МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ ТА ІНФОРМАТИЗАЦІЇ
(назва інституту (факультету))

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
(повна назва кафедри)

ПЛАН КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Моделювання та проектування програмного забезпечення»

за спеціальністю 121 «Інженерія програмного забезпечення»
(шифр та повна назва напряму (спеціальності))

Спеціалізації_____

Укладач(і): старший викладач Гаманюк І.М.
(науковий ступінь, вчене звання, П.І.Б. викладача)

Конспект лекцій розглянуто та схвалено
на засіданні кафедри інженерії програмного забезпечення
(повна назва кафедри).

Протокол № ____ від «___»_____20__р.

Завідувач кафедри В.В.ОНИЩЕНКО

Лекція № 11

Тема лекції:

Діаграми розгортання. Артефакти. Діаграми артефактів.

План лекції

Вступ.
1. Діаграми розгортання.
2. Вузол діаграми розгортання.
3. Артифакти.
4. Діаграми артифактів.

Заключна частина.

Література

1. Г. Буч, Д. Рамбо, І. Якобсон Язык UML. Руководство пользователя 2-е издание / Пер. с агнл. – ДМК издательство, 2006 – 496 с.
2. К. Ларман. Применение UML 2.0 и шаблонов проектирования. Практическое руководство 3-е издание / Пер. с агнл. – Издательский дом “Вильямс”, 2013. – 736 с.
3. Р. Мартин, М. Мартин. Принципы, паттерны и методики гибкой разработки на языке C# / Пер. с агнл. – Издательство “Символ-Плюс”, 2014. – 768 с.
4. Обзор проектирования архитектуры // Режим доступу: https://msdn.microsoft.com/ru-ru/hh144976.aspx
5. Проектирование программного обеспечения // Режим доступу: https://ru.wikipedia.org/wiki/

Вступ

Діаграма розгортання – важлива діаграма для ознайомлення з масштабами системи та обладнанням системи, забезпечення безпеки системи.

Діаграма артифактів – важлива діаграма для розуміння, що зроблено і де воно знаходиться.

Артефакты живут в материальном мире битов и потому являются важнейшими строительными блоками при моделировании физических аспектов системы. Артефакт – это физическая и заменяемая часть системы.

Артефакты используются для моделирования таких физических сущностей, которые могут располагаться на узле, – например, исполняемых программ, библиотек, таблиц, файлов и документов.

Обычно артефакт представляет собой физическую группировку таких логических элементов, как классы, интерфейсы и кооперации.

Многие операционные системы и языки программирования непосредственно поддерживают концепцию артефакта. Объектные библиотеки, исполняемые программы, компоненты . NETи Enterprise Java Beans, – все это примеры артефактов, которые могут быть представлены на UML. Но артефакты могут быть использованы и для представления других сущностей, которые принимают участие в работе исполняемых программ, – таких, как таблицы, файлы и документы.

Используя стереотипы – один из механизмов расширения UML, можно приспособить эту нотацию для представления специфических разновидностей артефактов.

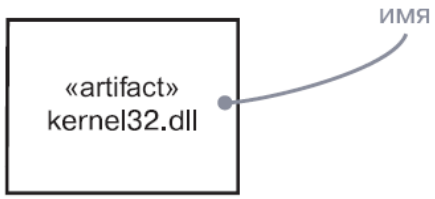

Рис.1. Артефакт

1. Діаграми розгортання.

Діаграми розгортання відображають відповідність конкретних програмних артефактів (пр.: файлів, які виконуються) обчислювальним вузлам (які виконують обробку).

Вони показують розміщення програмних продуктів у фізичній архітектурі системи і взаємодію (зазвичай мережеве) між фізичними елементами.

