ГАПОУ Самарской области
Тольяттинский машиностроительный колледж

Инструментальные средства разработки программного обеспечения

2015

Программное обеспечение

Под программным обеспечением (ПО) понимается совокупность программных и документальных средств для создания и эксплуатации систем обработки данных средствами вычислительной техники. В самом общем плане программное обеспечение для вычислительной техники может быть разделено (в зависимости от назначения) на *системное*, *инструментальное* и *прикладное* (как и всякая классификация данное деление условно).

Инструментальное программное обеспечение используется для создания программных продуктов в любой области, включая и системные программы.

Общие сведения о CASE-технологиях

Под термином **CASE-средства** понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

Основной целью **CASE-технологии** является разграничение процесса проектирования программных продуктов от процесса кодирования и последующих этапов разработки, максимально автоматизировать процесс разработки.

Методы структурного анализа и проектирования

Для выполнения поставленной цели CASE-технологии используют два принципиально разных подхода к проектированию: *структурный и объектно-ориентированный.*

Структурный подход предполагает декомпозицию (разделение) поставленной задачи на функции, которые необходимо автоматизировать. В свою очередь, функции также разбиваются на подфункции, задачи, процедуры. В результате получается упорядоченная иерархия функций и передаваемой информацией между функциями.

- *Структурный подход* подразумевает использование определенных общепринятых методологий при моделировании различных информационных систем:

  SADT (Structured Analysis and Design Technique);

  DFD (Data Flow Diagrams);

  ERD (Entity-Relationship Diagrams).

Существует три основных типа моделей, используемых при структурном подходе: функциональные, информационные и структурные.

- *Основным инструментом объектно-ориентированного подхода* является язык UML — унифицированный язык моделирования, который предназначен для визуализации и документирования объектно-ориентированных систем с ориентацией их на разработку программного обеспечения. Данный язык включает в себя систему различных диаграмм, на основании которых может быть построено представление о проектируемой системе.

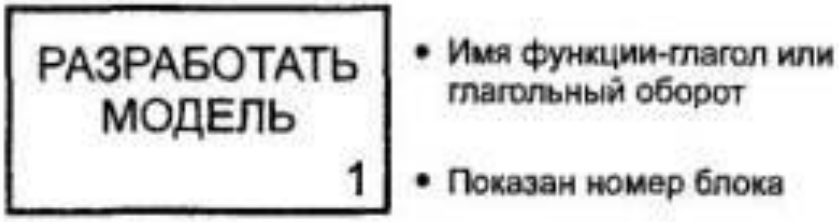
Методология функционального моделирования IDEFO. Общие сведения

IDEF0(ICAM DEFinition language 0) — Function Modeling — методология функционального моделирования. С помощью наглядного графического языка IDEF0 изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков — в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы.

Общие сведения методологии функционального моделирования IDEFO.

Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия.

- Первым из них является понятие **функционального блока (Activity Box)**. Функциональный блок графически изображается в виде прямоугольника и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «производить услуги», а не «производство услуг»).



Общие сведения методологии функционального моделирования IDEFO.

- Вторым «китом» методологии IDEF0 является понятие **интерфейсной дуги (Arrow)**. Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование (Arrow Label). По требованию стандарта, наименование должно быть оборотом существительного.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т. д.) или потоки данных и информации (документы, данные, инструкции и т. д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки. Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь по крайней мере одну управляющую интерфейсную дугу и одну исходящую.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Общие сведения методологии функционального моделирования IDEF0.

- третьим основным понятием стандарта IDEF0 является **декомпозиция (Decomposition)**. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

  Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Общие сведения методологии функционального моделирования IDEF0.

- Последним из понятий IDEF0 является **глоссарий (Glossary)**. Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т. д., которые характеризуют объект, отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Например, для управляющей интерфейсной дуги «распоряжение об оплате» глоссарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т. д. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Синтаксис IDEF0-диаграмм

Диаграммы являются основными рабочими элементами IDEF0-модели. Диаграммы представляют входные-выходные преобразования и указывают правила и средства этих преобразований. Каждая IDEF0-диаграмма содержит блоки (работы) и дуги (линии со стрелками). Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодейст вия и взаимосвязи между ними.

