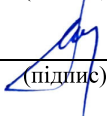


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**Національний університет кораблебудування імені адмірала Макарова****Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій****"Допущений до захисту"****Завідувач кафедри ІУСТ****к.т.н, доц. Михелєв І.Л.****" ____ " _____ 2025 р.*****КВАЛІФІКАЦІЙНА РОБОТА*****на здобуття ступеня вищої освіти "магістр"****Дослідження технологій розробки системи проєктування з використанням
хмарних обчислень
на тему:****Виконав: студент групи 6141 м****Богза Є.Д.**
(підпис)**Керівник роботи:****доцент кафедри ІУСТ, к.т.н.,**
(посада, науковий ступінь вчене звання)**Гайда А.Ю.**
(підпис)**м. Миколаїв – 2025 р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

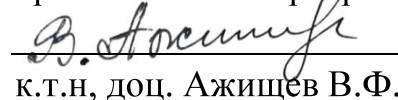
Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми



к.т.н, доц. Ажищев В.Ф.

"___" _____ 2025 р.

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

Студенту Богза Євгенію Дмитровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження технологій розробки системи проєктування з використанням хмарних обчислень

Керівник роботи Гайда Анатолій Юліанович

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (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)

Вступ, • Аналіз предметної області і постановка задачі, • Проведення досліджень за темою магістерської роботи, • Реалізація проєкту інформаційної системи (технічна документація по проєкту), • Результати розробки, • Організаційно-економічний розділ, • Розділ з охорони праці • Висновки, • Список використаної літератури, • Додатки Технічне завдання, текст програми, інструкція користувачу

5. Перелік презентаційних матеріалів

1 Варіанти обробки даних 2 Недоліки існуючої системи 3 Концепція розробляємої системи

4 Модель варіантів використання 6 Модель взаємодії системи із зовнішніми сутностями

7 Концептуальна модель даних 8 Фізична модель даних 9 Діаграма класів ПЗ

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	К.т.н. доц. Гайда А.Ю.	05.09.2024	22.12.2024
2	К.т.н. доц. Гайда А.Ю.	23.12.2024	15.06.2025
3	К.т.н. доц. Гайда А.Ю.	15.06.2025	25.09.2025
4	К.т.н. доц. Гайда А.Ю.	25.09.2025	14.11.2025
5	К.т.н. доц. Гайда А.Ю.	25.09.2025	14.11.2025
6	К.т.н. доц. Гайда А.Ю.	25.09.2025	14.11.2025

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	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент


 (підпис)
Богза Є.Д.
(ПІБ)

Керівник роботи


 (підпис)
Гайда А.Ю.
(ПІБ)

АНОТАЦІЯ

Магістерська дипломна робота на тему «Дослідження технологій розробки системи проєктування з використанням хмарних обчислень».

Магістерська дипломна робота викладена на 134 сторінках машинописного тексту формату А4, містить 28 рисунків, 10 таблиць та список використаних джерел з 23 найменувань. Робота має 3 додатки, які містять технічне завдання, текст програмного коду та інструкцію користувача.

Метою роботи є підвищення ефективності функціонування систем автоматизованого проєктування за рахунок використання технологій хмарних обчислень.

Об'єкт і предмет дослідження:

- **Об'єкт дослідження** - процеси проєктування архітектури ІС в системах автоматизованого проєктування.
- **Предмет дослідження** - моделі і технології розробки архітектури ІС в системах автоматизованого проєктування з використанням хмарних обчислень.

Наукова новизна:

- Запропоновано мультітенантна модель побудови програмного забезпечення систем автоматизованого проєктування з використанням технологій хмарних обчислень.
- Запропоновано модель визначення оптимальної конфігурації хмарного додатка в залежності від необхідних ресурсів.

Ключові слова: інформаційна система, хмарні технології, архітектура інформаційної системи, системах автоматизованого проєктування.

ABSTRACT

Master's thesis on the topic "Research on technologies for developing a design system using cloud computing". The master's thesis is presented on 134 pages of typewritten text in A4 format, contains 28 figures, 10 tables and a list of sources used with 23 names. The work has 3 appendices, which contain the technical specifications, the text of the program code and the user manual.

The purpose of the work is to increase the efficiency of the functioning of computer-aided design systems through the use of cloud computing technologies.

Object and subject of research:

Object of research - processes of designing IS architecture in automated design systems.

Subject of research - models and technologies of developing IS architecture in automated design systems using cloud computing.

Scientific novelty:

A multi-tenant model for building software for automated design systems using cloud computing technologies has been proposed.

A model for determining the optimal configuration of a cloud application depending on the required resources has been proposed.

Keywords: information system, cloud technologies, information system architecture, computer-aided design systems.

ЗМІСТ

<u>Вступ</u>	
<u>1 САПР ЯК ОРГАНІЗАЦІЙНО-ТЕХНІЧНА СИСТЕМА</u>	
<u>1.1 Введення до САПР</u>	
<u>1.2 Різновиди САПР</u>	
<u>1.3 Завдання проектування, вирішуються в сучасних САПР</u>	
<u>1.4 Структура САПР</u>	
<u>1.5 Особливості систем автоматизованого проектування</u>	
<u>1.6 Функції, характеристики і приклади CAD/CAM/CAE-систем</u>	
<u>2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ХМАРНИХ ОБЧИСЛЕНЬ</u>	
<u>2.1 Основні положення хмарних обчислень</u>	
<u>2.2 Види хмарних обчислень</u>	
<u>2.3 Моделі хмарних сервісів</u>	
<u>2.4 Економічні переваги хмарних обчислень у порівнянні з традиційними підходами розгортання додатків</u>	
<u>2.5 Переваги хмарних обчислень</u>	
<u>2.6 Недоліки та проблеми хмарних обчислень</u>	
<u>2.7 Порівняння Грід і хмарних обчислень</u>	
<u>2.8 Постачальники хмарних обчислень</u>	
<u>2.9 Прогнози розвитку ринку хмарних сервісів</u>	
<u>3 ЗАГАЛЬНІ ПРИНЦИПИ ПОБУДОВИ САПР НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ</u>	
<u>3.1 Існуючі моделі побудови САПР на основі хмарних обчислень</u>	
<u>3.2 Побудова архітектури ПЗ САПР, що використовує сервіси хмарних обчислень</u>	
<u>3.3 Побудова архітектури ПЗ на основі хмарних сервісів</u>	
<u>3.4 Концепція побудови систем проектування меблів з використанням технологій хмарних обчислень</u>	
<u>3.5 Інформаційне забезпечення</u>	

<u>4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ</u>	
<u>ПЗ ДЛЯ РОЗРОБКИ САПР</u>	
<u>4.1 Вступ.....</u>	
<u>4.2 Розрахунок витрат на створення й експлуатацію інформаційної</u>	
<u>системи</u>	
<u>4.3 Економічна ефективність розробки і впровадження системи</u>	
<u>5 ОХОРОНА ПРАЦІ</u>	
<u>5.1 Загальні відомості</u>	
<u>5.2 Аналіз шкідливих та небезпечних факторів в офісі фірми «Технос»</u>	
<u>5.3 Розрахунок системи штучного освітлення приміщення.....</u>	
<u>6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА.....</u>	
<u>Вступ.....</u>	
<u>6.1 Забруднення навколишнього середовища твердими побутовими</u>	
<u>відходами</u>	
<u>6.2 Розробка заходів щодо зменшення забруднення.....</u>	
<u>Висновки</u>	
<u>Список використаних джерел.....</u>	
<u>Додатки.....</u>	

Вступ

Системи автоматизованого проектування орієнтовані на автоматизацію процесу проектування, розробку складних технологічних процесів, конструкцій, і матеріалів і є сьогодні одним з ключових факторів забезпечення конкурентоспроможності будь-якого високотехнологічного виробництва.

На сьогоднішній день постійно зростає обчислювальна складність задач інженерного проектування, і потрібні значні обчислювальні ресурси для виконання інженерного моделювання. Вирішення цієї проблеми полягає у використанні багатопроцесорних систем.

Однак, через високі вимоги до технічного оснащення САПР і її дорожнечю з одного боку і вимог до наявності специфічних знань, умінь і навичок в області високопродуктивних обчислень з іншого, малим підприємствам-розробникам САПР практично не можливо вийти на економічний ринок – їх продукція не є конкурентоспроможною. Тим не менш, сьогодні ситуацію, що сформувалася, може поліпшити застосування хмарних технологій. Перехід до хмарної інфраструктури може практично вирівняти в можливостях підприємства малого і середнього бізнесу з великими розробниками САПР.

Технологія хмарних обчислень є ще досить молодого, і часто побудова застосунків з використанням хмарних сервісів має індивідуальний характер, що не має методичної бази, тому актуальною залишається задача розробки моделі побудови архітектури САПР, застосування якої надасть максимальний економічний ефект.

Метою даної роботи є підвищення ефективності функціонування систем автоматизованого проектування за рахунок використання технологій хмарних обчислень.

Для досягнення цієї мети необхідно вирішити наступні задачі:

1. Провести аналіз стану сучасного ринку САПР.
2. Дослідити завдання САПР в сучасних умовах ринку.
3. Дослідити тенденції перспективного розвитку і експлуатації САПР.

4. Провести аналіз сучасних інформаційних технологій для колективної розробки проектів.

5. Розробити модель побудови систем автоматизованого проектування на основі технологій хмарних обчислень, конкурентоспроможну на ринку САПР.

Об'єктом дослідження даної роботи є процеси проектування архітектури ПЗ в системах автоматизованого проектування. Предметом дослідження – моделі і технології розробки архітектури ПЗ в системах автоматизованого проектування з використанням хмарних обчислень.

1 САПР ЯК ОРГАНІЗАЦІЙНО-ТЕХНІЧНА СИСТЕМА

1.1 Введення до САПР

Автоматизація проектування займає особливе місце серед інформаційних технологій: автоматизація проектування – синтетична дисципліна, її складовими частинами є багато інших сучасних інформаційних технологій. Так, технічне забезпечення систем автоматизованого проектування (САПР) засноване на використанні обчислювальних мереж і телекомунікаційних технологій, в САПР використовуються персональні комп'ютери й робочі станції.

Математичне забезпечення САПР відрізняється багатством і різноманітністю використовуваних методів обчислювальної математики, статистики, математичного програмування, дискретної математики, штучного інтелекту. Програмні комплекси САПР відносяться до числа найбільш складних сучасних програмних систем, що базуються на операційних системах Unix, Windows мовах програмування. C, C++, Java та інших, сучасних CASE технологіях, реляційних і об'єктно-орієнтованих системах керування базами даних (СКБД), стандартах відкритих систем і обміну даними у комп'ютерних середовищах.

До теперішнього часу створена велика кількість програмно-методичних комплексів для САПР з різними рівнем спеціалізації і прикладною орієнтацією. В результаті автоматизація проектування стала необхідною складовою частиною підготовки інженерів різних спеціальностей; інженер, який не володіє знаннями і не вміє працювати до САПР, не може вважатися повноцінним фахівцем.

1.2 Різновиди САПР

Класифікацію САПР роблять за застосунком, цільовим призначенням, масштабом (комплексності вирішуваних завдань), характером базової підсистеми і т. д. [1].

За застосунками найбільш представницькими та широко використовуваними є наступні групи САПР:

- САПР для застосування у галузях загального машинобудування. Їх часто називають машинобудівними САПР або MCAD (Mechanical CAD) системами;
- САПР для радіоелектроніки. Їх назви – ECAD (Electronic CAD) або EDA (Electronic Design Automation) системи.
- САПР у області архітектури і будівництва;

Крім того, відоме велике число більш спеціалізованих САПР, або що виділяються у зазначених групах, або представляють самостійну гілку у класифікації. Прикладами таких систем є САПР великих інтегральних схем (ВІС); САПР літальних апаратів; САПР електричних машин і т.п.

За цільовим призначенням розрізняють САПР або підсистеми САПР, що забезпечують різні аспекти проектування. Так, у складі MCAD з'являються CAE / CAD / CAM системи:

- САПР функціонального проектування, інакше САПР-Ф або CAE (Computer Aided Engineering) системи.
- Конструкторські САПР загального машинобудування – САПР-К, часто звані просто CAD системами;
- Технологічні САПР загального машинобудування – САПР-Т, інакше звані автоматизованими системами технологічної підготовки виробництва АСТПП або системами CAM (Computer Aided Manufacturing).

За масштабом розрізняють окремі програмно-методичні комплекси (ПМК) САПР, наприклад, комплекс аналізу міцності механічних виробів відповідно до методом кінцевих елементів (МКЕ) або комплекс аналізу електронних схем; системи ПМК; системи з унікальними архітектурами не тільки програмного (software), але і технічного (hardware) забезпечень.

За характером базової підсистеми розрізняють такі різновиди САПР:

- САПР на базі машинної графіки та математичного моделювання. Ці САПР орієнтовані на додатки, де основною процедурою проектування є

конструювання, тобто визначення просторових форм і взаємного розташування об'єктів. Тому до цієї групи систем відноситься більшість графічних ядер САПР в області машинобудування.

- САПР на базі СУБД. Вони орієнтовані на додатки, в яких при порівняно нескладних математичних розрахунках переробляється великий обсяг даних. Такі САПР переважно зустрічаються в техніко-економічних програмах, наприклад, при проектуванні бізнес-планів, але мають місце також при проектуванні об'єктів, подібних щитів управління в системах автоматики.

- САПР на базі конкретного прикладного пакету. Фактично це автономно використовувані програмно-методичні комплекси, наприклад, імітаційного моделювання виробничих процесів, розрахунку міцності за методом кінцевих елементів, синтезу та аналізу систем автоматичного керування і т.п. Часто такі САПР відносяться до систем САЕ. Прикладами можуть служити програми логічного проектування на базі мови VHDL, математичні пакети типу MathCAD.

- Комплексні (інтегровані) САПР, що складаються із сукупності підсистем попередніх видів. Характерними прикладами комплексних САПР є САЕ / CAD / САМ-системи в машинобудуванні або САПР БІС. Так, САПР БІС включає в себе СУБД і підсистеми проектування компонентів, принципів, логічних і функціональних схем, топології кристалів, тестів для перевірки придатності виробів. Для управління такими складними системами застосовують спеціалізовані системні середовища.

1.3 Завдання проектування, вирішуються в сучасних САПР

Різноманітні задачі проектування, які вирішуються в сучасних САПР, можна об'єднати у чотири групи функцій [1]:

1. Геометричне моделювання.
2. Інженерний аналіз.
3. Огляд і оцінка проектних рішень.

4. Автоматизоване виготовлення креслень.

Геометричне моделювання. Геометричне моделювання в рамках САПР пов'язано з отриманням зрозумілого машині математичного опису геометричних властивостей об'єкта. При наявності такого опису образ об'єкта, що проектується можна відтворити на екрані графічного терміналу, а з ним можна маніпулювати за допомогою різних сигналів, що йдуть від центрального процесора САПР. Програмні засоби, що забезпечують геометричне моделювання, повинні бути зручні для ефективного використання їх в обчислювальному процесі.

Для проведення геометричного моделювання розробник конструює графічне відображення потрібного об'єкта на екрані терміналу системи ІМГ, вводячи в машину команди трьох типів. Команди першого типу забезпечують формування базових геометричних елементів, таких, як точки, лінії і кола. По командах другого типу здійснюються масштабування, повороти зображення та інші перетворення базових елементів. За допомогою команд третього типу виробляється компоновка різних елементів у цілісне зображення проектованого об'єкта бажаної форми. В ході геометричного моделювання машина перетворює сигнали що надходять в компоненти математичної моделі, запам'ятовує потрібну інформацію у файлах даних і відображає одержувану модель проектованого об'єкта в наочній формі на екрані терміналу. Згодом ця модель може вилучатись з машинних файлів в цілях проведення огляду, аналізу або зміни.

Відомо кілька різних методів представлення об'єкта при геометричному моделюванні. Основним є спосіб представлення об'єкта у каркасної формі, коли він відображається сукупністю з'єднувальних ліній. Каркасне геометричне моделювання існує у трьох видах залежно від конкретних можливостей використовуваної системи ІМГ:

- Двовимірне (типу 2D)-для плоских об'єктів.

- Двох-з-половиною-мірне (типу 21/2D) – дозволяє відтворювати на екрані тривимірні об'єкти, що не мають деталей з бічними стінками.
- Тривимірне (типу 3D)-дає можливість моделювати складні геометричні об'єкти у тривимірному відображенні.

Однак іноді навіть тривимірного каркасного уявлення проектованого об'єкта виявляється недостатньо для належного відображення складних форм. Тому існують різні методи, що розширюють можливості каркасного моделювання. Можливо, наприклад, відображення внутрішніх, невидимих зовні ребер об'єкта штриховими лініями або взагалі повне «стирання» прихованих ліній. В результаті зображення стало більш впорядкованим і більш наочним. В одних САПР видалення прихованих ліній відбувається автоматично, в інших користувач повинен сам вказувати лінії, що підлягають стиранню. Каркасна модель може набути ще більш естетичний вигляд, якщо скористатися при геометричному моделюванні засобами відображення поверхонь, що дозволяють створити у спостерігача відчуття монолітності представленого на екрані об'єкта. Однак при цьому в пам'яті ЕОМ модель як і раніше зберігається в каркасному відображенні.

Найбільш досконалий метод геометричного моделювання – це об'ємне уявлення монолітних тел. При використанні цього методу проектований об'єкт конструюється з монолітних геометричних тіл, званих графічними примітивами.

Ще одна можливість, наявна в ряді САПР, – це кольорова графіка. За допомогою кольору можна зробити зображення на екрані графічного дисплея набагато більш інформативним. Колір дозволяє, наприклад, виділяти окремі компоненти складальних вузлів, підкреслювати об'ємність або може використовуватися для цілого ряду інших цілей.

Інженерний аналіз. При виконанні майже будь-якого проекту технічного призначення потрібні в тій чи іншій формі процедури аналізу. Цей аналіз може включати розрахунки механічних напружень і зусиль, теплових процесів або навіть ґрунтуватися на рішенні диференціальних

рівнянь, що описують динамічну поведінку проектованого об'єкта. Необхідні процедури аналізу можуть бути автоматизовані за рахунок використання ЕОМ. Часто для цього буває необхідно, щоб група інженерного аналізу розробила спеціальні програми для вирішення конкретних задач проектування. У цілому ряді випадків для цього вдається використовувати універсальні програми інженерного аналізу, наявні в продажі у вигляді комерційних пакетів.

У готових до безпосереднього застосування САПР такі засоби або часто передбачаються у складі системного програмного забезпечення, або можуть включатися потім в бібліотеку програм і викликатися для використання в процесі роботи з кожною конкретною моделлю проектованого об'єкта. Ми тут зупинимось лише на двох характерних типах подібних програмних засобів: для вирішення задач аналізу властивостей мас і задач аналізу методом кінцевих елементів.

Задачі першого типу одержали в рамках САПР найбільшого поширення. Програмні засоби для вирішення цих задач дозволяють досліджувати такі властивості монолітних об'єктів, як площа поверхні, маса, об'єм, центр ваги і момент інерції. Стосовно до плоских поверхонь (або поперечним перетинам твердих тіл) відповідні обчислення охоплюють розрахунок периметра, площі та інерційних властивостей.

У деяких САПР є можливість автоматичного виділення вузлів і отримання мережевої структури для даного об'єкта. Користувач при цьому повинен лише задати параметри моделі на основі методу кінцевих елементів, і система самостійно зробить все потрібні обчислення.

Результат аналізу за методом кінцевих елементів часто найкраще відображається системою в графічній формі на екрані дисплея і легко сприймається користувачем завдяки наочності. Так, наприклад, при дослідженні механічних зусиль, які розвиваються в об'єкті, кінцевий результат може бути відображений на екрані у вигляді деформованої форми, поєднаної з зображенням ненавантаженого об'єкта. При цьому об'єкт до і

після деформації може відтворюватися різним кольором. Якщо отримані результати аналізу свідчать про небажані властивості поведінки проєктованого об'єкта, конструктор має можливість змінити його форму і повторити аналіз методом кінцевих елементів для переглянутої конструкції.

Огляд і оцінка проектних рішень. Перевірку точності проєктування можна легко виконати з використанням графічного терміналу. Напівавтоматичні стандартні програми визначення розмірів і допусків, які прив'язують розмірні характеристики до поверхонь, вказаних користувачем, дозволяють скоротити число помилок у визначенні розмірів. При цьому конструктор може отримувати зображення деталей, які його цікавлять, крупним планом і збільшувати відповідні зображення в цілях проведення більш ретельного аналізу.

Один з найбільш цікавих засобів оцінки проектних рішень, наявний в деяких САПР, це кінематичні моделі. Стандартні комерційні пакети кінематики забезпечують можливість динамічного відтворення руху простих проєктованих механізмів на зразок шарнірів і зчленованих ланок. Наявність таких засобів аналізу розширює можливості конструктора в частині візуального спостереження за роботою потрібного механізму і допомагає гарантувати відсутність зіткнень з іншими об'єктами. Якщо САПР не надає користувачеві графічних засобів відтворення кінематики механізмів, то конструктор найчастіше змушений вдаватися до моделей, побудованим з використанням шпильок і картону. Програми такого роду можуть бути надзвичайно корисні при конструюванні механізмів, призначених для виконання певних рухів або прикладання конкретних сил.

Автоматизоване виготовлення креслень. Автоматизоване креслення передбачає отримання виконаних на папері конструкторських креслень безпосередньо на основі інформації, що зберігається в базі даних САПР. У деяких автоматизованих конструкторських бюро на перших порах можливість автоматичного виготовлення креслень була визначальним чинником доцільності витрат на придбання САПР. Це не дивно, тому що

продуктивність САПР на зазначеній операції в порівнянні з креслярем зростає приблизно в п'ять разів.

Класифікація та кодування деталей. У доповнення до чотирьох описаних вище функцій САПР слід відзначити ще одну позитивну якість САПР: її база даних може використовуватися для розробки системи класифікації та кодування деталей (СКК). Сенс цієї системи полягає в тому, що подібні деталі групуються в класи, і властивість їх подібності відображається в схемі кодування. В результаті конструктори можуть використовувати систему класифікації та кодування в своїй роботі для відшукування вже існуючих конструкцій деталей замість проектування їх щоразу заново. Відомий і цілий ряд інших застосувань СКК в технологічних процесах.

1.4 Структура САПР

Система автоматизованого проектування являє собою організаційно-технічний комплекс, що складається з великого числа підсистем і компонентів. Підсистеми є основними структурними ланками САПР і розрізняються за призначенням і по відношенню до об'єкта проектування.

Існуючий вітчизняний і закордонний досвід в області автоматизації проектування свідчить про те, що розробка, впровадження та ефективне використання програмних комплексів, призначених для автоматизації процесу проектування, які реалізуються на базі сучасних ЕОМ, вимагають комплексного вирішення широкого спектру проблем: організаційних, технічних, математичних, програмних, лінгвістичних, інформаційних та ін. Вирішення цих проблем базується на відповідних видах забезпечення.

Складність розробок великих комплексів взаємопов'язаних програм полягає в тому, що ефективність вирішення кожної конкретної проблеми, як правило, визначається на завершальному етапі роботи, коли вся або більша частина системи починає функціонувати. Це зумовлює складність створення

високоєфективних програмних комплексів при первісній розробці. Як правило, система стає ефективною в ході порівняно тривалого процесу створення, випробувань, удосконалення та доведення.

Під проблемою синтезу структури САПР розуміються:

- визначення складу та взаємозв'язків елементів системи;
- вибір принципів організації взаємодії елементів;
- оптимальний розподіл функцій між людиною і ЕОМ;
- вибір організаційної ієрархії;
- вирішення питань організації інформаційного інтерфейсу між елементами системи.

Завдання синтезу структури САПР взаємопов'язані із задачами оптимізації функціонування системи.

Обмеженнями в процесі синтезу структури САПР є допустимі навантаження елементів (обсяги інформації), що переробляються в одиницю часу. Одним з таких елементів САПР є людина, яка, як показують дослідження, нерідко змушена приймати рішення в процесі роботи в системі зі швидкістю у багато разів більшою, ніж при традиційній роботі. Природно, він не може тривалий час витримувати таке навантаження.

Серед різноманітних систем управління (до яких, зокрема, належать і системи проектування) найбільш поширені системи з ієрархічною структурою. Їх характерними особливостями є:

- автономність окремих підсистем;
- управління підсистемами при неповній інформації;
- агрегація інформації при русі вгору по ієрархічних сходах;
- взаємовплив підсистем через наявності загальних обмежень.

Основи математичної теорії синтезу структури складних систем на даний час тільки закладаються. В якості критерію ефективності їх функціонування, як правило, приймається максимум інформації, отриманої в одиницю часу.

Як і будь-яка складна система, САПР складається з підсистем. Розрізняють підсистеми проектуючі і обслуговуючі.

Проекуючі підсистеми безпосередньо виконують проектні процедури. Прикладами проектуючих підсистем можуть служити підсистеми геометричного тривимірного моделювання механічних об'єктів, виготовлення конструкторської документації, схемотехнічного аналізу, проектування планування механообробного цеху.

Обслуговуючі підсистеми забезпечують функціонування проектуючих підсистем, їх сукупність часто називають системною середовищем (або оболонкою) САПР. Типовими обслуговуючими підсистемами є підсистеми управління проектними даними (PDM – Product Data Management), управління процесом проектування (DesPM – Design Process Management), користувальницького інтерфейсу для зв'язку розробників з ЕОМ, CASE (Computer Aided Software Engineering) для розробки і супроводу програмного забезпечення САПР, повчальні підсистеми для освоєння користувачами технологій, реалізованих в САПР.

1.5 Особливості систем автоматизованого проектування

Важливою специфікою САПР є їх комплексний характер, тобто в них здійснюються і складні обчислення, і обробка великого обсягу даних. Для САПР необхідні потужні засоби обробки геометричної інформації та експертних знань.

Для великих САПР ЕОМ характерне використання в ЦВК (центрального обчислювального комплексу) обчислювальних машин, що відносяться до найбільш продуктивних з наявних на період створення системи. Це пояснюється тим, що ряд задач проектування для свого вирішення вимагають виконання обчислень такого обсягу, який значно перевищує можливості наявних ЕОМ. Але для дрібних розробників САПР придбання аналогічних машин є непомірно дорогим. Розглянемо більш детально комплекс технічних і програмних засобів САПР.

ТЗ САПР включає в себе обчислювальний комплекс (ОК) на базі високопродуктивної обчислювальної техніки з великим обсягом оперативної і зовнішньої пам'яті, широким набором периферійних пристроїв для забезпечення діалогового режиму роботи, випуску текстової та креслярської документації та створення повноцінних баз даних [2].

Основні вимоги, що пред'являються до технічних засобів САПР:

1. Ефективність – комплекс технічних засобів (КТЗ) повинен забезпечувати ефективне виконання всієї сукупності функцій АП з метою отримання достатньо якісних рішень і проектної документації в прийнятні терміни.

2. Універсальність – забезпечення виконання всього процесу проектування без перебудови КТЗ.

3. Сумісність – засоби, які входять до КТЗ САПР, повинні бути технічно, інформаційно, програмно і експлуатаційно сумісні.

4. Гнучкість і відкритість – тобто допускати перебудову КТЗ в досить широких межах і дозволяти заміну застарілих засобів, модернізацію та розширення складу.

5. Надійність – необхідна вимога для нормального функціонування на протязі всього циклу проектування.

6. Точність (достовірність) – КТЗ повинен забезпечити потрібний рівень точності (достовірності) прийнятих рішень і даних. Точність залежить від точності ТЗ (методів округлення, розрядності), збоїв в устаткуванні, захищеності від зовнішніх впливів.

7. Захищеність – КТЗ САПР повинен бути захищений від зовнішніх впливів (перешкод, збоїв в системі живлення, некомпетентного і несанкціонованого втручання).

8. Можливість одночасної роботи достатньо широкого кола користувачів – КТЗ повинен дозволяти реалізувати САПР, яка є системою колективного користування для достатньо великої кількості фахівців (розробників САПР, проектувальників та ін.

9. Прийнятна вартість – вартість КТЗ повинна бути такою, щоб створена на його базі САПР забезпечила прийнятний економічний ефект.

