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 № 12

Тема лекції:

Кооперпації. Шаблони і каркаси. Системи та моделі.

План лекції

Вступ.
1. Кооперації.
2. Структура.
3. Поведінка.
4. Організація кооперацій.
5. Шаблони.
6. Механізми.
7. Каркаси.
8. Системи і підсистеми.
9. Моделі та представлення.
10. Трасування.

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

В контексте системной архитектуры кооперация позволяет выделить концептуальную часть, которая подчеркивает и статические, и динамические аспекты.

Кооперация помогает специфицировать реализацию вариантов использования и операций, а также моделировать архитектурно значимые механизмы системы.

Многие элементы хорошо структурированных систем в разных комбинациях принимают участие в функционировании различных механизмов.

В UML механизмы моделируются с помощью кооперацій. Кооперация именует совокупность взаимодействующих строительных блоков системы, включая как структурные, так и поведенческие элементы.

Кооперация не только именует системные механизмы, но и служит в качестве реализации вариантов использования и операций.

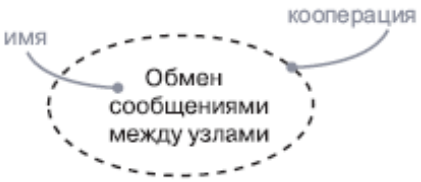

Рис.1. Кооперація

Все хорошо структурированные системы полны образцов (patterns).

Образец предлагает типичное решение типичной проблемы в данном контексте. Механизм – это образец проектирования, применяемый к сообществу классов; каркас (framework) – это, как правило, архитектурный образец, предлагающий расширяемый шаблон для приложений в некоторой предметной области.

Образцы используются для специфицирования механизмов и каркасов, образующих архитектуру системы. Вы делаете образец доступным, ясно идентифицируя все «вилки», «розетки», «кнопки» и «циферблаты», с помощью которых пользователь настраивает образец для применения его в определенном контексте.

Всякий раз, отрывая глаза от конкретных строк кода, вы обнаруживаете типичные механизмы, которые определяют способ организации классов и других абстракций. Например, в управляемой событиями системе применение образца проектирования, известного под именем «цепочка обязанностей», – это типичный способ организации обработчика событий. Если взглянуть чуть выше уровня этих механизмов, то вы увидите типичные каркасы, формирующие архитектуру всей системы. Например, в информационных системах применение трехслойной архитектуры – распространенный способ достижения четкого разделения обязанностей между пользовательским интерфейсом, хранением информации и бизнесFобъектами и правилами.

В UML вам нередко придется моделировать образцы проектирования, также называемые механизмами, которые можно представить в виде кооперацій.

Аналогичным образом архитектурные образ цы моделируются как каркасы, представляемые в виде пакетов со стереотипами.

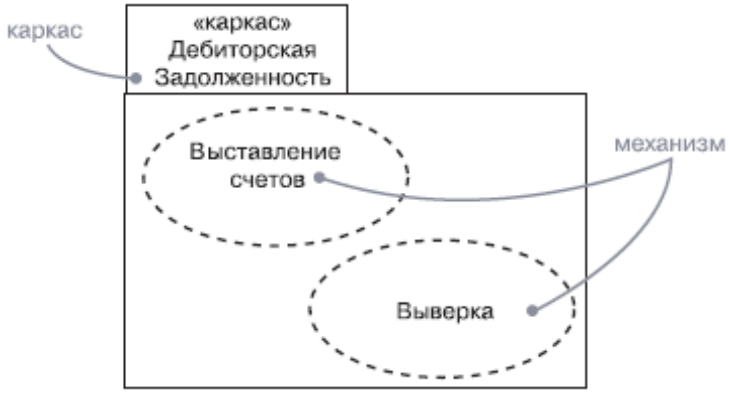

Рис.2. Механізми та каркаси.