Синтаксис блоков

Функциональные блоки на диаграмме изображаются прямоугольниками .

Блок представляет функцию или активную часть системы.

Каждая сторона блока имеет определенное назначение. Левая сторона предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов и вызовов [1, 11, 12].

Назначение дуг

В IDEF0 различают пять типов дуг: вход (input), управление (control), выход (output), механизм (mechanism), вызов (call) [11]. В основе методологии IDEF0 лежат следующие правила:

1. вход представляет собой входные данные, используемые или преобразуемые функциональным блоком для получения результата (выхода); блок может не иметь ни одной входной дуги (например блок, выполняющий генерацию случайных чисел);
2. выход представляет собой результат работы блока; наличие выходной дуги для каждого блока является обязательным; управление ограничивает или определяет условия выполнения преобразований в блоке;
3. в качестве дуг управления могут использоваться некоторые условия, правила, стратегии, стандарты, которые влияют на выполнение функционального блока; наличие управляющей дуги для каждого блока является обязательным;
4. механизмы показывают, кто, что и как выполняет преобразования в блоке; механизмы определяют ресурсы, непосредственно осуществляющие эти преобразования (например, денежные средства, персонал, оборудование и т.п.); механизмы представляются стрелками, подключенными к нижней стороне блока и направленными вверх к блоку; наличие дуг механизмов для блока не является обязательным;
5. вызовы представляют собой специальный вид дуги и обозначают об- ращение из данной модели или из данной части модели к блоку, входящему в состав другой модели или другой части модели, обеспечивая их связь; с помощью дуги вызова разные модели или разные части одной модели могут совместно использовать один и тот же блок; вызовы не являются компонентом собственно методологии SADT , предназначены для организации коллективной работы над моделью, разделения модели на независимые модели и объединения различных моделей предметной области в одну модель; вызовы представляются стрелка- ми, подключенными к нижней стороне блока и направленными вниз от блока; наличие дуги вызова для блока не является обязательным.

Декомпозиция и её стратегии

Декомпозиция - это процесс создания диаграммы, детализирующей определенный блок и связанные с ним дуги. Результатом ее является описание, которое представляет собой "разламывание" родительского блока на меньшие и более частные функции. Прибавьте к этому еще и тот факт, что слово "анализ" означает разложение на составляющие, и вы получите исходное обоснование термина "структурный анализ". Но декомпозиция - это больше, чем анализ. Она включает также *синтез*. Подлинная декомпозиция заключается в начальном разделении объекта на более мелкие части и последующем соединении их в более детальное описание объекта. Интересно отметить, что модель показывает только результат взаимодействия анализа и синтеза.

Декомпозиция и её стратегии

Декомпозиция - это процесс создания диаграммы, детализирующей определенный блок и связанные с ним дуги. Результатом ее является описание, которое представляет собой "разламывание" родительского блока на меньшие и более частные функции. Прибавьте к этому еще и тот факт, что слово "анализ" означает разложение на составляющие, и вы получите исходное обоснование термина "структурный анализ". Но декомпозиция - это больше, чем анализ. Она включает также *синтез*. Подлинная декомпозиция заключается в начальном разделении объекта на более мелкие части и последующем соединении их в более детальное описание объекта. Интересно отметить, что модель показывает только результат взаимодействия анализа и синтеза.

Что такое IDEF1X?

- *Методология IDEF1X* (IDEF1 Extended) – язык для семантического моделирования данных, основанных на концепции «*сущность-связь*». Является расширением стандарта IDEF1.
- Диаграмма «*сущность-связь*» ***ERD*** (*Entity-Relationship Diagram*) предназначена для разработки модели данных и обеспечивает стандартный способ определения данных и отношений между ними.
- Теоретической базой построения информационной модели является теория баз данных типа «сущность-связь».

Что такое IDEF1X?

- Согласно стандарту , основными составляющими модели IDEF1X являются:

1) люди, предметы, явления, о которых хранится информация (далее – сущности)

2) связи между этими элементами (далее – отношения)

3) характеристики этих элементов (далее – атрибуты)

Понятие отношения

- **Отношения** – связь между двумя и более сущностями. Именование отношения осуществляется с помощью грамматического оборота глагола (имеет, определяет, …).

