**Лабораторна робота №5 Фізичне представлення моделей систем в UML: діаграмикомпонентів та розгортання**

**Мета:** **Навчитися будувати діаграми** **компонентів та розгортання**

.

**Завдання до лабораторної роботи**

1. Ознайомитись з теоретичною частиною та рекомендаціями до виконання роботи.
2. Розробити діаграму компонентів для обладнання Банкомату. Показати компоненти: Обладнання читання картки, Клавіатура Банкомата, Принтер Банкомата, Екран Банкомата, Обладнання видачі готівки.
3. Розробити діаграму розгортання для системи обслуговування банкоматів. Показати компоненти: 3 банкомати з примітками щодо розташування, закрита мережа, сервер.
4. До всіх діаграм додати нотатки, де записати своє прізвище та групу, а також номер ЛР.
5. Для відсилки викладачу збережіть проект з іменем Вашої ЛР та розширенням xmi.
6. Оформите звіт, в якому надайте скріншоти створеної діаграми та відповіді на запитання в кінці ЛР.
7. Файл зі звітом викласти на платформу коледжу до репозиторію викладача вашого предмету з іменем у форматі :

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвищеанглійською>**. Наприклад, **PI3103R**buts.doc.

1. Файл зі звітом та створену Umbrello UML Modeller модель (файл xmi) відішліть на електронну адресу викладача.

**Строк виконання цієї роботи ІПЗ-31 26.04.2020, ІПЗ-32 20.04.2020**

**ТЕОРЕТИЧНА ЧАСТИНА**

**Діаграми компонентів (Component diagram)**

На діаграмах компонентів показано компоненти програмного забезпечення (або технології компонентів або просто розділи системи, які чітко відрізняються один від одного), а також елементи, з яких вони складаються, такі як файли з початковими кодами, програмні бібліотеки або таблиці реляційних баз даних. Компоненти можуть мати інтерфейси (тобто абстрактні класи з операціями), які надають змогу створювати асоціації між компонентами.

В мові UML виділяють три види компонентів.

1. *компоненти розгортання*, які забезпечують безпосереднє виконання системою своїх функцій. Такими компонентами можуть бути бібліотеки с розширенням dll, які динамічно підключаються, Web-сторінки на мові розмітки гіпертексту з розширенням html та файли довідки з розширенням hlр;
2. *компоненти-робочі продукти*. Зазвичай це файли з вихідними текстами програм, наприклад, з розширеннями h або срр для мови C++;
3. *компоненти виконання*, які представляють модулі для виконання — файли розширенням ехе.

Діаграма компонентів служить частиною фізичного представлення моделі, відіграє важливу роль у процесі об’єктно-орієнтованого аналізу та проектування (ООАП) і є необхідною складовою для генерації програмного коду.

Графічне зображення стереотипів і їх коротка характеристика приводяться в наступній таблиці (табл. 1). При цьому кожному з компонентів, як правило, відповідає окремий файл вихідної зборки програмного додатка.

Таблиця 1.Графічне зображення стереотипів компонентів і їх характеристика

| Графічне зображення й ім'я за замовчуванням | Назва стереотипу | Характеристика стереотипу компонента | |
| --- | --- | --- | --- |
|  | Subprogram Specification | Специфікація підпрограми. Містить опис змінних, процедур і функцій і не містить визначень класів |
|  | Subprogram Body | Тіло підпрограми. Містить реалізацію процедур і функцій, що не належать до якихось класів, при цьому не містить визначень класів або реалізацій операцій інших класів |
|  | Main Program | Головна програма. Реалізує базову логіку роботи програмного додатка й містить посилання на інші компоненти моделі |
|  | Package Specification | Специфікація пакета. Містить визначення класу, його атрибутів і операцій. У мові програмування З++ специфікації пакета відповідає окремий файл із розширенням «h» | | |
|  | Package Body | Тіло пакета. Містить код реалізації операцій класу. У мові програмування З++ специфікації пакета відповідає окремий файл із розширенням «cpp» | | |
|  | Task Specification | Специфікація завдання. Може містити визначення класу, його атрибутів і операцій, які передбачається використовувати в незалежному потоці керування | | |
|  | Task Body | Тіло завдання. Може містити реалізацію операцій класу, які мають незалежний потік керування. | | |
|  | Generic Subprogram | Типова підпрограма. Містить опис змінних, процедур і функцій, які можуть бути використані в декількох програмних додатках. При цьому типова підпрограма не містить визначень класів | | |
|  | Generic Package | Типовий пакет. Містить визначення класу, його атрибутів і операцій, яке може бути використане в декількох програмних додатках | | |
|  | Database | База даних. Містить визначення одного або декількох класів, їх атрибутів і, можливо, операцій. При цьому відповідні класи можуть бути реалізовано у формі однієї або декількох таблиць бази даних | | |

Використання розглянутих стереотипів суттєво збільшують наочність графічного представлення діаграми компонентів і дозволяють архітекторові уточнити характер реалізації моделі програмістом обраною мовою програмування.

***Інтерфейси***

Наступний елемент діаграми компонентів - інтерфейси. Вони вже розглядалися раніш, зупинимося лише на характерних для представлення на діаграмах компонентів. В загальному випадку інтерфейс графічно зображується колом, яке з’єднується з компонентом відрізком лінії без стрілок (рис. 2, а). При цьому, ім’я інтерфейсу, обов’язково повинно починатися з великої літери "I", записується поряд з колом. Семантично лінія означає реалізацію інтерфейсу, а наявність інтерфейсів у компонента означає, що даний компонент реалізує відповідний набір інтерфейсів.

