому розділі була вирішена задача побудови моделей системи електронного документообігу та сервісів в університеті обраним методом. Для цього використовувався об'єктно-орієнтований аналіз і проектування паралельної системи університету за допомогою методу COMET . Аналіз дозволив виявити набір підтримуваних операцій за допомогою моделювання прецедентів. Статична структура системи була визначена об'єктами та їх відносинами. Для повноти аналізу, особливо для простеження динаміки документообігу в університеті, проведено динамічне моделювання предметної області. Робота описуваної системи, як і більшість систем, залежить від стану в поточний момент часу, що показало моделювання стану. Різні об'єкти відправляють повідомлення і викликають перехід в той чи інший стан. Динаміка переходів виражена в діаграмах кооперації на прикладі динамічного моделювання прецеденту «Перевести студентів на наступний курс».

# 4 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ДЕКАНАТ

## 4.1 Концептуальна структура бази даних

Для проектування та програмування повноцінної електронної системи нашого документообігу в університеті потрібна багаточисленна команда розробників. У теперішньому дипломі показана розробка одного з її модулів. Приведене в четвертому розділі проектування бази даних деканата, як основна структурна складова університету, є однією із задач, поставлених перед цією дипломною роботою. База даних – це спільно використовуваний набір логічно пов’язаних даних (і опис цих даних), призначений для задоволення інформаційних потреб організації [15]. Основна ціль проектування БД – це скорочення обсягу збутових даних, економія обсягу використання моєї пам’яті, зменшення витрат на багатократні операції оновлення вихідних копій і усунення можливості виникнення суперечностей із-за зберігання в різних місцях, зведених в одному і тому ж об’єкті.

На рисунку 4.1 показані рівні проектування бази даних. На рівні логічного проектування вирішується задача відображення об'єктів предметної області в абстрактних моделях об'єктів даних. Це відображення не повинно суперечити семантиці предметної області та бути найбільш ефективним. А на фізичному рівні забезпечується ефективність виконання запитів до бази даних. В залежності від особливостей конкретної СУБД розміщуються дані у зовнішній пам'яті, створюються додаткові структури (наприклад, індекси). На першій стадії проектування бази даних використовується семантичне моделювання. У термінах семантичних моделей розробляється концептуальна схема бази даних, яка потім вручну перетворюється до реляційної схеми. Реляційне проектування починається з визначення необхідних сутностей.

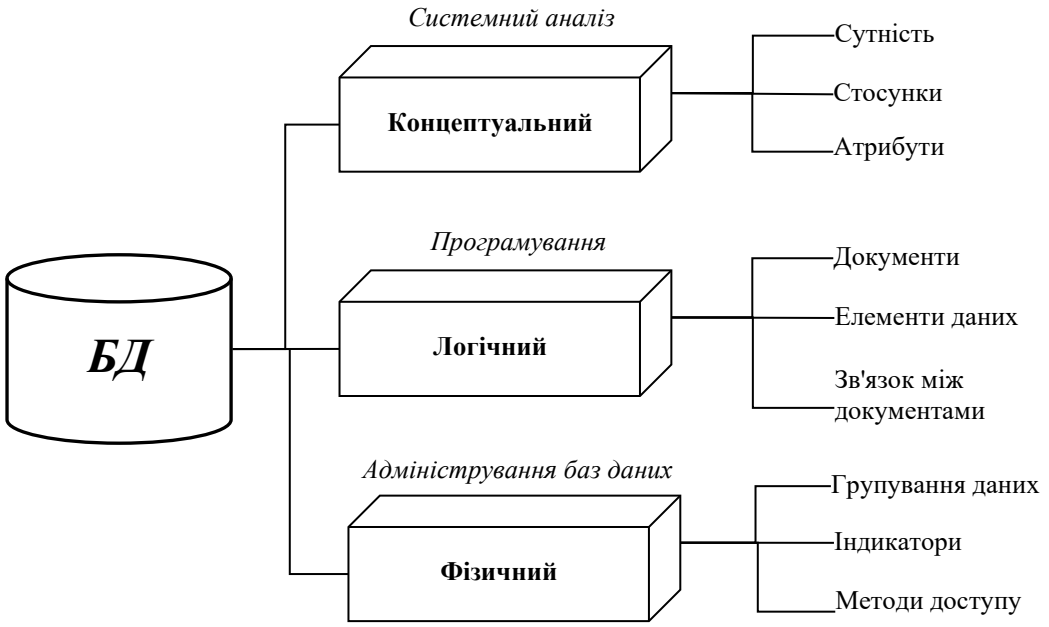


Рисунок 4.1 - Рівні проектування баз даних

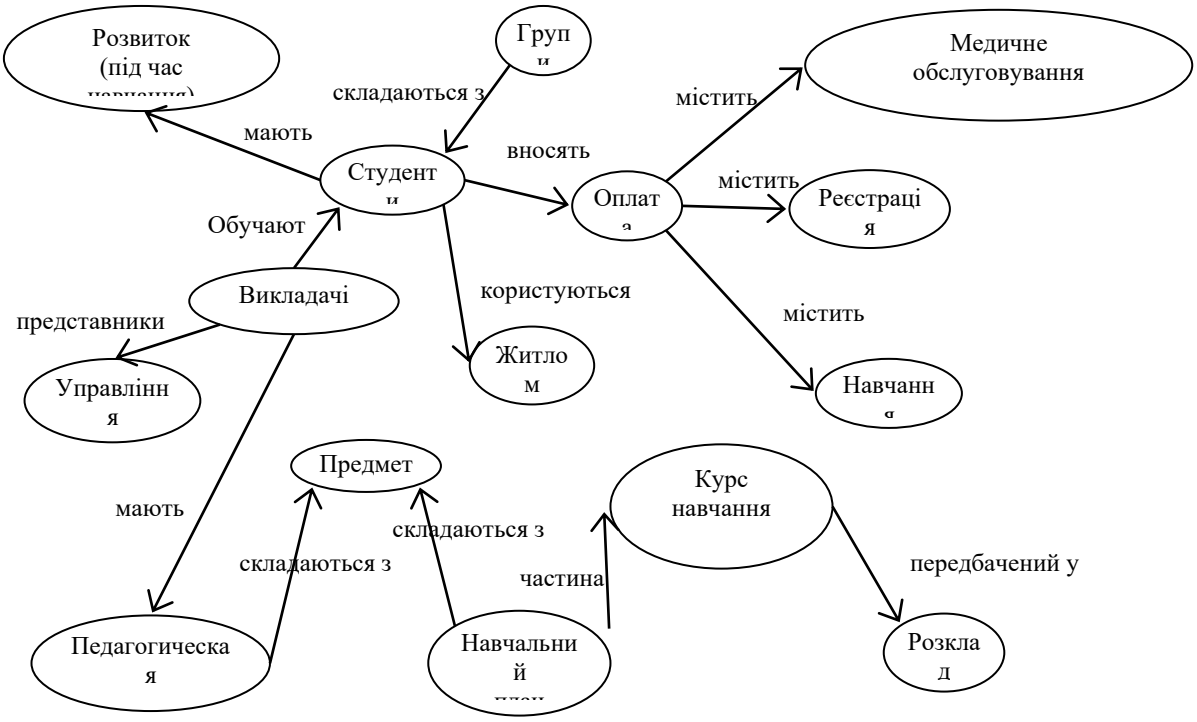


Рисунок 4.2 - Концептуальна схема

В університеті такими сутностями можуть бути студенти, викладачі, курси, спілкування, медобслуговування, оплата, прикази.

На рисунку 4.2 представлена концептуальна модель бази даних - інформаційної системи деканата. Реляційна база даних міститься у вигляді набору таблиць (сутностей), в яких зберігаються дані.

## 4.2 Логічна структура бази даних

Виду об'ємності графічного опису логічна схема для детальності - описана в текстовому вигляді з прикладами. Для логічного режиму створення об'єктної області зручно всього використовувати ER-модель (модель об'єкт-відношення). У діаграмах ER-моделі сутність представлена у вигляді прямокутника, що містить ім'я сутності. Сутності з'єднуються зв'язками, на кінцях яких виявляється, скільки екземплярів даної сутності можуть брати участь у зв'язку. У зв'язку може бути також включена асоціація (сутність, що зв'язує) у вигляді шестикутника. Атрибути сутностей позначаються овалами. У вигляді трапецій вказуються також сутності, що характеризують інші сутності. Для зображення моделі бази даних можна використовувати такі CASE- засоби, як ERwin (компанії Computer Associates), System Архітектор ( Popkin Software), PowerDesigner (Sybase), EasyCASE, EasyER (Visible Systems), ER / Studio (Embarcadero), Designer 2000 (Oracle) [16] .

Відомості про студентів університету зберігаються в таблиці з іменем [Students]. Кожен рядок у цій таблиці описує окрему сутність такими атрибутами (позначеними на рисунку 4.3): номер залікової книжки студента [Stud_RB], номер студентської картки [Stud_card], ім'я студента [Name], номер паспорта [Passport], дата народження [Birthday], адреса в Україні [Ukraine_Addr], номер телефону [Phone], адреса електронної пошти [Email], адреса на Батьківщині [Country], рік зарахування [Adm_yr] та оплата за навчання за договором [Tuition_fee]. Первинним ключем, тобто атрибутом,

який унікально ідентифікує всі інші атрибути в будь-якому рядку таблиці [Students], є номер залікової книжки студента [Stud_RB]. Усі інші атрибути функціонально залежать від первинного ключа. Оскільки первинний ключ має бути унікальним, у ньому не допускаються порожні значення.

