Министерство науки высшего образования Российской Федерации

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплине «Проектирование информационного обеспечения САПР»

**ЭССЕ**

**Глава №2 «МОДЕЛИРОВАНИЕ ДАННЫХ С ПОМОЩЬЮ ДИАГРАММЫ КЛАССОВ UML»**

Выполнил студент:

Группы ИВТАПбд-41

Галацков И.А.

Проверил:

Доктор технических наук,

Профессор Токмаков Г.П.

Ульяновск, 2024

**ГЛАВА 2. МОДЕЛИРОВАНИЕ ДАННЫХ С ПОМОЩЬЮ ДИАГРАММЫ КЛАССОВ UML**

**2.1 ВВЕДЕНИЕ**

Диаграмма классов UML (Unified Modeling Language) является одной из основных диаграмм, используемых в разработке ПО, для моделирования структуры и отношений классов в системе. Она представляет собой графическое представление классов, их атрибутов, методов и связей между ними.

Моделирование данных с помощью диаграммы классов UML помогает разработчикам лучше понять структуру и организацию данных в системе, а также способствует более эффективной разработке и поддержке ПО. Основные элементы диаграммы классов UML включают классы, операции и атрибуты.

**2.2 ОСНОВНЫЕ ПОНЯТИЯ ДИАГРАММ КЛАССОВ UML**

Язык UML основан на сущностях, отношениях и диаграммах.

Сущность - это объект системы, представленный в абстрактной форме.

Отношение - это способ связи между сущностями.

Диаграмма - это визуальное представление сущностей с их отношениями. Она создается с использованием графических элементов языка.

Сущности и отношения на диаграммах изображаются в графической форме и описываются текстовыми надписями и комментариями.

Диаграмма классов - это важный тип диаграммы UML, который показывает набор классов и их связи. Она также может содержать комментарии и ограничения. Ограничения могут быть написаны на естественном языке или на языке объектных ограничений OCL.

**2.2.1 КЛАССЫ**

Класс в UML - это структура, представляющая абстрактное понятие и являющаяся эквивалентом класса в объектно-ориентированном языке программирования. Класс создается разработчиками для создания объектов с одинаковой структурой.

Класс в UML - это именованное описание набора объектов с общими атрибутами, операциями, связями и семантикой. Он изображается графически в виде прямоугольника. У каждого класса должно быть уникальное имя. Рекомендуется использовать короткие и осмысленные имена классов, каждое из которых начинается с заглавной буквы.

**2.2.2 АТРИБУТЫ**

Атрибут в классе представляет собой свойство класса, определяющее диапазон значений, которые могут принимать объекты этого класса. Класс может иметь любое количество атрибутов, включая отсутствие атрибутов. Атрибут является абстракцией состояния объекта и представляет общее свойство для всех объектов данного класса. Каждый атрибут объекта класса должен иметь значение.

Имена атрибутов представляются внутри класса, под именем класса. В UML нет ограничений на способ именования атрибутов (имя может быть произвольной текстовой строкой), но на практике рекомендуется использовать короткие и осмысленные прилагательные и существительные, отражающие смысл атрибута класса. Написание первого слова в имени атрибута с заглавной буквы, а последующих слов - с заглавной буквы также является обычной практикой. При описании класса указываются соответствующие атрибуты.

**2.2.3 ОПЕРАЦИИ**

Операцией класса называется именованная услуга, которую можно запросить у любого объекта этого класса. Операция – это абстракция того, что можно делать с объектом. Класс может содержать любое число операций. Набор операций класса является общим для всех объектов данного класса. Операции класса определяются в разделе, расположенном ниже раздела с атрибутами. Для именования операций рекомендуется использовать глагольные формы, соответствующие ожидаемому поведению объектов данного класса.

**2.3 ЗАВИСИМОСТИ, ОБОБЩЕНИЯ И АССОЦИАЦИИ**

Ассоциация (association) представляет отношение между двумя классами, где один класс использует или включает другой класс. Например, в классе "Студент" может существовать ассоциация с классом "Курс", где студенты записываются на определенные курсы.