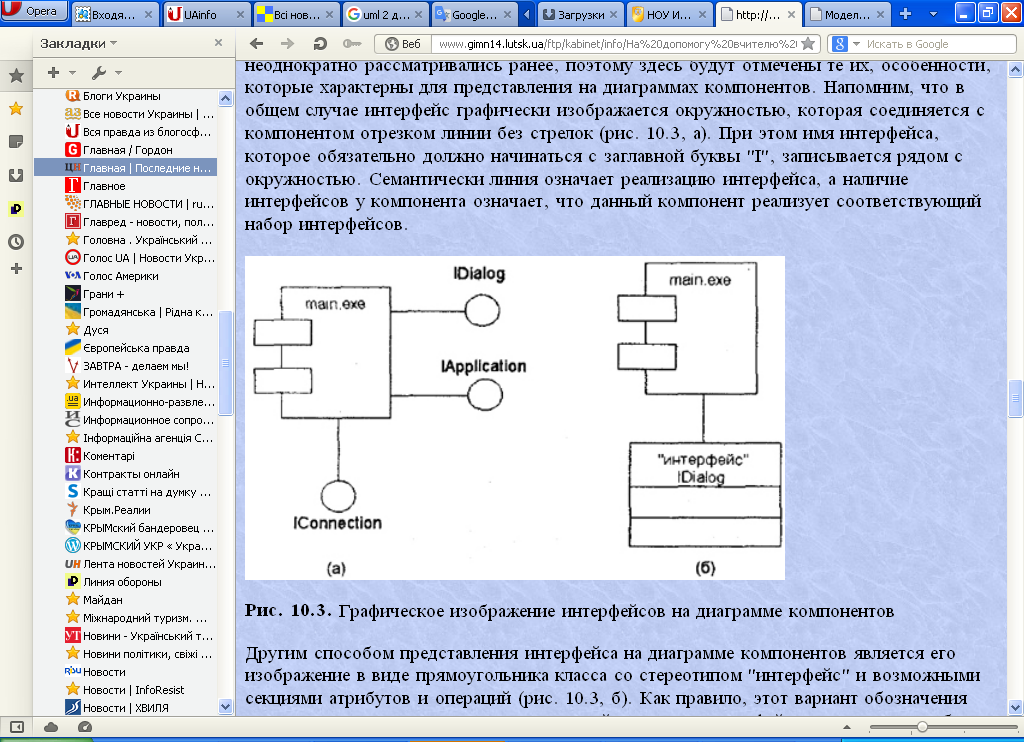


Рис. 2. Графічне зображення інтерфейсів на діаграмі компонентів

Іншим способом представлення інтерфейсу на діаграмі компонентів є його зображення у вигляді прямокутника класу зі стереотипом "інтерфейс" та можливими секціями атрибутів та операцій (рис. 10.3, б). Зазвичай, цей варіант позначення використовують для представлення внутрішньої структури інтерфейсу, яка може бути важливою для реалізації.

При розробці програмних систем інтерфейси забезпечують не тільки сумісність різних версій, але й можливість вносити суттєві зміни в одні частини програми, не змінюючи інші її частини. Тобто, призначення інтерфейсів суттєво ширше, ніж специфікація взаємодії з користувачами системи (акторами).

***Залежності***

Залежність не є асоціацією, а слугує для представлення тільки факту наявності такого зв’язку, коли зміна одного елементу моделі впливає або приводить до змін іншого елементу моделі. Відношення залежності на діаграмі компонентів зображується пунктирною лінією зі стрілкою, направленою від клієнта (залежного елементу) до джерела (незалежного елемента).

Залежності можуть відображати зв’язки модулів програми на етапі компіляції та генерації об’єктного коду. В іншому випадку залежність може відображати наявність в незалежному компоненті описів класів, які використовуються в залежному компоненті для створення відповідних об’єктів. Стосовно діаграми компонентів, тут залежності можуть зв’язувати компонент та імпортовані цим компонентом інтерфейси, а також різні види компонентів між собою.

В першому випадку накреслюють стрілку від компонента-клієнта до імпортованого інтерфейсу (рис. 3). Наявність такої стрілки означає, що компонент не реалізує відповідний інтерфейс, а використовує його в процесі свого виконання. Причому на цій же діаграмі може бути присутнім також інший компонент, який реалізує цей інтерфейс. Наприклад, зображений нижче фрагмент діаграми компонентів представляє інформацію про те, що компонент з ім’ям "main.exe" залежить від імпортованого інтерфейсу IDialog, який в свою чергу, реалізується компонентом с іменем "image.java". Для другого компонента цей же інтерфейс є експортованим.