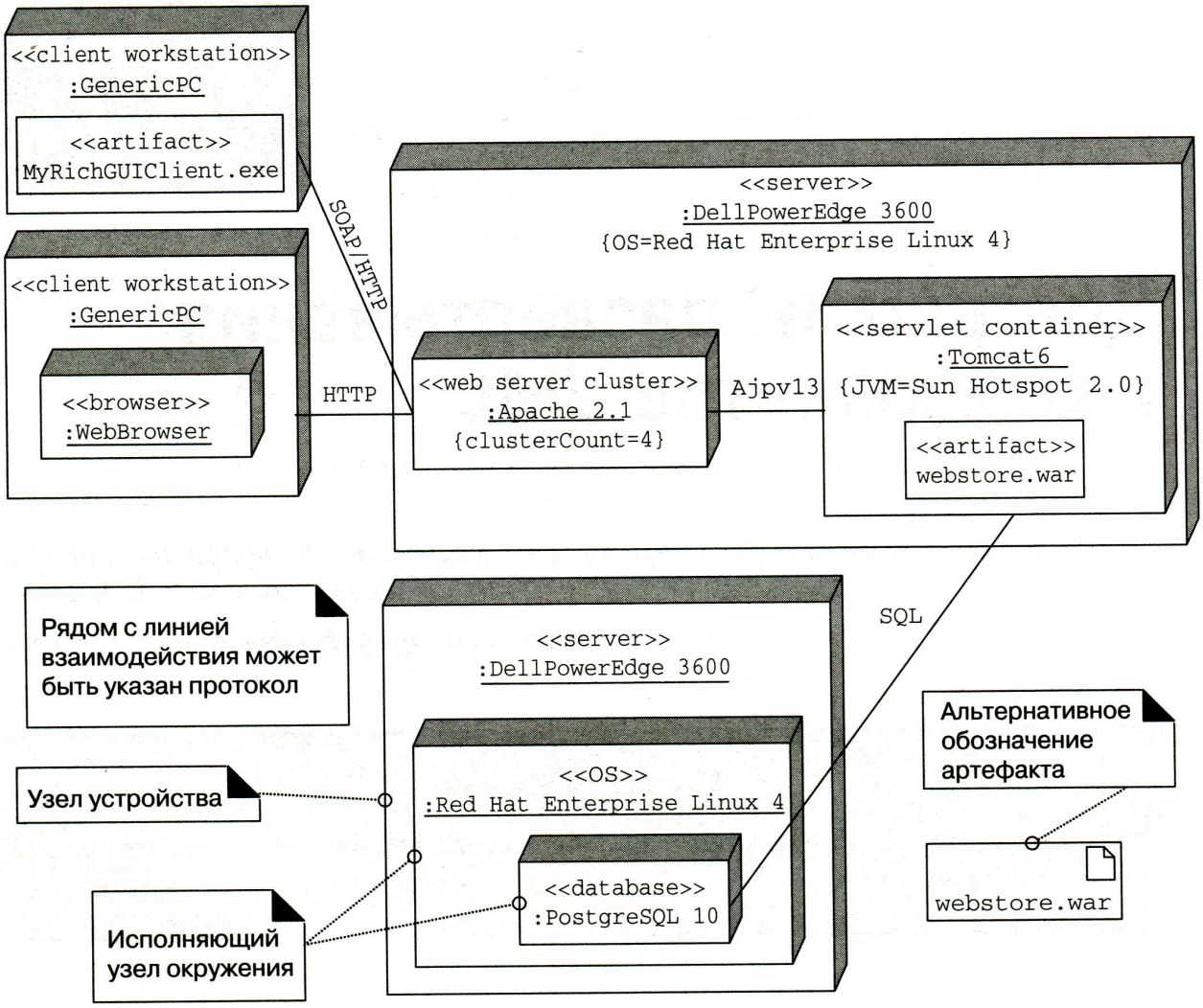

Рис. 1. Діаграма розгортання.

2. Вузол діаграми розгортання

Основним елементом діаграми розгортання є вузол.
Вузол відноситься до типів:

Вузол пристрою (чи просто пристрій). Це фізичний (пр.: цифровий чи електронний) обчислювальний ресурс з пам'яттю і процесорним елементом, на якому працює програмне забезпечення (пр.: комп'ютер, моб. телефон).

Виконуючий вузол оточення (execution environment node - EEN) – це прграмний обчислювальний ресурс, який працює в рамках іншого вузла (наприклад, комп'ютера) і який забезпечує виконання інших виконуємих програмних елементів.

Виконуючий вузол оточення

До виконуючого вузла оточення відносяться:
операційна система;
віртуальна машина (пр.: .NET, Java);
система управління базами даних (пр.: Microsoft SQL Server, Oracle Database);
Web-браузер, який відповідає за виконання сценаріїв JavaScript та т.і.;

механізм управління процесом виконання задач;

сервлет-контейнер

(Програма, що представляє собою сервер, який займається системною підтримкою сервлетов і забезпечує їх життєвий цикл відповідно до правил, визначеними в специфікаціях. Сервлет є інтерфейсом Java, реалізація якого розширює функціональні можливості сервера.) чи EJB-контейнер (EJB-контейнер - це те місце, де "живе" EJB-компонент. EJB-контейнер реалізує для компонент які знаходяться в ньому певні сервіси. EJB-компонент є набором Java-класів зі строго регламентованими правилами іменування методів.).

