

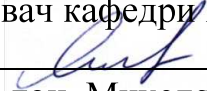
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування

імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"
Завідувач кафедри ІУСТ


к.т.н, доц. Михелєв І.Л.

" " 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: **Дослідження методів та технологій проектування інтерфейсів користувача на основі ERP-систем**

Виконав: студент 6 курсу, групи 6141м



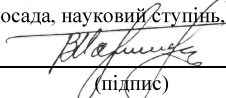
Багнюк І.В.

(підпис)

Керівник роботи:

К.Т.Н., доцент

(посада, науковий ступінь, вчене звання)



Партас В.К.

(підпис)

м. Миколаїв – 2024 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

В. Ажищев

к.т.н, доц. Ажищев В.Ф.

" ____ " ____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Багнюк І.В.

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів та технологій проектування інтерфейсів користувача на основі ERP-систем

Керівник роботи Партас В.К.

Затверджені наказом ректора № 1055-уч. від "08" жовтня 2024 року.

2. Термін подання роботи: 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

1. Дослідження інформаційних систем;

2. Розробка проектних рішень до інформаційної системи;

3. Результати дослідження методів та розробки інформаційної системи;

4. Розрахунок економічної ефективності впровадження інформаційної системи;

5. Охорона праці;

6. Охорона навколишнього середовища

5. Перелік презентаційних матеріалів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Партас В.К.к.т.н., доцент		
2	Партас В.К.к.т.н., доцент		
3	Партас В.К.к.т.н., доцент		
4	Партас В.К.к.т.н., доцент		
5	Партас В.К.к.т.н., доцент		
6	Партас В.К.к.т.н., доцент		

7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	01.11.2024	
2	Розробка концепції	14.11.2024	
3	Розділ з розробки проектних рішень по системі (та її частинах)	16.11.2024	
4	Реалізація проекту та розробка робочої документації	21.11.2024	
5	Обчислення економічної ефективності	14.11.2024	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2024	
7	Оформлення пояснювальної записки	26.11.2024	
8	Подання магістерської роботи на перевірку на плагіат	09.12.2024	
9	Подання магістерської роботи рецензенту	11.12.2024	
10	Подання магістерської роботи на державний захист	15.12.2024	

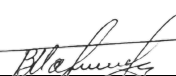
Студент


(підпис)

Багнюк І.В.

(ПІБ)

Керівник роботи


(підпис)

Партас В.К.

АНОТАЦІЯ

Магістерська дипломна робота на тему «Дослідження методів та технологій проектування інтерфейсів користувача на основі ERP-систем» присвячена аналізу існуючих методів для проектування інтерфейсів користувача, аналіз структури систем автоматизованого проектування, взаємозв'язок інтерфейсу з функціями, які виконуються в САПР, виявленню їх переваг та суттєвих недоліків. Об'єктом дослідження в роботі є процес проектування інтерфейсів систем автоматизованого проектування, предмет дослідження полягає у аналізі технологій та методів, які використовуються для проектування інтерфейсів користувача. У роботі проведено дослідження результатів впровадження систем, визначені покращення підприємств після впровадження якісно спроектованою інформаційної системи. Після дослідження проблем, які можуть вплинути на якість проектування, було запропоновано розробити інформаційну систему, яка б мала на меті створити файл конфігурації інтерфейсу системи автоматизованого проектування.

Метою роботи є підвищити продуктивність праці, забезпечити адекватне представлення дій і функцій системи. Актуальність роботи полягає в тому, що на даний момент не настільки багато уваги приділяється в розробці інтерфейсів саме їх зручності для користувачів. Користувачі, в залежності від сфери в котрій вони працюють, потребують і відповідну САПР, котра буде відповідати їх потребам. Отже існують багато моделей, правил проектування і принципів побудови інтерфейсів. Робота спрямована на те, щоб дослідити всі ці аспекти та розробити такі універсальні рекомендації по розробці САПР, які будуть корисні в будь-якій сфері використання.

Робота виконана українською мовою.

Ключові слова: інтерфейси, САПР, проектування, ERP-системи, комп'ютерні науки.

ABSTRACT

Master's thesis on "Research methods and technologies of interfaces to ERP systems" devoted to the analysis of existing methods for designing user interfaces, analysis of computer-aided design, interconnection interface with functions that occur in CAD, identify their strengths and significant weaknesses. The object of study is in the process of interface design computer-aided design, purpose of the study is to analyze technologies and methods used to design user interfaces. The paper studied the results of the implementation of identified improvements after the implementation of enterprise quality projected information system. After investigating the problems that can affect the quality of the design, it was proposed to develop an information system that was designed to create a configuration interface computer aided design.

The aim is to increase productivity, ensure adequate representation of actions and functions of the system. The relevance of the work lies in the fact that at the moment is not so much emphasis on the development of interfaces is their user-friendliness. Users, depending on the sector in which they operate, and require appropriate CAD, which will suit their needs. So there are many models rules and principles of design interfaces. Work intended to explore all these aspects and develop such universal recommendations for the development of CAD, which will be useful in any field of use.

This work was written in Ukrainian.

Keywords: interfaces, CAD, design, ERP systems, computer science.

ЗМІСТ

ВСТУП

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ ІНТЕРФЕЙСІВ СИСТЕМ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ТА АНАЛІЗ ЇХ ОСОБЛИВОСТЕЙ

- 1.1 Основні поняття систем автоматизованого проектування (САПР)
- 1.2 Аналіз особливостей користувацьких інтерфейсів
- 1.3 Аналіз методів та технологій проектування користувацьких інтерфейсів
- 1.4 Аналіз існуючих мов проектування
- 1.5 Аналіз підходів до проектування користувацьких інтерфейс

2 ФОРМУВАННЯ ВИМОГ ТА РОЗРОБКА КОНЦЕПЦІЇ ПРОЕКТУВАННЯ ЄДИНОЇ МОДЕЛІ ІНТЕРФЕЙСУ

- 2.1 Аналіз поняття користувацького інтерфейсу як системи
- 2.2 Аналіз особливостей проектування інформаційних систем
- 2.3 Аналіз методологій проектування інформаційних систем
- 2.4 Вибір методології проектування

3 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ КОНФІГУРАЦІЇ ІНТЕРФЕЙСУ

- 3.1 Загальносистемні рішення
- 3.2 Рішення з організаційного забезпечення
- 3.3 Рішення з інформаційного забезпечення
- 3.4 Рішення з технічного забезпечення
- 3.5 Рішення з програмного забезпечення

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ВСТУП

Інтерфейс користувача — це те середовище, той набір засобів, за допомогою яких користувач спілкується з комп'ютерною системою, ставить їй завдання й одержує результати. Це модель спілкування, стиль роботи й, звичайно, програми, що реалізують ці ідеї.

Основне завдання інтерфейсу — сховати від користувача технічні деталі й складності, що стосуються внутрішньої роботи системи, зробити спілкування із системою простим і зрозумілим навіть для непідготовленого користувача. Існує багато різновидів інтерфейсів.

Для САПР складність інтерфейсів полягає в тому, що вони містять багато функцій, що ускладнює розміщення команд на робочому просторі користувача для швидкого і зручного розуміння програми.

Магістерська дипломна робота присвячена аналізу існуючих методів для проектування інтерфейсів користувача, аналіз структури систем автоматизованого проектування, взаємозв'язок інтерфейсу з функціями, які виконуються в САПР, виявленню їх переваг та суттєвих недоліків.

Дана магістерська робота присвячена дослідженню методів та технологій проектування інтерфейсів користувача, доведенню необхідності вдосконалення вже існуючих методів, розробці інформаційної системи для моделювання конфігурацій інтерфейсу САПР. У роботі необхідно виконати аналіз існуючих методик та технологій проектування, визначити їх недоліки та переваги, прийти до можливості використання при проектуванні інтерфейсів методи та технології проектування ІС, обрати фактори та концепти впливу, на базі цього взаємовпливу зробити висновки щодо доцільності розробки системи моделювання файлу конфігурацій інтерфейсу.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати проблеми, які можуть виникнути при проектуванні інтерфейсів САПР.
2. Вивчити існуючі загальносвітові підходи, методи і системи, а також відповідні їм моделі проектування інтерфейсів користувача.

3. Привести доцільність використання UML методології для проектування інтерфейсів користувача.

3. Розробити методи та технології побудови моделей інтерфейсів користувача САД систем, провести аналіз.

5. Створити програмний продукт, який користувачу використовувати зручний йому інтерфейс за допомогою формування файлу конфігурацій інтерфейсу САПР.

1 ДОСЛІДЖЕННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ ІНТЕРФЕЙСІВ СИСТЕМ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ТА АНАЛІЗ ЇХ ОСОБЛИВОСТЕЙ

1.1 Основні поняття систем автоматизованого проектування (САПР)

САПР (CAD – computer Aided Design) – загальний термін для позначення усіх аспектів проектування з використанням засобів обчислювальної техніки. Зазвичай охоплює створення геометричних моделей виробу, а також генерацію креслень виробів. Треба вміти відрізняти, що термін САПР по відношенню промислових систем має більш поширене тлумачення ніж CAD. Він включає в себе як CAD так і CAM і CAE.

CAM (Computer Aided Manufacturing) – загальний термін для позначення систем автоматизованої підготовки виробництва. Традиційно вихідними даними для таких систем є геометричні моделі деталей, отримані з систем CAD.

CAE (Computer Aided Engineering) – система автоматичного аналізу проекту, загальний термін для позначення інформаційного забезпечення умов автоматизованого аналізу проекту, має за мету виявлення помилок чи оптимізація виробничих можливостей.

PDM (Product Data Management) – система управління виробничою інформацією, інструментальний засіб, котрий допомагає адміністраторам, інженерам, конструкторам та ін. керувати як даними та і процесами розробки виробу на сучасних виробничих підприємствах чи групі суміжних підприємств.

$CAD/CAM/CAE/PDM = САПР$. Схема САПР зображена на рисунку 1.1

Прогрес науки та техніки, потреби суспільства, що розвивається в нових промислових výroбах обумовлено необхідністю виконання проектних робіт. Вимоги до якості проектів, до термінів їх виконання стає все більш жорстоким при зростанні складності об'єктів, що проектуються. Окрім цього, темпи

морального старіння виробів сьогодні такі, що поставлені на конвеєр нові зразки частіше вже не відповідають сучасним вимогам.

Здійснення цих потреб стало можливим на основі широкого використання засобів ЕОМ на всіх етапах виробництва:

1) Контроль проектування, де зароджується вихідна модель виробу, технологічного проектування.

2) Проектування організації управління виробництвом з формуванням даних про матеріальні і інформаційні потоки виробництва.

3) Виготовлення виробів шляхом операцій над матеріальним об'єктом на основі створеної на попередніх етапах інформації.

4) Оцінка якості виробу на основі порівняння потрібних та реальних характеристик.

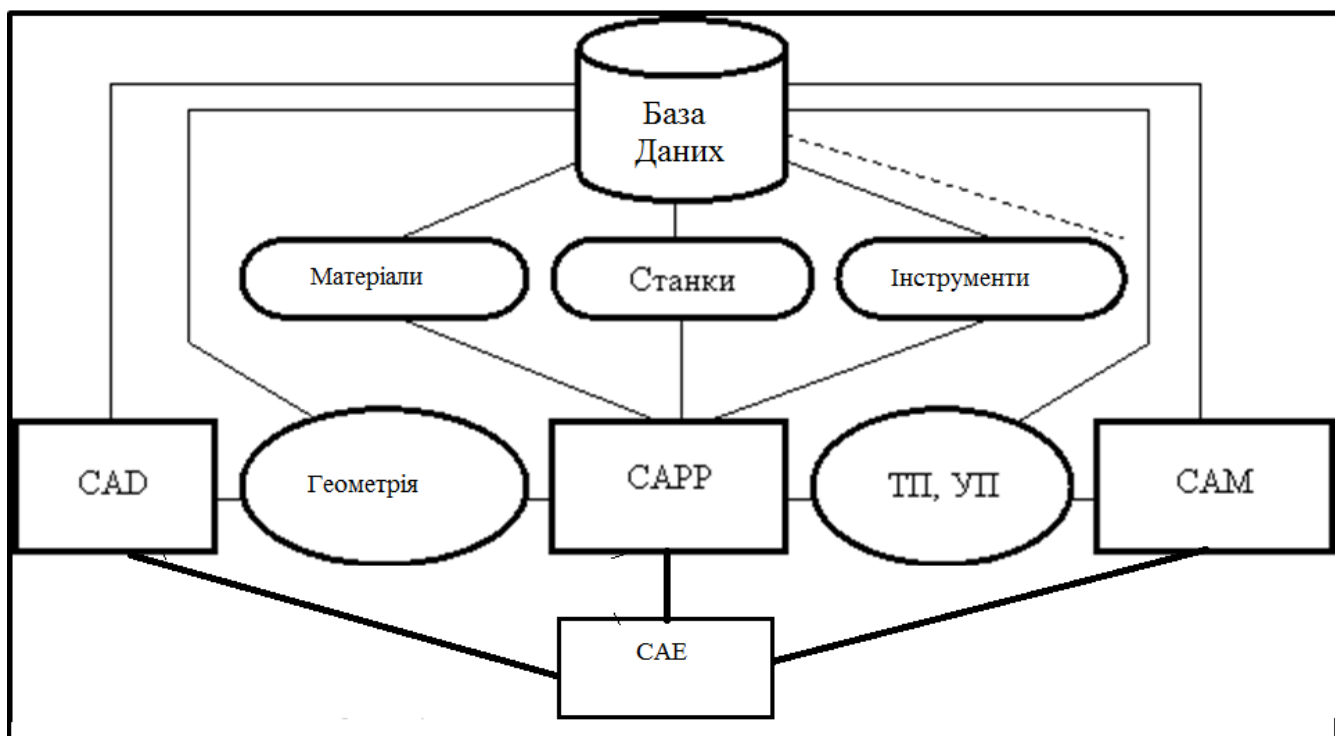


Рисунок 1.1 – Структурна схема САПР

Структура та види забезпечення САПР

Структура САПР. Як і будь-яка складна система, САПР складається з підсистем. Розрізняють підсистеми, котрі проектують та котрі обслуговують.

Підсистеми, котрі проектують, безпосередньо виконують проектні процедури. Прикладами підсистем, котрі проектують, можуть слугувати підсистеми геометричного тривимірного моделювання механічних об'єктів, виготовлення конструкторської документації, схематичного аналізу.

Підсистеми, котрі обслуговують, забезпечують функціонування підсистем, котрі проектують, і їх сукупність за частіше називають системним середовищем (або оболонкою) САПР. Типовими підсистемами, котрі обслуговують, є підсистеми керування проектними даними, підсистеми розробки й супроводу програмного забезпечення CASE (Computer Aided Software Engineering), підсистеми, котрі навчають, для освоєння користувачами технологій, реалізованих в САПР.

Види забезпечення САПР. Структурування САПР по різноманітним аспектам обумовлює те, що з'являються види забезпечення САПР. Прийнято виділяти сім видів забезпечення САПР:

- 1) Технічне (ТЗ), котре включає в себе різноманітні апаратні засоби (ЕОМ, периферійні пристрої, мережеве комутаційне обладнання, лінії зв'язку, вимірювальні засоби);

Вимоги до компонентів технічного забезпечення:

До компонентів технічного забезпечення відносять пристрої обчислювальної та організаційної техніки, засоби передачі даних, вимірювальні та інші пристрої і їх поєднання, що забезпечують функціонування ПТК і ксапів, в тому числі діалоговий, багатокористувацький і багатозадачний режим роботи, а також побудова ієрархічних і мережевих структур технічного забезпечення.

В якості кращою для САПР слід використовувати дворівневу структуру технічного забезпечення, що включає центральний обчислювальний комплекс і автоматизовані робочі місця (термінальні станції).

Компоненти технічного забезпечення повинні надавати можливість: кодування і введення інформації з її візуальним контролем і редагуванням; передачі інформації по різних каналах зв'язку, зберігання, контролю і

відновлення інформації, завантаження, зберігання та виконання програмного забезпечення, оперативного надання запитуваної інформації на пристрої виводу.

2) Математичне (МЗ), котре об'єднує математичні методи, моделі та алгоритми для виконання проектування. Основа - це алгоритми, за якими розробляється програмне забезпечення САПР. Серед різноманітних елементів математичного забезпечення є інваріантні елементи-принципи побудови функціональних моделей, методи чисельного рішення алгебраїчних і диференціальних рівнянь, постановки екстремальних завдань, пошуки екстремуму. Розробка математичного забезпечення є найскладнішим етапом створення САПР, від якого найбільшою мірою залежать продуктивність і ефективність функціонування САПР в цілому.

Вимоги до компонентів математичного забезпечення:

До компонентів математичного забезпечення відносять методи математичного моделювання об'єктів і процесів проектування, математичні моделі об'єктів і процесів проектування, алгоритми рішення задач в процесі проектування.

Взаємозв'язки між компонентами математичного забезпечення повинні забезпечувати формалізацію процесу проектування і його цілісність.

3) Програмне (ПЗ), котре представлене комп'ютерними програмами САПР. Програмне забезпечення САПР являє собою сукупність всіх програм та експлуатаційної документації до них, необхідних для виконання автоматизованого проектування. Програмне забезпечення ділитися на загальносистемне та спеціальне (прикладне) ПЗ. Загальносистемне ПЗ призначено для організації функціонування технічних засобів, т. тобто для планування і керування обчислювальним процесом, розподілу наявних ресурсів, що представлено різними операційними системами. У спеціальному ПО реалізується математичне забезпечення для безпосереднього виконання проектних процедур.

Вимоги до компонентів програмного забезпечення:

Компоненти програмного забезпечення, об'єднані в програмно-методичний комплекс (ПМК) та програмно-технічних комплексів (ПТК), повинні мати ієрархічну організацію, в якій на верхньому рівні розміщується монітор управління компонентами нижніх рівнів програмними модулями.

Програмний модуль повинен: регламентувати функціонально закінчений перетворення інформації; бути написаним на одному з стандартних мов програмування; задовольняти угодам про представлення даних, прийнятим в даній бути оформленим відповідно до вимог ЕСДП.

Монітор призначений для: управління функціонуванням набору програмних модулів ПМК, включаючи контроль послідовності і правильності виконання, реалізації спілкування користувача з ПМК і програмних модулів з відповідними базами даних (БД); збору статистичної інформації.

4) Інформаційне (ІЗ), котре складається з бази даних, СУБД, а також містить в собі інші дані, котрі використовуються при проектуванні; відмітимо, що вся сукупність даних, котрі використовуються при проектуванні, називається інформаційним фондом САПР, база даних разом з СКБД називається банком даних. Основу становлять дані, якими користуються проектувальники в процесі проектування безпосередньо для вироблення проектних рішень. Ці дані можуть бути представлені у вигляді тих чи інших документів на різних носіях, які містять відомості довідкового характеру про матеріали, параметрах елементів, відомості про стан поточних розробок у вигляді проміжних і остаточних проектних рішень.

Вимоги до компонентів інформаційного забезпечення:

Основною формою реалізації, компонентів інформаційного забезпечення є БД в розподіленій або централізованій формі, організація даних в яких забезпечує їх оптимальне використання в конкретних застосуваннях.

Сукупність БД САПР повинна задовольняти принципом інформаційного єдності, т. тобто використовувати терміни, символи, класифікатори, умовні позначення, способи представлення даних, прийняті в САПР об'єктів конкретних видів.

Незалежно від логічної організації даних БД повинні забезпечувати:

- Інформаційну сумісність проектують і обслуговують підсистем САПР;
- Незалежність даних на логічному і фізичному рівнях, у тому числі інваріантність до програмного забезпечення;
- Можливість одночасного використання даних з різних БД і різними користувачами;
- Можливість інтеграції неоднорідних БД для спільного їх використання різними підсистемами САПР;
- Можливість нарощування БД;
- Контрольовану надмірність даних.

Створення, підтримка і використання БД, а також взаємозв'язок між інформацією в БД і обробними її програмними модулями здійснюється системою управління базами даних (СКБД), яка є, як загальносистемний ПМК, частиною однієї з обслуговуючих підсистем.

5) Лінгвістичне (ЛЗ), котре виражене мовами спілкування між проектувальниками і ЕОМ, мовами програмування і мовами обміну даними між технічними засобами САПР.

Вимоги до компонентів лінгвістичного забезпечення:

До компонентів лінгвістичного забезпечення відносять мови проектування (МП), інформаційно-пошукові мови (ІПМ), і допоміжні мови, що використовуються в обслуговуючих підсистемах, і для зв'язку з ними проектують підсистем.

Компоненти лінгвістичного забезпечення повинні бути узгодженими з компонентами забезпечення інших видів, бути відносно інваріантними до конкретного змісту баз даних, надавати в компактній формі засоби для опису всіх об'єктів і процесів заданого для систем класу з необхідним ступенем деталізації і без суттєвих обмежень на об'єкт опису, бути розрахованими, в основному, на діалоговий режим їх використання.

МП повинні базуватися на термінах, прийнятих в конкретній системі, забезпечувати опис, управління і контроль процесу проектування, бути

орієнтованими на користувачів з різним рівнем професійної підготовки (в тому числі не мають спеціальної підготовки в області програмування), забезпечувати однозначне представлення інформації, стандартне опис однотипних елементів і високу надійність ідентифікації опису.

МП повинні являти собою набір директив, використовуючи які користувач здійснює процес формування моделі об'єкта проектування та її аналіз, забезпечувати можливість ефективного контролю завдань користувача, мати засоби видачі користувачеві довідок, інструкцій і повідомлень про помилки, передбачати можливість використання механізму вибору альтернативних директив з певного набору (функціональна клавіатура та ін).

ІПМ повинні включати словники, правила індексування вхідної інформації і правила формування пошукових приписів.

Словники ІПМ повинні містити терміни (у тому числі стандартизовані) відповідної галузі техніки й інші лексичні, одиниці, необхідні для індексування та пошуку проектної інформації з високою точністю і повнотою.

б) Методичне, котре містить в собі різноманітні методики проектування; іноді до нього відносять також математичне забезпечення. Під методичним забезпеченням САПР розуміють що входять до її складу документи, що регламентують порядок її експлуатації. Причому документи, що відносяться до процесу створення САПР, не входять до складу методичного забезпечення. Так в основному документи методичного забезпечення носять інструктивний характер, і їх розробка є процесом творчим.

Вимоги до компонентів методичного забезпечення:

До компонентів методичного забезпечення відносять: затверджену документацію інструктивно-методичного характеру, яка встановлює технологію автоматизованого проектування, правила експлуатації комплексу засобів автоматизованого проектування, ПМК, нормативи, стандарти та інші керівні документи, що регламентують процес і об'єкт проектування.

Компоненти методичного забезпечення повинні розміщатися на «машинних носіях інформації, що дозволяють здійснювати як довгострокове збереження документів, так і їх оперативний вивід в форматах, встановлених відповідними стандартами.

7) Організаційне, котре представляється штатним розкладом, посадовими інструкціями та іншими документами, котрі регламентують роботу проектного підприємства. Цей пункт приписує комплектування підрозділів САПР професійними грамотними фахівцями, мають навички та знання для роботи з перерахованими вище компонентами САПР. Від їх роботи буде залежати ефективність і якість роботи всього комплексу САПР (може навіть всього виробництва).

Вимоги до компонентів організаційного забезпечення:

Компоненти організаційного забезпечення повинні встановлювати організаційну структуру системи і підсистем, включаючи взаємозв'язок її елементів, завдання та функції служби САПР і пов'язаних з нею підрозділів проектної організації, права та відповідальність посадових осіб щодо забезпечення створення і функціонування САПР, порядок підготовки і перепідготовки користувачів САПР.

