Предмет: «Технология разработки программного обеспечения»

Тема: «Диаграммы компонентов»

Теоретический материал

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними.

* При разработке диаграмм компонентов преследуются цели:
* спецификация общей структуры исходного кода системы;
* спецификация исполнимого варианта системы.

Данная диаграмма обеспечивает согласованный переход от логического к физическому представлению системы в виде программных компонентов. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Основными элементами диаграммы являются компоненты, интерфейсы и зависимости между ними. Кроме этого, на ней могут отображаться ключевые классы, входящие в компоненты.

Компонент (англ. component) – это физическая часть системы. Компоненты, представляющие собой файлы с исходным кодом классов, библиотеки, исполняемые модули и т.п., которые должны обладать согласованным набором интерфейсов. Для их графического представления используются следующие графические символы (Рисунок 1)

.

Рисунок 1 – примеры компонентов

Внутри прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация в виде помеченного значения.

Компоненты могут иметь следующие стандартные стереотипы:

* «file» – любой файл, кроме таблицы:
* «executable» – программа (исполняемый файл);
* «library» – статическая или динамическая библиотека;
* «source» – файл с исходным текстом программы;
* «document» – остальные файлы (например, файл справки);
* «table» – таблица базы данных.

Внутри компонента (рисунок 2), как и класса, могут быть выделены дополнительные секции, в которых указываются предоставляемые (provided) или необходимые для работы (required) интерфейсы и классы, методы (operations), наименование файла-компонента (artifacts) и т.п.



Рисунок 2 – компонент с секциями

Интерфейс (англ. interface) – это внешне видимый, именованный набор операций, который класс, компонент или подсистема может предоставить другому классу, компоненту или подсистеме, для выполнения им своих функций (Рисунок 3). В некоторых языках программирования, в частности в Java, интерфейс представляет собой отдельный класс, включаемый и реализуемый (конкретизируемый) в части программного кода операций в составе других классов. На диаграмме компонентов интерфейс отображается так же, как и на диаграмме классов (слева от компонента необходимые для работы интерфейсы, справа - предоставляемые).



Рисунок 3 – Способы отображения интерфейсов

Отношение ассоциации отображается между компонентами и их интерфейсами. Отношение зависимости означает зависимость реализации одних компонентов от реализации других. Такое возможно в следующих случаях:

* в методах классов одного компонента (зависимого) осуществляется вызов методов или обращение к атрибутам классов другого компонента (независимого);
* компонент состоит из других компонентов (например, при сборке исполняемого файла из файлов с исходными кодами);
* компонент осуществляет чтение или запись данных в другой компонент;
* связь между таблицами БД;
* и т.д.

Ввиду многоцелевого назначения диаграммы компонентов при ее разработке следует придерживаться следующих правил и рекомендаций.

1. Перед разработкой диаграмм компонентов необходимо решить, из каких физических частей (файлов) будет состоять программная система. При этом должно быть решено две задачи – распределение классов по файлам исходных кодов и по подсистемам. В последнем случае может помочь распределение классов по специализированным (функционально-ориентированным на предметную область) пакетам. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы возможность повторного использования кода за счет рациональной декомпозиции системы, т. е. минимизировать количество связей между компонентами.

2. При спецификации общей структуры исходного кода системы необходимо учитывать специфику языка программирования, с помощью которого реализуются компоненты. В частности в Java рекомендуется отдельный класс описывать в отдельном файле, несмотря на то, что язык позволяет описывать несколько классов в одном файле и использовать механизм внутренних классов. Диаграмма компонентов, применяемая для рассматриваемой цели, изображена на следующем рисунке (Рисунок 4).



Рисунок 4 – Фрагмент диаграммы компонентов, специфицирующей структуру исходного кода

Диаграмма на рисунке 4 показывает состав классов (файлов), из которых состоит исполняемый компонент iskraPUT.jar, а также зависимости между классами.

3. Для спецификации исполнимого варианта системы необходимо иметь в наличии предварительную топологию системы, т. е. набросок диаграммы развертывания. Для каждого узла в сети может быть построена диаграмма компонентов, определяющая набор файлов, необходимых для работы подсистемы (подсистем) на отдельном рабочем месте (рисунок 5).



Рисунок 5 – Пример диаграммы компонентов, специфицирующей состав компонентов на рабочем месте пользователя

4. На диаграмме могут быть представлены отношения зависимости между компонентами и включенными в них классами. Эта информация имеет важное значение для обеспечения согласованности между логическим и физическим представлениями системы. В этом случае зависимость можно показать двумя способами:

* классы показать отдельно от компонента и связать компонент с каждым классом отношением зависимости. Например, на рисунке 4 вместо компонентов с расширением «java» показать соответствующие им классы;
* классы отобразить внутри символа компонента.

5. Рекомендуется минимизировать количество связей между исполняемыми компонентами, а необходимое взаимодействие осуществлять через хранимую в единой базе данных информацию.

6. Для наглядного отображения специфики компонентов можно вместо стандартного символа компонента со строковым стереотипом внутри использовать графические стереотипы.

Домашнее задание

Нарисовать диаграмму компонентов для интернет магазина бытовой техники.

Результат скинуть на почту с названием по образцу — Предмет\_Тема\_датаОтправки.

В теме письма указать ФИО преподавателя.

Дата исполнения до следующей пары.