Багато типів вузлів відображаються за допомогою стереотипів, таких як <<server>>, <<OS>>, <<database>> і <<browser>>.

Конкретний елемент EEN може не відображатися чи неформально зображатися за допомогою рядка властивостей UML, пр.: {OS=Linux}.

З'єднання між вузлами називається комунікаційним шляхом, який часто помічається іменем протоколу, що реалізує мережеве з'єднання.

Вузол може мати артефакт – конкретний фізичний елемент (пр.: виконуючий файл: .exe, .jar, файл даних: .xml і т.і.)

На діаграмах розгортання зазвичай зображають екземпляри, а не класи. Імена конкретних екземплярів підкреслюються, а відсутність підкреслення свідчить про використання класу, а не екземпляра.

3. Артефакт.

Артефакт – это физическая часть системы, которая существует на уровне платформы реализации. Изображается в форме прямоугольника с ключевым словом «artifact» внутри.

Имена

Каждый артефакт должен обладать именем, отличающим его от других артефактов. Имя представляет собой текстовую строку; взятая в отдельности, она называется простым именем. Имя, предваренное именем пакета, в котором находится данный артефакт, называется квалифицированным. Обычно, изображая артефакт, указывают только его имя. Как и в случае с классами, вы можете дополнять пиктограммы артефактов помеченными значениями или дополнительными разделами, чтобы уточнить подробности.

Рис. 2. Імена

Артефакты и классы

И классы, и артефакты являются классификаторами. Однако между ними есть существенная разница:

классы представляют логические абстракции, а артефакты – физические сущности, которые существуют в мире битов. В силу этого артефакты могут располагаться на узлах, а классы – нет;

артефакты представляют физическую упаковку битов на платформе реализации;

классы могут иметь атрибуты и операции. Артефакты могут реализовывать классы и методы, но сами по себе не имеют атрибутов и операций.

В частности, артефакт – это физическая реализация набора логических элементов, таких как классы и кооперации.

Связь между артефактом и классами, которые он реализует, может быть показана явно с помощью материализации.

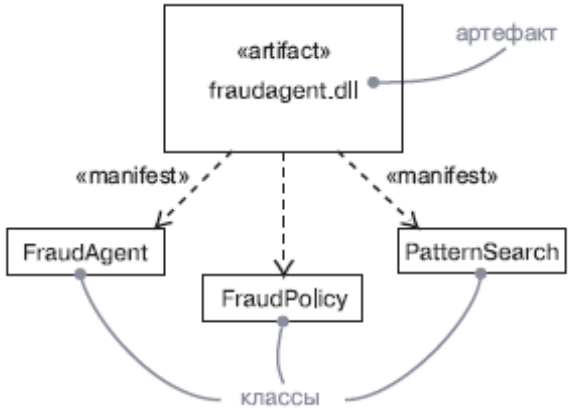

Рис. 3. Зв'язок між класами та артефактами.

4. Діаграми артефактів.

Диаграммы артефактов – это один из двух видов диаграмм, предназначенных для моделирования физических аспектов объектно-ориентированных систем. Диаграмма артефактов показывает организацию и зависимости между наборами артефактов.

Такие диаграммы используются для моделирования статического представления реализации системы, включая моделирование физических сущностей, размещаемых на узлах (например, исполнимых программ, библиотек, таблиц, файлов, документов разного рода). Диаграммы артефактов – это, по сути, диаграммы классов, которые сосредоточены на артефактах системы.

В UML диаграммы артефактов используются в целях визуализации статического аспекта физических артефактов и их связей, а кроме того, описания их деталей для конструирования.

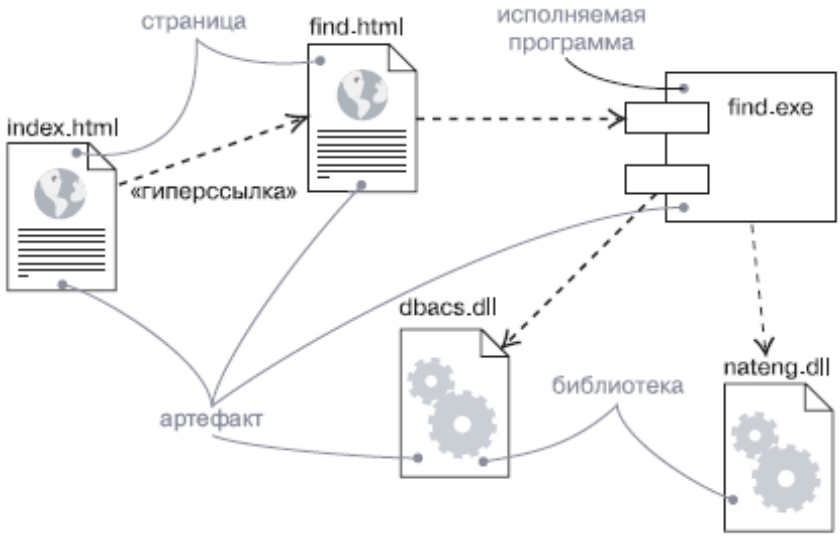

Рис. 4. Діаграма артефактів.