Класифікація САПР

1) САПР двовимірного проектування - « 2D-3D Легені - Нижній рівень»

Ці САПР служать для виконання майже всіх робіт з двовимірними кресленнями і мають обмежений набір функцій по тривимірному моделюванню. За допомогою цих систем виконуються близько 90% всіх робіт з проектування. Хоча наявні обмеження роблять їх не завжди досить зручними. Область їх роботи - створення креслень окремих деталей і зборок. Платою за зрослі можливості є ускладнення інтерфейсу і меншу зручність у роботі. Характерні представники таких САПР - AutoCAD, CADdy, CADMECH Desktop, MasterCAM, T- FlexCAD, OmniCAD, Компас- Графік.

2) САПР об'ємного моделювання « 3D - Середній рівень»

За своїми можливостями вони повністю охоплюють САПР «легкої ваги», а також дозволяють працювати зі збірками, за деякими параметрами вони вже не поступаються важким САПР, а в зручності роботи навіть перевершують. Обов'язковою умовою є наявність функції обміну даними (або інтеграції). Це не просто програми, а програмні комплекси, зокрема, SolidWorks SolidEdge, Cimatron, Form - Z, Autodesk Inventor, CAD SolidMaster, і все ще продовжує розвиватися, Mechanical Desktop, DesignSpace.

3) САПР об'ємного моделювання «3D Важкі - Верхній рівень»

Ці системи застосовуються для вирішення найбільш трудомістких завдань - моделювання поведінки складних механічних систем у реальному масштабі часу, що оптимізують розрахунків з візуалізацією результатів, розрахунків температурних полів і теплообміну і т.д. Зазвичай до складу системи входять як чисто графічні, так і модулі для проведення розрахунків і моделювання, постпроцесори для верстатів з ЧПУ. На жаль, ці найпотужніші САПР найбільш громіздкі та складні в роботі, а також мають значну вартість. Прикладами «важких» САПР можуть служити такі продукти, як ADAMS, ANSYS, CATIA, EUCLID3, Pro/ENGINEER, UniGraphics.

Класифікація САПР за рівнем складності зображена на рисунку 1.2

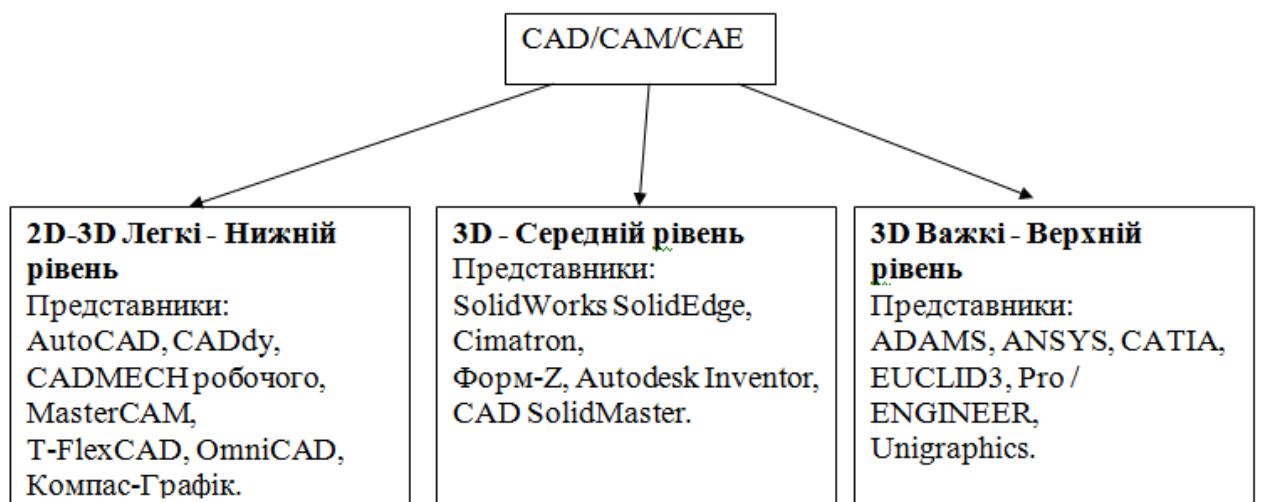


Рисунок 1.2 – Класифікація САПР за рівнем складності

Галузі використання САПР

Частіше за все САПР використовують в машинобудуванні (для цієї галузі ЕОМ використовується для вирішення двох задач – виготовлення креслень, проведення розрахунків) та автомобілебудуванні:

1) Авіабудування. В зв'язку зі складністю використовуються декілька систем для:

- Конструювання обводів фюзеляжу;
- Конструювання деталей з створюванням програм для станків;
- Виготовлення креслень;
- Моделювання процесів

Завдяки використанню фрезерних станків стало можливим виготовлення деталей легких і міцних. Отримати це із традиційного конструювання на базі листових заготовок неможливе.

2) Суднобудування.

- Ескізне проектування (опис корпусу): оцінка технічних і економічних даних, наближений опис корпусу, гідростатичні і гідродинамічні розрахунки;

- Проектування, параметризація корпусу: визначення розмірів, прийняття конструктивних рішень, збірочні та монтажні креслення, кінцеве визначення форми корпусу;

- Деталювання: конструювання деталей та технологічна підготовка, кастинг – мінімізація відходів розкрою, робочі креслення

3) Електротехніка. Особливе положення займають вироби з / п та електронної техніки. Тут розробляються різноманітні прості системи, які охоплюють весь виробничий процес. Прилади енергетики, прилади обробки інформації, обробки зв'язку, пром. і допом. електроніка, ширвжиток. САПР, яка здійснює автоматичний або напівавтоматичний контроль електричної схеми. Особливо ефективні ці системи, якщо проводити періодичні і непр. одні й ті ж операції, наприклад, включити виконання логічних символів, і т.п., системи конструювання друкованих плат, логічних схем, моделювання

електричних схем, проектування структурних інтегрованих схем, розрахунок параметрів інтегрованих схем і монтажних плат.

4) Будівництво: архітектура, проектування робочої документації, планування внутрішніх площ, планування комунікацій. Застосування САПР полегшує працю архітектора, допомагаючи йому здійснити вибірку всіх проміжних результатів. У прийнятті рішення йому допомагає наочне представлення результатів, швидке програвання альтернатив. Завдяки 3D можна отримати різні види, визначивши площину перерізу. 3D опис є передумовою для зображення внутрішніх приміщень при розгляді його з різних точок. При будівництві інт. споруд виникає необхідність обчислень. Застосування розрахункового прогнозування дозволяє скоротити витрату конструкційної сталі приблизно на 35%. Для планування сантехнічного, опалювального та іншого обладнання застосовуються як 2D так і 3D САПР. Загальний план, монтаж електротехнічного обладнання, монтаж вентиляційного обладнання, текстова інформація.

Переваги САПР

Зараз терміном САПР позначають процес проектування з використанням складних засобів машинної графіки, підтримуваних пакетами прикладних програм для вирішення на комп'ютерах аналітичних, кваліфікаційних, економічних і ергономічних проблем, пов'язаних з проектною діяльністю.

Переваги САПР:

1) Більш швидке виконання креслень (до 3 разів). Дисципліна роботи з використанням САПР прискорює процес проектування в цілому, дозволяє в стислі терміни випускати продукцію і швидше реагувати на зміну ринкових кон'єктури.

2) Підвищення точності виконання. На кресленнях, побудованих за допомогою системи САПР, місце будь-якої точки визначено точно, а для збільшення достатнього перегляду елементів є засіб, зване наїзд, або

масштабування, що дозволяє збільшувати або зменшувати будь-яку частину даного креслення в будь-яке число раз. На зображення, над яким виконується наїзд, не накладається практично ніяких обмежень.

3) Підвищення якості;

4) Можливість багаторазового використання креслення. Збережене креслення може бути використаний повторно для проектування, коли до складу креслення входить ряд компонентів, що мають однакову форму. Пам'ять комп'ютера є також ідеальним засобом зберігання бібліотек, символів, стандартних компонентів і геометричних форм.

5) САПР володіє креслярськими засобами (сплайни, сполучення, шари).

6) Прискорення розрахунків і аналізу при проектуванні. В даний час існує велика різноманітність ПЗ, яке дозволяє виконувати на комп'ютерах частина проектних розрахунків заздалегідь. Потужні засоби комп'ютерного моделювання, наприклад, метод кінцевих елементів, звільняють конструктора від використання традиційних форм і дозволяють проектувати нестандартні геометричні форми.

7) Зниження витрат на оновлення. Засоби аналізу та імітації в САПР, дозволяють різко скоротити витрати часу та грошей на тестування та вдосконалення прототипів, які є дорогими етапами процесу проектування;

8) Великий рівень проектування. Потужні засоби, комплексного моделювання. Можливість проектування нестандартних геометричних форм, які швидко оптимізуються;

9) Інтеграція проектування з іншими видами діяльності. Інтегровані обчислювальні засоби забезпечують САПР більш тісне взаємодії з інженерними підрозділами.

Роль САПР в виробничому циклі

Сукупність видів діяльності та функцій, необхідних для здійснення проекту і виготовлення виробів, називається виробничим циклом. Поведінка цього циклу визначається замовниками виробів і потребами ринків збуту.

Виробничий цикл може активізуватися різними шляхами в залежності від особливостей тієї чи іншої групи замовників-споживачів. В одних випадках функції проектування виконує сам замовник, а виготовлення виробу - сторонні фірми, в інших і проектування і виробництво виконує одна і та ж фірма. Однак, як би це не відбувалося, виробничий цикл завжди починається з вироблення концепції нового виробу (тобто виникнення певної ідеї). Спочатку концепція нового виробу опрацьовується, потім в деталях уточнюється, аналізується, удосконалюється і після конструктивної розробки втілюється в план виконання нового виробу. Цей план підкріплений документацією у вигляді набору конструктивних креслень, що показують, як повинно виготовлятися виріб і сукупність технічних описів, специфікацій, що відображають принципи функціонування виробу. Цим завершується діяльність по проектуванню, якщо не вважати конструкцію виробу протягом усього життєвого циклу.

Наступним видом діяльності стає виготовлення виробу, яке починається складанням плану, що визначає послідовність операцій, необхідних для виконання виробу. Іноді можуть знадобитися нове обладнання, інструмент і оснащення.

На етапі складання графіка виробництва обраного плану, згідно з яким фірма фактично бере на себе зобов'язання випустити певну кількість виробів у конкретні терміни. Після того, як сформовані плани-графіки по всім виробам, вони запускаються у виробництво, і потім проходять через операції контролю якості і відвантаження замовнику.

Авт. проектування і авт. виготовлення креслень та документації за допомогою ЕОМ має місце на етапах вироблення концепції нового виробу, конструювання виробу, розробки креслень. Метою САПР є видача конструкторської документації, яка може бути результатом пошуку аналогів (прототипу) або результатом конструювання оригінального виробу. З використанням САПР проектується 3D-модель деталі, виходять необхідні види моделей, виконується деталювання і деталювальні креслення із

застосуванням різних можливостей систем інженерного аналізу, наприклад, із застосуванням аналізу МКЕ і кінематики.

Технологічна підготовка. Надзвичайно різноманітні завдання АСТПП, до яких відноситься планування і використання технологічного обладнання, підготовка або проектування і виготовлення інструменту. Підготовка програм для тих. обладнання з ЧПУ, програм для роботів. На етапах планування произв. процесу і календарного випуску вироби ЕОМ використовується для забезпечення більш ефективного виконання цих функцій.

САПР – є складними інтегрованими ситемами, тому часто виникають такі проблеми, як передача даних між різними САПР, відмінне подання даних та відмінні підходи до побудови інтерфейсів. Процес проектування інтерфейсу САПР є дуже складним, тому що треба поєднати багато функцій, котрі виконують САПР. Виникає така задача – створити єдину модель для проектування інтерфейсів САПР.

1.2 Аналіз особливостей користувальницьких інтерфейсів

Інтерфейс користувача - це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії складають діалоги. Під діалогом в даному випадку розуміють регламентований обмін інформацією між людиною і комп'ютером, здійснюваний в реальному масштабі часу і спрямований на спільне вирішення конкретного завдання. Кожен діалог складається з окремих процесів введення/виводу, які фізично забезпечують зв'язок користувача і комп'ютера. Обмін інформацією здійснюється передачею повідомлення. Взаємодія людини з ЕОМ зображена на рисунку 1.3



Рисунок 1.3 – Схема взаємодії людини з ЕОМ

В основному користувач генерує повідомлення наступних типів:

- запит інформації
- запит допомоги
- запит операції або функції
- введення або зміна інформації

У відповідь користувач отримує підказки або довідки; інформаційні повідомлення, які потребують відповіді; накази, що вимагають дії; повідомлення про помилки та іншу інформацію.

Інтерфейс користувача комп'ютерного додатка включає:

- 1) засоби відображення інформації, що відображається інформацію, формати і коди;
- 2) командні режими, мова "користувач - інтерфейс";
- 3) пристрою і технології введення даних;
- 4) діалоги, взаємодія і транзакції між користувачем і комп'ютером, зворотний зв'язок з користувачем;
- 5) підтримку прийняття рішень у конкретній предметній області;
- 6) порядок використання програми і документацію на неї.

Інтерфейс користувача (КІ) часто розуміють тільки як зовнішній вигляд програми. Однак на ділі користувач сприймає через нього всю програму в цілому, а значить, таке розуміння є занадто вузьким. Насправді КІ об'єднує в собі всі елементи і компоненти програми, які здатні впливати на взаємодію користувача з програмним забезпеченням (ПЗ).

Це не тільки екран, який бачить користувач. До цих елементів відносяться:

- 1) набір завдань користувача, які він вирішує за допомогою системи;
- 2) використовувана системою метафора (наприклад, робочий стіл у MS Windows ®);
- 3) елементи управління системою;
- 4) навігація між блоками системи;
- 5) візуальний (і не тільки) дизайн екранів програми;
- 6) засоби відображення інформації, яка відображається інформація та формати;
- 7) пристрою і технології введення даних;
- 8) діалоги, взаємодія і транзакції між користувачем і комп'ютером;
- 9) зворотній зв'язок з користувачем;
- 10) підтримка прийняття рішень у конкретній предметній області;
- 11) порядок використання програми і документація на неї.

Аналіз недоліків існуючих систем автоматизованого проектування

Майже всі виробники CAD-систем створювали системи від початку до кінця, включаючи управління пам'яттю, драйвери пристроїв, інтерфейс користувача, інтерпретатор для мов. Жоден продукт не має інтерфейсу в стилі windows; строчно-командний інтерфейс зберігся, і все ще залишається необхідним. Найбільш важливий вплив на CAD зробила WINDOWS - оболонка. Наявність мільйонів комп'ютерів для WINDOWS, стимулює розробку швидких процесорів, великих моніторів, став більш ємним сам процес створення CAD, звільняючи користувача від турботи про драйвери

пристроїв, і певною мірою про інтерфейс. В даний час збільшується інтеграція функцій з CAD в ос. Найбільш відомий приклад - OPENGL в WINDOWS. Вже зараз opengl - найлегший і універсальний метод візуалізації об'єктів CAD. Він має дві переваги:

- незалежність від обладнання;
- усунення обробки дисплейних списків;

Негативну роль у розвитку CAD-систем грають недоліки в організації по CAD. Це обмежена відкритість для розширень, труднощі налаштування на національні стандарти, погана модульність. Пакет CAD без адаптації не дуже корисний для більшої частини користувачів, тому існує ринок виробників, які виконують вертикальну адаптацію для різних застосувань cad-систем. Зазвичай до складу систем загального призначення включався інструментарій для створення і підключення додатків, а також для адаптації і розширення інтерфейсу з користувачем. Однак існують дві причини по яких ця адаптація утруднена:

- 1) Додаток доводиться створювати поверх всій базової системи, хоча багато її функції не потрібні. Ціна навіть невеликий розробки в основному визначається вартістю базового пакета;
- 2) Інструментальні засоби виготовляються виробником базової системи самостійно, були не стандартними і не якісними.

Принципи проектування користувацьких інтерфейсів

При проектуванні користувацьких інтерфейсів слід дотримуватися принципів, які зведені в Таблиці 1.1

Таблиця 1.1 – Принципи проектування інтерфейсу користувача

Принцип	Опис
Облік знань користувача	Необхідно використовувати терміни і поняття, взяті з досвіду майбутніх користувачів системи, а об'єкти, керовані системою, повинні бути безпосередньо пов'язані з робочим середовищем користувача.

Узгодженість	Команди меню системи повинні бути одного формату, параметри повинні передаватися в усі команди однаково і пунктуація команд повинна бути схожою. Зручно, коли для всіх типів об'єктів системи підтримуються однакові методи.
Мінімум несподіванок	Одна й та ж дія в різних ситуаціях повинно призводити до аналогічної реакції.
Здатність до відновлення	У інтерфейсах повинні бути кошти запобігати помилки користувача, а також дозволяють коректно відновити інформацію після помилок. Це підтвердження деструктивних дій і можливість скасування дій.
Керівництво користувача	Інтерфейс повинен надавати необхідну інформацію у разі помилок користувача і підтримати кошти контекстно-залежної довідки
Облік різноманітності користувача	В інтерфейсі повинні бути кошти для зручного взаємодії з користувачами, що мають різний рівень кваліфікації і різні можливості

Для того щоб КІ був якісним (ефективним), він повинен мати такі властивості [20]:

1) Природність.

Природний інтерфейс - такий, який не змушує користувача істотно змінювати звичні для нього способи вирішення завдання. Це, зокрема, означає, що повідомлення і результати, що видаються додатком, не повинні вимагати додаткових пояснень. Доцільно також зберегти систему позначень і термінологію, що використовуються в даній предметній області.

2) Узгодженість .

А) узгодженість в межах продукту

Одна і та ж команда повинна виконувати одні й ті ж функції, де б вона не зустрілася .

Б) узгодженість в межах робочого середовища

Підтримуючи узгодженість з інтерфейсом , наданим операційною системою (WINDOWS), додаток може спиратися на те знання та навички користувача , які він отримав раніше при роботі з іншими додатками.

В) узгодженість у використанні метафор

Метафора - знайоме користувачеві поняття або образ, що допомагає забезпечити інтуїтивно зрозумілий інтерфейс (наприклад, кошик windows). Якщо поведінка деякого програмного об'єкта виходить за рамки того , що зазвичай мається на увазі під відповідною йому метафорою, у користувача можуть виникнути труднощі при роботі з таким об'єктом. (Наприклад , якщо у програмного об'єкта кошик визначити операцію запуск, користувач не зрозуміє призначення об'єкта без сторонньої допомоги).

3) Дружність інтерфейсу (принцип « прощення» користувача).

Ефективний інтерфейс повинен брати до уваги , що більшість користувачів вивчають особливості роботи з новим програмним забезпеченням методом проб і помилок. На кожному етапі роботи інтерфейс повинен дозволяти тільки відповідний набір дій і попереджати користувачів про тих ситуаціях, де вони можуть пошкодити системі або даним; ще краще, якщо у користувача існує можливість скасувати або виправити виконані дії .

Дружній інтерфейс в САПР. Приязна до користувача система забезпечує:

- задовільний інтерфейс з користувачем, де всі програми зв'язуються з користувачем однаковим чином; ключові слова, терміни і ключі управління мають однаковий зміст у всіх додатках;
- легко доступні екрани, підказки, поради і меню для керівництва початківцями користувачами без відсилання їх до друкованої документації;
- скасування всіх меню і порад досвідченим користувачам для підвищення швидкості роботи;
- спілкування з користувачем на звичному для нього мовою, а не на жаргоні програмістів;

- логічну послідовність етапів проектування, подібну з процесом проектування вручну.

4) Принцип « зворотного зв'язку».

Завжди слід забезпечувати зворотний зв'язок для дій користувача . Кожна дія користувача повинно отримувати візуальне, а іноді і звукове підтвердження того, що програмне забезпечення сприйняло введену команду; при цьому вид реакції, по можливості , повинен враховувати природу виконаної дії .

Зворотній зв'язок ефективна в тому випадку, якщо вона реалізується вчасно , тобто як можна ближче до точки останнього взаємодії користувача з системою. Коли комп'ютер обробляє надійшов завдання, корисно надати користувачеві інформацію щодо поточного стану процесу, а також можливість перервати цей процес у разі потреби.

5) Простота

Під простотою розуміється легкість у вивченні і використанні інтерфейсу. Крім того, він повинен надавати доступ до всього переліку функціональних можливостей програми. Один з можливих шляхів реалізації простоти - представлення на екрані інформації, мінімально необхідної для виконання чергового кроку завдання . Елементи повинні розміщуватися на екрані з обліків їх смислового значення і логічного взаємозв'язку .

6) Гнучкість

Гнучкість інтерфейсу - це його здатність враховувати рівень підготовки і продуктивність праці користувача. Властивість гнучкості припускає можливість зміни структури діалогу і / або вхідних даних.

7) Естетична привабливість.

Здатність інтерфейсу додатку забезпечити формування на екрані такого середовища, яка не тільки сприяла б розумінню користувачем представленої інформації, але і дозволяла б зосередитися на найбільш важливих її аспектах [4].

Оцінювання якості користуvalьницького інтерфейсу

Якість пі оцінюється на основі наступних приватних показників:

1) час, необхідний певному користувачеві для досягнення заданого рівня знань і навичок при роботі з додатком (наприклад , особа бойового розрахунку об'єкта ів повинно освоїти систему команди додатка, встановленого на його АРМ за 8 годин).

2) збереження отриманих робочих навичок після закінчення деякого часу (наприклад , після тижневої перерви АРМ повинно виконати певну послідовність операцій за заданий час) .

3) швидкість вирішення завдання за допомогою даного додатки (наприклад, АРМ повинно обробити за годину не менше 10 документів з помилкою не більше 1%).

4) суб'єктивна задоволеність користувача при роботі з системою, яка кількісно може бути виражена у відсотках або оцінкою за n - бальною шкалою.

Опис існуючих видів інтерфейсів

Інтерфейс - це, перш за все, набір правил. Як будь-які правила, їх можна узагальнити, зібрати в " кодекс", згрупувати за спільною ознакою . Таким чином, ми прийшли до поняття " вид інтерфейсу " як об'єднання по схожості способів взаємодії людини і комп'ютерів. Коротко можна запропонувати наступну схематичну класифікацію різних інтерфейсів спілкування людини і комп'ютера.

Залежно від типу користувача інтерфейсу інформаційні технології мають відповідну класифікацію, що зображена на рисунку 1.4. При цьому виділяється системний і прикладної інтерфейс.

Прикладний інтерфейс пов'язаний з реалізацією деяких функціональних інформаційних технологій .

Системний інтерфейс - це набір прийомів взаємодії з комп'ютером , який реалізується операційною системою або її надбудовою.

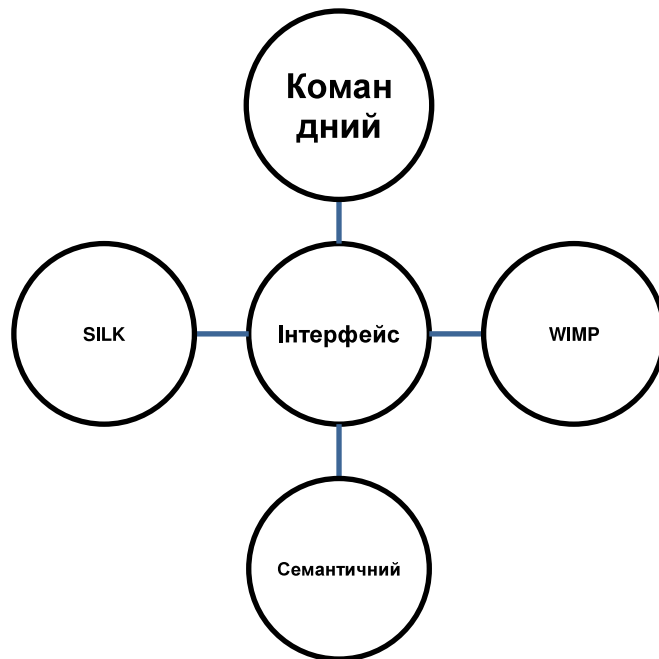


Рисунок 1.4 – Класифікація видів інтерфейсів

Сучасними видами інтерфейсів є:

1) Командний інтерфейс. Командний інтерфейс називається так по тому, що в цьому виді інтерфейсу людина подає "команди" комп'ютера, а комп'ютер їх виконує і видає результат людині . Командний інтерфейс реалізований у вигляді пакетної технології та технології командного рядка. Командний інтерфейс - найпростіший. Він забезпечує видачу на екран системного запрошення для введення команди . Наприклад, в операційній системі MS-DOS запрошення виглядає як C:\>, а в операційній системі UNIX - це зазвичай знак долара .