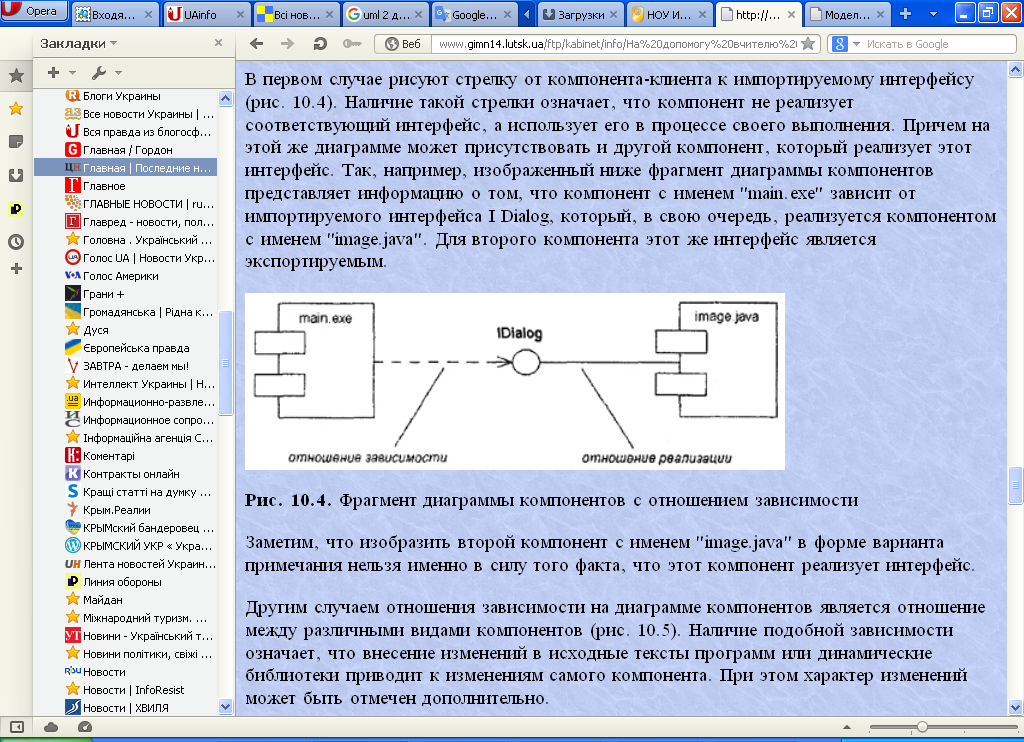


Рис. 3. Фрагмент діаграми компонентів з відношенням залежності

В середині символу компонента можуть зображаться інші елементи графічної нотації, такі як класи (компонент рівня типу) або об’єкти (компонент рівня екземпляру). В цьому випадку символ компонента зображується так, щоб умістити ці додаткові символи. Наприклад, зображений нижче компонент (рис. 4) є екземпляром та реалізує три окремих об’єкти.

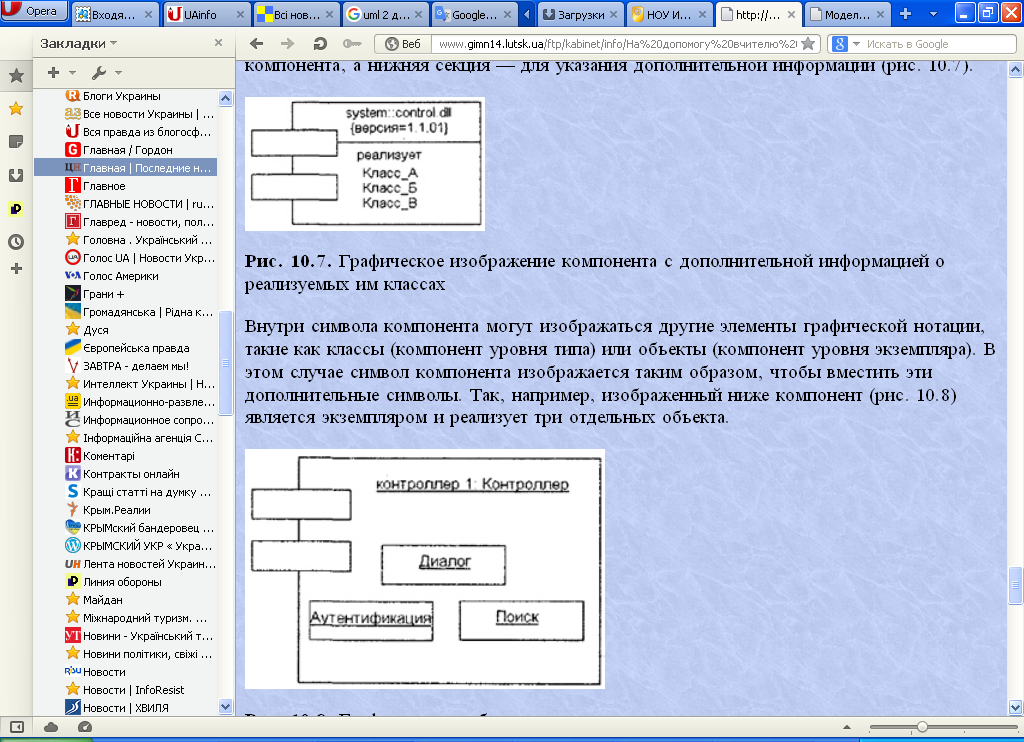


Рис. 4. Графічне зображення компонента рівня екземпляру, який реалізує окремі об’єкти

Об’єкти, які знаходяться в окремому компоненті-екземплярі, зображуються вкладеними в символ цього компонента. Така вкладеність означає, що виконання компонента викликає виконання відповідних об’єктів. Іншими словами, існування компонента протягом часу виконання програми забезпечує існування, а можлива, і доступ всіх вкладених в нього об’єктів. Доступ до цих об’єктів може бути додатково специфікований за допомогою квантифікаторів[[1]](#footnote-1) видимості, аналогічно видимості пакетів. Змістовний смисл видимості може різнитися для різних видів пакетів.

**Діаграма розгортання (deployment diagram)**

Фізичне представлення програмної системи не може бути повним при відсутності інформації щодо платформи та обчислювальних засобів її реалізації. Якщо розроблюється проста програма, яка може виконуватися локально на комп’ютері користувача без використання периферійних пристроїв та ресурсів, то потреби в розробці додаткових діаграм немає. Але при розробці корпоративних застосунків ситуація представляється інакше.

По-перше, складні програмні системи можуть реалізовуватися в мережевому варіанті на різних обчислювальних платформах та технологіях доступу до розподілених баз даних. Наявність локальної корпоративної мережі потребує рішення цілого комплексу додаткових задач по раціональному розміщенню компонентів по вузлах цієї мережі, що визначає загальну продуктивність програмної системи.