Диаграмма артефактов показывает набор артефактов и связей между ними. Изображается в виде графа с вершинами и дугами (ребрами).

Диаграммы артефактов применяются для моделирования статического представления реализации системы. Это представление в первую очередь поддерживает управление конфигурацией частей системы, состоящих из артефактов, которые могут быть собраны разными способами для построения работающей системы.

Когда моделируется статическое представление реализации системы, то диаграммы артефактов обычно используются одним из следующих четырех способов:

1. Для моделирования исходного кода. В большинстве современных объектноГориентированных языков программирования код пишется в интегрированных средах разработки, сохраняющих исходные тексты в файлах. Диаграммы артефактов можно применять для моделирования конфигурации этих файлов, которые представляют собой рабочие продукты, и установки вашей системы управления конфигурацией.

2. Для моделирования исполняемых версий. Версия (release) – это относительно полный и согласованный набор артефактов, поставляемый внутреннему или внешнему пользователю.

Версия в данном понимании сосредоточена на тех частях, которые необходимы для поставки работающей системы. При моделировании версий с помощью диаграмм артефактов происходят визуализация, специфицирование и документирование решений, принятых относительно физических составляющих системы, то есть артефактов размещения.

3. Для моделирования физических баз данных. Представьте себе физическую базу данных как конкретную реализацию схемы, существующую в мире битов. Схемы, по сути, описывают API для доступа к хранимой информации; модель же физической базы представляет способы хранения информации в таблицах реляционной базы или на страницах объектно-ориентированной базы данных. Для представления этих и иных физических баз данных можно использовать диаграммы артефактов.

4. Для моделирования адаптируемых систем. Некоторые системы достаточно статичны: их компоненты появляются на сцене, принимают участие в исполнении, а затем покидают ее. Другие системы более динамичны: они включают в себя мобильных агентов, или артефакты, мигрирующие с целью балансирования нагрузки и восстановления после сбоев. Поведение таких систем моделируется диаграммами артефактов совместно с некоторыми другими диаграммами UML.

Чтобы смоделировать исходный код системы, необходимо:

- Применяя прямое или обратное проектирование, идентифицировать набор представляющих интерес файлов исходного кода и смоделировать их как артефакты со стереотипом file.
- Для крупных систем использовать пакеты, чтобы показать группы исходных файлов.
- Рассмотреть возможность показа с помощью помеченных значений такой информации, как номер версии файла, его автор, дата последнего изменения. Для управления этими значениями применять инструментальные средства.
- Смоделировать зависимости компиляции между исходными файлами с помощью связей зависимости. Опять же, для того чтобы сгенерировать эти зависимости и управлять ими, понадобятся инструментальные средства.

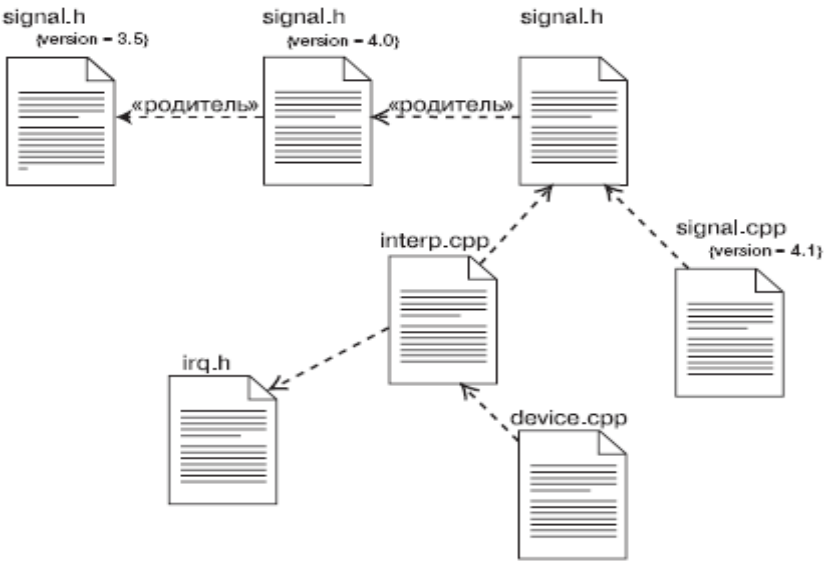

Рис. 5. Моделювання вхідного коду.