Обобщение (generalization) представляет отношение типа "является" между двумя классами, где один класс является более общим (родительским), а другой класс является более специфичным (подклассом). Например, класс "Фрукт" может быть родительским классом для классов "Яблоко" и "Груша", где яблоки и груши являются конкретными типами фруктов.

Зависимость (dependency) представляет отношение, при котором один класс зависит от другого класса. Это может быть отношение временной или логической зависимости, где изменения в одном классе могут повлиять на другой класс. Например, класс "Заказ" может зависеть от класса "Товар", потому что изменения в классе "Товар" могут повлиять на процесс оформления заказа.

**2.3.1 ЗАВИСИМОСТИ**

В диаграммах классов зависимости между классами обычно используются для отражения параметров операций одного класса, которые зависят от объектов другого класса. Когда интерфейс второго класса изменяется, это может повлиять на поведение объектов первого класса.

Зависимость обозначается прерывистой линией со стрелкой, указывающей на класс, от которого есть зависимость. У зависимости может быть собственное имя, но в большинстве случаев это не требуется.

Однако при проектировании реляционных баз данных зависимости обычно не используются, поскольку реляционные БД не поддерживают прямую связь между классами. Вместо этого используются другие конструкции, такие как внешние ключи и связи между таблицами.

**2.3.2 ОБОБЩЕНИЯ И МЕХАНИЗМ НАСЛЕДОВАНИЯ КЛАССОВ**

Обобщение (generalization) представляет отношение между более общим классом (суперклассом или родителем) и более специализированным подклассом (или потомком). Подкласс наследует все атрибуты и операции от суперкласса, но может иметь также и свои собственные атрибуты, и операции.

Обобщение может быть представлено как связь "является", означая, что подкласс является частным случаем суперкласса. Объекты подкласса могут использоваться везде, где используются объекты суперкласса. Это свойство называется полиморфизмом по включению, поскольку объекты потомка могут быть включены в множество объектов суперкласса.

Графически обобщение отображается сплошной линией с большой незакрашенной стрелкой, указывающей на суперкласс.

**2.3.3 АССОЦИАЦИИ: РОЛИ, КРАТНОСТЬ, АГРЕГАЦИЯ, КОМПОЗИЦИЯ**

Агрегация - это тип ассоциации, который отображает отношение "часть-целое" между двумя классами. Класс "целое" является более высокого уровня концепции, чем класс "часть". Например, класс "Автомобиль" может быть агрегирован из классов "Колесо", "Двигатель" и "Кузов".

Композиция - это особый вид агрегации, где объекты класса "целое" управляют жизненным циклом объектов класса "часть". Если объект класса "целое" уничтожается, то и объекты класса "часть" также будут уничтожены. Например, объект класса "Компьютер" может включать объекты классов "Процессор", "Память" и "Жесткий диск", и если компьютер будет выключен, то все его компоненты будут выключены вместе с ним.

Роль - это роль, которую каждый объект играет в ассоциации. Например, в ассоциации "Студент изучает Предмет" студент может играть роль "студент", а предмет - роль "предмет".

Кратность - это характеристика, указывающая, сколько объектов класса с определенной ролью может быть связано с объектом класса через ассоциацию. Например, в ассоциации "Студент изучает Предмет" кратность роли "студент" может быть указана как "1..\* ", что означает, что каждый предмет может быть изучаем одним или более студентами.

**2.4 ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ И ЯЗЫК OCL**

Один из самых серьезных и справедливо критикуемых недостатков языка моделирования UML является предоставление только средств визуального представления моделей, т. е. UML – это просто графический редактор диаграмм различного вида. Визуальные средства абстракты и поэтому далеко не всегда способны точно и формально отразить нюансы функционирования проектируемой системы.

