**Лабораторна робота №4. Планування системної інтеграції програмного забезпечення**

**Мета:**. Навчитися визначати перелік дій та їх послідовність для проведення системної інтеграції

**Завдання:**

1. Опрацювати лекцію №5 та теоретичні відомості.
2. Для розподіленої архітектури визначити перелік дій та їх послідовність для проведення системної інтеграції.
3. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва практичної роботи.
* Прізвище, група
* Назва проекту.
* Заповнені таблиця та відповіді на запитання.

По закінченню практичну 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:

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, **РІ4101Р**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

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

**Строк виконання цієї роботи ІПЗ-41 – 25.10.2021**

**ІПЗ-42 – 03.11.2021**

**Теоретичні відомості**

Поява й розвиток розподілених архітектур пов’язані з інтенсивним розвитком технічних і програмних засобів. У даному типі архітектури функціональні компоненти інформаційної системи розподіляються по наявних вузлах залежно від поставлених цілей і завдань. Можна виділити шість основних характеристик архітектури розподілених систем:

1. спільне використання ресурсів (як апаратних, так і програмних);
2. відкритість – можливість збільшення типів і кількості ресурсів;
3. паралельність – можливість виконання декількох процесів на різних вузлах системи (при цьому вони можуть взаємодіяти);
4. масштабованість – можливість додавати нові властивості й методи;
5. відмовостійкість – здатність системи підтримувати часткову функціональність за рахунок можливості дублювання інформації, апаратної і програмної складових.

До *недоліків розподілених систем* варто віднести:

* структурна складність;
* складно забезпечити достатній рівень безпеки;
* на керування системою потрібне велика кількість зусиль;
* непередбачена реакція на зміни.

Всі вони пов’язані в першу чергу зі складною структурою, різноплановим устаткуванням і складною системою розподілу прав доступу. Необхідно враховувати всі з них, інакше розроблена інформаційна система не зможе функціонувати в рамках очікуваних параметрів.

**Види розподілених архітектур.**

Існують такі види розподілених архітектур:

* архітектура «файл-сервер»;
* архітектура «клієнт-сервер»;
* архітектура web-додатків.

Файл-серверна архітектура має на увазі наявність виділеного мережного ресурсу для зберігання даних. Такий ресурс називається «файловим сервером». При такій архітектурі всі функціональні компоненти системи розташовані на користувальницькому комп’ютері, що називається «клієнтом», а самі дані перебувають на сервері.

Така організація системи має такі *переваги*:

* багатокористувальницький режим роботи з даними, що зберігаються на сервері;
* централізоване керування правами доступу до загальних даних;
* низька вартість розробки;
* висока швидкість розробки.

*Недоліки* файл-серверної архітектури:

* послідовний доступ до загальним даних і відсутність гарантії їхньої цілісності;
* продуктивність (залежить від продуктивності мережі, клієнта й сервера).

Класичне подання файл-серверної архітектури представлене на рис. 1.