Чтобы смоделировать исполняемую версию, необходимо:

- Идентифицировать набор артефактов, подлежащих моделированию. Обычно это выборка либо вся совокупность артефактов, размещенных на одном узле, или же ряд артефактов, распределенных по узлам системы.
- Рассмотреть стереотипы каждого артефакта в наборе. Для большинства систем обнаружится ограниченное число разных видов артефактов (таких как исполняемые программы, библиотеки, таблицы, файлы и документы). Для визуального представления этих стереотипов можно воспользоваться механизмами расширения UML.

- Для каждого артефакта в наборе рассмотреть его связь с окружающими его артефактами. Чаще всего это интерфейсы, которые экспортирует (реализует) один артефакт и импортируют (используют) другие. Если нужно раскрыть соединения в системе, – явно смоделировать эти интерфейсы. Если же необходимо представить модель на более высоком уровне абстракции, раскрыть эти связи, показывая только зависимости между артефактами.

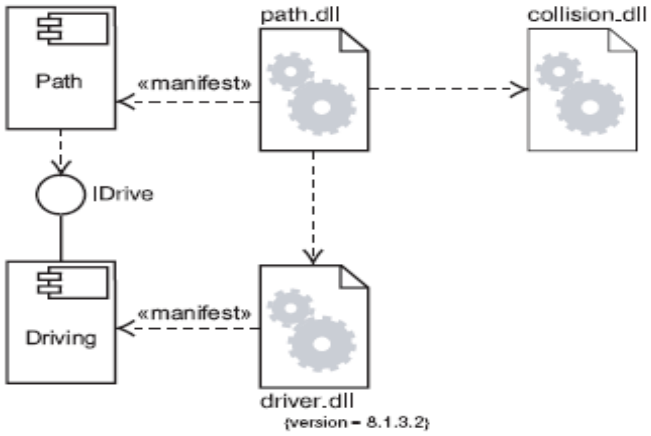

Рис.6. Моделювання версії, що виконується.

Моделювання фізичної бази даних

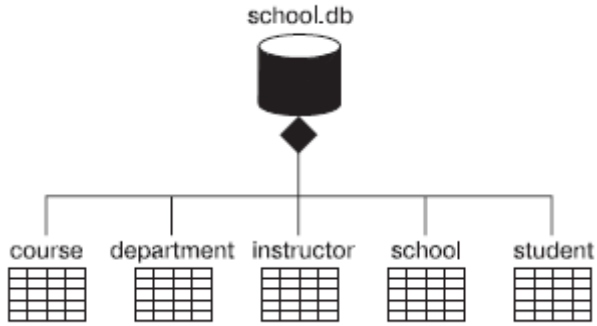

Рис.7. Моделювання фізичної бази даних

Моделювання систем, що адаптуються

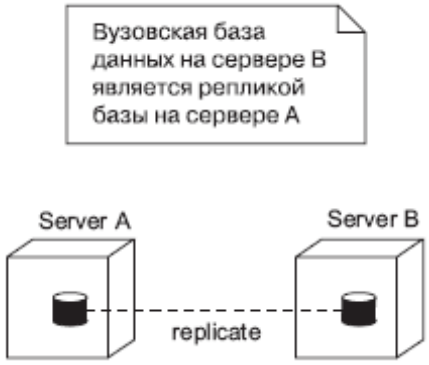

Рис. 8. Моделювання систем, що адаптуються.

Заключна частина

Іноді для уявлення системи достатньо відобразити діаграму рогортання.

Хорошо структурированная диаграмма артефактов обладает следующими свойствами:

- сосредоточена на выражении какогоFто одного аспекта статического представления реализации системы;
- содержит только те элементы, которые существенны для понимания данного аспекта;
- представляет уровень детализации, соответствующий ее уровню абстракции, включая лишь те дополнения, которые важны для понимания на этом уровне;
- не настолько лаконична, чтобы неверно информировать читателя о важной семантике.

Наочні посібники

Комп'ютер, мультимедійний проектор.

Завдання на самостійну роботу

1. Проектування за допомогою діаграми розгортання.

Старший викладач І.М.ГАМАНЮК
(ініціали, прізвище)