UML – это графический язык для визуализации, специфицирования, конструирования и документирования артефактов программной системы. Его используют для того, чтобы моделировать системы. Модель – это упрощение реальности, абстракция, создаваемая, чтобы лучше понять систему. Система, зачастую разложенная на ряд подсистем, – это множество элементов, организованных некоторым образом для выполнения определенной цели. Система описывается набором моделей, по возможности рассматривающих ее с различных точек зрения. Важными составными частями модели являются такие сущности, как классы, интерфейсы, компоненты и узлы. В UML допускается моделирование систем и подсистем как единого целого – тем самым органично решается проблема масшта бирования.

Хорошо структурированные модели помогают визуализировать, специфицировать, конструировать и документировать систему под разными (но вместе с тем взаимосвязанными) углами зрения.

Хорошо структурированные системы функционально, логически и физически согласованы, но при этом состоят из слабо связанных друг с другом подсистем.

В UML все абстракции программной системы организуются в виде моделей, каждая из которых представляет относительно независимый, но важный аспект разрабатываемой системы. Для визуализации интересующих вас наборов этих абстракций можно использовать диаграммы. Рассмотрение пяти различных представлений архитектуры системы особенно полезно для удовлетворения потребностей различных участников процесса разработки програм много обеспечения. В своей совокупности эти модели дают полное представление о структуре и поведении системы.

В больших системах множество элементов можно подвергнуть декомпозиции на более мелкие подсистемы, каждая из которых на более низком уровне абстракции может рассматриваться как отдель ная система.

В UML предусмотрены определенные средства для графического представления систем и подсистем. Эта нотация позволяет визуализировать декомпозицию системы на меньшие подсистемы. И система, и подсистема обозначаются пиктограммой компонента со стереотипом. Для моделей и

представлений предусмотрено особое графическое изображение (отличное от пакетов со стереотипом), хотя применяется оно редко, поскольку эти сущности в основном являются объектами манипуляции инструментальных средств, которыми пользуются для организации различных аспектов системы.

1. Кооперація.

Кооперация (collaboration) – это сообщество классов, интерфейсов и других элементов, которые работают совместно для обеспечения определенного поведения, более значимого, чем поведение суммы всех тех же составляющих. Кооперация также показывает, как некий элемент – например, классификатор (включая класс, интерфейс, компонент, узел или вариант использования) либо операция, реализуется набором классификаторов и ассоциаций, каждая из которых определенным образом играет определенную роль. Кооперация изображается в виде эллипса с пунктирной границей.

2. Структура.

Кооперации имеют две составляющие: структурную, которая описывает классы, интерфейсы и другие совместно работающие элементы, и поведенческую, которая описывает динамику взаимодействия этих элементов.

Структурная составляющая кооперации – это внутренняя (составная) структура, которая может включать любую комбинацию классификаторов, таких как классы, интерфейсы, компоненты, узлы. Внутри кооперации эти классификаторы могут быть организованы с использованием всех обычных связей UML, включая ассоциации, обобщения и зависимости. Фактически структурные аспекты кооперацій могут использовать полный диапазон средств структурного моделирования UML.

Однако в отличие от структурированных классов кооперация не владеет своими структурными элементами. Вместо этого она просто ссылается на классы, интерфейсы, компоненты, узлы и прочие структурные элементы, объявленные в другом месте, или использует их. Вот почему кооперация именует концептуальный, а не физический фрагмент системной архитектуры. Кооперация может распространяться на многие уровни системы. Более того, один и тот же элемент может принимать участие в нескольких кооперациях (а некоторые элементы не будут являться частью ни одной из них).