Колись раніше поширений командний інтерфейс має ряд істотних недоліків з точки зору користувача: численність команд, відсутність стандарту для додатків і т.д. Все це що обмежує коло його застосування.

Для подолання недоліків були зроблені спроби спрощення командного інтерфейсу. Так з'явилися спеціальні програмні оболонки, що полегшують спілкування користувача з операційною системою (програма Norton Commander та ін.)

Справжнім же рішенням проблеми стало створення та впровадження графічної оболонки для операційної системи. Програми, що мають командний

інтерфейс , називають консольними. Такий інтерфейс реалізують операційні системи MS DOS , Linux (без запуску графічної оболонки) .

2) WIMP - інтерфейс (Window - вікно , Image - образ , Menu - меню , Pointer - покажчик) . Характерною особливістю цього виду інтерфейсу є те , що діалог з користувачем ведеться не за допомогою команд , а за допомогою графічних образів - меню , вікон , інших елементів. Хоча і в цьому інтерфейсі подаються команди машині , але це робиться " опосередковано " , через графічні образи . Цей вид інтерфейсу реалізований на двох рівнях технологій : простий графічний інтерфейс і "чистий" WIMP - інтерфейс.

3) SILK - інтерфейс (Speech - мова, Image - образ, Language - мова, Knowledge - знання). Цей вид інтерфейсу найбільш наближений до звичайної, людської форми спілкування. У рамках цього інтерфейсу йде звичайний "розмова" людини і комп'ютера. При цьому комп'ютер знаходить для себе команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд він також перетворює в зрозумілу людині форму. Цей вид інтерфейсу найбільш вимогливий до апаратних ресурсів комп'ютера, і тому його застосовують в основному для військових цілей.

Опис особливостей командного інтерфейсу

Пакетна технологія . Історично цей вид технології з'явився перший . Вона існувала вже на релейних машинах Зюса і Цюзе (Німеччина , 1937 рік). Ідея її проста: на вхід комп'ютера подається послідовність символів , в яких за певними правилами вказується послідовність запусканих на виконання програм . Після виконання чергової програми запускається наступна і т.д. Машина за певними правилами знаходить для себе команди і дані. В якості цієї послідовності може виступати, наприклад, перфолента, стопка перфокарт, послідовність натиснення клавіш електричної друкарської машинки (типу CONSUL). Машина також видає свої повідомлення на перфоратор, алфавітно-цифровий друкуючий пристрій (ЦПУ), стрічку друкарської машинки. Така машина являє собою " чорний ящик" (точніше "білий шафа"), в який постійно

подається інформація і яка також постійно "інформує" світ про свій стан. Людина тут має малий вплив на роботу машини - він може лише призупинити роботу машини, змінити програму і знову запустити ЕОМ. Згодом, коли машини стали потужніший і могли обслуговувати відразу декількох користувачів, вічне очікування користувачів типу: "Я послав дані машині. Чекаю, що вона відповість. І чи відповість взагалі?" - Стало, м'яко кажучи, набридати. До того ж обчислювальні центри, слідом за газетами, стали другим великим "виробником" макулатури. Тому з появою алфавітно-цифрових дисплеїв почалася ера по-справжньому користувальницької технології - командного рядка.

Технологія командного рядка. При цій технології в якості єдиного способу введення інформації від людини до комп'ютера служить клавіатура, а комп'ютер виводить інформацію людині з допомогою алфавітно-цифрового дисплея (монітора). Цю комбінацію (монітор + клавіатура) стали називати терміналом, або консоллю. Команди набираються в командному рядку. Командний рядок являє собою символ запрошення і миготливий прямокутник - курсор. При натисканні клавіші на місці курсору з'являються символи, а сам курсор зміщується вправо. Це дуже схоже на набір команди на друкарській машинці. Однак, на відміну від неї, букви відображаються на дисплеї, а не на папері, і неправильно набраний символ можна стерти. Команда закінчується натисканням клавіші Enter (або Return). Після цього здійснюється перехід на початок наступного рядка. Саме з цієї позиції комп'ютер видає на монітор результати своєї роботи. Потім процес повторюється. Технологія командного рядка вже працювала на монохромних алфавітно-цифрових дисплеях. Оскільки вводити дозволялося тільки букви, цифри і розділові знаки, то технічні характеристики дисплея були не істотні. В якості монітора можна було використовувати телевізійний приймач і навіть трубку осцилографа.

Обидві ці технології реалізуються у вигляді командного інтерфейсу - машині подаються на вхід команди, а вона як би "відповідає" на них.

Переважним видом файлів при роботі з командним інтерфейсом стали текстові файли - їх і тільки їх можна було створити за допомогою клавіатури. На час найбільш широкого використання інтерфейсу командного рядка припадає поява операційної системи UNIX і поява перших восьмирозрядних персональних комп'ютерів з багатоплатформною операційною системою CP / M.

Опис особливостей графічного інтерфейсу

Як і коли з'явився графічний інтерфейс? Його ідея зародилася в середині 70 -х років , коли в дослідницькому центрі Xerox Palo Alto Research Center (PARC) була розроблена концепція візуального інтерфейсу . Передумовою графічного інтерфейсу стало зменшення часу реакції комп'ютера на команду , збільшення обсягу оперативної пам'яті , а також розвиток технічної бази комп'ютерів. Апаратним підставою концепції , звичайно ж , стала поява алфавітно-цифрових дисплеїв на комп'ютерах , причому на цих дисплеях вже були такі ефекти , як "мерехтіння" символів , інверсія кольору (зміна накреслення білих символів на чорному фоні зворотним, тобто чорних символів на білому фоні) , підкреслення символів. Ці ефекти поширилися не на весь екран, а тільки на один або більше символів. Наступним кроком стало створення кольорового дисплея, що дозволяє виводити, разом з цими ефектами , символи в 16 кольорах на тлі з палітрою (тобто колірним набором) з 8 кольорів . Після появи графічних дисплеїв , з можливістю виведення будь-яких графічних зображень у вигляді безлічі точок на екрані різного кольору, фантазії у використанні екрану взагалі не стало меж! Перша система з графічним інтерфейсом 8010 Star Information System групи PARC , таким чином, з'явилася за чотири місяці до виходу в світ першого комп'ютера фірми IBM в 1981 році. Спочатку візуальний інтерфейс використовувався тільки в програмах. Поступово він став переходити і на операційні системи, що використовуються спочатку на комп'ютерах Atari і Apple Macintosh , а потім і на IBM - сумісних комп'ютерах.

З більш раннього часу, і під впливом також і цих концепцій, проходив процес з уніфікації у використанні клавіатури і миші прикладними програмами. Злиття цих двох тенденцій і призвело до створення того користувача інтерфейсу, за допомогою якого, при мінімальних витратах часу і коштів на переучування персоналу, можна працювати з будь-якими програмним продуктом. Опис цього інтерфейсу, спільного для всіх додатків і операційних систем, і присвячена дана частина.

Простий графічний інтерфейс. На першому етапі графічний інтерфейс дуже схожий на технологію командного рядка. Відмінності від технології командного рядка полягали в наступним:

1) При відображенні символів допускалося виділення частини символів кольором, інверсним зображенням, підкресленням і мерехтінням. Завдяки цьому підвищилася виразність зображення.

2) Залежно від конкретної реалізації графічного інтерфейсу курсор може представлятися не тільки мерехтливим прямокутником, а й деякою областю, що охоплює кілька символів і навіть частину екрана. Ця виділена область відрізняється від інших, невиділених частин (зазвичай кольором).

3) Натискання клавіші enter не завжди приводить до виконання команди і переходу до наступного рядка. Реакція на натискання будь-якої клавіші в чому залежить від того, в якій частині екрану знаходився курсор.

4) Крім клавіші enter, на клавіатурі все частіше стали використовуватися "сірі" клавіші управління курсором.

5) Вже в цій редакції графічного інтерфейсу стали використовуватися маніпулятори (типу миші, трекбола і т.п) вони дозволяли швидко виділяти потрібну частину екрану і переміщати курсор.

Підводячи підсумки, можна навести такі відмітні особливості цього інтерфейсу.

1) виділення областей екрана.

2) перевизначення клавіш клавіатури залежно від контексту.

3) використання маніпуляторів і сірих клавіш клавіатури для управління курсором.

4) широке використання кольорових моніторів.

Поява цього типу інтерфейсу збігається з широким розповсюдженням операційної системи ms-dos. Саме вона впровадила цей інтерфейс в маси, завдяки чому 80-ті роки пройшли під знаком вдосконалення цього типу інтерфейсу, поліпшення характеристик відображення символів та інших параметрів монітора.

Типовим прикладом використання цього виду інтерфейсу є файлова оболонка NORTRON COMMANDER (про файлових оболонках дивись нижче) і текстовий редактор MULTI- EDIT. А текстові редактори лексикон, CHIWRITER і текстовий процесор MICROSOFT WORD FOR DOS є прикладом, як цей інтерфейс перевершив сам себе.

WIMP - інтерфейс

Другим етапом у розвитку графічного інтерфейсу став "чистий" інтерфейс WIMP, цей підвид інтерфейсу характеризується наступними особливостями.

1) Вся робота з програмами, файлами і документами відбувається у вікнах - певних окреслених рамкою частинах екрану.

2) Всі програми, файли, документи, пристрої та інші об'єкти представляються у вигляді значків - іконок. При відкритті іконки перетворюються у вікна.

3) Всі дії з об'єктами здійснюються за допомогою меню. Хоча меню з'явилося на першому етапі становлення графічного інтерфейсу, воно не мало в ньому головного значення, а служило лише доповненням до командного рядка. У чистому WIMP - інтерфейсі меню стає основним елементом управління.

4) Широке використання маніпуляторів для вказівки на об'єкти. Маніпулятор перестає бути просто іграшкою - доповненням до клавіатури, а стає основним елементом управління. За допомогою маніпулятора вказують

на будь-яку область екрану , вікна або іконки , виділяють її , а вже потім через меню або з використанням інших технологій здійснюють управління ними .

Слід зазначити , що wimp вимагає для своєї реалізації кольоровий растровий дисплей з високим дозволом і маніпулятор . Також програми, орієнтовані на цей вид інтерфейсу , висувають підвищені вимоги до продуктивності комп'ютера , обсягом його пам'яті , пропускну здатності шини і т.п. однак цей вид інтерфейсу найбільш простий у засвоєнні і інтуїтивно зрозумілий. Тому зараз WIMP - інтерфейс став стандартом де- факто.

Яскравим прикладом програм з графічним інтерфейсом є операційна система MICROSOFT WINDOWS.

Мовленнєва технологія.

З середини 90 -х років , після появи недорогих звукових карт і широкого поширення технологій розпізнавання мови , з'явився так званий "мовна технологія" SILK - інтерфейсу . При цій технології команди подаються голосом шляхом проголошення спеціальних зарезервованих слів - команд. Основними такими командами (за правилами системи "Горинич") є:

"Прокинься" - включення голосового інтерфейсу .

"Відпочивай" - вимкнення мовного інтерфейсу .

"Відкрити" - перехід в режим виклику тієї чи іншої програми . Ім'я програми називається в наступному слові.

"Буду диктувати" - перехід з режиму команд в режим набору тексту голосом.

" Режим команд " - повернення в режим подачі команд голосом .

і деякі інші.

Слова повинні вимовлятися чітко , в одному темпі. Між словами обов'язкове пауза . Через нерозвиненість алгоритму розпізнавання мови такі системи потребує індивідуальної попереднього налаштування на кожного конкретного користувача.

" Мовленнєва " технологія є найпростішою реалізацією SILK - інтерфейсу .

Біометричні технології

Ця технологія виникла наприкінці 90 -х років XX століття і на момент написання книги ще розробляється. Для управління комп'ютером використовується вираз обличчя людини , напрямок його погляду , розмір зіниці та інші ознаки. Для ідентифікації користувача використовується малюнок райдужної оболонки його очей , відбитки пальців і інша унікальна інформація . Зображення зчитуються з цифрової відеокамери , а потім за допомогою спеціальних програм розпізнавання образів з цього зображення виділяються команди. Ця технологія , очевидно , займе своє місце в програмних продуктах і додатках , де важливо точно ідентифікувати користувача комп'ютера .

Семантичний (громадський) інтерфейс

Цей вид інтерфейсу виник наприкінці 70 -х років XX століття , з розвитком штучного інтелекту. Його важко назвати самостійним видом інтерфейсу - він включає в себе і інтерфейс командного рядка , і графічний , і мовної , і мімічний інтерфейс. Основна його відмінна риса - це відсутність команд при спілкуванні з комп'ютером. Запит формується на природній мові , у вигляді пов'язаного тексту і образів. За своєю суттю це важко називати інтерфейсом - це вже моделювання " спілкування " людини з комп'ютером . З середини 90 -х років XX століття публікації , що відносяться до семантичного інтерфейсу , вже не зустрічалися. Схоже , що у зв'язку з важливим військовим значенням цих розробок (наприклад , для автономного ведення сучасного бою машинами - роботами, для " семантичної " криптографії) ці напрямки були засекречені. Інформація, що ці дослідження тривають, іноді з'являється в періодичній пресі (зазвичай в розділах комп'ютерних новин).

Опис особливостей типів інтерфейсів

Інтерфейси користувача бувають двох типів:

1) процедурно-орієнтовані:

- примітивні

- меню
- з вільною навігацією

2) об'єктно-орієнтовані:

- прямого маніпулювання.

Процедурно - орієнтований інтерфейс використовує традиційну модель взаємодії з користувачем , засновану на поняттях " процедура" і " операція". У рамках цієї моделі програмне забезпечення надає користувачеві можливість виконання деяких дій , для яких користувач визначає відповідність даних і наслідком виконання яких є отримання бажаного результату.

Об'єктно-орієнтовані інтерфейси використовують модель взаємодії з користувачем, орієнтовану на маніпулювання об'єктами предметної області. У рамках цієї моделі користувачеві надається можливість безпосередньо взаємодіяти з кожним об'єктом і ініціювати виконання операцій , у процесі яких взаємодіють кілька об'єктів. Завдання користувача формулюється як цілеспрямована зміна деякого об'єкта. Об'єкт розуміється в широкому сенсі слова - модель БД , системи і т.д. Об'єктно-орієнтований інтерфейс припускає , що взаємодія з користувачем здійснюється за допомогою вибору і переміщення піктограм відповідної об'єктно-орієнтованої області. Розрізняють однодокументним (SDI) і багатодокументна (MDI) інтерфейси .

Процедурно-орієнтовані інтерфейси :

- 1) Забезпечують користувачу функції, необхідні для виконання завдань;
- 2) Акцент робиться на завдання;
- 3) Піктограми являють додатки, вікна або операції ;
- 4) Зміст папок і довідників відбивається за допомогою таблиці - списку.

Об'єктно-орієнтовані інтерфейси :

- 1) Забезпечує користувачеві можливість взаємодії з об'єктами ;
- 2) Акцент робиться на вхідні дані та результати ;
- 3) Піктограми являють об'єкти;
- 4) Папки та довідники є візуальними контейнерами об'єктів.

Примітивним називається інтерфейс, який організовує взаємодію з користувачем і використовується в консольному режимі. Єдине відхилення від послідовного процесу, який забезпечується даними, полягає в організації циклу для обробки декількох наборів даних.

Інтерфейс Меню. На відміну від примітивного інтерфейсу, дозволяє користувачеві вибирати операцію зі спеціального списку, що виводиться йому програмою. Ці інтерфейси припускають реалізацію безлічі сценаріїв роботи, послідовність дій у яких визначається користувачами. Деревоподібна організація меню припускає строго обмежену реалізацію. При цьому можливі два варіанти організації меню:

кожне вікно меню займає весь екран

на екрані одночасно присутні кілька різнорівневих меню (Windows).

В умовах обмеженої навігації, незалежно від варіанту реалізації, пошук пункту більш ніж двох рівневого меню виявляється досить складним завданням.

Інтерфейс з вільною навігацією (графічний інтерфейс). Підтримує концепцію інтерактивної взаємодії з ПЗ, візуальну зворотний зв'язок з користувачем і можливість прямого маніпулювання об'єктом (кнопки, індикатори, рядка стану). На відміну від інтерфейсу Меню, інтерфейс з вільною навігацією забезпечує можливість здійснення будь-яких допустимих в конкретному стані операцій, доступ до яких можливий через різні інтерфейсні компоненти ("гарячі" клавіші і т.д.). Інтерфейс з вільною навігацією реалізується з використанням подієвого програмування, що передбачає застосування візуальних засобів розробки (за допомогою повідомлень).

1.3 Аналіз методів та технологій проектування користувацьких інтерфейсів

Розробка користувацького інтерфейсу (КІ) ведеться паралельно розробці програмного продукту в цілому і в основному передують його

впровадження. Процес розробки ергономічного ПІ розбивається на наступні етапи [10] :

1) Постановка завдання.

На цій стадії збираються і аналізуються дані про користувачів , формалізується функціональність, і визначаються об'єктивні критерії успіху проекту .

2) Формалізація контексту використання

Описуються такі властивості аудиторії системи :

Характеристики користувачів: їх досвід роботи з комп'ютером, знання предметної області, мотиви, розмір/важливість груп користувачів , зразки (типові ситуації) використання;

3) Цілі і завдання користувачів;

Завдання проекту: що послужило причиною створення проекту, етапи створення проекту, які результати повинні бути отримані, яка інформація необхідна і коли;

Технологія розробки та платформа , на якій працюватимуть користувачі;

Середовище, в якій створюватиметься, і використовуватися проект (фізична, ринкова, організаційна та культурна).

4) Формалізація об'єктивних критеріїв успіху.

Виділяються об'єктивні критерії оцінки ергономічності інтерфейсу , такі як показники ефективності , продуктивності , задоволеності користувачів (на більш ранніх етапах виділити ці критерії неможливо).

5) Аналіз цілей

Розробнику необхідно чітко усвідомлювати , що користувачам не потрібні інструменти самі по собі , потрібні лише результати їх роботи. Нікому не потрібний текстовий процесор - потрібна можливість із зручністю писати тексти . Нікому не потрібна програма обробки зображень - потрібні вже оброблені зображення. Це означає , що самі по собі функції нікому не потрібні і не важливі. Людям потрібно засіб взагалі , що робить можливим виконувати будь-яку роботу [4] .

6) Формалізація бізнес-ролей користувачів

Функціональність будь-якої системи розділяється на декілька ролей користувачів : різним користувачам потрібні різні блоки функціональності (у системах автоматизації ці ролі називаються бізнес-ролями) . Навігація по системі прямо залежить від цих ролей , оскільки в межах однієї ролі в навігацію не бажане включати функції з чужої ролі. Відповідно, на цьому етапі виділяються основні ролі користувачів з відносяться до цих ролям функціям . Так само , на цьому етапі проводяться співбесіди з кожним з представників певної ролі на предмет виявлення особливостей даної ролі та з'ясування , які додаткові (по відношенню до формальних) можливості слід передбачити .

7) Формалізація сценаріїв дій користувачів

На цьому етапі частково вивчаються , частково розробляються типові сценарії дій користувача: формалізуються дані , необхідні користувачам для виконання роботи , послідовність самої роботи , критерії завершеності цієї роботи, розробляється структура діалогу.

Мета цього етапу - написати словесний опис взаємодії користувача з системою, яка не конкретизуючи , як саме проходить взаємодія , але приділяючи можливо більшу увагу всіх цілях користувачів . Кількість сценаріїв може бути довільним , головне , що вони повинні включати всі типи завдань, що стоять перед системою , і бути скільки- реалістичними. Сценарії дуже зручно розрізняти по іменах беруть участь у них вигаданих персонажів.

8) Огляд інтерфейсу конкуруючих систем

Велика частина аудиторії будь-якої системи володіє навичками використання кількох конкуруючих систем; якщо розробляється інтерфейс повністю несхожий з конкурентами , користувачам доведеться переучуватися. Крім того , конкуруючі системи часто містять ефективні рішення , які корисно перейняти (або, що частіше трапляється , врахувати при проектуванні інтерфейсу).

9) Високорівневе проектування

На цій стадії починається безпосереднє проектування інтерфейсу; попередні етапи присвячені виключно збору даних і постановці завдання .

10) Проектування структури екранів системи

На цьому етапі , ґрунтуючись на сценаріях роботи і ролях користувачів , формується структура екранів системи, тобто визначається кількість екранів , функціональність кожного з них , навігаційні зв'язки між ними , формується структура меню та інших навігаційних елементів .

По суті , на цьому етапі виділяються окремі функціональні блоки. Під функціональними блоками будемо розуміти функцію або групу функцій , пов'язаних з призначенням або області застосування в разі програми і групу функцій / фрагментів інформаційного наповнення у разі сайту.

Існує три основних види зв'язку між блоками. Це логічний зв'язок , зв'язок за поданням користувачів і процесуальна зв'язок [14] .

Логічний зв'язок . Логічний зв'язок визначає взаємодію між фрагментами системи з точки зору розробника. Отримані зв'язку дуже суттєво впливають на навігацію в межах системи (особливо , коли система багатовіконна) . Тому щоб не перевантажувати інтерфейс , варто уникати як надто вже окремих блоків (їх важко знайти) , так і блоків , пов'язаних з великою кількістю інших .

На цьому етапі виділяються об'єктивні критерії оцінки ергономічності інтерфейсу , такі як показники ефективності , продуктивності , задоволеності користувачів (на більш ранніх етапах виділити ці критерії неможливо).

Зв'язок за поданням користувачів . Користувачі мають свою думку про систему , і це думка теж є важливим видом зв'язку . В інформаційних системах , коли необхідно гарантувати , що користувач знайде всю потрібну йому інформацію , необхідно встановлювати зв'язки між блоками , ґрунтуючись не лише на точці зору розробника , а й на уявленнях користувачів . Справа в тому , що найпоширеніший спосіб пошуку , а саме пошук по класифікації ознак , працює тільки в тому випадку , коли користувачі згодні з принципами цієї класифікації. Більшість же понять однозначно класифікувати бути не можуть

через наявність занадто великої кількості значущих ознак . Також проблема полягає в тому , що реальний класифікаційний ознака може відрізнятися від широко поширеного . Наприклад , помідор , який всі вважають овочем , насправді ягода. Не менш важко визнати ягодою кавун . Це означає , що класифікація , прийнята для ботаніка , не буде працювати для всіх інших , причому протилежне не менш справедливо [22].

З цієї ситуації є вихід. Існує простий спосіб класифікації - спосіб карткової сортування. Всі поняття , які потрібно класифікувати , пишуться на картках з розрахунку « одне поняття - одна картка ». Після чого групі користувачів з цільової аудиторії пропонується ці картки розсортувати (при цьому кожен суб'єкт отримує свій набір карток) . Утворені купки з карток потрібно розібрати на складові і звести результати від різних суб'єктів в один спосіб класифікації .