Технічні засоби в САПР вирішують завдання:

1. Введення вихідних даних опису об'єкту проектування;
2. Відображення введеної інформації з метою її контролю і редагування;
3. Перетворення інформації (зміни форми і структури представлення даних, перекодування та ін);
4. Зберігання інформації;
5. Відображення підсумкових і проміжних результатів вирішення;
6. Оперативного спілкування проектувальника з системою в процесі вирішення завдань.

До теперішнього часу склався стійко зростаючий ринок збуту апаратних і програмних засобів в області САПР, який виробив власні вимоги до базових ЕОМ, периферійних пристроїв і ЛВМ. У якості еталонних базових ЕОМ, що знаходяться безпосередньо на столах проектувальників РЕЗ, виступають протягом вже тривалого часу робочі станції (РС), пов'язані ЛВМ між собою та іншими ЕОМ.

Для визначення структури КТЗ і параметрів компонентів, які входять до нього, можуть служити обмеження: знизу – на число програм N , що входять до складу програмного забезпечення САПР; зверху – на середній час T реакції КТЗ на задачу проектування, яка надійшла; знизу – на обсяг оперативної пам'яті для зберігання програм проектування; зверху – на час, необхідний процесору для вирішення усередненої задачі в однопрограмних режимі, а також за номенклатурою периферійного обладнання КТЗ САПР.

Таким чином, завданням проектування технічного забезпечення є вибір оптимального складу технічних засобів САПР. Вихідною інформацією при цьому є результати аналізу завдань внутрішнього проектування та ресурсні вимоги до технічних засобів у вигляді критеріїв і обмежень.

Інформаційне забезпечення САПР

У комплекс засобів автоматизованого проектування входить інформаційне забезпечення, яке являє собою сукупність документів, що описують стандартні проектні процедури, типові проектні рішення, типові елементи і комплектуючі вироби, матеріали та інші дані, а також файли і блоки даних на машинних носіях із записом вказаних документів. Головною метою створення інформаційного забезпечення САПР є розробка інформаційної системи, що дозволяє правильно і швидко вирішувати проектні завдання.

Це може бути досягнуто своєчасною видачею джерелу запиту повної і достовірної інформації для виконання певної частини проектно-конструкторського процесу.

Основні вимоги до інформаційного забезпечення САПР наступні:

1. Наявність необхідної інформації для забезпечення як автоматизованих, так і ручних процесів проектування.
2. Можливість зберігання та пошуку інформації, що представляє результат ручних та автоматизованих процесів проектування.
3. Достатній обсяг сховищ інформації. Структура системи повинна допускати можливість нарощування ємності пам'яті разом із зростанням обсягу інформації, що підлягає зберіганню. Одночасно необхідно забезпечити компактність збереженої інформації і мінімальне зношування носіїв інформації.
4. Достатня швидкодія системи інформаційного забезпечення.
5. Можливість швидкого внесення змін і коригування інформації, доведення цих змін до споживача, а також отримання твердої копії документа.

При створенні інформаційного забезпечення САПР основна проблема полягає в перетворенні інформації, необхідної для виконання проектно-конструкторських робіт над певним класом об'єктів, у форму, прийнятну і найбільш раціональну для машинної обробки, і виведенні інформації на ЕОМ у вигляді, зручному для сприйняття людиною.

При створенні конкретних САПР виникає проблема вибору і оцінки автоматизованого банку даних. Правильність вибору автоматизованого банку даних багато в чому визначає ефективність створення і зручність функціонування САПР. Таким чином, ця проблема стосується не лише розробників, але і користувачів САПР.

В даний час інформаційне забезпечення САПР будується, як правило, за принципами, виробленим і апробованим при використанні автоматизованих банків даних в інформаційних системах загального призначення. Автоматизований банк даних відноситься до інваріантних засобів САПР. Тому творці САПР зазвичай не розробляють спеціальні автоматизовані банки даних, а вибирають з готових банків загального призначення і адаптують їх до цілей конкретної САПР.

Вибір СУБД для САПР, що розроблюються, повинен бути спрямований головним чином на задоволення вимог і потреб користувачів-проектувальників.

Фактори, що визначають СУБД, розбиваються на дві групи:

- Фактори, що характеризують передбачуване застосування СУБД;
- Фактори, що характеризують функціональні можливості СУБД.

У характеристику застосування зазвичай включають:

- Класифікацію і характеристику кінцевих користувачів і обслуговуючого персоналу СУБД (системні аналітики, адміністратори автоматизованого банку даних, системні та прикладні програмісти);
- Складність і форму запитів;
- Режим роботи (багатокористувацький, пакетний, інтерактивний і т. д.);
- Еволюційні характеристики (характеристики користувачів до і після впровадження СУБД, зростання оперативності користувачів, поява нових завдань);
- Характеристики даних;

- Характеристики ресурсів (конфігурація ЕОМ, програмне забезпечення, кваліфікацію обслуговуючого персоналу).

Функціональні можливості можна характеризувати наступними факторами:

- Призначення СУБД (узагальнена або спеціалізована на певне коло завдань);
- Підтримувані структури даних (ієрархічні, мережеві, реляційні);
- Обладнання, що використовується (продуктивність процесора, обсяг головної пам'яті, типи та ємність зовнішньої пам'яті);
- Програмне забезпечення, що використовується;
- Труднощі освоєння та експлуатації, наявність документації.

Вибір конкретного автоматизованого банку даних для САПР ускладнений тим, що його ефективність залежить від багатьох факторів. І важливо не тільки знати значення цих факторів для різних автоматизованих банків даних, але і вибирати найбільш значущі з них.

При розробці концепції САПР слід враховувати не тільки позитивні сторони автоматизованого банку даних, але і "платню" за реалізацію можливостей, що надаються їм. До головних компонентів "платні" за використання автоматизованого банку даних в САПР слід віднести:

- Витрати на управління автоматизованим банком даних: потрібні значні ресурси (пам'ять, машинний час) для функціонування СУБД і операційної системи ЕОМ;
- Труднощі дотримання розробниками та користувачами прикладних програм тих обмежень, які покладені в основу СУБД;
- Необхідність створення спеціального адміністративного підрозділу, відповідального за експлуатацію автоматизованого банку даних (адміністратор бази даних, адміністратор з управління системою і т. д.);
- Загроза безпеці даних, пов'язана з великою їх централізацією: несправності в технічних засобах і помилки в прикладних програмах можуть зіпсувати дані для багатьох програм і багатьох користувачів.

Програмне забезпечення САПР

До програмного забезпечення висуваються такі вимоги:

1. Економічність – оцінюється витратами обчислювальних ресурсів – машинного часу і оперативної пам'яті. Характер їх залежності від розмірності задачі визначається в першу чергу властивостями математичного забезпечення. Однак невдала програмна реалізація може істотно збільшити машинний час і об'єм оперативної пам'яті, що потребуються. Недостатня економічність ПО зазвичай виявляється основним фактором, що обмежує можливості вичерпного аналізу, оптимізації та структурного синтезу проєктованих об'єктів.

2. Зручність використання ПЗ визначається його надійністю, наявністю проблемно-орієнтованих вхідних мов і засобів діагностики помилок користувача.

3. Надійність ПЗ – властивість виконувати задані функції в заданих умовах. Функції та умови формулюються в термінах тієї предметної області, до якої відносяться проєктовані об'єкти. Основний показник надійності – ймовірність отримання правильного результату при використанні програми у сформульованих умовах.

4. «Правильність» ПЗ – властивість, що характеризує відповідність ПЗ специфікаціям математичного характеру, тобто правильність реалізації в ПЗ обраного математичного забезпечення (МЗ). Невідповідність вимог користувачів і обраного математичного забезпечення знижує надійність, але не впливає на правильність. Наприклад, якщо в програмі аналізу статичних станів деякого класу об'єктів безпомилково реалізований метод Ньютона, то програма буде правильною, але, мабуть, ненадійною, оскільки умови збіжності методу Ньютона будуть виконуватися не для будь-якого об'єкта із заданого класу при довільному завданні початкового наближення.

5. Універсальність ПЗ характеризується обмеженнями на застосування ПЗ. Ці обмеження можуть відноситися до типів та елементного складу аналізованих або синтезованих структур, діапазонам числових значень

внутрішніх і зовнішніх параметрів, переліком проектних операцій і процедур, що виконуються. Універсальність пов'язана з надійністю ПЗ – чим ретельніше і повніше виявлені і застережені обмеження, тим нижче ступінь універсальності програми, але вище її надійність. В САПР необхідно прагнути до досягнення високої надійності ПЗ. Тому експлуатація декількох вузькоспеціалізованих, але надійних програм переважніше застосування однієї універсальної програми, якщо за підвищення ступеня універсальності доводиться платити зниженням надійності.

6. Відкритість ПО характеризується можливостями внесення до нього змін в процесі експлуатації. Поняття відкритості близьке до поняття адаптованості, під яким мається на увазі можливість модифікації ПЗ для підтримки його працездатності і ефективності в умовах застосування, що змінюються.

7. Супроводжуємость ПЗ – властивість, близька властивості відкритості, характеризує зручність підтримки ПЗ в працездатному стані і забезпечується структурованістю ПЗ і наявністю необхідної експлуатаційної документації.

8. Мобільність ПЗ, яка зветься також переносимістю, визначається легкістю перебудови ПЗ, що експлуатувалося на ЕОМ з однією системою команд на ЕОМ з іншою системою команд. Програми, записані на машинно-орієнтованих мовах, не переносимі. Використання мов високого рівня створює передумови для створення мобільних програм. Однак для підвищення мобільності необхідні додаткові заходи по фіксації і виділенню в змінювані блоки елементів ПЗ, що відображають специфіку архітектури ЕОМ та пов'язаних з ними ОС.

Склад ПЗ САПР

Програмне забезпечення включає компоненти *загального і спеціального призначення*. Загальне ПЗ не відображає специфіки конкретної САПР і включає в себе компоненти, що забезпечують організацію і контроль обчислювального процесу, автоматизацію трудомістких етапів підготовки і

налагодження прикладних програм. Компоненти загального ПЗ не розробляються при створенні конкретної САПР [3].

ПЗ САПР загального призначення включає програми, що здійснюють управління, контроль та планування обчислювального процесу, розподіл ресурсів, введення/виведення даних та інші операції в підсистемах САПР і, в свою чергу, поділяється на *загальносистемне* та *базове*.

Базове ПЗ характеризується тим, що його компоненти відображають специфіку САПР даного класу, однак не розробляються при їх створенні, а купуються як комплектуючі вироби нарівні з компонентами загальносистемного ПЗ та призначені для використання багатьма проектними організаціями. Типовими прикладами базового ПЗ є ПЗ обслуговуючих підсистем САПР – графічних і текстових редакторів, СУБД, діалогових моніторів і т.п.

Загальносистемне ПЗ представлено операційними системами і характеризується тим, що його компоненти не відображають специфіки конкретної САПР і є промислово супроводжуваними програмно-методичними комплексами.

До функцій загальносистемного ПЗ відносяться:

1. управління процесом обчислень;
2. введення, виведення і частково обробка інформації;
3. діалоговий взаємозв'язок з користувачем в процесі проектування;
4. рішення загальноматематичних задач;
5. зберігання, пошук, сортування, модифікація даних, необхідних при проектуванні, захист їх цілісності та захист від несанкціонованого доступу;
6. контроль і діагностика роботи обчислювального комплексу.

Три перші і остання із зазначених функцій реалізуються в сучасних обчислювальних комплексах на базі операційних систем, для вирішення загальноматематичних задач до складу загальносистемного ПЗ включають відповідні бібліотеки стандартних програм. Для зберігання і використання різних даних створюються спеціальні системи управління базами даних.

Архітектура ПЗ САПР

На рисунку 1 розглянуто один з можливих варіантів структури ПЗ САПР.

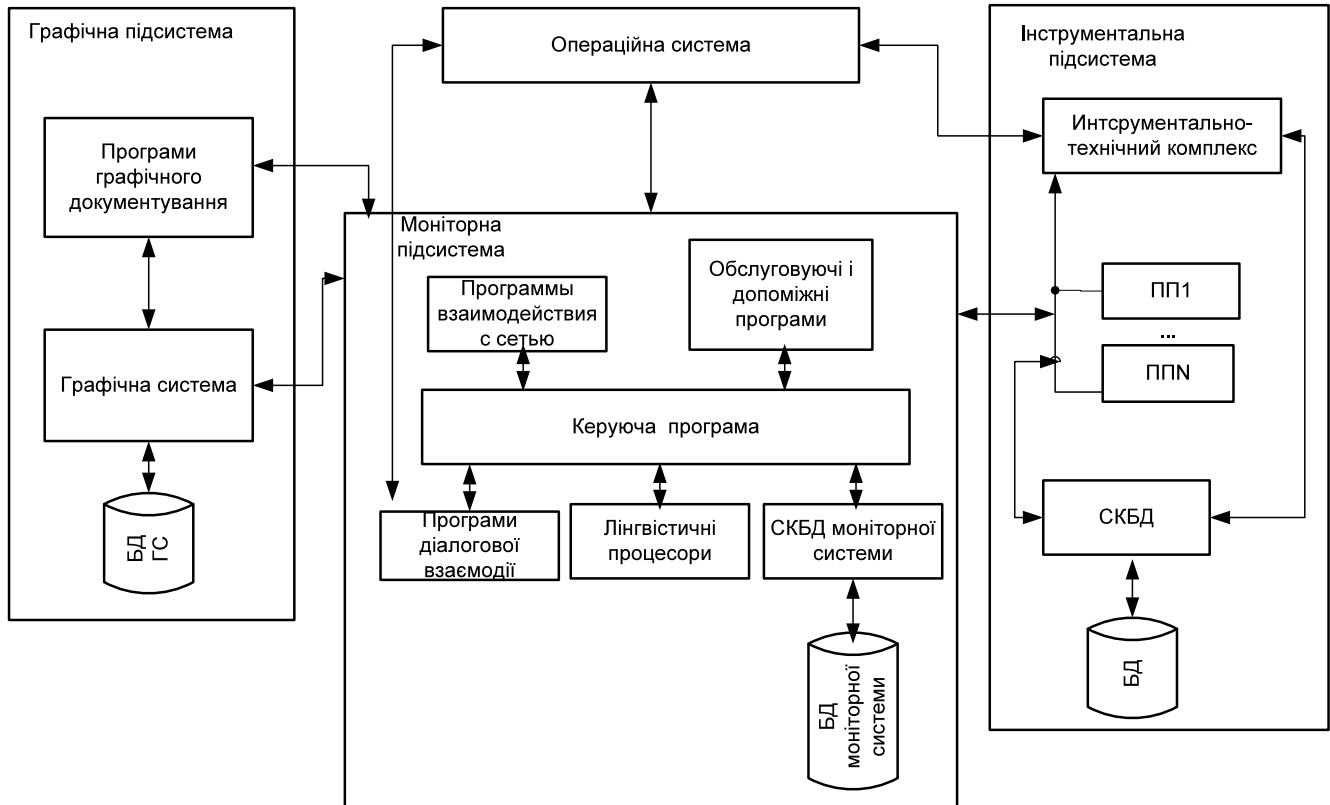


Рисунок 1.1 – Структура програмного забезпечення САПР

До загальносистемного ПЗ САПР відносяться наступні компоненти:

- Операційна система універсальної або спеціалізованої ЕОМ;
- Моніторна система (МС);
- Графічна система (ГС) (графічний і геометричний процесори);
- СКБД, що підтримують бази даних (БД) нормативно-довідкової інформації (НДІ) та архіву проектних рішень, а також, можливо, СКБД деяких ППП, що реалізують проектуючі підсистеми;
- Телемонітор і (або) ПЗ мереж ЕОМ;
- Програми формування графічної документації;
- Інструментальний технологічний комплекс для розробки програмного забезпечення.

Крім того, до числа загальносистемних компонентів ПЗ САПР часто відносять програмні засоби моделювання і оптимізації, загальнотехнічних розрахунків (бібліотеки підпрограм для науково-технічних розрахунків).

Для ППП, що вирішують завдання проектування, які вимагають високої реактивності, можуть застосовуватися універсальні СУБД.

Функції взаємодії з користувачем реалізують такі компоненти, як:

- Графічна система;
- Програми діалогової взаємодії і лінгвістичний процесор, що забезпечують алфавітно-цифровий діалог та інтерпретацію проблемно-орієнтованої мови проектування;
- Програми документування проектних рішень.

Функції взаємодії САПР з іншими автоматизованими системами, а також з іншими обчислювальними системами реалізують мережеві програми МС. В інших варіантах реалізації ПЗ САПР цю функцію можуть виконувати або ОС, або спеціальний телемонітор (мережеве ПЗ), яке має форму ППП, що розширює функції операційної системи.

При створенні конкретної САПР перераховані компоненти повинні розглядатися як типові проектні рішення і купуватися на договірних умовах як продукція виробничо-технічного призначення. Однак при створенні конкретної САПР не завжди можна вибрати відповідний компонент кожного типу, тому доводиться розробляти оригінальні моніторну систему, програми документування, спеціалізовану СУБД. Найбільша свобода вибору компонентів мається для ОС, СУБД і телемоніторів, найменша – для графічних систем, інструментально-технологічних комплексів, засобів моделювання та оптимізації.

1.6 Функції, характеристики і приклади CAD/CAM/CAE-систем

Функції CAD-систем у машинобудуванні підрозділяють на функції двовимірного (2D) і тривимірного (3D) проектування. До функцій 2D відносяться креслення, оформлення конструкторської документації; до

функцій 3D – отримання тривимірних моделей, метричні розрахунки, реалістична візуалізація, взаємне перетворення 2D і 3D моделей [4].

Серед CAD-систем розрізняють «легкі» та «важкі» системи. Перші з них орієнтовані переважно на 2D графіку, порівняно дешеві і менш вимогливі у відношенні обчислювальних ресурсів. Другі орієнтовані на геометричне моделювання (3D), більш універсальні, коштовні, оформлення креслярської документації в них зазвичай здійснюється за допомогою попередньої розробки тривимірних геометричних моделей.

Основні функції САМ-систем:

- розробка технологічних процесів,
- синтез керуючих програм для технологічного обладнання з числовим програмним управлінням (ЧПУ),
- моделювання процесів обробки, в тому числі побудова траєкторій відносного руху інструменту і заготовки в процесі обробки,
- генерація постпроцесорів для конкретних типів обладнання з ЧПК (NC – Numerical Control),
- розрахунок норм часу обробки.

Найбільш відомі наступні CAE/CAD/CAM-системи в машинобудуванні.

«Важкі» системи (в дужках вказана фірма, що розробила або поширює продукт):

- Unigraphics (EDS Unigraphics);
- Solid Edge (Intergraph);
- Pro/Engineer (PTC – Parametric Technology Corp.),
- CATIA (Dassault Systemes),
- EUCLID (Matra Datavision),
- CADD5.5 (Computervision, нині входить до PTC) та ін.

«Легкі» системи:

- AutoCAD (Autodesk);
- Адема; bCAD;

- Caddy (Ziegler Informatics);
- КОМПАС (Аскон, С.Петербург);
- СПРУТ (Sprut Technology, Москва-Набережні Челни);
- Кредо (НИВЦ АСК, Москва).

Системи, що займають проміжне положення (середньомасштабні):

- Cimatron, Microstation (Bentley),
- Euclid Prelude (Matra Datavision),
- T-FlexCAD (Топ Системи, Москва) та ін.

З ростом можливостей персональних ЕОМ грані між «важкими» і «легкими» CAD/CAM-системами поступово стираються.

Функції CAE-систем досить різноманітні, тому що пов'язані з проектними процедурами аналізу, моделювання, оптимізації проектних рішень. До складу машинобудівних CAE-систем насамперед включають програми для наступних процедур:

- моделювання полів фізичних величин, у тому числі аналіз міцності, який найчастіше виконується відповідно до методу кінцевих елементів (МКЕ);
- розрахунок станів і перехідних процесів на макрорівні;
- імітаційне моделювання складних виробничих систем на основі моделей масового обслуговування і мереж Петрі.

Приклади систем моделювання полів фізичних величин відповідно до МКЕ:

Nastran, Ansys, Cosmos, Nisa, Moldflow.

Приклади систем моделювання динамічних процесів на макрорівні: Adams і Dyna – в механічних системах, Spice – в електронних схемах, ПА9 – для багатоаспектного моделювання, тобто для моделювання систем, принципи дії яких засновані на взаємовпливі фізичних процесів різної природи.

Для зручності адаптації САПР до потреб конкретних застосунків, для її розвитку доцільно мати у складі САПР інструментальні засоби адаптації і

розвитку. Ці кошти представлені тією чи іншою CASE-технологією, включаючи мови розширення. У деяких САПР застосовують оригінальні інструментальні середовища.

Прикладами можуть служити об'єктно-орієнтоване інтерактивне середовище CAS.CADE в системі EUCLID, що містить бібліотеку компонентів, в САПР T-Flex CAD 3D передбачена розробка доповнень в середовищах Visual C++ і Visual Basic.

Висновок до розділу 1

Важливе значення для забезпечення відкритості САПР, її інтегрованості з іншими автоматизованими системами (АС) мають інтерфейси, що представляються реалізованими в системі форматами міжпрограмних обмінів. Очевидно, що, в першу чергу, необхідно забезпечити зв'язки між САЕ, CAD і САМ-підсистемами.

В якості мов – форматів міжпрограмних обмінів – використовуються IGES, DXF, Express (стандарт ISO 10303-11, входить в сукупність стандартів STEP), SAT (формат ядра ACIS) та ін.

Найбільш перспективними вважаються діалекти мови Express, що пояснюється загальним характером стандартів STEP, їх спрямованістю на різні додатки, а також на використання в сучасних розподілених проектних і виробничих системах. Дійсно, такі формати, як IGES або DXF, описують тільки геометрію об'єктів, у той час як в обмінах між різними САПР і їх підсистемами фігурують дані про різні властивості і атрибути виробів.

2 ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ХМАРНИХ ОБЧИСЛЕНЬ

Ідеологія «хмарних обчислень» за оцінками GartnerGroup отримала популярність в 2007 році завдяки швидкому прогресуванню обчислювальної потужності обладнання. Також цьому сприяла поява технологій віртуалізації і зростаюча в геометричній прогресії потреба організацій у горизонтальному масштабуванні своїх інформаційних систем.

Сфера «хмарних обчислень» на поточний момент є такою, що швидко розвивається і перспективною, оскільки традиційна ІТ-інфраструктура, побудована за принципом 1 сервер – 1 сервіс, має ряд відомих недоліків, таких як неефективне використання обчислювальних ресурсів і площ, не оптимальні витрати на електропостачання, вентиляцію, обслуговування і адміністрування та ін, які не раз вже були описані в ІТ-виданнях.

Традиційний підхід, з точки зору сервісно-орієнтованої моделі архітектури, характеризується насамперед ускладненням корпоративної ІТ-інфраструктури. З розвитком бізнесу компанії повстає питання про введення в дію нових ІТ-сервісів, що надаються користувачам як всередині організації, так і поза неї (клієнтам). Це як правило, призводить до появи розрізнених логічно і фізично, малопов'язаних між собою «острівців» ІТ-інфраструктури.

Звідси випливають проблеми організаційно-технічного характеру: від

Розуміючи основні мінуси традиційного підходу, одним з можливих варіантів побудови нової або оптимізації існуючої ІТ-інфраструктури може стати перехід до сервісної моделі послуг на основі концепції «хмарних обчислень» – Cloud Computing.

2.1 Основні положення хмарних обчислень

За визначенням Національного інституту стандартів і технологій США (NIST), хмарні обчислення (Cloud Computing) – це модель, що забезпечує зручний мережевий доступ на вимогу до загальних конфігурованих обчислювальних ресурсів (мереж, серверів, сховищ даних, застосунків і сервісів), якій оперативно надається з мінімальними зусиллями з управління

та взаємодії з сервіс-провайдером. Визначення хмарних обчислень описує шість основних характеристик, три сервісні моделі і чотири моделі розгортання [5].

Основні характеристики:

1. Самообслуговування на вимогу – користувачі здатні отримувати, контролювати і управляти обчислювальними ресурсами без допомоги системних адміністраторів.
2. Широкий мережевий доступ – обчислювальні сервіси надаються через стандартні мережі і гетерогенні пристрої.
3. Оперативна еластичність – ІТ-ресурси можуть оперативно масштабуватися в будь-яку сторону у міру потреби.
4. Пул ресурсів – ІТ-ресурси спільно використовуються різними програмами та користувачами в незв'язаному режимі.
5. Розрахунок вартості послуги – використання ІТ-ресурсу відстежується по кожному застосунку і користувачу, як правило, щоб забезпечити білінг по публічній хмарі і внутрішні розрахунки за приватну хмару.
6. Мультитенантність – можливість ізолювано обслуговувати користувачів з різних організацій (тобто незалежних передплатників SaaS) в рамках одного сервісу (однієї інсталяції або розгортання).

Три сервісних моделі: ПЗ як послуга (SaaS), платформа як послуга (PaaS), інфраструктура як послуга (IaaS).

Чотири моделі розгортання

1. Приватні хмари (private cloud) – використовується для надання сервісів усередині однієї компанії, яка є одночасно і замовником і постачальником послуг. Це варіант реалізації "хмарної концепції", коли компанія створює її для себе самої, в рамках організації. В першу чергу, реалізація private cloud знімає одне з важливих питань, яке неодмінно виникає у замовників при ознайомленні з цією концепцією – питання про захист даних з точки зору інформаційної безпеки. Оскільки "хмара"

обмежена рамками самої компанії, це питання вирішується стандартними існуючими методами. Для private cloud характерно зниження вартості обладнання за рахунок використання незадіяних або неефективно використовуваних ресурсів. А також, зниження витрат на закупівлі обладнання за рахунок скорочення логістики (не думаємо, які сервера закуповувати, в яких конфігураціях, які продуктивні потужності, скільки місця кожного разу резервувати і т.д.)

2. Публічні хмари – надають широкий доступ до обчислювальних ресурсів через Інтернет. Постачальник публічної хмари дає споживачам доступ до ресурсів на основі автоматичного розгортання систем, в основному через веб-інтерфейс. Клієнт орендує доступ до ресурсів у міру необхідності, оплачуючи тільки ті ресурси, які використовує. Публічні хмари пропонують доступ до великих наборів масштабованих ресурсів на тимчасовій основі без необхідності фінансових вкладень в інфраструктуру центру обробки даних. Всі користувачі даних хмар не мають можливості управляти і обслуговувати дану хмара, вся відповідальність з цих питань покладено на власника даної хмари.

3. Групові хмари – використовуються групою споріднених організацій, охочих скористатися загальним хмарним обчислювальним середовищем.

4. Гібридні хмари – з'являються, коли організація використовує і приватне, і публічне хмари для одного і того ж застосунка, щоб скористатися перевагами обох.

Одна з ключових ідей Cloud полягає якраз у тому, щоб з технологічної точки зору різниці між внутрішніми та зовнішніми хмарами не було і замовник міг гнучко переміщати свої завдання між власною і орендованою IT-інфраструктурою, не замислюючись, де конкретно вони виконуються.

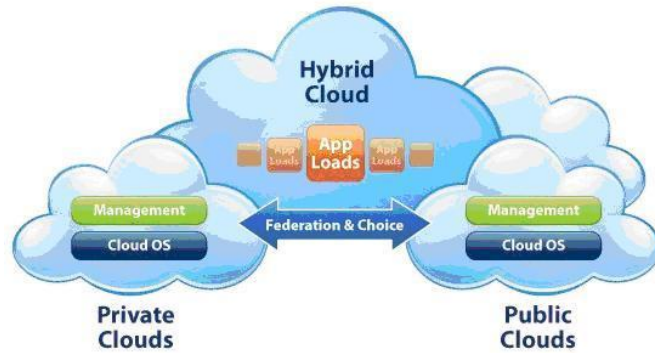


Рисунок 2.1– Взаємозв'язок хмар різних типів

Економічна ефективність хмарних обчислень у порівнянні з локально використовуваним апаратним і програмним забезпеченням полягає в істотному скороченні витрат на апаратно-програмну інфраструктуру і тісно пов'язана з технологічними перевагами, такими як високий рівень віртуалізації і масштабованості. Cloud Computing володіє наступними властивостями, які складають економічну ефективність цієї моделі:

- Забезпечує більш швидку окупність у порівнянні з традиційними інформаційними технологіями;
- Не вимагає попередніх капітальних витрат;
- Мінімізує операційні витрати;
- Вимагає менше технічних ресурсів;
- Надає більш простий рівень інтеграції

2.2 Види хмарних обчислень

Хмарні обчислення поділяються на кілька основних категорій за типами послуг, що надаються.

Програмне забезпечення як послуга (Software as a Service, скорочено SaaS) – бізнес-модель продажу програмного забезпечення, при якій власник (постачальник) ПЗ надає доступ до нього користувачам (замовникам) через Інтернет. Прикладами такого ПЗ є Feng Office Community Edition, Simple Groupware, Zarafa та ін.

Обладнання (обчислювальні потужності) як послуга (Hardware as a Service, скор. НаaS) – надання обчислювальних ресурсів обладнання (його процесорного часу, місця для зберігання даних і т.д.) у вигляді сервісів з використанням технологій віртуалізації. Сервіси зазвичай пропонуються як еквівалент реальним обчислювальним системам, таким як сервери, суперкомп'ютери та ін. Над програмною реалізацією цієї ідеї повністю або частково працюють проекти OpenVZ, FreeVPS, Linux-VServer, Apache Hama, GlusterFS Open Source Project, а також Moose File System (MooseFS) та ін, а надає такий сервіс на базі OpenSource рішень компанія Linode і деякі інші.

Комунікація як Сервіс (Communications as a Service, скор. СaaS) – побудоване в хмарі комунікаційне рішення для підприємства, яке забезпечує передачу мовного сигналу по мережі Інтернет або по будь-яким іншим IP-мережам (VoIP), обмін миттєвими повідомленнями (IM), відеоконференції. Модель СaaS дозволяє діловим клієнтам вибірково розгортати засоби комунікацій і послуг на підставі оплати послуг в строк для сервісів, що використовуються. З цим напрямом тісно пов'язані такі FOSS-проекти як Ekiga, iLBC, Speex.

Моніторинг як Сервіс (Monitoring-as-a-Service, скор. МаaS) є програмним забезпеченням, яке обслуговується в хмарі для моніторингу та забезпечення безпеки. Такими OpenSource рішеннями на сьогоднішній день є Ganglia, Zabbix, Nuperc HQ. Сюди ж з деякими застереженнями можна віднести і Nagios.

Інфраструктура як послуга (Infrastructure as a Service, скор. IaaS) – це надання комп'ютерної інфраструктури (як правило у формі віртуалізації) як послуги на основі концепції хмарних обчислень. По суті IaaS є комбінацією SaaS, НаaS, тому що вона включає до себе і те й інше, причому зазвичай у множині, а також СaaS і іноді МаaS з метою об'єднання і моніторингу всієї системи, і, тому, використовується в основному підприємствами. Вільними реалізаторами даної концепції є Eucalyptus, OpenNebula, OpenStack, Nimbus та ін.

Платформа як послуга (Platform as a Service, скор. PaaS) – надання програмної платформи і інструментів з певними характеристиками, необхідних для розробки, тестування, розгортання, підтримки різних застосунків. Сюди ж входять і готові до використання хмарні сервіси, які разом утворюють програмну платформу. Яскравими прикладами зі світу OpenSource в даний час є Xen Cloud Platform, Cloud Foundry, Apache Hadoop, Apache Hive та ін.

Комп'ютер (віртуальний робочий стіл) як послуга (Desktop as a Service, скор. DaaS) – надання віртуального комп'ютера, який кожен користувач може індивідуально налаштовувати під свої завдання. Таким чином, користувач, приходячи на роботу, просто вводить свої дані (зазвичай логін і пароль) і може працювати, використовуючи при цьому завдяки технологіям віртуалізації обчислювальні потужності стороннього сервера, а не свого ПК.

Робоче оточення як послуга (Workspace as a Service, скор. WaaS) – надання комплекту SaaS, призначеного для створення робочого оточення. На відміну від DaaS, в цьому випадку користувач отримує доступ тільки до ПЗ, в той час як всі обчислення відбуваються безпосередньо на його машині. По суті дана категорія є гібридом SaaS і PaaS, тому що на відміну від останньої є платформою, спрямованою не на розробку і тестування ПЗ, а на офісну роботу, але при цьому як перша в реалізації не використовує технологій віртуалізації. На даний момент реалізації даної технології надаються в основному різними великими компаніями, наприклад Google і Microsoft, і представляють в основному рішення з закритим вихідним кодом, іноді з використанням вільних і відкритих компонентів або їх первинників.

Все як послуга (Everything as a service, скор. EaaS) – концептуальна модель, що включає в себе елементи всіх перерахованих рішень. На даний момент повної її реалізації не існує – вона по суті є ідеалом для великих хмарних компаній.

2.3 Моделі хмарних сервісів

На думку дослідницької групи Університету Каліфорнії в Берклі, хмарні обчислення являють собою поєднання підходів Software as a Service (SaaS) і загальнодоступних обчислень (utility computing), причому до останньої категорії відносяться рішення, зазвичай класифікуються як Infrastructure as a Service (IaaS) і Platform as a Service (PaaS) [6]. Якщо SaaS-рішення орієнтовані на кінцевих користувачів, то пропозиції utility computing можуть бути затребувані і тими користувачами, які самі є постачальниками хмарних SaaS-послуг. Таким чином, роль одного і того ж учасника на "хмарному" ринку може бути неоднозначна: з одного боку, він є споживачем IaaS і PaaS-рішень, з іншого – постачальником SaaS (рис 2.2).



Рисунок 2.2 – Дві моделі хмарних сервісів

У таблиці показано розподіл зон відповідальності між ІТ-постачальником і ІТ-користувачем для даних моделей.

Таблиця 2.1 – Відповідальність провайдера і споживача у різних моделях споживання «хмарних» послуг.

☒ - відповідальність власника застосунку ☐ - відповідальність постачальника сервісів	Локально	IaaS	PaaS	SaaS
Застосунки	☒	☒	☒	☐
Дані	☒	☒	☐	☐
Середовища виконання	☒	☒	☐	☐
Бази даних	☒	☒	☐	☐
Операційна система	☒	☒	☐	☐
Віртуалізація	☒	☐	☐	☐
Сервер	☒	☐	☐	☐
Сховище	☒	☐	☐	☐
Мережа	☒	☐	☐	☐

2.4 Економічні переваги хмарних обчислень у порівнянні з традиційними підходами розгортання додатків.

Про економічні переваги хмарних обчислень найзручніше говорити, коли мова йде про створення систем «з нуля»: у цьому випадку не потрібно турбуватися про необхідність доопрацювання застосунків, про їх сумісність з успадкованими системами, про перенавчання співробітників, утилізацію старого обладнання та ін. Але в дійсності дуже мало компаній володіє такою свободою, і впровадження хмарних обчислень пов'язане не тільки з економічними вигодами, але і з додатковими витратами. У представленій нижче таблиці розглянуто чотири ймовірних сценарії впровадження хмарних технологій [6].

Таблиця 2.2 – Разові і періодичні витрати при різних підходах до впровадження хмарних обчислень

Види витрат	Розробка нового застосунку по традиційній моделі	Розробка нового застосунку в «хмарах»	Повна міграція застосунку до «хмари»	Продовження експлуатації застосунку, створеного по традиційній моделі
Разові витрати				
Устаткування	високі	немає	немає	немає
Розробка / удосконалення додатка	високі	високі	середні	немає
Утилізація	немає	немає	середні	немає

Навчання персоналу	низькі-середні	середні	середні	немає
Періодичні витрати				
Оренда хмарних сервісів	немає	середні	середні	немає
Технічна підтримка	немає	середні	середні	немає
Зарплата співробітникам	високі	середні	середні	високі
Оренда приміщення та інфраструктури	високі	немає	немає	високі

У першому випадку застосунок розробляється з нуля по традиційній моделі. У другому – розробка спочатку ведеться на основі хмарних технологій. У третьому випадку компанія вже має готовий застосунок і переводить його цілком у «хмари». Нарешті, в останньому – йдеться про продовження використання існуючого «нехмарного» застосунку без доопрацювання. У кожному випадку доводиться мати справу з разовими і періодичними витратами: перші мають місце тільки на етапі розробки і впровадження застосунка, а другі доводиться нести регулярно, поки додаток не буде знято з експлуатації. З наведеної таблиці видно, що при розробці нових застосунків традиційна модель менш приваблива, ніж хмарна. Якщо говорити про разові витрати, то вона вимагає високих витрат не тільки на розробку ПЗ, але і на закупівлю обладнання. При цьому в довгостроковій перспективі доведеться нести значні витрати, пов'язані з використанням власних обчислювальних потужностей: потрібно платити за оренду приміщення, оплачувати рахунки за електрику та Інтернет і видавати зарплату співробітникам. Спочатку хмарне впровадження, з одного боку, дозволяє повністю відмовитися від витрат, пов'язаних із закупівлею та подальшою експлуатацією власного обладнання, а з іншого – дозволяє заощадити за рахунок зарплати тих співробітників, які переважно відповідали б за роботу обладнання, а не застосунків. Однак, як було сказано вище, в реальній практиці частіше мова йде не про нові застосунки, а про вже

існуючі системи. Тут хмарна модель дозволяє позбутися періодичних витрат, пов'язаних з підтримкою власного обладнання, але для цього потрібно понести витрати, пов'язані з переробкою програми та згортанням власного дата-центру (або «серверної кімнати»). Зрозуміло, наведена таблиця дуже схематична – в залежності від конкретних завдань і умов кожної організації, найбільш доцільним в економічному плані може виявитися будь-який з чотирьох описаних варіантів. Тим не менше в загальному випадку хмарний варіант привабливий тим, що дозволяє вивільнити частину коштів, які інакше пішли б на рішення непрофільних для основного бізнесу ІТ-задач.

Для розрахунку вигідності використання хмарних обчислень дослідники з Берклі запропонували наступну формулу [6]:

$$T_{\text{хмари}} \times (I - C_{\text{г хмари}}) \geq T_{\text{цод}} \times \left(I - \frac{C_{\text{г хмари}}}{L} \right) \quad (1)$$

Де T – використані години;

I – прибуток;

$C_{\text{г}}$ – вартість 1 години;

L – середня завантаженість.

У лівій частині формули представлені доходи, які компанія може отримати за рахунок використання хмарних обчислень в розрахунку на задану кількість годин машинного часу. Права частина формули – доходи, які компанія може отримати при використанні власного ЦОДа. Істотна різниця полягає в тому, що при використанні внутрішнього дата-центру в розрахунок слід приймати фактор середнього завантаження потужностей. Наприклад, якщо потужності ЦОДа використовуються тільки на 10% (це абсолютно реалістична цифра – за даними Gartner, середній рівень використання серверних потужностей складає 18%, причому для серверів на архітектурі x86 цей показник знижується до 12%), то при розрахунку прибутковості *Вартість 1 год цод* доведеться помножити на 10, що суттєво знизить рівень прибутковості цього варіанту. Дослідники з Берклі

відзначають, що ідеальний в економічному відношенні варіант, коли параметр *Середня Завантаженість* наближається до одиниці, на практиці недосяжний – подібна ситуація фактично свідчить про катастрофічну перевантаженні потужностей. Тому оптимальний рівень завантаження потужностей ЦОД становить близько 60-70%. Відповідно до цієї формули, використання публічних хмарних сервісів тим вигідніше, чим менше середній рівень завантаження потужностей ЦОДа. Якщо очікується, що додаток буде стикатися з інтенсивним навантаженням нерегулярно (наприклад, онлайн-магазин, орієнтований на продаж товарів в сезон різдвяних знижок), то привабливість «хмар» однозначна. Крім того, вибір на користь хмарних обчислень доцільний в тих випадках, коли заздалегідь передбачити середнє і пікове завантаження неможливо. Наприклад, у випадку, якщо навантаження виявиться істотно нижче очікуваного, компанії не доведеться переплачувати за потужності, які простоюють, а якщо вона перевершить очікування, то додати відсутні ресурси на хмарному майданчику буде суттєво легше, ніж закупити додаткове обладнання для установки у власному ЦОДе.

Отже, хмарні обчислення володіють рядом об'єктивних економічних переваг порівняно з традиційною моделлю, яка передбачає розміщення обчислювальних потужностей на власному майданчику. Це особливо справедливо для тих ситуацій, коли навантаження на застосунки нерівномірне або погано передбачуване. У міру того як власники таких застосунків будуть активніше користуватися хмарними ресурсами, економічна ефективність «хмар» буде підвищуватися. З одного боку, ринок буде ставати більш консолідованим і конкурентним, що викличе зниження цін на хмарні ресурси. З іншого боку, розробники засвоять нові підходи до створення застосунків, що дозволить досягнути більшої економічної ефективності на рівні самих онлайн- застосунків за рахунок початкової оптимізації розподілу ресурсів у розрахунку на хмарні обчислення.

2.5 Переваги хмарних обчислень

Розглянемо основні переваги та достоїнства технологій хмарних обчислень.

Клієнтські комп'ютери. Користувачам немає необхідності купувати дорогі комп'ютери, з великим обсягом пам'яті і дисків, щоб використовувати програми через веб-інтерфейс. Також немає необхідності в CD і DVD приводах, тому що вся інформація і програми залишаються в "хмарі". Користувачі можуть перейти з звичайних комп'ютерів і ноутбуків на більш компактні і зручні нетбуки.

Доступ до документів. Якщо документи зберігаються в "хмарі", вони можуть бути доступні користувачам у будь-який час і в будь-якому місці. Більше немає такого поняття як забуті файли: якщо є Інтернет – вони завжди поруч.

Стійкість до втрати даних або крадіжці обладнання. Якщо дані зберігаються в "хмарі", їх копії автоматично розподіляються по декількох серверах, можливо знаходяться на різних континентах. При крадіжці або поломці персональних комп'ютерів користувач не втрачає цінну інформацію, яку він до того ж може отримати з будь-якого іншого комп'ютера.

Надійність. Дата-центри управляються професійними фахівцями, що забезпечують цілодобову підтримку функціонування віртуальних машин. І навіть якщо фізична машина "звалиться", завдяки розподілу застосунка на безліч копій воно все одно продовжить свою роботу. Це створює певний високий рівень надійності та відмовостійкості функціонування системи.

Економічність та ефективність. "Хмара" дозволяє враховувати і оплачувати тільки фактично спожиті ресурси строго за фактом їх використання;

Оренда ресурсів. Звичайні сервери середньої компанії завантажені на 10-15%. В одні періоди часу є потреба в додаткових обчислювальних ресурсах, в інших ці дорогі ресурси простоюють. Використовуючи

необхідну кількість обчислювальних ресурсів в "хмарі" в будь-який момент часу, компанії скорочують витрати на обладнання та його обслуговування. Це дає можливість замовнику відмовитися від закупівель дорогих ІТ-активів на користь їх навіть не оренди, а операційного споживання в міру потреби, при скороченні витрат на обслуговування своїх систем і отриманні від постачальника гарантій рівня сервісу.

Оренда ПЗ. Замість придбання пакетів програм для кожного локального користувача, компанії купують потрібні програми в "хмарі". Дані програми будуть використовуватися тільки тими користувачами, яким ці програми необхідні в роботі. Більше того, вартість програм, орієнтованих на доступ через Інтернет, значно нижче, ніж їх аналогів для персональних комп'ютерів. Якщо програми використовуються не часто, то їх можна просто орендувати з погодинною оплатою. Витрати на оновлення програм і підтримку в працездатному стані на всіх робочих місцях зовсім зведені до нуля.

Для постачальника ІТ-послуг економічний сенс хмари полягає в ефекті масштабу (обслуговувати великий однорідний центр обробки дешевше, ніж безліч маленьких різнорідних) і згладжування навантаження (коли споживачів багато, малоймовірно, що пікові потужності знадобляться всім їм одночасно).

Обслуговування. Так як фізичних серверів з впровадженням Cloud Computing стає менше, їх стає легше і швидше обслуговувати. Що стосується програмного забезпечення, то останнє встановлено, налаштовано і оновлюється в "хмарі". У будь-який час, коли користувач запускає віддалену програму, він може бути впевнений, що ця програма має останню версію — без необхідності щось перевстановлювати або платити за поновлення.

Спільна робота. При роботі з документами в "хмарі" немає необхідності пересилати один одному їх версії або послідовно редагувати їх. Тепер користувачі можуть бути впевненими, що перед ними остання версія

документу і будь-яка зміна, внесена одним користувачем, миттєво відбивається в іншого.

Відкриті інтерфейси. "Хмара" як правило, має стандартні відкриті API (інтерфейси прикладного програмування) для зв'язку з існуючими додатками і розробки нових – спеціально для хмарної архітектури.

Гнучкість і масштабованість – необмеженість обчислювальних ресурсів (пам'ять, процесор, диски). "Хмара" еластична і має можливість масштабуватися – ресурси виділяються і звільняються у міру потреби;

Продуктивні обчислення. У порівнянні з персональним комп'ютером обчислювальна потужність, доступна користувачу "хмарних" комп'ютерів, практично обмежена лише розміром "хмари", тобто загальною кількістю видалених серверів. Користувачі можуть запускати більш складні завдання, з великою кількістю необхідної пам'яті, а місця для зберігання даних виділяються тоді, коли це необхідно. Іншими словами, користувачі можуть при бажанні легко і дешево попрацювати з суперкомп'ютером без будь-яких фактичних придбань. Можливість запуску безліч копій програми на багатьох віртуальних машинах представляє переваги масштабованості: кількість примірників застосунка здатне практично миттєво збільшуватися на вимогу, в залежності від навантажень.

Зберігання даних. У порівнянні з доступним місцем для зберігання інформації на персональних комп'ютерах об'єм сховища в "хмарі" може гнучко і автоматично підлаштовуватися під потреби користувача. При зберіганні інформації в "хмарі" користувачі можуть забути про обмеження, що накладаються звичайними дисками, – "хмарні" розміри обчислюються мільярдами гігабайт доступного місця.

Інструмент для стартапів. В очах таких споживачів сервісу хмарних обчислень як компанії, початківці свій бізнес, основною перевагою даної технології є відсутність необхідності закуповувати все відповідне обладнання та ПЗ, а потім підтримувати їх роботу.

2.6 Недоліки та проблеми хмарних обчислень

Відзначимо основні недоліки та труднощі використання хмарних обчислень:

Постійне з'єднання з мережею. Хмарні обчислення майже завжди потребує з'єднання з мережею Інтернет. Якщо немає доступу в мережу – немає роботи, програм, документів. Багато "хмарних" програм вимагають гарного Інтернет-з'єднання з великою перепускною здатністю. Відповідно програми можуть працювати повільніше ніж на локальному комп'ютері. На думку провідних російських ІТ-компаній, основною перешкодою широкому розвитку хмар є відсутність широкосмугового доступу в Інтернет (ШПД) – насамперед у регіонах.

Безпека. Безпека даних теоретично може бути під загрозою. Не всі дані можна довірити сторонньому провайдеру в інтернеті, тим більше, не тільки для зберігання, але ще і для обробки. Все залежить від того, хто надає "хмарні" послуги. Якщо цей хтось надійно шифрує Ваші дані, постійно робить їх резервні копії, вже не один рік працює на ринку подібних послуг і має хорошу репутацію, то загрози безпеки даних може ніколи не статися. У користувача "хмарних" бізнес-застосунків можуть також виникнути і юридичні проблеми, наприклад пов'язані з виконанням вимог захисту персональних даних.

Держава, на території якої розміщений дата-центр, може отримати доступ до будь-якої інформації, яка в ньому зберігається. Наприклад, за законами США, де знаходиться найбільша кількість дата-центрів, в цьому випадку компанія-провайдер навіть не має права розголошувати факт передачі конфіденційної інформації будь-кому, крім своїх адвокатів.

Ця проблема є, напевно, однією з найістотніших в питанні виведення конфіденційної інформації в хмару. Шляхів її вирішення може бути кілька. По-перше, можна шифрувати всю інформацію, що поміщається на хмару. По-друге, можна просто її туди не поміщати. Однак, у всякому разі, у компаній, що користуються хмарними обчисленнями, це повинно бути

визначеним пунктом в списку питань інформаційної безпеки. Крім того, самі провайдери повинні покращувати свої технології, надаючи деякі послуги по шифруванню.

Функціональність "хмарних" застосунків. Не всі програми або їх властивості доступні віддалено. Якщо порівнювати програми для локального використання і їх "хмарні" аналоги, останні поки програють у функціональності. Наприклад, таблиці Google Docs або застосунки Office web application мають набагато менше функцій і можливостей, ніж Microsoft Excel.

Залежність від "хмарного" провайдера. Завжди залишається ризик, що провайдер онлайн-сервісів одного разу не зробить резервну копію даних – якраз перед катастрофою сервера. Ризик цей, втім, навряд чи перевищує небезпеку того, що користувач сам упустиť свої дані – втративши або розбивши мобільник або ноутбук, не створивши на домашньому ПК резервну копію. Крім того, прив'язавшись до тієї чи іншої послуги, ми в якійсь мірі також обмежуємо свою свободу – свободу переходу на стару версію софтвера, вибору способів обробки інформації і так далі.

2.7 Порівняння Грід і хмарних обчислень

Деякі дослідники вважають, що основною відмінністю хмарних обчислень від грід є віртуалізація: «Хмарні обчислення, на відміну від грід, застосовують віртуалізацію для максимізації обчислювальної потужності. Віртуалізація, за допомогою відділення логічного рівня від фізичного, вирішує безліч проблем, з якими стикаються грід-рішення»[7].

У той час як грід-системи забезпечують високе завантаження обчислювальних ресурсів за допомогою розподілу однієї складної задачі на декілька обчислювальних вузлів, хмарні обчислення йдуть по шляху виконання декількох завдань на одному сервері у вигляді віртуальних машин. Крім того, є особливості в основних варіантах використання грід і

хмарних обчислень. Тоді як грід, в основному, використовується для вирішення завдань за певний (обмежений) проміжок часу, хмарні обчислення в основному орієнтовані на надання «довгоживучих» сервісів.

Думки сходяться в одному – хмарні обчислення виросли з концепції грід. Фостер визначає взаємодію грід і хмарних обчислень наступним чином: «Ми вважаємо, що хмарні обчислення не просто перетинаються з концепцією грід. Насправді хмари виросли з грід-обчислень і ґрунтуються на концепції інфраструктури грід. Еволюція підходу полягає в тому, що замість надання «сирих» обчислювальних ресурсів і ресурсів зберігання забезпечується надання більш абстрактних ресурсів у вигляді сервісів».

Таким чином, можна вважати що грід і хмарні обчислення доповнюють один одного. Інтерфейси та протоколи грід можуть забезпечити взаємодію між хмарними ресурсами або ж забезпечити об'єднання хмарних платформ. Також, більш високий рівень абстракції, надаваний хмарними платформами, може допомогти користувачам грід-систем в організації прозорого і зручного надання ресурсів грід-платформ і залучити нові групи користувачів до використання таких ресурсів.

З точки зору користувача, різниця між хмарними обчисленнями і грід-обчисленнями буде полягати в наступному:

- *Хмарні платформи фокусуються на підході «все як сервіс».* Грід обчислення фокусуються на проміжному програмному забезпеченні, яке надається у вигляді відкритих вихідних кодів або ж у вигляді готових пакетів. При цьому комунальні обчислення є всього лише однією з форм надання грід. У порівнянні з цим, хмарні обчислення фокусуються виключно на платному наданні інформаційних ресурсів кінцевому користувачеві. При цьому проміжне програмне забезпечення, яке дозволило б забезпечити розробку власної хмари, поки не дуже поширене.

- *Грід і хмарні обчислення фокусуються на різних типах обчислень.* Спочатку грід-обчислення були орієнтовані на вирішення наукових задач за допомогою суперкомп'ютерних систем. В даний час грід застосовується для

науково-дослідних завдань, вирішення яких вимагає об'єднання декількох суперкомп'ютерних платформ. З іншого боку, хмарні обчислення орієнтовані не на вирішення окремих завдань, а на перманентне надання певних сервісів кінцевим користувачам. Вони забезпечують динамічний розподіл фізичних ресурсів для задоволення змінної завантаження таких сервісів.

- *Різні взаємовідносини з постачальниками ресурсів.* Грід-обчислення ґрунтуються на понятті віртуальних організацій, що включають до себе кілька різних окремих організацій з чіткими правилами взаємодії між ними і чіткими політиками надання програмно-апаратних ресурсів. Концепція хмарних обчислень забезпечує можливість будь-якої компанії використовувати хмарні сервіси для вирішення власних завдань, оплачуючи тільки ті ресурси, які необхідні.

- *Різні області застосування.* Грід-платформи надають базу для розгортання обчислювальної інфраструктури. Хмарні обчислення надають інтегрований підхід на всіх рівнях надання інформаційних ресурсів: IaaS, PaaS, SaaS.

- *Розширення кількості користувацьких інтерфейсів.* Грід-обчислення орієнтовані на представлення різних обчислювальних ресурсів в гетерогенних обчислювальних середовищах для вирішення конкретних завдань. Таким чином, інтерфейси грід орієнтовані на взаємодію обчислювальних інфраструктур на фізичному рівні за допомогою API, яким може скористатися тільки професійний програміст. Хмарні обчислення розробляються таким чином, щоб надавати інтерфейси кінцевим користувачам через веб-доступ або за допомогою API. На кожному шарі (IaaS, PaaS, SaaS) надається свій власний інтерфейс. Підвищення рівня абстракції дозволяє забезпечити застосування хмарних обчислень як на рівні окремих користувачів, так і на рівні корпоративних клієнтів.

Загалом, грід-обчислення забезпечують об'єднання гетерогенних обчислювальних ресурсів в єдину обчислювальну середу. Це те, з чого

починаються і на чому ґрунтуються хмарні обчислення. Хмарні обчислення забезпечують більш високий рівень абстракції, надаючи обчислювальні ресурси кінцевим користувачам (будь то приватні клієнти або організації) у вигляді сервісів.

2.8 Постачальники хмарних обчислень

При класифікації постачальників «хмар» виділяють дві великі групи: постачальники платформних «хмар» (ІТ-компанії, що пропонують хостинг на базі власного ПЗ) і постачальники «хмар» послуг, що використовують для створення сервісів ПЗ інших компаній (рис. 2.3) [7]. Перша група постачальників поділяється на три підгрупи: Google, Microsoft та інші великі компанії (IBM, Apple, а також Yahoo!, EMC, HP / EDS, Amazon, Facebook, Adobe і т.д.). До другої групи постачальників «хмарних» обчислень можна віднести незалежних сервіс-провайдерів, що надають послуги зовнішнім клієнтам, а також внутрішньокорпоративних провайдерів, що обслуговують філії, відділи та дочірні підрозділи, а також співробітників і партнерів.



Рисунок 2.3 – Типи «хмар»

Між платформними і сервісними постачальниками хмарних обчислень є конкуренція – незважаючи на великий потенціал ІТ-гігантів, послуги хостерів скоро будуть затребувані і в контексті хмарних обчислень. Цьому сприяє також і те, що середній і малий бізнес в пошуках зниження витрат на ІТ переходить до використання хостингу обчислювальних ресурсів і при цьому шукає найбільш дешеві пропозиції. При такому сценарії розвитку ринку вартість послуг стане не єдиною перевагою незалежних сервіс-

провайдерів – важливо те, що хостери будуть проявляти більш високу гнучкість і оперативність в роботі з клієнтами.

Поступово відходять у минуле і інші чинники, які гальмують розвиток «хмарного сервісу». Так, поряд з досить істотним підвищенням надійності мереж сьогодні зменшуються і побоювання компаній, пов'язані з потенційною можливістю контролю провайдером послуг «хмари» секретних даних підприємств.

Постачальники хмарних платформ

Розглянемо можливості та організацію найбільш поширених на сьогоднішній день платформ хмарних обчислень: Amazon Web Services, Google App Engine і (таблиця 2.3) [7].