Таким образом…

- Сущности представляют собой базовый тип информации, хранимый в БД, а отношения показывают, как эти типы данных взаимосвязаны друг с другом.

Правила определения сущности

1. Сущность должна иметь уникальное имя и именоваться существительным в единственном числе.

Пример: *Студент, Кредитная карта, Договор,…*

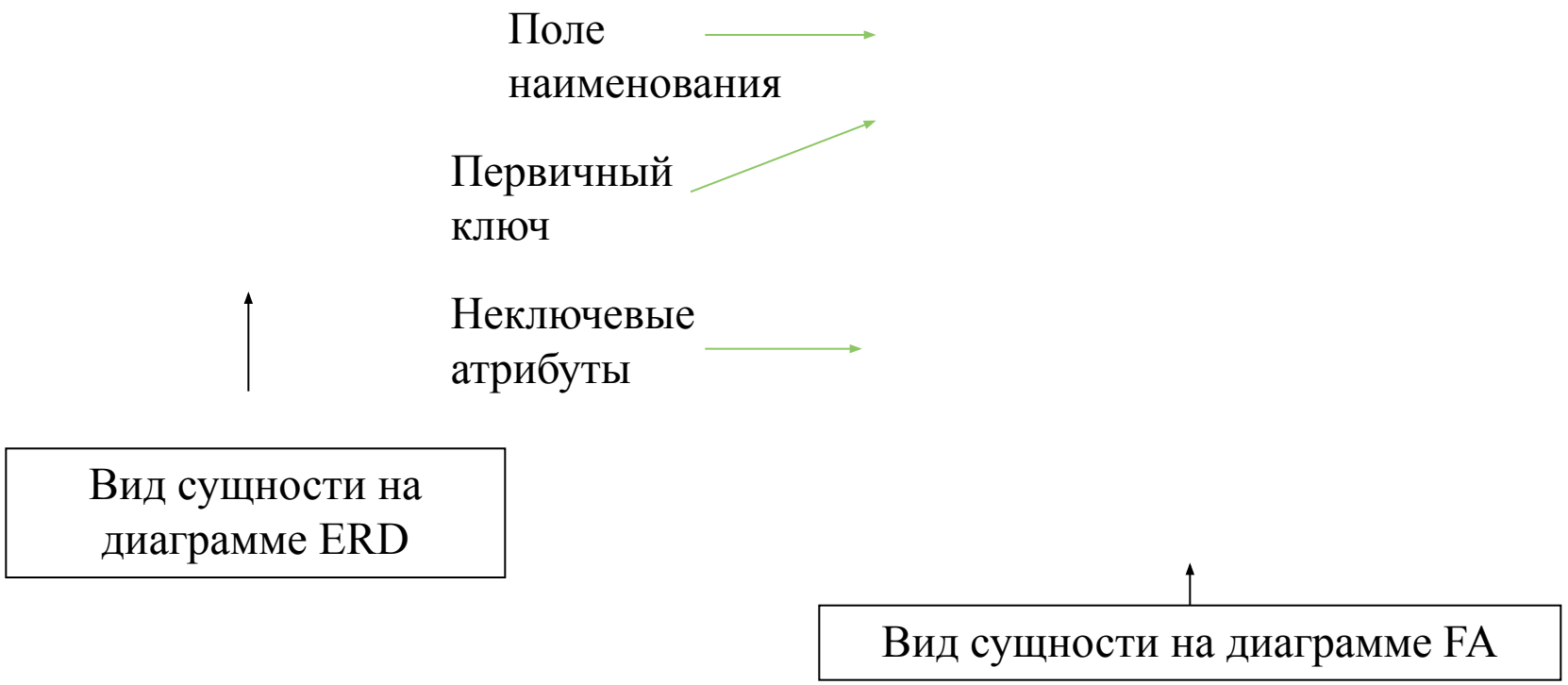
2. Сущность обладает одним или несколькими атрибутами, которые ей либо принадлежат, либо наследуются через отношения.
3. Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый образец сущности и называются ключом (составным ключом).

