**Лекція 14.Системна інтеграція програмного забезпечення**

**Систе́мна інтегра́ція** в інженерії — поєднання компонентів підсистем в єдину систему та забезпечення роботи окремих підсистем як єдиної системи. В області інформаційних технологій **системна інтеграція є процесом об'єднання різних обчислювальних систем і програмних за стосунків фізично або функціонально.** Системна інтеграція полягає у розробці комплексних рішень, призначених для досягнення максимальної ефективності функціонування системи шляхом налагодження ефективної взаємодії її підсистем, зокрема, для отримання комплексного вирішення низки інформаційних завдань. *Системна інтеграція* як діяльність, пов'язана із встановленням та налагодженням операційних систем, баз даних, офісних застосунків, засобів зв'язку, структурованих кабельних систем і активних мережевих пристроїв, пристроїв зберігання даних, підключення до Інтернету, контролю доступу, живлення, системи аварійної сигналізації і т.д.

Системний інтегратор об'єднує окремі системи, використовуючи різні методи, такі як зв'язування комп'ютерів у мережі, інтеграція корпоративних застосунків, управління бізнес-процесами або ручне програмування.

Таким чином, інтегрована система – це така система, яка має жорсткі зв’язки елементів та компонентів з елементами інших систем і не може самостійно від інших систем повноцінно функціонувати. Архітектура - це інтегрована структура для розгортання або підтримки існуючих ІТ - засобів і для придбання нових, що служить реалізації стратегії організації і досягненню цілей бізнесу. Повна ІТ - архітектура повинна включати як логічні, так і технічні компоненти.

**Сладові системної інтеграції у ІТ**:

* *інтеграція програмних застосунків* — діяльність, пов'язана із впровадженням спеціалізованих програмних рішень (галузевих);
* *інтеграція даних* — діяльність, що має на меті оптимальну організацію бази даних, при якій реалізовано всі необхідні взаємозв'язки між елементами даних, але база не містить повторів і зайвих елементів (так як по мірі використання бази даних вона має тенденцію роззосереджуватись)
* *мережева інтеграція* — діяльність, пов'язана із створення локальних і глобальних мереж, структурованих кабельних систем, впровадження оптоволоконних технологій передачі даних і т.д.;

Прикладом системної інтеграції програмних компонентів є структура операційної системи. Операційна система містить ряд окремих функціональних компонентів, кожен з яких призначений для вирішення своєї групи завдань, але правильно налагоджена взаємодія між ними дозволяє ефективно використовувати апаратні можливості ЕОМ в цілому. Зокрема, [операційна система](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [Windows](https://uk.wikipedia.org/wiki/Windows) містить:

|  |  |
| --- | --- |
| * диспетчер ядра, * диспетчер процесів і потоків, * диспетчер віртуальної пам'яті, * підсистема вводу-виводу, * диспетчер кешу, | * диспетчер Plug and Play, * диспетчер електроживлення, * диспетчер конфігурації, * диспетчер об'єктів. |

Сучасні WEB-сайти є результатом узгодженої роботи множини компонентів, інтеграція яких дозволяє створювати гнучкі та легко налагоджувані WEB-ресурси. В загальному випадку робота сайту може складатися з таких компонентів:

* *Сторінки сайту*. В найпростішому випадку (статичні) сторінки можуть бути описані за допомогою мови [html](https://uk.wikipedia.org/wiki/Html" \o "Html). Але для *динамічного* формування виду сторінки використовують одну із спеціальних скриптових мов програмування, наприклад, PHP або ASP.NET.
* [*Веб-сервер*](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) приймає запити від клієнтів та надає їм відповіді у виді html-сторінки, потокових або інших даних. Найпоширенішими [веб-серверами](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80" \o "Веб-сервер)  є [Apache](https://uk.wikipedia.org/wiki/Apache" \o "Apache) та [IIS](https://uk.wikipedia.org/wiki/IIS)
* *Програмний засіб для динамічного генерування сторінок*. Ним може бути [гіпертекстовий](https://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) [препроцесор](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80) PHP (для опрацювання сторінок, описаних за допомогою PHP) або платформа ASP.NET (для опрацювання сторінок, описаних засобами ASP.NET)
* *База даних* містить дані для сайту у виді послідовності структурованих записів.
* *Система керування базами даних*, яка забезпечує можливість створення, збереження, оновлення, пошук інформації та контролю доступу в базах даних.