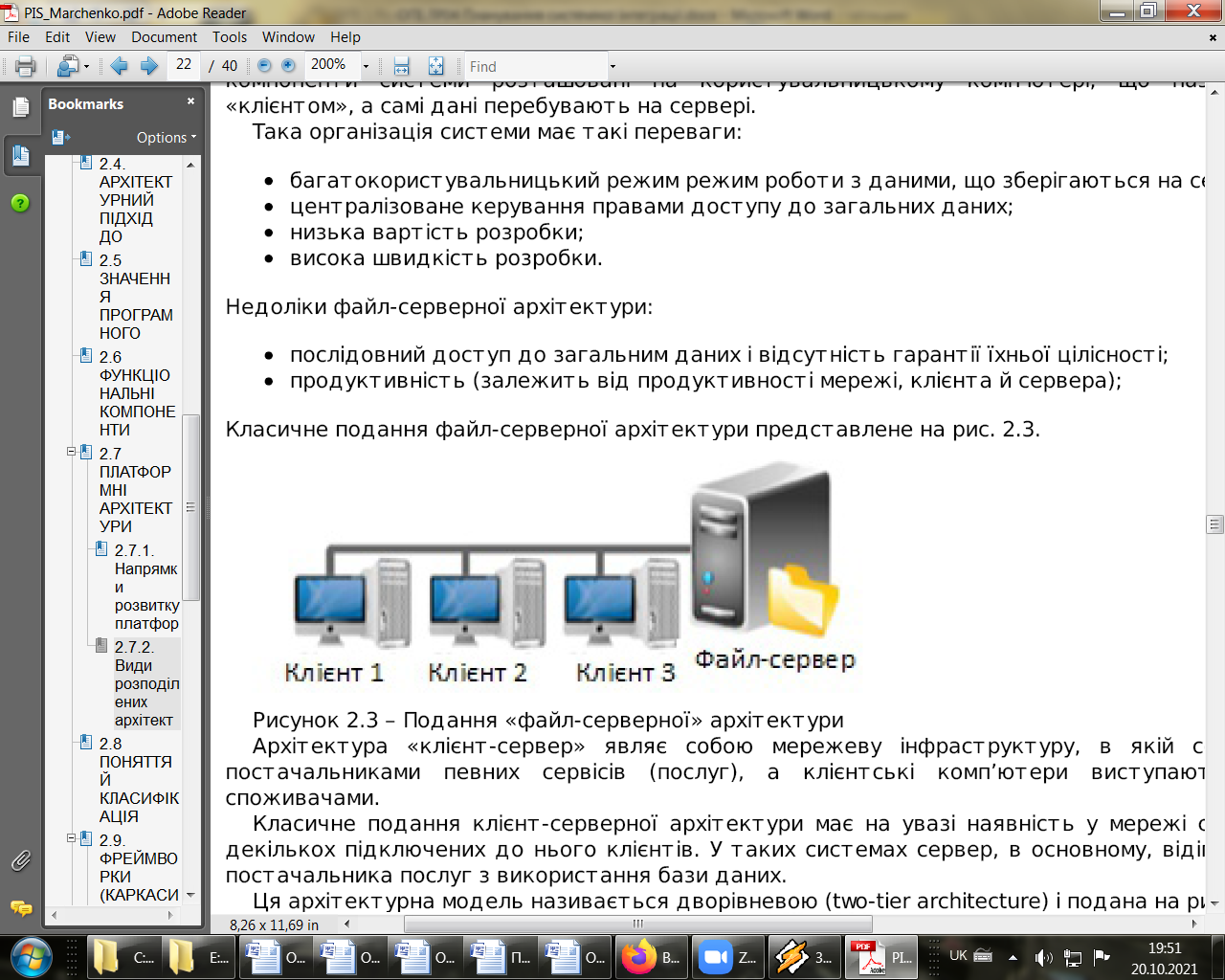


Рисунок 1 – Подання «файл-серверної» архітектури

Архітектура «клієнт-сервер» являє собою мережеву інфраструктуру, в якій сервери є постачальниками певних сервісів (послуг), а клієнтські комп’ютери виступають їхніми споживачами. Класичне подання клієнт-серверної архітектури має на увазі наявність у мережі сервера й декількох підключених до нього клієнтів. У таких системах сервер, в основному, відіграє роль постачальника послуг з використання бази даних. Ця архітектурна модель називається дворівневою (two-tіer archіtecture) і подана на рис. 2.

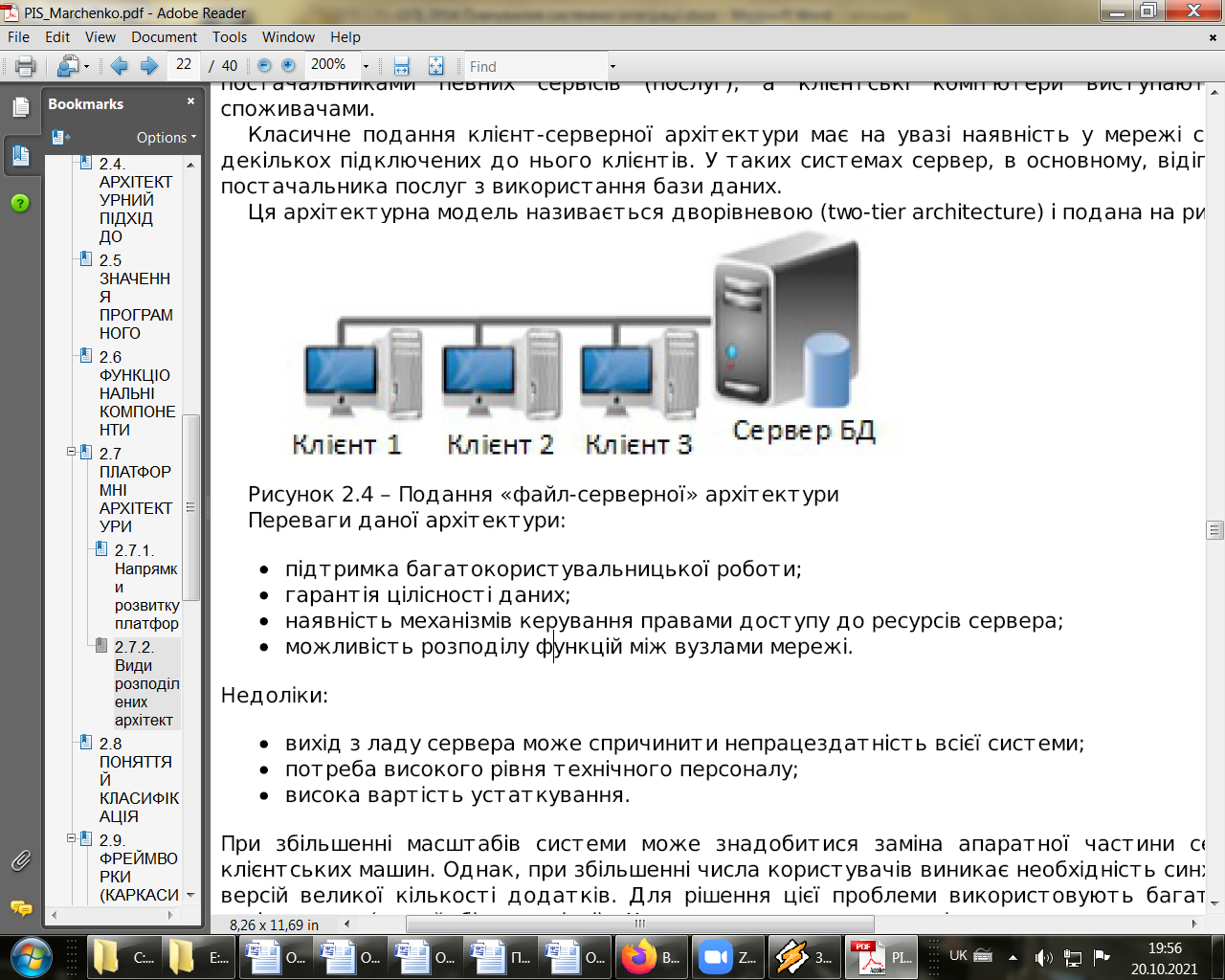


Рисунок 2 – Подання «файл-серверної» архітектури

*Переваги* даної архітектури:

* підтримка багатокористувальницької роботи;
* гарантія цілісності даних;
* наявність механізмів керування правами доступу до ресурсів сервера;
* можливість розподілу функцій між вузлами мережі.