Во-друге, інтеграція програмної системи з Інтернетом визначає необхідність рішення додаткових питань при проектуванні системи, таких як забезпечення безпеки, криптозахищенності та стійкого доступу до інформації для корпоративних клієнтів. Ці аспекти здебільшого залежать від реалізації проекту в формі фізично існуючих вузлів системи, таких як сервери, робочі станції, брандмауери, канали зв’язку та сховища даних.

Окрім того, технології доступу та маніпулювання даними в рамках загальної схеми "клієнт-сервер" також вимагають розміщення великих баз даних на різних сегментах корпоративної мережі, їх резервного копіювання, архівування, кешування для забезпечення необхідної продуктивності системи в цілому. Ці аспекти також потребують візуального представлення з метою специфікації програмних та технологічних особливостей реалізації розподілених архітектур.

Першою з діаграм фізичного представлення є діаграма компонентів. Другою формою фізичного представлення програмної системи є діаграма розгортання (синонім — діаграма розміщення). Вона застосовується для представлення загальної конфігурації та топології розподіленої програмної системи і містить розподілення компонентів по окремих вузлах системи. Також діаграма розгортання показує наявність фізичних з’єднань — маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи.

Діаграма розгортання призначена для візуалізації елементів та компонентів програми, існуючих лише на етапі її виконання (runtime). При цьому представляються тільки компоненти-екземпляри програми, які є виконуваними файлами або динамічними бібліотеками. Ті компоненти, які не використовуються на етапі виконання, на діаграмі розгортання не показуються. Наприклад, компоненти з вихідними текстами програм можуть бути присутніми тільки на діаграмі компонентів. На діаграмі розгортання вони не вказуються.

Діаграма розгортання містить графічні зображення процесорів, пристроїв, процесів та зв’язків між ними. На відміну від діаграм логічного представлення, діаграма розгортання є єдиною для системи в цілому, оскільки повинна повністю відображати особливості її реалізації. Ця діаграма, по суті, завершує процес ООАП для конкретної програмної системи та розробка цієї діаграми зазвичай є останнім етапом специфікації моделі.

Цілі розробки діаграми розгортання:

* визначити розподілення компонентів системи по її фізичним вузлам;
* показати фізичні зв’язки між всіма вузлами реалізації системи на етапі її виконання;
* виявити вузькі місця системи та реконфігурувати її топологію для досягнення потрібної продуктивності.

Для забезпечення цих вимог діаграма розгортання розробляється спільно системними аналітиками, мережевими інженерами та системотехніками.

Розглянемо окремі елементи, з яких складаються діаграми розгортання.

***Вузол***

***Вузол*** *(node)* є деяким фізично існуючим елементом системи, що має деякий обчислювальний ресурс. В якості обчислювального ресурсу вузла може розглядатися наявність деякого об’єму електронної або магнітооптичної пам’яті та/або процесора. В останніх версіях мови UML поняття вузла розширено та може включати в себе не тільки обчислювальні пристрої (процесори), але й інші механічні або електронні пристрої, такі як датчики, принтери, модеми, цифрові камери, сканери та маніпулятори.

Зауважимо, що можливість включення людей (персоналу) в поняття вузла дозволяє створювати засобами мови UML моделі самих різних систем, включно з бізнес-процесами та технічними комплексами. Реалізація бізнес-логіки підприємства вимагає розглядати в якості вузлів системи організаційні підрозділи, які складаються з персоналу. Автоматизація управлення технічними комплексами також вимагає розгляду в якості самостійного елемента людину-оператора, який може приймати рішення в нештатних ситуаціях та нести відповідальність за можливі наслідки цих рішень.

Графічно на діаграмі розгортання вузол зображується у формі тривимірного кубу (строго кажучи, псевдотрьохвимірного прямокутного паралелепіпеду). Вузол має власне ім’я, яке вказується в середині цього графічного символу. Самі вузли можуть представлятися як в якості типів (рис. 5, а), так і в якості екземплярів (рис. 5, б).

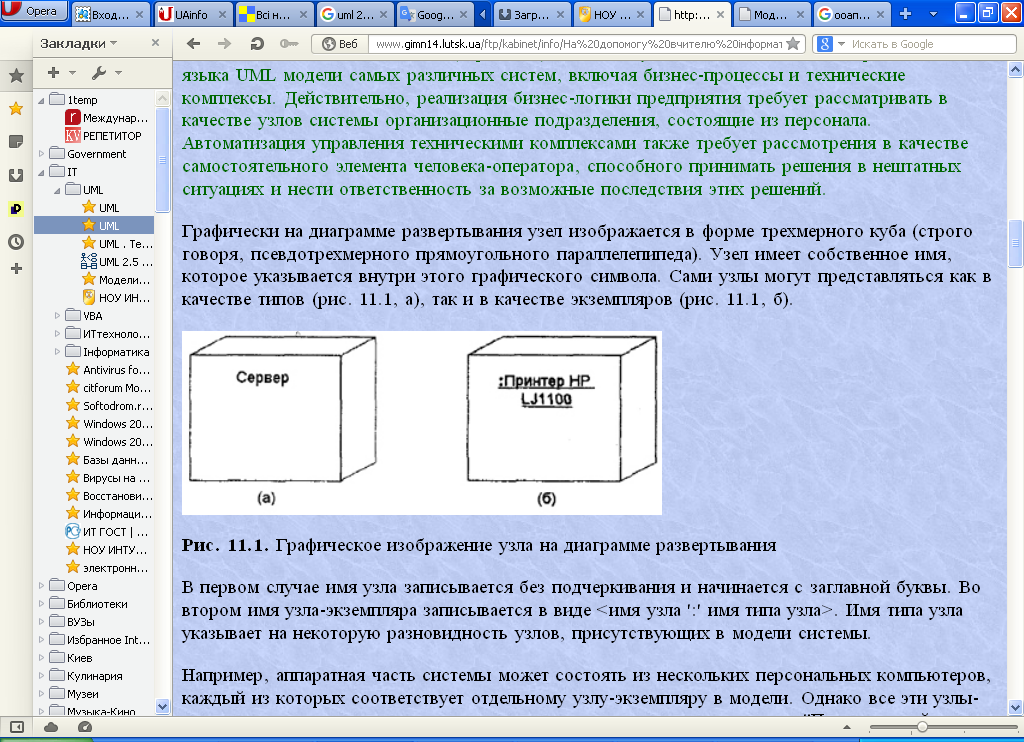


Рис. 5. Графічне зображення вузла на діаграмі розгортання

В першому випадку ім’я вузла записується без підкреслення та починається з великої літери. В другому - ім’я вузла-екземпляру записується у вигляді