Процесуальний зв'язок. Процесуальна зв'язок описує не цілком логічне , але природне для наявного процесу взаємодія . Встановлення якісної процесуальної зв'язку зазвичай досить важке завдання , оскільки єдиним джерелом інформації є спостереження за користувачем. У той же час встановлення такого зв'язку справа виключно корисне. Навіщо , наприклад , малювати на екрані складну систему навігації , якщо точно відомо , до якого блоку користувач перейде далі? У цьому сенсі часто виправдано нав'язувати користувачу - якийсь процесуальний зв'язок , жертвуючи зручністю, зате виграючи у швидкості навчання (оскільки користувачеві доводиться думати менше). Жорстко задана зв'язок дозволяє також зменшити кількість помилок. Класичним прикладом жорстко заданої процесуальної зв'язку є пристрій майстрів , при якому користувача змушують натискати кнопку Далі.

11) Проектування навігаційної системи

На основі розробленої раніше структури екранів на цьому етапі вибирається найбільш адекватна навігаційна система і розробляється її детальний інтерфейс.

Навігаційна система показує механізм розподілу функцій і завдань між вікнами програми . Навігаційна система визначає , яким чином користувачі зможуть переміщатися як між різними завданнями , так і всередині окремої задачі. Наприклад , чи можна буде залишити частково завершену задачу і почати іншу [18] .

12) Проектування структури довідкової системи

Розробляється довідкова система (до теперішнього етапу вже відома вся інформація, необхідна для вироблення структури довідкової системи , не просто описує інтерфейс , а й допомагає користувачеві вирішувати його завдання) .

13) Низькорівневе проектування

Розробляються інтерфейси конкретних екранів системи (склад , взаємне розташування і підтримують текст інтерфейсних елементів) .

14) Проектування основних екранів

На даному етапі розробляються інтерфейси основних екранів системи .

15) Тестування

На основі об'єктивних критеріїв успіху інтерфейсу і сценаріїв дій користувачів розробляються тестові завдання , які виконуються користувачами з фіксацією всіх значущих характеристик діяльності (таких як продуктивність праці, кількість людських помилок). Після цього виконується підрахунок відповідних показників і порівняння їх із заданими . На підставі отриманих даних інтерфейс або допрацьовується , або вважається розробленим.

16) Проектування другорядних екранів

На даному етапі розробляються інтерфейси другорядних екранів системи . До них відносяться діалогові вікно і всілякі повідомлення.

17) Фінальне тестування

На основі об'єктивних критеріїв успіху інтерфейсу і сценаріїв дій користувачів розробляються залишилися тестові завдання , які виконуються користувачами з фіксацією всіх значущих характеристик їх діяльності . Після

цього виконується підрахунок відповідних показників і порівняння їх із заданими . На підставі отриманих даних інтерфейс або допрацьовується , або вважається розробленим.

1.4 Аналіз існуючих мов проектування

Найбільш відомі мови проектування:

VHDL (англ. VHSIC (Very high speed integrated circuits) Hardware Description Language) - мова опису апаратури високошвидкісних інтегральних схем . Мова проектування VHDL є базовою мовою при розробці апаратури сучасних обчислювальних систем .

Був розроблений в 1983 р. на замовлення Міністерства оборони США з метою формального опису логічних схем для всіх етапів розробки електронних систем , починаючи модулями мікросхем і закінчуючи великими обчислювальними системами .

Verilog - це мова опису апаратури , що використовується для опису та моделювання електронних систем. Ця мова (також відомий як Verilog HDL) дозволяє здійснити проектування , верифікацію і реалізацію (наприклад, у вигляді HBIC) аналогових , цифрових та змішаних електронних систем на різних рівнях абстракції.

SystemC - мова проектування та верифікації моделей системного рівня , реалізований у вигляді C + бібліотеки з відкритим вихідним кодом [1]. Бібліотека включає в себе ядро подієвого моделювання , що дозволяє отримати виконувану модель пристрою.

Мова застосовується для побудови транзакційних і поведінкових моделей , а також для високорівневого синтезу .

ДРАКОН (Доброзичливий Русский Алгоритмічна мова , Який Забезпечує Наочність) - візуальний алгоритмічний мова, створена в рамках космічної програми Буран . Розробка даного мови була розпочата в 1986 р. під керівництвом Володимира Паронджанова . У розробці мови брали участь Російське космічне агентство (НВЦ автоматики та приладобудування ім.

Акад. Н.А. Пілюгіна , м. Москва) та Російська академія наук (Інститут прикладної математики ім. Акад. М.В. Келдиша) .

Одне із завдань , що ставилися перед розробниками , було створення єдиного універсального мови , який повинен був замінити спеціалізовані мови ПРОЛ2 (для розробки бортових комплексних програм Бурана) , ДИПОЛЬ (для створення наземних програм Бурана) і ЛАКС (для моделювання) [3] .

Роботи з розробки мови були закінчені в 1996р. (через 3 роки після закриття програми Буран) , коли була створена автоматизована технологія проектування програмних систем (CASE - технологія) Графіт- Флокс . Ця технологія експлуатується , починаючи з 1996р. в багатьох великих космічних програмах : міжнародний проект Морський старт , розгінний блок космічних апаратів Фрегат , модернізована ракета -носій Протон -М та ін

Правила мови ДРАКОН по створенню діаграм оптимізовані для сприйняття алгоритмів людиною. Таким чином , мова є одним з інструментів посилення інтелекту

Мова ДРАКОН може успішно застосовуватися для специфіцирования протоколів взаємодії (наприклад , клієнт-серверних).

Процедури проектування діалогових режимів

Більшість програмних продуктів, орієнтованих на кінцевого користувача, працюють в діалоговому режимі взаємодії з користувачем, при якому ведеться обмін повідомленнями, що впливають на обробку даних.

У режимі діалогу здійснюються запуск функцій обробки, зміна властивостей об'єктів, проводиться налаштування параметрів видачі інформації на друк і т.п.

Системи, що підтримують діалоговий інтерфейс, поділяються на класи [13]:

- з жорстким сценарієм діалогу (стандартизоване уявлення інформації обміну);
- дескрипторні системи (формат ключових слів повідомлень);

- тезаурусні системи (семантична мережа дескрипторів, що утворюють словник системи), що представляють аналог гіпертекстових систем);

- з мовою ділової прози (подання повідомлень на мові, природному для професійного користування).

При проектуванні КІ необхідно визначити:

- 1) Структуру діалогу.

- 2) Можливий сценарій розвитку діалогу.

- 3) Зміст керуючих повідомлень і даних, якими можуть обмінюватися користувачі і додаток (семантику повідомлень).

- 4) Візуальні атрибути інформації, що відображається (синтаксис повідомлень) [1].

Вибір структури діалогу - це перший з етапів, який повинен бути виконаний при проектуванні КІ

Аналіз організації діалогів в САПР

Діалог - узагальнюючий термін для всіх видів заздалегідь спланованого взаємодії людини з автоматизованою системою . Діалог (діалогове взаємодія) є регламентований обмін інформацією між людиною і обчислювальною машиною , здійснюваний в реальному масштабі часу і спрямований на спільне вирішення конкретного завдання.

Діалог в САПР є методом вирішення слабо формалізованих задач . При цьому поділ функцій між людиною і ЕОМ наступне : людина ставить завдання, а система надає засоби для вирішення підзадач. Виробляється спільне рішення підзадач, остаточне об'єднання результатів, а прийняття проектних рішень залишається за людиною. Діалогової системою (підсистемою) називається система (підсистема), що забезпечує функціонування в режимі діалогу. Діалоговий режим являє собою спосіб організації обчислювального процесу , заснований на діалоговому взаємодії користувачів і обчислювальною машиною (системою) .

Тип діалогу.

Тип діалогу характеризує структурні та лінгвістичні засоби для оформлення діалогових обмінів. Розглядаються такі типи діалогу : абстрактний , конкретний , технічної реалізації. Абстрактний тип діалогу визначає основні властивості діалогового обміну. Для цього типу описуються такі основні компоненти діалогового обміну, як дія і реакція на нього , а також їх взаємозв'язок з наступних позицій : характеру ініціатора (користувач або система) і дії ; можливості вибору (необмежена, обмежена , фіксована); однозначності формулювання завдання. Конкретний тип діалогу пов'язаний з характеристикою повідомлень , що входять в діалоговий обмін . При цьому розглядаються такі складові:

- алфавіт і словник (елементи подання повідомлень); структура подання повідомлень (внутрішній і зовнішній формати , формальна надмірність);

- структура змісту повідомлення (семантичні властивості, форма і інтерпретація).

Внутрішній формат повідомлення визначає синтаксис вхідного повідомлення, а зовнішній формат описує розташування повідомлення на екрані.

Форма діалогу.

Форма діалогу характеризує зміст діалогового розмови , описує повторення однакових типів діалогу та поєднання різних типів діалогу в одному діалоговому обміні і розмові. Основні форми діалогу наступні:

- 1) директивна;
- 2) таблична;
- 3) фразова на обмежено-природній мові ;
- 4) об'єктно-орієнтована .

При директивній формі діалогу застосовуються директиви (команди). Таблична форма діалогу включає конкретні типи діалогу : вибір операції для виконання за меню; заповнення й редагування бланка (шаблону даних); виведення результатів у табличній формі. Фразова форма діалогу використовує обмежено- природна мова . Спілкування ведеться у вільному

форматі , але можлива фіксація окремих фраз, шаблони фраз, програмовані клавіші ключових слів і використання меню. Об'єктно-орієнтована форма діалогу є розвитком директивної і табличної і являє собою аналогію роботи в світі об'єктів предметної області за рахунок використання системи вікон, меню і піктограм.

Структура діалогу .

Структура діалогу - пов'язана сукупність станів діалогу , досяжних при спілкуванні користувача з діалоговою системою . Стан діалогу включає в себе три компоненти :

- використовувану форму діалогу ;
- досягнуту в системі ситуацію, яка визначає сукупність функцій, що надаються користувачеві;
- передісторію діалогу - послідовність діалогових обмінів , що призвели в цей стан , і зміст областей даних.

Основне графічне представлення структури діалогу - діаграма станів (граф діалогу) - орієнтований зважений граф . Кожна вершина графа відповідає певному стану діалогу , а дуга визначає зміну стану . Ваги переходів характеризують зміст діалогу : вхідні і вихідні повідомлення , виконувані програми і програмні умови. Існують і інші способи опису структури діалогу : мережі Петрі, таблиці переходів, автоматні моделі , формальні граматики і фрейми.

Сценарій діалогу.

Сценарій діалогу - детальний опис структури та змісту діалогу , достатній для його реалізації автоматичним шляхом. Явна виділення сценарію дозволяє контролювати допустимі послідовності станів , локалізувати зміни структури діалогу , спростити розробку і налагодження діалогових програм, забезпечує незалежне тестування діалогу. Сценарій діалогу крім опису структури містить у собі інформаційну та операційну моделі діалогу. Формально сценарій визначається наступним чином:

$D = \langle S, R, I, W \rangle$, де S - безліч станів ;

A - безліч операцій ;
 $C = Q \cup F$ - безліч умов ;
 Q - безліч вхідних повідомлень (запитів) ;
 F - безліч програмних умов;
 R - безліч вихідних повідомлень (реакцій) ;
 $G : S \times C \rightarrow S$ структура (граф) діалогу ;
 $S \rightarrow \wedge$
 I : - інформаційна модель діалогу ;
 $S \times C \rightarrow B \rightarrow A$,
 W : - операційна модель діалогу.
 $B \times C \rightarrow A$

Автоматичне ведення діалогу згідно сценарію діалогу називається інтерпретацією діалогу. Основними рисами реалізації сценарію є: формальний апарат , що лежить в основі опису сценарію; спосіб опису сценарію діалогу; підтримувана форма і структура діалогу; структура вузла сценарію діалогу; можливість вкладеного опису сценарію; спосіб зберігання і формат сценарію; наявність засобів збереження та відновлення стану діалогу; наявність протоколу станів діалогу та коштів для повернення до попередніх станам; організація діалогової прикладної програми при використанні сценарію. В якості формального апарату для специфікації сценарію діалогу використовується теорія графів, теорія автоматів, теорія формальних граматик, мережі Петрі і теорія ігор. Опис сценарію виробляється в табличному вигляді або за допомогою спеціальної мови, можливо також графічне представлення сценарію діалогу.

Аналіз процедур і операцій проектування , дослідження технології обробки даних, що лежать в їх основі, показали, що організації діалогу в САПР притаманні такі особливості:

- колективне використання проектувальниками інформаційних та операційних ресурсів;

- суміщення діалогового і пакетного режимів обробки;
- великий обсяг даних , що вводяться і виведених програмами ;
- значна частка текстової оброблюваної інформації;
- багатоаспектне уявлення проектної інформації в системі;
- наявність декількох джерел отримання вихідних даних і декількох споживачів результатів;
- необхідність введення , редагування і відображення інформаційних об'єктів складної структури вимагає забезпечення цілісності і несуперечності , синхронного відстеження змін у зовнішніх і внутрішніх уявленнях одного і того ж об'єкта;
- складна і мінлива структура діалогу ;
- діалог ведеться з різними категоріями користувачів , в основному з користувачами - непрограмістів .

1.5 Аналіз підходів до проектування користувацьких інтерфейсів

Підходи до розуміння користувача інтерфейсу

Людино-машинний або, інакше, користувацький інтерфейс забезпечує взаємодію, тобто обмін діями і реакціями на ці дії, між користувачем і комп'ютером [5].

Дослідження в області людино-машинного взаємодії показують, що будь-який користувацький інтерфейс повинен забезпечувати виконання таких чотирьох функцій:

- 1 . управління комп'ютером шляхом дій користувача: ініціація , переривання , скасування комп'ютерних процесів і т.д.
- 2 . введення даних , здійснюваних користувачем, і відгук системи ;
- 3 . відображення даних , що вводяться користувачем;
- 4 . підтримка користувача в процесі діяльності , що включає в себе зворотний зв'язок і збір інформації про помилкові або випадкових діях користувача.

Добре спроектований користувацький інтерфейс повинен відповідати представленим нижче принципам:

- 1) мати низький поріг входження , тобто сприяти швидкому освоєнню користувача інтерфейсу , формуванню у користувача стійких навичок ;
- 2) забезпечувати введення інформації природним чином , не демонструючи користувачеві хід обчислювального процесу;
- 3) задовольняти робочі потреби користувача , а не загострювати його увагу на процесі обробки даних.

Для отримання ефективного результату розробки користувацького інтерфейсу використовують різні підходи до проектування. Основні з них перераховані нижче [17] :

1 . Підхід , орієнтований на користувача (User - Centered Design) , який характеризується наступними ознаками (стандарт ISO 13407) :

- Активним залученням користувачів в процес проектування і тестування програмного продукту ;
- Чітким розумінням користувача вимог і завдань;
- Оптимальним розподілом функцій між користувачами і технологіями ;
- Інтерактивністю і мультидисциплінарні підходу .

Доведено , що застосування даного підходу при розробці інтерфейсу здатне привести до досягнення високих показників в області юзабіліті (згідно стандарту ISO 9241-11 , юзабіліті - «це ступінь ефективності , продуктивності та задоволеності , з якою продукт може використовуватися певними користувачами для досягнення визначених цілей у певному контексті »), призводить до скорочення витрат на розробку і підвищенню ефективності продукту як щодо бізнесу (додаткова прибуток) , так і в задоволеності користувачів (підвищення лояльності до продукту і розробнику) .

2. Підхід , орієнтований на діяльність (Activity - Centered Design) . Згідно з визначенням Дональда Нормана (Donald Norman) , діяльність включає завдання, які складаються з дій , в свою чергу складених з операцій. Підхід до проектування інтерфейсів , пропонований Норманом , приділяє увагу , перш

за все , розуміння діяльності користувача . Він стверджує , що людина пристосовується до наявних інструментів і що розуміння діяльності , виконуваної людиною за допомогою інструментів , може позитивно позначатися на інтерфейсі цих інструментів.

3. Цілеорієнтований підхід (Goal Centered Design). В основі даного методу лежать кінцеві цілі користувачів, які повинні бути ними досягнуті за допомогою взаємодії з програмним продуктом.

4. Підхід , орієнтований на дані (Data Centered Design) . Проектування інтерфейсу підтримує таку модель взаємодії користувача з системою , при якій первинними є оброблювані дані , а не вимагаються для цього програмні засоби . Іншими словами , при такому підході основна увага користувача концентрується на тих даних , з якими він працює , а не на пошуку і завантаження необхідного додатку . При використанні цього підходу основним програмним об'єктом є документ , який являє собою деяке абстрактне пристрій зберігання даних , що використовуються для виконання завдань користувачів і для їх взаємодії . Документ повинен бути доступний як різним додаткам , використовуваним для його обробки , так і всім взаємодіючим користувачам .

5. Ітеративний підхід (Agile) - метод послідовних наближень. Суть ітеративного підходу полягає у створенні спочатку самого найпростішого прототипу з метою формування у замовника і самого проектувальника спільного бачення проекту. Потім поступово прототип допрацьовується і деталізується. Такий підхід зображено на рисунку 1.5

При розробці інтерфейсу доцільно гнучко користуватися існуючими підходами, враховуючи при виборі методів призначення розроблюваного продукту, цільову аудиторію, час і бюджет розробки.



Рисунок 1.5 – Модель ітеративного підходу

2 ФОРМУВАННЯ ВИМОГ ТА РОЗРОБКА КОНЦЕПЦІЇ ПРОЕКТУВАННЯ ЄДИНОЇ МОДЕЛІ ІНТЕРФЕЙСУ

2.1 Аналіз поняття користувацького інтерфейсу як системи

Користувальницький інтерфейс – це сукупність методологічних, інформаційних, програмних, лінгвістичних засобів, які забезпечують успішне управління функціонуванням програмної системи.

Інформаційна система - це взаємопов'язана сукупність інформаційних, технічних, програмних, математичних, організаційних, правових, ергономічних, лінгвістичних, технологічних та інших засобів, а також персоналу, призначена для збору, обробки, зберігання та видачі економічної інформації та прийняття управлінських рішень .

Система - це сукупність взаємопов'язаних елементів , підпорядкованих єдиній меті.

Ознаки системи

- Елементи системи взаємопов'язані і взаємодіють в рамках системи .
- Кожен елемент системи може в свою чергу розглядатися як самостійна система, але він виконує лише частину функцій системи .
- Система як ціле виконує певну функцію, яка не може бути зведена до функцій окремо взятого елемента .
- Підсистеми можуть взаємодіяти як між собою, так і з зовнішнім середовищем і змінювати при цьому своє утримання або внутрішню будову .
- Система управління реалізує функцію управління і складається з таких підсистем, як прогнозування, планування, облік, аналіз, контроль і регулювання

Виходячи з визначень Інтерфейсу і ІС можна сказати, що те, що то і є це сукупність взаємопов'язаних елементів , підпорядкованих єдиній меті .

Відповідно ознаки системи подібні ознаками інтерфейсу. Звідси можна зробити висновок, що інтерфейс і є система, що дозволяє нам розглядати моделі проектування ІС для проектування інтерфейсів.

2.2 Аналіз особливостей проектування інформаційних систем

Життєвий цикл - період створення і використання інформаційної системи, який охоплює її різноманітні стани, починаючи з моменту виникнення необхідності в даній інформаційній системі і закінчуючи моментом її повного виходу з експлуатації.

Системне проектування (розробка) і використання інформаційної системи має пройти наступний життєвий цикл інформаційної системи:

- передпроектний аналіз (досвід створення інших аналогічних систем, відмінності і особливості та інше), аналіз зовнішніх проявів системи;
- внутрішньосистемний аналіз, внутрішній аналіз (аналіз підсистем системи);
- системне представлення системи (опис системної цілі, системних відношень і зв'язків із зовнішнім середовищем, інших системам і системних ресурсів - матеріальних, енергетичних, інформаційних, організаційних, людських, часових і просторових);
- визначення критеріїв адекватності, ефективності і стійкості (надійності);
- функціональний опис підсистем системи (опис моделей, алгоритмів функціонування підсистем);
- макетний опис системи, оцінка взаємодії підсистем системи, при цьому можливе використання «макетів» критеріїв адекватності, стійкості, ефективності;
- "збір" і тестування системи – реалізація повноцінних функціональних підсистем і критеріїв, оцінка моделі по сформульованим критеріям;
- функціонування системи;
- визначення цілей подальшого розвитку системи і її додатків;
- уточнення, модифікація, розширення можливостей системи в режимі її функціонування (з ціллю її еволюціонування).

Життєвий цикл носить ітеративний характер: реалізовані етапи життєвого циклу, починаючи з самих ранніх, циклічно повторюються

відповідно до нових вимог і змін зовнішніх умов. На кожному етапі життєвого циклу формується набір документів і технічних рішень, які є початковими для наступних рішень.

Найбільш відомими моделями життєвого циклу є каскадна модель, ітераційна модель, спіральна модель та модель Microsoft Solutions Framework.

1. Каскадна модель (рисунок 2.1) передбачає розробку інформаційної системи протягом декількох послідовних етапів, причому перехід на наступний етап відбувається лише тоді, коли робота на попередньому етапі була повністю завершена. Дана модель задовольняла потреби проектування невеликих і нескладних інформаційних систем в 70-х роках. Проте через звичку та недолік знань каскадна модель широко використовується і зараз.

Переваги каскадного підходу виявляються при побудові інформаційних систем, для яких на самому початку розробки можна сформулювати всі вимоги. В цьому випадку для кожного етапу можна визначити точні терміни завершення і вартість робіт.

В разі використання каскадної моделі для реалізації складних проектів виникає велика кількість труднощів: неможливість виправити помилки на попередніх етапах; неточності в документації і звітності; значна тривалість проектування, оскільки сформульовані вимоги до проекту можна змінити лише в кінці розробки.

Розподілення ресурсів при каскадному підході відбувається відразу на першому етапі, після чого відбувається перехід на наступний етап проектування. Це може призвести к тому, що виникне проблема неправильного розподілення ресурсів, яка в свою чергу виявиться на наступних етапах, а повернутися назад проектувальник вже не матиме змогу. І в результаті, початкове розподілення ресурсів буде некоректним та хибним, буде потребувати змін або детальнішого аналізу, а кінцевий результат характеризуватиметься витратами, матеріальними збитками та неефективністю роботи всієї інформаційної системи. Треба також зазначити,

що не всі ресурси можна буде врахувати відразу, деякі можуть з'явитися через проміжок часу, а при каскадному підході стає неможливим їх включення в систему та подальший аналіз і розподіл.

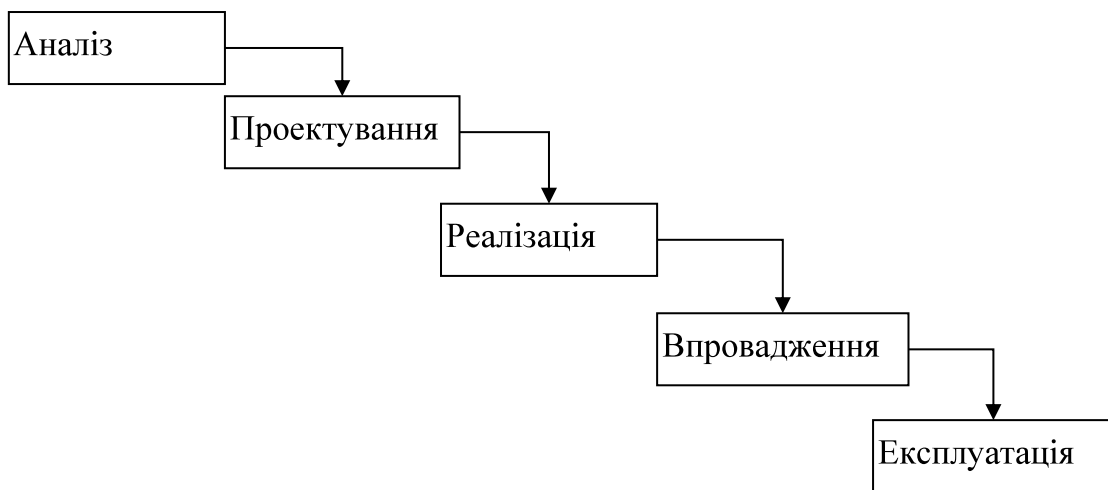


Рисунок 2.1 - Каскадна модель життєвого циклу

2. Ітераційна модель, зображена на рисунку 2.2, з'явилася в 70-80 роки, і є вдосконаленою каскадною моделлю. Для усунення такого недоліку своєї попередниці, як відсутність можливості виправлення помилок на попередніх етапах, в ній реалізується зворотний зв'язок. Проте, використання цієї моделі також можливо лише в разі нескладних проектів інформаційних систем.