*Недоліки*:

* вихід з ладу сервера може спричинити непрацездатність всієї системи;
* потреба високого рівня технічного персоналу;
* висока вартість устаткування.

При збільшенні масштабів системи може знадобитися заміна апаратної частини сервера й клієнтських машин. Однак, при збільшенні числа користувачів виникає необхідність синхронізації версій великої кількості додатків. Для рішення цієї проблеми використовують багатоланкові / багаторівневі архітектури (три й більше рівні). Частина загальних додатків переноситься на спеціально виділений сервер, тим самим знижуються вимоги до продуктивності клієнтських машин. Клієнти з низькою обчислювальною потужністю називають «тонкими клієнтами», а з високою продуктивністю – «товстими клієнтами». При багатоланковій архітектурі з виділеним сервером додатків існує можливість використання портативних пристроїв. Багатоланкова архітектура показана на рис. 3.

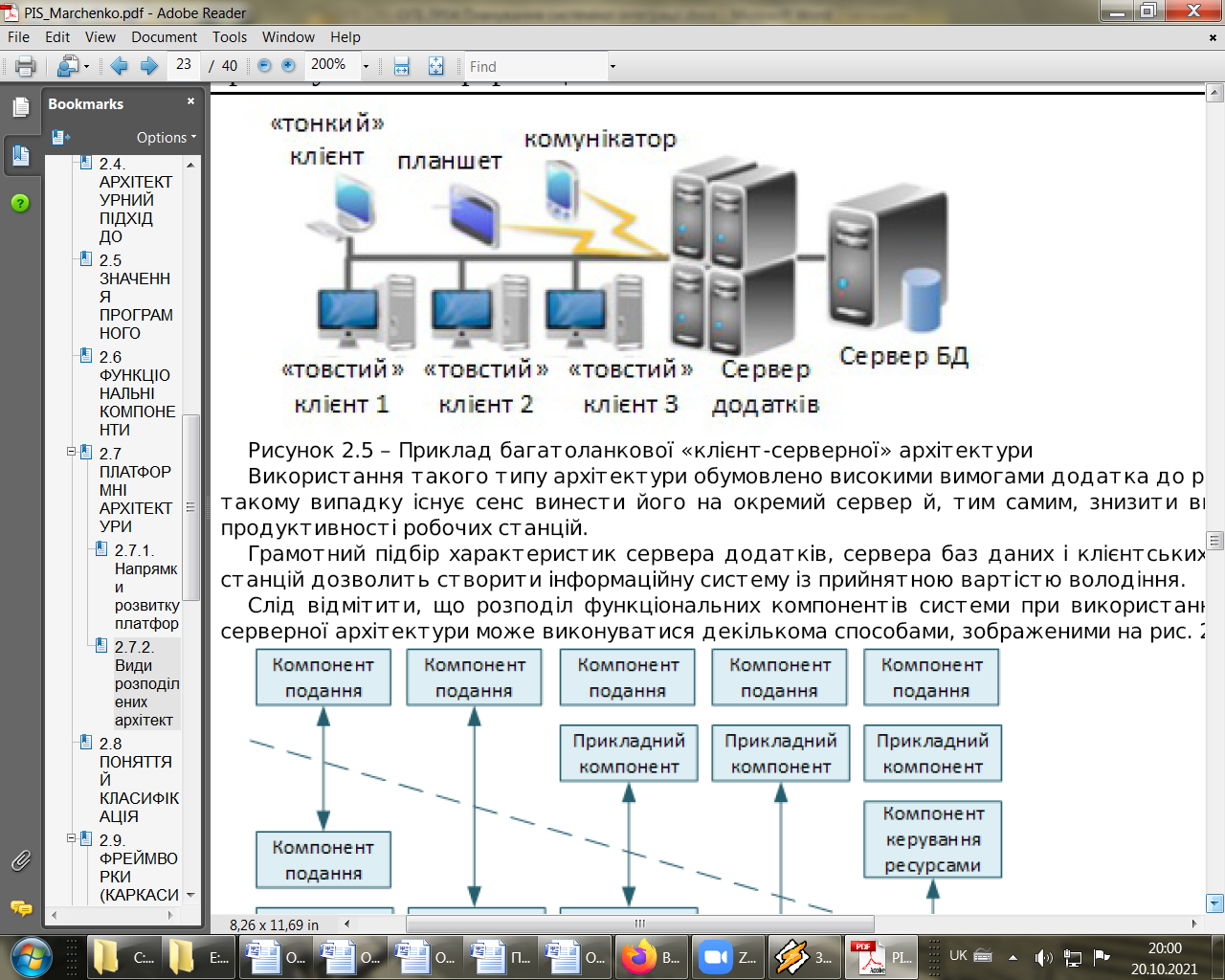


Рисунок 3 – Приклад багатоланкової «клієнт-серверної» архітектури

Використання такого типу архітектури обумовлено високими вимогами додатка до ресурсів. У такому випадку існує сенс винести його на окремий сервер й, тим самим, знизити вимоги до продуктивності робочих станцій. Грамотний підбір характеристик сервера додатків, сервера баз даних і клієнтських робочих станцій дозволить створити інформаційну систему із прийнятною вартістю володіння. Потрібно відмітити, що розподіл функціональних компонентів системи при використанні клієнт-серверної архітектури може виконуватися декількома способами, зображеними на рис. 4.

