**Практична робота №3-1.**  **Архітектура програмного забезпечення. Побудова функціональної, структурної та укрупненої схеми архітектури ПС**

***Мета:*** отримати практичні навички в обґрунтованому виборі компонентів архітектури програмного забезпечення для реалізації завдання та його нефункціональних властивостей.

**КОНТРОЛЬНІ ПИТАННЯ:**

1. Що таке архітектура ПЗ?

2. Що таке описова та розпорядча архітектура ПЗ?

3. Що таке архітектурний дрейф та ерозія?

4. Що таке відновлення архітектури?

5. Що таке компонент ПЗ?

6. Що таке з’єднувач ПЗ?

7. Що таке архітектурна конфігурація або топологія?

8. Що таке нефункціональна властивість ПЗ?

9. Що таке ефективність ПЗ?

10. Що таке складність ПЗ?

11. Що таке надійність ПЗ?

12. Що таке адаптованість ПЗ?

***Завдання:***

1. Опрацювати теоретичний матеріал до лабораторної роботи. Зосередити увагу на основних поняттях у галузі архітектури ПЗ.

2. На основі визначених функціональних та нефункціональних вимог, які необхідно реалізовувати в програмній системі побудувати функціональну, структурну і укрупнену схему архітектури ПС. Тема визначена на попередніх лабораторних (обрана власна тема).

3. Обрати компоненти та з’єднувачі, які будуть використовуватися та обґрунтувати їх вибір.

4. Зробити:

* Опис застосування, що розробляється з точки зору користувача.
* Опис основних функціональних та нефункціональних вимог, використовуючи результати попередніх лабораторних робіт.
* Функціональна, структурна і укрупнена схема архітектури ПС.
* Опис компонентів, які будуть використовуватися: сервіси, які забезпечують, основні архітектурні рішення. Обґрунтувати їх вибір.

По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com). Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**KPZ <Номер групи><Номер лекції / лабораторної> [літера позначення типу роботи L – лекція, R – лабораторна]<Прізвище англійською>**.. Наприклад, **KPZ4101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**KPZ<Номер групи>-ЛР<Номер лабораторної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-41,42,43 – 24.10.2025**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, Тему в заголовку листа записати

**KPZ <Номер групи>-Запитання-<Прізвище >**.

**Методичні рекомендації**

***Архітектура програмного забезпечення (ПЗ)*** – це множина основних проектних рішень про ПЗ. Архітектура ПЗ є планом розробки майбутнього програмного рішення, а також основою для подальшого життєвого циклу ПЗ. Проектні рішення охоплюють всі аспекти розроблюваного ПЗ, такі як структуру, поведінку, взаємодію з іншим ПЗ та нефункціональні властивості. «Основний» означає ступінь важливості, який надає проектному рішенню статус архітектурного. Тобто не всі проектні рішення є архітектурними, саме тому вони не впливають на архітектуру. Проте важливість проектних рішень у великій мірі залежить від цілей, які переслідуються при розробці ПЗ. Аспект тимчасовості архітектури означає, що у будь який момент часу ПЗ має тільки одну архітектуру і вона буде змінюватися з часом.

***Розпорядча системна архітектура складається***з проектних рішень, що були прийняті перед конструюванням ПЗ. Також її можна назвати продуманою чи призначеною архітектурою.

***Описова архітектура ПЗ визначає***те, як система була побудована. Це «реалізована» чи «введена у використання» архітектура. Під час змін у ПЗ, в ідеалі її розпорядчу архітектуру змінюють першою. На практиці під час змін ПЗ, змінюють її описову архітектуру. Зазвичай це трапляється через неуважність або інертність розробників.

***Архітектурний дрейф***– це введення основних проектних рішень до описової архітектури ПЗ, які не включені в розпорядчу архітектуру, але які не конфліктують з її рішеннями.

***Архітектурна ерозія*** – це введення архітектурних рішень до описової архітектури ПЗ, при цьому рішення не повинні порушувати його розпорядчої архітектури.