Ітераційне повернення на попередні етапи дозволяє виправити невеликі недоліки в проектуванні. Інакше може виникнути потреба в перегляді раніше сформульованих вимог, а це може призвести до перепроєктування всієї інформаційної системи. При використанні цієї моделі для реалізації складних проектів виникає величезна кількість ітерацій, що призводить до заплутаності в документації і збільшує тривалість життєвого циклу.

Розподілення ресурсів при ітераційному підході відбувається аналогічно каскадному, з однією лише відмінністю – присутність зворотного зв'язку дозволяє повернутися на попередній етап, якщо на наступному виявлені неточності чи помилки розподілення. Але ця модель не вирішує питання розподілення ресурсів, які можуть з'явитися у процесі розробки, при цьому додається суттєва проблема – велика кількість можливих переходів та

повернень. Тобто, цей підхід характеризується великою кількістю дій при переході з етапу на етап. І при цьому ж, кількість цих дій не покращую та не вирішує проблему економічного та вигідного розподілення ресурсів при проектуванні інформаційної системи.

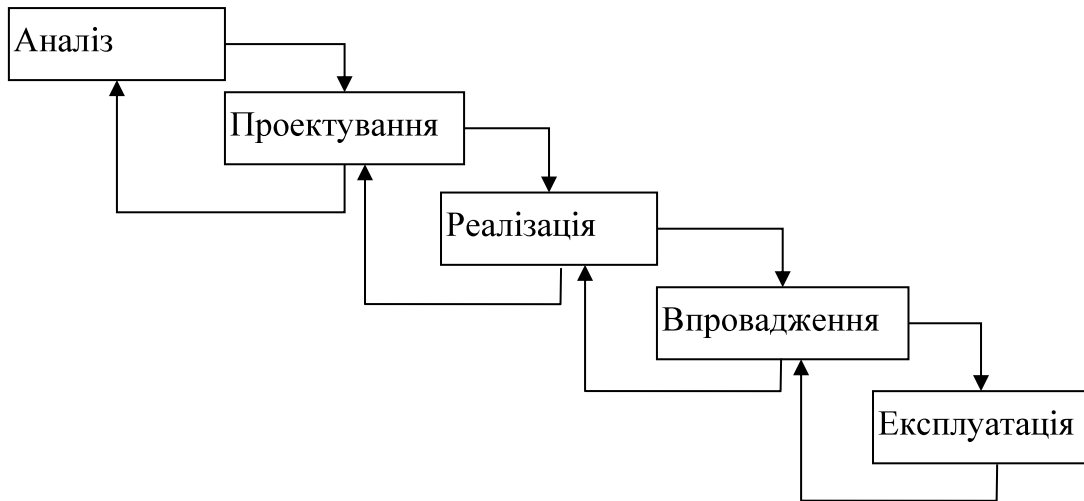


Рисунок 2.2 - Ітераційна модель життєвого циклу

3. Спіральна модель передбачає циклічне виконання всіх етапів каскадної моделі, внаслідок чого рівень реалізації технічних рішень перевіряється за допомогою прототипів. Кожен виток спіралі відповідає створенню фрагмента інформаційної системи, на ньому уточнюються цілі і характеристики проекту, визначається його якість і плануються роботи.

У основі спірального життєвого циклу (рисунок 2.3) лежить застосування RAD-технології. Інформаційна система розробляється шляхом розширення програмних прототипів, повторюючи шлях від деталізації вимог до деталізації програмного коду на кожному етапі. Даний життєвий цикл не має недоліків двох попередніх: модель дозволяє виконувати повернення до попередніх етапів, а невелике число ітерацій зменшує вірогідність виникнення помилок та невідповідностей. Вище перелічені позитивні сторони моделі говорять про те, що вона повністю відповідає особливостям сучасних проектів інформаційних систем.

Проте вдосконалення спіральної моделі ще не завершено, оскільки вона також має ряд недоліків: через високу адаптивність вона непридатна у

випадках контрактів з фіксованою ціною; великий обсяг проектної документації; відсутність чітких критеріїв виконання проекту.

Розподілення ресурсів у спіральній моделі відбувається спіралью, тобто на кожному з витків (етапів) уточнюються умови, економічність розподілення та перевіряється наявність нових ресурсів, відповідність та адекватність їх розподілу. За допомогою такої особливості є можливість корекції та перерозподілу ресурсів, повернення на попередні етапи, що дає змогу більш економічно та вигідно підходити до задачі розподілу та оптимізації. При цьому кількість ітерацій значно менша, це впливає і на швидкодію проектування.

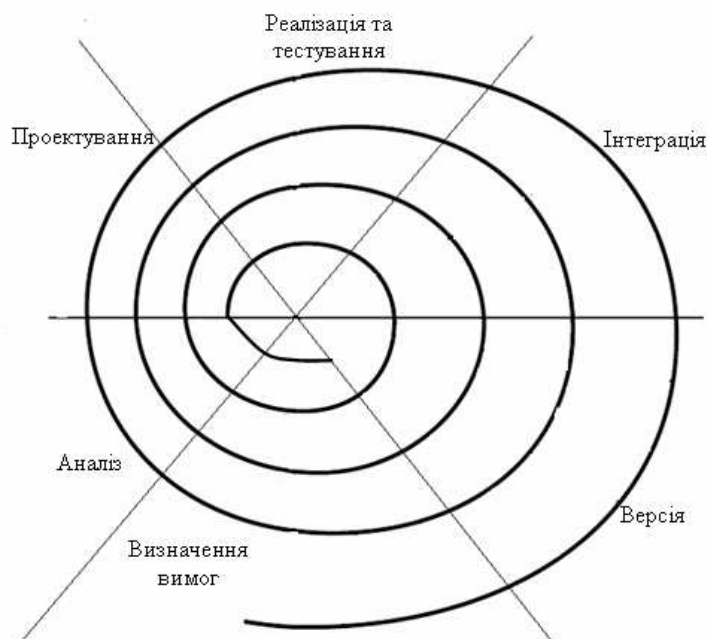


Рисунок 2.3 - Спіральна модель життєвого циклу

4. Модель Microsoft Solutions Framework (MSF) була розроблена компанією Microsoft для своїх власних цілей, але отримала доволі велику популярність і серед інших розробників. Модель, яка зображена на рисунку 2.5, увібрала в себе краще з трьох моделей, що описано вище, - спіральної, ітераційної та каскадної.

Проект реалізується поетапно, з наявністю відповідних контрольних точок, а сама послідовність етапів може повторюватися по спіралі. Складається з чотирьох етапів: аналіз, планування, розробка, стабілізація, розгортання. Кожен етап закінчується досягненням певного результату,

наприклад, після аналізу проекту йде схвалення продуманої концепції. У результаті створюється певний документ, в якому записується результат виконання даного етапу, для того, щоб кожен з розробників чітко розумів свою роль в проекті і задачі, які йому потрібно вирішити. Модель ітераційна і при проходженні всіх етапів, проект можна допрацювати з урахуванням попередньої ітерації. Але, оскільки, закінчення кожного етапу чітко вказане, то відсутнє нескінченне повторення одного й того ж процесу, як у спіральній моделі.

В цій моделі розподілення ресурсів відбувається найефективніше через те, що розробка йде по спіралі, враховує зміни та дає можливість перерозподілу в залежності від потреб та вимог, що виникають у процесі розробки інформаційної системи. У моделі MSF чітко визначено, хто яку роботу виконує, тому процес займає менше часу на узгодження виконавців на кожному з етапів. Треба зазначити, що ця модель має ітераційний характер, тому при розробці є змогу повернутися на потрібний етап за виконати зміни та корекцію попереднього помилкового розподілу. При цьому відсутня проблема постійного повтору ітерацій, так як кожен етап чітко окреслений по часу та має фіксований початок та кінець.

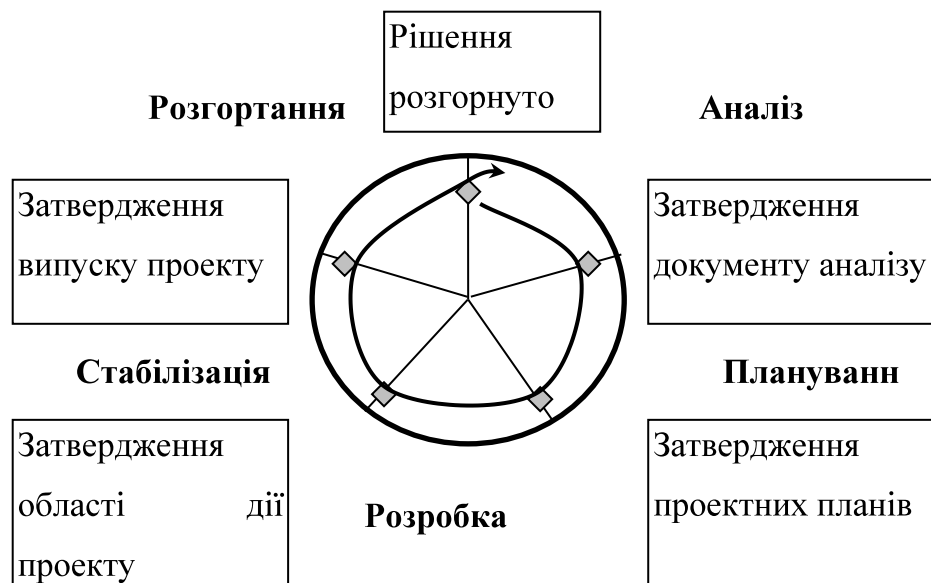


Рисунок 2.5 - Модель Microsoft Solutions Framework

Немає необхідності при проектуванні прив'язуватися до конкретних моделей життєвого циклу і процесів. Реалізація проекту передбачає вирішення ряду завдань, специфіка яких і визначає потрібні процеси проектування і послідовність їх вирішення. Отже, при проектуванні ІС на промислових підприємствах для підвищення ефективності процесу розробки слід використовувати різні життєві цикли і процеси. Каскадну і ітераційну модель життєвого циклу слід застосовувати в канонічній технології проектування, інші моделі - в індустріальній автоматизованій і типовій технологіях.

Враховуючи усі переваги та недоліки моделей, для проектування ІС на промисловому підприємстві доцільним є використання спіральної моделі життєвого циклу. Треба зазначити, що і модель MSF є придатною для виконання подібного завдання, проте надамо перевагу саме спіральній, через ряд причин. По-перше, саме у цій моделі першочергово робиться упор на предпроектну стадію, саме на той етап, де відбувається аналіз вимог та безпосереднє розподілення ресурсів. По-друге, легко можна вирішити основну проблему цієї моделі - визначення моменту переходу на наступний етап. Для необхідно ввести тимчасові обмеження на кожного з етапів життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не уся запланована робота закінчена. План складається на основі статистичних даних, отриманих в попередніх проектах, і особистого досвіду розробників. По-третє, хоча у моделі MSF є доволі багато переваг, її реалізація складніша, займає більш часу, потребує більш глибокого аналізу та більшої кількості людей, вона представляє собою загальну методологію, тому для її використання треба проводити детальний аналіз актуальності, вибір методу. Все це займає час, ресурси, можливо і фінанси, а результат не буде настільки відмінним від результату спіральної моделі, щоб вибір саме цієї моделі визначити доцільнішим.

Від якості робіт, що проектуються, залежить ефективність функціонування системи, тому кожна стадія розділяється на ряд етапів і передбачає складання документації, що відображає результати робіт.

Методи проектування інформаційних систем

Методологія побудови інформаційних систем складається з трьох компонентів:

1. Набір моделей (типів моделей) для опису вимог до інформаційної системи, проектних та програмних рішень. Кожна модель звичайно містить як визначення конструкцій (нотацію), так і правила їх використання (синтаксис).
2. Метод застосування набору моделей для побудови інформаційної системи. Метод звичайно використовує фіксований набір моделей і визначає послідовність їх побудови для опису різних аспектів системи, що створюється.
3. Процес організації проектних робіт, який включає різноманітні технології – планування, управління проектом тощо.

Кожен проект можна розглядати як реалізацію конкретного процесу застосування метода. В залежності від обмежень по строкам і вартості, в конкретну реалізацію можуть бути включені лише окремі частини повного методу і процесу. Крім послідовності етапів методи проектування відрізняються об'єктами дослідження і синтезу.

Класифікація методів проектування показана на рисунку 2.6.

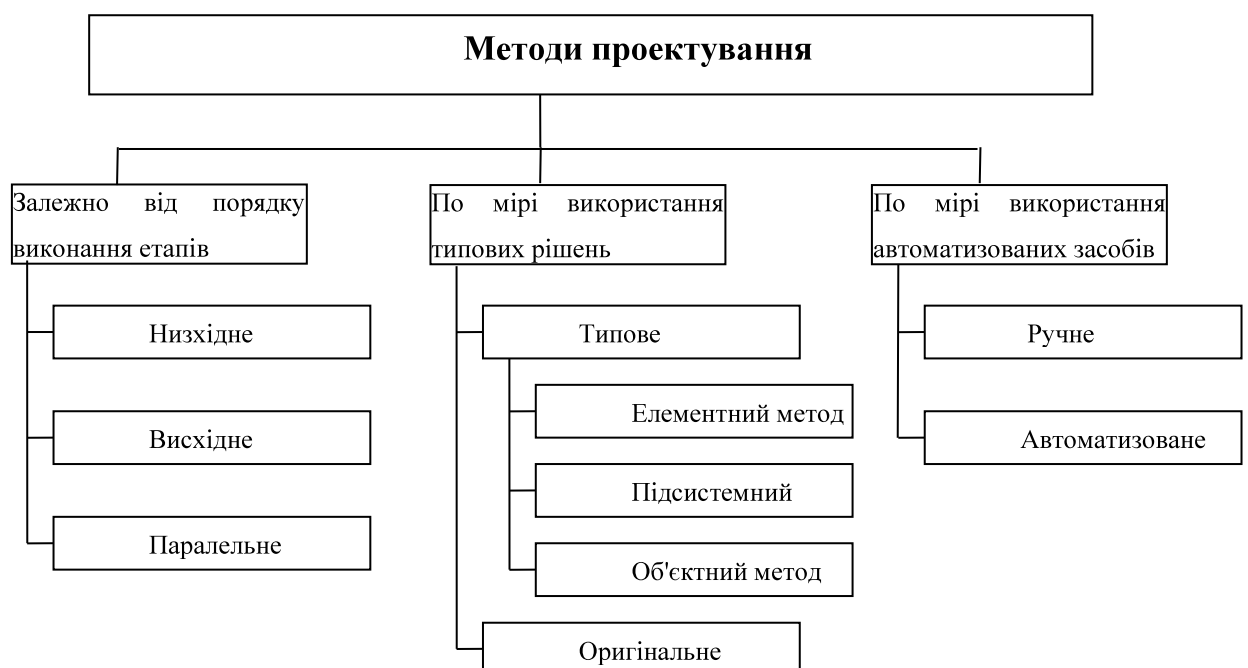


Рисунок 2.6 - Класифікація методів проектування

По мірі використання автоматизованих засобів методи проектування розділяються на методи ручного і автоматизованого проектування.

Методи ручного проектування застосовуються для проектування компонентів інформаційної системи без використання спеціальних інструментальних програмних засобів, а програмування здійснюється на алгоритмічних мовах.

Методи автоматизованого проектування передбачають використання спеціалізованих програмних засобів.

Очевидно, що при ручному проектуванні можливі високі тимчасові витрати й недостатня точність проектних рішень. Однак дані методи завжди використовуються разом при проектуванні, тому що кожен етап проектування має свою власну задачу, яку не завжди можна вирішити за допомогою методу автоматизованого проектування. Наприклад, визначення вимог до інформаційної системи, що створюється, оформлення техніко-економічного обґрунтування неможливо здійснити за допомогою автоматизованих засобів. В той же час метод ручного проектування для побудови логічної і фізичної структури інформаційної системи істотно ускладнює задачу проектувальника та негативно впливає на подальші етапи проектування, які реалізують свої задачі на основі цих даних.

Залежно від порядку виконання етапів проектування розрізняють методи низхідного (зверху-вниз), висхідного (знизу-вгору) і паралельного проектування.

При *висхідному проектуванні* задачі нижчих ієрархічних рівнів вирішуються раніше, ніж задачі вищих ієрархічних рівнів. Даний метод доцільно застосовувати при проектуванні інформаційних систем з невеликою кількістю функцій. При проектуванні інформаційних систем на підприємствах із складною ієрархією, метод висхідного проектування може привести до такого результату, як «клаптева автоматизація». При такому підході основна увага спрямована на автоматизацію функцій найнижчого

рівня організації, які, по-перше, згодом достатньо проблематично буде пов'язати з функціями більш високого рівня, а по-друге буде потрібно постійне доопрацювання деяких компонентів інформаційної системи. Елементи інформаційної системи визначені, проте відомості про властивості і структуру інформаційної системи носять гаданий характер.

Застосування *низхідного методу проектування* інформаційних систем дозволяє раніше вирішувати задачі, які пов'язані з вищими ієрархічними рівнями. Рівні управління на підприємстві взаємозв'язані, оброблена інформація на оперативному рівні поступає на тактичний, а потім і на стратегічний рівні управління. Також складність задач на стратегічному рівні значно більше, ніж на нижчих, тактичному і оперативному. Низхідний метод проектування надає можливість створювати інформаційну систему, враховуючи стратегію організації. Це в свою чергу дозволяє зв'язати задачі нижчого рівня з головною задачею підприємства. Сучасні виробники інформаційних систем при проектуванні використовують метод низхідного проектування.

Проте, як в низхідному, так і у висхідному проектуванні, присутня невизначеність, яка може привести до значних помилок, збільшити терміни і собівартість проектування. При низхідному проектуванні слабо враховуються потреби нижніх рівнів, а при висхідному – верхніх рівнів. Дану проблему здатний вирішити *метод паралельного проектування*.

Використання цього методу дозволяє проводити паралельно проектування інформаційної системи на трьох рівнях: стратегічному, тактичному і оперативному. Учасники робіт мають можливість обмінюватися інформацією на всіх рівнях, внаслідок чого знижується ймовірність появи якої-небудь ризикової події, забезпечуються вимоги, як стратегічних цілей організації, так і тактичних, і оперативних. Перевагою паралельного проектування є те, що на конкретному етапі проектування враховуються критерії з інших етапів.

По мірі використання типових проектних рішень розрізняють наступні методи проектування: оригінального і типового проектування.

Методи оригінального проектування використовуються тоді, коли проектні рішення розробляються "з нуля" відповідно до вимог до інформаційної системи.

Оригінальне проектування інформаційної системи на промисловому підприємстві здійснюється у разі, коли на ринку програмних продуктів відсутні засоби, які здатні забезпечити потреби організацій, або існуючі програмні рішення на підприємстві не мають в своєму складі компонента, який здатний забезпечити інформаційну підтримку необхідної функції управління.

Методи типового проектування застосовуються для створення інформаційної системи на основі готових типових проектних рішень (програмних модулів).

Дані методи застосовуються, коли підприємство ставить перед собою наступні задачі:

- створити інтегровану інформаційну систему з існуючих програмних комплексів;
- створити інформаційну систему з існуючих проектних рішень, що спрямована на реалізацію конкретної функції на певному рівні управління.

Методи типового і оригінального проектування часто перетинаються між собою, якщо підприємство створює інтегровану інформаційну систему (типове проектування) і паралельно доопрацьовує деякі модулі програмних продуктів (оригінальне проектування) відповідно до своїх задач.

Проектування інформаційної системи з використанням методів типового проектування декомпозується на безліч складових елементів. На кожному рівні декомпозиції застосовується певний метод типового проектування: елементний, підсистемний і об'єктний.

При елементному методі як типовий елемент системи використовується типове рішення по завданню. Застосування цього методу доцільне тоді, коли необхідно створити інформаційну систему, яка вирішує невелику кількість задач, або створити один модуль. У інших випадках можливі великі затрати часу на з'єднання різнорідних елементів системи.

Типовим елементом в *підсистемному методі* є окремі підсистеми, розроблені з урахуванням функціональної повноти і мінімізації зовнішніх зв'язків. Даний метод використовується у разі потреби проектування інформаційної системи, що забезпечує окрему функцію управління. Наприклад, фінансове управління, матеріально-технічне постачання, бухгалтерський облік. Проблема при використанні даного методу може виникнути в області інтеграції інформаційної системи, що розробляється, із програмними рішеннями, що існують на підприємстві. До готових типових рішень можна віднести наступні системи: «1С предприятие», Project Planner, Oracle Financial Services Applications.

При *об'єктному методі* у якості типового елемента використовується повний набір функціональних підсистем для об'єктів управління певної галузі. Даний метод слід застосовувати при необхідності автоматизації комплексу взаємозв'язаних функцій управління. Наприклад, для комплексної автоматизації служб підприємства по управлінню виробництвом, ресурсами, фінансами. Однак при використанні даного методу також можуть виникнути труднощі, тому що необхідно враховувати специфіку предметної області при проектуванні. Застосування готових типових рішень (Парус, Галактика, ІТ-предприятие) у більшості випадків потребує зміни структури підприємства, послідовності виробничих процесів, функціональних обов'язків співробітників.

Ознаки вибраних методів дозволяють визначити характер необхідної технології проектування для конкретної інформаційної системи. Розрізняють канонічну та індустріальну технології проектування.

Канонічне проектування ґрунтується на застосуванні виконавцями ручних методів проектування, без використання інструментальних засобів. Дана технологія реалізується в проектуванні невеликих локальних інформаційних систем.

Індустріальна технологія проектування ґрунтується на використанні вже готових засобів розробки інформаційних систем.

Таблиця 2.1 містить порівняння існуючих методів, визначення недоліків та переваг кожного з них.

Таблиця 2.1. Порівняння існуючих типових методів проектування інформаційних систем

Методи		Елемент системи	Приклади ІС	Переваги	Недоліки
Типове використання рішень	Елементарний	Типове рішення по завданню		Дозволяє якісно та швидко створити один модуль, що вирішує нескладну задачу	При складній ІС треба багато часу для з'єднання різномірних елементів системи
	Підсистемний	Окремі підсистеми, розроблені з урахуванням повноти і мінімізації зовнішніх зв'язків	«ІС підприємств», Project Planner, Oracle Financial Services Applications	Дозволяє спроектувати окрему функцію управління	Проблема інтеграції з програмними рішеннями, що вже існують на підприємстві
	Об'єктний	Повний набір функціональних підсистем для об'єктів управління галузі	Парус, Галактика, ІТ-предприємство	Використання для автоматизації комплексу взаємозв'язаних функцій управління	Необхідно враховувати специфіку предметної області, потребує зміни структури підприємства

Існують і інші підходи до визначення методів проектування інформаційних систем. Наведемо ці підходи та класифікацію методів.