**2.4.1 ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА OCL**

Из языка UML в OCL заимствованы, в первую очередь, следующие понятия:

– класс, атрибут, операция;

– объект, понимаемый как экземпляр класса;

– ассоциация;

– тип данных (включая набор предопределенных типов);

– значение как экземпляр типа данных. В дополнение к скалярным типам данных, заимствованным из UML, в OCL предопределены структурные типы, которые являются разновидностями типов коллекций (collection):

– математическое множество (set), неупорядоченная коллекция, не содержащая одинаковых элементов;

– мультимножество (bag), неупорядоченная коллекция, которая может содержать повторяющиеся элементы-дубликаты;

– последовательность (sequence), упорядоченная коллекция, которая может содержать элементы-дубликаты. Язык OCL предназначен, главным образом, для определения ограничений целостности данных, соответствующих модели, которая представлена в терминах диаграммы классов UML.

**2.4.2 ИНВАРИАНТ КЛАССА**

Под инвариантом класса в OCL понимается условие, которому должны удовлетворять все объекты данного класса. Если говорить более точно, инвариант класса – это логическое выражение, вычисление которого должно давать true при создании любого объекта данного класса и сохранять истинное значение в течение всего времени его существования. При определении инварианта требуется указать имя класса и выражение, определяющее инвариант указанного класса.

В спецификации языка эти операции условно разделены на следующие группы:

– операции над значениями предопределенных типов данных;

– операции над объектами;

– операции над множествами;

– операции над мультимножествами;

– операции над последовательностями.

**2.4.3 ОПЕРАЦИИ НАД МНОЖЕСТВАМИ, МУЛЬТИМНОЖЕСТВАМИ И ПОСЛЕДОВАТЕЛЬНОСТЯМИ**

В языке OCL поддерживается широкий набор операций над коллекциями, такими как множества, мультимножества и последовательности. Рассмотрим только те операции, которые используются при моделировании реляционных баз данных.

Операция select используется для обработки множества, мультимножества или последовательности на основе заданного логического выражения. Результатом этой операции является новая коллекция, состоящая из элементов исходной коллекции, для которых логическое выражение является true.

Операция collect также применяется к коллекциям и состоит из трех вариантов: для множества, мультимножества и последовательности. Она применяет заданное выражение к каждому элементу входной коллекции и составляет новую коллекцию, содержащую результаты.

Операции exists, forAll и size применяются к коллекциям и имеют дополнительный параметр - логическое выражение. Операция exists возвращает true, если хотя бы один элемент входной коллекции удовлетворяет логическому выражению. Операция forAll возвращает true, если все элементы входной коллекции удовлетворяют логическому выражению. Операция size возвращает количество элементов в коллекции.

Операции union, intersect и symmetricDifference применяются к двум коллекциям. Они возвращают новую коллекцию, которая является результатом объединения, пересечения или симметричной разности соответствующих элементов двух коллекций. В случае операции union над множеством и мультимножеством, результатом является мультимножество, то есть дубликаты не исключаются. В случае операции union над двумя множествами, из результата исключаются возможные дубликаты.

**2.4.4 ПЛЮСЫ И МИНУСЫ ИСПОЛЬЗОВАНИЯ ЯЗЫКА OCL ПРИ ПРОЕКТИРОВАНИИ РЕЛЯЦИОННЫХ БД**

Плюсы использования языка OCL при проектировании реляционных баз данных:

1. Формальное и однозначное определение ограничений целостности: OCL позволяет определить ограничения в терминах концептуальной схемы базы данных без двусмысленностей, которые могут быть присутствовать в естественных языках.

2. Полезность для сопровождения базы данных: Наличие проектной документации на OCL может быть полезным для сопровождения базы данных, даже если приходится переводить инварианты OCL в ограничения целостности SQL вручную.

Негативные стороны использования языка OCL при проектировании реляционных баз данных:

1. Сложность и неочевидность: OCL является сложным языком, и некоторые его конструкции не всегда являются очевидными.

2. Противоречие с наглядностью и интуитивной ясностью диаграммной части UML: Синтаксис и линейная форма OCL могут быть противоречивыми с наглядностью и интуитивной ясностью диаграмм классов в UML.

3. Ограничения выражаемости: Невозможно однозначно утверждать, что на OCL можно выразить любое ограничение целостности, которое можно выразить с помощью SQL, или наоборот. Трудно доказать или опровергнуть это предположение.