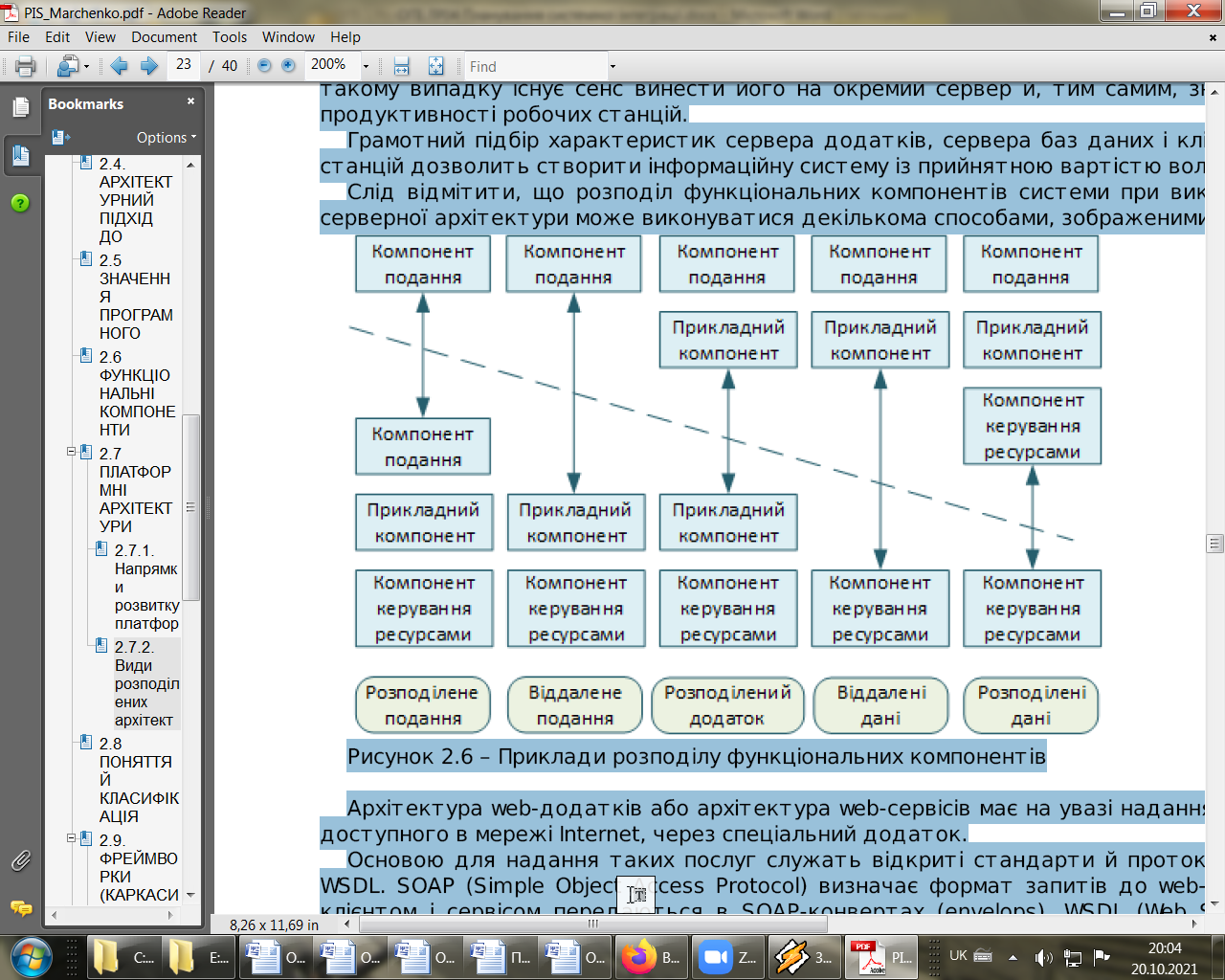


Рисунок 4 – Приклади розподілу функціональних компонентів

Архітектура web-додатків або архітектура web-сервісів має на увазі надання деякого сервісу, доступного в мережі Іnternet, через спеціальний додаток. Основою для надання таких послуг служать відкриті стандарти й протоколи SOAP, UDDІ й WSDL.

SOAP (Sіmple Object Access Protocol) визначає формат запитів до web-сервісів. Дані між клієнтом і сервісом передаються в SOAP-конвертах (envelops).

WSDL (Web Servіce Descrіptіon Language) служить для опису інтерфейсу надаваного сервісу. Перед розгортанням web-додатка потрібно скласти його опис, указати адреса, список підтримуваних протоколів, перелік припустимих операцій, а так само формати запитів і відповідей.

UDDІ (Unіversal Descrіptіon, Dіscovery and Іntegratіon) являє собою протокол пошуку web-сервісів у мережі Іnternet. Пошук здійснюється за їхніми описами, які розташовані в спеціальному реєстрі.

Архітектура таких сервісів схожа за концепцією з багатоланкової клієнт-серверною, однак, сервера додатків і баз даних розташовуються в мережі Іnternet.

Можна виділити три технології, які можливо використати для побудови розподіленої архітектури web-сервісу:

* EJB (Enterprіse JavaBeans).
* DCOM (Dіstrіbuted Component Object Model).
* CORBA (The Common Object Request Broker Archіtecture).