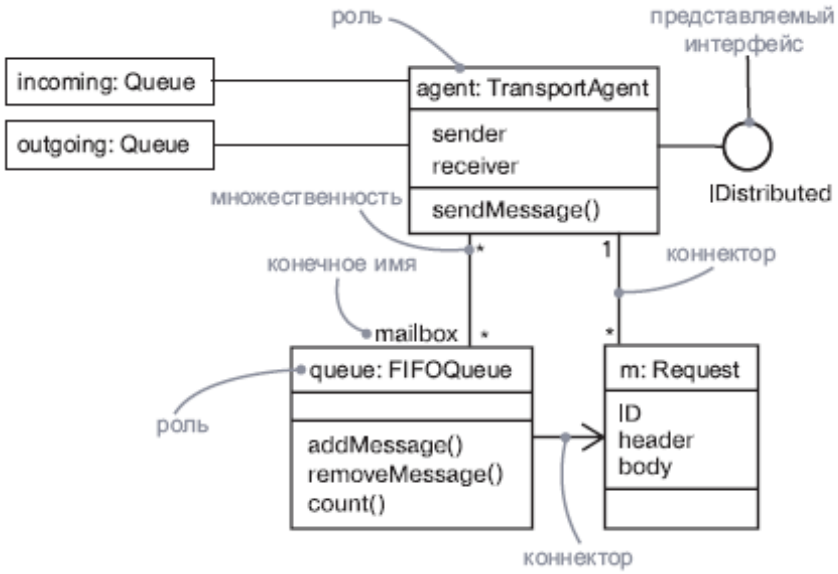

Рис. 4. Структурні аспекти кооперації.

3. Поведінка.

Если структурная составляющая кооперации обычно изображается как диаграмма составной структуры, то поведенческая в большинстве случаев представлена диаграммой взаимодействия.

Эта диаграмма описывает взаимодействие, соответствующее поведению, суть которого – обмен сообщениями между объектами в некотором контексте для достижения определенной цели. Контекст взаимодействия устанавливает сама кооперация, определяющая классы, интерфейсы, компоненты, узлы и другие структурные элементы, экземпляры которых могут принимать участие во взаимо действии.

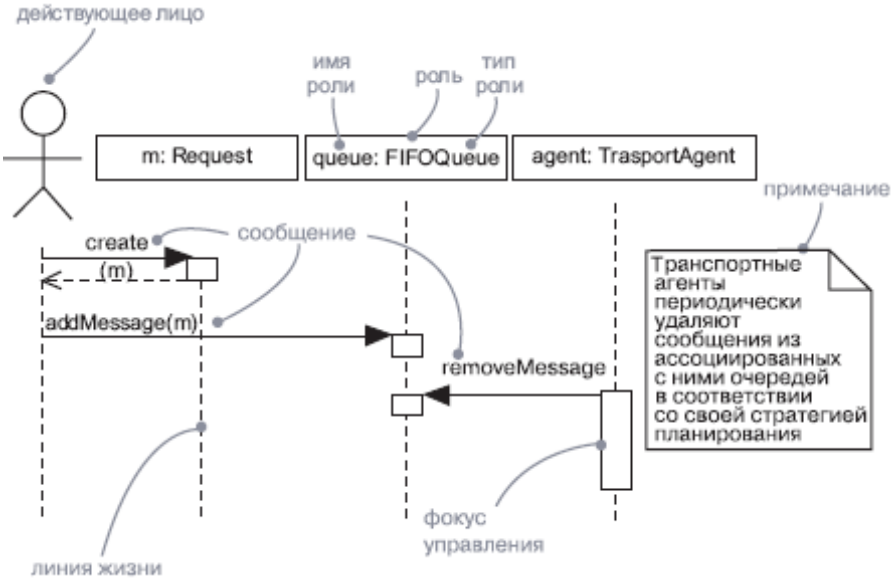

Рис.5. Аспекти поведінки кооперації.

4. Організація кооперацій.

В кооперациях системы находится сердце ее архитектуры, потому что лежащие в основе системы механизмы представляют существенные проектные решения. Все хорошо структурированные объектно-ориентированные системы состоят из регулярного (то есть построенного в соответствии с четкими правилами) множества кооперацій относительно небольшого размера, поэтому очень важно научиться их организовывать. Существует два вида относящихся к кооперациям связей, которые следует принимать во внимание.

