Методология объектно-ориентированного моделирования

Подготовили: Акжан Н.

Молдагожин Б.

Торгамбаев С.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML)

- Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью UML жизненный цикл информационных систем, и, одновременно, UML является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Создание UML началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов OMT и Booch. В настоящее время консорциум пользователей UML Partners включает в себя представителей таких грандов информационных технологий, как Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

Основные разработчики языка UML (Three amigos)

Grady Booch
Гради Буч

Dr. James Rumbaugh
Джеймс Рамбо
(Джим Румбах)

Dr. Ivar Jacobson
Айвар Джекобсон
(Ивар Якобсон)

- **OMG** (Object Management Group) — название консорциума, созданного в 1989 году для разработки индустриальных стандартов с их последующим использованием в процессе создания масштабируемых неоднородных распределенных объектных сред.
- Официальный сайт: www.omg.org

Стандарт UML предлагает следующий набор диаграмм для моделирования:

- * диаграммы вариантов использования (use case diagrams) – для моделирования бизнес-процессов организации и требований к создаваемой системе);
- * диаграммы классов (class diagrams) – для моделирования статической структуры классов системы и связей между ними;
- * диаграммы поведения системы (behavior diagrams):
- o диаграммы взаимодействия (interaction diagrams):
- * диаграммы последовательности (sequence diagrams) и
- * кооперативные диаграммы (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами;
- o диаграммы состояний (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;
- o диаграммы деятельностей (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей;
- * диаграммы реализации (implementation diagrams):
- o диаграммы компонентов (component diagrams) – для моделирования иерархии компонентов (подсистем) системы;
- o диаграммы развертывания (deployment diagrams) – для моделирования физической архитектуры системы.

Диаграммы вариантов использования

- Понятие варианта использования (use case) впервые ввел Ивар Якобсон и придал ему такую значимость, что в настоящее время вариант использования превратился в основной элемент разработки и планирования проекта.

- Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать.

- Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в
- том случае, когда им действительно необходимы некоторые варианты использования. На языке UML действующие лица представляют в виде фигур

- Действующие лица делятся на три основных типа:
- * пользователи;
- * системы;
- * другие системы, взаимодействующие с данной;
- * время(время становится действующим лицом, если от него зависит запуск каких-либо событий в системе).

Класс

- Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.
- Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.
- На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 секции:
- * Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
- * В средней секции содержится список атрибутов
- * В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).

- Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.
- На усмотрение конкретной реализации могут быть введены дополнительные секции, например, исключения (Exceptions).

Операции

- Операции реализуют связанное с классом поведение. Операция включает три части – имя, параметры и тип возвращаемого значения.
- Параметры – это аргументы, получаемые операцией «на входе». Тип возвращаемого значения относится к результату действия операции.
- На диаграмме классов можно показывать как имена операций, так и имена операций вместе с их параметрами и типом возвращаемого значения. Чтобы уменьшить загруженность диаграммы, полезно бывает на некоторых из них показывать только имена операций, а на других их полную сигнатуру.
- В языке UML операции имеют следующую нотацию:
- Имя Операции (аргумент: тип данных аргумента, аргумент2:тип данных аргумента2,...): тип возвращаемого значения

- Следует рассмотреть четыре различных типа операций:
- * Операции реализации;
- * Операции управления;
- * Операции доступа;
- * Вспомогательные операции.

Связи

- Связь представляет собой семантическую взаимосвязь между классами. Она дает классу возможность узнавать об атрибутах, операциях и связях другого класса. Иными словами, чтобы один класс мог послать сообщение другому на диаграмме последовательности или кооперативной диаграмме, между ними должна существовать связь.
- Существуют четыре типа связей, которые могут быть установлены между классами: ассоциации, зависимости, агрегации и обобщения.

Множественность

- Множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью этой связи с одним экземпляром другого класса в данный момент времени.
- Например, при разработке системы регистрации курсов в университете можно определить классы Course (курс) и Student (студент). Между ними установлена связь: у курсов могут быть студенты, а у студентов – курсы. Вопросы, на который должен ответить параметр множественности: «Сколько
- курсов студент может посещать в данный момент? Сколько студентов может за раз посещать один курс?»
- Так как множественность дает ответ на оба эти вопроса, её индикаторы устанавливаются на обоих концах линии связи. В примере регистрации курсов мы решили, что один студент может посещать от нуля до четырех курсов, а один курс могут слушать от 0 до 20 студентов.