***Відновлення архітектури***– це процес відтворення архітектури на основі компонентів ПЗ, що були отримані на етапі її конструювання. ПЗ не може виконувати свого призначення до того, як воно буде розгорнуте та налаштоване*.* Модулі ПЗ фізично розміщують на пристроях, на яких вони будуть виконуватися. Архітектурне подання розгортання часто є критичним для оцінювання чи буде система відповідати вимогам. Можливі критерії оцінки: використання пам’яті, споживання енергії, вимоги до пропускної спроможності мережі та інші.

Архітектура ПЗ має бути композицією елементів, котрі репрезентують обробку інформації, інформацію, та комунікацію.

***Компонент***– це архітектурна одиниця, яка відповідає за функціональність ПЗ та/або зберігання даних, надає доступ до своїх сервісів за допомогою зовнішнього інтерфейсу і має чітко визначену залежність від контексту у якому цей компонент застосовують. У складних системах взаємодія між компонентами є більш важливою, ніж функціональність окремих частин ПЗ.

***З’єднувач***– це архітектурна одиниця, що відповідає за здійснення взаємодії між компонентами. У багатьох системах з’єднувачі репрезентовані звичайним викликом процедур, або спільним доступом до даних, але в цілому з’єднувачі можуть бути більш складними рішеннями. Компоненти та з’єднувачі, скомпоновані у спеціальному порядку, складають готову архітектуру ПЗ.

***Архітектурна конфігурація*** *або топологія* – це специфікація з’єднань між з’єднувачами та компонентами.

***Нефункціональні властивості системи***(НВС) – це обмеження на те, як ПЗ реалізує і доставляє свою функціональність. Наприклад: ефективність, складність, розширюваність, надійність. Забезпечення необхідної функціональності часто є досить складним завданням через потреби ринку, конкуренцію, жорсткі терміни, обмежені бюджети, тощо. Однак успіх системи цілковито залежить від НВС.

***Роль архітектури***– це забезпечення НВС на рівні архітектурних блоків: компонентів, з’єднувачів, конфігурації.

***Ефективність***– це якість, яка відображає здатність ПЗ до задоволення вимог продуктивності при одночасній мінімізації використання його ресурсів.

***Складність*** вказує до якої міри ПЗ або однин з його компонентів, містить проектні рішення чи реалізацію, які важко зрозуміти і перевірити.

***Масштабованість ПЗ*** – це можливість системи бути зміненою з урахуванням нових вимог.

***Неоднорідність***– це якість ПЗ, що передбачає, що ПЗ складається з декількох різнорідних компонентів або функціонує в декількох різнорідних обчислювальних середовищах одночасно.

***Пристосовуваність***– є здатністю ПЗ до задоволення нових вимог і пристосовуватися до нових умов роботи під час його життєвого циклу.

***Надійність***– це набір властивостей ПЗ, що дозволяє розраховувати, що ПЗ буде функціонувати так, як було заплановано.

Розглядаючи архітектуру великих організацій, прийнято використовувати поняття «корпоративна архітектура». Її можна представити у вигляді сукупності декількох типів архітектур:

* бізнес архітектура (Busіness archіtecture);
* ІТ-архітектура (Іnformatіon Technology archіtecture);
* архітектура даних (Data archіtecture);
* програмна архітектура (Software archіtecture);
* технічна архітектура (Hardware archіtecture).