Во-первых, это связь между кооперацией и тем, что она реализует. Кооперация может реализовывать либо классификатор, либо операцию. Это означает, что кооперация описывает структурную или поведенческую реализацию соответствующего классификатора или операции. Например, вариант использования, который именует набор сценариев, выполняемых системой, может быть реализован в виде кооперации. Этот вариант использования вместе с его действующими лицами и окружающими вариантами использования представляет контекст кооперации. Аналогично кооперацией может быть реализована и операция (которая именует реализацию
некоторой системной услуги). В таком случае контекст формирует данная операция вместе со своими параметрами и возможным возвращаемым значением. Такая связь между вариантом использования или операцией и реализующей кооперацией моделируется с помощью связи реализации.

Во-вторых, существуют связи между самими кооперациями. Одни из них могут уточнять описания других; это может быть смоделировано в виде связи уточнения. Подобные связи между кооперациями обычно отражают связи уточнения между вариантами использования, которые они представляют.

Рис.6. Організація кооперацій.

5. Шаблони.

Образец (pattern) – это общее решение типичной проблемы в данном контексте. Механизм – это образец проектирования, применимый к сообществу классов. Каркас (framework) – это архитектурный образец, предлагающий расширяемый шаблон для приложений в некоторой предметной области.

Занимаясь разработкой архитектуры новой системы или развитием существующей, вы в любом случае никогда не начинаете с нуля. Напротив, прежний опыт и соглашения наталкивают вас на применение типичных приемов решения типичных проблем.

Любая хорошо структурированная система включает в себя множество образцов на различных уровнях абстракции. Образцы проектирования описывают структуру и поведение сообщества классов, а архитектурные образцы – структуру и поведение системы в целом.

Образцы входят в UML просто потому, что являются важной составляющей словаря разработчика. Явно выделяя их в системе, вы делаете ее более понятной и простой в сопровождении.

Одна лишь фраза «система организована как набор конвейеров и фильтров» очень многое говорит о системной архитектуре – понять это, глядя на код классов, было бы куда сложнее.

Образцы помогают визуализировать, специфицировать, конструировать и документировать артефакты программной системы. Можно заниматься прямым проектированием системы, выбирая подходящий набор образцов и применяя их к абстракциям, специфичным для данной предметной области, или же обратным проектированием, выявляя содержащиеся в системе образцы (хотя вряд ли этот процесс можно назвать очень продуктивным). Впрочем, при поставке системы было бы еще лучше описать характерные для нее образцы, чтобы помочь тому, кому придется в будущем повторно использовать или модифицировать ваш код.

На практике интерес представляют только два вида образцов – образцы проектирования и каркасы. В UML предусмотрены средства моделирования и тех и других. При моделировании любого образ ца вы обнаружите, что он, как правило, является автономным в некотором большом пакете, если не считать зависимостей, связывающих этот образец с остальными частями системы.

6. Механізми.

Механизм – это образец проектирования, примененный к сообществу классов. Например, существует типичная проблема проектирования, с которой сталкивается программист, пишущий на языке Java: как изменить класс, который умеет реагировать на некоторое множество событий, таким образом, чтобы он реагировал на события иного рода, не затрагивая исходного кода этого класса? Типичное решение проблемы – применение образца адаптера (adaptor pattern), структурного образца проектирования, который конвертирует один интерфейс в другой. Этот образец является настолько общим, что имеет смысл дать ему название, а затем использовать в моделях всякий раз, когда возникает аналогичная проблема.

При моделировании механизмы проявляют себя двойственно.

Во-первых, механизм просто именует набор абстракций, работающих вместе для реализации типичного поведения, представляющего некоторый интерес. Такие механизмы моделируются как простые кооперации, поскольку они являются всего лишь именами для сообщества классов. Раскрыв такую кооперацию, можно увидеть ее структурные аспекты (обычно изображаемые на диаграмме классов), а также поведенческие аспекты (обычно изображаемые на диаграммах взаимодействия).