Правила определения сущности

1. Каждая сущность может обладать любым количеством отношений с другими сущностями.
2. Если внешний ключ целиком используется в составе первичного ключа, то сущность является зависимой от идентификатора.
3. В нотации IDEF1X сущность изображается в виде прямоугольника, в зависимости от уровня представления данных могут быть некоторые различия

Графическое представление сущности

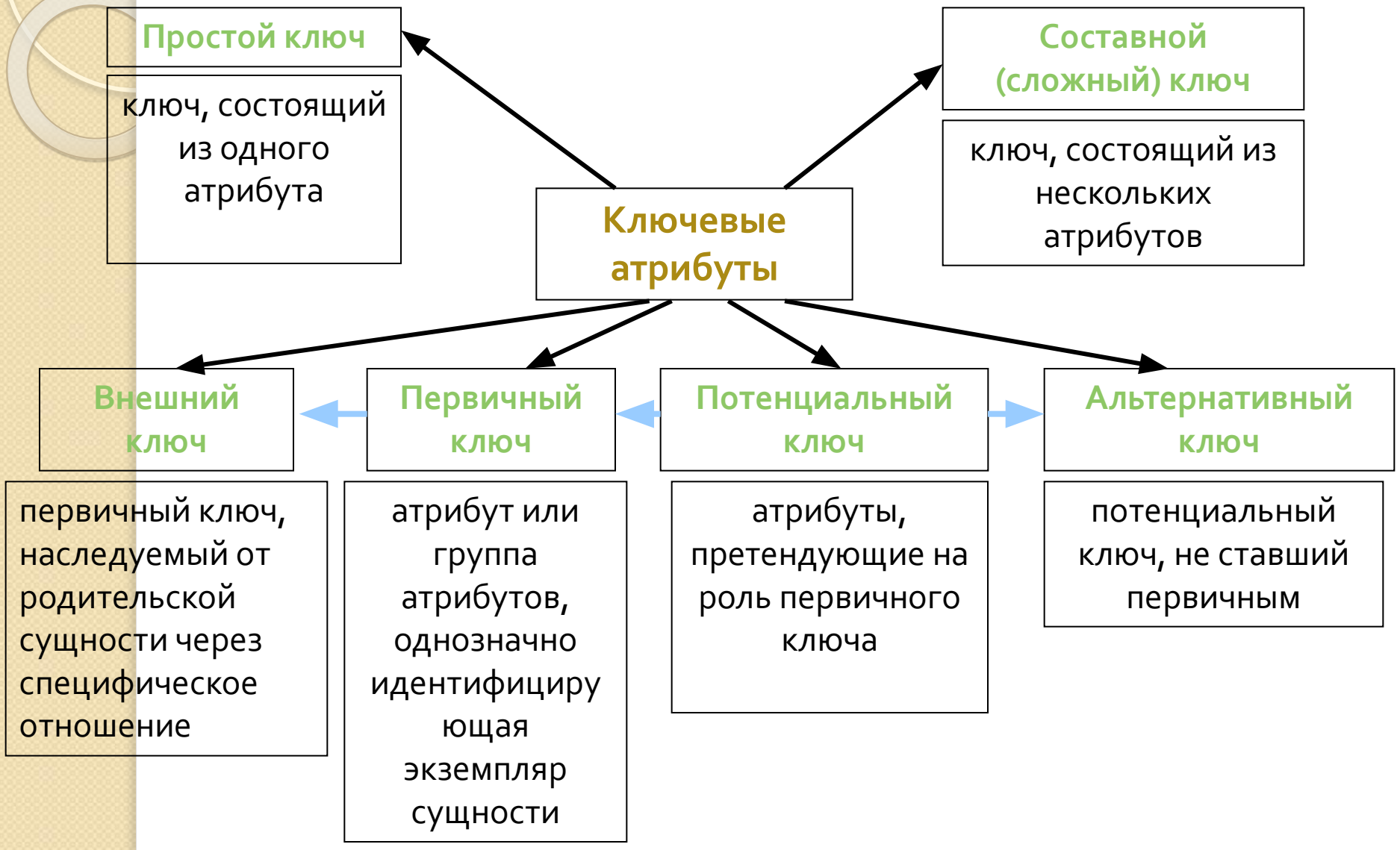
Различают следующие уровни представления сущности: диаграмма «сущность-связь» (*ERD*), модель данных, основанная на ключах (*KB*), полная атрибутивная модель (*FA*)



Правила определения атрибутов

1. Каждый атрибут каждой сущности обладает уникальным именем.
2. Сущность может обладать любым количеством атрибутов.
3. Различают собственные и наследуемые атрибуты. Собственные атрибуты являются уникальными в рамках модели. Наследуемые передаются от сущности-родителя при определении идентифицирующей связи.