Підхід, запропонований Клещевим і Романовим, в якому методи умовно згруповані як: структурно-функціональні, віртуальні (універсальні), функціонально-технологічні, об'єктні. Очевидно, що таким чином виокремлені групи сильно взаємопов'язані і суттєво перетинаються по змісту. Однак структурно-функціональний аналіз є методологічною основою практично у всіх методах проектування інформаційних систем.

Віртуальні (універсальні) методи дають змогу описати абстрактний набір структур розрахункових процесів і каналів взаємодії, які відображають оригінальну структуру об'єкта, що автоматизується. Опис предметів на віртуальному рівні визначає внутрішню структуру систем, але не враховує програмних і інформаційних взаємозв'язків між процесами, задачами, що їх реалізують і об'єктами автоматизації.

Віртуальний рівень проектування отримав широке розповсюдження для інформаційних систем, орієнтованих на рішення простих задач або на операції обробки типових даних. Для інформаційних систем, що реалізують постійно змінюючи функції організаційного управління, як сама сукупність задач, так і, особливо, їх параметри у великій мірі характеризуються невизначеністю не лише на початкових стадіях проектування, але і в процесі експлуатації. В період усього періоду створення і розвитку таких інформаційних систем, як організаційна структура самого об'єкта автоматизації, так і функціонально-технологічні задачі, що вирішується, знаходяться в процесі розвитку.

Структурно-функціональні методи проектування і розробки полягають у декомпозиції структури інформаційної системи на окремі підсистеми і модулі в цілях аналізу їх технічного, системного і прикладного складу і наступного синтезу структури системи для реалізації заданих функцій, що потребуються.

Ці методи широко використовують вже відпрацьовані алгоритми і промодельовані задачі і гарно себе зарекомендували при проектуванні універсальних і спеціалізованих систем, окремих машин і пристроїв. Якщо зміни і нові функції незначно відображуються на структурі цих пристроїв,

машин і систем, то їх адаптація до нових умов функціонування не викликає надто великих фінансових втрат.

Функціонально-технологічні методи проектування інформаційних систем забезпечують аналіз неперервно змінного спектра організаційних, управлінських функцій і технологій.

Існуючою перевагою функціонально-технологічних методів є формування і застосування динамічних типових конструкторських рішень при створенні ІС: типових прикладних рішень, типових технологічних підсистем і типових операцій по обробці інформації, а також пов'язаних з ними програмно-технологічних рішень.

Застосування типових конструкторських рішень забезпечує високу динаміку проробки технічних рішень для всіх видів процесів і взаємозв'язок їх по трьом рівням проектування: концептуальному, на якому здійснюється аналіз і синтез системних архітектур; рівні функціональних об'єктів, де формуються рішення по створенню і вдосконаленню інформаційної системи; рівні елементів ІС – створення програмно-технічної інфраструктури об'єктів, що функціонують.

Об'єктні методи забезпечують єдину концепцію для предпроектного аналізу, проектування, програмування функціональних задач.

Концептуальне єдинство об'єктних методів забезпечується за рахунок використання базових елементів: об'єктів, класів і властивостей наслідування. Абстрактний об'єкт складається з трьох частин: імені об'єкта, стану (змінних стану) і методів (операцій).

Об'єктний аналіз представляє собою метод аналізу предметної області, заснований на переважному виявленні об'єктів цієї предметної області і встановленні взаємних зв'язків між ними.

Об'єктне проектування представляє собою метод проектування, що поєднує в собі процес об'єктної декомпозиції, прийоми представлення моделей проектованої інформаційної системи як на логічному та фізичному рівнях, так і в статичній і динамічній формах. В цілому концепція об'єктних

методів спирається на декомпозицію об'єкта, декларований опис об'єктів класами, а також покрокове програмування з використанням механізму наслідування.

Типові технологічні підсистеми, створені за допомогою об'єктних методів, є розподіленими системами розрахунків, так як пам'ять розподілена між об'єктами і кожна операція виконується лише в пам'яті даного об'єкта.

Перевагами об'єктних методів є їх орієнтованість на задачі аналізу, проектування і програмування інформаційних систем з урахування динаміки інформаційних вимог користувачів, реалізація яких традиційними методами вкрай трудомістка. На рисунку 2.7 запропоновано класифікацію методів за цим підходом і визначені сфери застосування.



Рисунок 2.7 - Класифікація методів за підходом, запропонованим
Клещевим і Романовим

Підхід, запропонований Прозоровим, класифікація методів в якому здійснюється на підставі поняття життєвого циклу інформаційної системи, підходів до суттєвої та об'єктної декомпозиції і локалізації інформаційної системи. Від представлений наступними методами: метод ОМТ (Object Modeling Technique), методи сімейства (Icam DEFinition) і метод COMET (Concurrent Object Modeling and Architectural Design Method).

У *методі ОМТ* інформаційна система, що проектується представляється у вигляді трьох взаємопов'язаних моделей:

- об'єктної моделі, яка визначає статичні аспекти системи, в основному пов'язані з даними (статична структура);
- динамічної моделі, яка описує роботу окремих частин системи (динамічна структура);
- функціональної моделі, яка описує взаємодію окремих частин системи, яке виникає у процесі її роботи (взаємодія описується як у ростині даних, так і у ростині управління).

Ці три моделі дозволяють отримати три ортогональні представлення системи в одній нотації. Сукупність моделей системи може бути проінтерпретована на комп'ютері, що дозволяє продемонструвати характер роботи з майбутньою системою і суттєво спрощує узгодження попереднього проекту системи.

Цей метод обмежується двома фазами життєвого циклу ІС: аналізом вимог спільно з побудовою об'єктної моделі і реалізацією програмної системи.

Сімейство методів проектування нотацій IDEF базується на методології ICAM (Intergated Computer Aided Manufacturing) і складається з наступних представників:

- IDEF0 – метод і нотація опису бізнес-процесів;

- IDEF1 – метод і нотація опису взаємозв'язків між інформаційними потоками;
- IDEF1X – метод і нотації розробки реляційних баз даних;
- IDEF3 – метод і нотація опису технологічних процесів;
- IDEF5 – метод і нотація опису онтологічних досліджень.

Метод COMET заснований на методиці COMADM (Concurrent Object Modeling and Architectural Design Method), маючи з ним незначні відмінності, обумовлені використанням у методі COMET нотації UML. Вказані два методи розроблені професором кафедри програмо техніки університету Джорджа Мейсона Хасаном Гома, авторитетним ідеологом у області проектування розподілених додатків і додатків реального часу.

У загальному вигляді метод COMET представляє собою метод об'єктного моделювання і архітектурного проектування паралельних систем, в основі якого лежить створення об'єктно-орієнтованого програмного забезпечення. Життєвий цикл даного методу характеризується значним числом ітерацій.

Основними етапами метода COMET є:

- Етап моделювання функціональних потреб – основна увага приділяється збору і класифікації функцій до системи, в той час, як сама система розглядається як «чорна скринька»;
- Етап аналітичного моделювання виконується в термінах сутностей моделі. Основна увага на даному етапі приділяється предметній області, при цьому структуру сутностей моделі описується за допомогою статичного представлення моделі, а характер поведінки – за допомогою динамічного представлення моделі. Статистичне представлення моделі виконується у термінах класів (об'єктів) предметної області відношень між ними, в той час як динамічне представлення моделі виконується в термінах взаємодії між об'єктами;

- Етап архітектурного (імітаційного) моделювання виконується у термінах структури імітаційної моделі (класи і відношення між ними. На даному етапі основна увага приділяється об'єктній і часовій декомпозиції сутностей моделей, формуються базові критерії розбиття системи на основні частини (підсистеми, модулі та ін.);
- Етап програмного моделювання виконується в термінах програмної моделі (атрибути і операції класів). На даному етапі особлива увага приділяється програмній реалізації імітаційної моделі.

На рисунку 2.8 приведена умовна схема класифікації методів, яка була запропонована Прозоровим.

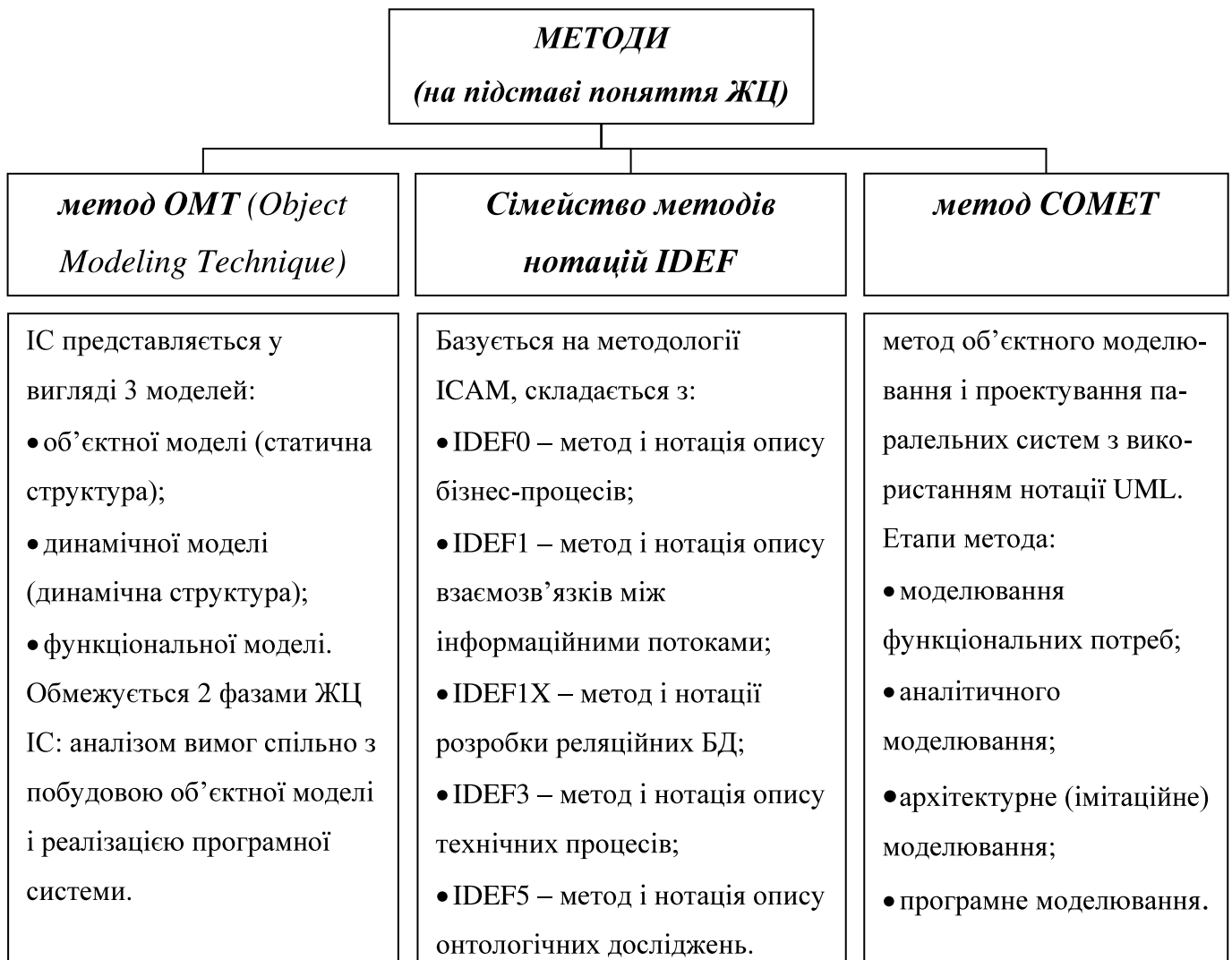


Рисунок 2.8 - Класифікація методів за підходом, запропонованим Прозоровим

2.3 Аналіз методологій проектування інформаційних систем

Методологія RAD. Методологія RAD отримала в наш час широке застосування у сфері створення автоматизованих інформаційних систем. Вона охоплює всі етапи життєвого циклу інформаційних систем.

Основні принципи RAD полягають в наступному:

- використовується спіральна модель розробки;
- повне завершення робіт на кожному етапі життєвого циклу не є обов'язковим;
- застосування CASE-засобів і засобів швидкої розробки додатків;
- тестування і розвиток проекту здійснюється одночасно з розробкою.

Засоби RAD дають можливість реалізувати нову технологію створення інформаційних систем: об'єкти формуються як деякі діючі моделі (прототипи), функціонування яких узгоджується з користувачем, а потім розробник може переходити до остаточного формування додатків, не втрачаючи з виду загальної картини проектованої системи.

Серед основних засобів методології RAD слід назвати об'єктно-орієнтоване (візуальне) програмування. Перевага його в тому, що воно оперує стандартними інтерфейсними об'єктами. До першої групи таких об'єктів можна віднести списки, вікна, тексти, які можна легко зв'язати з базами даних і відобразити на екрані монітора. Друга група об'єктів є стандартними елементами керування - кнопки, перемикачі, прапорці, меню, за допомогою яких здійснюється керування даними.

Метод RAD добре зарекомендував себе при створенні невеликих аплікацій.

Методологія функціонального моделювання робіт SADT

Методологія SADT (Structured Analysis and Design Technique - технологія структурного аналізу і проектування) розроблена Дугласом Т. Россом в 1969-1973 роках. Технологія спочатку створювалася для проектування систем більш загального призначення в порівнянні з іншими структурними методами , що вирости з проектування програмного

забезпечення. SADT - одна з найвідоміших і широко використовуваних методик проектування. Нова назва методики, прийняте в якості стандарту - IDEF0 (Icam DEFinition) - частина програми ICAM (Integrated Computer – Aided Manufacturing - інтегрована комп'ютеризація виробництва) , що проводиться по ініціативи ВПС США .

Процес моделювання в SADT включає збір інформації про досліджувану область, документування отриманої інформації і представлення її у вигляді моделі і уточнення моделі. Крім того, цей процес підказує цілком певний шлях виконання погодженої та достовірної структурної декомпозиції, що є ключовим моментом у кваліфікованому аналізі системи. SADT унікальна в своїй здатності забезпечити як графічний мову, так і процес створення несуперечливої і корисною системи описів.

Кожен блок IDEF0- діаграми може бути представлений декількома блоками, з'єднаними інтерфейсними дугами, на діаграмі наступного рівня. Ці блоки представляють подфункції (підмодулі) вихідної функції . Кожен з підмодулей може бути декомпонований аналогічним чином. Число рівнів не обмежується, зате рекомендується на одній діаграмі використовувати не менше 3 і не більше 6 блоків.

Діаграми потоків даних DFD (Data Flow Diagrams)

Діаграми потоків даних використовуються для опису руху документів і обробки інформації як додаток до IDEF0 . На відміну від IDEF0 , де система розглядається як взаємопов'язані роботи і стрілки представляють собою жорсткі взаємозв'язку , стрілки в DFD показують лише те , як об'єкти (включаючи дані) рухаються від однієї роботи до іншої. DFD відображає функціональні залежності значень, що обчислюються в системі, включаючи вхідні значення , вихідні значення і внутрішні сховища даних. DFD - це граф, на якому показано рух значень даних від їх джерел через перетворюють їх процеси до їх споживачів в інших DFD містить процеси, які перетворюють дані, потоки даних, які переносять дані, активні об'єкти, які виробляють і споживають дані, і сховища даних, які пасивно зберігають дані .

Методологія об'єктного проектування на мові UML (UML - діаграми)

Універсальна мова моделювання (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об'єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об'єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему. У Umbrello UML Modeller реалізовано підтримку таких типів діаграм:

Діаграма випадків використання показує дієвих осіб (людей або інших користувачів системи), випадки використання (сценарії використання системи) та їх взаємодію

Діаграми класів, на яких буде показано класи та зв'язки між ними

Діаграми послідовності, на яких показано об'єкти і послідовність методів, якими ці об'єкти викликають інші об'єкти.

Діаграми співпраці, на яких буде показано об'єкти та їх взаємозв'язок з наголосом на об'єкти, які беруть участь у обміні повідомленнями

Діаграми стану, на яких буде показано стани, зміну станів і події у об'єкті або частині системи

Діаграми діяльності, на яких буде показано дії та зміни однієї дії іншою, які є наслідком подій, що сталися у певній частині системи

Діаграми компонентів, на яких буде показано програмні компоненти високого рівня (на зразок KParts або Java Beans).

Діаграми впровадження, на яких буде показано екземпляри компонентів та їх взаємодію.

Діаграми взаємозв'язку сутностей, на яких буде показано дані, взаємозв'язки і умови обмеження зв'язків між даними.

2.4 Вибір методології проектування

Розглянемо на що орієнтовані пред'явлені методології проектування інформаційних систем (рисунок 2.9).

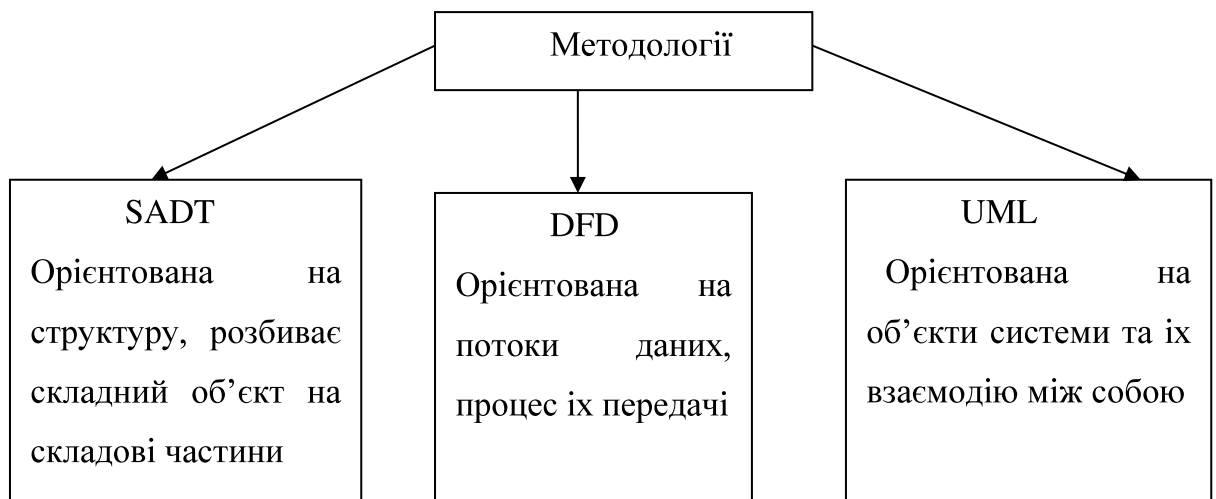


Рисунок 2.9 – Методології проектування ІС

Користувальницький інтерфейс САПР представлений як динамічна система, об'єкти котрої є – елементи (кнопки, піктограми, діалогові вікна), котрі взаємодіють між собою. Тому є сенс використовувати методологію проектування UML для розробки єдиної моделі проектування інтерфейсу САПР, так як ця методологія орієнтована на об'єкти та їх зв'язок.

3 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ КОНФІГУРАЦІЇ ІНТЕРФЕЙСУ

3.1 Загальносистемні рішення

Для побудови складної інформаційної системи необхідним етапом є проектування. Набір з декількох наочних моделей (модельних поглядів) створює у свідомості фахівців інтегральний образ складної комп'ютерної системи, яку вони спільно проектують. Разом з тим, наочні моделі є ефективним засобом документування комп'ютерних систем та їх програмного забезпечення, а також мовою спілкування між програмістами, системними аналітиками та замовниками систем.

Найбільш відомими візуальними моделями, що використовуються для проектування систем та їх програмного забезпечення, є діаграми мови UML. UML (від англ. Unified Modeling Language – уніфікована мова моделювання) – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, званою моделлю UML. UML була створена для визначення, візуалізації, проектування і документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як коди, що інтерпретується, можлива кодогенерація.

Мова UML призначена для розв'язання наступних завдань :

надати у розпорядження користувачів готову до використання виразну потужну мову візуального моделювання, що дозволяє їм розробляти осмислені моделі й обмінюватися ними;

передбачити внутрішні механізми розширюваності й спеціалізації для розширення базових концепцій мови;

забезпечити максимальну незалежність проекту створення програмного забезпечення від конкретних мов програмування (цільових мов програмної реалізації) й процесів розробки;

забезпечити формальну основу для однозначної інтерпретації цієї мови моделювання (мова повинна бути одночасно точною і доступною для розуміння, без надлишкового формалізму);

стимулювати розширення ринку об'єктно-орієнтованих інструментальних засобів створення програмного забезпечення;

інтегрувати кращий практичний досвід щодо використання мови та реалізації програмних засобів її підтримки.

Концептуальна модель мови включає основні будівельні блоки, правила їх поєднання та деякі загальні для усієї мови механізми.

Словник мови UML містить сутності – абстракції, що є основними елементами моделі, та відношення – основні сполучні будівельні блоки. Сутності та відношення за певними правилами поєднуються в конструкції – діаграми.

В UML визначено чотири типи сутностей :

структурні сутності, які розподіляються на основні сутності, різновиди основних сутностей та решту структурних сутностей; до основних сутностей відносяться: клас (Class), інтерфейс (Interface), кооперація (Collaboration), прецедент (Use case), активний клас (Active class), компонент (Component), вузол (Node); до різновидів основних сутностей відносяться: актор (Actor), сигнал (Signal), утиліта (Utility, утиліти є видами класів); процес (Process) і нитка (Thread, нитки є видами активних класів); до решти відносяться: застосування (Application), документ (Document), файл (File), бібліотека (Library), сторінка (Page) та таблиця (Table, таблиці є видами компонентів);

сутності поведінки (Behavioral things), до яких відноситься взаємодія (Interaction) та автомат (State machine);

сутності, що групують, серед яких поки що виділений тільки пакет (Packages);

анотаційні сутності, серед яких поки що виділена тільки примітка (Note).

Основними типами відношень в UML є відношення: залежності (Dependency), асоціації (Association) (різновидом асоціації є відношення

агрегування (Aggregation)), узагальнення (Generalization) та реалізації (Realization). Існують також їх варіації, наприклад, уточнення, трасування, включення та розширення (для залежностей).

Для побудови коректно оформленої моделі в UML визначено правила, що дозволяють коректно і однозначно визначати: імена сутностей, відношень та діаграм, область дії імен (контекст, в якому ім'я має деяке значення), видимість імен (для використання іншими елементами), цілісність (правильність та узгодженість співвідношення елементів), виконання моделі.

Ефективність та спрощення застосування мови забезпечується використанням певних угод, так званих, загальних механізмів: специфікацій (Specifications), доповнень (Adornments), прийнятих розподілів (Common divisions) та механізмів розширення (Extensibility mechanisms).

Кожен елемент нотації UML має унікальне графічне позначення та специфікацію – текстове представлення синтаксису та змістовної семантики відповідного будівельного блоку.

Практично усі будівельні блоки характеризуються дихотомією "клас/об'єкт" та "інтерфейс/реалізація". Це основні підходи поділу реальності при об'єктно-орієнтованому моделюванні систем. Мова UML дозволяє стандартизувати як самі операції та методи мов програмування так і їхню термінологію.

Отже, для проектування розроблюваної автоматизованої інформаційної системи доцільно буде використовувати технологію UML. В якості середовища проектування було використано *StarUML* та *Microsoft Office Visio 2007*.

Проектування моделей інтерфейсу користувача САПР

При проектуванні об'єктів існує два характерні способи: дискретний та послідовний, відповідно цьому можна побудувати дві моделі сценаріїв проектування інтерфейсів користувача.

Такі моделі можна зобразити за допомогою діаграми послідовності.

Дискретна модель зображує таку роботу, при якій після кожної дії виконується повернення в початковий стан як зображено на рисунку 3.1.

Послідовна модель підтримує послідовне виконання дій без обов'язкового повернення в початкову точку. Така модель зображена на рисунку 3.2.

Моделі можуть бути обрані в залежності від об'єкту проектування.