В загальному випадку процес відображення сторінки для користувача виглядає так. Користувач за допомогою веб-огдядача робить запит до певної сторінки сайту. Запит надходить до [веб-сервера](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), який звертається до файлу, що містить код сторінки. У випадку статичної сторінки у форматі [html](https://uk.wikipedia.org/wiki/Html" \o "Html) сервер відразу надсилає її користувачеві. Для динамічних сторінок [веб-сервер](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) викликає відповідний програмний засіб (наприклад, [PHP](https://uk.wikipedia.org/wiki/PHP)), який при необхідності звертається за допомогою СУБД до бази даних та формує кінцевий html-код. Після цього [веб-сервер](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) надсилає кінцевий код сторінки до клієнта, який за допомогою веб-оглядача побачить його у виді сторінки сайту.

Системна інтеграція у виробництві виконується в автоматизованих системах управління, системах автоматики. Так, **с**истеми промислової автоматики включають:

* *машини і апарати*, що виробляють продукцію чи реалізують технологічні процеси: обладнання для складання і складальні лінії, обладнання міжопераційного транспортування, системи пакування, преси, роботи і т.д.;
* *контрольно-вимірювальне обладнання* на технологічних машинах і апаратах (вимірювальна апаратура, задавачі, вимірювальні перетворювачі, показуючи засоби) та системи візуалізації перебігу технологічних процесів;
* *виконавчі механізми*: клапани, двигуни, приводи, засувки, насоси і насоси-дозатори;
* *пристрої керування*: програмовані логічні контролери, промислові комп'ютери, операторські панелі;
* *Програмне забезпечення* для керування і візуалізації виробничих і промислових процесів: програмне забезпечення контролерів, [HMI](https://uk.wikipedia.org/wiki/%D0%9B%D1%8E%D0%B4%D0%B8%D0%BD%D0%BE-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)/[SCADA](https://uk.wikipedia.org/wiki/SCADA)[[1]](#footnote-1), розподілених систем керування (DCS);
* Засоби комунікації: промислові мережі, радіо-модемний зв’язок, GPRS[[2]](#footnote-2).

З точки зору впровадження промислової автоматизації як завершеного проекту, можна виділити два основних його етапи:

* *проектування*: може бути реалізованим спеціалізованими проектними організаціями, хоча більшість інтеграторів створюють власні рішення, які потім впроваджують. Це випливає з того, що компанії, які виступають генеральними підрядниками, пропонують технологію і мають свої власні команди розробників і електриків чи механіків, що впроваджують ці проекти. Таким шляхом створюються, як машини-автомати для виготовлення чи складання та виробничі лінії, а також інфраструктурні рішення (наприклад, для очищення стічних вод, теплопостачання чи котелень). Відділення процесу проектування від впровадження має місце для випадку великих проектів. Потім, з фінансових міркувань інвесторами, часто проводиться тендер чи конкурс для вибору виконавця, що надає можливість проектанту бути залученим до впровадження розробки;
* *впровадження* — це поєднання у функціональну одиницю обладнання, програмного забезпечення та ліцензій. Вартість реалізації включає в себе вартість необхідного обладнання, програмного забезпечення, ліцензій та вартість послуги, як правило, принаймні рівною значенню необхідних закупок компонентів. Вартість послуги зазвичай включає в себе витрати на впровадження системи, роботу програмістів, консалтинг та навчання користувачів, технічну документацію, послуги впровадження та розробки застосунків. Від рішення інвестора залежить спосіб отримання рішення: «під ключ» (постачання обладнання і програмного забезпечення виконавцем) чи придбання складових компонентів системи самостійно.

Більш ніж дві третини часу та фінансів в ІТ витрачається на «склеювання» несумісного та спроби узгодити модулі, написані різними людьми в різний час на різних мовах з використанням різних технологій та під різні платформи. Такі **фактори впливають на інтеграцію**.