Атрибут [Name] можна вважати вторинним ключем у таблиці [Students]. Вторинний ключ використовується тільки для пошуку даних, якщо користувачеві невідоме значення первинного ключа - номера студентського квитка. На відміну від первинного ключа, вторинний ключ не вимагає унікальності. Одне й те саме значення поля [Name] у таблиці [Students] може зустрітися неодноразово, але завдання значення вторинного ключа значно звужує область пошуку потрібного студента. З отриманого набору сутностей з однаковим значенням атрибута [Name] можна знайти потрібного студента, наприклад, за датою народження. При цьому автоматично визначається значення первинного ключа за цим студентом. Знаючи первинний ключ, можна отримати всі дані щодо студента.

Номер залікової книжки визначається під час підготовки наказу про зарахування абітурієнта до університету. Номер наказу про зарахування [No of Adm Order], номер залікової книжки студента [Stud_ Id] і дату зарахування [Date_Adm] представлено в таблиці з назвою [No of Admission order]. Аналогічно зберігаються дані про переведення з курсу на курс і відрахування студентів у таблицях [No of Admission order], [No Transfer Order] and [No of Dismissal order]. У цих таблицях номер залікової книжки студента не є унікальним, у кількох рядках таблиці може бути одне й те саме значення атрибута [Student ID]. Цей атрибут у таблицях [No Transfer Order] and [No of Dismissal order] є зовнішнім ключем стосовно таблиці [Students]. Це означає, що значення зовнішнього ключа [Student ID] у таблицях із даними за наказами збігаються зі значеннями первинного ключа [Stud_RB] у таблиці [Students].

Foreign key однієї таблиці містить таке значення, яке посилається на наявний дійсний кортеж у другій таблиці. Таким чином, забезпечується

цілісність на рівні посилання між таблицею [Students] та іншими таблицями. Одному рядку таблиці [Students] можуть відповідати від нуля до кількох рядків в інших таблицях. Тому для опису відношення між таблицею [Students] і, наприклад, таблицею [No of Admission order] використовується тип зв'язку один-ко-множиною (1:∞). Such attributes like [No of Adm Order], [Date_Adm], [No of Dismis order], [Date_Dismiss], [No Trans_Order] and [Date_Trans] у відповідних таблицях можна використовувати для пошуку даних.

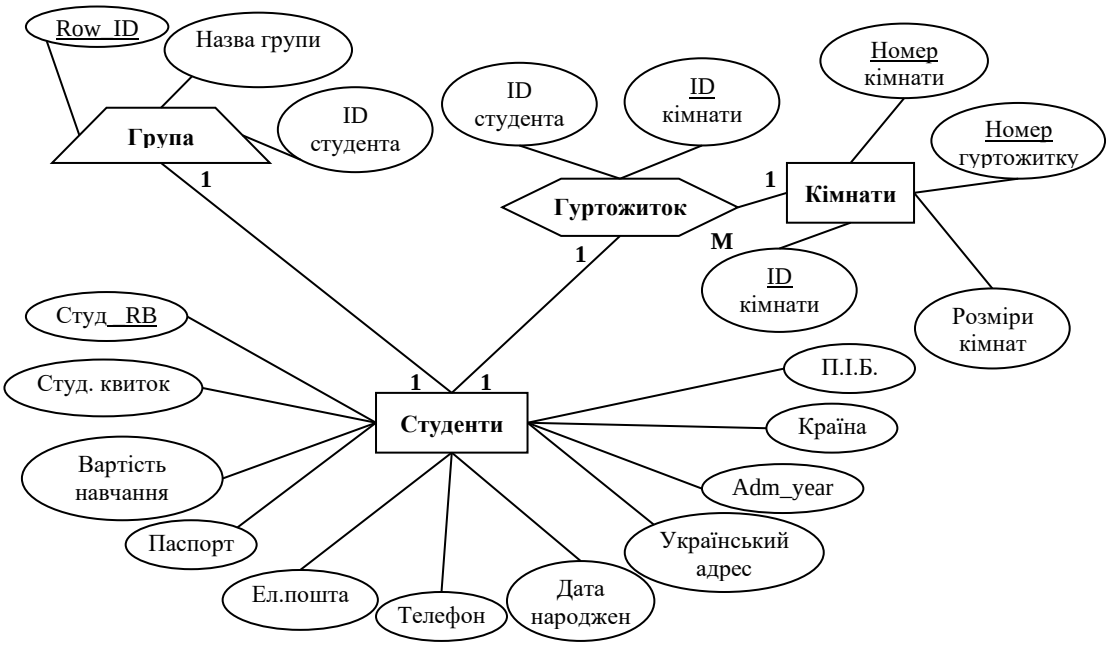


Рисунок 4.3 - ER-діаграма області даних

В університеті студенти розподілені за групами. У таблицю [Group] заносяться найменування групи [Group_name] and numbers of student record books [Student_ID], як показано на малюнку 4.3. The attribute of number of student record book у таблиці [Group] є зовнішнім ключем стосовно таблиці [Students]. Для опису відношення між таблицею [Students] і таблицею [Group] використовується тип зв'язку один-ко-множиною (1:∞).

Атрибут назви групи [Group_name] може використовуватися як

вторинний ключ для пошуку даних. Для зменшення об'єму збережених даних і підвищення швидкості введення даних рекомендується винести найменування всіх груп в окрему таблицю, в якій кожному рядку з назвою групи поставити у відповідність унікальний номер. Цей ідентифікатор потрібно використовувати в таблиці [Group] замість атрибута [Group_name] для вказівки до якої групи належить той чи інший студент.

Іногородні студенти мають бути поселені в гуртожиток за кілька днів перед початком навчального року. За цей короткий термін потрібно розселити велику кількість студентів. У такому разі необхідна електронна електронна картотека по гуртожитках. Підсистема з обліку місць у гуртожитку складається з трьох таблиць. Перша таблиця з назвою [Hostel] зберігає дані про номери гуртожитків [Hostel_No], адреси гуртожитків [Hostel Address], ім'я керівника кожного гуртожитку [Head_Hostel] і контактний номер телефону гуртожитку [Phone_No].

У таблицю [Rooms] занесено дані щодо місткості гуртожитків. У ній присутні такі атрибути, як номер гуртожитку [hostel_No], номер кімнати в гуртожитку [Room_No], кількість студентів у кожній кімнаті [Room capacity]. Крім того, кожній із кімнат по всіх гуртожитках присвоюється унікальний номер [ID_room]. Цей ідентифікатор є первинним ключем у таблиці [Rooms]. На малюнку 4.3 сутність [Habitant] показано як асоціацію між сутностями [Rooms] і [Student]. У третю таблицю - таблицю [Habitant], занесено дані про заповненість кімнат у гуртожитках. За унікальним номером кімнати [ID_room] записується один або кілька numbers of student record books [Student_ID]. Між таблицею [Rooms] і таблицею [Habitant] використовується тип зв'язку один-многим (1:∞), тобто в таблиці [Habitant] може зберігатися по кілька рядків, що належать до єдиного рядка таблиці [Rooms].

Для обліку поселення абітурієнтів або учасників конференцій, які не є студентами і не мають залікової книжки університету, можна додати атрибут імені в таблицю [Habitant]. Тоді поле, що ідентифікує студента, у цьому разі залишатиметься незаповненим. У всіх інших випадках під час занесення

даних у цю таблицю для студента заповнюється поле номера залікової книжки, а поле імені залишається порожнім.

Іноземні студенти, які навчаються в університеті, зобов'язані регулярно проходити перереєстрацію. Для обліку зареєстрованих студентів створено таблицю [Registration], до якої для кожного студента вносять student record book number [Student_ID], date of registration [Date], deadline of registration [Deadline]. Student record book number [Student_ID] є зовнішнім ключем стосовно таблиці [Students]. Оскільки в таблиці [Registration] зберігаються дані за всіма реєстраціями, які проходив іноземний студент, то поле номера залікової книжки в таблиці [Registration] не може бути унікальним. Іншими словами, одному рядку таблиці [Students] відповідає кілька рядків у таблиці [Registration], тобто між цими таблицями існує зв'язок типу один-многим.

Одними з найважливіших даних є дані щодо оплати. Студенти, які навчаються за контрактом, і студенти, які проживають у гуртожитку, мають у встановлений термін здійснювати оплату. Дані з оплати зберігаються в таблицях [Tuition pay't] and [Hostel pay't]. В обох таблицях присутні поля student record book number [Student_ID], date of payment [Date], amount paid [Amt_Paid] and deadline of current payment [Deadline]. У таблицях оплати зберігаються всі здійснені студентом оплати протягом терміну навчання, тому між таблицею [Students] і таблицями [Tuition pay't] and [Hostel pay't] використовується тип зв'язку один-ко-множиною (1:∞).

Безсумнівно, основною діяльністю студентів в університеті є навчання. Вони вивчають спеціальні та загальні навчальні курси за обраною програмою, складають сесію наприкінці кожного семестру, отримуючи підсумкові оцінки. Викладачі навчають студентів і приймають у них іспити. Для збору інформації щодо викладачів існує таблиця [Lecturers]. У ній зберігається вся необхідна для організації навчального процесу інформація. Є ім'я викладача [Name], його домашня адреса [Address], номер телефону [Phone], адреса електронної пошти [Email], основна кафедра [Department], посада викладача [Position], академічний ступінь [Degree]. Для ідентифікації

та зв'язку з іншими таблицями кожному рядку з даними щодо окремого викладача присвоєно унікальний номер [Teacher_ID]. Він же і є первинним ключем таблиці [Lecturers].