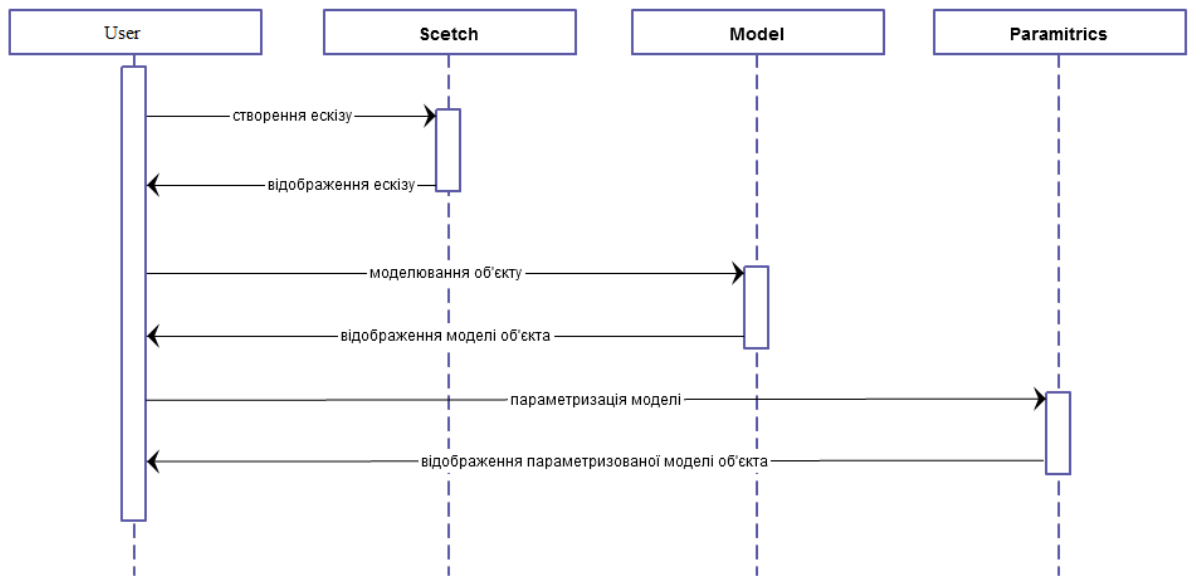


Рисунок 3.1 – Дискретна модель

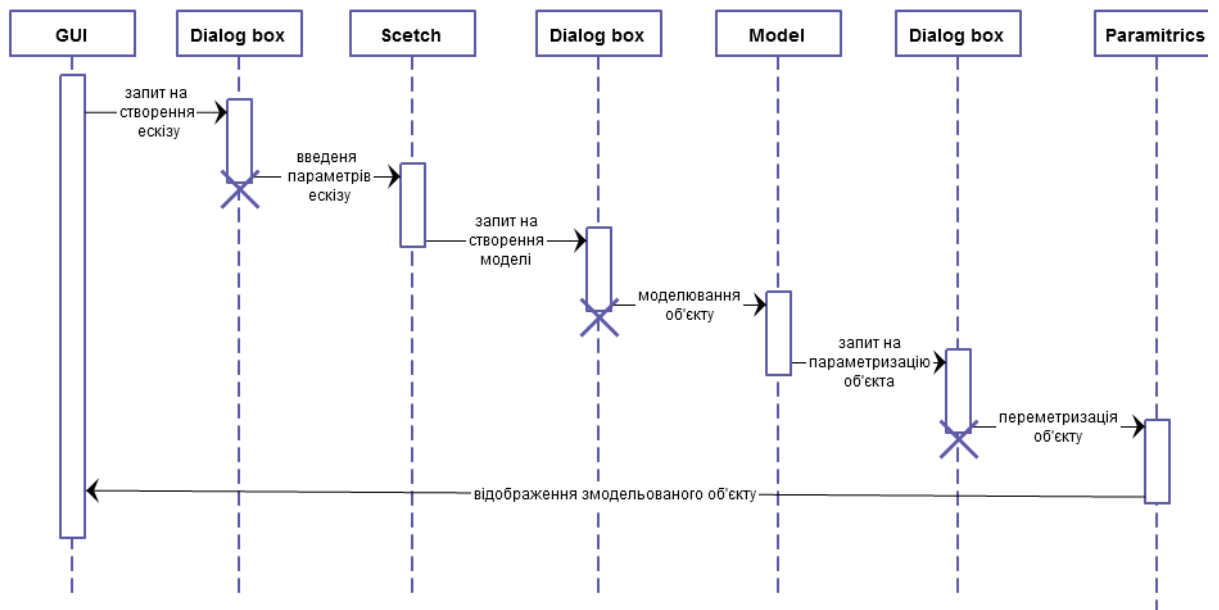


Рисунок 3.2 – Послідовна модель

Діаграма використання

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами.[1] Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, взаємодіючих із системою за допомогою так званих варіантів використання. Варіант використання (use case) служить для опису сервісів, що система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, чинений системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання: • асоціації (association relationship)

- включення (include relationship)
- розширення (extend relationship)
- узагальнення (generalization relationship)

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме — за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML — це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного). Відношення розширення

(extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задається базовим не завжди, а тільки при виконанні додаткових умов.

Така модель показує які функції відображає інтерфейс користувача CAD (рисунок 3.3).

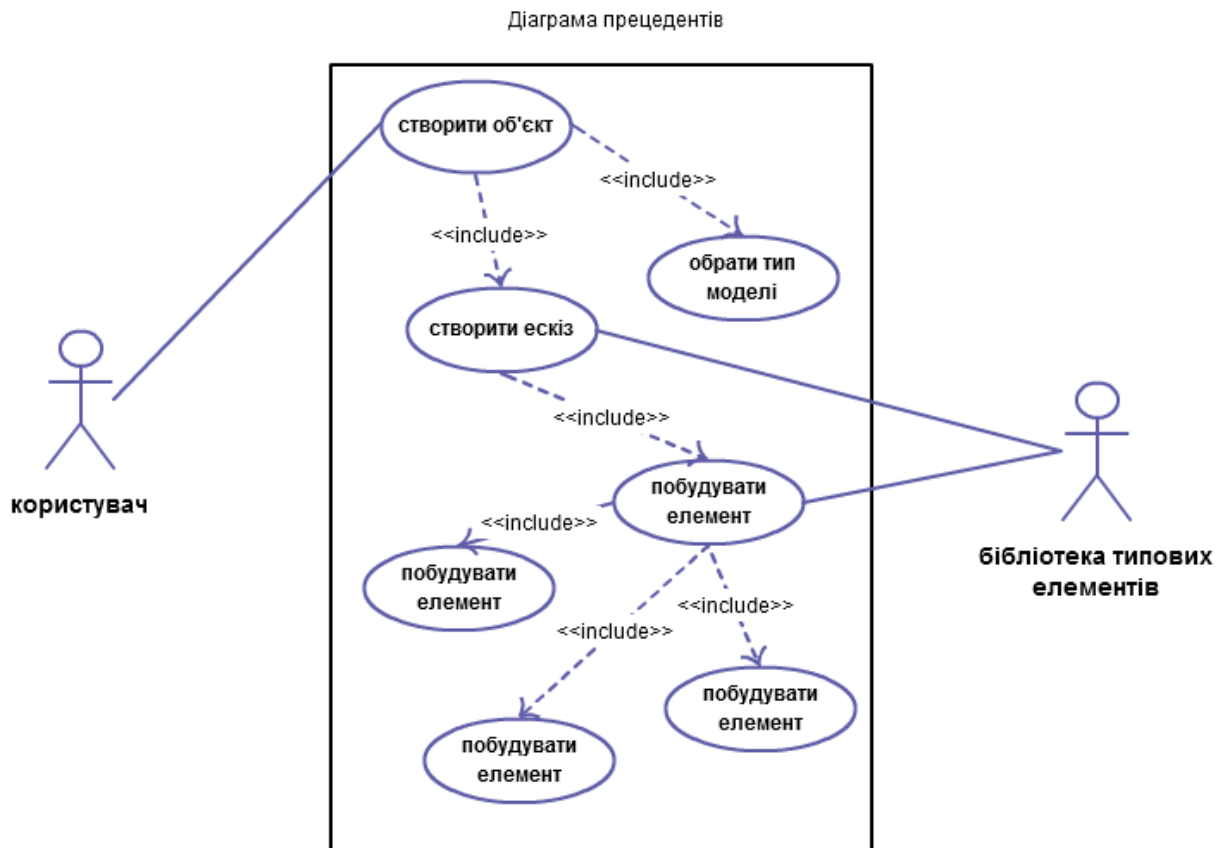


Рисунок 3.3 – Діаграма варіантів використання

Діаграма діяльності

Діаграма діяльності — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія (англ. action) є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально,

включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Така діаграма у випадку проектування інтерфейсів відображає діяльність елементів інтерфейсу, які повинні виконуватись одночасно, наприклад введення даних повинні супроводжуватися відображенням графіки, що зображено на рисунку 3.4.

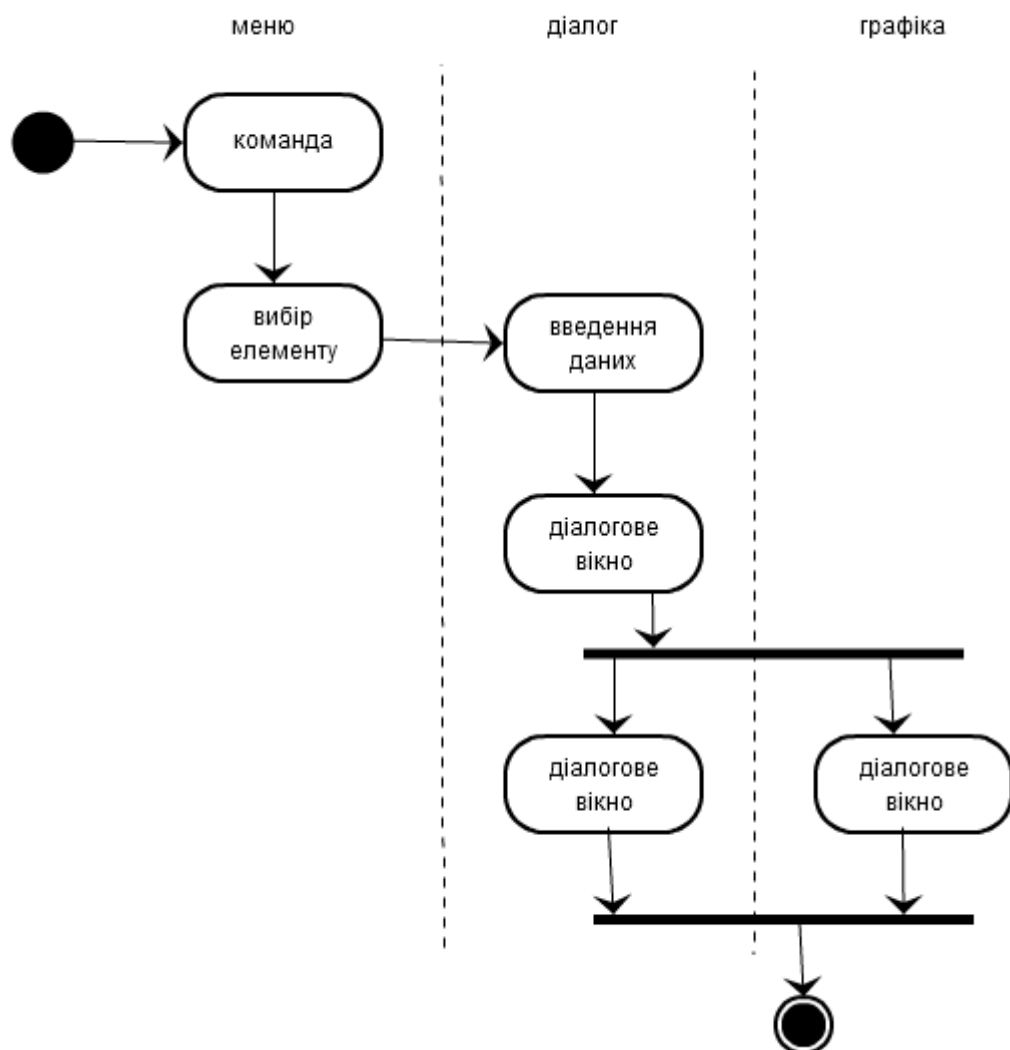


Рисунок 3.4 – Діаграма діяльності

Діаграма станів

На діаграмах станів зображають різні стани об'єкта під час його існування і стимули, які призводять до переходу об'єкта з одного стану у інший.

При проектуванні інтерфейсів діаграмою станів можна показати в яких станах перебуває інтерфейс користувача при виконанні певних дій. Така модель зображена на рисунку 3.5

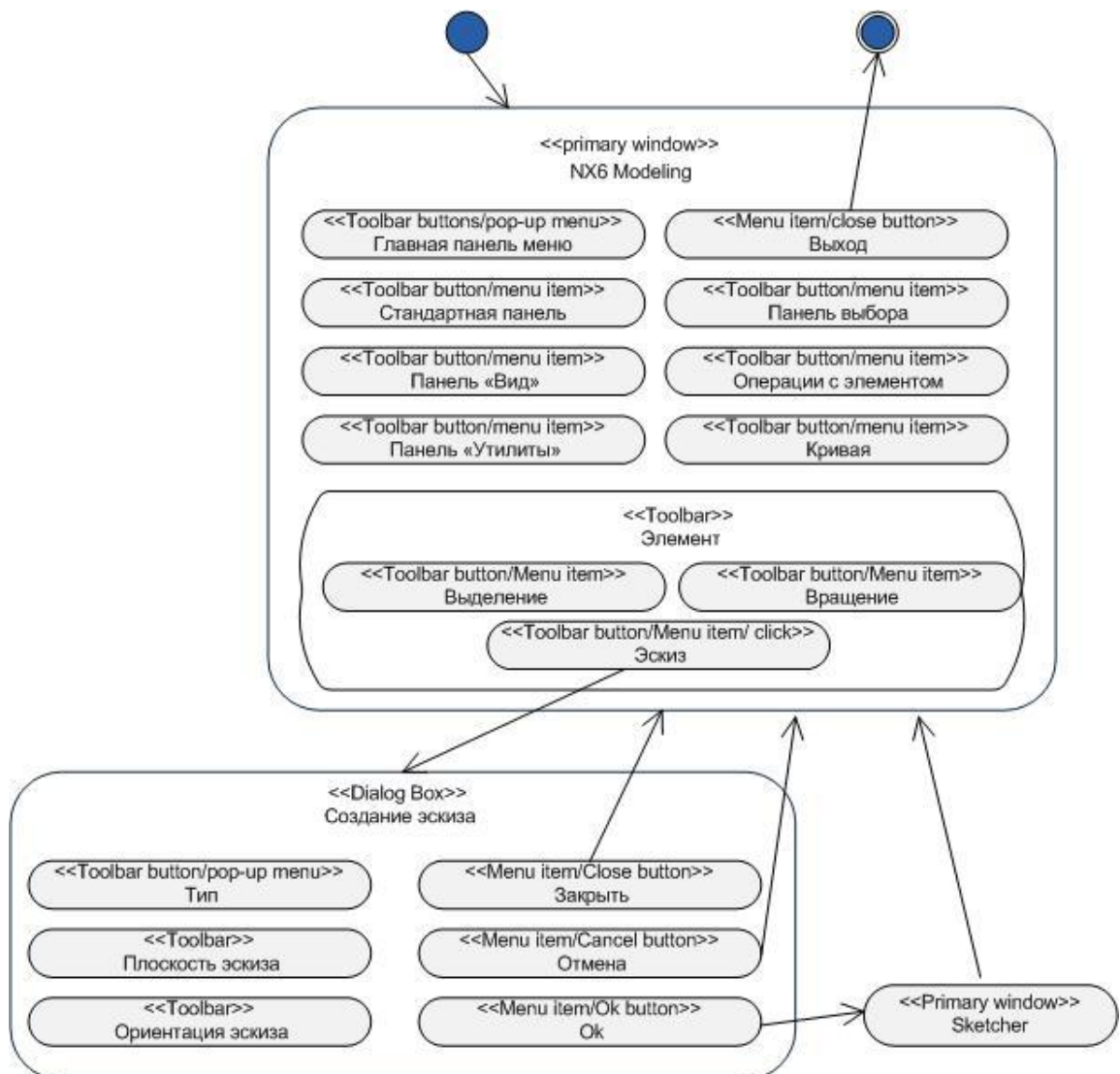


Рисунок 3.5 – Діаграма стану інтерфейсу користувача

Місце підсистеми в інформаційній системі

Моделі для проектування інтерфейсу можуть бути обрані в залежності відоб'єкту проектування. При інтеграції САПР чи при паралельному використанні доцільно зберігати один і той самий сценарій виконання проектних рішень.

На основі того, що сучасні САПР передбачають можливість запису сценаріїв пропонується розробити інформаційну систему по керуванню видом, складом і послідовністю виконання проектних дій.

На даний момент існує така концепція, що при переході з однієї системи в іншу, користувач повинен вивчати роботу в новій з нуля, така концепція зображена на рисунку 3.6.

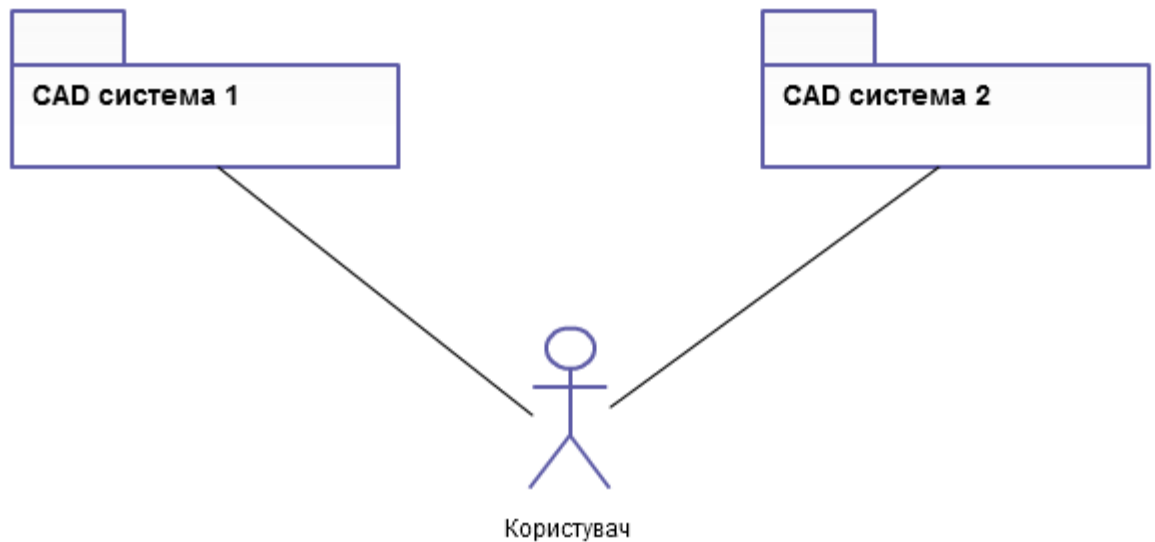


Рисунок 3.6 – Існуюча концепція

Пропонується вирішення такої проблеми концепцією, що зображена на рисунку 3.7 – розробити таку систему, що буде перетворювати та передавати дані конфігурації інтерфейсу однієї системи в іншу за допомогою XML файлу.

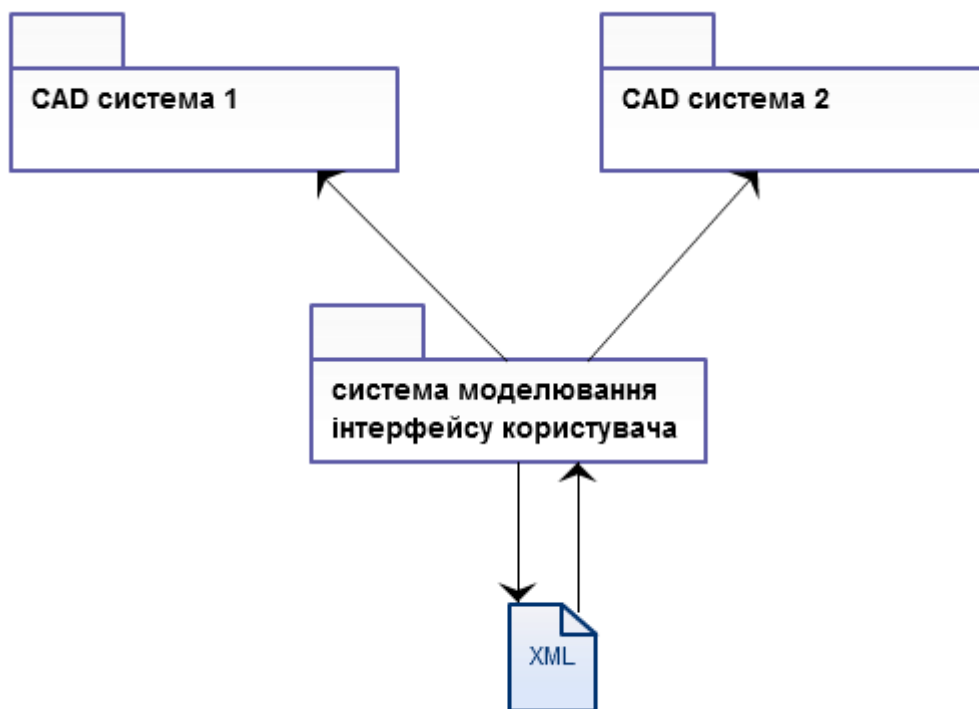


Рисунок 3.7 – Концептуальна модель системи

Опис функцій, що автоматизується

Ціль розробки підсистеми – забезпечити адекватне представлення дій і функцій CAD системи.

Автоматизації підлягають наступні функції:

- створення файлу конфігурації інтерфейсу користувача;
- перетворення файлу конфігурації однієї системи в формат іншої.

Загальний опис системи

Система дозволяє перетворити конфігураційний файл інтерфейсу користувача однієї CAD систему в іншу. Або створити свій файл конфігурації за запитом інженера, в залежності від його потреб в проектуванні.

Адміністратор за потребами інженера в системі обирає потрібну дію та налаштовує для неї інтерфейс користувача, обираючи потрібні інструменти панелі проектування та задає атрибути цих елементів.

Розробка контекстної моделі системи

Перший крок у проектуванні системи – це створення контекстної моделі. Необхідно скласти контекстну діаграму у вигляді діаграми варіантів

використання мови UML. Діаграма прецедентів – в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Прецеденти є основним засобом визначення необхідної поведінки системи. Як правило, вони використовуються для описання вимог до системи, тобто, що має робити система. Основними поняттями, пов'язаними з прецедентами є актори, прецеденти (варіанти використання), та суб'єкт. Суб'єкт – це система, що розглядається і до якої відносяться прецеденти. Користувачі та будь-які інші системи, що можуть взаємодіяти із суб'єктом, представлено як акторів. Актори завжди представляють сутності, що знаходяться за межами системи. Поведінка суб'єкта описується одним або більше прецедентами, що визначаються відповідно до потреб акторів.

Виділимо прецеденти системи:

- Завантажити елементи для конкретної дії;
- Ввести дані;
- Ввести дані з XML файлу;
- Зберегти конфігурації;
- Редагувати налаштування для окремої дії;
- Отримати XML файл;
- Створити конфігурацію для CAD;
- Імпортувати конфігурацію з CAD.

Виділимо діючих осіб (акторів):

- системний адміністратор.

На основі списку прецедентів та акторів складемо діаграму варіантів використання (модель прецедентів). Модель зображено на рисунку 3.1.