Таблиця 2.3 – Основні характеристики найбільш поширених хмарних платформ

Платформи Характеристики	Amazon Web Services	Google App Engine	Microsoft Windows Azure
Тип	Iaas	Paas	Paas
Сервіси, що розроблюються	Віртуальні сервіси, сервіси зберігання	Web- застосунки	Як Web-застосунки, так й не Web-застосунки
Віртуалізація	Рівня ОС із запущеним гіпервізором Xen	Контейнер застосунків	Рівня ОС
Інтерфейс доступу користувача	Утиліти консолі Amazon EC2	Web- консоль адміністрування	Портал Microsoft Windows Azure
Web APIs	Да	Да	Да
Середовище розробки	Відсутня	Python, Java	Microsoft.NET

Amazon Web Services

Рішення, запропоноване компанією Amazon, стало стандартом «de facto» в області хмарної інфраструктури. Не дивлячись на те, що рішення, запропоноване компанією Amazon, не є унікальним, всі інші рішення розглядаються або як доповнення до того, що пропонує Amazon Web Services (AWS), або як конкуренти (хоча може бути і не явні) даної платформи. Основною відмінністю AWS від всіх інших платформ є те, що

кожен розробник може використовувати власне середовище виконання застосунків. При цьому можна вибрати один з безлічі образів віртуальних машин, пропонованих компанією Amazon, так і використовувати власне рішення.

На рисунку 2.4 представлені найбільш суттєві елементи рішення, пропонованого в рамках системи AWS. Як можна помітити, Amazon пропонує дуже широкий набір можливостей організації обчислень, зберігання інформації та надання інформації. При цьому безліч сервісів по управлінню і білінгу, які пропонує компанія, просто не помістилися на діаграму.



Рисунок 2.4 – Архітектура Amazon Web Services

Ядром Amazon Web Services служить система Amazon Elastic Compute Cloud (Amazon EC2) спільно з сервісами зберігання. EC2 надає користувачеві вибір віртуальних машин, які можна запустити в розподіленому обчислювальному середовищі. Віртуальна машина (називається Amazon Machine Image – AMI) може бути заснована практично на будь-якій операційній системі (різні версії Windows і дистрибутиви Linux) і забезпечувати роботу з будь-якої програмної інфраструктурою (MySQL, Oracle та ін.)

Поряд з віртуальними машинами, Amazon забезпечує декілька механізмів зберігання даних:

- *Simple Storage Service (S3)* являє собою сервіс зберігання даних, який надає доступ на основі REST і SOAP. Система S3 розподілена по всьому світу: сервери розташовані в Європі, Азії та Сполучених Штатах Америки. При цьому забезпечується можливість роботи з даними розміром від одного байта до 5TB.

- *Система Elastic Block Storage (EBS)* являє собою високопродуктивний віртуальний жорсткий диск розміром від 1Гб до 1Тб. Він може бути підключений до будь-якої віртуальної машини, що працює в рамках EC2 або ж може бути поміщений на тривале зберігання в систему S3.

- *Сервіси реляційних баз даних* – це веб-сервіси, що забезпечують установку, управління та масштабування реляційних баз даних в хмарі. На даний момент надається підтримка баз даних на основі MySQL (в подальшому планується впровадження і баз даних Oracle).

- *Amazon CloudFront* є веб-сервісом для надання контенту.

- Забезпечується статична і потокова передача даних. У зв'язку з розподіленою інфраструктурою системи, CloudFront може забезпечити мінімальну латентність надання інформації, вибираючи найбільш географічно-найближчий до користувача сервер.

Однією з відмінних особливостей платформи, наданої Amazon, є надання сервісів інтеграції застосунків, що забезпечують прозору адресацію і доступність:

- *Elastic IP* забезпечує прив'язку статичної IP адреси до аккаунту користувача. При цьому, Elastic IP може забезпечити обхід помилок і збоїв в роботі окремих віртуальних машин, забезпечуючи автоматичне перепризначення IP адреси іншій віртуальній машині.

- *Simple Queue Service (SQS)* забезпечує розробників практично нескінченною кількістю черг. Кожен авторизований застосунок може реєструватися в черзі, відправляти, отримувати та видаляти повідомлення.

При цьому не прийняті повідомлення можуть залишатися в системі аж до 4-х днів.

- *Simple Notification Service (SNS)* – це сервіс, який забезпечує оповіщення про зміну стану за передплатою. Користувачі системи, хмарні застосунки та пристрої можуть відправляти і отримувати оповіщення із хмари.

- *Сервіси реляційних баз даних* – це веб-сервіси, що забезпечують установку, управління та масштабування реляційних баз даних в хмарі. На даний момент надається підтримка баз даних на основі MySQL (в подальшому планується впровадження і баз даних Oracle).

Також, зараз Amazon виходить на корпоративний ринок, надаючи послуги створення віртуальних приватних хмар, що розширюють можливості корпоративних обчислювальних інфраструктур. По суті, пропонується організація віртуальних приватних мереж (VPN), що забезпечують безпечний та прозорий зв'язок між внутрішньою корпоративною мережею та EC2. Забезпечується імпорт корпоративних віртуальних машин, а також можливість пересилки віртуальних машин на фізичних носіях, минаючи інтернет (підвищується надійність і безпеку передачі даних).

Рішення, пропоноване Amazon Web Services, забезпечує максимально можливу гнучкість при розробці та впровадженні хмарних рішень. Найбільш активними користувачами даної платформи є великі організації, або проекти, яким необхідна максимальна масштабованість і гнучкість розробки. При цьому таке рішення може виявитися надмірно складним і навантаженим для окремих розробників або застосунків, яким не потрібні настільки потужні механізми масштабування.

Google App Engine

Платформа Google App Engine є одним з найбільш відомих хмарних сервісів, орієнтованих на надання платформи (PaaS) для хмарних застосунків. Поряд з готовим середовищем виконання хмарних застосунків

надаються розширені засоби для адміністрування і розробки застосунків. На рисунку 2.5 представлена принципова схема архітектури системи Google App Engine.

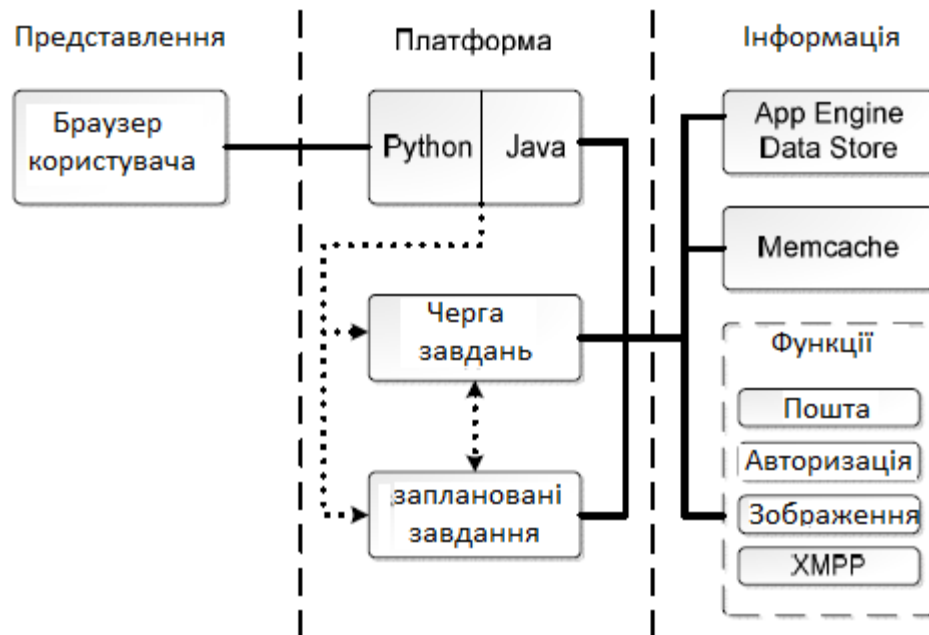


Рисунок 2.5 – Архітектура системи Google App Engine

На рисунку представлена спрощена схема архітектури системи Google App Engine. Користувач отримує доступ до хмарного застосунку за допомогою веб-браузера. До недавнього часу Google App Engine забезпечувало можливість розробки тільки на базі мови Python, але нещодавно була представлена підтримка розробки на базі будь-якої мови, яка може виконуватися всередині віртуальної машини Java. Таким чином, в даний момент Google App Engine може запускати застосунки, написані на Java, Python, Scala і т.п. Розробникам Google App Engine надається повноцінний комплект засобів розробки (SDK), що включає до себе повноцінну симуляцію роботи Google App Engine на робочій машині.

Сховище *App Engine Datastore* – це високорозподілена система зберігання даних, яка ґрунтується на пропрієтарній базі даних BigTable, розробленої компанією Google. Методи роботи з сховищем дуже сильно відрізняються від методів роботи з реляційними базами даних. Основною відмінністю є те, що в App Engine Datastore відсутні схеми даних. Всі дані

зберігаються у вигляді блоків «ключ» — «значення» — «мітка часу». Відповідно, вся відповідальність на відповідність збережених даних бізнес-логіці застосунку лягає на розробника. Природно, в SDK надаються спеціальні бібліотеки, які дозволяють забезпечити узгодженість даних. Також, Google розробили власну мову запитів (мова GQL), яка схожа на вирази SELECT в мові SQL, але з великою кількістю обмежень (наприклад, відсутній оператор JOIN).

Також, застосунку заборонені операції роботи з локальною файловою системою (дозволено тільки читання файлів, які були завантажені разом з кодом застосунку). З іншого боку, Google App Engine надає великий набір бібліотечних функцій для виконання стандартних операцій:

- роботи з поштовими повідомленнями;
- авторизації та аутентифікації користувачів;
- обробки зображень;
- завантаження та обробки веб-сторінок;
- планування завдань;
- обробки даних за допомогою mapreduce;
- зберігання великих (до 2 gb) обсягів інформації;
- обміну повідомленнями на основі протоколу xmpp (jabber).

Таким чином, не дивлячись на те, що Google App Engine не дозволить розгорнути хмарний застосунок, просто скопіювавши старий код на Java або Python, він може забезпечити можливість надання високопродуктивного сервісу без істотних витрат на інфраструктуру. Але процес розробки хмарного застосунку може потребувати великих витрат, пов'язаних з використанням нових моделей роботи з даними і обмеженнями віртуальної машини.

Microsoft Windows Azure

Платформа Windows Azure — це PaaS від корпорації Microsoft, що дозволяє запускати застосунки, розроблені на базі платформи Microsoft .NET

на серверах компанії Microsoft [6]. Забезпечується автоматичне керування обчислювальними ресурсами, балансування навантаження і реплікація даних.

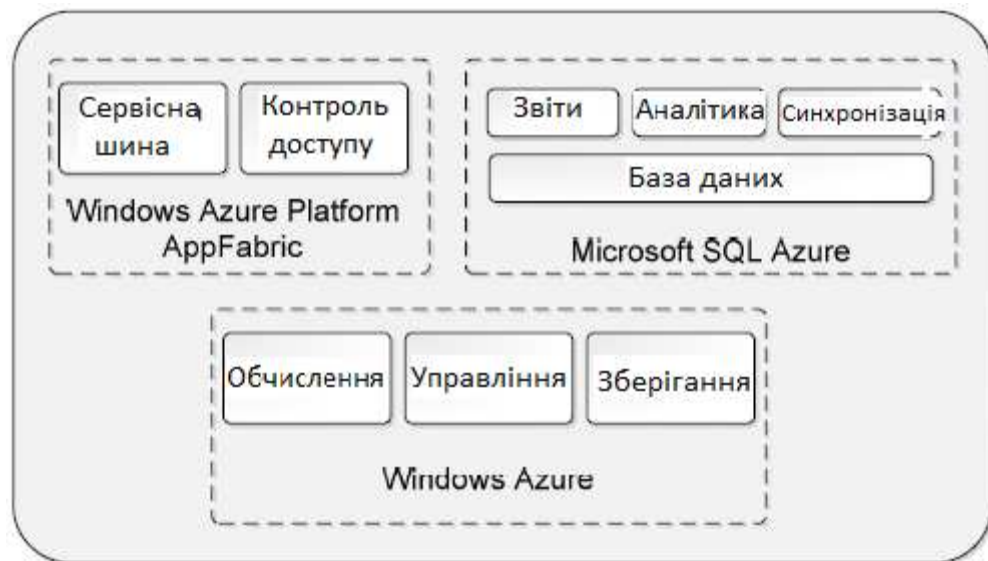


Рисунок 2.6 – Платформа Windows Azure

Доступ розробника до платформи здійснюється за допомогою інструментарію, інтегрованого в останні версії Visual Studio (Windows Azure SDK, Windows Azure Tools for MVS). При цьому підтримується можливість локального тестування за стосунків до їх публікації на сервісі Azure. Windows Azure створена на основі технологій віртуалізації, схожих з технологією Windows Server Hyper-V і управляється за допомогою спеціального інфраструктурного шару, званого Windows Azure Fabric Controller. Завдання даного шару – організація масиву віртуальних машин на основі Windows Server у вигляді єдиного віртуального сервера, що забезпечує автоматичне управління ресурсами, балансування навантаження та ін.

Платформа Windows Azure складається з трьох основних компонентів: обчислювальних сутностей, сутностей зберігання і фабрик.

Обчислювальні сутності – це контейнери для застосунків з підтримкою сучасних технологій розробки, включаючи .NET, Java, PHP, Python, Ruby on Rails і нативний код. Існує дві основних ролі, які можуть виконувати обчислювальні сутності. *Веб-роль (Web Role)* – це веб-

застосунок, доступний користувачу через інтернет, доступ до якого можна отримати по засобом веб-браузера. *Прикладна роль (Worker Role)* – це програма, яка виконує деяке обчислювальне навантаження у фоновому режимі.

Сутності зберігання – це набір сервісів, що забезпечують зберігання даних. Кожен сервіс забезпечує свій тип зберігання даних.

- *Бінарні об'єкти* – забезпечується зберігання великих наборів неструктурованих байтів (файлів). Забезпечується можливість іменування файлів і роботи з метаданими. Максимальний обсяг зберігається файлу – не більше 1 Тб.

- *Таблиці* – використовуються для зберігання структурованих даних. Таблиці представляють собою набір однорідних рядків (званих сутностями), структура яких визначається набором колонок (званих властивостями). Один об'єкт може мати до 256 властивостей. Таблиці розподілені таким чином, щоб максимально підтримувати балансування навантажень.

- *Черги* – механізм, що забезпечує асинхронну взаємодію застосунків за допомогою збереження і передачі повідомлень. Допускається використання необмеженого числа черг, а черги можуть містити необмежене число повідомлень.

- *Диски* – аналогічно AWS, Windows Azure дозволяє управляти дисками в рамках своїх віртуальних машин. Azure дозволяє працювати з томами NTFS, забезпечуючи їх доступність для застосунків. Це може дозволити спростити процес міграції звичайних застосунків в хмару, тому з'являється можливість збереження стану у файловій системі.

Також платформа Windows Azure забезпечує ряд сервісів, доступних розробникам як Інтернет, так і класичних десктопних застосунків.

Найбільш важливим є компонент SQL Azure, що забезпечує надання реляційної бази даних Microsoft SQL Server як сервісу. Так як дана технологія заснована на класичному підході SQL Server, вона не забезпечує

такої масивної масштабованості, як сутності зберігання (максимальний об'єм інформації становить 10 Гб на одну базу даних).

Очевидно, що рішення Microsoft буде користуватися популярністю на ринку хмарних сервісів. Перш за все, це пов'язано з поширеністю рішень Microsoft на ринку корпоративних систем, а технологія Windows Azure надає безліч сервісів, покликаних полегшити перенесення таких застосунків в хмарну платформу. При цьому природно, що вся інфраструктура Azure заточена на програмні продукти і технології, що надаються компанією Microsoft. Відповідно, якщо ваш додаток розробляється на основі альтернативної ідеології, цілком можливо варто пошукати альтернативні хмарні платформи.

Вибір хмарного PaaS-постачальника

Існуючі сьогодні PaaS-рішення настільки різноманітні за характером вирішуваних завдань, що виділити їх родову властивість досить важко. Якщо говорити в цілому, то все PaaS-рішення дозволяють підвищити ефективність праці розробників додатків.

У таблиці 2.4 вказані основні фактори, на які варто звернути увагу при виборі PaaS-постачальника.

Таблиця 2.4 – Основні критерії оцінки PaaS-постачальників.

№	Критерій оцінки	Опис
1	Ціна	- За якою моделлю здійснюється оплата за використання сервісу (за фактом обсягу використаних ресурсів / фіксований пакет)? - Чи існує безкоштовний ознайомчий варіант?
2	Технології, що підтримуються	Чи підтримує PaaS-постачальник мови програмування, системи зберігання даних і інфраструктурні сервіси, які потрібні зараз і можуть знадобитися в майбутньому?
3	Можливості інтеграції	Чи надаються механізми інтеграції з мобільними платформами, з хмарними системами і інтерфейсами сторонніх постачальників? * Чи підтримуються стандартні інтерфейси обміну інформацією (REST, JSON та ін)?

4	Репутація	• Чи відбувалися збої в роботі постачальника і наскільки оперативно вони усувалися? • Чи підтримує постачальник можливість розподіленого розміщення додатків і даних одночасно в декількох дата-центрах?
5	Додаткові сервіси	• Чи надає постачальник додаткові засоби розробки? • Які умови технічної підтримки? • Чи існує каталог додаткових модулів, що дозволяють розширити базову функціональність? • Чи є у продукту активно підтримуваний форум, вікісайти і блоги?
6	Низькорівневий контроль	• Чи надає PaaS-постачальник можливість донастройки компонентів системи або ж розробник має доступ тільки до високорівневим API?

Для вибору відповідного до наших цілей хмарного провайдера, складемо порівняльну характеристику найбільш відомих [8].

Оберемо 15 критеріїв порівняння:

1. **План оплати** – Провайдери пропонують плани оплати типу: "сплачуєш-по-факту-використання" (зазвичай в погодинному режимі); щомісячна оплата, де користувач отримує знижку на послуги в обмін на річне обслуговування; комбінація попередніх способів оплати.

2. **Середньомісячна ціна** – Орієнтовна вартість в доларах за «хмарний» сервер з 1 CPU, 2GB, усереднена для датацентрів і серверів Windows / Linux.

3. **Service Level Agreement (SLA)** – Угоди по рівню сервісу в процентах.

4. **Кількість датацентрів** – Кількість датацентрів, які використовуються для «хмарних» серверів.

5. **Сертифікації** – В тому випадку, якщо вендор має різні сертифікати безпеки та надійності типу PCI або SAS 70.

6. Вертикальне масштабування – По можливості вертикальне масштабування додаванням більшої кількості пам'яті, CPU та обсягу сховищ.

7. Горизонтальне масштабування – Швидке розгортання нових серверів.

8. Підтримка – Трирівнева шкала оцінювання:

низький рівень – Компанії, що надають безкоштовну підтримку тільки он-лайн (форуми і т.д.), інша підтримка повинна оплачуватися.

середній рівень – Компанії, що надають один тип 24x7 підтримки безкоштовно (телефон або он-лайн чат, форуми).

високий рівень – Компанії, що надають набір пропозицій, включених в базову ціну.

9. Моніторинг – Трирівнева шкала оцінювання:

низький рівень – Компанії без інтегрованих рішень моніторингу / сповіщення. Необхідно розгортання сторонніх інструментів або покупка додаткових послуг.

середній рівень – Компанії з дуже простим набором простих інтегрованих засобів (декілька індикаторів або без оповіщення).

високий рівень – Компанії з повноцінними безкоштовними інтегрованими засобами моніторингу.

10. API – Наявність API для взаємодії з серверами.

11. Безкоштовна ознайомча версія – Наявність безкоштовної ознайомлювальної версії, яку клієнт може використовувати для тестування роботи з сервісом.

12. ОС, які підтримуються – Кількість операційних систем, які підтримуються незалежно від версії, доступних у вигляді налаштованих образів.

13. Різноманітність типів екземплярів – Кількість різних конфігурацій серверів. Деякі провайдери пропонують повністю налаштовуються сервера в термінах CPU (позначені як "конфігуруються").

14. **Вартість вихідного трафіку** – Вартість в доларах кожного гігабайта вихідного трафіку. Компанії, що надають безкоштовний вихідний трафік, мають значення ціни 0.

15. **Вартість вхідного трафіку** – Вартість в доларах кожного гігабайта вхідного трафіку. Компанії, що надають безкоштовний вихідний трафік, мають значення ціни 0.

Зведемо всі дані до таблиці 2.5.

Таблиця 2.5 – порівняльна характеристика провайдерів хмарних послуг.

Провайдер	План оплати	Среднемесчная цена (US\$)	SLA	Датацентры	Сертификации	Вертикальное масштаб.	Горизонтальное масштаб.	Поддержка	Мониторинг	API	Бесплатн. ознакомит. версия	ОС	типы экземпляров	Стоимость исходящего трафика (GB)	Стоимость входящего трафика (GB)
Amazon EC2	Pay-as-you-go or Year + Discount	80,81	99,95%	7	+	—	+	низкий	высокий	высокий	+	9	12	0,12	0
BitRefinery	Monthly	137	100%	1	+	+	+	высокий	низкий	-----	-	3	конфиг.	0	0
GoDaddy	Monthly	39,99	99,90%	8	-	—	+	высокий	низкий	-----	-	4	5	0	0
GoGrid	Pay-as-you-go or Monthly	273,6	100%	2	-	+	+	высокий	низкий	средний	-	4	1	0,29	0
Hosting.com	Monthly	270	100%	4	+	+	+	высокий	средний	-----	-	3	конфиг.	0	0
NephoScale	Pay-as-you-go or Year + Discount	146	99,95%	1	-	+	+	средний	низкий	высокий	+	4	6	0,13	0
OpSource	Pay-as-you-go or Monthly	87,6	100%	4	+	+	+	высокий	средний	высокий	-	4	конфиг.	0,15	0
Rackspace	Pay-as-you-go	51,1	100%	9	+	+	+	высокий	высокий	высокий	-	8	8	0,18	0
ReliaCloud	Monthly	135,05	100%	2	+	—	+	средний	низкий	средний	-	5	5	0,12	0
Softlayer	Pay-as-you-go or Monthly	135,05	?	7	+	—	+	низкий	высокий	высокий	-	6	13	0,1	0
Terremark*	Pay-as-you-go	133,39	100%	9	+	+	+	низкий	низкий	средний	-	5	конфиг.	0,17	0,17
Windows Azure	Pay-as-you-go	94,9	100%	8	+	+	+	высокий	средний	высокий	+	7	4	0,12	0

Зробимо вибір найкращої альтернативи ранжируванням хмарних провайдерів за притаманними їм характеристиками. Оцінимо характеристики кожного провайдера за шкалою значень від 0 до 1, де «0» відповідає відсутності даної характеристики у провайдера.

Підсумкова оцінка кожного «альтернативного» провайдера обчислюється як сума добутків оцінки кожної характеристики, властивої провайдеру і вагового коефіцієнта даної характеристики. Так як всі характеристики (критерії), за якими відбувається порівняння, однаково важливі, то привласнимо кожній ваговій коефіцієнт рівний 1.

Отримані результати ранжирування представлені в таблиці 2.6.

Таблиця 2.6 – Результати ранжирування хмарних провайдерів.

Провайдер	План оплаты	Среднемсячная цена (US\$)	SLA	Датацентры	Сертификация	Вертикальное масштаб.	Горизонтальное масштаб.	Поддержка	Мониторинг	API	Бесплатн. ознакомит. версия	ОС	типы экземпляров	Стоимость исходящего трафика (GB)	Стоимость входящего трафика (GB)	Результат
Amazon EC2	0,6	0,704641813	0,75	0,77777778	1	0	1	0	1	1	1	1	0,461538462	0,586206897	1	0,725
BitRefinery	0	0,499269006	1	0,11111111	1	1	1	1	0	0	0	0,3	1	1	1	0,596
GoDaddy	0	0,853837719	0,5	0,88888889	0	0	1	1	0	0	0	0,4	0,192307692	1	1	0,459
GoGrid	1	0	1	0,22222222	0	1	1	1	0	1	0	0,4	0,038461538	0	1	0,514
Hosting.com	0	0,013157895	1	0,44444444	1	1	1	1	0,5	0	0	0,3	1	1	1	0,619
Nephoscale	0,6	0,466374269	0,75	0,11111111	0	1	1	0,5	0	1	1	0,4	0,230769231	0,551724138	1	0,577
OpSource	1	0,679824561	1	0,44444444	1	1	1	1	0,5	1	0	0,4	1	0,482758621	1	0,770
Rackspace	0,3	0,813230994	1	1	1	1	1	1	1	1	0	0,9	0,307692308	0,379310345	1	0,779
ReliaCloud	0	0,506396199	1	0,22222222	1	0	1	0,5	0	1	0	0,6	0,192307692	0,586206897	1	0,504
Softlayer	1	0,506396199	0	0,77777778	1	0	1	0	1	1	0	0,7	0,5	0,655172414	1	0,607
Terremark*	0,3	0,51246345	1	1	1	1	1	0	0	1	0	0,6	1	0,413793103	0	0,585
Windows Azure	1	0,653143275	1	0,88888889	1	1	1	1	0,5	1	1	0,8	0,153846154	0,586206897	0	0,771
BEC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
Макс. Значение		273,6		9								9	13	0,29		

У результаті найбільш придатними альтернативами для нас є наступні постачальники хмарних послуг: Rackspace, Windows Azure, OpSource.

2.9 Прогнози розвитку ринку хмарних сервісів

Прогноз Edge Strategies і Microsoft

За даними опублікованого в першому кварталі спільного дослідження Edge Strategies і Microsoft, протягом найближчих п'яти років варто очікувати подвоєння кількості платних хмарних сервісів, споживачами яких виступають компанії з числа малого та середнього бізнесу. Кількість дрібних компаній, що використовують хоча б один платний хмарний сервіс, потроїться в найближчі три роки.

Як показують результати проведеного дослідження, 59% компаній, що використовують хмарні сервіси, відзначають значне підвищення ефективності роботи ІТ-служб у порівнянні з 30% компаній з традиційним пристроєм ІТ-інфраструктури.

Незважаючи на уповільнення темпів економічного зростання, 63% підприємств SMB, споживаючих хмарні сервіси, в найближчі 12-18 місяців розраховують на збільшення обсягів продажів. Безпосередньо пов'язують використання сучасних технологій з прогнозованим поліпшенням бізнес-показників 55% опитаних респондентів, з підвищенням конкурентоспроможності на ринку – 58%. Половина організацій, що взяли участь в опитуванні, вважає, що важливість хмарних обчислень для операційної діяльності з часом зросте.

Тільки 20% компаній впевнені, що дані в хмарі зазнають більшу небезпеку ніж у внутрішньокорпоративних системах. Навпаки, не бачать ніякої різниці між цими двома варіантами 36% респондентів або 49% з числа більш великих компаній в сегменті SMB.

У дослідженні брали участь відповідальні за прийняття рішень ІТ-менеджери понад 3000 компаній у сегменті SMB в 13 країнах світу.

Прогноз Gartner

Gartner прогнозує, що глобальні доходи від продажу послуг software-as-a-service (SaaS) виростуть на 17,9% в порівнянні з минулим роком і досягнуть \$ 14,5 млрд. Тенденція росту збережеться аж до 2015 року, коли обсяг ринку досягне \$ 22,1 млрд.

Закономірно, самим зрілим і одночасно багатообіцяючим регіоном для SaaS залишається Північна Америка і, зокрема, США. За прогнозами Gartner, доходи від продажів відповідних послуг тут виростуть з \$ 7,8 млрд до \$ 9,1 млрд. Замовники вважають за краще використовувати в рамках моделі SaaS переважно електронну пошту та офісні застосунки, фінансове ПЗ, програми для управління витратами. Системи веб-конференцій в Північній Америці теж більш затребувані, ніж в інших регіонах, з причини великої кількості територіально-розподілених структур.

Прогноз Forrester Research

У квітні аналітична компанія Forrester Research опублікувала прогноз розвитку ринку публічних хмарних обчислень. Згідно з відомостями звіту, обсяг хмарного ринку складе \$ 241 млрд, що на \$ 200 млрд більше, ніж

Прогноз Forrester Research представлений окремо по різних сегментах хмарних обчислень:

- Інфраструктура як сервіс (IaaS),
- Платформа як сервіс (PaaS),
- ПЗ як сервіс (SaaS) і
- Бізнес-процеси як сервіс (BPaaS).

Серед найцікавіших висновків звіту слід згадати прогноз про те, що пік ринку IaaS доведеться, коли його обсяги досягнуть \$ 5,9 млрд. Після цього ринок буде поступово стискуватися в зв'язку з задоволенням попиту і наростанням конкуренції.