Кооперации подобного типа охватывают разные уровни абстракции системы, то есть какойFто конкретный класс, вероятно, будет участвовать в нескольких кооперациях.

Во-вторых, механизм именует шаблон для набора абстракций, работающих совместно для обеспечения некоторого типичного поведения, представляющего интерес. Такие механизмы моделируются в виде параметризованных кооперацій, которые изображаются в UML подобно шаблонным классам. Если раскрыть такую кооперацию, можно увидеть ее структурные и поведенческие аспекты. Если свернуть ее, то можно увидеть, как образец применяется к системе, связывая шаблонные части кооперации с существующими абстракциями системы.

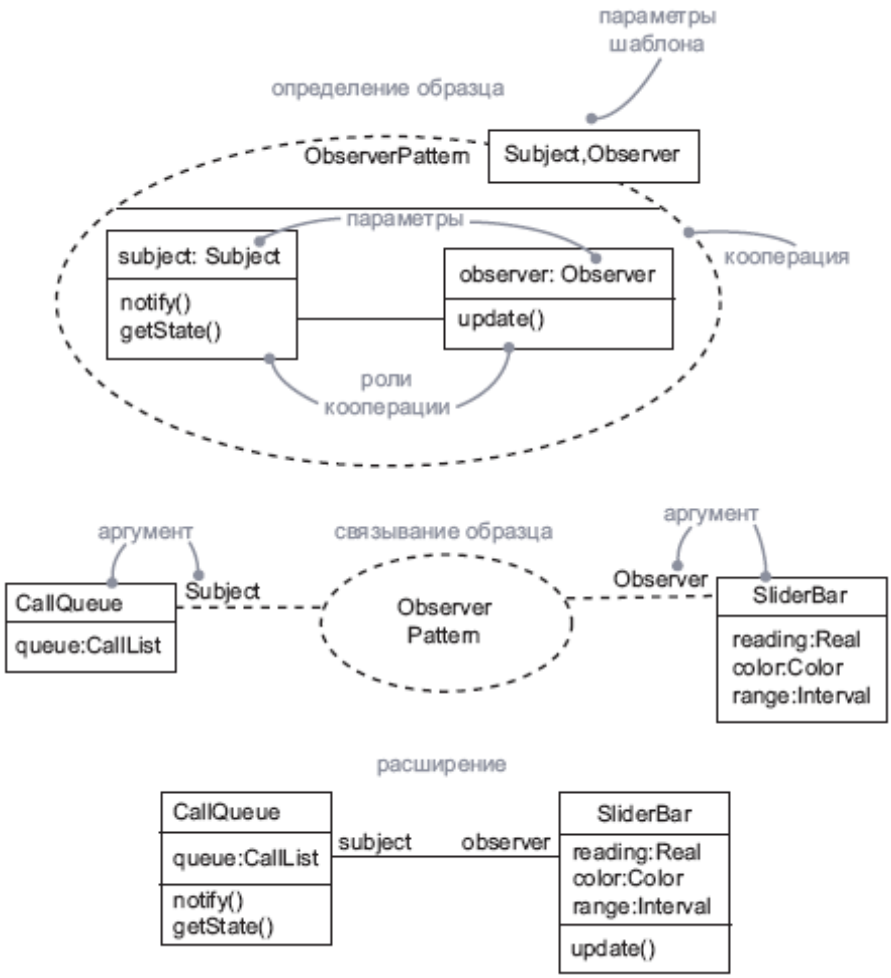

Рис. 7. Механізми.

7. Каркаси.

Каркас (framework) – это архитектурный образец, предлагающий расширяемый шаблон для приложений в некоторой предметной области. Например, в системах

реального времени часто можно встретить архитектурный образец «циклический исполнитель» (cyclic executive), который разделяет время на кадры и подкадры, где обработка происходит в строгих временных рамках. Выбор этого образца вместо управляемой событиями архитектуры оказывает влияние на всю систему. Данный образец (как и его альтернатива) является настолько общим, что имеет смысл назвать его каркасом.