* **Прискорення процесів**. Розвиток організації вимагає змінювати структури даних, бізнес – процеси, дизайн, користувацький інтерфейс. Причому два останніх аспекти знаходяться в постійній зміні.
* **Розподіленість.** Організаціїукрупнюються, розростаються, а задачі, що ними вирішуються стають все більше комплексними, з‘являється організаційна та географічна роззосередженість.
* **Гетерогенність**. В крупному проекті, майже ніколи нема можливості притримуватися платформ та інструментів одного виробника, що вимагає враховувати та підтримувати особливості декількох платформ.
* **Спадковість.** Неможливість повністю відмовитись від застарілих ліцензійних систем, морально застарілих технологій, старого апаратного забезпечення, особливо якщо вони дають непогані показники продуктивності та надійності, але не сприяють інтеграції.
* **Хаотичність**. Не завжди є можливість повністю формалізувати, специфікувати та структуризувати дані, внаслідок чого частина моделі “слабко-зв‘язаною”, не піддається або слабко піддається машинній обробці, аналізу, індексації, обрахунку.
* **Обумовленість**. Інформаційні системи часто обмежені не тільки технічними рамками, але й звичками людей, особливостями законодавства, та великою кількістю інших факторів, які не залежать від розробника.
* **Інтерактивність**. Споживач інформації постійно підвищує свої очікування щодо швидкості реакції системи, швидкодії, та оперативності доставки інформації. Більшість процесів наближуються до виконання в реальному часі.
* **Мобільність**. Користувач систем став переміщуватися швидше, а взаємодія с ним ведеться через канали зв‘язку загального користування в транспорті, вдома и на вулиці тощо.
* **Безпека**. Перехід на використання передачі даних через Інтернет вимагає застосування шифрування.
* **Високонавантажуваність**. На складність інтеграції впливають: кількість користувачів в системі, інтенсивність потоку обробки даних, обсяги даних та ресурсоємність обчислень.
* **Неперервність циклу роботи**. Інтеграція и апгрейд систем майже завжди повинні проводитися без зупинки їх функціонування, поступово та непомітно для організації та її клієнтів.
* **Міжсистемна інтеграція**. Часто потрібно інтегруватися з партнерами, клієнтами, поставщиками, підрядчиками, а інколи з державними структурами.

**Параметри, які визначають складність інтеграції та відповідні рішення**

* **Концептуальна різниця** — засновується на тому, що розробники різних систем з самого початку прийняли різні рішення та допущення,які концептуально не узгоджуються між собою. **Вирішується** введенням ще одного рівня абстракції, який концептуально не суперечить обом підходам. Є два варіанти реалізації: (а) коли створена система стає централізованою, а дві та більше, що інтегруються, перетворюються у підсистеми та (б) коли ми використовуємо архітектуру брокера (посередника, який не є центром), при цьому системи залишаються незалежними, а брокер забезпечує прошарок між ними.
* **Технологічна різниця** — коли маємо несумісні формати обміну даними, протоколи взаємодії та інтерфейси. **Вирішується** написанням конвертів, прошарків, брокерів тощо, не зовсім красивих, але достатньо надійних.
* **Несумісність ліцензій.**  Рішення може бути в кожному випадку індивідуальне, на організаційному рівні.