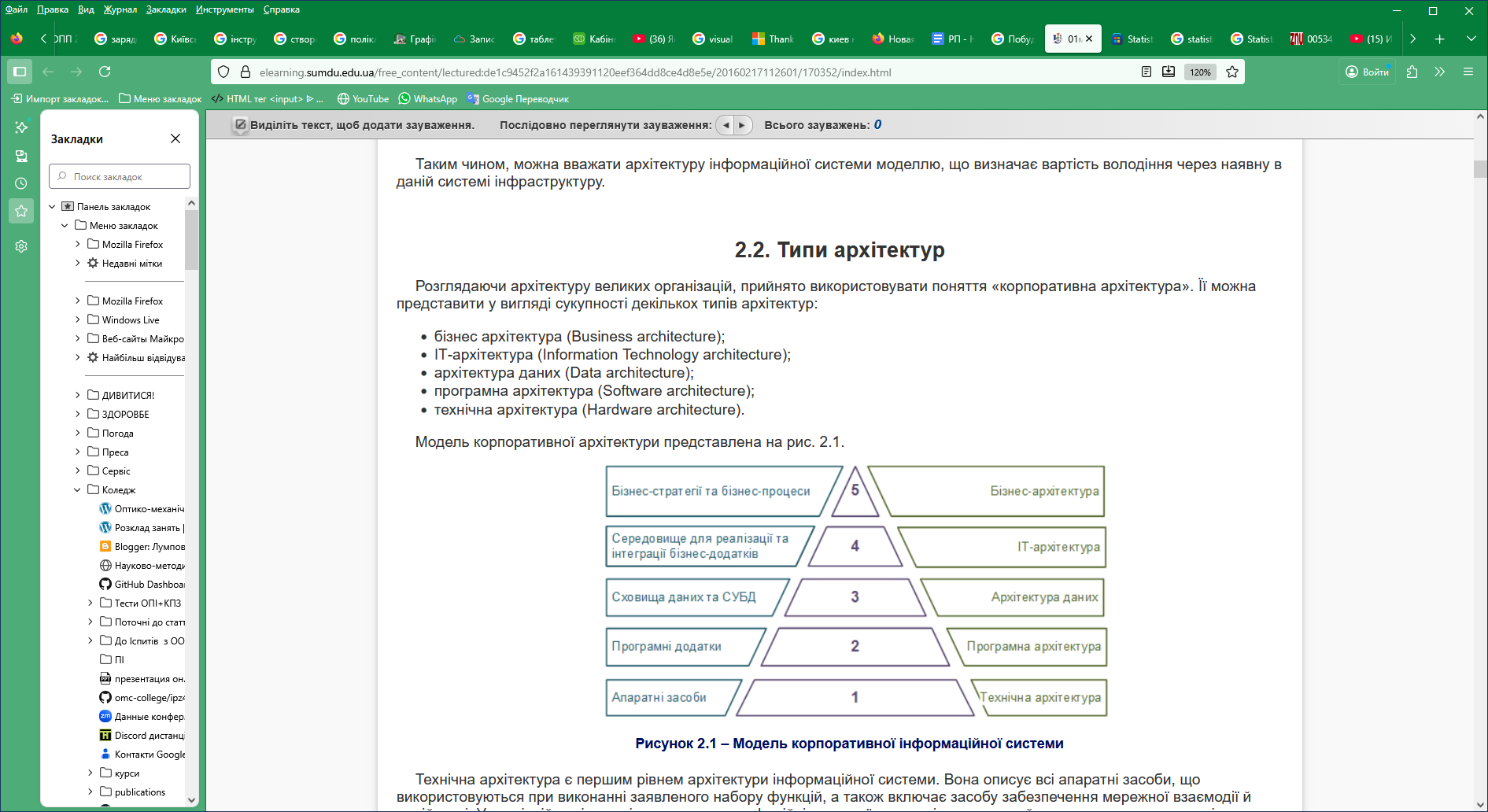
Модель корпоративної архітектури представлена на рис. 1.

Рисунок 1 – Модель корпоративної інформаційної системи

Технічна архітектура є першим рівнем архітектури інформаційної системи. Вона описує всі апаратні засоби, що використовуються при виконанні заявленого набору функцій, а також включає засобу забезпечення мережної взаємодії й надійності. У технічній архітектурі вказуються периферійні пристрої, мережні комутатори й маршрутизатори, жорсткі диски, оперативна пам’ять, процесори, сполучні кабелі, джерела безперебійного живлення й т.п.

Програмна архітектура являє собою сукупність комп’ютерних програм, призначених для рішення конкретних завдань. Даний тип архітектури призначений для опису додатків, що входять до складу інформаційної системи. На даному рівні описують програмні інтерфейси, компоненти й поведінку.

Архітектура даних поєднує в собі як фізичні сховища даних, так і засоби керування даними. Крім того, до неї входять логічні сховища даних, а при орієнтованості розглянутої компанії на роботу зі знаннями, може бути виділений окремий рівень – архітектура знань (Knowledge archіtecture). На цьому рівні описуються логічні й фізичні моделі даних, визначаються правила цілісності, складаються обмеження для даних.

