МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ БЕЛОРУССКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА»

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И АНАЛИЗА ДАННЫХ

РЕФЕРАТ

УНИФИЦИРОВАННЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ UML

Петрова Андрея Александрович
студента 2 курса группы 14
специальности «Прикладная информатика»

Научный руководитель:
кандидат физико-математических наук
Сталевская С.Н.

Минск, 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 3
ГЛАВА 1 ПОНЯТИЕ И ОСОБЕННОСТИ ЯЗЫКА UML 4
ГЛАВА 2 КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ UML 6
2.1. Сущности как вид строительных блоков UML 6
2.2. Отношения как вид строительных блоков UML 8
2.3. Диаграммы в UML 9
ЗАКЛЮЧЕНИЕ 11
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13

ВВЕДЕНИЕ

Большинство существующих методов объектно-ориентированного анализа и проектирования (ООАП) включают как язык моделирования, так и описание процесса моделирования. **Язык моделирования** – это нотация (в основном графическая), которая используется методом для описания проектов.

Нотация представляет собой совокупность графических объектов, которые используются в моделях; она является синтаксисом языка моделирования. Например, нотация диаграммы классов определяет, каким образом представляются такие элементы и понятия, как класс, ассоциация и множественность.

Процесс – это описание шагов, которые необходимо выполнить при разработке проекта.

Унифицированный язык моделирования UML (Unified Modeling Language) – это преемник того поколения методов ООАП, которые появились в конце 80-х и начале 90-х гг.

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

ГЛАВА 1
ПОНЯТИЕ И ОСОБЕННОСТИ ЯЗЫКА UML

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО систем. UML призван поддерживать процесс моделирования ПС на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

К середине 90-х годов различными авторами было предложено несколько десятков методов ОО моделирования, каждый из которых использовал свою графическую нотацию. При этом любой их этих методов имел свои сильные стороны, но не позволял построить достаточно полную модель ПС, показать ее «со всех сторон», то есть, все необходимые проекции (См. статью 1). К тому же отсутствие стандарта ОО моделирования затрудняло для разработчиков выбор наиболее подходящего метода, что препятствовало широкому распространению ОО подхода к разработке ПС.

По запросу **Object Management Group (OMG)** – организации, ответственной за принятие стандартов в области объектных технологий и баз данных назревшая проблема унификации и стандартизации была решена авторами трех наиболее популярных ОО методов – Г.Бучем, Д.Рамбо и А.Джекобсоном, которые объединенными усилиями создали версию UML 1.1, утвержденную OMG в 1997 году в качестве стандарта.

Любой язык состоит из словаря и правил комбинирования слов для получения осмысленных конструкций. Так, в частности, устроены языки программирования, таковым является и UML. Отличительной его особенностью является то, что словарь языка образуют графические элементы. Каждому графическому символу соответствует конкретная семантика, поэтому модель, созданная одним разработчиком, может однозначно быть понята другим, а также программным средством, интерпретирующим UML. Отсюда, в частности, следует, что модель ПС, представленная на UML, может автоматически быть переведена на ОО язык программирования (такой, как Java, C++, VisualBasic), то есть, при наличии хорошего инструментального средства визуального моделирования, поддерживающего UML, построив модель, мы получим и заготовку программного кода, соответствующего этой модели.

Следует подчеркнуть, что UML – это именно язык, а не метод. Он объясняет, из каких элементов создавать модели и как их читать, но ничего не говорит о том, какие модели и в каких случаях следует разрабатывать. Чтобы создать метод на базе UML, надо дополнить его описанием процесса разработки ПС.

UML использует графические обозначения для создания модели системы(рис.1). Данный язык был создан для определения, визуализации, проектирования и документирования программных систем, а также его используют для моделирования бизнес-процессов, системного проектирования.