**Засоби вирішення проблем, що виникають при інтеграції**

* **Стандартизація** — потрібно і важливо використовувати якнайбільше міжнародних, державних та галузевих стандартів, а при їх нестачі, але необхідності потрібно вводити корпоративні стандарти та просувати їх у відповідних організаціях для прискорення поширення и популяризації.
* **Інтеграція на рівні брокерів**. **Переваги**: універсальність — практично завжди можна створити додатковий програмний модуль, який буде звертатися в обидві системи, та ще й в різний спосіб (наприклад, в одну через базу даних, а в іншу через RPC[[3]](#footnote-3)). **Недоліки**: складність, трудомісткість, і відповідно висока вартість розробки, впровадження та супроводу.
* **Інтеграція на рівні даних** — мається на увазі, коли декілька за стосунків можуть звертатися до однієї бази даних або в декілька баз даних, зв‘язаних реплікаціями. **Переваги**: низька вартість інтеграції, а при використанні однієї СУБД це стає дуже привабливим рішенням. **Недоліки**: якщо база даних не екранована збереженими процедурами і не має відповідних обмежень цілісності (у вигляді вказівок каскадних операцій и триггерів), то різні застосунки можуть приводити дані в суперечливі стани. Якщо БД екранована і цілісність забезпечується, то і в цьому випадку, в паралельно працюючих з однією БД застосунках, будуть дублюючи частини коду, які виконують однакові або схожі операції. Окрім того, при змінах структури БД потрібно буде окремо переписувати код за стосунків, що з нею працюють.
* **Інтеграція на рівні сервісів** — це інтеграція, заснована на фіксації інтерфейсів та форматів даних з двох сторін а також така, що дозволяє налагоджувати швидке відпрацювання міжкорпоративної бізнес-логіки. **Недоліки**: присутня фіксація, а якщо структури або процеси змінюються, то створюються проблеми та вузько спеціалізовані, окремі рішення.
* **Інтеграція на рівні користувача** — це крайній випадок, не автоматизована інтеграція, коли користувачі переміщують дані між системами через копіпаст, файли, пошту та інші засоби, що не мають регламенту. Такі методи часто застосовуються у той період, коли програмні системи не готові, а розвиток компанії не дозволяє чекати, але таки методи не є сприйнятливими.
* **Динамічна інтерпретація метаінформації** — це випадок, коли структура компонентів ІС з **динамічною інтерпретацією метамоделей** замикається знизу на самописуваєме сховище (обмежене рівнем абстракції метамоделі), а зверху замикається на користувача, який може модифікувати не тільки дані, але й метадані, переналагоджуючи ІС під постійно змінювані вимоги задачі, що вирішується. В цьому полягає гнучкість підходу та спрощення модифікації систем, зниження необхідного рівня кваліфікації користувача при внесенні змін в структуру та функції. Місце же архітектора програмних рішень та програмістів в створенні програмного забезпечення з підвищеним рівнем абстракції, тобто в реалізації метамоделі (що є більш складною інженерною задачею). Цей підхід виправдовує себе завдяки підвищенню повторного використання коду з високою абстракцією, автоматизації багатьох задач зв‘язування компонентів, зниженню впливу людського фактору при модифікації та інтеграції систем, а також спрощення інтеграції між компонентами програмних систем.

Для вирішення завдання забезпечення інтеграції програмних компонентів розподіленої системи необхідно:

* обґрунтувати засоби інтеграції;
* розробити алгоритм інтеграції;
* оцінити якість інтеграції.

**Обґрунтування засобів інтеграції**

Для інтеграції автоматизованих систем найбільше поширення отримали засоби, що оперують з універсальним форматами опису даних та їхнім передаванням. До універсальних засобів опису даних належить XML – стандарт побудови мов розмітки ієрархічно структурованих даних, який підтримується більшістю компаній-виробників засобів розробки програмного забезпечення – Microsoft, Sun (Oracle), IBM та ін.

Серед протоколів транспортного рівня одним з найуніверсальніших є http, який підтримується всіма основними операційними системами та розробниками. У якості програмної реалізації вибраних форматів даних і транспорту запропоновано веб-сервіс, який надає набір методів, що викликаються за допомогою SOAP-протоколу через HTTP. Модель взаємодії підсистем регламентованого обміну інформацією показано на рис. 1.

Запропонована модель передбачає виклик методів веб-сервісу за SOAP-протоколом та передавання даних за протоколом SOAP[[4]](#footnote-4), http чи ftp.