Ключевые атрибуты



Примеры ключевых атрибутов

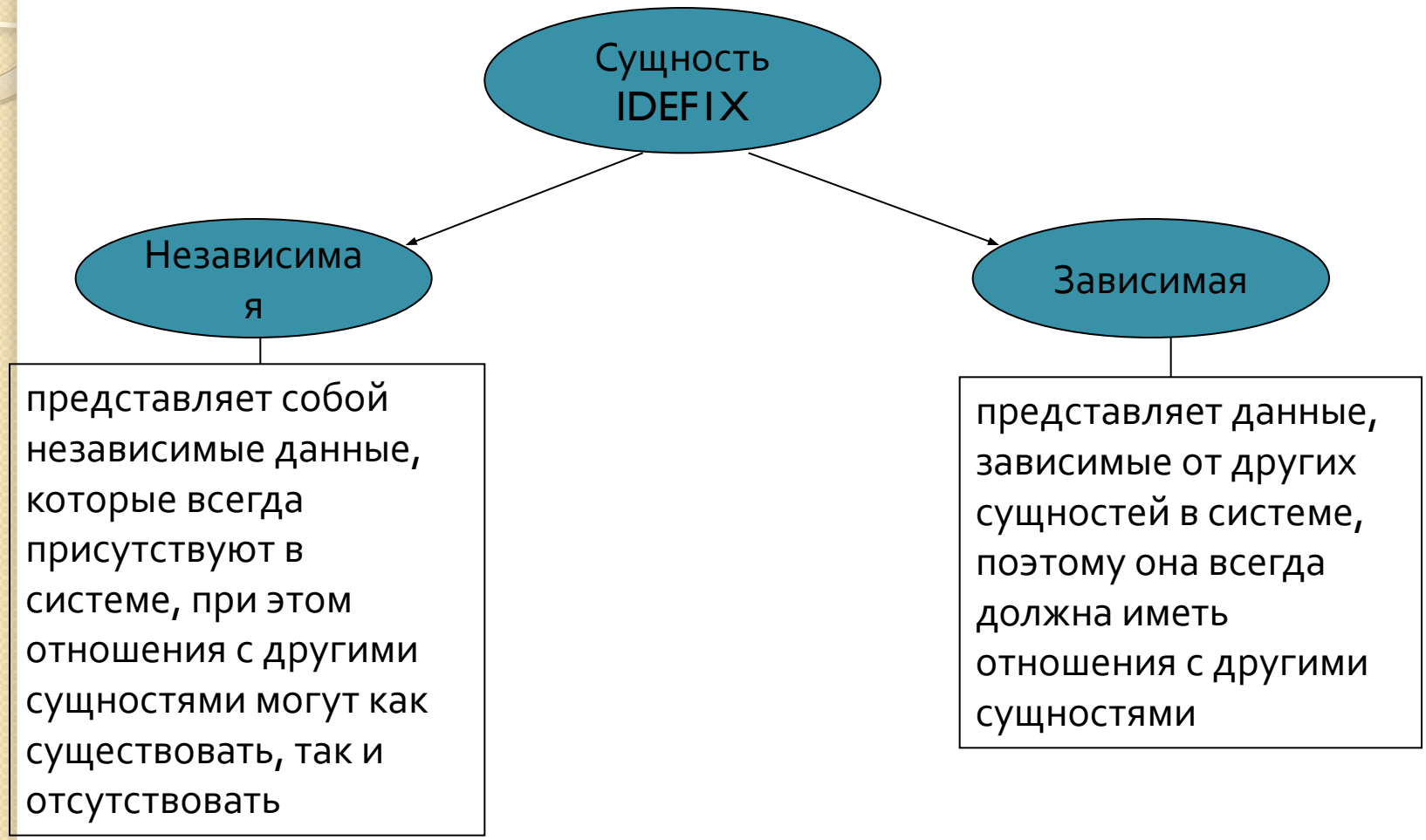
№_зачетнойКнижки – первичный простой ключ;