У таблиці [Subjects] зберігаються дані для предметів, що викладаються студентам: назва предмета [Sub_Name], тип семестрового контролю (тест або іспит) [Sem_ctrl], кількість навчальних годин [Hours], наявність предметної програми [Sub_program], унікальний ідентифікатор викладача з поточного предмета [Teacher], кількість книжок у бібліотеці [Books], кількість книжок для лекцій [lecture_books], кількість книжок для практичних занять [practice_books] і кількість книжок для лабораторних занять [lab_books].

Кожному навчальному предмету присвоюється ідентифікаційний номер [Sub_ID], який теж є атрибутом таблиці [Subjects]. Оскільки предмет можуть вести кілька викладачів із різною кількістю аудиторних годин, типом контролю і кількістю книг у бібліотеці, то для кожного викладача з одного й того самого предмета потрібно вводити ці дані. Тому первинним ключем, що визначає всі інші атрибути в рядку таблиці [Subjects], обрано ідентифікатор рядка [Row_ID]. Одному рядку таблиці [Lecturers] відповідає від одного до кількох рядків у таблиці [Subjects], оскільки один викладач може вести кілька предметів.

Таким чином, для опису відношення між таблицею [Lecturers] і таблицею [Subjects] використовується тип зв'язку один-ко-многим (1:∞). Зовнішнім ключем для цього зв'язку є атрибут of identifier of the teacher [Teacher] у таблиці [Subjects]. До таблиці [Subjects] можна додати також поля для вказівки наявності електронних версій методичних вказівок до практичних занять, лабораторних робіт і конспектів лекцій, разом з електронними адресами цих ресурсів.

Результати вивчення студентами навчальних дисциплін подано в таблиці [Progress]. У цій таблиці є такі поля: номер залікової книжки студента [Student_ID], академічний рік [Year], дата контролю [Date], ідентифікатор предмета [Sub_ID], результат контролю - оцінка або залік

[Mark], [Test]. Знання з кожної дисципліни перевіряються в усіх її слухачів, тому одному рядку таблиці [Subjects] відповідатиме кілька рядків у таблиці [Progress]. Таким чином, для опису відношення між таблицею [Subjects] і таблицею [Progress] використовується тип зв'язку один-ко-множиною. Зовнішнім ключем цього зв'язку є атрибут of the identifier of the subject [Sub_ID] у таблиці [Progress]. Таблиця [Progress] пов'язана також із таблицею [Students]. Для кожного студента в таблиці [Progress] зберігаються результати екзаменаційних сесій. Для опису відношення між таблицею [Students] і таблицею [Progress] використовується тип зв'язку один-ко-многим. Зовнішнім ключем цього зв'язку є атрибут of number of student's record book [Student_ID] у таблиці [Progress].

## 4.3 Схема даних

На наступному етапі проектування бази даних з ER-діаграмми отримується реляційна схема. Сутності представлені таблицями (що вже було вказано в попередньому підрозділі). Атрибути сутності стоять стовпцями в таблиці. Унікальні ідентифікатори для кожної таблиці вказуються як первинні ключі. Атрибути, що беруть участь у зв'язках «один-до-багатьох» і «один-к-одному» становляться зовнішніми ключами ми. Створюються індекси для полів, за якими можливо відбуватиметься пошук або сортування.

Схема даних (або фізична модель бази даних) являє собою групу пов'язаних таблиць із зазначенням їхніх полів. Для схеми даних, наведеної на рисунку 4.4, дані в таблицях задовольняють вимогам реляційної моделі. Кожне значення, що міститься на перетині рядка і колонки, є атомарним (тобто таким, що не розділяється на кілька значень). Значення даних в одній і тій самій колонці належать до одного й того самого типу (домену), доступного для використання в цій СУБД. Кожен запис у таблиці унікальний, тобто в таблиці не існує двох записів із набором значень її полів, що

повністю збігаються. Кожне поле всередині таблиці має унікальне ім'я. Послідовність полів у таблиці несуттєва. Послідовність записів також несуттєва.

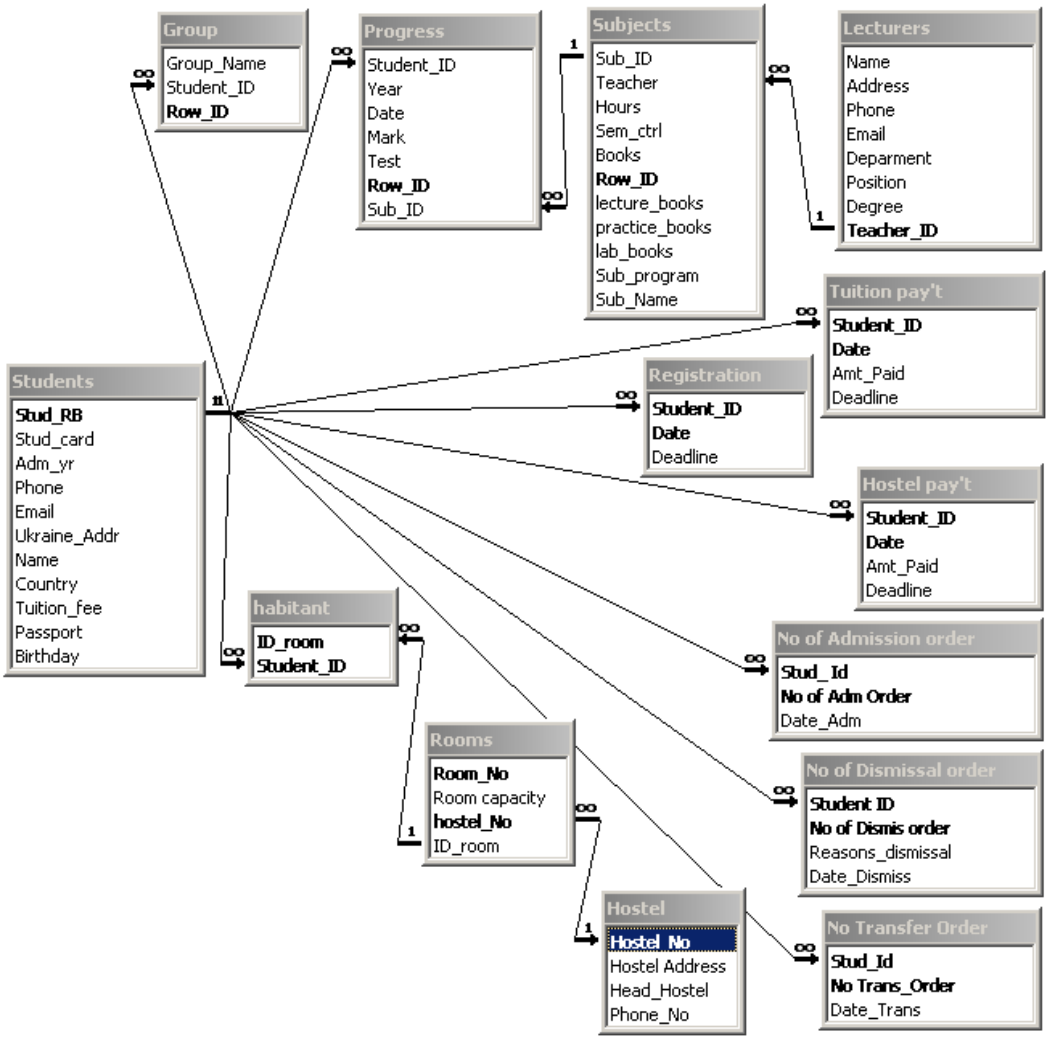


Рисунок 4.4 - Схема даних інформаційної системи деканат

Оскільки послідовність колонок у таблицях не існує, звернення до них здійснюється за іменами, і ці імена кожної внутрішньої таблиці є унікальними (но не зобов’язані бути унікальними для всієї бази даних). Перший ключ будь-якої таблиці повинен містити унікальні непусті значення для цієї таблиці. Це одно із правил ссылочной целостности. Соблюдение правил ссылочной целостности автоматично контролюється СУБД.

## 4.4 Нормалізація і схема функціональних зв'язків

В основі процесу проектування бази даних лежить метод нормалізації. Теорія нормалізації заснована в своїй черзі на концепції нормальних форм. Таблиця знаходиться в даній нормальній формі, якщо вона відповідає певним набору вимог. Існує п'ять нормальних форм, однак реально використовуються тільки перші три. Також перші дві нормальні форми є проміжними кроками для приведення бази даних до третьої нормальної форми.

Процес нормалізації складається в декомпозиції відносин, що знаходяться в попередній нормальній формі, у двох або більше відносинах, що відповідають вимогам наступної нормальної форми. Так досягається покращення характеристик бази даних при включенні, зміні та видаленні даних. Окончательная ціль нормалізації – це отримання такого проекту бази даних, в якому кожен факт з’являється тільки в одному місці, т.е. виключена ізбутовість інформації. Таким чином, виключається можлива суперечність хранимих даних.

Реляційна таблиця за визначенням знаходиться вже в першій нормальній формі. У такій таблиці в позиції на пересеченні кожної строчки і таблиці таблиці завжди знаходиться єдине атомарне значення. Одна, яка при цій такій таблиці може містити всі вихідні дані. Наприклад, приказ про зарахування студентів видається для кожної групи. Тому один і той же номер показу по зачисленню студента буде повторюватися кілька разів у таблиці [No of Admission order]. Результатом ізбиточності даних є аномалії модифікації даних.