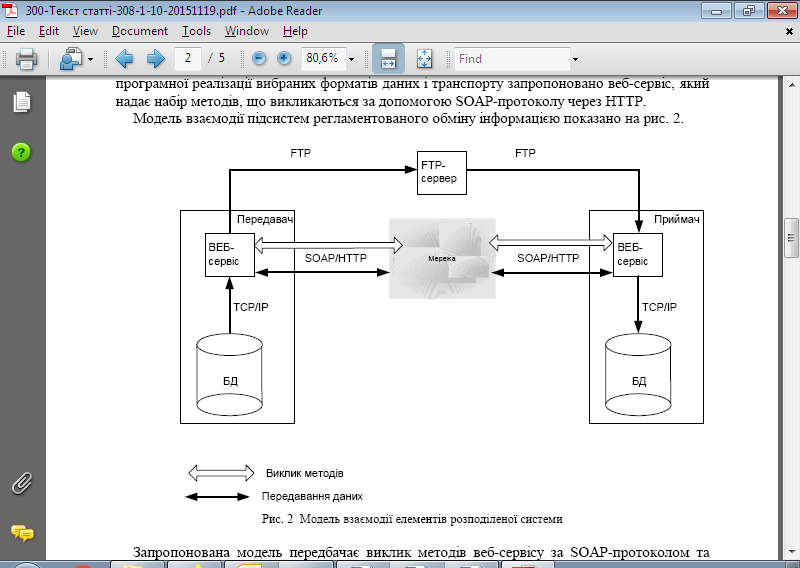


Рис. 1 Модель взаємодії елементів розподіленої системи

**Алгоритм взаємодії програмних компонентів розподіленої системи**

В процесі виконання задачі приймає участь три елементи: підсистема-передавач, підсистема-отримувач та підсистема координатор.

Алгоритм взаємодії програмних компонентів розподіленої інформаційної системи для задач складається з таких кроків:

1. Узгодження параметрів передавання даних: визначення формату даних,

можливість архівування та шифрування, вибір протоколу передавання.

2. Конвертування даних у потрібний формат підсистемою-передавачем.

3. Передавання даних.

4. Отримання даних підсистемою-отримувачем та їхня обробка.

5. Підтвердження успішної обробки даних.

6. Установлення параметрів наступного інформаційного обміну.

UML-діаграму послідовності взаємодії елементів розподіленої системи показано на рис. 2.

**Оцінка якості інтеграції**

Критерієм оцінки ефективності взаємодії елементів розподіленої системи є час виконання задачі *T* , який визначається за формулою

*T =*t *форм* +*t конв*1 +*t перед* +*t конв*2 +*t викор* ,

де *t форм* – час формування необхідної інформації; *t конв*1 – час конвертування інформації в потрібний вигляд системою-передавачем; *t перед* – час передавання даних; *t конв* 2 – час конвертування даних системою-отримувачем; *t викор* – час, необхідний для використання інформації.



Рис. 2 UML-діаграма послідовності взаємодії елементів розподіленої системи

**Покроковий та поетапний підхід до інтеграції.**

**Покрокова інтеграція** полягає в тому, що код розроблюється та тестується малими компонентами, які потім поступово інтегруються в єдине ціле. Протилежний підхід до інтеграції називається **поетапною інтеграцією**. Принцип цього підходу полягає в одночасній інтеграції декількох (нових та змінених) компонентів. Основний недолік поетапної організації полягає в тому, що в інтеграцію включено багато змінних, що ускладнює пошук помилок, оскільки помилка може знаходитися у будь-якому компоненті, у взаємодії нових компонентів з ядром системи і у взаємодії між новими компонентами.

Переваги покрокової інтеграції.

* ***Простота пошуку помилок***. При виникненні помилки зрозуміло, де її шукати: або безпосередньо в новому або зміненому компоненті, або в точці взаємодії з системою. При покроковій інтеграції помилки часто виявляються послідовно одна за одній, що спрощує їх пошук.
* ***Компоненти тестуються в повному обсязі***. Інтеграція компонентів здійснюється в процесі їх інтеграції і тестування. Завдяки цьому працездатність компонентів перевіряється частіше ніж у випадку проведення інтеграції в один захід.
* ***Компоненти швидше включаються в роботу***. Розробники швидше бачать результати своєї роботи, не чекаючи завершення розробки всієї системи.

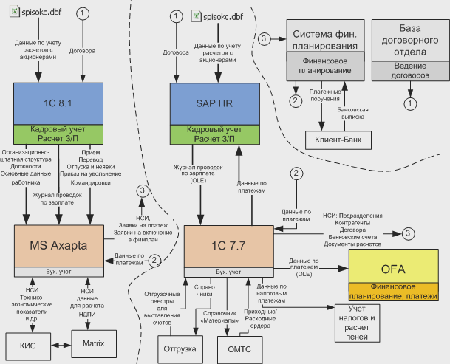
При ітераційній розробці інтеграція виконується хоча б один раз на кожній ітерації, відповідно, план виконання ітерації повинен містити перелік варіантів використання та класів, які повинні бути реалізовані на цій ітерації.