ФИО+дата_рождения – альтернативный ключ

ФИО+дата_рождения – первичный составной ключ;

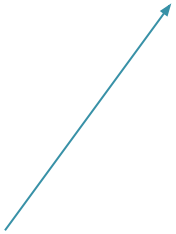
№_зачетнойКнижки – альтернативыний ключ

Типы сущностей в IDEF1X



Типы зависимых сущностей

1. Характеристическая - это зависимая дочерняя сущность, которая связана только с одной родительской сущностью и по смыслу хранит информацию о характеристиках родительской сущности



Характеристическая сущность

2. Категориальная – дочерняя сущность в иерархии наследования

Типы зависимых сущностей

3. Ассоциативная - сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущности

Типы зависимых сущностей

Именующая - частный случай ассоциативной сущности, не имеет собственных атрибутов, только атрибуты родительской сущности

Правила отношений

1) При определении отношения типа *«родитель-потомок»*:

1.1. Экземпляр потомка связан с одним родителем

1.2. Экземпляр-родитель может быть связан с несколькими экземплярами потомков.

2) В идентифицирующем отношении сущность-потомок всегда является зависимой от идентифицирующей сущности.

Виды отношений

а) *идентифицирующее отношение*
Сущность А1 однозначно определяет сущность А2. Ее первичный ключ наследуется в качестве первичного ключа сущностью А2 (внешний ключ)

б) *неидентифицирующее отношение*
Сущность А1 связана с сущностью А2, но однозначно не определяет ее. Первичный ключ сущности А1 наследуется в качестве неключевого атрибута сущности А2

в) *отношение «многие-ко-многим»* (неспецифическое). Сущности А1 и А2 имеют формальную связь, но наследования атрибутов не происходит.

г) *отношение категоризации* (см. далее)

Правила отношений

3) Сущность может быть связана с любым количеством других сущностей как в качестве родителя, так и в качестве потомка.

4) Отношение определяется мощностью. *Мощность* связи служит для обозначения отношения количества экземпляров родительской сущности к числу экземпляров дочерней.

4 типа мощности отношений

а) общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности

б) когда одному экземпляру родительской сущности соответствует 1 или много экземпляров дочерней (0 исключается).

4 типа мощности отношений

в) когда одному экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности.

г) когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

Пример иерархии категорий

Основные правила построения информационной модели

1. Все стрелки (вход, выход, управление, механизм) функциональной модели становятся потенциальными сущностями, а функции, связывающие их, трансформируются в отношения между этими сущностями. Для этого составляется **пул** – список потенциальных сущностей.
2. Число сущностей и связей в IDEF1X-модели считается необозримым, если их количество превышает 25-30. Поэтому далее рассматривается совокупность сущностей и отношений для каждой функции.

Основные правила построения информационной модели

3. Информационная модель функции должна позволять воспроизвести структуру документа и часть информации в нем, а также воспроизвести информацию порождаемого документа.
4. Текстовые пояснения заносятся в глоссарий или оформляются гипертекстом.
5. На основании определения типов отношений, анализа функций и дальнейшего изучения предметной области определяются атрибуты.

Построение информационной модели процесса постройки садового домика

1. На основе функциональной модели IDEF0 составим пул – список потенциальных сущностей.

- ***Пул:***

1. Дом
2. Крыша
3. Материалы
4. Проект дома
5. Стены
6. Строители
7. Фундамент
8. Каменщики
9. Плотники
10. Кровельщики
11. Мастера по отделке