Що стосується сегмента SaaS, то він буде відчувати впевнене зростання: якщо його обсяг складе \$ 21,2 млрд, то він досягне позначки \$ 93 млрд. У цей момент сегмент SaaS буде близький до насичення. Що стосується ринку PaaS, то його обсяги будуть залишатися відносно невеликими порівняно з SaaS, хоча вже сегмент PaaS перевершить за обсягом IaaS. Крім того, аналітики Forrester Research пророкують, що ринок засобів віртуалізації також буде поступово стискуватися в зв'язку з їх заміщенням понад новими технологіями.

Прогноз Forrester Research

Примітно, що сегмент BPaaS, який сьогодні не всі згодні відносити до хмарних обчислень, за прогнозами Forrester Research, до 2020 року перевершить традиційний сегмент інфраструктурних хмарних сервісів IaaS і наблизиться до сегменту PaaS. Основне зростання BPaaS припадає на період з 2015 по 2020 рр..

Прогноз Cisco

Cisco опублікувала свій перший звіт за результатами дослідження глобальних тенденцій розвитку хмарних обчислень – "Глобальний індекс розвитку хмарних технологій" (Cisco Global Cloud Index). Він містить прогноз про те, що річний обсяг "хмарного" трафіку зросте в 12 разів з 130 ексабайт до 1,6 зеттабайт, а середньорічні темпи його зростання складуть 66 відсотків.

Один зеттабайт дорівнює одному секстильйонів байт або одному триллиону гігабайт. Обсяг трафіку в 1,6 зеттабайт – це:

22 трлн годин потокової музики;

5 трлн годин ділових веб-конференцій з веб-камерами;

1,6 трлн годин онлайн-відеопотоків високого дозволу (HD).

Хмара стала самим швидкозростаючим сегментом трафіку в центрах обробки даних (ЦОД). До сукупний річний обсяг трафіку ЦОД зросте в 4 рази до 4,8 зеттабайт, а середньорічні темпи зростання цього трафіку складуть 33 відсотки. В даний час частка "хмарного" трафіку в загальному трафіку ЦОД становить 11%. До року вона повинна перевищити 33 %. Таким чином, хмара набуває критично важливе значення для майбутнього інформаційних технологій, а також для доставки відео та контенту.

У звіті Cisco про розвиток хмарних технологій міститься прогноз про "перетікання" робочого навантаження із традиційних центрів обробки даних в хмару, а також аналіз «хмарної готовності», тобто готовності основних географічних регіонів і їх мережевої інфраструктури до підтримки корпоративних і користувальницьких хмарних послуг різного типу . В іншому документі – Cisco Global Cloud Index Forecast and Methodology, ("Глобальний індекс розвитку хмарних технологій – прогнози і методологія") докладно описуються результати і методологія даного дослідження.

У порівнянні з традиційними центрами обробки даних, ЦОД, що підтримують хмарні обчислення, будуть відрізнятися більш високою продуктивністю, завантаженістю ресурсів і простотою управління. Віртуалізація стане найважливішим каталізатором, що прискорює процеси консолідації апаратних і програмних засобів, автоматизації та інтеграції підходів до інформаційної безпеки.

Прогноз IHS iSuppli

Поставки хмарних серверів досягнуть 875 тис. одиниць, збільшившись на 35% порівняно і майже в два рази в порівнянні. Такий прогноз опублікувало аналітичне агентство IHS iSuppli. Експерти очікують високі темпи зростання в межах від 23% до 30% протягом кожного з наступних трьох років аж, коли загальний обсяг поставок хмарних серверів досягне 1,8 млн одиниць.

Висновок до розділу 2

Таким чином, хмарні сервера будуть становити все більшу частину від загального числа серверів, чийі поставки зростуть від трохи більше 5% ринку та до 15%, припускають в IHS iSuppli. Хмари при цьому можуть бути як персоналізовані і орієнтовані на приватного споживача, наприклад, сервіси Apple, Google чи Amazon, які дозволяють користувачам зберігати і управляти власними даними, або використовуватися в бізнес-цілях корпоративними замовниками.

3 ЗАГАЛЬНІ ПРИНЦИПИ ПОБУДОВИ САПР НА ОСНОВІ ХМАРНИХ ОБЧИСЛЕНЬ

3.1 Існуючі моделі побудови САПР на основі хмарних обчислень

На сьогоднішній день не існує єдиної компонентної архітектури побудови хмарних застосунків САПР а також чітких готових схем переходу (міграції) архітектури на хмарну платформу. Але, не дивлячись на це, можна назвати декілька реалізацій побудови САПР в хмарі. В основному це системи, що використовують клієнт-серверну модель, в якій передбачається, що передплатники (користувачі хмарних послуг) відправляють по мережі повідомлення серверним комп'ютерам, які у відповідь на отримані повідомлення виконують відповідну роботу.

Найпопулярніші варіанти використання хмарних ресурсів САПР представлені на рисунку 3.1.

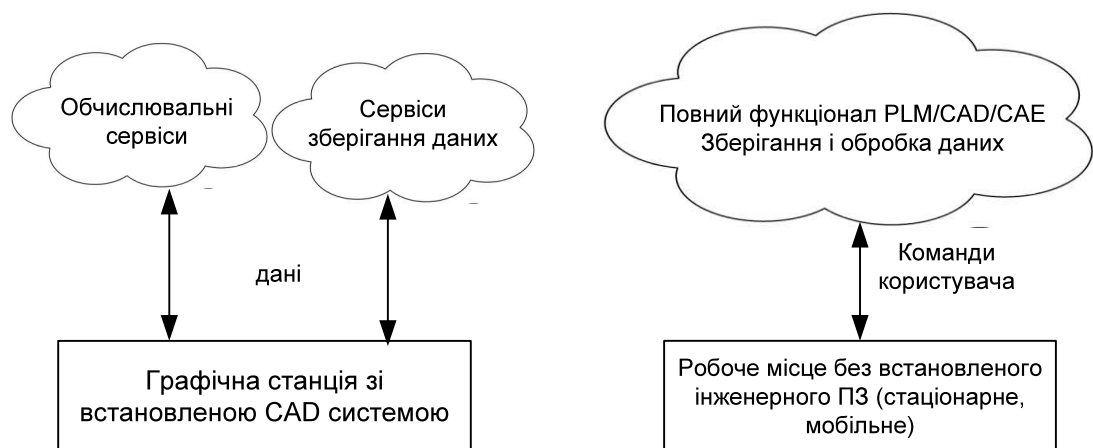


Рисунок 3.1 – Основні схеми використання хмарних ресурсів САПР

Перший варіант передбачає використання тільки деяких сервісів, що надаються провайдерами хмарних послуг, зазвичай це сервіси зберігання та обробки даних, системи резервного копіювання даних, системи управління версіями, сервіси обробки для наукових і технічних завдань. У цьому випадку у розробника як і раніше залишається графічна робоча станція з установленою САПР, але при цьому він отримує з неї доступ до різних хмарних сервісів. У такій реалізації серверна частина представляється інтегрованим сервером СУБД і сервером застосунків і відповідає за роботу

всього комплексу, фізичне зберігання і безпеку даних. Клієнтська частина являє собою автономний застосунок для роботи з будь-якими системами, які щодня використовуються для підготовки інженерної документації. Недоліком такого способу використання хмарних послуг є те, що розробник як і раніше не має можливості віддаленої роботи, повинен піклуватися про постійне оновлення використовуваного програмного продукту, відволікатися від процесу розробки на підтримку і експлуатацію вже існуючих обчислювальних ресурсів, або, в разі їх нестачі, на купівлю відповідного обладнання.

Другий варіант передбачає перенесення всієї інфраструктури інженерних служб у хмару. При такій реалізації у розробника пропадає необхідність у використанні спеціальних технічних засобів, а також в інженерному ПО, встановленому на робочому місці. Такі проекти ґрунтуються на концепції віртуалізації застосунків і дозволяють запускати звичайні Windows-застосунки у вікні браузера. При віртуалізації застосунків по мережі від клієнта до сервера передаються тільки кліки і рухи миші, натиснення клавіш клавіатури, а від сервера до клієнта – зміни екрана. Дана реалізація теж має ряд істотних недоліків, а саме: великий час відгуку і невідповідність вимозі до якості графічного зображення, що неприпустимо при розробці застосунків САПР.

Відомі й інші способи організації застосунків на основі хмарних обчислень, але вони мають індивідуальний характер і інформація про організації архітектур таких додатків не поширюється, оскільки є комерційною таємницею.

При проектуванні застосунків на платформі Windows Azure необхідно створити обліковий запис виставлення рахунків, підписавшись на службу Microsoft Online Services. На рисунку 3,2 показана конфігурація тарифікації Windows Azure для стандартної підписки [9].

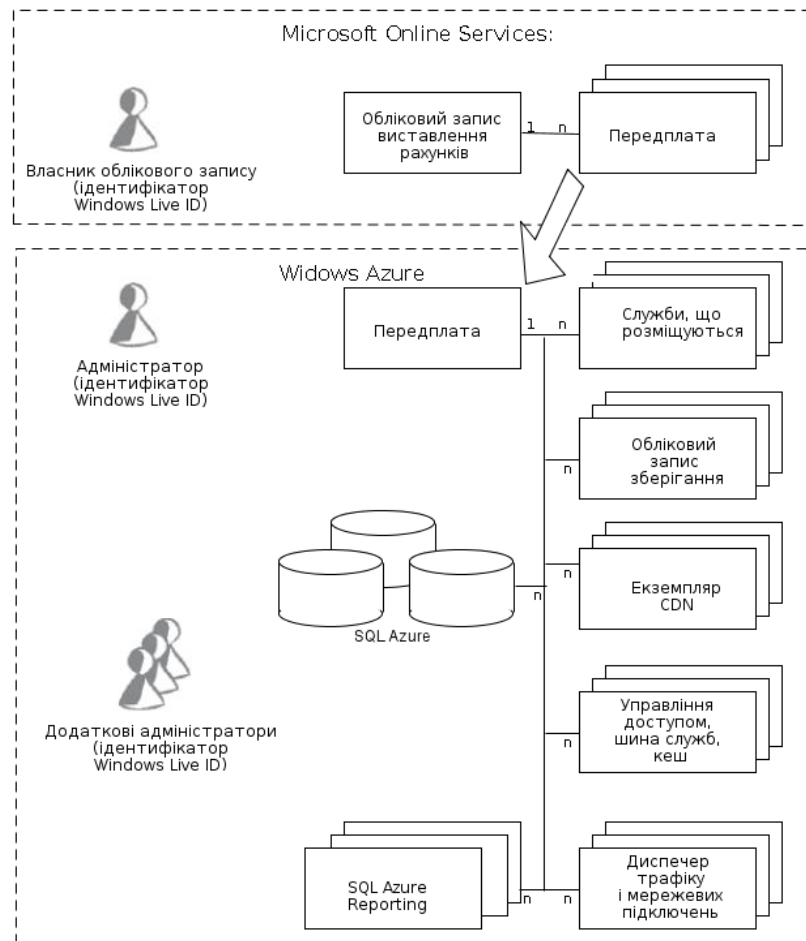


Рисунок 3.2 – Конфігурація виставлення рахунків Windows Azure для стандартної підписки

Кожен обліковий запис виставлення рахунків має одного власника облікового запису, який визначається за ідентифікатором Windows Live. Власник облікового запису може створювати підписки і керувати ними, переглядати дані для виставлення рахунку та дані про використання, а також вказувати адміністратора служби для кожної підписки. До складу підписки Windows Azure може входити наступне:

- Розміщені служби, що складаються з ролей і екземплярів в кожній ролі.
- Ролі та екземпляри можуть бути зупинені у виробничому (production) режимі або в тестовому (staging) режимі.
- Облікові записи сховищ, що складаються з екземплярів сховищ таблиць, великих двійкових об'єктів і черг.
- Примірники мережі доставки вмісту.

- Бази даних SQL Azure.
- Екземпляри служб SQL Azure Reporting Services.
- Екземпляри служб управління доступом, шини інтеграції і кеша.
- Екземпляри підключення віртуальної мережі і диспетчера трафіку.

Оцінка витрат використання «хмарних» ресурсів

У загальному випадку цільова функція витрат Q використання «хмарних» ресурсів розраховується таким чином:

$$Q = (A(s1, n1, t) + B(s2) + C(s3, v1) + D(v2) + E(v3) + F(n2) + G(s4)) \rightarrow \min,$$

де

- $A(s1, n1, t)$ – функція витрат на використання обчислювальних ресурсів, що залежить від розміру, числа і тривалості роботи екземплярів за стосунку;
- $B(s2)$ – функція витрат на використання бази даних, що залежить її від розміру;
- $C(s3, v1)$ – функція витрат на використання сховища Azure Storage, що залежить від розміру використовуваного сховища і обсягу транзакцій;
- $D(v2)$ – функція витрат на вихідний трафік з ЦОД Azure, що залежить від обсягу переданих даних;
- $E(v3)$ – функція витрат на використання ресурсів CDN, яка вимірюється в Гбайт вихідних даних;
- $F(n2)$ – функція витрат на використання сервісів Access Control і Service Bus, яка залежить від числа з'єднань;
- $G(v4)$ – функція витрат на використання служб кешування, що залежить від розміру використовуваного кеша.

Для більшості розроблюваних застосунків базовий тарифний план враховує використання 4 видів ресурсів: процесорного часу, оперативної пам'яті, систем зберігання та мережі (вхідний і вихідний трафік) – це функції $A(s1, n1, t)$, $B(s2)$, $C(s3, v1)$ і $D(v2)$, отже особливу увагу при проектуванні хмарного застосунку необхідно звернути на організацію обчислювальних

ресурсів і структуру зберігання даних. Інші функції ($E(v3)$, $F(n2)$, $G(s4)$) або дорівнюють нулю, або константному значенню і тому в оптимізаційній моделі не враховуються. При побудові архітектури ПЗ необхідно прагнути до мінімізації всіх складових функції.

Виходячи з мети завдання і існуючих змінних можна зробити висновок, що знаходження оптимального значення функції Q зводиться до розв'язання задачі лінійного програмування.

Кількість і розмір екземплярів, які мінімізують функцію витрат на обчислювальні ресурси можна розрахувати, вирішивши задачу лінійного програмування, задану таблицею 3.1.

Таблиця 3.1 – Початкові дані задачі

Тип ресурсів	Тип екземпляра					Необхідна кількість ресурсів в програмі
	Надмалий	Малий	Середній	Великий	Надвеликий	
оперативна пам'ять (Гб)	0,75	1,75	3,5	7	14	A1
Швидкість ЦП (Гц)	1	1,6	2*1,6	4*1,6	16*1,6	A2
продуктивність	1	3	5	7	7	A3
кількість ядер ЦП	1	1	2	4	8	A4
Витрати (\$)	0.04	0,12	0,24	0,48	0,96	

Значення A1,A2, A3,A4 задає розробник ПЗ.

Нехай x_1 – кількість екземплярів надмалого розміру, x_2 – кількість екземплярів малого розміру, x_3 – середнього розміру, x_4 – великого та x_5 – надвеликого, тоді у загальному вигляді цільова функція має вигляд:

$$A = 0,04*x_1 + x_2*0,12 + x_3*0,24 + x_4*0,48 + x_5*0,96 \rightarrow \min$$

Обмеження:

$$\begin{cases} 0,75*x_1 + 1,75*x_2 + 3,5*x_3 + 7*x_4 + 14*x_5 \geq A1 \\ 1*x_1 + 1,6*x_2 + 3,2*x_3 + 4,8*x_4 + 6,4*x_5 \geq A2 \\ 1*x_1 + 3*x_2 + 5*x_3 + 7*x_4 + 7*x_5 \geq A3 \\ 1*x_1 + 1*x_2 + 2*x_3 + 4*x_4 + 8*x_5 \geq A4 \\ x_1, x_2, x_3, x_4, x_5 - \text{цілі числа} \end{cases}$$

При виборі схем зберігання потрібно керуватися наступними характеристиками, представленими у таблиці 3.2 .

Таблиця 3.2 – основні характеристики Azure Storage и SQL Azure.

Критерії порівняння	Azure Storage	SQL Azure
Вартість сховища	\$0.125 за ГБ/ місяць, обчислюється на основі денного навантаження.	Від 9.99\$ на місяць. Обчислюється на основі розміру БД.
Вартість транзакції	\$0.01 за 100,000 транзакцій.	\$0.00 – транзакції не оплачуються.
Оплачувані операції	Усі – До вартості сховища додається вартість транзакцій.	Ніякі – Вартість не залежить від кількості транзакцій, тільки від розміру БД.

Продовження табл. 3.2

Критерії порівняння	Azure Storage	SQL Azure
Вартість вихідного трафіку	0.12 – \$0.19 за ГБ, залежить від регіону	0.12 – \$0.19 за ГБ, залежить від регіону
Максимальний розмір запису	1 Мб – Не більше 255 властивостей, в які включені три обов'язкові: PartitionKey, RowKey, Timestamp.	2 Гб – До 1024 стовпців Використання varchar (max), varbinary (max), xml, text, i image дозволяє використовувати додатково 2 Гб.
Максимальний розмір даних	100 Тб на таблицю	150 Гб на БД
Максимальна кількість записів, що повертаються в одному запиті	1,000 Не більше 1000 записів у одному запиті. Якщо в результаті більше цієї кількості, повертається токен продовження.	Необмежено Якщо невірно налаштувати, може виникнути проблема з обмеженням кількості через таймаутів запиту та підключення.

У загальному випадку Azure Storage використовується тоді, коли застосунок зберігає та витягує(сканує, фільтрує) великі набори даних, але не вимагає їх обробки. SQL Azure використовується коли програма вимагає великих наборів реляційних даних.

Слід зазначити, що вибір схеми зберігання і використання даних також може мінімізувати і значення функції D(v2), попри те, що виміряти

імовірнісні витрати на оплату вихідних даних досить складно – об'єм переданих даних може коливатися, оскільки залежить від застосунку і запитів кінцевих користувачів.

Вибір типу архітектури

Також вартість архітектури слід враховувати і при виборі типу архітектури застосунків.

Існує кілька типів побудови архітектури застосунків, побудованих на основі технологій хмарних обчислень: виділена і мультітенантна моделі [10].

Виділена модель має окремий, логічний екземпляр програми для кожного клієнта, в той час як мультітенантна модель має один логічний екземпляр застосунка, який спільно використовується багатьма клієнтами. Важливо відзначити, що мультітенантна модель пропонує користувачам розмежований доступ до даних застосунка. У застосунку клієнт В не повинен бачити або змінювати проекти або інформацію бібліотек клієнта А.

Мультітенантна архітектура переважно актуальна для застосунків, що обслуговують групи користувачів, які мають бути ізольовані один від одного. Для того щоб не мультітенантний застосунок мав спроможність обслуговувати різних користувачів, потрібна виділена інфраструктура, як показано на рисунку 3.3а).

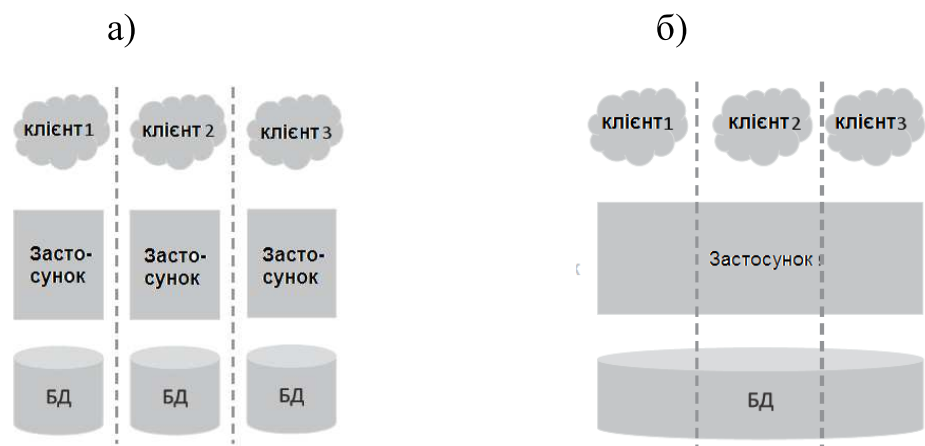


Рисунок 3.3 – Типи архітектур застосунків, побудованих на основі технологій хмарних обчислень. (а – виділена архітектура; б – мультітенантна архітектура)

Альтернативним підходом є проектування мультітенантного застосунка з вбудованими можливостями обслуговувати користувачів з різних організацій, як показано на рис 3.3б.

Існує три основні моделі реалізації мультітенантного сховища:

1. Окремі бази даних. Кожен користувач має власну базу даних з власними схемами даних. Ця модель має перевагу простоти реалізації, але якщо кількість користувачів в розрахунку на базу даних буде відносно невелика, це приведе до втрати ефективності і швидкому підвищенню вартості споживаних обчислювальних ресурсів.

2. Бази даних з різними схемами, що спільно використовуються. Усі користувачі працюють з однією базою даних, але мають різні набори зумовлених полів. Такий підхід також простий в реалізації, дозволяє збільшити кількість користувачів кожної бази даних і підвищує ефективність використання сховища. Проте зазвичай він призводить до розрядженого наповнення таблиць бази даних.

3. Бази даних, що спільно використовуються, зі спільно використововуваною схемою. Всі користувачі працюють з однією і тією ж базою даних, і для зберігання розширень даних використовуються спеціальні техніки. Перевага цього підходу тому, що він дозволяє завести практично необмежену кількість власних полів. Однак він набагато складніший у реалізації індексації, пошуку, запитів та оновленні.

З точки зору клієнта нарахування фіксованої плати за послугу означає, що клієнт заздалегідь точно знає, які будуть витрати у наступному платіжному періоді. Це також означає, що система тарифікації стає набагато простіше. Деякі витрати, такі як витрати, пов'язані зі сховищем і транзакціями, будуть змінними і будуть залежати від кількості наявних клієнтів і від того, як вони використовують службу. Інші витрати, такі як витрати на обчислені ресурси або вартість екземпляра SQL Azure, будуть чітко зафіксовані. Для того, щоб отримати підсумковий прибуток, необхідно

продати достатню кількість підписок, які покривають фіксовані та змінні витрати.

Проаналізувавши структуру і склад програмних, інформаційних і технічних засобів САПР а також існуючих моделей побудови систем автоматизованого проектування за допомогою сервісів хмарних обчислень, можна зробити висновок, що головним завданням побудови САПР у хмарі є визначення компонентів комплексу технічних засобів, перенесення яких у хмару дасть найбільш високий економічний ефект.

При побудові додатків у хмарі необхідно врахувати ряд факторів, що впливають на подальшу економічну ефективність розроблювальних продуктів, а саме:

1. Максимально мінімізувати витрати на обчислювально-технічний комплекс за рахунок відмови від купівлі дорогого обчислювального комплексу, його підтримки та експлуатації, але при цьому отримана система повинна відповідати всім вимогам, що пред'являються до технічного забезпечення сучасних САПР.

2. Розібратися в тонкощах і умовах роботи вибраного хмарного провайдера з тим, щоб правильно організувати зберігання і обробку даних в хмарі. Неефективна і непродумана організація даних інколи може не лише не дати жодного економічного ефекту розробникові, але і принести величезні збитки.

3.2 Побудова архітектури ПЗ САПР, що використовує сервіси хмарних обчислень

У даній роботі пропонується варіант побудови архітектури САПР, при якому до хмари переносяться найбільш ресурсомісткі модулі, описані розділі 1.5.3.3 – Архітектура ПЗ САПР.

Після аналізу існуючої архітектури локальних систем автоматичного проектування (рис.1), було прийнято рішення про реалізацію такої моделі

архітектури САПР, при якій у хмарне середовище переносяться модулі, що вимагають великих обчислювальних ресурсів, при цьому ефективність роботи системи не повинне постраждати. У пропонованій системі моделювання, побудова деталей, інженерні розрахунки повинні виконуватися у хмарі – це збільшить швидкодію системи і знизить витрати на покупку потрібного устаткування. Також хмару пропонується перенести максимальну кількість статичної інформації (довідкову систему, бібліотеку типових рішень і т.п). а також систему управління базами даних. Модулі, що відповідають за видове перетворення моделі і графічне виведення, було вирішено залишити на робочому місці проектувальника, оскільки це допоможе забезпечити дотримання високих вимог до якості графічних зображень в завданнях САПР а також збільшити швидкодію системи в цілому, при відображенні об'єктів на дисплеї проектувальника. Зв'язок між модулем робочої станції і хмарою здійснюватиметься шляхом обміну інформацією між хмарним і призначеним для користувача моніторами. Інтерфейс користувача, що також знаходиться на робочій станції проектувальника, забезпечує інтерактивну взаємодію з призначеним для користувача монітором, підготовку і редагування початкових даних. Структура пропонованої архітектури програмного забезпечення САПР представлена 3.4[10].

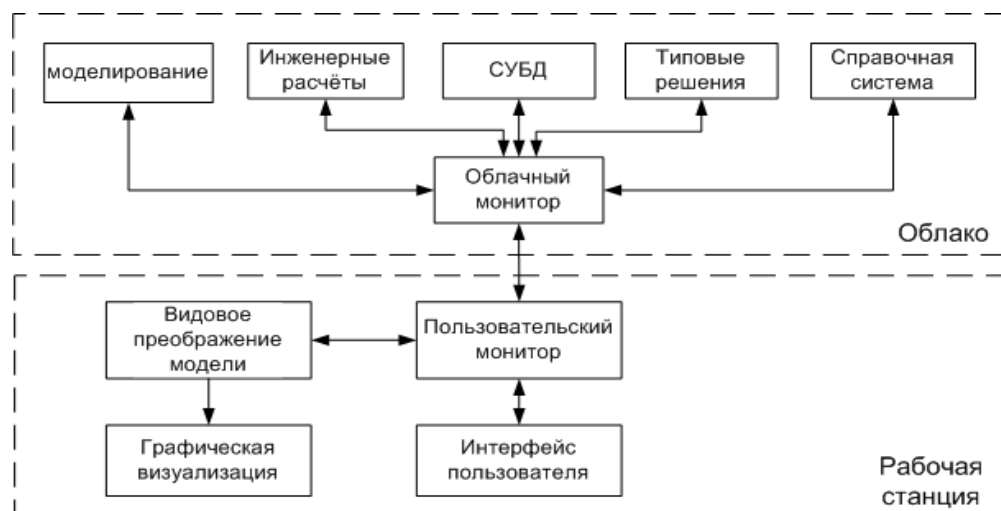


Рисунок 3.4 – Архітектура програмного забезпечення САПР, що використовує сервіси хмарних обчислень

3.3 Побудова архітектури ПЗ на основі хмарних сервісів

Програмний продукт «Астра Конструктор Мебелі» призначений для проектування і дизайну меблів і надає декілька можливих способів проектування меблів:

1. Створення виробу з деталей – при проектуванні меблів створюються деталі довільних форм і розмірів, які потім об'єднуються в секції. Для кожної деталі задаються певні властивості: розміри, товщина, матеріал, напрям волокон та ін.

2. Меблі бібліотеки – проект будується на основі типових виробів, узятих з бібліотеки типових виробів. Для цих виробів також можна задавати або змінювати їх властивості. Отримані готові вироби можна заносити в цю бібліотеку і, надалі, користуватися ними як стандартними типовими виробами.

«Астра Конструктор Мебелі» дозволяє проектувати меблі і дизайн інтер'єру у будь-якій з проекцій(план, фронт, вигляд збоку) або у видах Перспектива і Аксонометрія, а також обертати сцену і переглядати її під довільними точками зору.

У програмі здійснена можливість автоматичної установки кріпильних елементів на деталі виробу і їх ручної корекції. Каталог кріплень можна необмежено поповнювати елементами типу : конфірмат, мініфікс, рафікс, стягування BLUM та ін.

Для готового проекту автоматично формується комплект креслень: аксонометричний вид виробів і креслення деталей. На кресленнях деталей позначені координати отворів під кріплення і їх властивості. Здійснена можливість ручного редагування креслень (додавання / видалення розмірів і текстових написів).

Програма забезпечує реалістичний перегляд проекту з урахуванням текстур матеріалу і перспективного відображення, з можливістю швидкої зміни планування приміщення, текстури матеріалу і т.д.

Для готового проекту "Астра Конструктор Мебели" формуються такі звіти як: специфікація деталей проекту, відомість замовлення і відомість витрати матеріалів.