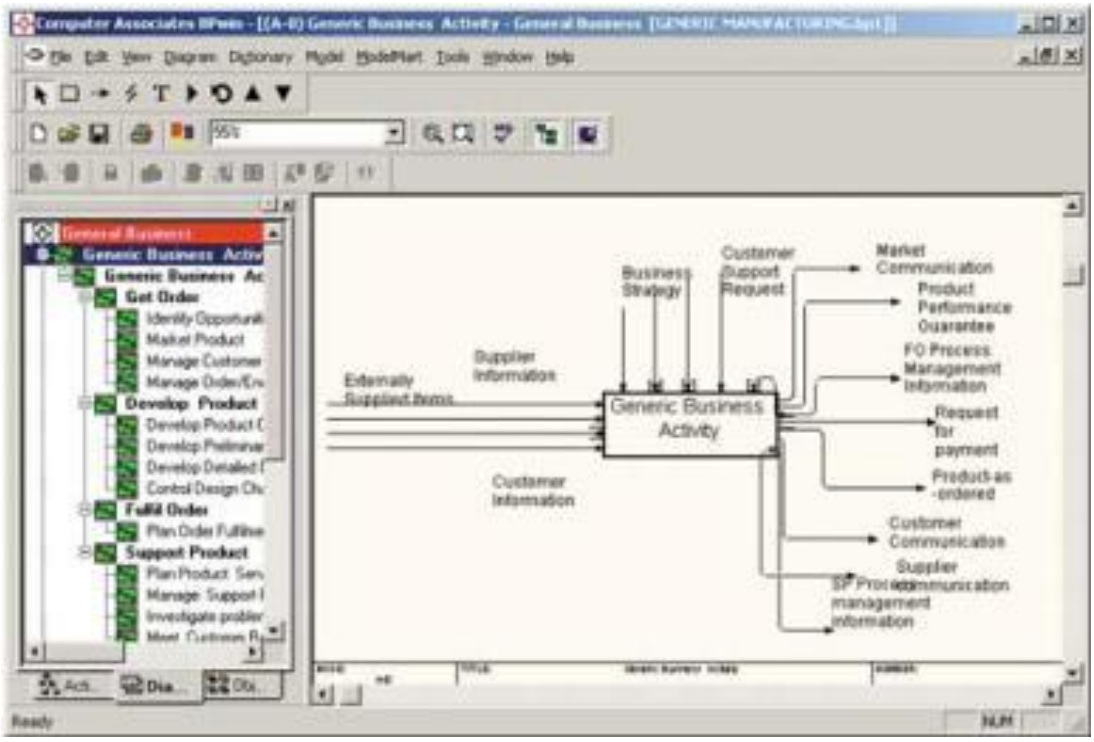

Рисунок 1. Графические обозначения для создания модели системы.

ГЛАВА 2
КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ UML

Для понимания UML необходимо усвоить его концептуальную модель, которая включает в себя три составные части:

– основные строительные блоки языка;

– правила их сочетания;

– некоторые общие для всего языка механизмы.

Словарь языка UML включает три вида строительных блоков:

– сущности;

– отношения;

– диаграммы.

2.1. Сущности как вид строительных блоков UML

Сущности – это абстракции, являющиеся основными элементами моделей.

Имеется четыре типа сущностей:

1. Структурные (класс, интерфейс, компонент, вариант использования, кооперация, узел);
2. Поведенческие (взаимодействие, состояние);
3. Группирующие (пакеты);
4. Аннотационные (комментарии).

Каждый вид сущностей имеет свое графическое представление.

Сущности являются основными объектно-ориентированными блоками языка. С их помощью можно создавать корректные модели.

Структурные сущности – это имена существительные в моделях на языке UML. Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует несколько разновидностей структурных сущностей.

Класс (Class) – это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Класс реализует один или несколько интерфейсов.

Интерфейс (Interface) – это совокупность операций, которые определяют сервис (набор услуг), предоставляемый классом или компонентом. Таким образом, интерфейс описывает видимое извне поведение элемента. Интерфейс может представлять поведение класса или компонента полностью или частично; он определяет только спецификации операций (сигнатуры), но никогда – их реализации.

Кооперация (Collaboration) определяет взаимодействие; она представляет собой совокупность ролей и других элементов, которые, работая совместно, производят некоторый кооперативный эффект, не сводящийся к простой сумме слагаемых. Кооперация, следовательно, имеет как структурный, так и поведенческий аспект. Один и тот же класс может принимать участие в нескольких кооперациях; таким образом, они являются реализацией образцов поведения, формирующих систему.

Прецедент (Use case) – это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (Actor). Прецедент применяется для структурирования поведенческих сущностей модели. Прецеденты реализуются посредством кооперации.