Ідеєю для створення EJB було бажання створити інфраструктуру для легкого додавання й видалення компонентів зі зміною функціональності сервера. EJB дозволяє розроблювачам створювати власні додатки із заздалегідь створених модулів. При цьому можлива їхня зміна, що робить процес розробки гнучким і набагато більше швидким. Дана технологія сумісна з CORBA й Java APІ. Взаємодія між клієнтів і сервером у цьому випадку представляється як взаємодія EJB-об’єкта, що генерується спеціальним генератором, і EJB-компонента, написаного розроблювачем. При необхідності викликати метод в EJB-компонента, що перебуває на сервері, викликається однойменний метод EJB-об’єкта, розташованого на стороні клієнта, що зв’язується з необхідним компонентом і викликає необхідний метод.

Переваги EJB:

* просте й швидке створення;
* Java-оптимізація;
* кросплатформність;
* вбудована безпека.

Недоліки EJB:

* складність інтегрування з додатками;
* погана масштабованість;
* низька продуктивність;
* відсутність міжнародної стандартизації.

DCOM являє собою розподілену програмну архітектуру від компанії Mіcrosoft. З її допомогою програмний компонент одного комп’ютера може передавати повідомлення програмному компоненту іншого комп’ютера, причому з’єднання встановлюється автоматично. Для надійної роботи потрібно забезпечити захищене з’єднання між зв’язаними компонентами, а також створити систему перерозподіл трафіку.

Переваги DCOM:

* незалежність від мови;
* динамічне знаходження об’єктів;
* масштабованість;
* відкритий стандарт.

Недоліки DCOM:

* складність реалізації;
* залежність від платформи;
* пошук через службу Actіve Dіrectory;
* відсутність іменування сервісів через URL.

Технологія CORBA розглядає всі додатки в розподіленій системі як набір об’єктів. Об’єкти можуть одночасно виступати в ролі клієнта й сервера, викликаючи методи інших об’єктів і відповідаючи на їхні виклики. Застосування даної технології дозволяє будувати системи, що перевершують по складності й гнучкості системи з архітектурою клієнт-сервер (як дворівневої, так і трирівневої).

Переваги CORBA:

* незалежність від платформи;
* незалежність від мови;
* динамічні виклики;
* динамічне виявлення об’єктів;
* масштабованість;
* індустріальна підтримка.

Недоліки:

* відсутність іменування по URL;
* практично повна відсутність реалізації CORBA-сервісів.

При грамотному підході до побудови архітектури інформаційної системи може знадобитися використання відразу декількох з розглянутих технологій. Кожна з них буде реалізовувати певний функціонал, що може зажадати також декількох типів платформних архітектур. В одній системі можуть працювати кілька файлів-серверів, кілька серверів додатків і кілька серверів баз даних. У такий спосіб можна розподіляти навантаження в системі або групувати набори сервісів відповідно виконуваним функціях.

**ІНТЕГРАЦІЯ ІНФОРМАЦІЙНИХ СИСТЕМ**

Термін інтеграція має широке значення. Під ним можна розуміти об’єднання інформаційних систем, додатків, різних компаній або людей.

Інтеграція поділяється на зовнішню і внутрішню. Внутрішня інтеграція передбачає об’єднання корпоративних додатків в одній організації (Enterprise Application Integration), а зовнішня – інтеграцію інформаційних систем організацій (Business-to-Business Application).

**Інтеграційні підходи.**

Існують чотири основні типові інтеграційні підходи:

1. інтеграція на рівні даних;
2. інтеграція на рівні бізнес-функцій і бізнес-об’єктів;
3. інтеграція на рівні бізнес-процесів;
4. портали.

Інтеграція на рівні даних (Information- Oriented Integration) має на увазі наявність в системах баз даних, для роботи з якими необхідно розробити єдиний програмний інтерфейс.

До основних технологічних рішень даного підходу відносяться:

* системи реплікації даних;
* федеративні бази даних;
* використання API для доступу до EPR- системам[[1]](#footnote-1).

*Реплікація* є процесом синхронізації даних між різними джерелами. Необхідність у цьому виникає в момент зміни блоку інформації в розподілених системах зберігання для гарантії коректності і несуперечності даних, які використовуються в усіх модулях або додатках інформаційної системи. Зазвичай функції реплікації покладають на проміжне програмне забезпечення.

*Федеративні бази даних* (Federated Database Systems) надають єдиний інтерфейс до розподілених даних. Це забезпечує інтеграцію безлічі автономних даних, які можуть бути фізично розташовані на різних пристроях в мережі. Такі бази даних прийнято називати віртуальними.

*Використання API для доступу до ERP-систем* покликане спростити механізми обміну інформацією між одними додатками і програмним забезпеченням, призначеним для управління функціонуванням виробничих інформаційних систем (ERP).

Інтеграція на рівні бізнес-функцій і бізнес об’єктів передбачає реалізацію спільно використовуваних служб (сервісів). Служба може бути набором функцій, використовуваному в декількох додатках. Набір служб і буде бізнес-функціями. При використанні сервіс-орієнтованої архітектури, бізнес-функції можна розглядати як бізнес - сервіси, а при компонентному підході – бізнес-об’єктами (бізнес-компонентами).

Інтеграція на рівні бізнес-процесів розрізняється залежно від рівня інтеграції. При внутрішній інтеграції взаємодіє велика кількість сервісів, а при зовнішній інтеграції, в основному, два. Самі бізнес-процеси функціонують над виділеними службами, для управління якими існує спеціальний інтерпретована мова.

Портали можна вважати графічними інтерфейсами бізнес-процесів, оскільки вони призначені для персоналізованого доступу до інформації та консолідації даних з декількох джерел.