Всі перелічені вище функції являються ресурсозатратними і для коректного виконання вимагають значних обчислювальних ресурсів від робочих місць проектувальників.

Програма поширюється серед користувачів на умовах ліцензійної угоди. Підтримує тільки операційну систему Windows.

При встановленні цього програмного продукту на локальній машині проектувальника створюються дві директорії: "Проекти" і "Бібліотека". У першій директорії знаходяться усі збережені файли проектів користувача. У "Бібліотеці" знаходиться інформація про доступні вироби і елементи, представлена файлами з розширенням .lba а також інформація про матеріали, фасади, текстури, кромки і так далі, представлена графічними файлами і файлом опису.

Головними перевагами використання клієнтами нової моделі системи є:

1. видалене використання продукту (можливість проектування і використання власних бібліотек з будь-якої точки світу, де є доступ до мережі Інтернет);
2. використання завжди найновіших версій продукту;
3. можливість разового використання продукту, у випадку якщо розробникові потрібно усього лише кілька разів в рік/місяць використати можливості цього програмного продукту – тепер йому не доведеться оплачувати вартість усього пакету, він платить лише за ті послуги, якими фактично користувався, що являється економічно вигідніше за перший варіант використання ПЗ.
4. Застосунки в хмарі мають набагато більшу функціональність, ніж локальні, із-за використання певних хмарних сервісів, що надаються провайдерами хмарних послуг.

5. Значне зниження вимог до технічного забезпечення користувача – для використання цього програмного продукту, не доведеться спеціально купувати потужну обчислювальну техніку.

6. Зниження вимог до адміністрування програмних продуктів(налаштування програм, установка, переустановлення ОС і т.д).

Також важливою метою для фірми-розробника є підвищення конкурентоспроможності її продуктів серед інших розробників САПР як на внутрішньому, так і на зовнішньому ринках України :

1. За рахунок того, що використання цього програмного продукту відбуватиметься тільки по підписці, не буде можливості для виникнення піратських копій, які частенько віднімають величезний шматок прибутку компанії-розробника.

2. Можливість створювати проекти, аналогічні по своїй складності і можливостям проектам провідних фірм-розробників, зі значно меншими засобами на купівлю, підтримку, обслуговування і експлуатацію обчислювальних засобів є економічно-привабливим для користувачів, що відповідає одній з головних цілей компанії-розробника – підвищення задоволеності користувачів її застосунками.

3. Можливість інтерактивного спілкування з користувачами, що дозволить більше детально вивчити їх потреби і побажання, дізнатися про проблеми в роботі з цим ПЗ, якщо такі виникають у клієнтів.

4. Локальна версія продукту доступна тільки користувачам операційної системи Windows. Хмарна версія продукту буде платформонезалежною, що дозволить розширити круг користувачів цього продукту.

5. Можливість додавання нового функціоналу, не доступного для реалізації в застосунках, працюючих локально.

Завдання системи

Перед системою, що розробляється, стоїть декілька завдань:

1 Хмарне застосування, що розробляється, повинне реалізовувати увесь функціонал локального застосування, при цьому доступ до нього повинен відбуватися через веб-браузер, встановлений на клієнті. Архітектура застосування повинна відповідати принципам, описаним в розділі 3.4 — Побудова архітектури ПЗ САПР, що використовує сервіси хмарних обчислень.

2 Окрім основного функціонала, розробники надають можливість скористатися вже готовими проектами, що розташовуються у бібліотеці готових рішень. Оскільки ця функція є платною, то збереження проекту на локальний комп'ютер або копіювання у буфер обміну без попередньої оплати неможливе. Для доступу в цю бібліотеку користувачеві у своєму локальному застосуванні необхідно оновити інформацію про проекти, що представляється у вигляді іконок готових проектів (посилається запит в хмару, в запиті у відповідь надсилаються усі доступні іконки готових рішень). При виборі однієї з іконок дані про іконку вирушають в хмару, де відбувається побудова відповідної моделі, далі ці моделі вирушають в локальну програму, де відбувається її видове перетворення в новому проекті, що створився. При зміні користувачем моделі, що відобразилася, дані про кліки миші посилаються до хмари, де відбувається перерахунок моделі, після чого в запиті у відповідь посилаються тільки змінені дані, тобто усі операції обчислення і обробки даних відбуваються засобами хмари.

У разі, якщо користувач захоче придбати цей проект для подальшої роботи з ним (розкрий і тому подібне), він посилає відповідний запит в хмару, у відповідь на який з хмари йому приходить файл проекту разом з даними про матеріали : кромках, фурнітурі і т.д, якщо вимагається (у разі, якщо дані про матеріали відмінні від стандартних, таких, що поставляються в програмах для локального використання).

3.4 Концепція побудови систем проектування меблів з використанням технологій хмарних обчислень

У результаті порівняння хмарних платформ, виявилось, що найбільш відповідною платформою "хмарних сервісів" для цілей компанії-розробника є платформа Windows Azure від Microsoft. Тому в якості платформи для архітектури ми використовуємо Microsoft Windows Azure і такі його компоненти як:

- Web Role – екземпляр застосунку в Azure для запуску веб-проектів;
- Table Storage – сховище даних в Azure дозволяє зберігати структуровані дані;
- Blob Storage – інше сховище даних в Azure, але вже для збереження файлів;
- Worker Role – екземпляр застосунку або сервісу, якому не потрібний веб-інтерфейс;
- Queue – хмарна черга повідомлень для сервісів Azure;
- Azure SQL – хмарна SQL база даних, яка заснована на сервері MS SQL.

На рис.3.1 показано високорівневе представлення архітектури, що розробляється.

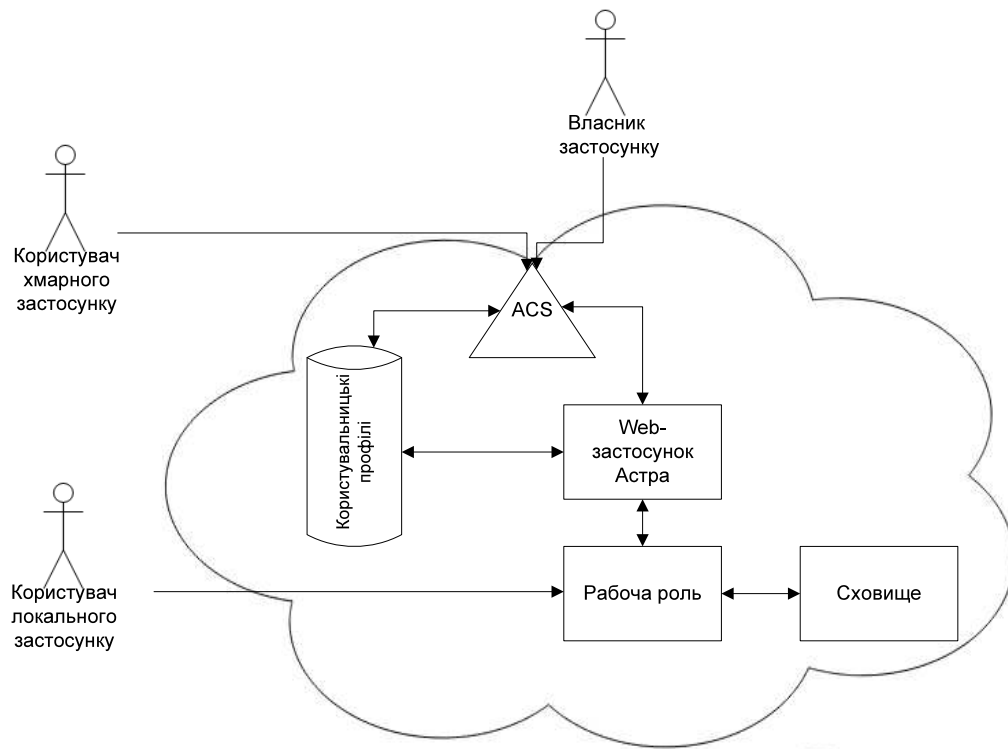


Рисунок 3.1 – Високорівневе представлення архітектури застосунку
Айстра

На наведеному рисунку показані три групи користувачів, що звертаються до застосунку: власник застосунку, користувачі хмарного і локального застосунків. Для власника застосунку і користувачів, що використовують застосунок через веб-браузер потрібна перевірка достовірності і авторизації в застосуванні. Для цих цілей в застосуванні використовується служба Access Control Services(ACS).

ACS працює на підставі набору тверджень, що надаються користувачем (рис. 4.2). Одно твердження може бути ім'ям користувача, інше – адресою його електронної пошти, і так далі. Ці твердження надаються службою сертифікації, яка знає, як аутентифікувати користувача і де узяти його атрибути. Клієнтське застосування, в ролі якого може виступати браузер або насичений клієнт, прозоро взаємодіє з цією службою, отримуючи ці твердження і передаючи їх у застосування. Зрештою, застосування отримує усі необхідні посвідчення користувача у вигляді

набору тверджень. Крім того, твердження підписані, що забезпечує криптографічну гарантію їх походження.



Рисунок 3.2 – надання твердження користувачем

Увесь процес авторизації і аутентифікації представлений діаграмою на рисунку 3.3.

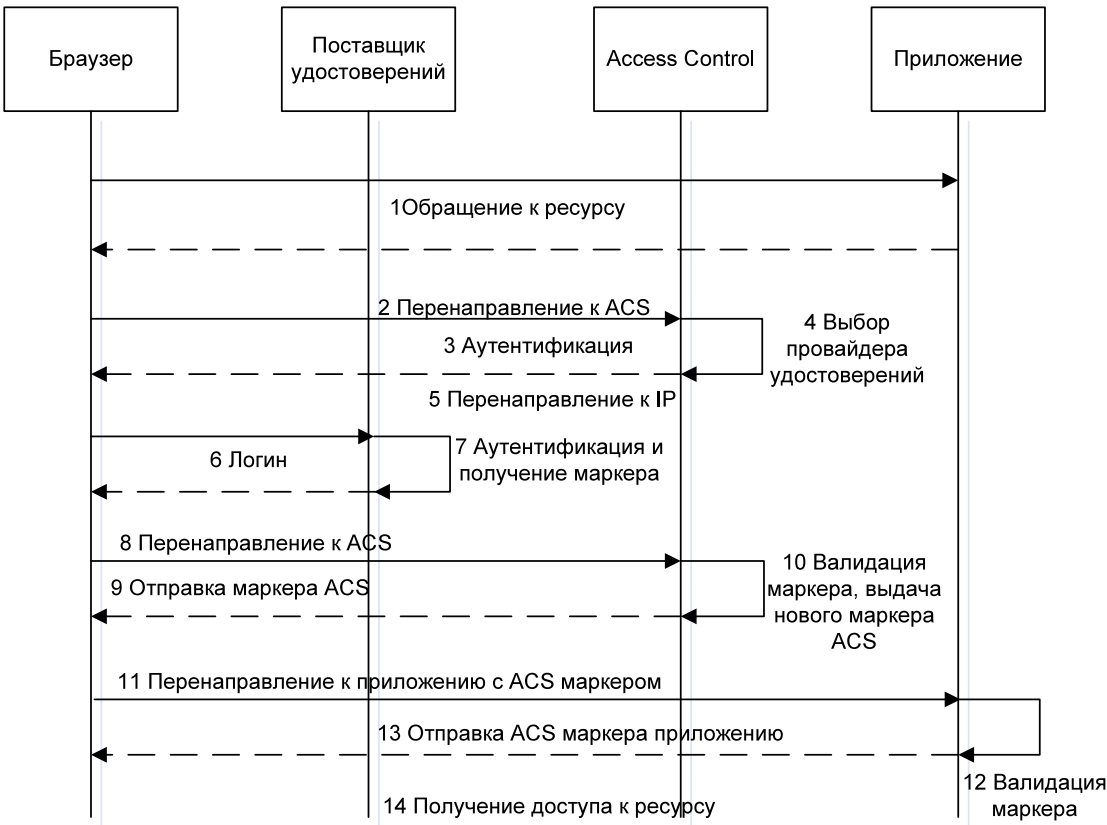


Рисунок 3.3 – Схема последовательностей при авторизації та аутентифікації

3.5 Інформаційне забезпечення

У застосуванні необхідно організувати зберігання інформації про наступні дані:

1 дані користувачів(інформація про самого користувача і файли його проектів) ;

2 файли готових проектів і файли, представлені у бібліотеці стандартних рішень.

Оскільки кожен авторизований користувач застосування має ідентичний набір даних і має доступ до однієї бібліотеки стандартних рішень і бібліотеки готових проектів, то було прийнято рішення про організацію загального мультитенантного застосування з мультитенантним сховищем, що використовує різні для кожного користувача схеми використання. Загальна архітектура застосування представлена на рисунку 3.4.

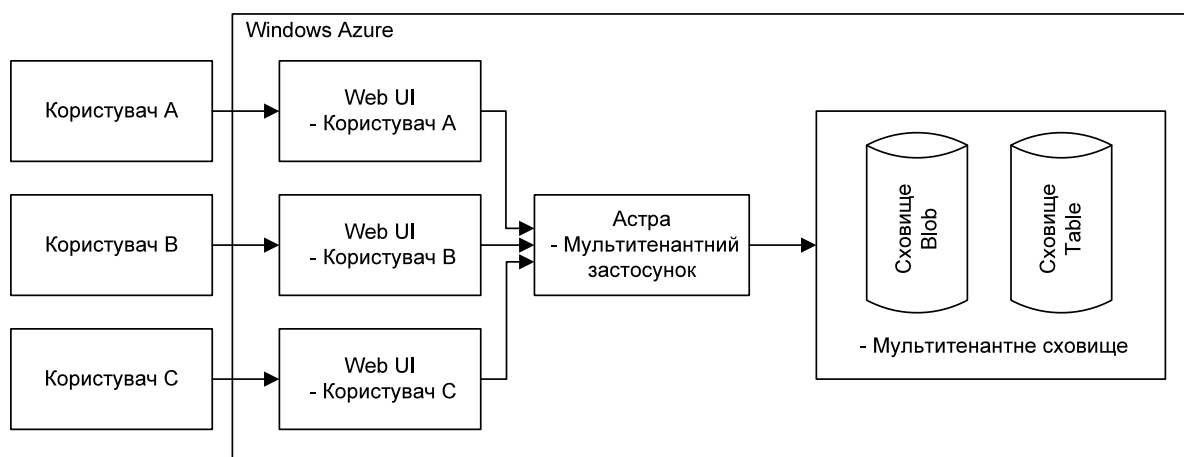


Рисунок 3.4 – Мультитенантня архітектура приложения

Для зберігання даних застосування пропонується використати наступні служби, Windows Azure, що надаються :

1 Table Storage – для зберігання структурованих даних. Таблиця складається з набору об'єктів, кожен з яких має набір назв властивостей і їх значень. Один об'єкт може мати до 256 властивостей. У одній таблиці може зберігатися до терабайта даних.

2 Blob Storage – для зберігання великих двійкових об'єктів. Об'єкти blob зберігаються в контейнерах Blob Container і їх зона дії обмежена цими контейнерами. Кожен blob може бути розміром до 50ГБ і мати унікальне у рамках контейнера строкове ім'я. З blob або контейнером Blob можуть бути асоційовані метадані, які задаються у вигляді пар <ім'я, значення> і можуть

досягати розміру 8KB для blob. Метадані blob можуть бути отримані і задані окремо від даних blob.

Інформація про усі проекти застосування зберігається в таблиці Project. Усі інші дані – в сховищі Blob (рис 3.5):

1. Файли, створені самим користувачем (доступні тільки авторові)

1.1 Самі файли проектів в контейнері Files_User

1.2 Файли використовуваних матеріалів в контейнері Materials_User

1.3 Файли готових елементів в контейнері Elements_User

1.4 Файли іконок проектів в контейнері Icons

У разі, якщо автором проекту виступає власник застосування, то файли проекту розміщуються в контейнері Files_ReadyMade – це файли готових рішень, використання яких можливе тільки після купівлі. Доступ до певних об'єктів цього контейнера відкривається тільки у разі попередньої оплати цієї послуги.

2 Файли бібліотеки стандартних рішень (загальнодоступні файли)

2.1 Файли використовуваних матеріалів в контейнері Materials

2.2 Файлів готових елементів в контейнері Elements

До контейнерів Materials, Materials_User, Elements і Elements_User мають бути асоційовані метадані, які пояснюють застосунку правила використання цих файлів програмою.

3 Файли власника застосунку (доступні тільки власникові застосунку)

3.1 Дані користувача – контейнер Tenants.

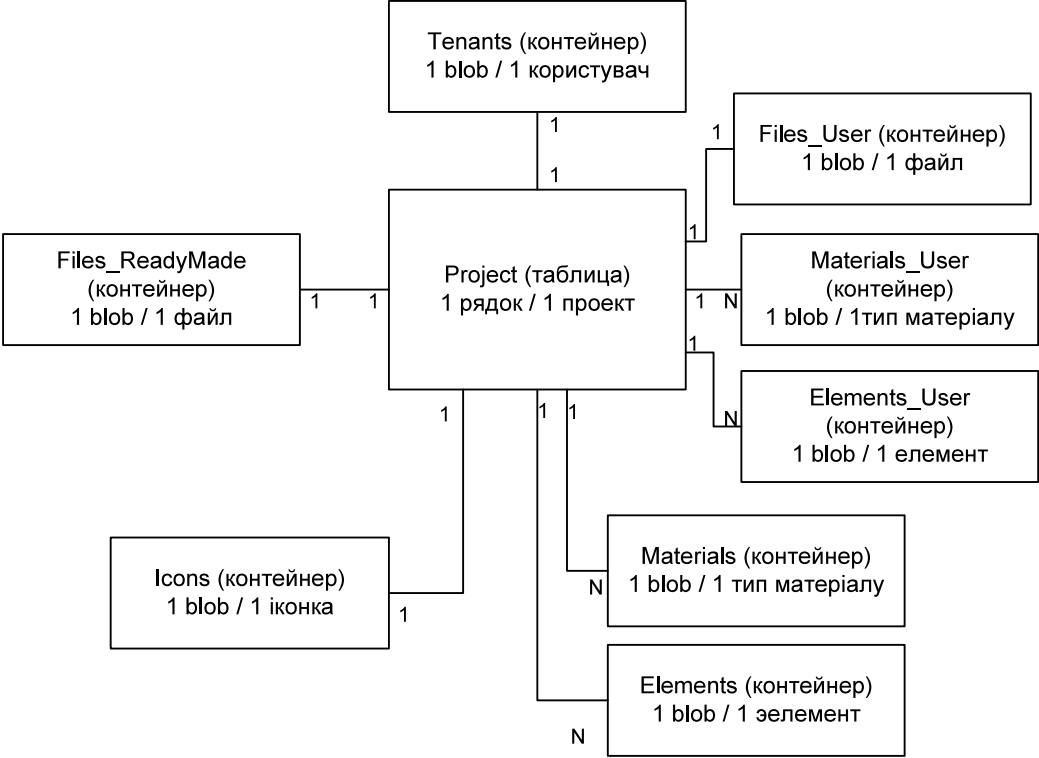


Рисунок 3.5 – Модель даних для мультитенантного застосунку

У таблиці 3.1 розглянуті поля таблиці Project. Ця таблиця містить список усіх проектів користувача.

Таблиця 3.1 – Опис полів таблиці Project

№	Ім'я поля	Тип поля	Примітки
1	PartitionKey	String	Ключ секції – містить ім'я клієнта. Введено для можливості швидкої фільтрації по імені клієнта.
2	RowKey	String	Ключ рядка. Унікальний ID суті у рамках секції.
3	Timestamp	Date Time	Значення виставляється автоматично табличним сховищем Windows Azure. (Версія сутності, що зберігається системою)
4	SlugName	String	Ім'я проекту в "склеєному вигляді"
5	CreatedOn	Date Time	Дата створення. (Якщо користувач робив зміни в проекті, то значення цього поля відрізнятиметься від значення поля Timestamp)
6	State	String	Стан моделі при редагуванні проекту. (Відносно цього стану обчислюються параметри, що відправляються користувачеві для відображення моделі, при зміні/редагуванні проекту).

Контейнер Tenants містить наступну інформацію:

- (string) ClaimType – тип твердження
- (string) ClaimValue – значення твердження
- (string) Name – ім'я користувача
- (string) DatabaseName – ім'я бази даних
- (string) DatabaseUserName – ім'я бази даних користувача
- (string) DatabasePassword – пароль до бази даних

На рисунку 3.6 представлено загальне представлення сховища Blob для кожного облікового запису користувача.

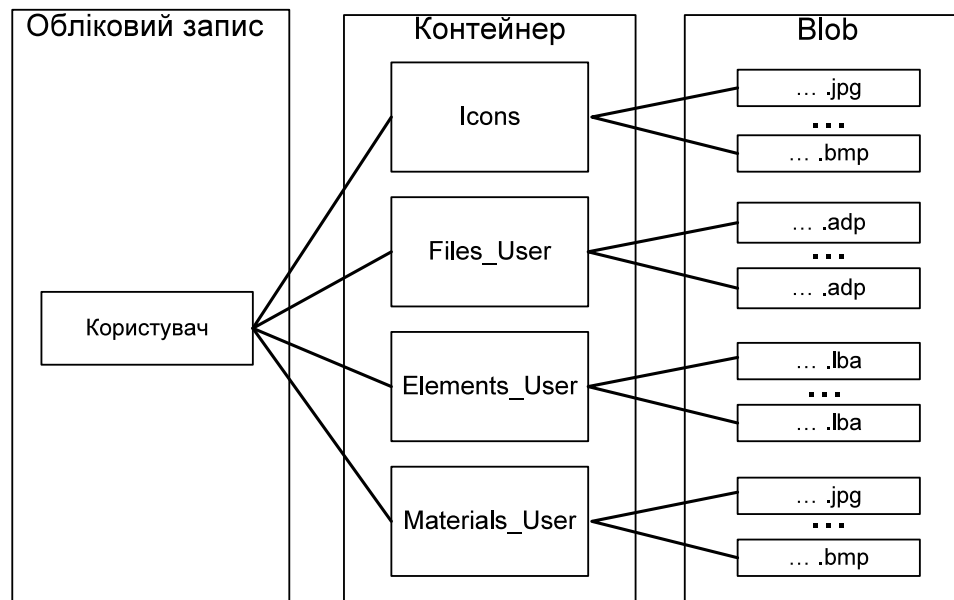


Рисунок 3.6 – Загальне представлення сховища Blob для кожного облікового запису користувача

Для зберігання цих призначених для користувача профілів, в застосуванні використовуватиметься табличне сховище. Кожен профіль містить таку інформацію :

- Ім'я користувача (передплатника).
- Аліас користувача (передплатника).

- Тип підписки (стандартна, пробна) – тип підписки визначає набір доступних передплатникові функцій і може впливати на список додаткової інформації, потрібної від передплатника для адаптації.

- Спосіб оплати – відомості про кредитну карту.

Задамо наступні правила обмеження для робочої ролі застосування, що розробляється:

1 Перше правило обмеження завжди активне. Воно задає для робочої ролі мінімальну кількість екземплярів, яка дорівнює двом, і максимальне – рівне трьом. Це правило має найнижчий ранг.

2 Друге правило обмеження активне кожену добу з 12: 00 до 16: 00 (мітка А на рис 3.7 показує, як це правило стає активним уперше). Воно задає для робочої ролі мінімальну кількість екземплярів, що дорівнює двом, і максимальне –чотирьом. Це правило має низький ранг.

3 Третє правило обмеження активне в проміжках, починаючи з 15 листопада до 31 грудня і з 15 липня до 31 серпня. (мітка В на рис 3.7 показує, як це правило стає активним уперше). Воно задає для робочої ролі мінімальну кількість екземплярів, що дорівнює трьом, і максимальне – чотирьом. Це правило має високий ранг.

4 Четверте правило обмеження активне з 15 листопада до 31 грудня і з 15 липня до 31 серпня в проміжках між 12: 00 і 16: 00 годинами (мітка С на рис 3.7 показує, як це правило стає активним уперше). Воно задає для робочої ролі мінімальну кількість екземплярів, що дорівнює трьом, і максимальне –п'яти. Правило має найвищий серед чотирьох правил ранг, тому при одночасному застосуванні з іншими правилами отримує пріоритет над ними.

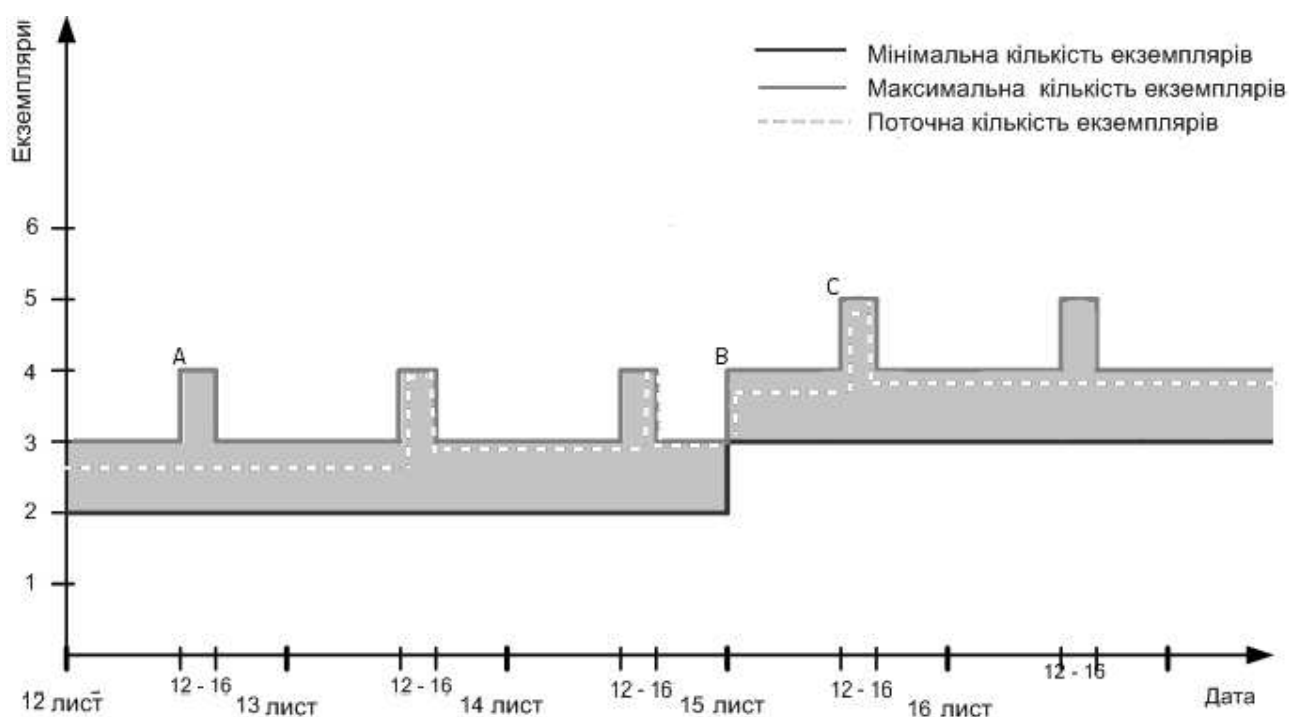


Рисунок 3.7 – Використання декількох правил обмеження за відсутності правил реагування.

На рисунку 3.8 показана поведінка блоку автоматичного масштабування застосунку, якщо на додаток до декількох правил обмеження визначені правила реагування.