**Документальний опис процесу інтеграції**

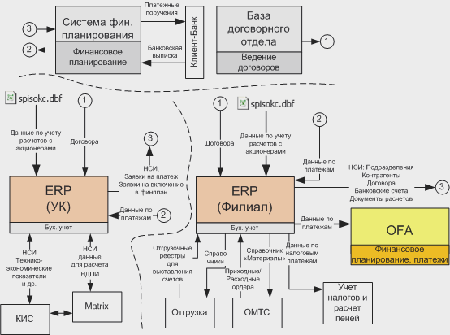
На етапі проектування формується концепція інтеграції ПЗ і оформлюється у вигляді окремого документу або включається як розділ в проектну документацію. В цей документ зазвичай має таку структуру.

1. Опис поточних інтеграційних процесів об‘єкту автоматизації.
2. Інформаційні системи, що інтегруються.
3. Схема потоків даних.
4. Періодичність обміну даними.
5. Опис інтерфейсів інтеграції інформаційних систем.
6. Організаційні заходи.
   1. ***Опис поточних інтеграційних процесів об‘єкту автоматизації.*** В цьому розділі документу надається опис існуючих потоків даних (процесів обміну даних між існуючими інформаційними системами) у вигляді схеми потоків даних.

Нижче наведено приклад схеми потоків даних взаємодії між системами бухгалтерського обліку, кадрового обліку та іншими інформаційними системами.

[](https://www.prj-exp.ru/images.php?img=IsIs_Data_Flows)

* 1. ***Інформаційні системи, що інтегруються***. В цьому розділі надається перелік інформаційних систем між якими планується організовувати періодичний обмін даними.
  2. ***Схема потоків даних.*** В цьому розділі надається схема потоків даних, яка описує майбутні потоки інтеграції (обміну даними) між інформаційними системами. Нижче наданий приклад схеми потоків даних при організації синхронізації в рамках впровадження проекту ERP[[5]](#footnote-5), коли інтегруються системи фінансового планування керуючої компанії і філіалу, а також система ведення договорів керуючої компанії і файли з даними керуючої компанії по обліку розрахунків з акціонерами для філіалу та керуючої компанії.

[](https://www.prj-exp.ru/images.php?img=ToBe_Data_Flows)

* 1. ***Періодичність обміну даними.*** Надається періодичність виконання обміну даних між інформаційними системами.

|  |  |  |  |
| --- | --- | --- | --- |
| Система джерела | Приймаюча система | Група даних | Періодичність обміну |
| ERP | MS Axapta | Дані нормативно-довідкової інформації | Щоденно |
| Дані для проводок | 1 раз на місяць. Після виконання розрахунків по заробітній платі до 15 числа |

І т.д.

* 1. ***Опис інтерфейсів інтеграції інформаційних систем***. В цьому розділі надається опис інтерфейсів (структура записів файлів, таблиць БД тощо) для кожного потоку і типу даних окремо.
  2. ***Організаційні заходи***. В цьому розділі надається перелік організаційних заходів, які потрібно виконати до початку розробки процесів інтеграції (обміну даними). Наприклад:

Для забезпечення реалізації інтеграційних процесів необхідно провести такі заходи:

1. Сформувати вимоги, проектні рішення по структурах вивантаження даних з …
2. Сформувати регламент створення нових даних та супроводження існуючих в нормативно-довідковій інформації.
3. Підготувати розпорядчі документи для організації внесення змін в механізм завантаження даних в …
4. …