Побудова архітектури програмної системи включає створення функціональної схеми (що система робить), структурної схеми (як система організована) та укрупненої схеми (високорівневий огляд компонентів). Функціональна схема описує функції та процеси, структурна показує зв'язки між основними блоками, а укрупнена схема дає загальне уявлення про систему.

Типи схем архітектури програмної системи

* **Функціональна схема:**
  + Візуалізує функції, які виконує система.
  + Може описувати потоки даних, алгоритми обробки та взаємодію між функціональними блоками.
  + Наприклад, схема може показувати, як система обробляє запит від користувача, відправляє його до бази даних, отримує відповідь і виводить результат.
* **Структурна схема:**
  + Показує загальну структуру системи, її основні компоненти (модулі, класи, сервіси) та зв'язки між ними.
  + Слугує для розуміння організації коду та розподілу відповідальності між різними частинами системи.
  + Наприклад, вона може показувати, що система складається з модулів автентифікації, управління користувачами, обробки замовлень та бази даних, а також які зв'язки існують між цими модулями.
* **Укрупнена схема (високорівнева схема):**
  + Дає загальний, високорівневий огляд системи, показуючи ключові компоненти та їхні основні взаємодії.
  + Використовується для швидкого розуміння загальної картини системи на початкових етапах проектування або для представлення її нетехнічним спеціалістам.
  + Приклад: схема, що показує клієнтський додаток, серверний додаток та зовнішні сервіси, а також зв'язки між ними (наприклад, через API).

Процес побудови

1. **Визначте вимоги:**

Зрозумійте, що система повинна робити (функціональна частина) та які вимоги до неї пред'являються (наприклад, продуктивність, безпека).

1. **Визначте компоненти:**

Розбийте систему на логічні частини або модулі (структурна частина).

1. **Побудуйте укрупнену схему:**

Створіть високорівневу схему, яка показує основні блоки та їхні взаємозв'язки.

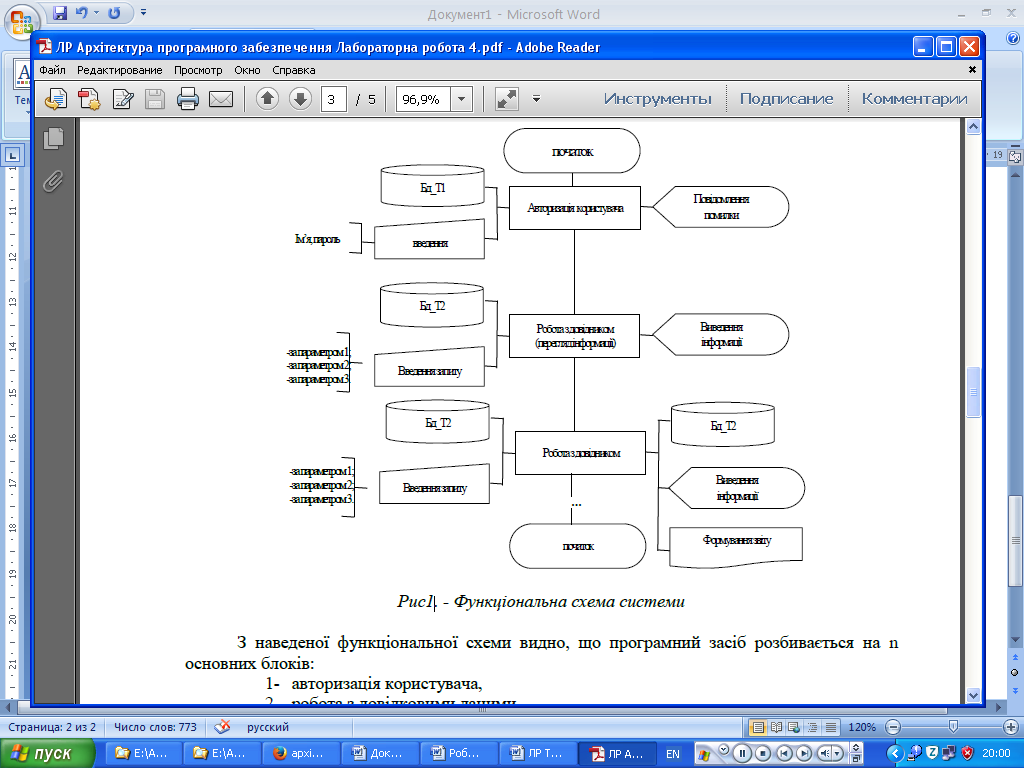
1. **Розробіть функціональну схему:**

Опишіть, як система виконує свої функції, використовуючи, наприклад, діаграми процесів.