Реляційна таблиця перебуває в другій нормальній формі, якщо вона перебуває в першій нормальній формі і її неключові поля повністю залежать від усього первинного ключа, а не від його частини. Приклад залежності від частини складеного первинного ключа - таблиця [No of Dismissal order]. Первинний ключ у цій таблиці складається з атрибутів [Student ID] і [No of

Dismis order]. Причому атрибут [Date_Dismiss] залежить від одного складника первинного ключа - поля [No of Dismissal order], а атрибут [Reasons_dismissal] залежить від другого - поля [Student ID].

Для переходу до другої нормальної форми необхідно визначити, на які частини можна розбити первинний ключ, так щоб деякі з неключових полів зависелі тільки від однієї з цих частин. Створюється нова таблиця для такої частини ключа і групи, що зависять від кожного поля, які переміщуються в цю таблицю. Частина первинного ключа стане при цьому первинному ключі нової таблиці. Потім із вихідної таблиці видаляються поля, переміщені в інші таблиці, крім тих, які є зовнішніми ключами.

У свою чергу, таблиця знаходиться в третій нормальній формі, якщо вона знаходиться у другій нормальній формі, і всі її неключові поля залежать тільки від первинного ключа (тобто немає транзитивних залежить від мостів від первинного ключа) . Зависимість від неключового поля називається транзитивною, якщо один атрибут залежить від іншого неключового атрибута, який у свою чергу залежить від первинного ключа.

Для виявлення всіх функціональних залежностей відображаються діаграми. На рисунку 4.5 наведено функціональні залежності в таблиці [Lecturers]. Показаны зависимости неключевых атрибутов от основного ключа. Однак атрибут [Position] також залежить від атрибута [Degree], так як кожна посада відповідає певним вимогам при прийнятті на роботу, в тому числі, і наукову ступінь.

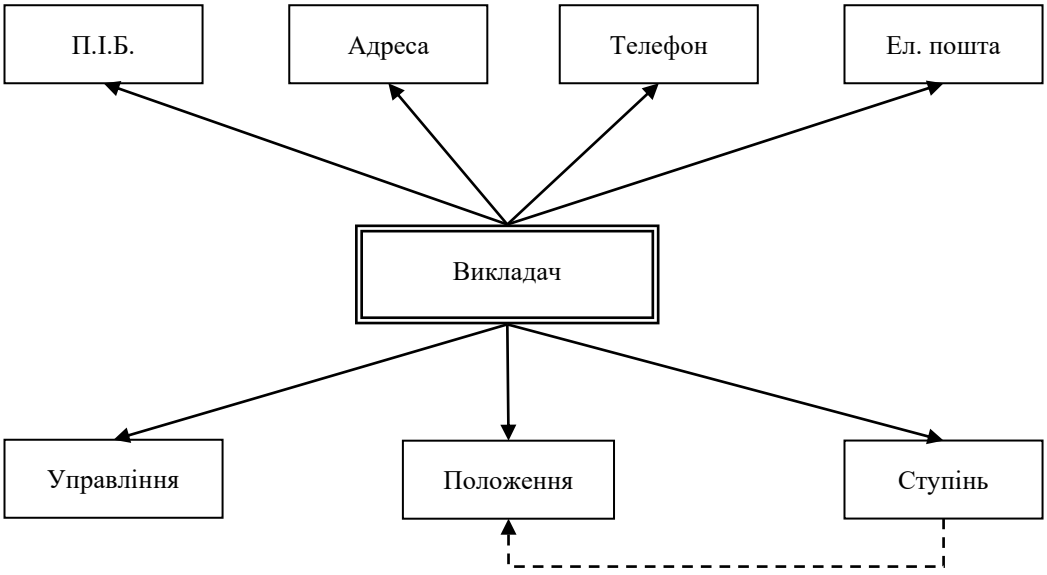


Рисунок 4.5 - Функціональні залежності в таблиці [ Викладачі ]

На рисунку 4.6 також показані функціональні залежності між атрибутами в таблиці [Subjects]. В цьому випадку первинний ключ визначає розподіл дисциплін по викладачам. Кожен преподавець для своєї дисципліни має набір книг, методичних вказівок, робочу програму та кількість аудиторних годин. Однак у таблиці [Subjects  назва атрибута дисципліни [Sub _ name] залежить від атрибута ідентифікатора дисципліни [Sub _ ID], а через нього транзитивно залежить від первинного ключа. Для нормалізації ці два атрибути повинні бути перенесені в нову вкладку з перечнем найменувань дисципліни та їх ідентифікаторами. Однак засобами СУДБ реалізована автопідстановка ідентифікатора дисципліни по її назві та вибору зі списку назв.

Також у таблиці [Subjects] є функціональна залежність між ідентифікатором дисципліни та ідентифікатором викладача. Кожен викладач може вести лише певні дисципліни. Але на занесення даних у таблицю така функціональна залежність не має значного впливу, тому що від первинного ключа поле [Teacher] залежить усе ж таки більшою мірою.

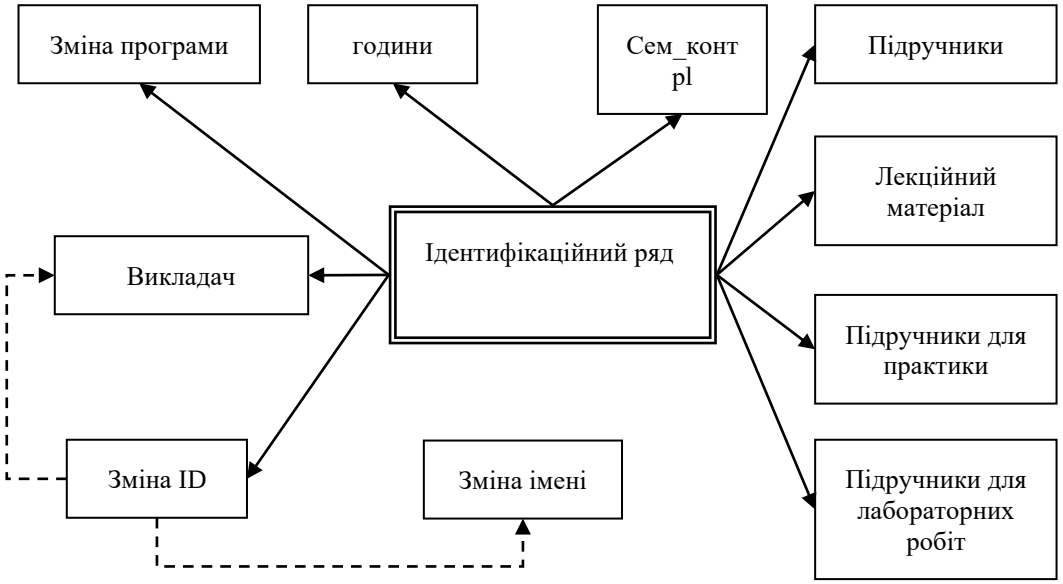


Рисунок 4.6- Функціональні залежності в таблиці [ Предмети ]

Для переходу до третьої нормальної форми визначаються всі поля, від яких транзитивно залежать інші неключові поля. Для кожного такого поля і безлічі зависаючих від нього полів створюється окрема таблиця, в якій первинним ключем стає поле, від якого зависять інші атрибути . З вихідної таблиці ці атрибути видаляються, крім тих, які є зовнішніми ключами для зв’язку з новою таблицею.

Зазвичай проектування бази даних завершується на наведених відносинах до третьої нормальної форми. В окремих випадках потрібно про - продовження нормалізації далі. Наступний рівень нормалізації – це нормальна форма Бойса-Кодда (BCNF), що виражає вимогу щодо відсутності ви залежних атрибутів первинного ключа від неключових атрибутів. Це обозначає, що первинний ключ повинен бути єдиним в рамках однієї таблиці. Існують ще четверта і п'ята нормальна форми, які дуже рідко використовуються на практиці.

## 4.5 Реалізація бази даних

База даних інформаційної системи деканат реалізована і

використовується в університеті із застосуванням СУБД Microsoft Access, в відповідному з пред'явленими в даний час з її вимогами щодо доступу до даних. При проектуванні використовувалися візуальні засоби СУБД для створення таблиць, їх атрибутів, зв'язків між таблицями. Для подальшого впровадження електронної системи документообороту пропонується база даних реалізована з використанням СУБД MySQL . У проект бази даних включені різні типи об'єктів, такі як таблиці для зберігання даних, індекси для сортування даних і підтримки ключів, обмеження і правила для підтримки ссылочной повноти та обмеження кількості даних, перегляди для представлення даних, тригери для обробки подій, запити для отримання вибору даних, форми для введення та відображення даних, звіти для роздруківки даних за запитами. Реалізовані обмеження включають як установку допустимих значень для будь-якого поля таблиці, так і referential constraints. Referential constraints вимагають, щоб значення поля зв’язку в одній таблиці відповідали одному з уже наявних значень в іншій таблиці. Для генерації ідентифікаторів використовуються лічильники з метою уникнення випадкового дублювання цілей при введенні даних користувачем.

```
SELECT DISTINCT [Group].[Group_Name], [Students].[Name]
FROM ([Students] LEFT JOIN [Group] ON
[Students].[Stud_RB]=[Group].[Student_ID]) LEFT JOIN [Progress] ON
[Students].[Stud_RB]=[Progress].[Student_ID]
WHERE ((([Progress].[Mark])=5));
```

Рисунок 4.7 – Вибір даних за умовою

Запити реалізовані із застосуванням мови SQL (Structured Query Language). Ця мова, що є стандартом, дозволяє створювати, редагувати та видаляти таблиці, за умовами вибирати та оновлювати дані, що зберігаються в таблицях. Приклад запиту на мові SQL наведено на рисунку 4.7.