**Найпоширеніші методи інтеграції.**

* **Інтеграція на рівні брокерів**. Перевагою цього методу є універсальність: зазвичай можна реалізувати додатковий програмний модуль , який звертається в інші системи різними способами. Наприклад, до однієї системи через базу даних, а до іншої за допомогою RPC (англ. Remote Procedure Call — віддалений виклик процедур). Недоліком цього підходу є трудомісткість і складність реалізації та як наслідок висока вартість розробки, впровадження та підтримки.
* **Інтеграція на рівні інтерфейсів (фізичних, програмних, користувацьких).** При цьому виді інтеграції застосунки зв‘язуються «кожний з кожним». При великій кількості за стосунків, а також для успадкованих (Legacy Software) та вбудованих (Embedded System) систем цей підхід ускладнює взаємодію, є малоефективним, оскільки не забезпечує якісно нових запитів до об‘єднаних даних. Підхід ефективний для невеликої кількості програмних застосунків. Для вирішення такого типу інтеграції використовуються стандартні застосунки з відкритими програмними інтерфейсами (англ. Open Application Programming Interface, OAPI – відкрити програмний інтерфейс прикладного програмування).
* **Інтеграція на функціонально-прикладному та організаційному рівнях.** Цей тип інтеграції побудований ра об‘єднанні декількох однотипних або схожих функцій в макрофункції, в яких перерозподіляються ресурси, потоки даних, керування та механізми виконання. Це спричиняє реорганізацію інформаційних структур, бізнес-процесів і, відповідно, перебудову схем їх інформаційного та документаційного забезпечення. Перевагою цього виду інтеграції є прозорість та керованість процесів, процеси стають менш витратними, скорочується кількість обслуговуючого персоналу, скорочується кількість помилок. Недоліком є значна трансформація або комплексний реінженірінг всієї мережі процесів, що може спричинити певні ризики. Доцільно проводити таку інтеграцію, коли організація готується до впровадження корпоративної інформаційної системи на платформі популярного рішення, що, в свою чергу, вимагає уніфікації та приведення бізнес-процесів до визначеного стандарту.
* **Інтеграція на рівні корпоративних програмних за стосунків.** Цей вид інтеграції передбачає спільне використання виконавчого коду, а не тільки внутрішніх даних за стосунків, що інтегруються. Програми поділяються на компоненти, які потім інтегруються за допомогою стандартизованих програмних інтерфейсів (API) та спеціалізованого програмного забезпечення для зв‘язування. Такій підхід дозволяє створити з цих компонентів універсальну програмну платформу (ядро), яка може бути використана всіма застосунками. Кожний застосунок буде мати лише один інтерфейс для взаємодії з цим ядром, що спрощує задачу інтеграції. Створену у такий спосіб систему легше адмініструвати, підтримувати та масштабувати. Для оцінювання можливості інтеграції застосунків, які будуть зв‘язуватися в рамках проекту, потрібно проведення аналізу внутрішньої архітектури застосунків.
* **Інтеграція за допомогою WEB-сервісів.** Цей вид базується на наданні стандартного для WEB-служб інтерфейсу доступу до застосунків та їх даним, наприклад, за допомогою стандартного протоколу доступу до об‘єктів SOAP (англ. Simple Object Access Protocol). За допомогою SOAP браузер користувача може одночасно порівнювати дані на декількох обраних веб-сайтах та надати порівняльний звіт користувачу. Іншим прикладом може бути одночасний доступ співробітників географічно роззосередженого підприємства через відповідні WEB-сервіси (портальне рішення), які працюють скрізь, де є Інтернет. Проблематичним є використання цього рішення в системах, де в процесі взаємодії потрібно проводити узгоджені зміни в транзакції в декількох підсистемах, тобто провести логічну транзакцію, яка зачіплює ці підсистеми, оскільки при збійній ситуації потрібно проводити відкат даних, механізм якого не передбачений. Існує попередній стандарт WS-Transactions, який має вирішити цю проблему, але на цей час ще відсутня його повноцінна реалізація.
* **Інтеграція на рівні даних.**. Цей вид інтеграції передбачає, що декілька програмних застосунків можуть звертатися до однієї БД або в декілька БД, зв‘язаних реплікаціями. Розглядалося вище.
* **Інтеграція на рівні сервісів.** Це інтеграція, заснована на фіксації інтерфейсів та форматів даних з двох сторін а також така, що дозволяє налагоджувати швидке відпрацювання міжкорпоративної бізнес-логіки. Розглядалося вище.

***Контрольні запитання*.**

1. В чому полягає системна інтеграція?
2. Назвіть складові системної інтеграції?
3. Наведіть приклади системної інтеграції?
4. Яким чином засоби рефакторингу впливають на ризики програмного забезпечення?
5. Які фактори впливають на інтеграцію?
6. Назвіть засоби вирішення проблем, що виникають при інтеграції
7. Який підхід до інтеграції, покроковий чи поетапний, з вашої точки зору є найкращим і чому?
8. Як документується опис процесу інтеграції ?
9. Назвіть найпоширеніші методи інтеграції.

***Література.***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник. URL: <http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf> (дата звернення: 08.01.2021).
2. Соммервил И. Инженерия программного обеспечения / И.Соммервил.–М. : Издательский дом «Вильямс», 2002. –623 с.
3. Динамическая интерпретация метамоделей – URL: <https://habr.com/ru/post/154891/> (дата звернення: 08.01.2021).
4. Концепция интеграции информационных бизнес-систем – URL: <https://www.prj-exp.ru/integration/concept_integration.php> (дата звернення: 08.01.2021).
5. Думченков И. А. Обзор методов интеграции информационных систем, их преимуществ и недостатков // Молодой ученый. — 2018. — №23. — С. 176-177. — URL https://moluch.ru/archive/209/51296/ (дата звернення: 30.12.2019).

***Для самостійного вивчення***: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

1. **SCADA** ([аббр.](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0) от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ***S****upervisory****C****ontrol****A****nd****D****ata****A****cquisition* — *диспетчерское управление и сбор данных*) — [программный пакет](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D1%8B%D1%85_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC" \o "Пакет прикладных программ), предназначенный для разработки или обеспечения работы в реальном времени систем сбора, обработки, отображения и архивирования информации об объекте мониторинга или управления. Высокомасштабируемая платформа SCADA, обеспечивает общекорпоративную стандартизацию. HMI - программное обеспечение для визуализации процессов построения [↑](#footnote-ref-1)
2. **Загальний сервіс пакетної радіопередачі** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *General Packet Radio Service*, **GPRS**) — [стандарт](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), який використовує не зайняту голосовим зв'язком смугу частот для передачі інформації. Використовується в мобільних пристроях для передачі [MMS](https://uk.wikipedia.org/wiki/MMS), [WAP](https://uk.wikipedia.org/wiki/WAP)-серфінгу та повноцінного з'єднання з [Інтернетом](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Розрізняють так звані класи GPRS — рівень підтримки стандарту конкретним приладом. Існують класи від першого до дванадцятого — чим вищий клас, тим більшу швидкість передачі даних може, теоретично, забезпечити телефон. [↑](#footnote-ref-2)
3. **Виклик віддалених процедур** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Remote procedure call,****RPC***) — [протокол](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB), що дозволяє програмі, запущеній на одному комп'ютері, звертатись до функцій (процедур) програми, що виконується на іншому комп'ютері, подібно до того, як програма звертається до власних локальних функцій. [↑](#footnote-ref-3)
4. **SOAP** (*Simple Object Access Protocol*) — протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML. Спочатку SOAP призначався, в основному, для реалізації віддаленого виклику процедур (RPC), зараз протокол використовується для обміну повідомленнями в форматі XML. SOAP можна використовувати з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTP та інші. Проте його взаємодія з кожним із цих протоколів має свої особливості, які потрібно відзначити окремо. Найчастіше SOAP використовується разом з HTTP. [↑](#footnote-ref-4)
5. **ERP** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Enterprise Resource Planning*, *планирование ресурсов предприятия*) — организационная стратегия интеграции [производства](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE" \o "Производство) и [операций](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F%D0%BC%D0%B8" \o "Управление операциями), [управления трудовыми ресурсами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D0%BE%D0%BC" \o "Управление персоналом), [финансового менеджмента](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BD%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BC%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%BC%D0%B5%D0%BD%D1%82" \o "Финансовый менеджмент) и [управления активами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%B0%D0%BC%D0%B8" \o "Управление активами), ориентированная на непрерывную балансировку и оптимизацию ресурсов предприятия посредством специализированного интегрированного [пакета прикладного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D1%8B%D1%85_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC), обеспечивающего общую [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81" \o "Бизнес-процесс) для всех сфер деятельности [↑](#footnote-ref-5)