Рисунок 3.8 – Контекстна модель системи

3.6 Рішення з організаційного забезпечення

Один з модулів системи, що проектується, вимагає реалізацію одного з методів побудову XML файлу. Враховуючи специфіку поставленої задачі доцільно було б використати для рішення поставлених задач один із засобів швидкої розробки. Наприклад, Java Архітектура для XML Binding (JAXB). JAXB дозволяє розробникам мапувати (ставити у відповідність) класи Java у XML файли. JAXB має дві основні властивості: здатність створювати класи Java з XML і навпаки - створювати XML файли з класів Java.

JAXB особливо корисна, коли специфікація є складною і часто змінюється.

JAXB є частиною платформи Java SE і одним з інтерфейсів API у платформі Java EE, також є частиною Java Web Services Development Pack (JWS DP).

Інструмент "xjc" може бути використаний для конвертації XML Schema а також ряду інших схем(з версії Java 1.6 підтримуються експериментально RELAX NG, XML DTD, так WSDL) у класи Java.

На основі аналізу поставленої задачі мовою програмування було обрано java. Це зумовлено вимогою про кроссплатформеність підсистеми. Виходячи з цього середовищем програмування обрано NetBeans IDE.

1) Java — об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування.

2) Середовище програмування NetBeans IDE

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній

конфігурації. Середовище розробки NetBeans за умовчанням підтримує розробку для платформ J2SE і J2EE.

Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle, проте розробка NetBeans ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

За якістю і можливостями останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, авто доповнення мовних конструкцій на льоту, шаблони коду та інше.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

Розробка робочої документації

Робоча документація на автоматизовану інформаційну систему моделювання інтерфейсу користувача САПР – частина документації, що необхідна для розробки й налаштування системи в цілому, а також та документація, що входить до системи програмно-технічних, програмно-методичних комплексів і компонентів технічного, програмного, інформаційного й організаційного забезпечення.

Перерахування найменувань розроблюваних документів і їхня комплектація представлені в технічному завданні.

Технічний опис представлений у Додатку А.

Програмний текст модуля розподілу ресурсів представлено у Додатку Б.

Методика випробувань представлена у Додатку В.

3.7 Рішення з інформаційного забезпечення

Для створення файлу конфігурації було обрано XML файл, котрий має структуру зображену на рисунку 3.9

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої, шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних.

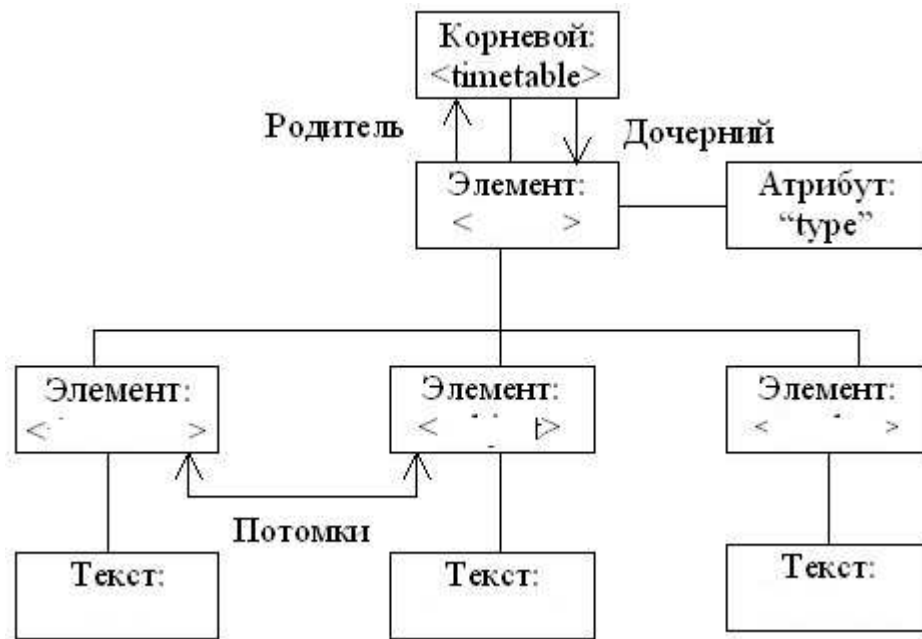


Рисунок 3.9 – Структура файлу конфігурації

Файл конфігурації містить в собі елементи та їх атрибути, список яких представлений на рисунку 3.10

Element	<text></text>	
Attributes	type	static, edit
	label	"..."
	listener	app, handlerID, this
	action	notify, close, ...
Contains	, <tooltip>	
Element	<button></button>	
Attributes	type	push, popup, text
	label	"..."
	listener	app, handlerID, this
	action	notify, close, ...
	statusmsg	"..."
Contains	, <tooltip>	
Element	<toolbar></toolbar>	
Attributes	align	top, bottom
	label	"..."
	dockable	true, false
	<button>, <spacer>, <text>, <menu>, <tooltip>	
Contains	<button>, <spacer>, <text>, <menu>, <tooltip>	
Element	<distview></listview>	
Attributes	type	bigicon, smallicon, detail
	label	"..."
	src	data source url
Contains	<tooltip>	

Рисунок 3.10 – Елементи та атрибути конфігураційного файлу

Розробка концептуальної моделі бази знань

З метою проектування інформаційного сховища автоматизованої системи розробимо концептуальну модель даних. Концептуальну модель зазвичай представляють у вигляді діаграми «сутність-зв'язок» (Entity-Relationship Diagram, ERD). В мові UML ця діаграма відсутня, але вона дуже зручна. З її допомоги визначаються важливі для предметної області об'єкти, їхні властивості (атрибути) і відносини між ними (зв'язки). В концептуальній моделі нашої бази буде зберігатися дія, елементи які відносяться до дії та атрибути цих елементів. Наприклад, дія «Проектування деталей», елементи – це кнопки(інструменти проектування): Коло, лінія, сплайн, витягування та ін. А атрибути – позиція, колір та ін. Концептуальна модель зображена на рисунку 3.11

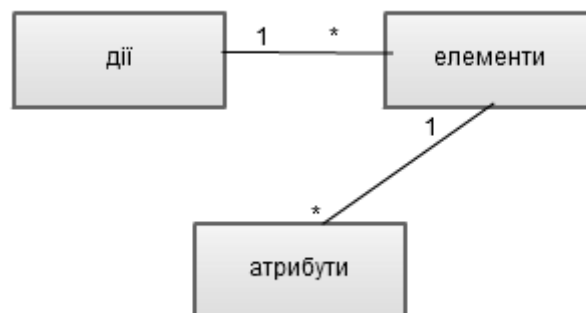


Рисунок 3.11 – Концептуальна модель бази знань

3.8 Рішення з технічного забезпечення

Під час проектування автоматизованої інформаційної системи розподілу ресурсів при проектуванні інформаційної системи був проведений аналіз структури і задач, на які націлена дана інформаційна система. На основі цього аналізу були зроблені висновки про склад і параметри технічних засобів, які будуть задіяні при використанні інформаційної підсистеми.

Модулі системи розташовані в різних підрозділах підприємства. Отже, необхідно передбачити розробку структури й параметрів мережного забезпечення. Необхідно визначити достатні й оптимальні вимоги до технічного забезпечення безпосередньо для кожного відділу. При цьому необхідно визначити склад і параметри технічних засобів таким чином, щоб була передбачена можливість використання існуючих обчислювальних систем, засобів комунікації. Це приведе до зменшення фінансових витрат на впровадження інформаційної підсистеми.

Розташування кожного елемента системи проектувалося з урахуванням технічних вимог до установки й експлуатації, зазначених у супровідній документації до встаткування й техніки безпеки:

проведення мережних кабелів у спеціальних кожухах, що запобігають механічним ушкодженням;

заборона проведення силових кабелів поблизу мережного встаткування;

кабелі не повинні бути передавлені або натягнуті;

корпуси системних блоків повинні бути заземлені.

Роботу автоматизованої системи забезпечуватиме сервер баз даних, робочі станції.

Вимоги до технічних засобів:

1. Персональний комп'ютер на базі процесора Intel Pentium 500 МГц та вище.
2. Оперативна пам'ять – 64 Мб і вище.
3. Місце на жорсткому диску – не менше 10 Мб.
4. Відеоадаптер SVGA 800x600 16 біт кольорів та вище.

5. CD-ROM x24 та вище.
6. Маніпулятор «миша».
7. Клавіатура.
8. Мінімальний об'єм дискового простору необхідний для встановлення складає 2,5 Мб.

Вимоги до операційної системи та програмного забезпечення:

Для роботи «Електронний журнал» потрібна встановлена операційна система Microsoft Windows 98/2000/XP і JDK 7

3.9 Рішення з програмного забезпечення

З метою опису сутностей системи та зв'язків між ними, на основі розробленої моделі даних системи було розроблено діаграму класів. З урахуванням поставлених задач діаграма має включати сутності та зв'язки між ними. Модель класів представлено на рисунку 3.12.

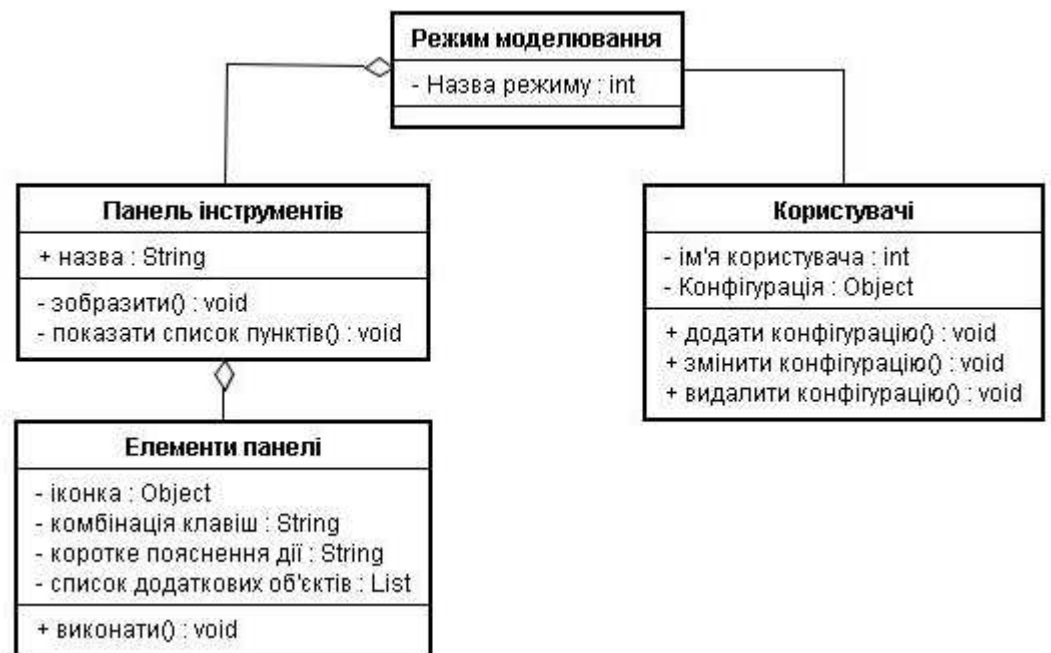


Рисунок 3.12 – Діаграма класів

Діаграма варіантів використання

При моделюванні поведінки проекрованої або аналізованої системи виникає необхідність не лише представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій. Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності, які можна вважати окремим випадком діаграм станів. Саме вони дозволяють реалізувати в мові UML особливості процедурного і синхронного управління, обумовленого завершенням внутрішніх діяльностей і дій. Основним напрямом використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їх виконання.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку. Діаграма діяльності — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Діаграма діяльності нашої системи зображена на рисунку 3.13

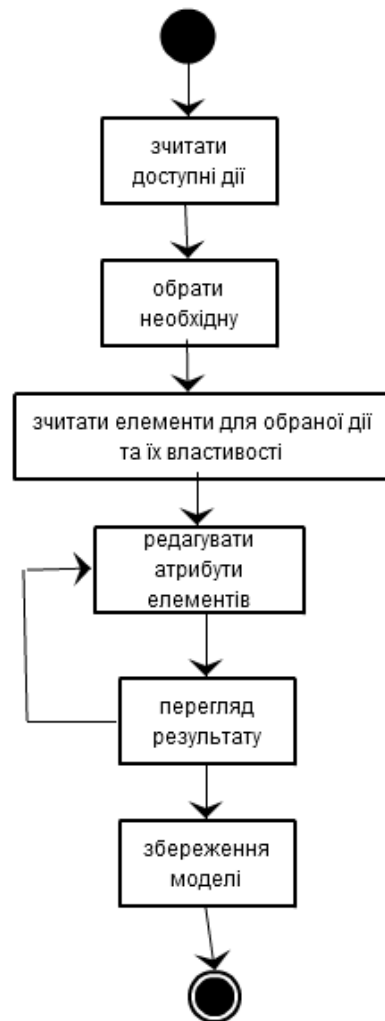


Рисунок 3.13 – Діаграма діяльності системи

Рішення з реалізації програмного забезпечення

Для подальшого програмування частини програмного забезпечення мовою програмування JAVA на підставі розробленого проекту була створена діаграма компонентів, яку наведено на рисунку 3.14

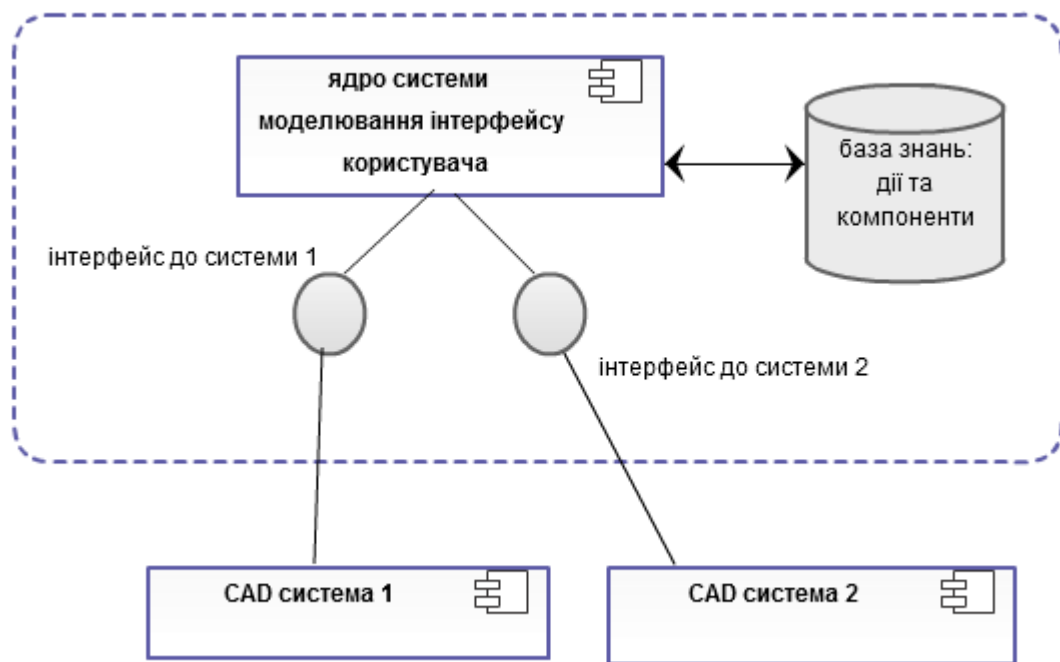


Рисунок 3.14 – Діаграма компонентів системи

Специфікація програмних модулів

Специфікацію програмних модулів приведено у таблиці 3.6.

Таблиця 3.6. Специфікація програмних модулів

<i>Модуль</i>	<i>Вхідні дані</i>	<i>Вихідні дані</i>
Створення файлу конфігурацій	Назва процесу проектування, елементи та їх атрибути	Файл конфігурацій
Завантаження файлу конфігурацій	Ім'я файлу	Зміст файлу конфігурацій

Вибір та опис мови програмування

На основі аналізу поставленої задачі мовою програмування було обрано java. Це зумовлено вимогою про кроссплатформеність підсистеми. Виходячи з цього середовищем програмування обрано NetBeans IDE.

1) Java — об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформи-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

2) Середовище програмування NetBeans IDE

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за умовчанням підтримує розробку для платформ J2SE і J2EE.

Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle, проте розробка NetBeans ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

За якістю і можливостями останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних

конструкцій кольором, авто доповнення мовних конструкцій на льоту, шаблони коду та інше.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

ВИСНОВКИ

Після дослідження особливостей проектування інтерфейсів користувача можна скласти певну картину того, що пріоритетними напрямками в побудові інтерфейсів є, нажаль, те, що воно спрямоване більше на проектувальника, а не на самого користувача.

CAD – є складною системою, котра має багато функцій та модулів, які виконуються. З точки зору інтерфейсу це є дуже важливо, тому що треба раціонально розмістити всі ці функції на робочому просторі так, щоб полегшити роботу користувача. Отже існують багато моделей, правил проектування і принципів побудови інтерфейсів. І при побудові інтерфейсів треба приділяти велику увагу на те, щоб він був інтуїтивно зрозумілим та зручним в використанні для будь-якого користувача.

Усі методи проектування інтерфейсів, які існують і використовуються для виконання цієї задачі мають один суттєвий недолік – відсутність врахування впливу функцій програмного продукту, їх кількості в системах автоматизованого проектування. Актуальним та доцільним у цьому випадку є використання системної методології UML для проектування інтерфейсів користувача САПР. Аналіз можливостей використання методології UML до проектування інтерфейсів користувача САПР показав, що користувацький інтерфейс САПР представлений як динамічна система, об'єкти котрої є – елементи (кнопки, піктограми, діалогові вікна), котрі взаємодіють між собою. Тому є сенс використовувати методологію проектування UML для розробки єдиної моделі проектування інтерфейсу САПР, так як ця методологія орієнтована на об'єкти та їх зв'язок.

В результаті виконання даної дипломної роботи була розроблена автоматизована система розробки конфігурації інтерфейсів САПР. ресурсів. Для цього були виконані аналіз предметної області та постановка задачі, розроблено технічне завдання та проектні рішення інформаційної системи, що включають:

загальносистемні рішення;

рішення з інформаційного забезпечення;

рішення з математичного забезпечення;

рішення з організаційного забезпечення;

рішення з програмного забезпечення;

рішення з технічного забезпечення.

У рамках цих проектних рішень був розроблений програмний модуль, генерує файл конфігурації інтерфейсу обраної САПР.

Крім того було виконано:

розрахунок економічної ефективності впровадження інформаційної системи;

розробка методичного програмного забезпечення з цивільного захисту;

дослідження з охорони праці та охорони навколишнього середовища.

З урахуванням усього вищенаведеного можна сказати, що цілі, поставлені на початку даної дипломної роботи були повністю виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ясінецька І.А. Державний земельний кадастр як інформаційна база/ Економіка і суспільство/2018. – 680 с.
2. Тимошевська Т.І. Застосування гіс-технологій під час виконання кадастрових робіт у населених пунктах/ Вчені записки ТНУ імені В.І. Вернадського/2021. – 283 с.
3. Попов А.С. Кадастрові та реєстраційні системи країн світу/ Монографія/2018. – 216 с.
4. Enemark S. Cadastral Systems – the Nordic Approach / S. Enemark / The Danish Journal for Mapping and Land Use. – Landinspektoren, 2019. – No. 5. – P. 464-471.
5. Enemark S. Land Administration Infrastructures for Sustainable Development / S. Enemark / The Seventh United Nations Regional Cartographic Conference for the Americas "On Spatial Data and Development: Building a Sustainable Infrastructure", 22-26 January, 2021, New York [Електронний ресурс]. – Режим доступу: http://www.fig.net/council/enemark_papers/unrcca_enemark_1.pdf
6. Sevatdal H. Land Administration in Norway / H. Sevatdal [Електронний ресурс]. – Режим доступу: http://www.myland.org.ua/userfiles/file/SEVATDAL_ML.pdf
7. Rossinelli S. The Swedish Cadastral System: Essay Report / S. Rossinelli [Електронний ресурс]. – Режим доступу: <http://www.geo21.ch/ethz/20018/essays/E1-Schweden.pdf>
8. Кошкарев А.В. Инфраструктура пространственных данных Финляндии / А.В. Кошкарев // Пространственные данные. – 2018. – №1. [Электронный ресурс]. – Режим доступу: <http://www.gisa.ru/44536.html>
9. Vitikainen A. Land Consolidation and the Guidelines on Land Administration / A. Vitikainen [Електронний ресурс] – Режим доступу: http://portal.survey.ntua.gr/main/labs/photo/research/wg_33/wpla/papers/TPS6.4.Arvo%20Vitikainen.doc

10. Евдокимов П. В. С. Практическое руководство / Наука і Техніка /2022 – 416 с.
11. Черін, А.Г. Стандартизація геоінформаційних сервісів [Текст] // Вісн. геодез. та картогр. – 2019. – № 4. – С. 34-39.
12. Lithuania: Country Report 2010 [Електронний ресурс] – Режим доступу: <http://www.cadastraltemplate.org/countryreport/Lithuania-27Sep 2021.pdf>
13. Остапцов С.А. Документообіг по охороні праці на підприємстві – загальні запитання / С.А. Остапцов, В.В. Новиков, С.А. Солод // Черезвичайні ситуації: промислова та екологічна безпека. – 2018. – № 1. – С. 52-56. – ISSN 2305-6843
14. Марк Прайс. C# 9 и .NET 5 Разработка и оптимизация / / C# 9 and .NET 5 – Modern Cross-Platform Development 5th Edition / 2022 – 832 с.
15. Лященко А.А./ Сервіс-орієнтована архітектура кадастрових геоінформаційних систем та кадастрових геопорталів / 2018 – 256 с.
16. Про затвердження Вимог до структури, змісту та формату оформлення результатів робіт із землеустрою в електронному вигляді (обмінного файлу) // Наказ Держкомзему від 02.11.2009 р. № 573, зареєстрований в Мінюсті України 15.02.2010 № 157/17452. URL: <https://zakon.rada.gov.ua/laws/show/z0157-10#Text>
17. Лященко А.А., Ціпенко О.В. Наскрізнi геоінформаційні технології грошової оцінки земель населених пунктів. Інженерна геодезія. Київ: КНУБА, 2019. Вип. 42. – С. 155–160.
18. Закону України «Про державний земельний кадастр» [Електронний ресурс]. – Режим доступу: <https://xn--80aagahqwyibe8an.com/zakon-ukrajiny/stattya-printsipi-derjavnogo-zemelnogo-116636.html>
19. Карпінський, Ю.О. Технологічні моделі розподілених кадастрових ГІС в Intranet/Internet-мережах [Текст] / Ю.О. Карпінський, А.А. Лященко // Містобудування та територіальне планування. – Вип.17. – К.: КНУБА, 2020. – С. 106

20. Постановою Кабінету Міністрів України. Порядок інформаційної взаємодії між кадастрами та інформаційними системами [Електронний ресурс]. – Режим доступу: www.tax.gov.ua/data/normativ/000/000/63039/ZATVERDZHENO.doc
21. Бардус І.О., Лазарєв М.І., Ніценко А.О.. «Бази даних у схемах (на основі фундаменталізованого підходу)». - 2017
22. Лященко, А. А. ... // Містобудування та територіальне планування : науково-технічний збірник / МОН України, КНУБА. – Київ : КНУБА, 2024 – С. 122
23. Остапцов С.А. Документообіг по охороні праці на підприємстві – загальні запитання / С.А. Остапцов, В.В. Новиков, С.А. Солод // Черезвичайні ситуації: промислова та екологічна безпека. – 2018. – № 1. – С. 52-56. – ISSN 2305-6843
24. Петрова, Корощенко, Айзман. Охорона праці а підприємстві та навчальному процесі Петрова, Корощенко, Айзман. – 2017. – 336 с.
25. Основи екології та охорони навколишнього середовища:навчально-методичний посібник / Е.Т. Тимонова, И. А. Тимонов. –2011. – 228 с.