Компонент (Component) – это физическая заменяемая часть системы, которая соответствует некоторому набору интерфейсов и обеспечивает его реализацию. В системе можно встретить различные виды устанавливаемых компонентов, такие как COM+ или Java Beans, а также компоненты, являющиеся артефактами процесса разработки, например файлы исходного кода. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации.

Поведенческие сущности (Behavioral things) являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей.

Взаимодействие (Interaction) – это поведение, суть которого заключается в обмене сообщениями (Messages) между объектами в рамках конкретного контекста для достижения определенной цели. С помощью взаимодействия можно описать как отдельную операцию, так и поведение совокупности объектов. Взаимодействие предполагает ряд других элементов, таких как сообщения, последовательности действий (поведение, инициированное сообщением) и связи (между объектами).

Состояние (State machine) – это алгоритм поведения, определяющий последовательность состояний, через которые объект или взаимодействие проходят на протяжении своего жизненного цикла в ответ на различные события, а также реакции на эти события. С помощью автомата можно описать поведение отдельного класса или кооперации классов. С автоматом связан ряд других элементов: состояния, переходы (из одного состояния в другое), события (сущности, инициирующие переходы) и виды действий (реакция на переход).

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Есть только одна первичная группирующая сущность, а именно пакет.

Пакеты (Packages) представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить структурные, поведенческие и даже другие группирующие сущности. В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки.

Аннотационные сущности – пояснительные части модели UML. Это комментарии для дополнительного описания, разъяснения или замечания к любому элементу модели. Имеется только один базовый тип аннотационных элементов – примечание (Note).

Примечание – это просто символ для изображения комментариев или ограничений, присоединенных к элементу или группе элементов

Этот элемент является основной аннотационной сущностью, которую можно включать в модель UML. Чаще всего примечания используются, чтобы снабдить диаграммы комментариями или ограничениями, которые можно выразить в виде неформального или формального текста. Существуют вариации этого элемента, например требования, где описывают некое желательное поведение с точки зрения внешней по отношению к модели.

2.2. Отношения как вид строительных блоков UML

В языке UML определены четыре типа отношений:

– зависимость;

– ассоциация;

– обобщение;

– реализация.

Эти отношения являются основными связующими строительными блоками в UML и применяются для создания корректных моделей.

Зависимость (Dependency) – это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой.

Ассоциация (Association) – структурное отношение, описывающее совокупность связей; связь – это соединение между объектами. Разновидностью ассоциации является агрегирование (Aggregation) – так называют структурное отношение между целым и его частями

Обобщение (Generalization) – это отношение «специализация/обобщение», при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (родителя или предка). Таким образом, потомок (Child) наследует структуру и поведение своего родителя (Parent).

Реализация (Realization) – это семантическое отношение между классификаторами, при котором один классификатор определяет «контракт», а другой гарантирует его выполнение. Отношения реализации встречаются в двух случаях: во-первых, между интерфейсами и реализующими их классами или компонентами, а во-вторых, между прецедентами и реализующими их кооперациями.

Четыре описанных элемента являются основными типами отношений, которые можно включать в модели UML. Существуют также их вариации, например уточнение (Refinement), трассировка (Trace), включение и расширение (для зависимостей).

2.3. Диаграммы в UML

Диаграммы UML разделяются на две группы: структурные диаграммы и диаграммы поведения. Такая классификация соответствует концепции сложности. Сложность системы зависит как от организации элементов системы (структуры), так и от способа их взаимодействия друг с другом (поведения).

Структурные диаграммы используются для демонстрации статической структуры элементов в системе. Они могут изображать архитектурную организацию системы, ее физические элементы, текущую конфигурацию, а также специфические элементы предметной области. К структурным диаграммам языка UML относятся следующие диаграммы:

- Диаграммы пакетов (package diagram)
- Диаграммы классов (class diagram)
- Диаграммы компонентов (component diagram)
- Диаграммы развертывания (deployment diagram)
- Диаграммы объектов (object diagram)
- Диаграммы композитных структур (composite structure diagram)

Структурные диаграммы часто используются в сочетании с диаграммами поведения для описания определенных аспектов системы. Каждый класс может иметь соответствующую диаграмму конечных автоматов (state machine diagram), описывающую поведение его экземпляров. Аналогично, совместно с диаграммами объектов, представляющих определенный сценарий, диаграммы взаимодействия (interaction diagram) демонстрируют моменты времени или порядок передачи сообщений по мере их вычисления.