Каркас – это нечто большее, чем механизм. Фактически можно считать каркас разновидностью микроархитектуры, включающей в себя множество механизмов, совместно работающих над решением типичной проблемы для типичной предметной области. Специфицируя каркас, вы тем самым описываете «скелет» архитектуры вместе со всеми ее органами управления, которые применяются пользователем для адаптации к нужному контексту.

В UML каркас моделируется в виде пакета со стереотипом. Заглянув внутрь этого пакета, можно увидеть механизмы, существующие в любом из представлений системной архитектуры. Например, там обнаружатся не только параметризованные кооперации, но также варианты использования (которые объясняют, как надо работать с этим каркасом), а также простые кооперации (представляющие набор абстракций, на базе которых можно строить систему, – например, путем порождения классов-потомков).

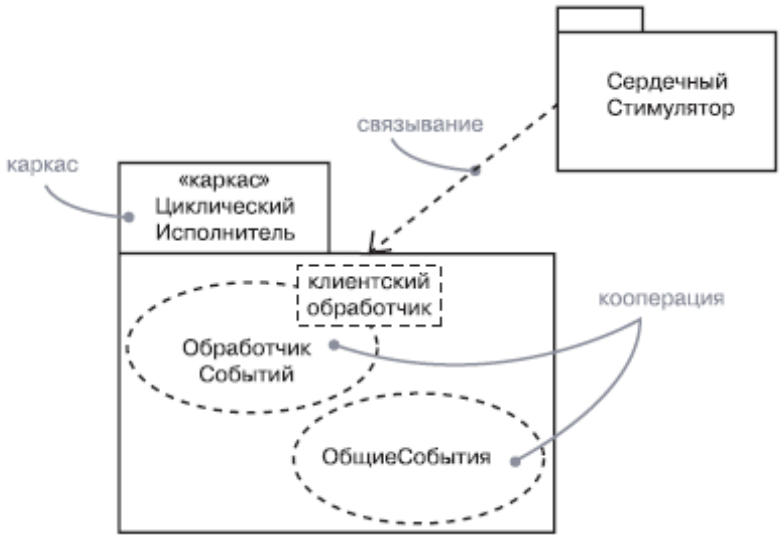

Рис.8. Каркаси.

8. Системи та підсистеми.

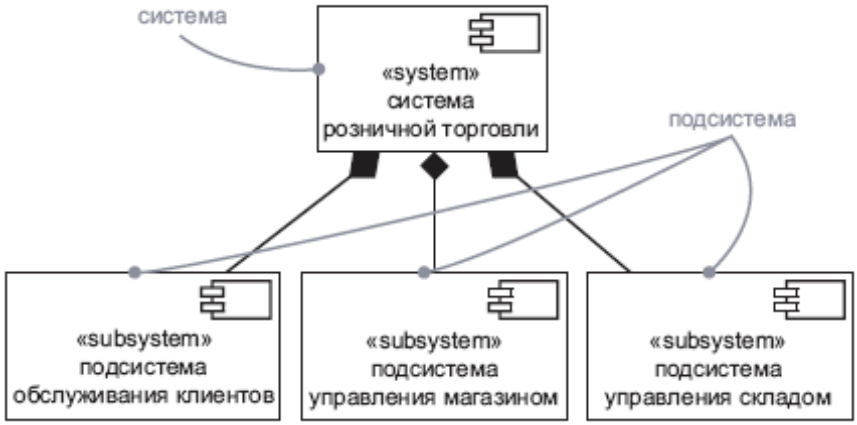

Рис.9. Системи та підсистеми.

Система (system), возможно декомпозированная на ряд подсистем, – это множество элементов, организованных некоторым образом для выполнения определенной цели. Она описывается набором моделей, зачастую с различных точек зрения.