Приведений вище SQL -запит дозволяє здійснити вибір імен студентів, які мають відмінну оцінку хоча б по одному з предметів.

## 4.6 Висновки

В цьому розділі наведено рішення задачі проектування однієї з найбільш значущих частин електронної системи документообороту. Перед тим, як запустити будь-який документ на обробку, необхідно вилучити окремі збережені дані та вставити їх у документ. Задача ефективного зберігання і доступу до даних залишається актуальною. Описане в цьому розділі проектування реляційної бази даних для деканату представ дає можливість ефективно збирати дані та досліджувати функціонування університету. У запропонованій моделі узяті до уваги основні вимоги, що пред'являються до розробки бази даних:

1 користувач бачить не фізичне представлення даних, а логічні, що робить модель простої для розуміння та впровадження;
2 представлені таблиці забезпечують повну незалежність даних, а також структурну незалежність, оскільки є виключно логічною структурою;
3 між таблицями забезпечена зв'язок на рівні посилань;
4 вирішене питання з контрольованою відсутністю даних.

# 5 ВИБІР КРИТЕРІЄВ І ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

## 5.1 Планування експерименту

Кожне наукове дослідження, проведене з використанням будь-якої моделі, передбачає планування і проведення експерименту для збирання даних про функціонування моделі. Імітаційні моделі відображають процеси, що відбуваються в реальних системах. У нашому випадку, коли в моделі враховуються випадкові фактори, необхідно провести багато експериментів із різними значеннями випадкових величин. Експеримент проводиться для отримання статистичних даних, достатніх для прийняття необхідного рішення.

Визначимо цілі моделювання системи університету. Основною метою моделювання є структурна оптимізація, тобто пошук оптимальної структури модельованої системи. Стратегією досягнення мети експерименту обрано порівняння параметрів реально існуючої системи з моделлю, в якій передбачено використання інформаційних технологій. Для досягнення поставленої мети необхідно провести аналіз ефективності роботи з документами в університеті та прогнозування ефективності електронної системи документообігу в цьому випадку.

Створювана система, її об'єкти і процеси характеризуються як складні. Під час їх вивчення потрібен системний підхід, що передбачає багатокритеріальність, багатофакторність, відповідний метод опису та ефективність проведення досліджень. Саме властивості досліджуваної системи визначають вибір способу моделювання. Ці властивості враховуються під час розроблення алгоритмічного та програмного забезпечення. У більшості випадків моделювання складних систем проводиться з використанням експериментально-статистичних методів. Один із таких методів буде використано для моделювання системи університету.

Планування експерименту – це розробка такого плану експерименту,

який дає можливість з мінімальними витратами і за мінімальну кількість запусків моделей отримати значні результати або знайти оптимальне рішення по функціонуванню системи [9]. При цьому необхідно визначити умови проведення експерименту, вхідні дані для кожного експерименту, кількість запусків моделей і тривалість запуску моделей. Також визначається, які статистичні дані будуть зібрані в результаті проведення експерименту. Якщо використовуються випадкові числа, то визначаються умови їх генерації моделюючі системи. Після проведення експерименту визначається чутливість моделі до вхідних даних, оцінюється точність вихідних даних.

Експеримент проводиться по розробленій в попередніх главах моделі системи документообороту. Виду обмежень щодо кількості операційних блоків у студентській версії імітаційної програми GPSS World , була промодельована робота одного деканату та відповідно до його кафедри. Так як для кожного декана є один і той же набір документів, то для моделювання загально університетської системи документи, що надходять від одного декана, були розмножені за кількістю деканатів в університеті.

Задача імітаційного моделювання документообороту університету є досить унікальною. Крім того, моделювання спрямовано на аналіз системи конкретного університету. Тому типові вхідні дані для таких завдань відсутні. Для вирішення цієї проблеми пропонується використовувати вхідні дані для експерименту, отримані за допомогою вимірювання часу обробки кожного з типів документів для описуваного університету. Вимірювальний експеримент проводився в вимірюванні тимчасових і кількісних характеристик для кожного з маршрутів оброблених документів для деканата навчання студентів на іноземних мовах.

Мінімальна кількість запусків моделей визначає швидкість переходу системи в стаціонарний режим. Стаціонарний режим настає в тому випадку, коли ймовірні характеристики системи вже не залежать від течії модельного часу. Момент досягнення стаціонарного режиму визначається за проміжними результатами запусків моделі. Період документообороту в університеті рівнь

одному семестру. За одиницю вимірювання модельного часу взята одна хвилина. Тривалість запуску моделі визначається з розрахунку тривалості семестру в модельних хвилинах і залежить від швидкості процесора комп'ютера, на якому здійснюється моделювання.

Для моделювання випадкових величин в моделі використовуються засоби GPSS. У GPSS є безліч різних генераторів рівномірно розподілених випадкових чисел в інтервалі [0;1] з іменами $RN_j$, де $j = \overline{1,8}$. За замовчуванням всі вісім генераторів при зверненні до нього видають один і тут же послідовність випадкових чисел.

У результаті проведення експерименту визначаються такі дані, як відносно пропускна здатність ( $P_{serv}$або ймовірність того, що заявка, довільно вибрана з вхідного потоку, виявиться в потоці обслужених заявок) і абсолютна пропускна здатність (або інтенсивність обслуговування). Остання характеризується середнім числом заявок, які можуть бути обслужені системою за одиницю часу

$$\lambda_{serv} = P_{serv} \cdot \lambda \,. \tag{5.1}$$

В свою чергу ймовірність відмови $P_{ref}$ - це ймовірність того, що заявка, довільно вибрана з вхідного потоку з інтенсивністю $\lambda$, виявиться в потоці відмовлених заявок з інтенсивністю $\lambda_{ref}$

$$P_{ref} = \frac{\lambda_{ref}}{\lambda} = \frac{\lambda - \lambda_{serv}}{\lambda} = 1 - P_{serv}. \tag{5.2}$$

Наступним параметром є середній час очікування $\bar{t}_{wait}$. Ця величина рівна інтервалу часу, протягом якого заявка знаходиться в черговій частині. Середній час розміщення заявки в системі $\bar{t}_{syst}$ рівночасно з часом від моменту надходження заявки в модель до моменту появи її у вихідному потоці

$$\bar{t}_{syst} = \bar{t}_{wait} + \bar{t}_{serv}. \quad (5.3)$$

Також будуть оцінені такі параметри, як середня довжина черговості $\bar{r}$ (математичне очікування числа заявок, що знаходяться в черговості) і середнє число заявок у системі $\bar{z}$ (математичне очікування числа заявок, що знаходяться в черговості та в каналах обслуговування). Середнє число заявок в системі рівне сумі довжини чергових і середнього числа зайнятих каналів (з кожним каналом в проміжок часу може бути подана тільки одна заявка)

$$\bar{z} = \bar{r} + \bar{k}. \quad (5.4)$$

Своєю чергою, середнє число зайнятих каналів дорівнює оцінці математичного очікування числа зайнятих обслуговуванням каналів, що є випадковою величиною. Цей показник характеризує ступінь завантаження системи [10].

## 5.2 Застосування імітаційної моделі системи

### 5.2.1 Застосування теорії масового обслуговування для моделювання документообігу в університеті

Функціонування системи описано в третьому розділі. Попереднє моделювання та якісна оцінка виконані методом системного аналізу та проектування. У п'ятому розділі вирішена задача оцінки параметрів досліджуваної системи. Для моделювання такої системи, яка залежить від переходу із стану в стан, зручно використовувати методи теорії масового обслуговування. При цьому необхідно спростити модель, так як для вирішення поставленої задачі не потрібна дуже висока точність моделювання.

Система університету представлена як система масового обслуговування, за якою проходять документи за встановленими маршрутами. Система призначена для обслуговування цих заявок. З основних структурних одиниць виділено те, що повинні бути відображені в імітаційній моделі, а саме ректорат, деканати та кафедри. На рисунку 5.1 система документообороту представлена у вигляді системи масового обслуговування. На малюнку 5.1 система документообігу представлена у вигляді системи масового обслуговування.

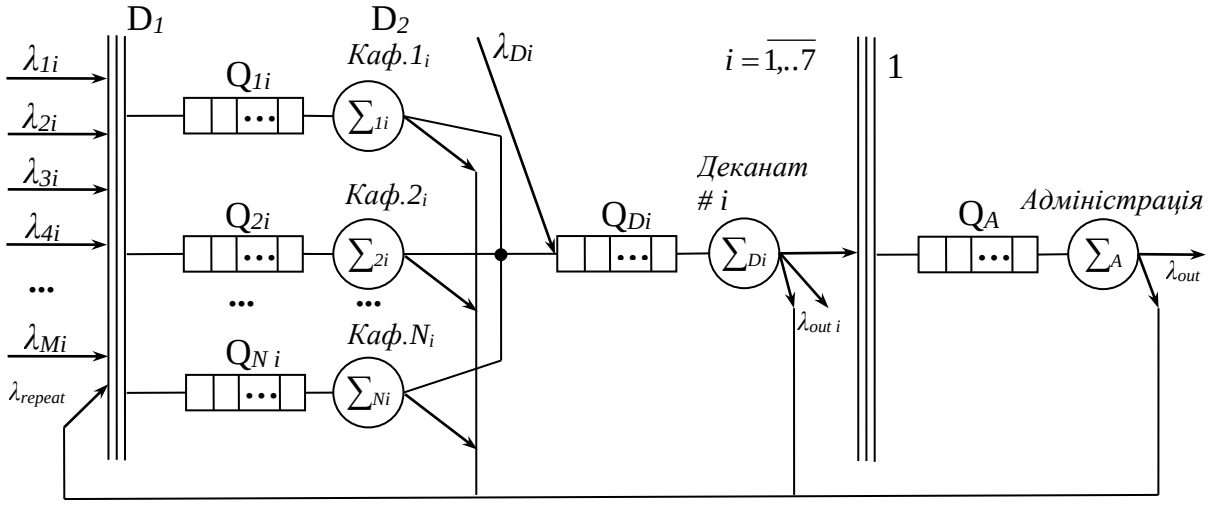


Рисунок 5.1 – Система документообороту університету як система масового обслуговування

На малюнках наведені основні параметри системи масового обслуговування. Конфігурація системи масового обслуговування складається з обслуговуваних приладів різного рівня та чергових заявок перед ними. Модель на рисунку 5.1 описує типовий деканат і відноситься до його кафедри. На кафедри і в деканат ведеться потік документів, які необхідно розглянути, заповнити і затвердити. В залежності від типу документа йому присвоюється визначений маршрут. Для більшості документів може знадобитися підпис ректора. Так як потоки документів ідентичні для всіх

деканатів університету, то моделюється один деканат, а обслужений в ньому потік заявок копіюється за кількістю деканатів і направляється на підпис ректора.