Диаграммы поведения. Диаграммы, рассмотренные выше, носят статический характер. Однако события, происходящие в системах программного обеспечения, являются динамическими: объекты создаются и уничтожаются, объекты передают сообщения другим объектам и системам, внешние события активизируют операции над определенными объектами. В объектно-

ориентированном проектировании семантика динамического поведения и способы ее реализации описываются с помощью следующих средств:

- Диаграмма прецедентов использования (use case diagram)
- Диаграмма деятельности (activity diagram)
- Диаграмма конечных автоматов (state machine diagram)
- Диаграммы взаимодействий (interaction diagram)
- Диаграмма последовательностей (sequence diagram)
- Диаграмма коммуникации (communication diagram)
- Диаграмма обзора взаимодействий (interaction overview diagram)
- Диаграмма синхронизации (timing diagram)

ЗАКЛЮЧЕНИЕ

Слово "моделирование", входящее в название UML, имеет множество смысловых оттенков и сложившихся способов употребления. В частности, английские слова modeling и simulation оба переводятся словом "моделирование", хотя означают разные вещи. В первом случае речь идет о составлении модели, которая используется только для описания моделируемого объекта или явления. Во втором случае подразумевается составление модели, которая может быть использована для получения существенной информации о моделируемом объекте или явлении. При этом во втором случае обычно добавляется уточняющее прилагательное: численное моделирование, математическое моделирование и др. UML является языком моделирования в первом смысле, хотя известны некоторые успешные попытки использования UML и во втором смысле.

В отношении разработки программного обеспечения так сложилось, что результаты фаз анализа и проектирования, оформленные средствами определенного языка, принято называть моделью∇. Деятельность по составлению моделей естественно назвать моделированием. Именно в этом смысле UML является языком моделирования.

В этой книге термины "программное обеспечение", "программная система", "программа" и "приложение" используются как синонимы. Авторы понимают, что в действительности между ними существуют определенные различия, однако в контексте книги эти различия не имеют большого значения.

Таким образом, модель UML – это, прежде всего, описание объекта или явления, а также и кое-что другое, а именно все, что авторам UML удалось включить в язык, не нарушая принципа унификации, к изложению которого мы переходим в следующем разделе.

Описывая историю создания UML, его авторы характеризуют эпоху до UML как период "войны методов". Пожалуй, "война" – это слишком сильно сказано, но, действительно, UML является отнюдь не первым языком моделирования. К моменту его появления насчитывались десятки других, различающихся системой обозначений, степенью универсальности, способами применения и т.д. Авторы языков и теоретики программирования препирались между собой, выясняя, чей подход лучше, а разработчики всю эту "войну методов" равнодушно игнорировали, поскольку ни один из методов не дотягивал до уровня индустриального стандарта.

Толчком к изменению ситуации послужили следующие обстоятельства. Во-первых, массовое распространение получил объектно-ориентированный подход к разработке программных систем, в результате чего возникла потребность в соответствующих средствах. Другими словами, появления чего-то подобного UML с нетерпением ждали практики. Во-вторых, три крупнейших специалиста в этой области, авторы наиболее популярных методов, решились объединить усилия именно с целью унификации своих (и не только своих) разработок в соответствии с социальным заказом.

Приложив заслуживающие уважения усилия, авторы UML при поддержке и содействии всей международной программистской общественности смогли свести воедино (унифицировать) большую часть того, что было известно и до них. В результате унификации получилась теоретически изящная и практически полезная вещь – UML.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мартин Фаулер. UML. Основы, 3 е издание. Спб Символ, – 2005 г.

2. Джим Арлоу, Айла Нейштадт. UML 2 и Унифицированный процесс, 2-е издание. Спб Символ, – 2008 г.

3. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. UML. Руководство пользователя. Спб ДМК пресс, – 2000 г.

4. Вендров А.М. Проектирование программного обеспечения ЭИС М.: «Финансы и статистика», – 2003 г.

5. Галицына О.Л., Максимов Н.В. Базы данных М.: «Форум-ИНФРА-М», – 2006 г.