Подсистема (subsystem) – это объединение элементов, ряд которых составляет спецификацию поведения других ее элементов.

Система и подсистема изображаются в виде пиктограммы компонента со стереотипом.

Модель (model) – это упрощение реальности, абстракция, создаваемая для лучшего восприятия системы.

Представление (view) – это проекция модели, рассматриваемая под определенным углом зрения: в ней отражены одни сущности и опущены другие, которые с данной точки зрения не представляют интереса.

Система, собственно, и есть та сущность, которую вы разрабатываете и для которой строите модели. Система охватывает все артефакты, составляющие эту сущность, включая модели и элементы этих моделей (такие как классы, интерфейсы, компоненты, узлы и связи между ними). Все, что необходимо для визуализации, специфицирования, конструирования и документирования системы, является ее частью, а все, что для этой цели не требуется, лежит за пределами системы.

В UML система изображается в виде компонента со стереотипом, как показано на рис. 32.1. Являясь компонентом со стереотипом, она владеет некоторыми элементами. Если заглянуть внутрь системы, то можно увидеть все ее модели и отдельные элементы (в том числе и диаграммы), зачастую декомпозированные на более мелкие подсистемы. Являясь классификатором, система может иметь экземпляры (допускается существование нескольких экземпляров системы, размещенных в разных точках), атрибуты и операции (внешние по отношению к системе действующие лица способны воздействовать на систему в целом), варианты использования, автоматы и кооперации; все они могут принимать участие в специфицировании поведения системы.

В ряде случаев она обладает даже интерфейсами, что оказывается важным при конструировании системы систем.

Подсистема – это просто часть системы, которая используется для декомпозиции сложной системы на более простые, слабо зависящие друг от друга составляющие. То, что на одном уровне абстракции выглядит системой, на другом – более высоком – может рассматриваться как подсистема.

Основная связь между системой и ее подсистемами – это композиция. Система (целое) может состоять из одной или нескольких подсистем (частей) либо вообще не содержать таковых. Допускается наличие связей обобщения между подсистемами. Благодаря этому можно моделировать семейства подсистем, ряд которых представляет общие виды систем, а другие являют собой специализацию этих систем, рассчитанную на конкретные условия. Подсистемы могут быть соединены друг с другом различными способами.

9. Моделі та представлення.

Модель – это упрощение реального мира; реальность в ней описывается в контексте моделируемой системы. Проще говоря, модель – это абстракция системы. В то время как подсистема представляет собой разбиение множества элементов большей системы на независимые части, модель – это разбиение множества абстракций, используемых для визуализации, специфицирования, констру ирования и документирования этой системы. Различие тонкое, но важное. Вы декомпозируете систему на подсистемы, чтобы их можно было разрабатывать и размещать в некоторой степени независимо друг от друга. Абстракции же системы или подсистемы вы разбиваете на модели, дабы лучше понять то, что собираетесь разрабатывать или размещать. Сложная система, например самолет, состоит из многих частей (каркас, реактивные двигатели, авиационная электроника, подсистема обслуживания пассажиров), причем эти подсистемы и система в целом могут моделироваться с разных точек зрения – в частности, с точки зрения конструкции, динамики, электросистемы, моделей отопления и кондиционирования.

Модель содержит набор пакетов. Явно представлять ее приходится не так уж часто. Однако инструментальные средства должны каким-то образом манипулировать моделями и обычно используют для их представления нотацию пакетов.

Модель владеет пакетами, которые, в свою очередь, состоят из неких элементов. Модели, ассоциированные с системой или подсистемой, исчерпывающим образом разбивают ее элементы. Это означает, что каждый элемент принадлежит одному и только одному пакету. Как правило, артефакты системы или подсистемы организуются в несколько неперекрывающихся моделей. Все возможные модели охватываются пятью представлениями архитектуры программного обеспечения.

Модель (к примеру, процесса) может содержать так много артефактов – скажем, активных классов, связей и взаимодействий, – что в большой системе всю их совокупность нельзя охватить сразу.