Головне призначення процесу інтеграції – об’єднання функцій додатків або модулів для надання нової функціональності.

При інтеграції додатків можна виділити два основних типи завдань:

* завдання інтеграції корпоративних додатків;
* завдання інтеграції додатків з різних інформаційних систем.

Для вирішення завдань першого типу застосовуються системи EAI, які іноді називаються A2A (Application-to-Application Integration), а для вирішення завдань другого типу застосовуються системи B2B (Business-to-Business Integration).

У деяких ситуаціях дуже складно визначити різницю між інтеграцією A2A і B2B, оскільки складність деяких рішень всередині інформаційних систем може перевищувати складність рішень для їх спільного функціонування.

**Топології інтеграції.**

Існують три альтернативних топології інтеграції:

1. Точка-точка (Point-to-Point).

2. Шлюз (hub-and-spoke).

3. Шина (Bus).

У топології «точка-точка» всі об’єкти мають прямі зв’язки один з одним (рис. 5, а). Слід зазначити, що кожен зв’язок може бути реалізований яким завгодно способом. Варіанти реалізації залежать від вимог і характеристик взаємодії між об’єктами. До недоліків топології можна віднести:

* недостатня гнучкість;
* складність підтримки численних з’єднань «точка-точка»;
* зміни одного об’єкта впливають на решту;
* логіка маршрутизації часто програмується в коді об’єктів;
* відсутність загальної моделі безпеки;
* використання різних API;
* низька надійність;
* складність створення фреймворків;
* складність підтримки асинхронного взаємодії;

Для скорочення числа використовуваних інтерфейсів слід використовувати топологію із загальним шлюзом (рис. 5, б) або топологію із загальною шиною (рис. 5, в). Такі моделі інтеграції реалізуються на рівні проміжного програмного забезпечення.

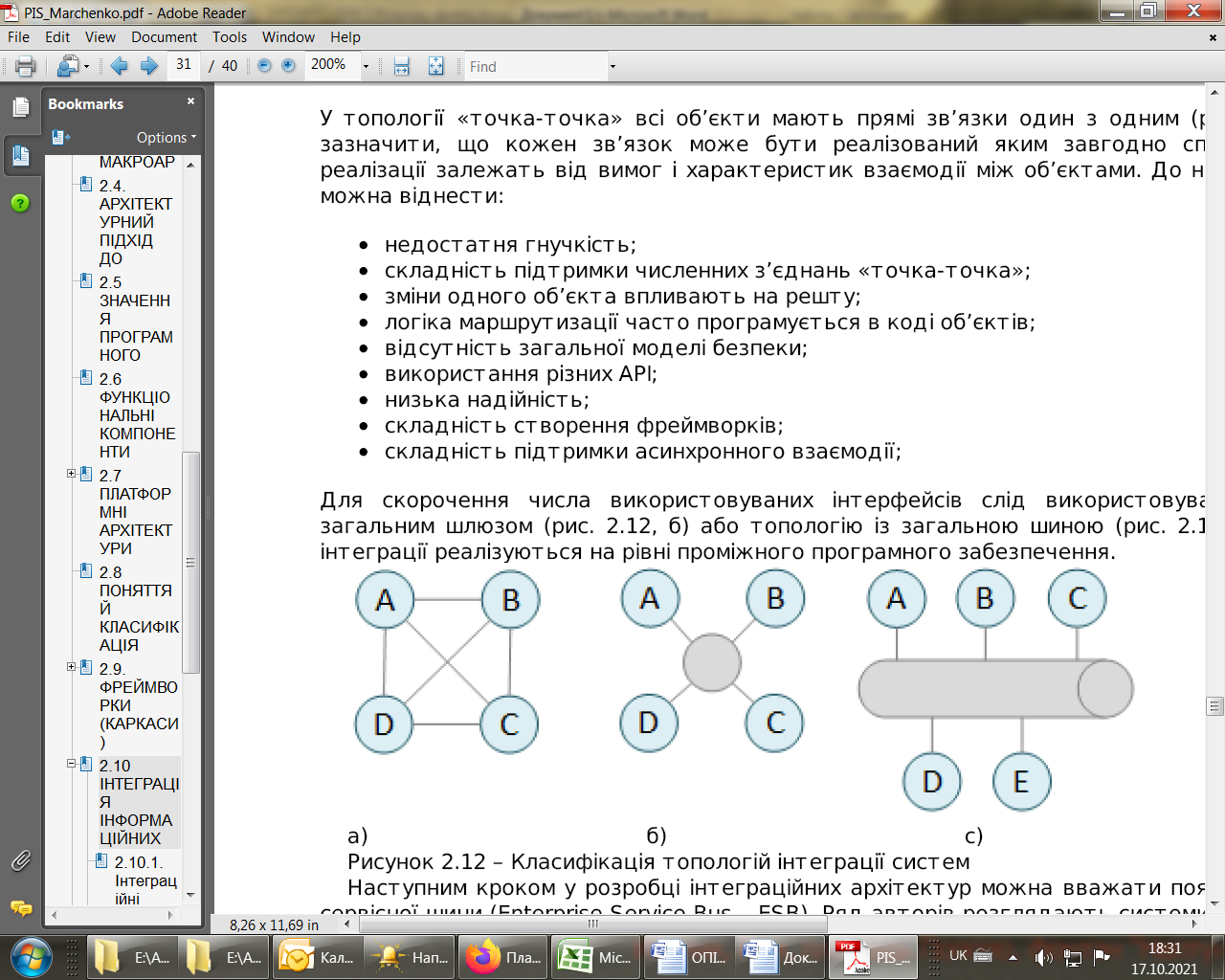


Рисунок 5 – Класифікація топологій інтеграції систем

Наступним кроком у розробці інтеграційних архітектур можна вважати появу корпоративної сервісної шини (Enterprise Service Bus – ESB). Ряд авторів розглядають системи ESB, як наступну сходинку розвитку EAI. Однак є кілька відмінностей:

EAI – централізована архітектура, з обміном інформації через хаб (брокер), а ESB – шинна архітектура, яка може бути реалізована у вигляді декількох розподілених систем; на відміну від EAI, ESB орієнтована на використання відкритих стандартів.

Ці дві відмінності наочно демонструють можливість використання ESB як інтеграційної платформи, що дозволяє використовувати різні механізми інтеграції. ESB дозволяє проводити як внутрішню, так і зовнішню інтеграції, і являє собою шину (backbone), що працює як слабкозв’язана система, керована подіями.

Концепції сервіс-орієнтованих архітектур (СОА) і ESB дуже сильно пов’язані. ESB підтримує принцип реалізації СОА: поділ подання служби і її реалізації.

Функції ESB:

* надання інтерфейсів взаємодії;
* відправлення і маршрутизація повідомлень;
* перетворення даних;
* реакція на події;
* управління політиками;
* віртуалізація.

На підставі функцій ESB, можна сформувати типовий список вимог, що пред’являються з боку користувачів:

* велика пропускна здатність;
* підтримка декількох стилів інтеграції;
* забезпечення можливості додатків працювати з сервісами як безпосередньо, так і через адаптери.

ESB є, по суті, логічним компонентом архітектури, що призводить інтеграційну інфраструктуру у відповідність принципу СОА. Архітектурами, побудованими за принципом ESB, складніше управляти, але вони більш гнучкі і масштабовані, більш того, впровадження СОА не зажадає змін у всіх елементах системи, в результаті чого зможе відбуватися поетапно.

Можна уявити ESB у вигляді п’ятирівневої структури:

* рівень сполучення (адаптери та інтерфейси);
* транспортна підсистема;
* рівень реалізації бізнес-логіки;
* рівень управління бізнес-процесами;
* рівень бізнес-управління.

Рівень сполучення покликаний вирішувати проблему використання різних інтерфейсів. На цьому рівні функціонують адаптери, які відстежують події в додатках і в інтеграційній підсистемі, і забезпечують перетворення переданих об’єктів при взаємодії з транспортною підсистемою.

Існує можливість, окрім заздалегідь створених адаптерів інтеграційної платформи, використовувати створені самостійно.

Адаптери можна розділити на дві категорії: технологічні (застосовуються для інтеграції технологічних компонент, за відсутності у них API), адаптери для додатків (застосовуються для інтеграції з конкретним додатком).

Транспортна підсистема надає можливість асинхронного взаємодії інтегрованим додаткам. Даний рівень також відповідає за управління та безпеку інформації, може виконувати маршрутизацію повідомлень і їх обробку.

Рівень реалізації бізнес-логіки надає функції для трансформації і маршрутизації повідомлень. На цьому рівні функціонують брокери повідомлень, що обмінюються повідомленнями через транспортну підсистему.

Брокер повідомлень може виконувати такі функції:

1. Прийняття повідомлень і їх відправка за вказаними адресами.

2. Перетворення форматів повідомлень.

3. Агрегування і фрагментація повідомлень.

4. Взаємодія з репозиторіями.

5. Вибірка даних через виклики Web-служб.

6. Обробка помилок і подій.

7. Маршрутизація повідомлень за адресою, вмістом, темою.

Управлення бізнес-процесами на однойменному рівні здійснюється за допомогою BPEL (Business Process Execution Language) за допомогою web-сервісів.

Рівень бізнес-управління представляє з себе надбудову на попередньому рівню і призначений для управління бізнес-процесами в термінах відповідної предметної області.

Підхід ESB має безліч переваг і дозволяє будувати інтеграційні архітектури практично будь-якої складності. Типова структура інтеграційної системи представлена на рис. 6.

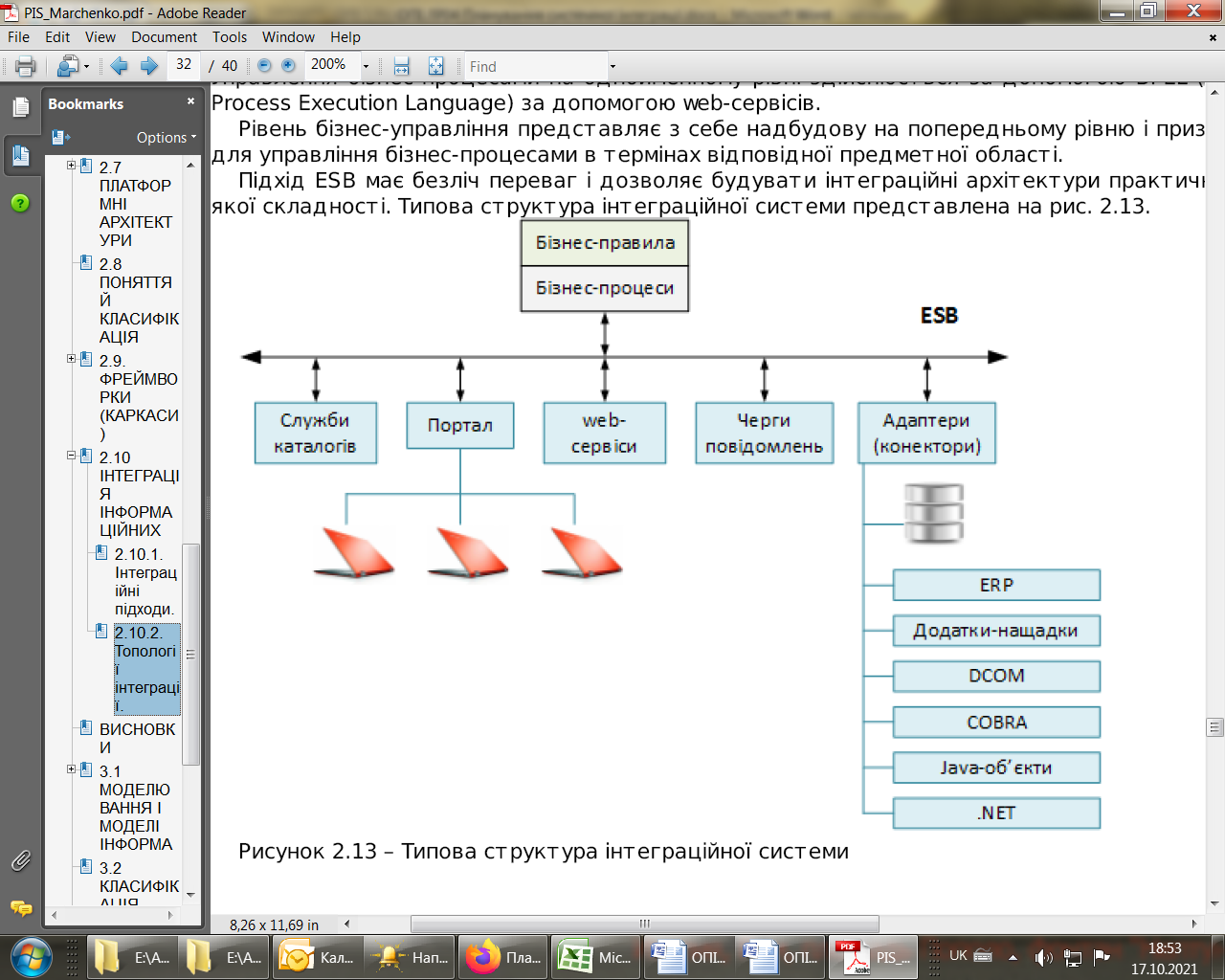


Рисунок 6 – Типова структура інтеграційної системи

Узагальнемо.

Під архітектурою програмних систем розуміється сукупність рішень щодо:

* організації програмної системи;
* вибору структурних елементів, що складають систему і їх інтерфейсів;
* поведінки цих елементів у взаємодії з іншими елементами;
* об’єднання цих елементів у підсистеми;
* архітектурного стилю, що визначає логічну й фізичну організацію системи: статичні і
* динамічні елементи, їх інтерфейси і способи їх об’єднання.

Архітектура програмної системи охоплює не тільки її структурні і поведінкові аспекти, але й правила її використання та інтеграції з іншими системами, функціональність, продуктивність, гнучкість, надійність, можливість повторного застосування, повноту, економічні та технологічні обмеження, а також питання для користувача інтерфейсу.

По мірі розвитку програмних систем все більшого значення набуває їх інтеграція одна з одною з метою побудови єдиного інформаційного простору підприємства. Як можна бачити з вищенаведених визначень інтеграція є найважливішим елементом архітектури.

Для того, щоб побудувати правильну і надійну архітектуру і грамотно спроектувати інтеграцію програмних систем необхідно чітко слідувати сучасним стандартам в цих областях.

Без цього велика ймовірність створити архітектуру, яка нездатна розвиватися і задовольняти зростаючим потребам користувачів ІТ. В якості законодавців стандартів у цій галузі виступають такі міжнародні організації як SEI (Software Engineering Institute), WWW (консорціум World Wide Web), OMG (Object Management Group), організація розробників Java - JCP (Java Community Process), IEEE (Institute of Electrical and Electronics Engineers) та інші

Кожен архітектор інформаційних систем повинен враховувати три важливі рекомендації.

1. Контролюйте рамки проекту.

2. Завжди пам’ятайте, для чого потрібна проектуєма модель. Моделі даних призначені для реалізації, а інформаційні моделі – для документації та обміну інформацією з людьми.

3. Не дозволяйте інструментальним засобам визначати ваш погляд на речі. Важко документувати обширні і несуворі відношення спадкування, використовуючи тільки SQL або ER-діаграми.

За матеріалами

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiokOuY6tHzAhXsAxAIHeIODvMQFnoFCKMBEAE&url=http%3A%2F%2Fkist.ntu.edu.ua%2FtextPhD%2FPIS\_Marchenko.pdf&usg=AOvVaw3R88zA\_Buzw78FfSpyudTz

1. Абревіатура ERP походить від англійського (Enterprise Resource Planning System), що перекладається як система планування ресурсів підприємства. Продукт являє собою цілісну систему управління компанії, яка об'єднує ресурси підприємства з різним підрозділам. То є всі необхідні ресурси, підрозділи, функції та інший інструментарій, необхідний для ефективної роботи, знаходиться в одній комп'ютерній системі. Доступ до інформації отримують всі підрозділи на підприємстві, що суттєво спрощує роботу і забезпечує обмін інформацією. Система:

   * інтегрує завдання і бази даних всіх відділів компанії;
   * забезпечує створення єдиного інформаційного середовища;
   * допомагає у вирішенні будь-яких завдань на підприємстві.

   [↑](#footnote-ref-1)