*<ім’я вузла ':' ім’я типу вузла>*

Ім’я типу вузла вказує на деякий різновид вузлів, присутніх в моделі системи.

Наприклад, апаратна частина системи може складатися з декількох персональних комп’ютерів, кожен з яких відповідає окремому вузлу-екземпляру в моделі. Але всі ці вузли-екземпляри до одного типу вузлів, а саме; вузлу с іменем типу "Персональний комп’ютер". Так, на представленому вище рисунку (рис. 5, а) вузол з іменем "Сервер" відноситься до загального типу та не конкретизується. Другий вузол (рис. 5, б) є анонімним вузлом-екземпляром конкретної моделі принтера.

Як і на діаграмі компонентів, зображення вузлів можуть розширюватися, щоб включити деяку додаткову інформацію про специфікації вузла. Якщо додаткова інформація відноситься до імені вузла, то вона записується під цим іменем у формі поміченого значення (рис. 6).

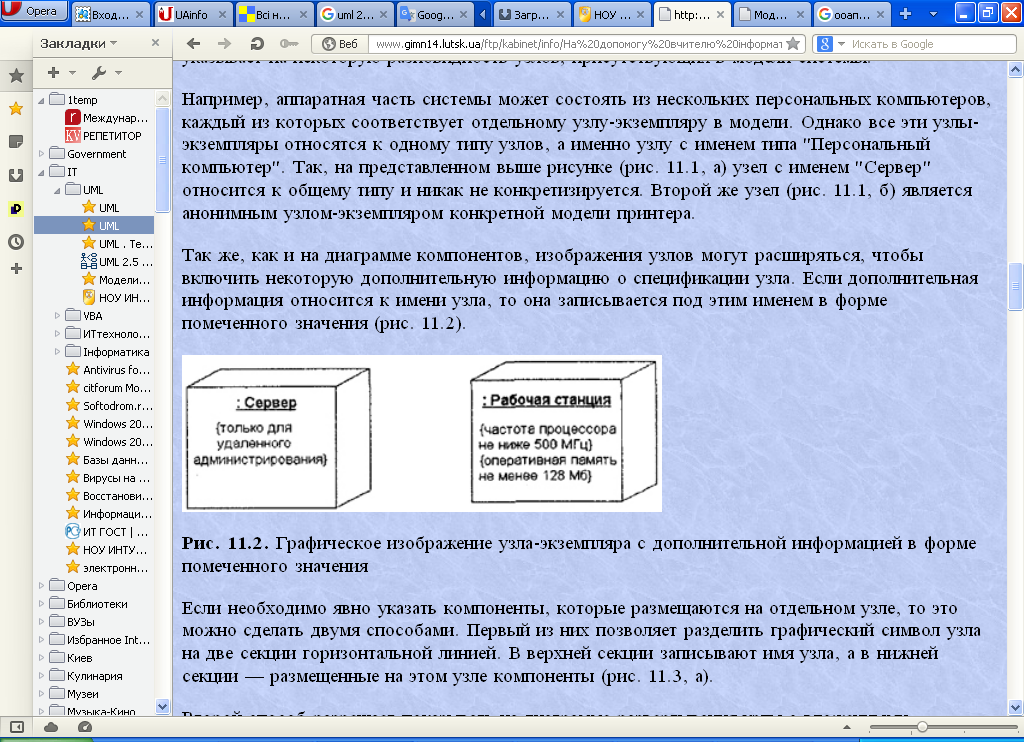


Рис. 6. Графічне зображення вузла-екземпляра з додатковою інформацією у формі поміченого значення

Якщо необхідно явно вказати компоненти, які розміщуються на окремому вузлі, то це можна зробити двома способами. Перший з них дозволяє розділити графічний символ вузла на дві секції горизонтальною лінією. В верхній секції записують ім’я вузла, а в нижній секції — розміщені на цьому вузлі компоненти (рис. 7, а).

Другий спосіб дозволяє показувати на діаграмі розгортання вузли з вкладеними зображеннями компонентів (рис. 7, б). В якості таких вкладених компонентів можуть бути тільки виконувані компоненти.

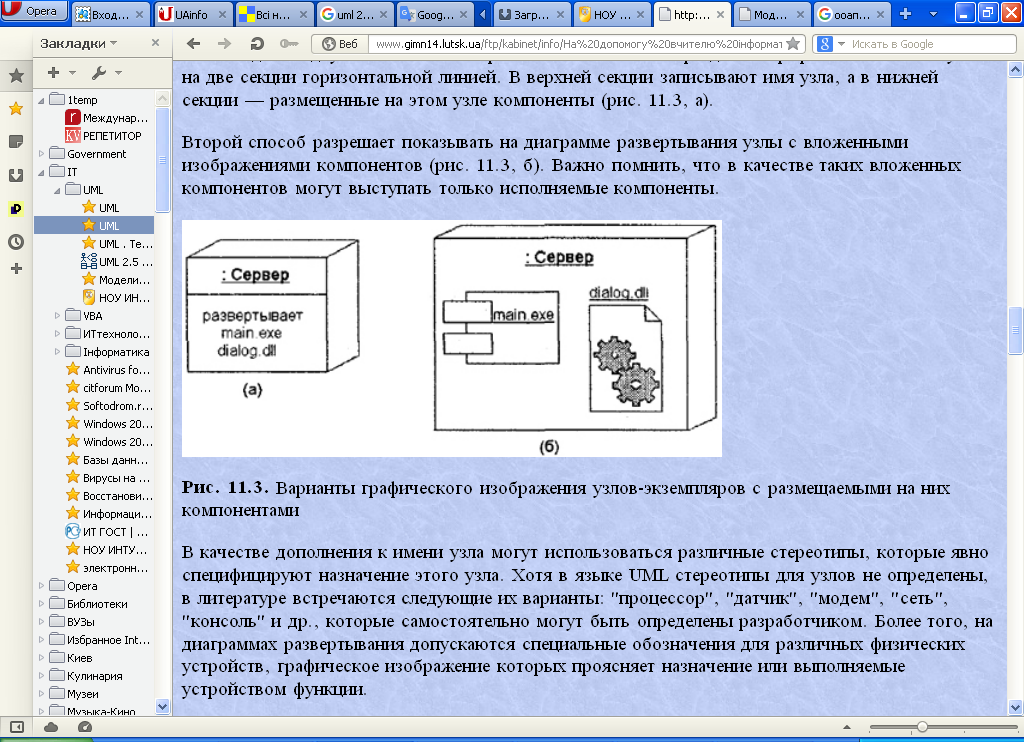


Рис. 7. Варіанти графічного зображення вузлів-екземплярів з розміщеними на них компонентами

В якості доповнення до імені вузла можуть використовуватися різні стереотипи, які явно специфікують призначення цього вузла. В мові UML стереотипи для вузлів не визначені, але в літературі зустрічаються такі їх варіанти: "процесор", "датчик", "модем", "мережа", "консоль" тощо, які самостійно можуть бути визначені розробником. Також на діаграмах розгортання допускаються спеціальні позначення для різних фізичних пристроїв, графічне зображення яких прояснює призначення або виконувані пристроєм функції. Додаткові графічні зображення для вузлів діаграми розгортання використовуються виключно для наочності їх представлення. Наприклад, процесор можна зобразити або у вигляді загального вузла (рис. 5), або у формі зображення зовнішнього виду комп’ютера. Відповідно, консоль може бути зображена в вигляді клавіатури.

***З’єднання***

Окрім безпосереднього зображення вузлів на діаграмі розгортання вказуються відношення між ними. В якості відношень виступають фізичні з’єднання між вузлами та залежності між вузлами та компонентами, зображення яких також можуть бути присутніми на діаграмах розгортання.

З’єднання є різновидом асоціації та зображуються відрізками ліній без стрілок. Наявність такої лінії вказує на необхідність організації фізичного каналу для обміну інформацією між відповідними вузлами. Характер з’єднання може бути додатково специфікований приміткою, поміченим значенням або обмеженням. Так, на представленому нижче фрагменті діаграми розгортання (рис. 8) за допомогою поміченого значення явно визначені вимоги до швидкості передачі даних в локальній мережі, а у формі приміток надані рекомендації щодо технології фізичної реалізації з’єднань.

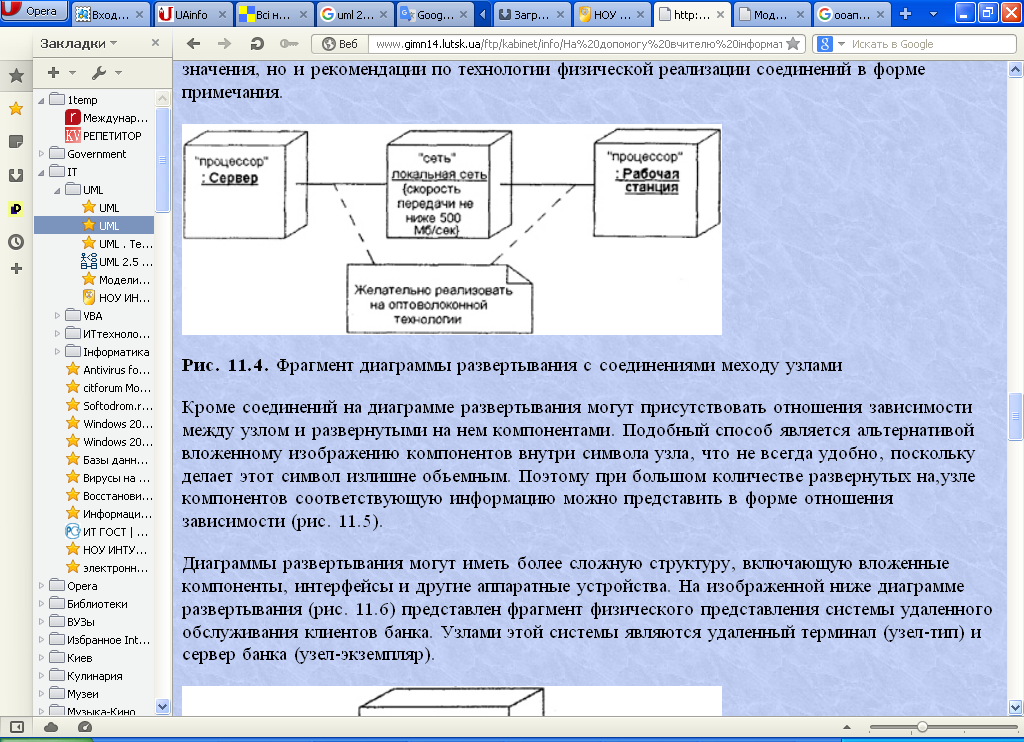


Рис. 8. Фрагмент діаграми розгортання зі з’єднаннями між вузлами

Окрім з’єднань на діаграмі розгортання можуть бути присутніми відношення залежності між вузлом та розгорнутими на ньому компонентами. Такий спосіб є альтернативою вкладеному зображенню компонентів в середині символу вузла, що не завжди зручно, оскільки робить цей символ занадто об’ємним. Саме тому при великій кількості розгорнутих на вузлі компонентів відповідну інформацію можна представити у формі відношення залежності (рис. 9).