Построение информационной модели процесса постройки садового домика

2. Определим сущности

Построение информационной модели процесса постройки садового домика

3. З
у

Основные понятия и назначение языка UML

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

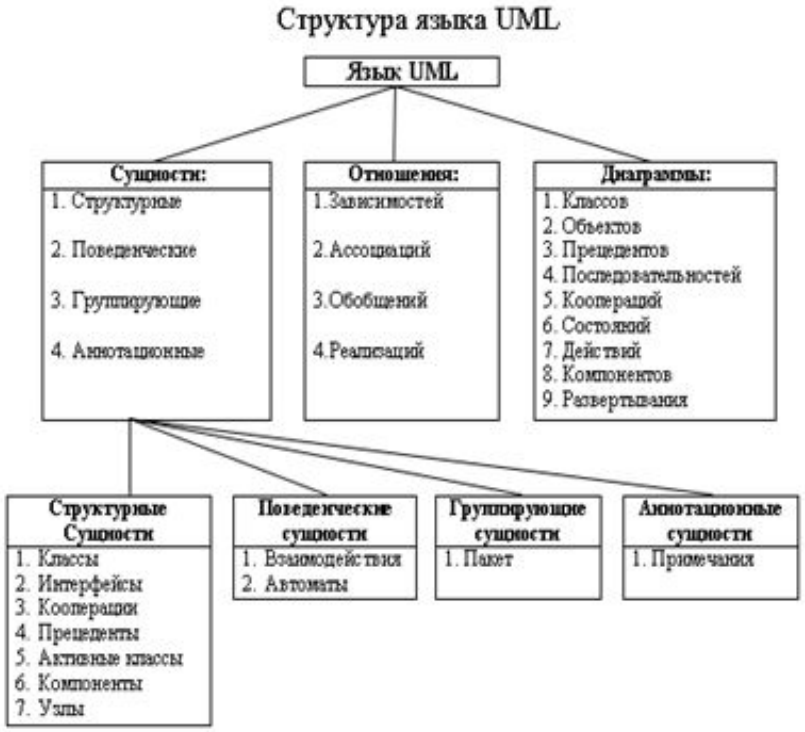
Основное назначение языка UML - *визуальное моделирование и документирование моделей сложных систем самого различного целевого назначения*.

Общие сведения о пакетах в языке UML

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): • диаграммы взаимодействия (interaction diagrams): ♦ диаграммы последовательности (sequence diagrams) и ♦ кооперативные диаграммы (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами; • диаграммы состояний (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое; • диаграммы деятельностей (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей; – диаграммы реализации (implementation diagrams): • диаграммы компонентов (component diagrams) – для моделирования иерархии компонентов (подсистем) системы; • диаграммы размещения (deployment diagrams) – для моделирования физической архитектуры системы.

Общие сведения о пакетах в языке UML

- Язык UML имеет сложную иерархическую структуру, показанную на рисунке.
- Как видно из рисунка, первый иерархический уровень языка UML составляют сущности, отношения между сущностями и наглядные диаграммы. Рассмотрим последовательно эти три основные понятия языка UML.



Диаграммы для моделирования:

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):
- *диаграммы взаимодействия* (interaction diagrams):
- *диаграммы последовательности* (sequence diagrams) и *кооперативные диаграммы* (collaboration diagrams) – для моделирования процесса обмена сообщениями между объектами;
- *диаграммы состояний* (statechart diagrams) – для моделирования поведения объектов системы при переходе из одного состояния в другое;
- *диаграммы деятельностей* (activity diagrams) – для моделирования поведения системы в рамках различных вариантов использования, или моделирования деятельностей;
- *диаграммы реализации* (implementation diagrams):
- *диаграммы компонентов* (component diagrams) – для моделирования иерархии компонентов (подсистем) системы;
- *диаграммы размещения* (deployment diagrams) – для моделирования физической архитектуры системы.

Набор диаграмм для моделирования в UML

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

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

Набор диаграмм для моделирования в UML

Диаграммы взаимодействия . Диаграммы взаимодействия (interaction diagrams) описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой. Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций. Информационное (informative) сообщение – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния. 14 Сообщение-запрос (interrogative) – это сообщение, запрашивающее выдачу некоторой информации об объекте-получателе. Императивное (imperative) сообщение – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий. Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams).

Набор диаграмм для моделирования в UML

Диаграммы последовательности. Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. Например, вариант использования «Снять деньги» предусматривает несколько возможных последовательностей, такие как снятие денег, попытка снять деньги, не имея их достаточного количества на счету, попытка снять деньги по неправильному идентификационному номеру и некоторые другие.

Набор диаграмм для моделирования в UML

Диаграммы классов. Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.