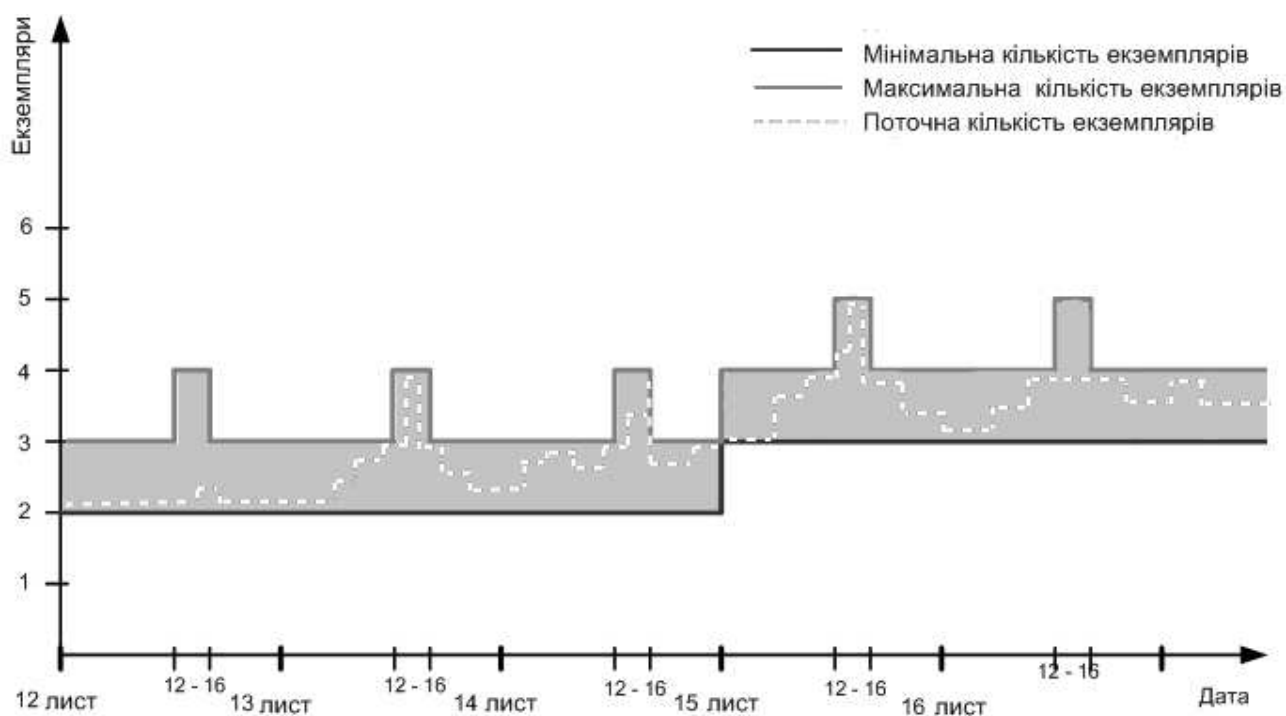


Рисунок 3.8 – Правила обмеження, які взаємодіють з правилами

Необхідні інструменти для запуску застосування

Для запуску застосування "Айстра" потрібні інструменти, представлені таблицею 3.2.

Таблиця 3.2 – Необхідні інструменти для запуску застосування

№	Інструмент	Опис
1	Visual Studio 2010 з пакетом оновлення 1(SP1)	середовище розробки
2	MVC 3 Framework	фреймворк для створення веб-застосунків, який реалізує шаблон Model – view – controller.
3	пакет SDK Windows Azure для .NET	набір засобів, документації, компіляторів, заголовків і бібліотек, які можуть використовуватися розробниками при створенні програм, працюючих в операційних системах Microsoft Windows з використанням власної (Win32) або керованої (.NET Framework) моделей програмування.
4	Windows Identity Foundation	середовище виконання – платформа для побудови застосунків, що підтримують посвідчення.
5	Microsoft Internet Information Services 7 (IIS)	пропріетарний набір серверів для декількох служб Інтернету від компанії Майкрософт (необхідно для роботи сайту управління в режимі імітації або для розгортання Autoscaler локально).

Висновок до розділу 3

Не завжди можна спрогнозувати, в який момент часу зросте потреба в конкретному застосуванні або коли виникне її різкий сплеск, тому задамо для застосування, що розробляється, декілька правил реагування, що коригують кількість екземплярів робочої ролі :

1 Якщо середнє значення лічильника продуктивності навантаження ЦП протягом останньої години перевищує порогове значення 80%, збільшити кількість екземплярів робочої ролі на одиницю.

2 Якщо середнє значення лічильника продуктивності навантаження ЦП протягом останньої години впаде нижче порогового значення 40%, зменшити кількість екземплярів робочої ролі на одиницю.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПЗ ДЛЯ РОЗРОБКИ САПР

4.1 Вступ

На даний момент сучасні обчислювальні засоби широко використовуються в повсякденній діяльності людини. Сьогодні обчислювальна техніка дозволяє вирішувати найрізноманітніші задачі: від звільнення людини від монотонної ручної праці до розробки складних технічних розрахунків, що не під силу виконати одній людині. Сучасні засоби програмування дозволяють вирішувати задачі, що стояли в недавньому минулому перед цілими колективами наукових співробітників.

Розвиток нових технологій і ріст застосування обчислювальної техніки привели до широкого застосування ЕОМ для вирішення проблем, пов'язаних з обробкою інформації, її контролем і аналізом. Сьогодні на основі засобів обчислювальної техніки розробляються і впроваджуються різні автоматизовані інформаційні системи (управління, проектування, технологічної підготовки виробництва). Успіх у впровадженні цих систем, їхня роль в інтенсифікації розвитку народного господарства нашої країни багато в чому залежить від фахівців із системотехніки, які знають методику аналізу і проектування цих систем, можливості обчислювальної техніки і які володіють математичними методами, що використовуються при постановці та вирішенні задач.

Використання даного програмного продукту приведе до значної економії, що виражається в наступних показниках:

- підвищення продуктивності праці;
- прискорення обробки інформації;
- економія часу на обробку інформації.

Найбільш важливим моментом для розробника, з економічної точки зору, є процес формування вартості програмного продукту. Очевидно, що вона являє собою дуже специфічний товар з безліччю особливостей.

Створення автоматизованої системи вимагає одноразових витрат на її розробку, придбання необхідних технічних засобів і поточних витрат на функціонування системи. Економія від функціонування автоматизованої системи визначається з урахуванням витрат на її експлуатацію. Відношення цієї економії до витрат на створення автоматизованої системи характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

4.2 Розрахунок витрат на створення й експлуатацію інформаційної системи

Витрати на розробку системи складаються з витрат на зарплату розробника, на амортизацію ЕОМ, на якій виконується розробка, на експлуатацію цієї ЕОМ, на засоби розробки та витрат на матеріали і комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячний оклад якого складає 2000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 2200 грн/міс, а вартість сучасної ПЗВМ складає 5000 грн. (приведена вартість машини на базі Pentium IV). При вартості кіловат-години електроенергії рівної 0.24 грн., розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 8.1

Таблиця 4.1 - Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	45.00
Заправлення картриджа до принтера	1.00
Диск	3.00
Література	20.00
Непередбачені витрати	100.00
Разом матеріали і комплектуючі вироби.	169.00

Вартість програми розрахуємо по формулі:

$$C_{\text{пр}} = (Z_{\text{зп}} + Z_{\text{сз}} + Z_{\text{зг}} + Z_{\text{е}}) * T + Z_{\text{м}} \quad (4.1)$$

де T – тривалість розробки, мес.

$Z_{\text{зп}}$ - основна і додаткова заробітна плата обслуговуючого персоналу, грн.

$Z_{\text{сз}}$ - відрахування на соціальні заходи (38% від основної і додаткової заробітної плати), грн.

$Z_{\text{зг}}$ - загальногосподарські витрати (50% від основної заробітної плати), грн.

$Z_{\text{е}}$ - витрати на електроенергію.

$Z_{\text{м}}$ - витрати на основні і допоміжні матеріали.

$Z_{\text{е}}$ при споживанні потужності 0.25 Вт, тривалості роботи на місяць, рівної $21 * 6 = 126$ годин, вартості кіловат-години електроенергії 0.24 грн.,

$$Z_{\text{е}} = 126 * 0.24 * 0.25 = 7.56 \text{ грн.}$$

Таблиця 4.2 - Витрати на розробку системи.

Найменування витрат	Одини	Кількіс
Тривалість розробки	Мес	5
Основна і додаткова заробітна	Грн.	900
Відрахування на соціальні	Грн.	128
Загальногосподарські витрати	Грн.	50
Витрати на основні і допоміжні	Грн.	122
Витрати на електроенергію	Грн.	7.56

Відповідно до формули 6.1 вартість програми:

$$C_{\text{пр}} = (2200 + 128 + 50 + 7.56) * 2 + 165 = 3636.12 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$A_{\text{об}} = 5000 * 0,6 = 3000 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали (гнучкі диски, папір) визначаються в розмірі 5% вартості основного устаткування:

$$B_{\text{м}} = 5000 * 0,05 = 250 \text{ грн.}$$

Річний обсяг робіт ПЗВМ у годинах визначається в такий спосіб:

$$\Phi_M = 253,3 * T_3 \quad (6.2)$$

де T_3 - це середнє місячне завантаження устаткування (близько 4 годин)

253,3 - середня кількість робочих днів у році.

Отже, річний обсяг роботи ПЕОМ складе:

$$\Phi_M = 253,3 * 4 = 1013 \text{ годин}$$

Витрати на електроенергію $З_e$ при 1013 годинах роботи устаткування в рік складуть

$$З_e = 1013 * 0,24 * 0,25 = 60,78 \text{ грн.}$$

Експлуатаційні витрати для ПЗВМ за рік складуть:

$$З_{зр} = 3000 + 250 + 60.78 = 3310.78 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програми складуть:

$$З_{се} = 2336.12 + 3310.78 = 5646.9 \text{ грн.}$$

4.3 Економічна ефективність розробки і впровадження системи

Основним показником економічної ефективності функціонування системи є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

- високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);
- підвищені витрати на канцтовари;
- витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться:

- підвищення продуктивності праці;

– вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 0,6 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 0,6 працівника в рік складає:

$$(2000 * 12) * 0.6 = 14400 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$\mathcal{E}_{\text{год}} = \Delta C_n - E_n \cdot k,$$

де ΔC_n — вивільнені кошти після впровадження програмного продукту (14400 грн.) мінус експлуатаційні витрати (3310.78 грн.) = 11089,22 грн.

E_n — коефіцієнт ефективності (дорівнює коефіцієнту амортизації (0,6));

k — одноразові витрати на впровадження продукту (2336.12)

$$\text{€ год} = 11089,22 - 2336,12 * 0,6 = 9687,548 \text{ (грн.)}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{2336.12}{3310.78} \approx 0,71 \text{ (року)}$$

Отже строк окупності системи складає приблизно 8,5 місяців.

5 ОХОРОНА ПРАЦІ

5.1 Загальні відомості

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці (ст. 1 Закону "Про охорону праці"). Як правовий інститут, охорона праці – це чималий комплекс правових норм.

Ст. 4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і як обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своєї політики в майбутньому. З урахуванням вкладеного, принципи, зазначені в ст.4 Закону "Про охорону праці", мають певне регулятивне значення.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовується працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

Охорона праці - один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну, відповідальність за порушення правил охорони праці.

5.2 Аналіз шкідливих та небезпечних факторів в офісі фірми

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загостило проблеми збереження власного та суспільного здоров'я, вимагає вдосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку, негативних наслідків впливу ПК на здоров'я користувачів.

Шкідливі та небезпечні фактори діляться на:

- а) Фізичні;
- б) Хімічні;
- в) Біологічні;
- г) Психофізіологічні.

Для робітників кафедри можна виділити наступні фактори, що негативно впливають на здоров'я людини:

а) небезпечний фізичний фактор - ураження електричним струмом при взаємодії з токопроводимими частинами обладнання;

б) шкідливі фізичні фактори:

- 1) низькочастотне електромагнітне випромінювання монітора;
- 2) недостатня освітленість робочого місця;
- 3) температура зовнішньої середовища;
- 4) шум;

в) шкідливі психофізіологічні фактори.

- 1) постійне напруження зору;
- 2) постійне статичне навантаження;
- 3) нервово-психічне перевантаження;
- 4) напружена зорова робота;

Освітлення - отримання, розподіл та використання світлової енергії для забезпечення нормальних умов праці. Таким чином, освітлення повинне бути достатнім, тобто відповідати вимогам санітарних норм, рівномірним,

не повинно осліплювати очі ті не створювати відблиски на робочій поверхні, за спектральним складом має наближатись до сонячного світла, Оптимальним вважається таке освітлення, при якому втома зору найменша. Менш допустимою величиною освітлення вважається така, нижче якої відбувається порушення зорової функції.

Дія світла на очі людини викликає трансформацію енергії електромагнітного випромінювання у нервові імпульси, які по зоровому нерву передаються в головний мозок - до зорової зони кори великих півкуль.

Слід зазначити, що світло має надзвичайно важливе значення для життєдіяльності людини. Воно впливає на стан вищих психічних функцій та на фізіологічні процеси в організмі людини. Відомо, що нормальне освітлення тонізуюче діє, викликає гарний настрій, покращує діяльність вищої нервової системи людини.

Рациональне влаштоване освітлення виробничих приміщень позитивно впливає на працівників, підвищує ефективність та безпеку праці, знижує втому та травматизм, забезпечує високу працездатність.

Санітарні та гігієнічні норми щодо виробничого освітлення, котрі базуються на психологічних особливостях сприйняття світла та його вплив на організм людини, можуть бути зведені до наступного:

- а) Спектральний склад світла, який забезпечується електричним джерелом, повинен бути близьким до сонячного;
- б) Рівень освітлення повинен бути достатнім і відповідати гігієнічним нормам, які враховують зорові роботи;
- в) Освітлення повинно забезпечувати рівномірність та стабільність світла у приміщенні, щоб запобігти частоті переадаптації та втоми зору.

Гігієнічними нормами вимагається максимально використовувати природне освітлення, оскільки денне світло краще сприймається органами зору.

Штучне освітлення у виробничих та адміністративно-громадських приміщеннях має здійснюватись системою загального рівномірного освітлення, допускають застосування системи комбінованого освітлення.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк. Як джерела світла для штучного освітлення мають застосовувати переважно люмінесцентні лампи типу ЛБ.

Шум, як фізичне явище, являє собою хвильові коливання матеріальних тіл - твердих, газоподібних або рідких. Виникнення звукових відчуттів людини пов'язане з коливаннями повітря.

Шум - один з найнесприятливіших факторів на підприємствах, що знижує працездатність робітників, їх уважність і створює передумови для виробничого травматизму та професійних захворювань.

Шум, як фізіологічне явище, являє собою несприятливий фактор для людини, викликає гіпертонію, глухоту, розлад центральної нервової системи, злоякісні пухлини тощо.

Інтенсивність шуму та вібрації є тими шкідливими факторами виробничого середовища, що негативно впливають не лише на слух, але і на нервову, серцево-судинну та інші системи організму. Судинні розлади та захворювання нервової системи в більшій мірі обмежують працездатність людини, ніж зниження слуху.

Виходячи з гігієнічних нормативів, витікає, що знижувати треба не будь-яку віброакустичну активність машин, а тільки ту, яка шкідливо впливає на людський організм, тобто за рівнем вище за гранично допустимий норматив. Таким нормативом є гранично допустимий спектр шуму та вібрації. Саме граничний спектр може бути взятий за «умовний віброакустичний нуль» (УВАН). УВАН - такий рівень віброакустичної активності машини і процесу, який забезпечує віброакустичну нешкідливість і безпеку.

Усунення промислового шуму може бути досягнуто лише в результаті проведення комплексу заходів, особливо в галузі вдосконалення

технологічних процесів і обладнання. При цьому, в першу чергу, повинні бути усунені найсильніші джерела шуму. Знешумлення окремої одиниці обладнання в загальному технологічному процесі при великій кількості комп'ютерів, що продовжують випромінювати шум, недоцільно. Необхідне знешумлення тих комп'ютерів, шумові характеристики яких виходять за рамки вимог санітарних норм.

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот мають відповідати вимогам СН 3222-85, ГОСТ 12.1.003-85, ГР 2411-81. Вони наведені в таблиці 5.1.

Таблиця 5.1 – Допустимі рівні звуку

Вид трудової діяльності	Рівні звукового тиску в дБ октавних смугах із середньо геометричними частотами, ГЦ									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБа/дБа _{екв}
Програмісти ЕОМ	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ЕОМ ті оператори комп'ютерного набору	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів ЕОМ	103	91	83	77	73	70	68	66	64	75

Для нормальної життєдіяльності людини важливим є забезпечення нормальних метеорологічних умов у виробничих приміщеннях, котрі суттєво впливають на самопочуття людини.

В поняття «метеорологічні умови», або «мікроклімат», виробничих приміщень входять ті фізичні фактори виробничого середовища, які

впливають на тепловий стан організму і котрі необхідно постійно контролювати. Це температура, вологість, швидкість руху повітря, барометричний тиск і теплове випромінювання. Від стану виробничого приміщення залежить самопочуття і здоров'я людини.

Якщо кількість кисню в повітрі зменшиться до 12% то утруднюється дихання. В таких умовах людина напружує дихальний апарат, дихає частіше; такий стан людина витримує до 0,5 години.

Перегрівання організму відбувається за умов надлишкового конвективного випромінювання тепла нагрітих поверхонь. Розрізняють 2 фази перегрівання. Перша - фізіологічна, при котрій вступають в активну реакцію пристосувальні функції організму. При цьому активізується робота серцево-судинної та дихальної систем, відбувається інтенсивне потовиділення, котре сягає 5л за зміну. З потім втрачається велика кількість мінеральних солей та вітамінів.

Охолодження організму можливе найчастіше в зимовий і перехідний періоди року при виконанні робіт на відкритому повітрі, або робіт, які проводяться в неопалюваних виробничих і складських приміщеннях.

Тривала дія низької температури може викликати різні небажані зміни в організмі людини. Охолодження пов'язане зі значними тепловиділеннями через кінцівки рук і ніг. На охолодження впливає вологість і швидкість руху повітря.

Вологість повітря визначається ступенем його насиченості водяною парою.

Вологість повітря суттєво впливає на терморегуляцію людського організму. Підвищення відносної вологості повітря у виробничому приміщенні ускладнює терморегуляцію, зменшення тепловиділення організмом. Фізіологічно оптимальною є відносна вологість в межах 40..60%. Норми мікроклімату для приміщень з ЕОМ представлені в таблиці 5.2

Таблиця 5.2 – Норми мікроклімату для приміщень з ЕОМ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка – 1а	22-24	40-60	0,1
	Легка – 1б	21-23	40-60	0,1
Тепла	Легка – 1а	23-25	40-60	0,1
	Легка – 1б	22-24	40-60	0,2

Праця працівників невиробничої сфери характеризується тривалою багатогодинною працею в одноманітному напруженому положенні, малою руховою активністю при значних локальних динамічних навантаженнях.

Робоче положення «сидячи» супроводжується статичним навантаженням значної кількості м'язів ніг, плечей, шиї та рук, що дуже втомлює. М'язи перебувають довгий час у скороченому стані і не розслаблюються, що погіршує кровообіг. В результаті, виникають больові відчуття в руках, шиї, верхній частині ніг, спині та плечових суглобах. Внаслідок динамічного навантаження на кістково-м'язовий апарат кистей рук виникають больові відчуття різної сили в суглобах та м'язах кистей рук; оніміння та уповільнена рухливість пальців; судоми м'язів кисті; ниючий біль в ділянці зап'ястя.

У результаті, виникають локальні м'язові перенапруження, хронічні розтягнення м'язів травматичного характеру, що можуть викликати професійні захворювання: дисоціативні моторні розлади, захворювання периферійної нервової та кістково-м'язової систем.

Крім того, робота «сидячи» призводить до зниження м'язової активності - гіподинамії. За браком рухів відбувається зниження споживаного кисню тканинами організму, сповільнюється обмін речовин. Це сприяє розвитку атеросклерозу, ожиріння, може стати причиною дистрофії міокарда, хронічного головного болю, запаморочення, безсоння, роздратування.

Трудова діяльність працівників невиробничої сфери належить до категорії робіт, які пов'язані з використанням великих обсягів інформації, із застосуванням комп'ютерних робочих місць, із частим прийняттям

відповідних рішень в умовах дефіциту часу, безпосереднім контактом із людьми різних типів темпераменту тощо. Це зумовлює високий рівень нервово-психічного перевантаження, знижує функціональну активність центральної нервової системи, призводить до розладів в її діяльності, розвитку втоми, перевтоми, стресу.

Тривала робота на комп'ютеризованому робочому місці призводить до значного навантаження на всі елементи зорової системи і зумовлює втому та перевтому зорового аналізатора. Напружена зорова робота викликає «очні» (біль, печія та різь в очах почервоніння повік та очей, ломота у надбрівній частині тощо) та «зорові» (пелена перед очима, подвоєння предметів, мерехтіння, швидка втома під час зорової роботи) порушення органів зору, що може викликати головний біль, посилення нервово-психічного напруження, зниження працездатності.

5.3 Розрахунок системи штучного освітлення приміщення

Освітлення відіграє важливу роль у житті людини. Біля 90% інформації сприймається через зоровий канал, тому правильно виконане раціональне освітлення має важливе значення для виконання всіх видів робіт. Раціональне освітлення повинно відповідати таким умовам: бути достатнім (відповідним нормі і: рівномірним; не утворювати тіней на робочій поверхні; не засліплюєш и працюючого; напрямок світлового потоку повинен відповідати зручному виконанню робіт.

За своєю природою світло - це видиме випромінювання електромагнітних хвиль довжиною від 380 до 780нм. Випромінювання вище 780нм називають інфрачервоним, нижче - ультрафіолетовим. Залежно від джерела світла виробниче освітлення може бути 3-х видів:

а) природне - пряме або відбите світло сонця (небосхила), що освітлює приміщення через світлові прорізи в зовнішніх відгороджувачих конструкціях.

б) штучне - здійснюється штучними джерелами світла і призначене для освітлення приміщень у темні години доби, або таких приміщень, які не мають природного освітлення.

в) сполучене - одночасне поєднання природного і штучного освітлення.

Для розрахунку системи штучного освітлення приміщення застосовуються методи: питомої потужності і коефіцієнт використання світлового потоку.

Для розрахунку використаємо такі формули:

$H_0 = H - h_p$ - відстань від стелі до робочої поверхні

$h_c = 0,2 * H_0$ - відстань від стелі до світильника

$h = H_0 - h_c$ - висота підвіса світильника над освітлювальною поверхнею

$H_{\pi} = h + h_p$ - висота підвіса світильника над підлогою

$L = 1,5 * h$ - відстань між центрами світильників

$N = S / L^2$ - необхідна кількість світильників

$i = (A * B) / (h * (A + B))$ - індекс приміщення

$\Phi_p = (E_{\min} * S * K_3 * Z) / (N * \eta)$ - світловий потік однієї лампи

$E_{\phi} = E_{\min} * (\Phi_{\pi} / \Phi_p)$ - фактична освітленість

$P_{\text{общ}} = P_{\pi} * N$ - загальна потужність

Розрахуємо освітленість в приміщенні кафедри Національного університету кораблебудування. Приміщення має такі параметри, довжина $A = 7$ м, ширина $B = 3$ м, висота $H = 2,5$ м, висота робочої поверхні $h_p = 1$ м, світильник типа УМП-15. Мінімальна освітленість для лампи розжарювання за нормами $E_{\min} = 100$ лк. Коефіцієнт відображення стелі $S_{\pi} = 50\%$, стін $S_c = 30\%$, робочої поверхні $S_p = 10\%$. Напруга в мережі 220В.

$H_0 = H - h_p = 2,5 - 1 = 1,5$ м.

$h_c = 0,2 * H_0 = 0,3$ м.

$h = H_0 - h_c = 1,2$ м.

$H_{\pi} = h + h_p = 2,2$ м.

Для досягнення рівномірності освітлені приймаємо співвідношення:

$$L/h = 1,5$$

$$L = 1,5 * 1,2 = 1,8 \text{ м.}$$

$$N = S / L^2 = (7*3)/1,8^2 = 6,48$$

Приймаємо 6 світильників (два ряди по три світильники)

$$i = (A * B) / (h * (A + B)) = (7*3) / (1,2 * (7+3)) = 1,75$$

З таблиці коефіцієнтів використання світлового потоку при $i = 1,75$, $S_{\text{п}} = 50\%$, $S_{\text{с}} = 30\%$, $S_{\text{р}} = 10\%$ для світильника типа УНМ-15 коефіцієнт використання світлового потоку $\eta = 0,53$.

$$\Phi_{\text{р}} = (E_{\text{min}} * S * K_3 * Z) / (N * \eta) = (100 * 21 * 1,3 * 1,15) / (6,48 * 0,53) = 3139,5 / 3,43 = 915 \text{ лм}$$

Обираємо відповідну лампу - Б 215-225-75.

$$E_{\text{ф}} = E_{\text{min}} * (\Phi_{\text{л}} / \Phi_{\text{р}}) = 100 * (950 / 915) = 103 \text{ лк}$$

$$P_{\text{общ}} = P_{\text{л}} * N = 75 * 6 = 450 \text{ Вт}$$

З наведених вище результатів, отримали, що не треба розробляти заходи щодо зміни освітленості приміщення учбового відділу. Схема розташування освітлювальних приладів представлена на рисунку 5.1.

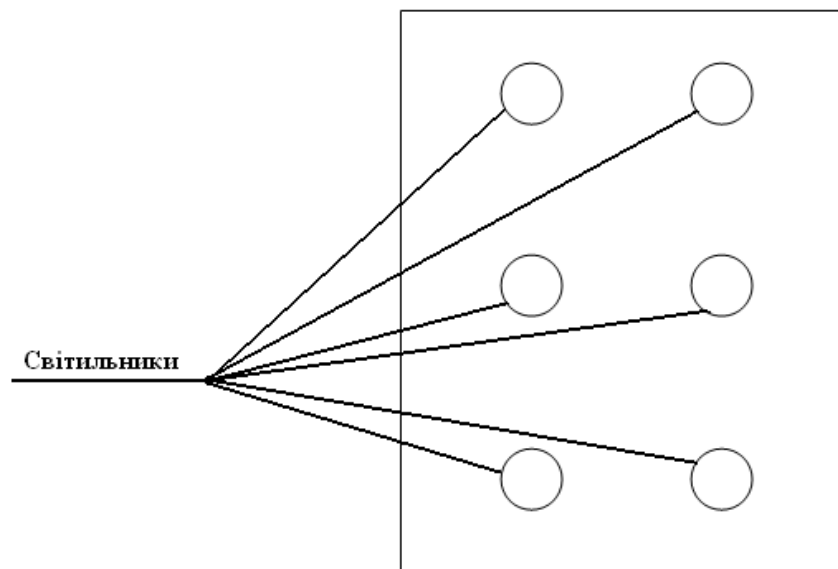


Рисунок 5.1 – Схема розташування освітлювальних приладів

5.4 Основні положення техніки безпеки під час експлуатації ЕОМ

До роботи з персональними ЕОМ і зовнішніми пристроями ЕОМ

допускаються особи, які пройшли медичний огляд, вступний інструктаж, інструктаж і навчання на робочому місці, перевірку знання по охорони праці.

Оператори і користувачі ЕОМ зобов'язані:

1) Дотримуватись правил внутрішнього розпорядку

2) Знати і дотримуватися правил з охорони праці в обсязі виконуваних вимозі.

3) Виконувати тільки ту роботу, яка визначена інструкцією: з експлуатації устаткування та посадовими інструкціями, затвердженими адміністрацією підприємства, і за умови, що безпечні способи її виконання добре відомі

4) Знати й уміти надавати першу медичну допомогу при ураження електричним струмом і при інших нещасних випадках

5) Дотримуватися інструкції про заходи пожежної безпеки.

Якщо не виконувати профілактичні заходи і не дотримуватися режиму роботи, робота з ЕОМ, як правило, супроводжується зоровим і загальним перевтомою.

Організація робочого місця:

1) Приміщення повинно мати природне і штучне освітлення.

2) Площа на одне робоче місце з ЕОМ для дорослих користувачів повинна бути не менше 6,0 кв.м, обсяг - не менш 20,0 куб.м.

3) У виробничих та адміністративно-громадських приміщеннях у випадках переважної роботи з документами допускається застосування системи комбінованого освітлення.

4) Освітленість на поверхні столу в зоні розміщення робочого та документа повинна бути 300-500 лк.

5) Робочі місця по відношенню до світлових отворів повинні розташовуватися так, щоб природне світло падало з боку переважно сторінки. З метою профілактики перевтоми і перенапруження при роботі з дисплеями необхідно виконувати під час регламентованих перерв комплекси спеціальних вправ (вони наведені нижче).