Діаграми розгортання можуть мати більш складну структуру, яка включає вкладені компоненти, інтерфейси та інші апаратні пристрої. На зображеній нижче діаграмі розгортання (рис. 10) надано фрагмент фізичного представлення системи віддаленого обслуговування клієнтів банку. Вузлами цієї системи є віддалений термінал (вузол-тип) та сервер банка (вузол-екземпляр).

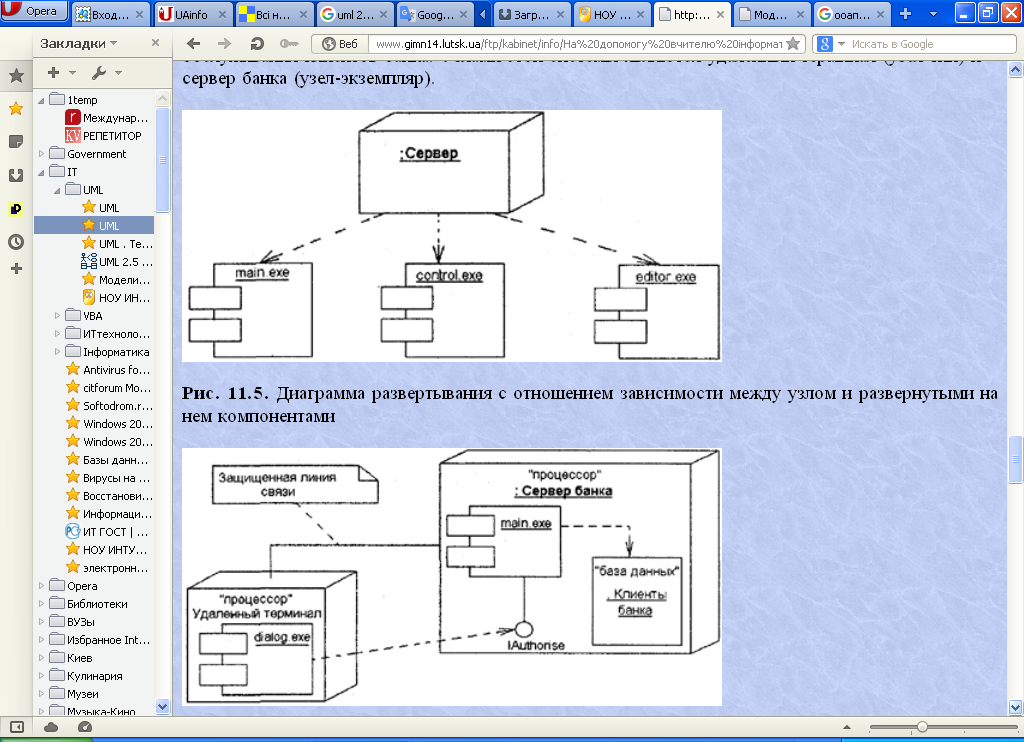


Рис. 9. Діаграма розгортання з відношенням залежності між вузлом та розгорнутими на ньому компонентами

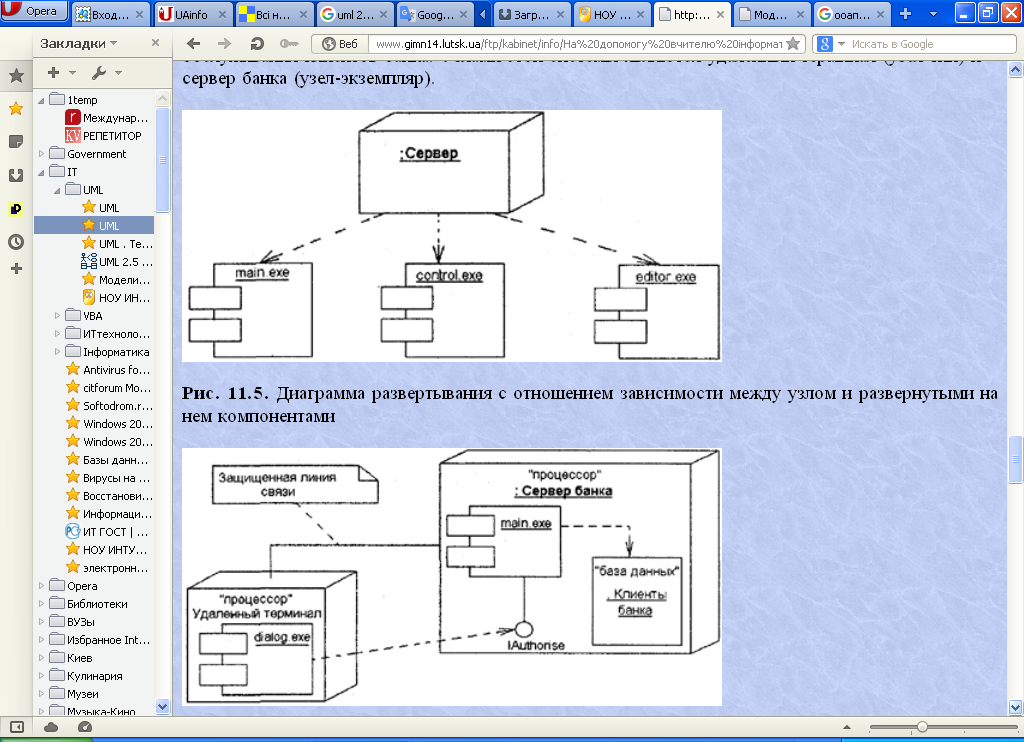


Рис. 10. Діаграма розгортання для системи віддаленого обслуговування клієнтів банку