Задані інтенсивності вхідного потоку заявок $\lambda_{1i}$ - $\lambda_{Mi}$ , які моделюють різні типи документів, що надходять на обробку. Кожна інтенсивність відповідає одному з типів документів. В якості обслуговуючих приладів беруться кафедри, деканат і адміністрація, приймаючі документи на розгляд. Для приборів обрана безпріоритетна дисципліна обслуговування. Час обслуговування задано рівномірним законом розподілу випадкових величин, як найбільш прийнятний для опису процесу обслуговування неоднорідного потоку заявок. Перед приладами поставлені реєстратори чергових $Q_{1i} - Q_{Ni}$, $Q_{Di}$, $Q_A$ для урахування часу очікування обслуговування. Для спрощення всі чергові прийняті безпріоритетними необмеженими за кількістю місць і за часом очікування.

Використання теорії масового обслуговування, теорії ймовірностей, теорії марковських процесів, а також диференціальних і алгебраїчних рівнянь дозволяє швидко отримати математичну модель для вирішення достатньо широкого кола задач дослідження комп'ютерних систем. Однак при подібному вирішенні багатьох завдань необхідно здійснити значне усунення і погіршення моделі раді можливості отримати хоча б наближене рішення задачі. Крім того, математичні моделі мають ряд істотних недоліків. До нього відносяться складність аналітичного опису числових процесів і, як слідство - значні спрощення. Потоки заявок представлені як простіші, закон розподілу тривалості обслуговування заявок передбачається експоненціальним.

З допомогою математичних моделей неможливо представити обслуговування заявок одночасно кількома приладами. Штучні утворення аналітичних моделей з метою використання такого математичного апарату для реальних інформаційних комп'ютерних систем дослідження іноді подається під сумнів результатів аналітичного моделювання. Тому аналітичний метод моделювання був доповнений імітаційним методом, що

має більш широку сферу застосування для моделювання функціонування інформаційних систем. Представлення моделі у вигляді схем системи масового обслуговування необхідно для правильного представлення імітаційної моделі.

### 5.2.2 Імітаційне моделювання проектованої системи

При вирішенні задач, пов’язаних з оцінкою різних варіантів організації процесів у системах, з пошуком оптимальної структури, складу та конфігурації ресурсів обладнання при змінному потоку завдань, найбільш ефективними надаються програмні імітаційні моделі. Зручними для цих цілей є спеціальні мови імітаційного моделювання, що представляють широкий набір засобів опису об'єкта моделювання та засобів проведення експериментів. Для розробки моделей використовується засіб імітаційного моделювання GPSS World (General Purpose Simulating System). Ця система застосовна для побудови дискретних імітаційних моделей і проведення експериментів з ними. Для вирішення цієї задачі система має спеціальні засоби опису поведень ймовірних систем.

Імітаційна система GPSS World призначена для швидкого одержання результатів експериментів з найбільшою надійністю та можливості керування процесом моделювання. Вбудовані засоби аналізу даних можуть обчислювати довірчі інтервали та проводити дисперсійний аналіз. GPSS-модель складається з блоків і транзактів. Послідовність блоків визначає напрямки руху транзактів. Кожен блок має свою підпрограму, яка виконується в той момент, коли блок стає активним. Для відображення динамічних процесів у реальних системах, GPSS має внутрішній механізм передавання керування. Керування від блоку до блоку передається через транзакт, що переміщається по них.

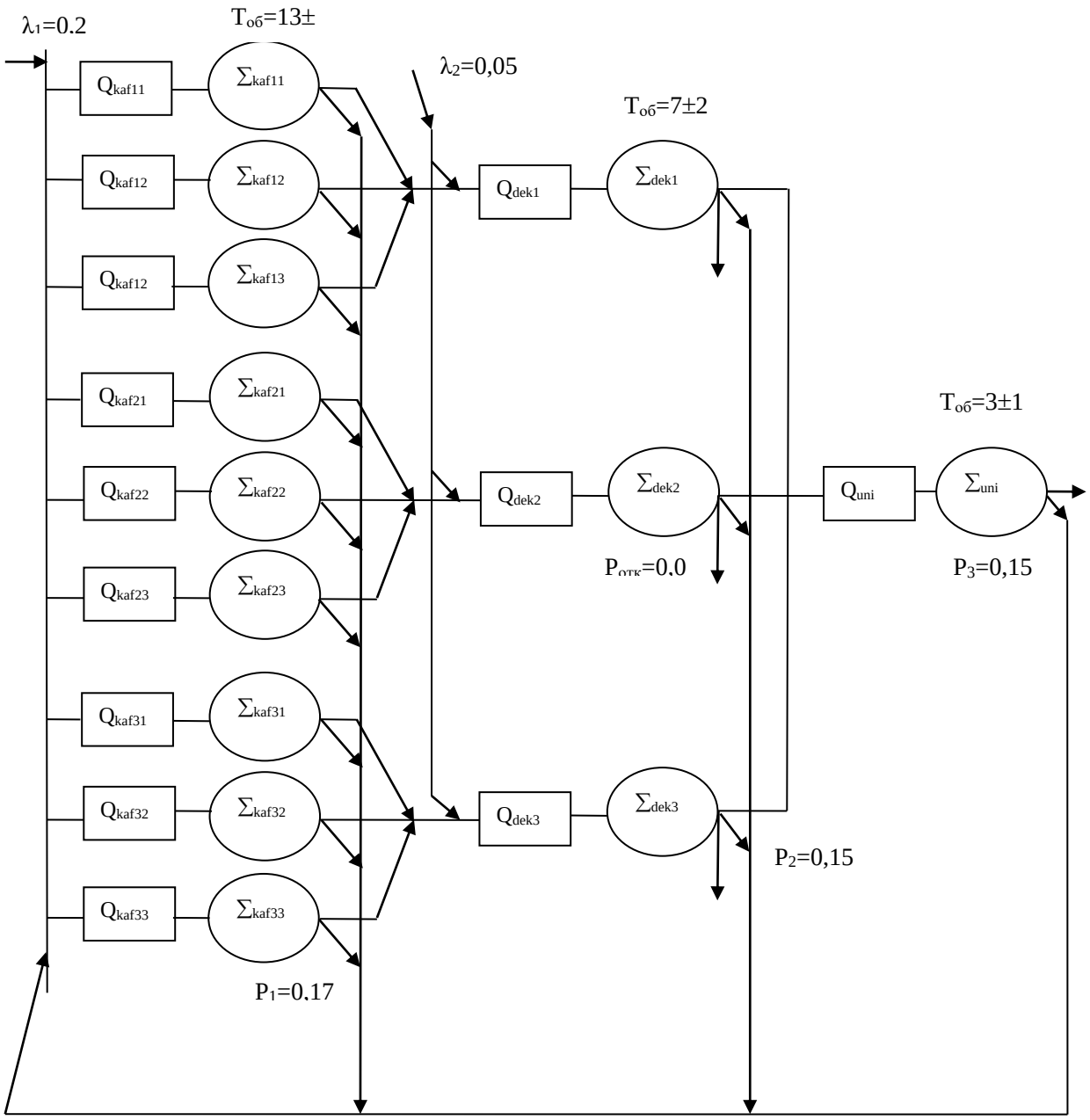


Рисунок 5.2 – Структура GPSS моделі

Реалізацію моделі мовою GPSS наведено в додатку Б.

Опис роботи програми: Функції kaff та dekf є дискретними і можуть приймати одне з трьох значень, які відповідають різниці між мітками відповідних блоків GPSS, призначених для моделювання роботи кафедр та деканатів. За допомогою цих функцій здійснюється направлення документів на одну з кафедр та деканатів з рівною ймовірністю. Вхідні потоки документів розподілені за експонентним законом, для цього була задана

функція expon.

Змінні fork, ford1, ford2 потрібні для обчислення різниці між мітками mark-time, які були використані у програмі для розрахунку часу перебування у деканатах та на кафедрах. Змінна ford1 обчислює час першого потоку заявок, а ford2 для другого.

Таблиці kaf, dec1, dec2, univ, fquni призначені для побудови відповідних гістограм.

Документи з першого потоку позначаються одиницею, це потрібно для того, щоб розрізняти документи з першого та другого потоку, потім вони потрапляють у блок програми, позначений LUNI1.