Представление системной архитектуры можно воспринимать как одну из проекций модели. Для каждой модели предусмотрен ряд диаграмм, с помощью которых удобно обозревать принадлежащие ей сущности. Представление охватывает подмножество сущностей, входящих в состав модели. Границы моделей представления обычно пересекать не могут. В следующем разделе будет показано,

что между моделями нет прямых связей, хотя между элементами, содержащимися в различных моделях, могут существовать связи трассировки.

10. Трасування.

Специфицирование связей между такими элементами, как классы, интерфейсы, компоненты и узлы, – важная структурная составляющая модели. Для управления артефактами процесса разработки сложных систем, многие из которых существуют в нескольких версиях, большую роль играет определение связей между такими элементами, как документы, диаграммы и пакеты, присутствующие в разных моделях.

В UML концептуальные связи между элементами, содержащимися в разных моделях, можно представлять с помощью трассировки (trace relationship). К элементам в рамках одной модели трассировку применять нельзя. Изображается она в виде зависимости со стереотипом. Часто можно не обращать внимания на направление такой зависимости, хотя стрелка обычно указывает на элемент, возникший раньше по времени или более специфический Чаще всего связи трассировки используются, чтобы показать путь от требований до реализации, на котором лежат все промежуточные артефакты, а также для отслеживания версий.

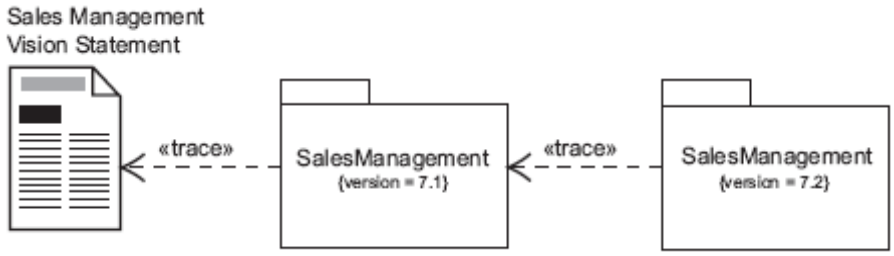

Рис.10. Трасування.

Заключна частина

Хорошо структурированная кооперация обладает следующими свойствами:

- включает как структурный, так и поведенческий аспекты;
- представляет собой четкую абстракцию некоторого идентифицируемого взаимодействия в системе;
- редко бывает полностью независимой, но чаще перекрывается со структурными элементами других кооперацій;
- понятна и проста.

При моделировании образцов в UML следует помнить, что они работают на многих уровнях абстракции, начиная от отдельных классов и заканчивая системой в целом. Самые интересные виды образцов – это механизмы и каркасы. Хорошо структурированный образец обладает следующими свойствами:

- решает типичную проблему типичным образом;
- включает структурную и поведенческую составляющие;
- раскрывает элементы управления и стыковки, с помощью которых его можно настроить на разные контексты;

- охватывает различные индивидуальные абстракции в системе.

Изображая образец в UML, необходимо:

- раскрывать те его элементы, которые следует адаптировать для применения в конкретном контексте;
- приводить варианты использования образца, а также способы его адаптации.

Чтобы смоделировать систему систем, вам следует:

- Идентифицировать основные функциональные составляющие системы, которые можно разрабатывать, выпускать и размещать до некоторой степени независимо. На результаты этого разбиения системы часто влияют технические, политические и юридические факторы.
- Для каждой подсистемы специфицировать ее контекст также, как это делается для системы в целом (притом в число действующих лиц, окружающих подсистему, включаются все соседние подсистемы, поэтому необходимо проектировать их совместную работу).
- Смоделировать архитектуру каждой подсистемы так же, как это делается для всей системы.

Наочні посібники

Комп’ютер, мультимедійний проектор.

Завдання на самостійну роботу

1. Проектування за допомогою діаграми шаблонів.

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