1. **Створіть детальну структурну схему:**

Деталізуйте зв'язки між компонентами, показаними на укрупненій схемі, додавши більше деталей про модулі та їхню взаємодію.

***Функціональна схема*** будується з метою розуміння всіх функцій, що виконує програма.

****

З наведеної функціональної схеми видно, що програмний засіб розбивається на n основних блоків:

1- авторизація користувача,

2- робота з довідковими даними,

3- робота з даними по запитах,

4- …

Дані передаються або з БД, або вводяться з клавіатури. Після кожного блоку передбачено перегляд результатів роботи даного блоку. Дані, які змінюються в процесі роботи зберігаються в БД.

При виборі архітектури програмного засобу були поставлені такі завдання і вимоги:

- створення структури даних, що чітко відображають специфіку предметної області;

- моделювання реально існуючих процесів;

- забезпечення оптимальності структур даних;

- поділ і угруповання функцій програмного засобу на підзадачі;

- забезпечення максимальної надійності програмного засобу;

- забезпечення функціональної повноти відповідно до постановки завдання;

- мінімізація інформаційних потоків усередині системи, що дозволяє скоротити час обробки інформації;

- забезпечення наочності модельованих процесів шляхом візуалізації.

Так само в розробляється програмний засіб в якості основних вимог була закладена простота і зручність використання.

В результаті була вибрана модульна структура з функціональною зв'язностю і низьким зчепленням. Тобто структури даних і функції винесені в модулі за функціональною ознакою, що забезпечує реалізацію конкретних підзадач в рамках окремого модуля. Даний підхід дозволяє спростити контроль над збереженням цілісності логіки, а так само спрощує супровід і модернізацію програмного комплексу. Низьке зчеплення модулів[[1]](#footnote-1) дозволяє проводити модернізацію і налагодження кожного модуля окремо, а так само проводити розширення функціональності програмного комплексу шляхом створення додаткових модулів і приєднання його в загальну структуру шляхом підключення його до головного модулю. Все вище перераховане забезпечує значну гнучкість у використанні програмного засобу.

****

Опис модулів програмної системи:

Модуль A в цьому модулі відбувається авторизація користувача, якщо авторизація пройшла успішно модуль запускає модуль MENU;

Модуль MENU основний модуль програмної системи з нього здійснюється зв'язок з іншими модулями ПС;

Модуль AB\_1 викликається з модуля MENU для додавання даних. Після своєї роботи модуль передає управління модулю MENU;

Модуль AB\_2 викликається з модуля MENU для зміни даних. Після своєї роботи модуль передає управління модулю MENU;

Модуль AB\_3 викликається з модуля MENU для видалення даних. Після своєї роботи модуль передає управління модулю MENU;

Модуль AC\_1 …

Модуль AC\_2 …

Модуль AC\_3 …

Модуль AC\_4 …

….

Укрупнену схему надано на рис.3.

**Сервіс: базові поняття**

Сервіс визначається, з одного боку, як відкритий компонент, який може бути елементом швидкої композиції в прикладні програми. З іншого боку, сервіс пропонується як готовий ресурс, який реалізує деякі додаткові можливості, необхідні всім різнорідним програмами для технічної підтримки, потрібної потенційним користувачам. Як правило, описи сервісів містять в собі інформацію про їхні можливості, інтерфейси, поведінку і характеристики. Завдяки такому опису користувач може знайти сервіси, вибрати потрібні і інтегрувати їх в композиційну структуру, як готовий ресурс. Зазвичай розглядається три види сервісів:

1) загальні системні сервіси, які є в кожному загальносистемному середовищі для підтримки процесів проектування і реалізації розподіленої ПС (РПС) на основі сформульованих моделей ПС та РПС;

2) об'єктні сервіси, які підтримують об'єкти і класи, операції ЖЦ, послуги необхідні для розробки РПС в об'єктно-орієнтованому середовищі;

3) веб-сервіси, які базуються на інформаційних ресурсах Інтернет та забезпечують створення елементів РПС шляхом композиції або інтеграції компонентів ПС і сервісів, здатних до функціонування в Інтернеті або Інтранеті.