Лістинг 5.1 - Блок програми

```
LUNI1 assign 1,LDEK11
assign 1+,fn$dekf
transfer ,P1
```

Тут за допомогою 1-го атрибуту транзакту здійснюється направлення документа до одного з деканатів. Далі знаходиться блок, який моделює надходження другого потоку документів, документи позначаються двійкою в 10м атрибуті транзакту і також вирушають до одного з деканатів, але обминаючи обробку на кафедрах.

Документи, з першого потоку, потрапивши на вхід одного з деканатів позначених мітками LDEK11, LDEK21, LDEK31 відповідно, відправляються на обробку однієї з кафедр цього деканату, це моделюється наступними блоками:

Лістинг 5.2 – Моделювання обробки запитів

```
LDEK11 mark 4
assign 2,LKAF11
assign 2+,fn$kaff
transfer ,P2
```

Перед кожним деканатом та кафедрою за допомогою блоків queue та

depart організовано збір статистики про черги. Обробка на кафедрі закінчується тим, що 17% документів надсилаються на перегляд за міткою LRE. Інші вирушають на обробку в деканаті, яка закінчується тим, що 5% документів отримують відмову, 15% вирушають на доопрацювання, а інші пропускаються до адміністрації університету. Це все описується блоками:

Лістинг 5.3 – Доопрацювання запитів

```
LDEK12 queue qdek1
seize dek1
depart qdek1
advance 7,2
release dek1
mark 3
mark 5
transfer .05, LOTK
transfer .15,LUNI2,LRE
```

Обробка в адміністрації університету моделюється такими блоками:

Лістинг 5.4 – Обробка запитів

```
LUNI2 queue quni
seize uni
depart quni
advance 3,1
release uni
tabulate fquni
transfer .15,LEND,LRE
```

Тут відбувається обробка документів та табулювання поточної довжини черги, потім 15% документів відправляються на доопрацювання, а інші відправляються на мітку LEND, де виконується збір відомостей про час перебування на кафедрах, деканатах та університеті, після чого документи залишають модель. Результати моделювання:

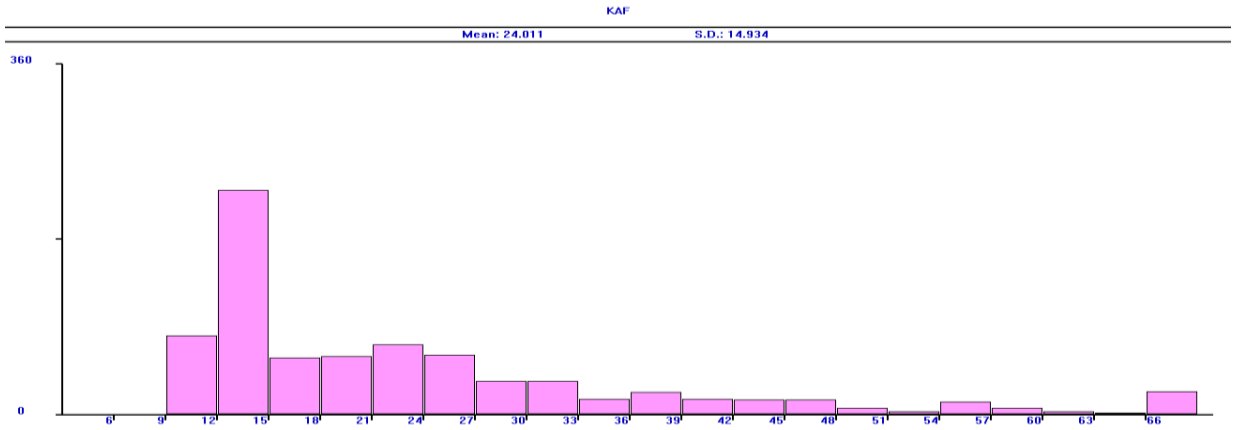


Рисунок 5.3 – Гістограма часу перебування документів на кафедрах

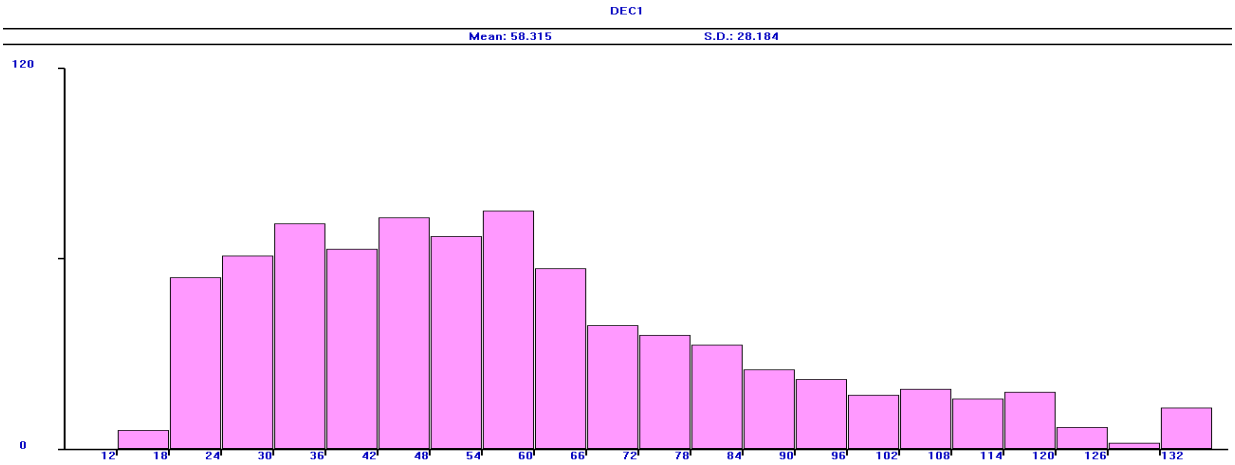


Рисунок 5.4 – Гістограма часу перебування документів першого потоку у деканатах

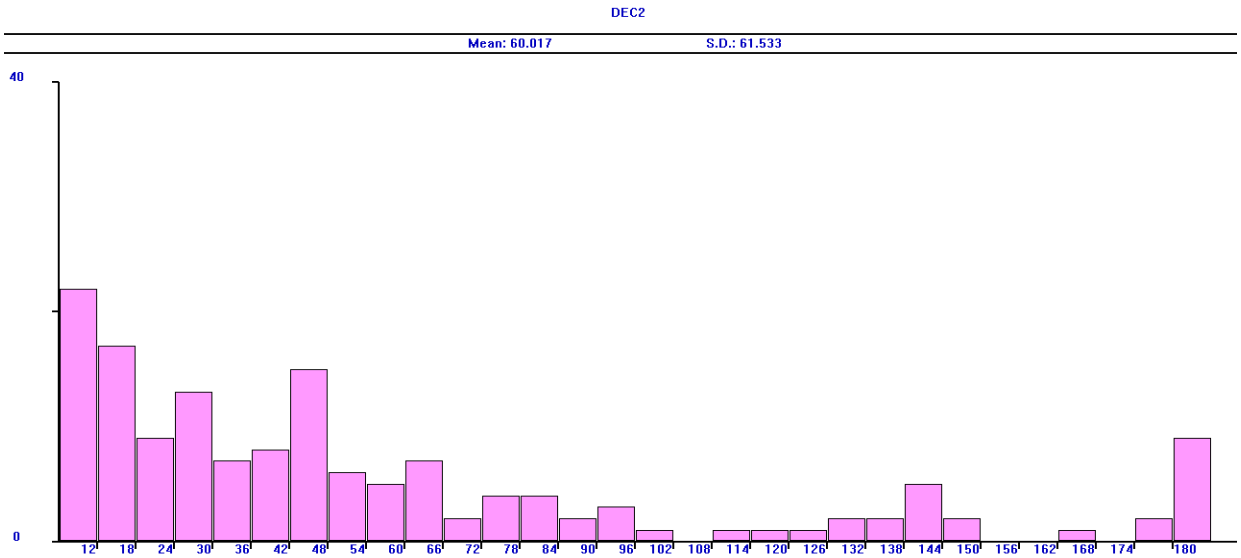

Рисунок 5.5 – Гістограма часу перебування документів другого потоку у деканатах

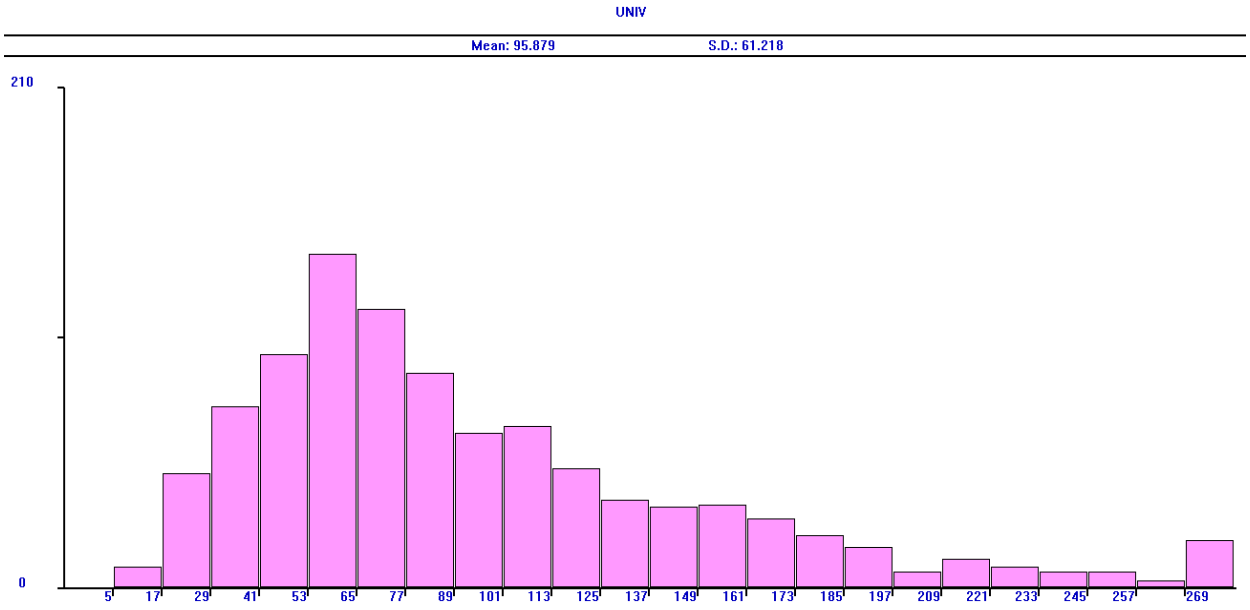


Рисунок 5.6 – Гістограма часу перебування документів в університеті

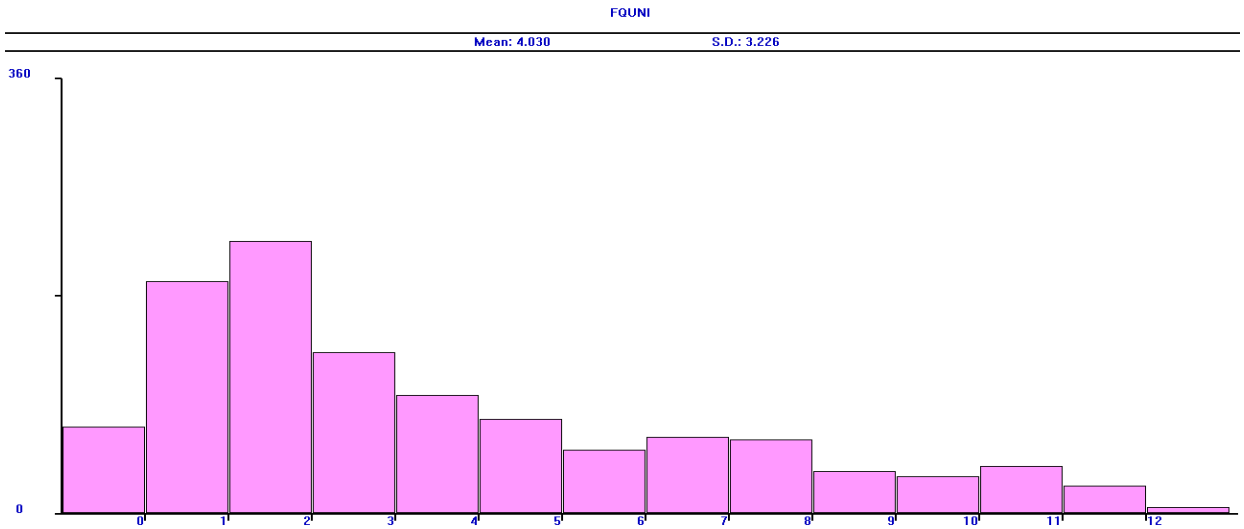


Рисунок 5.7 – Гістограма стану черги документів до адміністрації університету

Аналіз результатів моделювання.

Моделювання було проведено для 1000 документів, тобто коли тисячний документ залишив модель, процес моделювання було завершено. При інтенсивності першого вхідного потоку $\lambda_1$=0,25 блоком generate було

сформовано 854 транзакти, а другим при інтенсивності $\lambda_2$=0,05 сформовано 166. Оскільки система з множинними зворотними зв'язками, то на вхід університету (моделі) було подано 1500 документів. Усього на доопрацювання було відправлено 646 документів. До першого деканату було відправлено 496 документів, до другого 500, а до третього 504. При цьому коефіцієнти використання деканатів становили 0,93, 0,9 та 0,945 відповідно. Максимальна довжина черги документів до першого деканату склала 12 документів, до другого 9 та до третього 16. Коефіцієнти використання кафедр першого деканату – 0,62, 0,59 та 0,63, другого деканату – 0,6, 0,52 та 0, 73, третього – 0,62, 0,64 та 0,64 відповідно. У середньому максимальна довжина черги на кожну кафедру 5 документів. Коефіцієнт використання адміністрації 0,95, при цьому максимальна довжина черги 13 документів, а середня довжина 4 документи, що легко помітити з гістограми малюнку 6.

З рисунку 5.3 можна побачити, що основна маса документів затримується на кафедрі на 12-15 одиниць модельного часу, інші документи затримуються довше оскільки їм доводиться стояти у черзі і деякі з них вирушають на доопрацювання.

Гістограма, зображена рисунку 5.4, показує, що у середньому документи з першого потоку, включаючи обробку на кафедрах, затримуються в деканатах на 58 одиниць модельного часу. Можна помітити невелику частину документів, які провели у деканатах лише 12-18 одиниць модельного часу – це ті документи, які не стояли у чергах та не вирушали на перегляд.

Наступна гістограма рисунку 5.5 наочно відображає час перебування документів другого потоку в деканатах. Більшість були одразу оброблені та відправлені в адміністрацію, решта ще тривалий час оберталася в університеті, простоюючи в чергах і повертаючись на доопрацювання.

На рисунку 5.6 подано гістограму часу перебування документів в університеті. У середньому документ проводить в університеті 96 одиниць модельного часу. На гістограмі є дещиця документів, які провели в університеті 5-17 одиниць модельного часу – це документи з другого потоку,

які отримали відмову.

На рисунку 5.7 представлено динаміку стану черги до адміністрації університету. Не важко помітити, що середня довжина черги 4 документи.

З гістограм, звіту та аналізу можна зробити висновок, що модель працює адекватно, отримані хороші коефіцієнти використання.

# ВИСНОВКИ

У роботі проводився аналіз інформаційної системи університету, під час якого було вирішено такі завдання. Для вибору методу, яким виконано моделювання предметної області, здійснено аналіз провідних методологій і методів системного аналізу та проектування. Зокрема, було розглянуто Каскадну модель життєвого циклу, Спіральну модель, Об'єктно-орієнтовані методи, Unified Software Development Process і його модифікацію Rational Unified Approach.

Запропоновано реалізувати електронну систему документообігу як розподільну систему. Як зазначено в пункті 2.10, такий тип системи є надійним і легко масштабованим. Крім того, розподільні системи володіють і іншими корисними властивостями, такими як відкритість, прозорість і паралельність. На жаль, при використанні таких систем виникають проблеми з управлінням і підтримкою безпеки при можливості доступу з великого числа комп'ютерів. Крім того, розподільні системи складно проектувати. Тому для системного аналізу і проектування системи документообігу використаний метод COMET, спрямований на проектування подібних систем.

В ході виконання роботи отримані наступні результати . Застосування методу COMET дозволило створити моделі різних рівнів системного аналізу, такі як використання моделі, концептуальну статичну модель із власними класами, діаграми контексту класів і стереотипів класів додатків, діаграми наявних і діаграми кооперації. У системі виявлені підсистеми та існуючі між ними зв’язки, виділені клієнтські та серверні частини. Також побудовані імітаційні моделі для моделювання процесів обробки документів у системі. У четвертій голові наведені етапи розробки бази даних для інформаційної системи деканат іноземних студентів.

Практична вартість роботи полягає в отриманні ряду моделей адміністративної системи для подальшого проектування та застосування в

еквівалентних системах. Розроблена база даних може бути використана в будь-якому з деканатів як частина інформаційної системи університету. Наукова новизна роботи міститься в застосуванні методу COMET для опису адміністративної системи та застосуванні методів імітаційного моделювання в заданій предметній області для отримання показників ефективності системи документообороту в університеті.

Результати імітаційного моделювання показали, що введення в університет електронної системи документообігу приведе до зменшення часу обробки документів і підвищення ефективності роботи підрозділів. З переваг ЕСД також відзначені такі функції, як керування маршрутом документа, обмеження правого доступу до різних документів, різні типи звітів і представлення документів, пошук документів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cockburn, A., Surviving Object-Ori¬ en¬ ted Projects, Addison-Wesley, 1998, Cockburn, A., Crystal/Clear: A Human-Powe¬ red Methodology for Small Teams, Addison-Wesley, 2000

2. Beck, K., Extreme Programming Explained: Embrace Change, Addison-Wesley, 1999

3. Humphreys, W. Introduction to the Personal Software Process, Addison-Wesley, 1997

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

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

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

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

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

9. Томашевский В.М. Моделирование систем. Киев, Издательская группа BHV, 2005. 353 с.

10. Gorbachov VA: "Techniques of System Modeling", SMIT, Kharkov, 2005

11. H. Gomaa, Designing Concurrent, Distributed, and Real-time Applications with UML, Addison-Wesley, 2000

12. Jacobson I. Object-Oriented Software Engineering. S.1.: ASM press., 1992. – 528 p.

13. Вендров А. М. CASE-технологии: современные методы и средства проектирования инфор¬мационных систем. – М. Финансы и статистика,

1998. – 175с.

14. Selic B., Gullekson G., Ward P. T. Real-Time Object Oriented Modelling. – S.1.: John Wiley & Sons, 1994. – 525 p.

15. Thomas M. Connolly, Carolyn E. Begg. Database Systems. A Practical Approach to Design, Implementation, and Management. Third Edition, Москва, издательский дом «Вильямс», 2003г., 1440 стр. с илл.

16. А. Федоров, Н. Елманова Введение в базы данных, журнал КомпьютерПресс 3'2000