6) Схеми розміщення робочих місць з ЕОМ повинні враховувати відстані між робочими столами з відеомоніторами (в напрямі тилу поверхні одного відеомонітора і екраном іншого), яка повинна бути не менше 2,0 м відстань між бічними поверхнями відеомонітора не менше 1,2 м. 7) Робочий стіл повинен регулюватися по висоті в межах 680-760 мм, за відсутності такої можливості його висота має складати 1600х900 мм. Під стільницею робочого столу має бути вільний простір для ніг з розмірами по висоті не менше 600 мм, по ширині 500 мм, по глибині 650 мм.

7) Розмір екрана повинен бути не менше 31 см (14 дюймів) по діагоналі, при цьому відстань від очей до екрану повинна бути в межах 40-80 см.

8) Робочий стіл (крісло) повинен бути підйомно-поворотним і регульованим по висоті та кутах нахилу сидіння і спинки, а також - відстані спинки від переднього краю сидіння. Конструкція його повинна забезпечувати:

- а) ширину і глибину поверхні сидіння не менше 400 мм;
- б) поверхню сидіння з заокругленим верхнім краєм;
- в) регулювання висоти поверхні сидіння у межах 400-550 мм і кутом нахилу вперед на 15 град., і тому до 5 град.;
- г) висоту опорної поверхні спинки 300 плюс-мінус 20 мм, ширину не менше 380 мм та радіус кривизни горизонтальної площині - 400 мм;
- д) кут нахилу спинки у вертикальній площині в межах 0 ± 30 град.;
- е) врегулювання відстані спинки від переднього краю сидіння в межах 250-400 мм;
- є) стаціонарні або знімні підлокітники довжиною не менше 250 мм: і шириною 50-70 мм; регулювання підлокітників по висоті над сидінням у межах 230 ± 30 мм і внутрішнього відстані: між підлокітниками в межах 350-500 мм.

Робоче місце має бути обладнане підставкою для ніг, що має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах до 150 мм і кутом нахилу опорної поверхні підставки до 20 град. Поверхня підставки має бути рифленою і мати по передньому краю бортик висотою 10

мм.

Робоче місце з ЕОМ повинне бути оснащено легко переміщуваним пюпітром для документів.

Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача, або на спеціальною регульованою по висоті робочої поверхні, відокремленої від основний стільниці. Про кожний нещасний випадок на виробництві потерпілий або очевидець негайно сповіщає безпосереднього керівника.

За невиконання даної інструкції винні притягуються до відповідальності згідно з правилами внутрішнього трудового розпорядку або стягненням, визначених Кодексом законів про працю України.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Вступ

Взаємовідносини суспільства і природи полягають у тому, що фактори економічного зростання — трудові ресурси, засоби виробництва і природні ресурси — у комплексі використовуються суспільством для розвитку виробництва. Питання взаємовідносин суспільства і природи та використання природних ресурсів стають дедалі актуальнішими. І це зрозуміло, бо з розвитком виробництва вилучаються все нові багатства природи, зростає вартість сировини, збільшується кількість відходів, що викидаються у навколишнє середовище. Однак, і це очевидно, було б неправильно вирішувати проблеми збереження ресурсів і середовища шляхом припинення росту або навіть скорочення обсягів виробництва. Такі припущення суперечать закономірностям розвитку людського суспільства і практично нездійсненні.

Природне середовище — невід'ємна умова життя людини і суспільного виробництва, оскільки воно є необхідним середовищем існування людини і джерелом потрібних йому ресурсів.

Під впливом людини відбуваються величезні зміни природного середовища, з чим пов'язана необхідність його охорони. В ХХ ст. людина проклала нові шляхи переміщення енергії і речовини в географічній оболонці, подекуди значною мірою порушивши екологічну рівновагу.

Отже, **охорона довкілля** — система заходів щодо раціонального використання природних ресурсів, збереження особливо цінних та унікальних природних комплексів і забезпечення екологічної безпеки. Це сукупність державних, адміністративних, правових, економічних, політичних і суспільних заходів, спрямованих на раціональне використання, відтворення і збереження природних ресурсів землі, обмеження негативного впливу людської діяльності на навколишнє середовище.

Охорона довкілля включає в себе охорону атмосферного повітря, вод суші та вод Світового океану, земель, флори і фауни, геологічного середовища.

Мета охорони навколишнього середовища– протидія негативним змінам у навколишньому середовищі, які мали місце в минулому, відбуваються зараз або можуть бути.

Актуальність. Проблема охорони навколишнього середовища перетворилася в глобальну проблему і пов'язана зі зростаючим антропогенним впливом. Це зумовлено демографічним вибухом, урбанізацією, що прискорюється, і розвитком гірн. розробок і комунікацій, забрудненням навколишнього середовища відходами, надмірним навантаженням на орні землі, пасовища, ліси, водойми. У результаті гірничо-техн. діяльності у світі порушено не менше 15 – 20 млн га земель, з них 59% площі використано під різні гірничі виробки, 38% – під відвали пустої породи або відходів збагачення, 3% – місця осідання, провалів і ін. порушень поверхні, пов'язаних з підземними розробками. Інколи порушення правил ведення гірничих робіт чи масштабна аварія призводить до катастрофічних незворотних наслідків.

Комплексна охорона навколишнього середовища здійснюється на рівні підприємств, населених пунктів, регіонів, держав і глобально – в масштабах всієї планети. Велика робота ведеться під егідою ООН, з ініціативи якої в 1972 р. створена постійно діюча Програма ООН по довкіллю (ЮНЕП). У рамках ООН природоохоронні проблеми вирішують також: Всесвітня метеорологічна організація (ВМО), Всесвітня організація охорони здоров'я (ВОЗ), Міжнародна морська організація (ММО), Міжнародне агентство з атомної енергії (МАГАТЕ), Міжнародна комісія з навколишнього середовища і розвитку (МКНСР), ЮНЕСКО та ін. Велику увагу проблемам О.н.с. приділяють Організація економічної співпраці і розвитку (ОЕСР), Європейське економічне співтовариство (ЄЕС), Організація американських держав (ОАД), Ліга арабських країн з питань

освіти, культури і наук (АЛЕКСО). Генеральна Асамблея ООН прийняла в 1982 р. Всесвітню хартію природи, яка є розвитком Стокгольмської декларації про довкілля (1972), і Всесвітню стратегію охорони природи, розроблену МСОП (1980). В останні десятиліття XX ст. під егідою ООН розроблена Концепція сталого розвитку, яка передбачає глобальні (в просторі і часі) підходи до охорони навколишнього середовища.

В Україні питання охорони навколишнього середовища офіційно перебувають у компетенції Міністерства екології і природних ресурсів, але напряду стосуються кожного громадянина. Тому існує низка формальних і неформальних організацій, товариств і рухів охорони довкілля, що дозволяють діяти локально і більш оперативно, ніж державним структурам.

6.1 Забруднення навколишнього середовища твердими побутовими відходами

В даній дипломній роботі розглядається діяльність підприємств, які займають розробкою та проектуванням меблів. В процесі діяльності утворюються різні тверді побутові відходи. До них можна віднести папір, картонні коробки, різні пакети та т.інш, в цілому офісні відходи.

Проблема твердих побутових відходів характерна не тільки для цього закладу, з нею стикаються майже всі установи та заклади, які займаються подібною справою. Ця проблема є не тільки проблемою відокремлених підприємств, установ та закладів, але й всього міста та держави в цілому.

Проблема екологічної небезпеки твердих побутових відходів гостро стоїть перед кожним населеним пунктом України. Ця небезпека пов'язана з усіма стадіями поводження з ТПВ, починаючи з їх збирання і транспортування та закінчуючи переробленням, утилізацією або захороненням. Побутові відходи вносять вагомий внесок у забруднення навколишнього природного середовища. Вони утворюються в процесі життя і діяльності людини всюди, де вона знаходиться: у житлових будинках, виробничих підприємствах, адміністраціях, громадських організаціях,

невиробничих установах, лікувальних, торговельних та інших закладах. Це найбільша за обсягами накопичення група відходів споживання, яка відрізняється від усіх інших відходів за своїм походженням та складом.

Тверді побутові відходи створюють санітарно-гігієнічну та епідеміологічну небезпеку внаслідок неприємних запахів та шкідливих хімічних сполук, які можуть бути присутні або утворюються в них при біологічному розкладанні органічних компонентів відходів, а також внаслідок присутності в них збудників інфекційних та паразитарних хвороб та внаслідок високої бактеріальної і гельмінтологічної забрудненості.

Практика первісного накопичення та збору і вивезення змішаних міських твердих побутових відходів, що склалася в більшості населених пунктів України, обумовлює антисанітарні умови на всіх етапах поводження з ТПВ внаслідок біологічного розкладання (гниття) органічних компонентів, суттєво ускладнює сортування та відбір сировини, перетворює полігон та звалища для розміщення ТПВ в екологічно небезпечні об'єкти, на яких виділяється біогаз, що забруднює атмосферне повітря та фільтрат, що забруднює ґрунтові води.

Проблема знешкодження або часткової утилізації твердих побутових відходів є досить актуальною з точки зору негативного впливу на навколишнє природне середовище. ТПВ вносять вагомий внесок у забруднення навколишнього природного середовища, оскільки містять в своєму складі велику кількість вологих органічних речовин, які піддаються розкладу і дають інтенсивне забруднення як повітря, ґрунту, так і поверхневих і ґрунтових вод. Щорічно в Україні утворюються майже 50 млн. м³ побутових відходів, причому до 40% з яких містять ресурсоцінні компоненти, які можуть бути повернені у господарський обіг (папір, скло, пластик, тощо). Тому одним із головних напрямків вирішення проблеми зменшення екологічного навантаження на довкілля є використання відходів в якості вторинної сировини.

Тверді побутові відходи є цінним джерелом вторинних ресурсів (в тому числі чорних, кольорових, рідкісних металів), а також «безкоштовним» енергоносієм, тому що побутове сміття може використовуватись як енергетична сировина для паливної енергетики. Важливо, щоб процеси утилізації не порушували екологічну безпеку держави. Кількість твердих відходів постійно зростає із-за зміни способу життя.

6.2 Розробка заходів щодо зменшення забруднення

Комплексне управління відходами починається із змін поглядів на те, чим являються побутові відходи.

Традиційні підходи до проблеми твердих побутових відходів орієнтувались на зменшення небезпечного впливу їх на навколишнє середовище шляхом ізоляції сміттєзвалища від ґрунтових вод, очистка викидів сміттєспалювальних заводів і т. п.

Нетрадиційний підхід до проблеми твердих побутових відходів орієнтується на вхідний контроль побутових відходів.

Основна концепція комплексного управління відходами передбачає, що побутові відходи складаються з різних компонентів, які в ідеальній ситуації не повинні змішуватися між собою, а повинні утилізуватися окремо один від одного найбільш вигідними екологічно-економічними методами.

Комбінація технологій і заходів, включаючи скорочення кількості відходів, вторинну переробку і компостування, захоронення на полігонах та сміттєспалювання повинні використовуватися для утилізації тільки того чи іншого специфічного компонента ТПВ. Всі технології та заходи повинні використовуватися в комплексі, взаємодоповнюючи одне одного.

Муніципальна система утилізації твердих побутових відходів повинна розроблятися з урахуванням конкретних місцевих проблем та базуватися на місцевих ресурсах. Досвід в утилізації ТПВ повинен здобуватися шляхом розроблення та виконання невеликих програм.

Комплексний підхід до переробки відходів повинен базуватись на стратегічному довгостроковому плануванні для забезпечення гнучкості та адаптації до майбутніх змін у складі та кількості твердих побутових відходів і доступності технологій утилізації.

Участь місцевої влади, а також всіх груп населення (сміттєвиробників) в реалізації програми комплексного управління твердими побутовими відходами.

Концепція комплексного управління відходами передбачає, що на додаток до традиційних методів утилізації твердих побутових відходів (сміттєспалювання та захоронення) повинні стати їх невід'ємною частиною заходи по скороченню кількості відходів, вторинна переробка відходів і компостування. Тільки комбінація декількох способів може сприяти ефективному рішенню проблеми твердих побутових відходів.

Захоронення твердих побутових відходів використовуються як міра їх утилізації в наступних випадках:

- високої токсичності матеріалів;
- негорючості компонентів відходів та неможливості їх переробки в будівельні матеріали;
- залишки горючих компонентів відходів;
- всі побутові відходи у випадку відсутності установки для їх спалювання.

Захоронення здійснюється на санітарних полігонах, які відповідають екологічним вимогам та санітарно-епідеміологічним нормам і являють собою сучасну складну інженерну споруду, обладнану системами боротьби із забрудненнями ґрунту, води, повітря.

До розміщування та функціонування сучасних санітарних полігонів включають наступні вимоги:

- геологічні (не допускається розміщення полігону на тектонічних розломах та сейсмічно небезпечній зоні);

- гідрологічні (віддаленість від відкритих водоймищ, низький рівень ґрунтових вод, не затопленість їх території паводковими водами);
- санітарні (будівництво та експлуатація полігону згідно санітарних правил та вимог);
- екологічні (не допустити розміщення полігону в природоохоронній зоні, екологічна експертиза проекту, експлуатація полігону, функціонування його згідно екологічних норм і правил, зменшення його впливу на навколишнє природне середовище та здоров'я людей, державна реєстрація його як джерела екологічного лиха, план виводу його з експлуатації та рекультивації земель зайнятих під ним, контроль і моніторинг навколишнього природного середовища на полігоні і його санітарній зоні).
- фінансові (надання фінансових гарантій безпечного функціонування полігону, страхування на випадок екологічного лиха).

Санітарні полігони ще тривалий час будуть залишатися основним способом переробки твердих побутових відходів. Альтернативою йому може стати первинна покомпонентна утилізація відходів.

Висновки

У результаті виконання магістерської роботи були досліджені основні тенденції розвитку САПР на економічному ринку, були виявлені основні проблеми, що виникають при проектуванні і розробці САПР, серед яких можна відмітити наступні:

1. Труднощі, що виникають при колективній роботі над проектом. Оскільки усі учасники такого процесу не можуть застосовувати одну і ту ж систему проектування, то їм важко буває використати проекти колег, побудовані за допомогою іншого програмного інструменту, і вони витрачають масу часу, виправляючи їх або створюючи наново. В результаті підприємства стикаються з труднощами при інформаційному обміні, принаймні ті, хто застосовує різні САПР.

2. Висока вартість володіння САПР. Вартість володіння складається з витрат на купівлю ліцензій і апаратного забезпечення. В якості останніх в Україні найчастіше використовуються звичайні універсальні ПК, в яких абсолютно не враховується специфіка САПР. А тим часом ефективність усього комплексу значною мірою залежить від апаратних засобів. Проте, як і у ПЗ, вартість володіння професійною технікою складається не лише з витрат на купівлю, але і з витрат на експлуатацію. Експлуатаційні витрати зазвичай включають витрати із-за втрати інформації, ремонту і простою устаткування.

Проте існують і приховані витрати:

1) кожен перехід на нову версію вимагає витрат. Складні моделі, створені за допомогою попередньої версії САПР, не завжди правильно інтерпретуються в наступній. Для перевірки сумісності клієнти витрачають час і гроші на попереднє тестування кожної чергової версії САПР;

2) постачання ліцензій з обов'язковою оплатою щорічного технічного супроводу;

3) припинення підтримки старих версій;

4) практики стягування додаткових сум(штрафів) за пропущений платний супровід і навіть за несвоєчасну оплату річного супроводу;

5) платна підписка на оновлення ПЗ.

Застосування технологій хмарних обчислень допоможе уникнути більшості цих проблем і витрат як з боку клієнтів САПР, так і з боку розробників.

В результаті виконання роботи були

1 Проаналізовані перспективи розвитку САПР і показано, що роботи по проектуванню мають розподілений характер і тенденцію до організації віртуальних підприємств.

2 Досліджені технології хмарних обчислень, основні принципи побудови застосувань на основі технологій хмарних обчислень, внаслідок чого визначена концепція архітектури програмного забезпечення САПР в «хмарі».

3 Проаналізовані основні характеристики існуючих хмарних платформ, що дозволило вибрати найбільш відповідні для розгортання САПР.

4 Був проведений вибір PaaS-постачальників хмарних обчислень, що дозволяє вибрати тих, що найбільш задовольняють вимогам до розробки ПО САПР і надання його як послуги.

5 Запропонована архітектура побудови ПО САПР, що передбачає розподіл функцій між ресурсами «хмари» і робочої станції, враховуючі недоліки існуючих.

6 Показана модель побудови САПР для проектування меблів, в яких розробники є постачальниками SaaS- послуг з одного боку, і споживачами PaaS з іншою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gartner. Predicts 2024: The Changing Role of the Identity and Access Management Leader [Електронний ресурс]. Gartner Research, 2023. Режим доступу: <https://www.gartner.com/en/newsroom/press-releases/>.
2. Muñoz-Ausecha C., Ruiz-Rosero J., Ramírez-González G. RFID Applications and Security Review. Computation. 2021. Vol. 9, No. 6. P. 69. DOI: 10.3390/computation9060069.
3. Golightly L. та ін. Securing Distributed Systems: A Survey on Access Control Techniques and Trends. Journal of Information Security and Applications. 2023. Режим доступу: <https://www.sciencedirect.com/science/article/>.
4. Zhang Y. та ін. Attribute-Based Encryption for Cloud Computing Access Control: A Survey. 2020. Режим доступу: <https://smusg.elsevierpure.com/en/publications>.
5. Espressif Systems. ESP32 Series Datasheet [Електронний ресурс]. 2023. Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.
6. NXP Semiconductors. MFRC522 Standard Performance MIFARE and NTAG Frontend [Електронний ресурс]. Режим доступу: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>.
7. Alizadeh M. та ін. A Survey on Attacks in RFID Networks. 2021. Режим доступу: <https://oiji.utm.my/index.php/oiji/article/download/52/29>.
8. Landauer M. та ін. Deep Learning for Anomaly Detection in Log Data: A Survey. 2022. Режим доступу: <https://arxiv.org/pdf/2207.03820.pdf>.
9. Chen S., Li L. Unsupervised Anomaly Detection for IoT Cybersecurity. IEEE Internet of Things Journal. 2023.

10. Goldstein M., Uchida S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PLOS ONE. 2016. Vol. 11, No. 4. e0152173. DOI: 10.1371/journal.pone.0152173.
11. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest. Proc. 2008 IEEE International Conference on Data Mining. 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.
12. Müller H. та ін. Anomaly Detection in Physical Access Control Systems Using Machine Learning. Proc. 2022 Int. Conf. on Cyber Security and Protection of Digital Services. 2022.
13. Splunk. Splunk Machine Learning Toolkit [Електронний ресурс]. Режим доступу: https://www.splunk.com/en_us/products/machine-learning-toolkit.html.
14. Kim J. та ін. LSTM-based Anomaly Detection for Smart Home Security. IEEE Transactions on Dependable and Secure Computing. 2022. Vol. 19, No. 5. P. 3456–3468. DOI: 10.1109/TDSC.2021.3094275.
15. Tax D. M. J., Duin R. P. W. Support Vector Data Description. Machine Learning. 2004. Vol. 54, No. 1. P. 45–66. DOI: 10.1023/B:MACH.0000008084.60811.49.
16. Kingma D. P., Welling M. Auto-Encoding Variational Bayes. arXiv:1312.6114. 2013. Режим доступу: <https://arxiv.org/abs/1312.6114> (дата звернення: 15.11.2025).
17. Li S., Wang X. A Hybrid Machine Learning Approach for Anomaly Detection in Industrial IoT. IEEE Transactions on Industrial Informatics. 2023. Vol. 19, No. 9. P. 10045–10055.
18. Arduino. Arduino Reference [Електронний ресурс]. 2024. Режим доступу: <https://www.arduino.cc/reference/en/>.
19. Balboa M. MFRC522 Library for Arduino [Електронний ресурс]. 2023. Режим доступу: <https://github.com/miguelbalboa/rfid>
20. Pandas Development Team. pandas: powerful Python data analysis toolkit [Електронний ресурс]. Режим доступу: <https://pandas.pydata.org>.

- 21.Scikit-learn Developers. Scikit-learn: Machine Learning in Python [Электронный ресурс]. Режим доступа: <https://scikit-learn.org/stable/> (дата звернения: 15.11.2025).
- 22.TensorFlow Development Team. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Электронный ресурс]. Режим доступа: <https://www.tensorflow.org/>
- 23.OMG. UML Specification — Unified Modeling Language [Электронный ресурс]. 2025. Режим доступа: <https://www.omg.org/spec/UML/>.
- 24.Bootstrap Team. Bootstrap Documentation 5.3 [Электронный ресурс]. 2025. Режим доступа: <https://getbootstrap.com/docs/5.3/>.
- 25.Flask Development Team. Flask Documentation (3.0.x) [Электронный ресурс]. 2025. Режим доступа: <https://flask.palletsprojects.com/en/3.0.x/>.
- 26.SQLite Consortium. SQLite Documentation [Электронный ресурс]. 2025. Режим доступа: <https://www.sqlite.org/docs.html>

ДОДАТОК – Текст програми

Лістинг 1 – клас OrderGUI.java

```
package GUI;

import OrderAgent.OrderAgent;

public class OrderGUI extends javax.swing.JFrame {

    OrderAgent orderAgent;

    public OrderGUI(OrderAgent a) {
        orderAgent = a;
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        jTextField4 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)), " Наряд на розкрій "));

        jLabel2.setText("Номер замовлення:");

        jLabel1.setText("Загальна довжина пізів:");

        jLabel3.setText("Кількість пізів:");

        jLabel4.setText("Номер карти розкрою:");

        jButton1.setText("Додати");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("<");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
    }
}
```



```

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel1)
    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel3)
    .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel4)
    .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jButton1)
    .addComponent(jButton2))
    .addContainerGap())
);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
        .addContainerGap())
    );
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
        .addContainerGap())
    );

pack();
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String name = jTextField1.getText();
    orderAgent.getAID().setName(name);
}

```

```

        double length = Double.parseDouble(jTextField2.getText());
        orderAgent.setLength(length);
        int number = Integer.parseInt(jTextField3.getText());
        orderAgent.setNumber(number);
        int mapNumber = Integer.parseInt(jTextField4.getText());
        orderAgent.setMapNumber(mapNumber);
        System.out.println("Agent by order #" + orderAgent.getAID().getName() +
" is ready");
        this.setVisible(false);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        this.setVisible(false);
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    // End of variables declaration
}

```

Лістинг 2 – клас OrderAgent.java

```

package OrderAgent;

import GUI.OrderGUI;
import jade.core.Agent;

public class OrderAgent extends Agent {

    double cutsLength;
    int cutsNumber;
    int cutsMapNumber;
    String instruction;
    String note;

    @Override
    public void setup() {
        OrderGUI orderGUI = new OrderGUI(this);
        orderGUI.setVisible(true);
    }

    public void setLength(double length) {
        cutsLength = length;
    }

    public void setNumber(int number) {
        cutsNumber = number;
    }

    public void setMapNumber(int mapNumber) {
        cutsMapNumber = mapNumber;
    }

    public double getLength() {

```

```

        return cutsLength;
    }

    public int getNumber() {
        return cutsNumber;
    }

    public int getMapNumber() {
        return cutsMapNumber;
    }

    @Override
    protected void takeDown() {
        System.out.println("OrderAgent #" + getAID().getName() + "
terminating");
    }
}

```

ЛІСТИНГ 3 – клас WorkSchedule.java

```

package ResourceAgent;

import OrderAgent.OrderAgent;
import java.util.ArrayList;

public class WorkSchedule {

    ArrayList<OrderAgent> schedule = new ArrayList<OrderAgent>();

    public void addOrder (OrderAgent order) {
        schedule.add(order);
    }

    public void addOrder (int i, OrderAgent order) {
        schedule.add(i, order);
    }

    public void setOrder (int i, OrderAgent order) {
        schedule.set(i, order);
    }

    public void removeOrder (int i) {
        schedule.remove(i);
    }

    OrderAgent getOrder (int i) {
        return schedule.get(i);
    }
}

```

ЛІСТИНГ 4 – клас MenagementGUI.java

```

package GUI;

public class MenagementGUI extends javax.swing.JFrame {

    public MenagementGUI() {
        initComponents();
    }
}

```

```
}
```

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {
```

```
    jPanel1 = new javax.swing.JPanel();
```

```
    jScrollPane1 = new javax.swing.JScrollPane();
```

```
    jTable1 = new javax.swing.JTable();
```

```
    jLabel1 = new javax.swing.JLabel();
```

```
    jTextField1 = new javax.swing.JTextField();
```

```
    jButton1 = new javax.swing.JButton();
```

```
    jButton2 = new javax.swing.JButton();
```

```
    jButton3 = new javax.swing.JButton();
```

```
    jButton4 = new javax.swing.JButton();
```

```
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
    jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLineBorder(new
    java.awt.Color(0, 0, 0)), " Графік виконання "));
```

```
    jTable1.setModel(new javax.swing.table.DefaultTableModel(
```

```
        new Object [][] {
```

```
            {null, null, null, null, null},
```

```
            {null, null, null, null, null},
```

```
            {null, null, null, null, null},
```

```
            {null, null, null, null, null}
```

```
        },
```

```
        new String [] {
```

```
            "№", "Номер замовлення", "Номер карти розкрою", "Інструкція", "Примітка"
```

```
        }
```

```
    ));
```

```
    jScrollPane1.setViewportView(jTable1);
```

```
    jLabel1.setText("Зараз виконується:");
```

```
    jButton1.setText("Відмітити завершення");
```

```
    jButton2.setText("<");
```

```
    jButton3.setText("Змінити пріоритет");
```



```

jButton4.setText("Перенести замовлення на іншу ділянку");

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 415, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                        .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, 309, Short.MAX_VALUE))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jButton2)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jButton3)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jButton4)))
            .addGap(10, 10, 10)
        )
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(10, 10, 10)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton1))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton2)

```

```

        .addComponent(jButton3)
        .addComponent(jButton4))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
            );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            );

    pack();
} // </editor-fold>

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new MenagementGUI().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;

```

```

private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField1;
// End of variables declaration
}

```

Лістинг 5 – клас ResourceAgent.java

```

package ResourceAgent;

import OrderAgent.OrderAgent;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.domain.introspection.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class ResourceAgent extends Agent {

    WorkSchedule schedule;

    @Override
    protected void setup() {
        getAID().setName("ResourceAgent");
        System.out.println(getAID().getName() + " is ready");
        schedule = new WorkSchedule();
        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        ServiceDescription sd = new ServiceDescription();
        sd.setType("CutOut");
        sd.setName("JADE-order-cutting");
        dfd.addServices(sd);
        try {
            DFService.register(this, dfd);
        } catch (FIPAException fe) {
            fe.printStackTrace();
        }
        addBehaviour(new OfferRequestsServer());
        addBehaviour(new PurchaseOrdersServer());
    }

    @Override
    protected void takeDown() {
        try {
            DFService.deregister(this);
        } catch (FIPAException fe) {
            fe.printStackTrace();
        }
        System.out.println(getAID().getName() + " terminating");
    }

    public void updateSchedule(final OrderAgent order) {
        addBehaviour(new OneShotBehaviour() {

            public void action() {
                schedule.addOrder(order);
            }
        });
    }
}

```

```

        System.out.println("Order " + order.getName() + " inserted into
Work Schedule");
    }
    });
}

private class OfferRequestsServer extends CyclicBehaviour {

    public void action() {
        MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.CFP);
        jade.lang.acl.ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
            String title = msg.getContent();
            jade.lang.acl.ACLMessage reply = msg.createReply();
            Integer price = (Integer) schedule.getOrder(0).getNumber();
            if (price != null) {
                reply.setPerformative(ACLMessage.PROPOSE);
                reply.setContent(String.valueOf(price.intValue()));
            } else {
                reply.setPerformative(ACLMessage.REFUSE);
                reply.setContent("not-available");
            }
            myAgent.send(reply);
        } else {
            block();
        }
    }
}

private class PurchaseOrdersServer extends CyclicBehaviour {

    public void action() {
        MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
        jade.lang.acl.ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
            String title = msg.getContent();
            jade.lang.acl.ACLMessage reply = msg.createReply();
            Integer price = (Integer) schedule.remove(order);
            if (price != null) {
                reply.setPerformative(ACLMessage.INFORM);
                System.out.println(title + " sold to agent " +
msg.getSender().getName());
            } else {
                reply.setPerformative(ACLMessage.FAILURE);
                reply.setContent("not-available");
            }
            myAgent.send(reply);
        } else {
            block();
        }
    }
}
}

```