На цій діаграмі розгортання вказана залежність компонента реалізації діалогу "dialog.exe" на віддаленому терміналі від інтерфейсу ІAuthorise, реалізованого компонентом "main.exe", який, в свою чергу, розгорнутий на анонімному вузлі-екземплярі "Сервер банка". Останній залежить від компонента бази даних "Клієнти банку", який розгорнутий на цьому ж вузлі.

Примітка вказує на необхідність використання захищеної лінії зв’язку для обміну даними в цій системі. Другий варіант запису цієї інформації полягає в доповненні діаграми вузлом зі стереотипом "закрита мережа".

Розробка вбудованих систем передбачає не тільки створення програмного коду, але й узгодження між собою всіх апаратних засобів та механічних пристроїв. В якості прикладу розглянемо фрагмент моделі керування віддаленим механічним засобом типу транспортної платформи. Така платформа призначена для переміщення в агресивних середовищах, де присутність людини неможлива з фізичних причин.

Транспортна платформа оснащується власним мікропроцесором, цифровою відеокамерою, датчиками температури та місця розташування, а також керуючими приводами для змінення напрямку та швидкості переміщення платформи. Керуюча та телеметрична інформація від платформи по радіолінії передається в центр управління, оснащений керуючим комп’ютером, маніпуляторами управління та великим інформаційним табло.

На мікропроцесорі платформи розгорнуті програмні компоненти для реалізації найпростіших керуючих впливів на приводи, що дозволяє дискретно змінювати напрямок та швидкість переміщення платформи. На комп’ютері центру управління розгорнуті програмні компоненти аналізу телеметричної інформації, яка характеризує стан окремих пристроїв' платформи, а також реалізовані алгоритми керування переміщенням платформи в цілому.

Варіант фізичного представлення цієї транспортної системи показано на діаграмі розгортання на рис. 11.

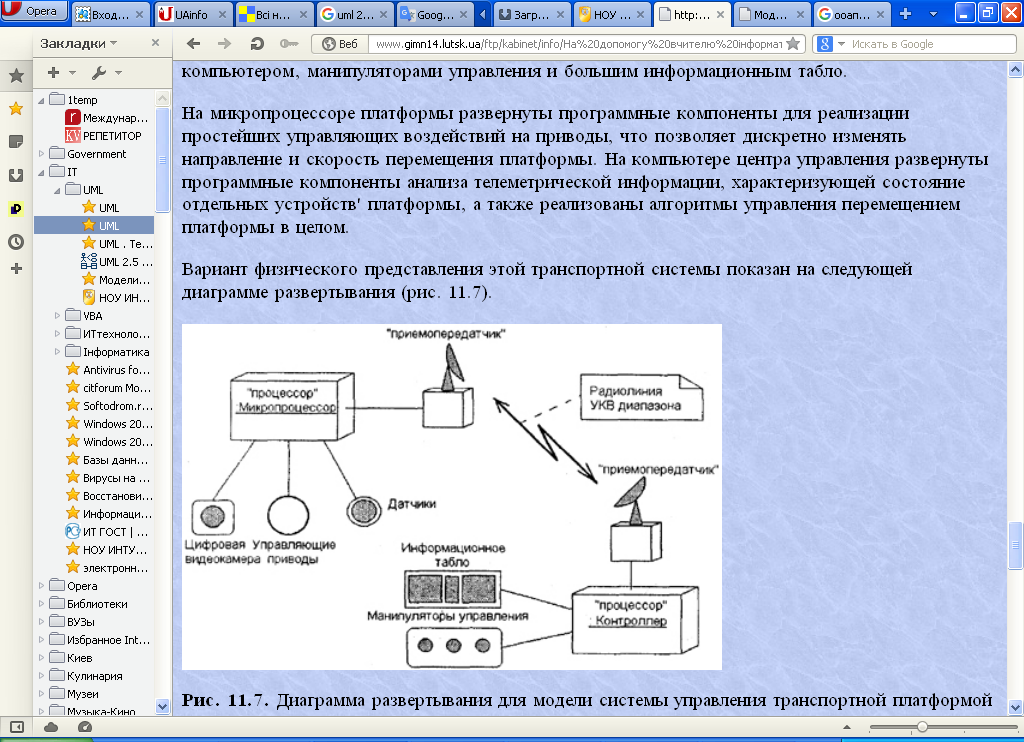


Рис. 11. Діаграма розгортання для моделі системи управління транспортною платформою

Дана діаграма містить лише загальну інформацію щодо розгортання системи та в подальшому може бути деталізована при розробці безпосередньо програмних компонентів управління. Як видно з рис.11, при розробці цієї діаграми розгортання використано додатковий стереотип "приемопередатчик", який відсутній в описі мови UML, та спеціальні зображення для окремих апаратних та механічних пристроїв.

**Контрольні запитання**.

1. На якому етапі ЖЦ ПЗ використовується діаграма компонентів? Відповідь поясніть.
2. На якому етапі ЖЦ ПЗ використовується діаграма розгортання? Відповідь поясніть.

1. Квантифікація - це введення кількісних характеристик для оцінки складних якісно обумовлених понять. [↑](#footnote-ref-1)