Сервіси можуть забезпечувати реалізацію базових функцій управління компонентами в середовищі:

1) пошук компонентів;

2) доступ до їх ресурсів;

3) організація обміну інформацією між компонентами;

4) динамічне управління функціонуванням, обумовленим сукупністю компонентів в розподіленому інформаційному середовищі.

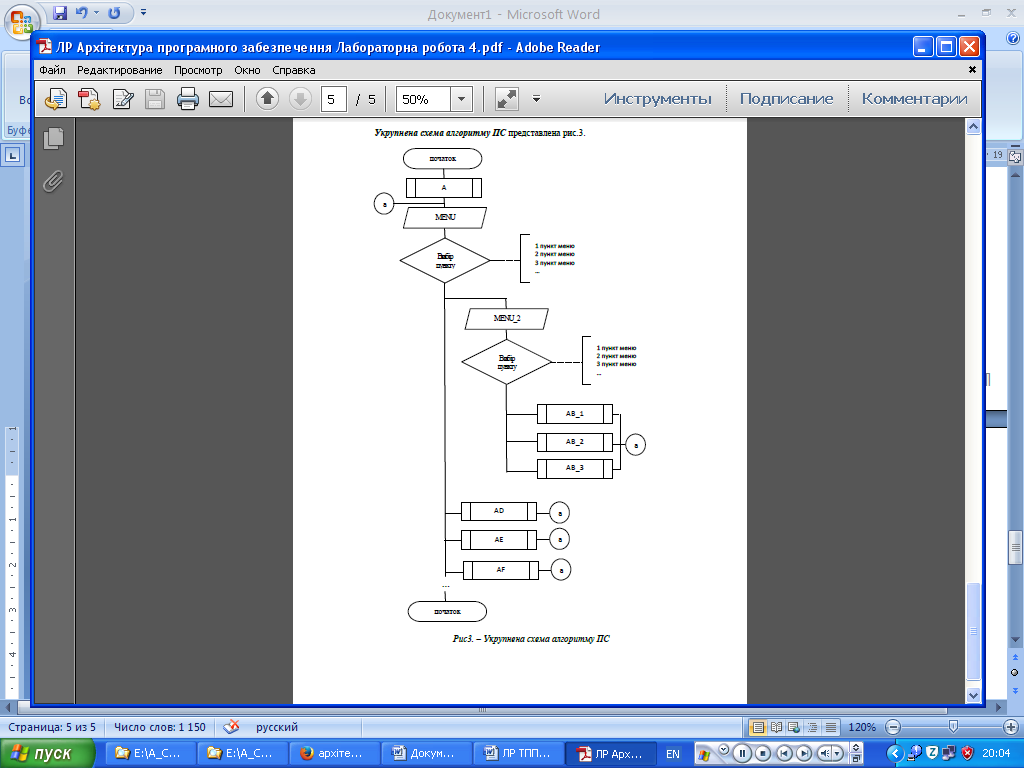
Модель сервісів РПС базується на уніфікації та сумісності, що дозволяє розглядати РПС як набір сервісів, їх функціональність і взаємодію.

Уніфікація сервісів досягається шляхом:

1) типізації функціональності сервісів та інших характеристик;

2) застосування уніфікованих мов для опису сервісів і їх взаємодії;

3) використання стандартних базових технологій.

****

1. **Зчеплення модуля** – це міра його залежності по способу передачі даних від інших модулів. Чим слабше зчеплення модуля з іншими модулями, тим сильніше його незалежність від інших модулів. [↑](#footnote-ref-1)