Имена связей

- Имена связей
- Связи можно уточнить с помощью имен связей или ролевых имен. Имя связи – это обычно глагол или глагольная фраза, описывающая, зачем она нужна. Например, между классом Person (человек) и классом Company (компания) может существовать ассоциация. Можно задать в связи с этим вопрос, является ли объект класса Person клиентом компании, её сотрудником или владельцем?

Пакет. Механизм пакетов

- В контексте диаграмм классов, пакет - это вместилище для некоторого набора классов и других пакетов. Пакет является самостоятельным пространством имен.
- В UML нет каких-либо ограничений на правила, по которым разработчики могут или должны группировать классы в пакеты. Но есть некоторые стандартные случаи, когда такая группировка уместна, например, тесно взаимодействующие классы, или более общий случай - разбиение системы на подсистемы.
- Пакет физически содержит сущности, определенные в нем (говорят, что "сущности принадлежат пакету"). Это означает, что если будет уничтожен пакет, то будут уничтожено и все его содержимое.

Диаграммы состояний

- Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий.
- Существует много форм диаграмм состояний, незначительно отличающихся друг от друга семантикой.

Специальные состояния

- На диаграмме имеются два специальных состояния – начальное (start) и конечное (stop). Начальное состояние выделено черной точкой, оно соответствует состоянию объекта, когда он только что был создан.
- Конечное состояние обозначается черной точкой в белом кружке, оно соответствует состоянию объекта непосредственно перед его уничтожением. На диаграмме состояний может быть одно и только одно начальное состояние. В то же время, может быть столько конечных состояний, сколько вам нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии, могут выполняться различные процессы. Процессы, происходящие, когда объект находится в определенном состоянии, называются действиями (actions).
- С состоянием можно связывать данные пяти типов: деятельность, входное действие, выходное действие, событие и история состояния.

Деятельность

- Деятельностью (activity) называется поведение, реализуемое объектом, пока он находится в данном состоянии. Деятельность – это прерываемое поведение. Оно может выполняться до своего завершения, пока объект находится в данном состоянии, или может быть прервано переходом объекта в другое состояние. Деятельность изображают внутри самого состояния, ей должно предшествовать слово do (делать) и двоеточие.

Входное действие

- Входным действием (entry action) называется поведение, которое выполняется, когда объект переходит в данное состояние. Данное действие осуществляется не после того, как объект перешел в это состояние, а, скорее, как часть этого перехода. В отличие от деятельности, входное действие рассматривается как непрерываемое. Входное действие также показывают внутри состояния, ему предшествует слово entry (вход) и двоеточие.

Выходное действие

- Выходное действие (exit action) подобно входному. Однако, оно осуществляется как составная часть процесса выхода из данного состояния. Оно является частью процесса такого перехода. Как и входное, выходное действие является непрерываемым.
- Выходное действие изображают внутри состояния, ему предшествует слово exit (выход) и двоеточие.

События

- Событие (event) – это то, что вызывает переход из одного состояния в другое. Событие размещают на диаграмме вдоль линии перехода.
- На диаграмме для отображения события можно использовать как имя операции, так и обычную фразу.
- Большинство переходов должны иметь события, так как именно они, прежде всего, заставляют переход осуществиться. Тем не менее, бывают и автоматические переходы, не имеющие событий. При этом объект сам перемещается из одного состояния в другое со скоростью, позволяющей осуществиться входным действиям, деятельности и выходным действиям.

Ограждающие условия

- Ограждающие условия (guard conditions) определяют, когда переход может, а когда не может осуществиться. В противном случае переход не осуществится.
- Ограждающие условия изображают на диаграмме вдоль линии перехода после имени события, заключая их в квадратные скобки.
- Ограждающие условия задавать необязательно. Однако если существует несколько автоматических переходов из состояния, необходимо определить для них взаимно исключающие ограждающие условия. Это поможет читателю диаграммы понять, какой путь перехода будет автоматически выбран.

Действие

- Действием (action), как уже говорилось, является непрерываемое поведение, осуществляющееся как часть перехода. Входные и выходные действия показывают внутри состояний, поскольку они определяют, что происходит, когда объект входит или выходит из него. Большую часть действий, однако, изображают вдоль линии перехода, так как они не должны осуществляться при входе или выходе из состояния.

Диаграммы размещения

- Диаграмма размещения (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе.
- Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.
- Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов.

Диаграммы компонентов

- Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.
- Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы.